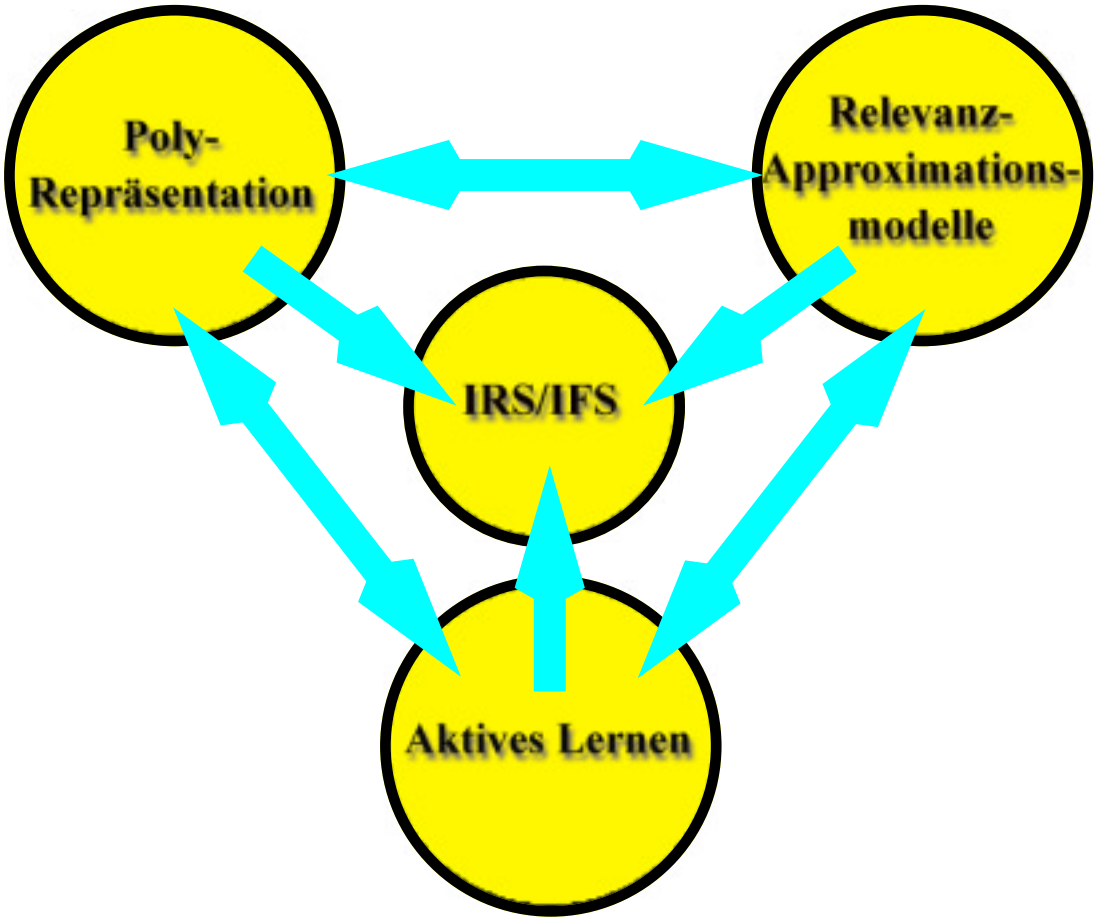
Polyrepräsentation, Relevanz-Approximation und aktives Lernen im Vektorraummodell des Information-Retrievals



von

Günter Bachelier

Dissertation

zur Erlangung des Grades eines Doktors der Philosophie der Philosophischen Fakultäten der Universität des Saarlandes

Juni 2001

Tag der Promotion: 31.01.2002
Dekan der Philosophischen Fakultät III: Herr Univ.-Prof. Dr. Ernst Löffler
Erstberichterstatter: Herr Univ.-Prof. Dr. Harald H. Zimmermann
Zweitberichterstatter: Herr Univ.-Prof. Dr. Werner Tack

Vorwort

Konzipiert wurde diese Arbeit als elektronische Publikation in Form einer pdf-Datei mit einer Vielzahl interner und externer Links, sodass einige kurze Anwendungshinweise vorangestellt werden sollen.

Blaue, nicht unterstrichene Textbereiche repräsentieren Hyperlinks innerhalb der Arbeit, während blau unterstrichene Textbereiche auf externe Quellen (URLs) verweisen.

Eine größere Anzahl von pdf-Dateien, die im Literaturverzeichnis angegeben sind, können über die ACM Digital Library geladen werden. Der Zugang zur ACM Digital Library erfordert jedoch die Mitgliedschaft bei ACM, sowie eine Online-Anmeldung unter http://www.acm.org/dl/.

Einige Dateien, die im Literaturverzeichnis angegeben sind, besitzen den Typ XXX.djvu. Um entsprechende Dateien mit einem Browser zu betrachten, wird das Browser-Plugin “DjVuBrowser” benötigt, welches unter http://www.djvuzone.org/ geladen werden kann (Stand Mai 2001).

Die URL-Quellen beziehen sich bis auf die Angaben des Typs XXX.djvu auf den Stand April/Mai 2001.

Danken möchte ich Prof. Dr. Harald H. Zimmermann, dass ich die dargestellten Themenbereiche frei auswählen und kreativ bearbeiten konnte, sowie David Cohn für die Diskussion bezüglich einiger meiner Ideen zum aktiven Lernen (Cohn (2000[75])).

Günter Bachelier M. A.

Für meine Mutter,
ohne deren Unterstützung dies in der vorliegenden Form
nicht möglich gewesen wäre.

Inhalt

1) Einleitung und Überblick

1.1) Information-Retrieval-Systeme als Spezialfall von Informationssystemen

Informationssysteme (IS) können durch einen 5- bzw. 7-Tupel bechrieben werden (Panyr (1986a: 22[247])):

$$IS = (A, W, Q, I, E). \text{ bzw. } IS = (A, W, Q, I, E, U, D). \quad (1)$$

- A: Inputfunktion (Erschließungsfunktion, Lernfunktion) zum Aufbau der internen Repräsentationen.
- W: Interne Repräsentationen (Wissensbasis).
- Q: Inputmenge als Menge aller zugelassenen Inputkonfigurationen (Problemformulierung, Suchfrage)
- I: Outputfunktion (Inferenzfunktion, Retrievalfunktion und Relevanz-Feedbackfunktion).
- E: Outputmenge als Menge aller möglichen Outputkonfigurationen (Problemlösung, Systemvorschlag).
- U: Updatefunktion der internen Repräsentationen.
- D: Dialogkomponente.

Information-Retrieval-Systeme (IRS) können als Spezialfall eines IS beschrieben werden, indem die einzelnen Komponenten des Tupels (A, W, Q, I, E) spezifiziert werden, was im Kontext des Standard-Retrieval-Prozesses in einem Vektorraummodell geschehen soll. Gegeben ist zu einem Zeitpunkt t eine Dokumentmenge D^t, die durch eine Dokument-Indexierungsfunktion $A_{IR(D)}$ auf eine Dokumentvektorenmenge DVM^t abgebildet wurde. Die einzelnen Dokumentvektoren x_i sind Element eines metrischen, n^t-dimensionalen Dokumentvektorraumes DVR, mit n^t als der Anzahl der Merkmale (Features), auf der die Indexierung basiert und die in der Menge F^t zusammengefasst werden. Der Dokumentvektorraum wird allgemein als Teilraum von $R^{n(t)}$ beschrieben, z.B. durch $[0, 1]^{n(t)}$. Die Query-Indexierungsfunktion $A_{IR(Q)}$ wird vereinfachend definiert als Abbildung aus der Menge $Q(\Theta)$ der möglichen bzw. zugelassenen Queries über einem endlichen Alphabet Θ, in DVR. Es folgt die Anwendung der Retrieval-Funktion, die abhängig ist von der momentanen Dokumentvektorenmenge DVM^t, dem Queryvektor q_i^t und dem metrischen Dokumentvektorraum DVR mit seinen definierenden Eigenschaften, wobei hier ausschließlich die Metrik d_{DVR} betrachtet wird. Sei Γ_{DVR} die Menge aller Metriken, die in einem Dokumentvektorraum DVR angewendet werden können, ohne dass hier auf die Definition der Metrik eingegangen werden soll (siehe Abschnitt 3.6.3)). Die Retrieval-Funktion kann somit spezifiziert werden als eine Abbildung der Potenzmenge $P^{DVM(t)}$ der Dokumentvektorenmenge DVM^t, dem DVR und Γ_{DVR} auf $P^{DVM(t)}$, indem das Tripel aus DVM^t, dem Queryvektor q_i^t und eine Metrik d_{DVR} auf die query-abhängige Ergebnis-Dokumentvektorenmenge DVM_i^t abgebildet wird. D.h. die Retrieval-Funktion besitzt die allgemeine Form $ret(DVM^t, q_i^t, d_{DVR})$ bzw. $ret(DVM^t, q_i^t, d_{DVR}, \varepsilon)$, wenn eine einfache Best-Match-Retrievalstrategie betrachtet wird, bei der alle Dokumentvektoren aus DVM^t selektiert werden, deren Abstand von q_i^t kleiner-gleich einer Distanzschwelle $\varepsilon \in R^+$ ist. Der letzte Schritt besteht in der Erzeugung der Dokumentmenge D_i^t, die zu der Ergebnismenge DVM_i^t korrespondiert. Vereinfachend wurde auf die Beschreibung einer Ranking-Funktion verzichtet, die aus DVM_i^t eine geordnete Liste von Dokumentvektoren erzeugt. Wird diese Darstellung des Retrievalprozesses als Grundlage verwendet, so ergibt sich die grundlegende Beschreibung eines IRS ohne eine Relevanz-Feedbackfunktion als Element von I durch das folgende Tupel:

$$IRS = ((A_{IR(D)}, A_{IR(Q)}), (D^t, DVM^t, F^t, N_{DV}{}^t), Q(\Theta), ret(DVM^t, q_i^t, d_{DVR}), \{P^{DVM(t)}, P^{D(t)}\}). \quad (2)$$

- $A = (A_{IR(D)}, A_{IR(Q)})$:	Dokument-Indexierungsfunktion, Query-Indexierungsfunktion.
- $W = (D^t, DVM^t, F^t, (N_{DV}{}^t))$:	Dokumentmenge, Dokumentvektorenmenge, Merkmalsmenge, (Klassifikation $N_{DV}{}^t$), (Wörterbücher, Thesauri u.ä. bleiben hier unberücksichtigt)
- $Q = Q(\Theta)$:	Menge aller Zeichensequenzen über einem endlichen Alphabet als Menge aller möglichen bzw. zugelassenen Queries.
- $I = ret(DVM^t, q_i, d_{DVR})$:	Retrievalfunktion als Funktion der Dokumentvektorenmenge, eines Queryvektors und der Metrik des Dokumentvektorraums DVR.
- $E \in \{P^{DVM(t)}, P^{D(t)}\}$:	Menge aller möglichen System-Outputs als Potenzmenge der Dokumentvektorenmenge oder Potenzmenge der Dokumentmenge.

1.2) Problemkomplexität des Information Retrievals

Das Ziel des Information Retrievals (IR, (Rijsbergen (1979[279]), Salton & McGill (1987[294]), Grossman & Frieder (1999[154])) ist die Präsentation einer Teilmenge oder geordneten Liste (Ranking) von Dokumenten aus einer Gesamt-Dokumentmenge als Output auf einen Input in Form einer Anfrage (Query) durch einen Nutzer (Agenten). Es wird angenommen, dass die Anfrage ein spezielles Informationsbedürfnis des Agenten repräsentiert, sodass die Ergebnis-Dokumentmenge zur Befriedigung dieses Informationsbedürfnisses beitragen soll. Die internen Selektionsoperationen des Information Retrieval Systems (IRS), welche eine Ergebnis-Dokumentmenge erzeugen, sollen derart sein, dass nach der Integration des Inhaltes der nachgewiesenen Dokumente in das kognitive System des Agenten, das Informationsbedürfnis gegenüber dem Zustand vor der Integration kleiner wird bzw. sich niveliert. Eine alternative Sicht geht nicht von der Verringerung des Informationsbedürfnisses aus, sondern allgemein von dessen Transformation in ein neues Informationsbedürfnis, d.h. der Constraint eines monoton fallenden Maßes an Informationsbedürfnis wird aufgegeben. Dadurch wird die Situation modellierbar, dass ein tiefer gehendes Verständnis eines Problemes andere oder sogar mehr Fragen aufwirft, sodass ein anderes bzw. größeres Informationsbedürfnis aus der Wissenserweiterung des Agenten durch Integration von Informationen aus nachgewiesenen Dokumenten folgen kann (siehe auch Swanson (1987[328])).

Das Ziel der IR-Forschung ist die Suche nach adäquaten und implementierbaren Modellen des IR-Prozesses, wobei Adäquatheit durch unterschiedliche Performancemaße wie z.B. Recall und Precision operationalisiert wird. In den letzten 40 Jahren IR-Forschung hat sich das Problem der Performanceverbesserung bezüglich der verwendeten Maßen als ausgesprochen hartnäckig dargestellt, was sich im Kontext des IR im Internet noch verschärft hat. Es stellt sich die grundsätzliche Frage, warum das IR ein so komplexes Problem ist, wofür die Kombination einer Reihe von Eigenschaften verantwortlich ist, die im Nachfolgenden skizziert werden sollen:

1) Hochdimensionale Zusammenhänge.
2) Nicht-lineare und multimodale Zusammenhänge.
3) Dynamische Zusammenhänge (nicht-stationäre Funktionen)
4) Quellen von Unsicherheit (uncertainty) und Vagheit (fuzzyness).
5) Diversität der Agenten und ihrer Ziele
6) Mehrziel-Anforderungen.

1.2.1) Hochdimensionale Zusammenhänge

Natürlichsprachliche Texte und insbesondere Fachtexte zeichnen sich durch eine große Anzahl darin enthaltener Terme aus, sodass zur Repräsentation dieser Texte ein entsprechend hochdimensionaler Parameterraum (Dokumentvektorenraum) notwendig ist. Dies muss kombiniert betrachtet werden mit einer großen Anzahl von Dokumenten, was im Fall vektorraum-basierter IR in einer hochdimensionalen Term-Dokument-Matrix mündet.

Der Versuch, durch unterschiedliche Verfahren eine Dimensionsreduktion dieser Räume zu erreichen, wurde dementsprechend in der IR-Forschung früh unternommen, wobei die Clusterung von Dokumenten und Termen im Zentrum der Bemühungen stand (Sparck Jones (1971[319]), Crouch (1972[84]), Robertson (1977a[283], b[284], c[285], 1979[286]), Salton (1971[293]), Panyr (1986a[247]), Bachelier (1995[14])). Neuere Verfahren versuchen mit einer Hauptkomponenten-Analyse (Principal Component Analysis (PCA); Oja (1993[240]), Diamantaras & Kung (1996[94]), Lee (1998[200])) eine Dimensionsreduktion zu erreichen wie das Latent Semantic Indexing (LSI; Dumais et al. (1988[100]), Furnas et al. (1988[136]), Bartell et al. (1992[27]), Soboroff et al. (1998[315]), Hofmann (1999[165]), Ando (2000[11])). Hierzu zählen auch konnektionistische Verfahren, die nicht-lineare Faktorenanalysen durch Self-Organizing-Maps (SOMs; Ritter et al. (1991[282]), Kohonen (1989[184], 1995[185]), Bachelier (1998a[15])) approximieren und dies auf den Fall des IR abstimmen (Scholtes (1993[301]), Bachelier (1995[14])).

In der Literatur zum Machine Learning und zur Optimierung ist die Problematik hochdimensionaler Zusammenhänge unter dem Begriff curse-of-dimensionality bekannt, der von Bellman (1967[41]) eingeführt wurde (siehe z.B. Venkatesh et al. (1992[348]), Warwick & Kárny (1997[357])). Bezeichnet wird damit das exponentielle Wachstum von Regionen eines Suchraumes bei einem linearen Wachstum der Dimensionsanzahl, womit der Suchaufwand in dem entsprechenden Raum bei deterministischen Verfahren ebenfalls exponentiell steigt. Als Lösungsansatz gegen den curse-of-dimensionality können Verfahren zur Randomization (siehe z.B. Traub et al. (1988[338]), Wozniakowski (1996a[372], b[373])) eingesetzt werden, wie z.B. Monte Carlo Verfahren (Fishman (1995[114])), die unabhängig von der Dimensionalität des zugrunde liegenden Raumes arbeiten. Erkauft wird diese Eigenschaft damit, dass die gefundenen Lösungen nur mit einer bestimmten Wahrscheinlichkeit in einem definierten Lösungsintervall liegen, d.h. dass die Lösung nur für den sogenannten average-case gelten (siehe auch Abschnitt 5.2.3)).

1.2.2) Nicht-lineare und multimodale Zusammenhänge

Indexierungs- und Retrievalfunktionen in vektorraumbasierten Modellen verwenden nicht-lineare, reelle Gewichtungen von Merkmalen und Dokumenten. Dies ergibt sich daraus, dass es sinnvoll ist, Gewichtungen auf ein reelles Intervall wie [0, 1] zu normieren, da die Interpretierbarkeit beliebig großer oder kleiner Gewichte verloren geht. Das gleiche gilt für Schätzungen der Relevanz durch ein IRS, (siehe Kapitel 4)) wobei reelle Relevanzschätzungen als nicht-lineare, multimodale Funktion der Distanzen von Dokumentvektoren ein Schwerpunkt der vorliegenden Arbeit bildet.

Unimodale Funktionen sind Funktionen mit genau einem Extremwert. Eine unimodale Relevanzfunktion würde bedeuten, dass an genau einem Punkt im Dokumentvektorenraum ein Relevanzmaximum existiert, wobei das Dokument mit dem am nächsten liegenden Dokumentvektor das relevanteste Dokument ist. Notwendig muss von einer monoton fallenden Relevanzfunktion um das Maximum einer unimodalen Funktion ausgegangen werden, d.h. je weiter ein Dokumentvektor vom Relevanzmaximum entfernt liegt, desto geringer wird dessen Relevanz bewertet. Die Modellierung des Retrievals mit einem Queryvektor und einer unimodalen Funktion ist die Standardsituation in den vektorraumbasierten Modellen, d.h. das Relevanzmaximum wird an der Stelle des Queryvektors bzw. des am nächsten liegenden Dokumentvektors angenommen, und allen anderen Dokumentvektoren wird explizit oder implizit ein Relevanzwert als monoton fallende Funktion der Distanz zu dem Maximum zugeordnet. Diese Vorgehensweise ist stark simplifizierend, da mit einer unimodalen Funktion alle Fälle nicht modellierbar werden, bei denen mehrere Cluster relevanter Dokumentvektoren existieren, zwischen denen nicht-relevante Dokumentvektoren liegen. Diese Fälle lassen sich nur mit multimodalen Funktionen modellieren, bei denen jeder Cluster relevanter Dokumentvektoren durch ein eigenes Maximum der Relevanzfunktion beschrieben werden kann. Diese Argumentation wird in Kapitel 4) verwendet, um die effektive Verwendung multimodaler, nicht-linearer Relevanz-Approximationsmodelle zu begründen.

Demgegenüber verwenden probabilistische Modelle nicht-lineare Verteilungsannahmen, welche die Indexierung und das Retrieval als nicht-lineare Prozesse kennzeichnet, jedoch werden nur unimodale Verteilungen verwendet, die nur durch eine Kombination im Sinne eines „Mixture Models“ (McLachlan & Basford (1987[214])) zu einer multimodalen Funktion aggregiert werden können.

Nicht-lineare und multimodale Zusammenhänge in hochdimensionalen Räumen zu modellieren, ist eine mathematisch anspruchsvolle Aufgabe und erfordert große Rechen- und Speicherressourcen, sodass effektive Verfahren im IR entsprechend aufwendig werden.

1.2.3) Dynamische Zusammenhänge (nicht-stationäre Funktionen)

Die offensichtliche Dynamik in IRS besteht in der Aufnahme neuer Dokumente und deren Integration in die gegebene Repräsentationsstruktur, die sich dadurch lokal bzw. global verändern kann. Insbesondere beim Vorliegen von Verfahren zur Dimensionsreduktion kann die Integration neuer Informationsobjekte gravierende Folgen besitzen, wobei eine dynamische Anpassung einer Clusterstruktur eine vergleichsweise problemlose Reorganisation darstellt (Crouch (1972[84])). Verändert sich die Anzahl der Objekte (Informationsobjekte oder Cluster) in einem Raum eines vektorraum-basierten IRS, so verändert sich die Dimension der Merkmals-Dokument-Matrix, und somit die Dimension des komplementären Raumes über sequentiell und rekursiv abhängige Merkmals- und Dokument-Graphen (siehe Abschnitt 3.8); siehe auch Bachelier (1995[14])). Dies erfordert globale Reorganisations-Operationen oder im Extremfall einen Neuaufbau der Strukturen in dem betreffenden Raum, sodass die Anzahl der dynamischen Anpassungen unter dem Constraint der gewünschten Aktualität minimiert werden sollte.

Durch die Aufnahme neuer Dokumente besteht immer die Chance, dass darin Terme enthalten sind, die bislang dem IRS unbekannt waren. Eine Aufnahme neuer Terme verändert die Dimension des Dokumen-

traumes, mit der Notwendigkeit der Reindexierung aller vorhandenen Dokumente und der Anpassung etwaig vorhandener Strukturierungen wie einer Klassifikation.

Diese dynamischen Prozesse der Reorganisation von Repräsentationsstrukturen machen die Retrieval-Funktion zu einer nicht-stationären Funktion, d.h. der gleiche Input kann zu unterschiedlichen Zeitpunkten einen unterschiedlichen Output erzeugen. Die nicht-stationäre Retrieval-Funktion ist von Ereignissen abhängig, die bezogen auf das IRS als extern zu bezeichnen sind, da das IRS keinen Einfluss auf die Produktion von Dokumenten und den darin enthaltenen Termen besitzt. Der einzige Einfluss besteht in der Entscheidung, ob neue Dokumente und Terme repräsentiert werden sollen. Ein IRS, das diese Entscheidung selbst treffen könnte, wäre autonom im Sinne autonomer Agenten, doch diese Entscheidung wird dem IRS ebenfalls von externen Instanzen in Form der Systembetreiber vorgegeben.

Es existiert ein Trade-of zwischen hochdimensionalen Zusammenhängen und dynamischen Funktionen beim vektorraum-basierten IRS in Bezug auf die Merkmalsanzahl. Werden keine Terme als Merkmale verwendet, sondern Zeichen-n-Gramme (Scholtes (1993[301]), Bachelier (1995[14])), so besitzt der Dokumentvektorenraum a priori die maximale Dimension, die somit nicht mehr verändert werden muss. Sei Θ ein endliches Alphabet mit $|\Theta|$ Zeichen (siehe Kapitel 3)), so besitzt ein Dokumentvektorenraum auf der Basis von Zeichen-n-Grammen eine Dimensionsanzahl von $|\Theta|^n$. D.h. jede Dimension des Dokumentvektorenraumes repräsentiert ein mögliches Zeichen-n-Gramm, wobei der Wert, der einer Komponente eines Dokumentvektors zugeordnet wird, d.h. die Merkmalsgewichtung, eine nicht-lineare Funktion der Häufigkeit des Auftretens des entsprechenden n-Gramms in dem zugrunde liegenden Dokument ist. Die Stabilität der Dimension des Dokumentvektorenraums wird somit erkauft durch seine große Anzahl an Dimensionen, wobei das Gesamtystem dennoch dynamisch bleibt, da Dokumente neu indexiert werden, was die Verteilungsstruktur im Dokumentvektorenraum, sowie die Dimension des Merkmalsraumes verändert.

Allgemein wird der Umgang, die Modellierung und Optimierung und Control von nicht-stationären Funktionen als schwierig bewertet, wobei robuste, adaptive Verfahren wie evolutionäre Algorithmen notwendig werden (Goldberg & Smith (1987[145]), Grefenstette (1992[153]), Dasgupta & McGregor (1992[89]), Cobb & Grefenstette (1993[69]), Vavak et al. (1996[346]), Mori et al. (1997[224]), Mori et al. (1998[225]), Stagge (1998[322]), Dozier (2000[97]), Santo & Kita (2000[295])). In Kombination mit der Nicht-Linearität dieser Funktionen und den hochdimensionalen Räumen trägt diese Systemeigenschaft entscheidend zu der Gesamtkomplexität bei.

Die dynamische Aufnahme neuer Informationsobjekte koppelt ein IRS an die offene Welt (Popper (1994[262], 1995[263]), Mühlenbein (1994[229], 1995[230])), was explizite Folgen für die Methoden hat, da wahrscheinlichkeitstheoretische Induktionsmethoden nur in geschlossenen Welten gelten (Popper (1994: 376ff[262])). Ein IRS als offenes System (dissipatives System; Nicolis & Prigogine (1977[235])) in einer offenen Welt zu betrachten, wurde jedoch in der IR-Forschung bislang nicht als Modellierungsansatz verwendet.

1.2.4) Unsicherheit (uncertainty) und Vagheit (fuzzyness)

In Motro & Smets (1997[228]) werden Ursachen und Vorschäge zum Umgang mit Unsicherheit und Vagheit in Informationssystemen allgemein und IRS im speziellen (Turtle & Croft (1997[343])) gemacht (siehe auch Natke & Ben-Haim (1997[232])). Als Ursachen werden von Turtle & Croft (1997:191f[343]) folgende Faktoren genannt:
a) Dokumentrepräsentation
b) Repräsentation der Informationsbedürfnisse
c) Retrievalfunktion
d) Repräsentation im Kontext des Relevanz-Feedbacks

Die Indexierungsfunktion als Vorgang der Dokumentrepräsentation in vektorraumbasierten IRS bildet ein Dokument bzw. ein Merkmalshäufigkeitsvektor auf einen Dokumentvektor als Element eines metrischen Vektorraumes ab. Die Auswahl der Merkmale definiert die Struktur der Repräsentation, sodass Unsicherheiten bezüglich einer adäquaten Auswahl bestehen. Werden aus einer Dokumentmenge Terme automatisch extrahiert, so muss spezifiziert werden, welche Zeichensequenzen als Terme verwendet werden sollen, wodurch eine andere Unsicherheit bezüglich der Adäquatheit besteht.

Neben der Merkmalsauswahl ist die Merkmalsgewichtung die wesentliche Quelle der Unsicherheit, da eine Vielzahl von Vorschlägen für Gewichtungsfunktionen gemacht wurden (siehe z.B. Singhal (1997[312]), Zobel & Moffat (1998[376])). Welche Indexierungsfunktion für eine gegebene Dokumentmenge in Abhängigkeit von bestimmten Eigenschaften, die den Agenten als zukünftigen Nutzern zugeschrieben werden, hinreichend gut ist, kann nicht vorhergesagt werden (Zobel & Moffat (1998[376])).

Die Formulierung des Informationsbedürfnisses des Agenten in Form einer Query, führt weitere Unsicherheiten, Vagheiten und subjektive Faktoren wie der aktive Gebrauch eines Vokabulars ein und wird in der IR-Forschung am häufigsten zitiert. Wird das IR im Kontext eines Problemlösungsprozesses betrachtet (siehe Abschnitt 3.9.1)), und wird eine neuro-kognitive Sichtweise vertreten, die von funktionalen Hirnarealen ausgeht, so bedeutet die Queryformulierung eine Projektion interner Repräsentationsstrukturen aus einem funktionalen Problemlösungsareal auf Strukturen eines Sprachsyntheseareals. Es kann dabei nicht einmal von einer Abbildung als linkstotale, rechtseindeutige Relation ausgegangen werden, d.h. dass alle momentan existierenden Repräsentationsstrukturen aus dem Problemlösungsareal auf eine Struktur im Sprachsyntheseareal abgebildet werden können. Es besteht somit die Möglichkeit, dass Repräsentationsstrukturen nicht abgebildet werden und dass abgebildete Repräsentationen nicht umkehrbar abgebildet werden können. Sind z.B. bestimmte Vokabularelemente dem Agenten unbekannt, so können kognitive Strukturen darauf nicht abgebildet werden, was eine interne Quelle der Vagheit darstellt.

Wird ein interaktives Retrieval wie ein Relevanz-Feedback betrachtet (siehe Abschnitt 3.9), siehe auch Panyr (1987b[252]), Harman (1992[157]), Buckley & Salton (1995[63]), Cool et al. (1996[77]), Dunlop (1997[101]), Lundquist et al. (1997[203])), d.h. werden mehrere Dokumentlisten durch das IRS nachgewiesen, die der Agent bewerten soll, so kommen subjektive Faktoren, bezogen auf das Verstehen der Dokumente hinzu. D.h. die Abbildung von internen Strukturen aus dem Sprachanalyseareal auf Strukturen des Problemlösungsareals bildet eine weitere Quelle von Unsicherheiten und Vagheiten.

Weiterhin berücksichtigt werden muss, dass die Relationen bzw. Abbildungen zeitlich variabel sind, d.h. in allen Arealen verändern sich sowohl die Strukturen der Repräsentionen als auch die Anzahl der Repräsentationen. Entsteht in einem Areal eine neue Repräsentation durch interne Prozesse, so muss es in einer anderen Region, in die Repräsentationen abgebildet werden, keine oder noch keine korrespondierenden Strukturen geben, auf die abgebildet werden könnte. Denkbar wäre ein Prozess, dass zunächst auf nicht korrespondierende oder Default-Strukturen abgebildet wird, und dass im Rahmen einer Konsistenzherstellung neue Strukturen gebildet werden, auf die künftig eine Abbildung der entsprechenden Repräsentation erfolgt.

Als dritter Punkt wird von Turtle & Croft (1997[343]) die Retrievalfunktion angeführt, die als ret(DVM^t, q_i^t, d_{DVR}, ε) definiert wurde als Funktion der Dokumentvektorenmenge, eines Queryvektors, der Metrik des DVRs und einem Distanzschwellenwert. Unsicherheit besteht bezogen auf die externe Festlegung der Indexierungsfunktion $A_{IR(Q)}$ der Query, der Metrik und des Schwellenwertes. Die Metrik ist zudem zentraler Punkt jeder Rankingfunktion, die eine Ergebnis-Dokumentvektorenmenge auf eine geordnete Dokumentvektorenliste abbildet.

Als vierter Punkt werden von Turtle & Croft (1997[343]) Repräsentationen im Kontext des Relevanz-Feedbacks erwähnt. Relevanz-Feedback-Verfahren machen ein IRS zu einem adaptiven IRS, wobei interne Repräsentationen des IRS kurzfristig bzw. langfristig modifiziert werden. Wie diese Repräsentationen modifiziert werden, wird durch Adaptions-Funktionen definiert, die extern festgelegt werden müssen, und somit die Quelle von Unsicherheiten bezüglich der adäquaten Modifikation darstellen. Z.B. gehen Queryvektor-Relevanz-Feedback-Verfahren davon aus, dass die Query-Indexierungsfunktion $A_{IR(Q)}$ nicht adäquat ist, sodass die Ursprungsquery mit Hilfe der in der ersten Iteration nachgewiesenen Dokumentvektoren und der Relevanzbewertungen durch den Agenten in einen neuen Queryvektor mit Hilfe einer Queryvektor-Adaptions-Funktion überführt wird. Die Verkettung von $A_{IR(Q)}$ und der Adaptionsfunktion wird implizit als neue und adäquatere Query-Indexierungsfunktion verwendet. Neben dem Queryvektor-Relevanz-Feedback (siehe Abschnitt 3.9.2)) wird die Veränderung der Dokumentvektoren (Dokumentvektor-Relevanz-Feedback (siehe Abschnitt 3.9.3)) verwendet, während die Modifikation anderer Repräsentationsformen in der IR-Literatur unüblich ist (siehe Abschnitte 3.9.4) bis 3.9.8)).

1.2.5) Diversität der Agenten und ihre Ziele

Im Abschnitt über dynamische Zusammenhänge wurde bereits die Integration neuer Dokumente und Merkmale als Form der Kopplung eines IRS mit der offenen Welt beschrieben. Da das IRS keinen Einfluss auf die Produktion von Dokumenten besitzt, ist dies ebenfalls unter dem Gesichtspunkt der Unsicherheit interpretierbar, d.h. die Entwicklung der Repräsentationen in einem IRS sind durch diesen externen Faktor mit Unsicherheiten verbunden. Als weiterer externer Faktor wird die Diversität der Agenten betrachtet, d.h. es ist unsicher, welche Agenten mit welchen kognitiven Strukturen mit dem IRS zukünftig interagieren werden und welche Ziele sie damit verfolgen bzw. welche Probleme sie damit lösen wollen, wenn das IR im Kontext eines Problemlösungsprozesses betrachtet wird. Es werden immer neue Themen, Interessen, Ziele und Probleme auf der Ebene von Individuen und Gruppen gebildet, sowie immer neue Nutzergruppen aus unterschiedlichen Kulturen mit den IRS interagieren.

Weiterhin entstehen immer neue Mikro-Kulturen (Tribes) in denen Individuen mit gleichen Interessen, Zielen und Problemen sich global zusammenschließen, wobei ein Individuum zu einer Vielzahl unterschiedlicher Gruppen gehören kann. Ein solches Individuum kann nicht mehr mit einem Nutzermodell beschrieben werden, sondern es müsste kontextabhängig, d.h. in Abhängigkeit von der Gruppe, zu der das Individuum sich zum Zeitpunkt der Interaktion zugehörig fühlt, Nutzermodelle erzeugt werden, was eine Form von Polymorphismus darstellt. Durch diese Veränderungen wird die Diversität der Nutzer nicht nur in den Kontext der Unsicherheit, sondern auch in den Kontext dynamischer Zusammenhänge, d.h. nicht-stationärer Funktionen gestellt. Eine Polyrepräsentation von Nutzermodellen wird somit ein immer wichtig werdender Faktor, zumal eine wachsende Geschwindigkeit, in der sich die Diversität der Themen, Interessen, Ziele und Probleme vergrößert, beobachtbar ist.

Weiterhin kann ein Nutzer in einem engen Zeitintervall verschiedene Anfragen stellen, die aus unterschiedlichen Zielen und Problemen resultieren, sodass auch die Bildung eines Nutzermodells durch das IRS keinerlei Garantien besitzt, um Unsicherheiten bezüglich der Ziele des Nutzers zu verringern. Das gleiche Argument kann für Langzeit-Nutzer und periodische Nutzer etwa im Kontext von Information-Filter-Systemen angewendet werden, deren Anfragen, Ziele und Probleme einer zeitlichen Drift unterliegen kann.

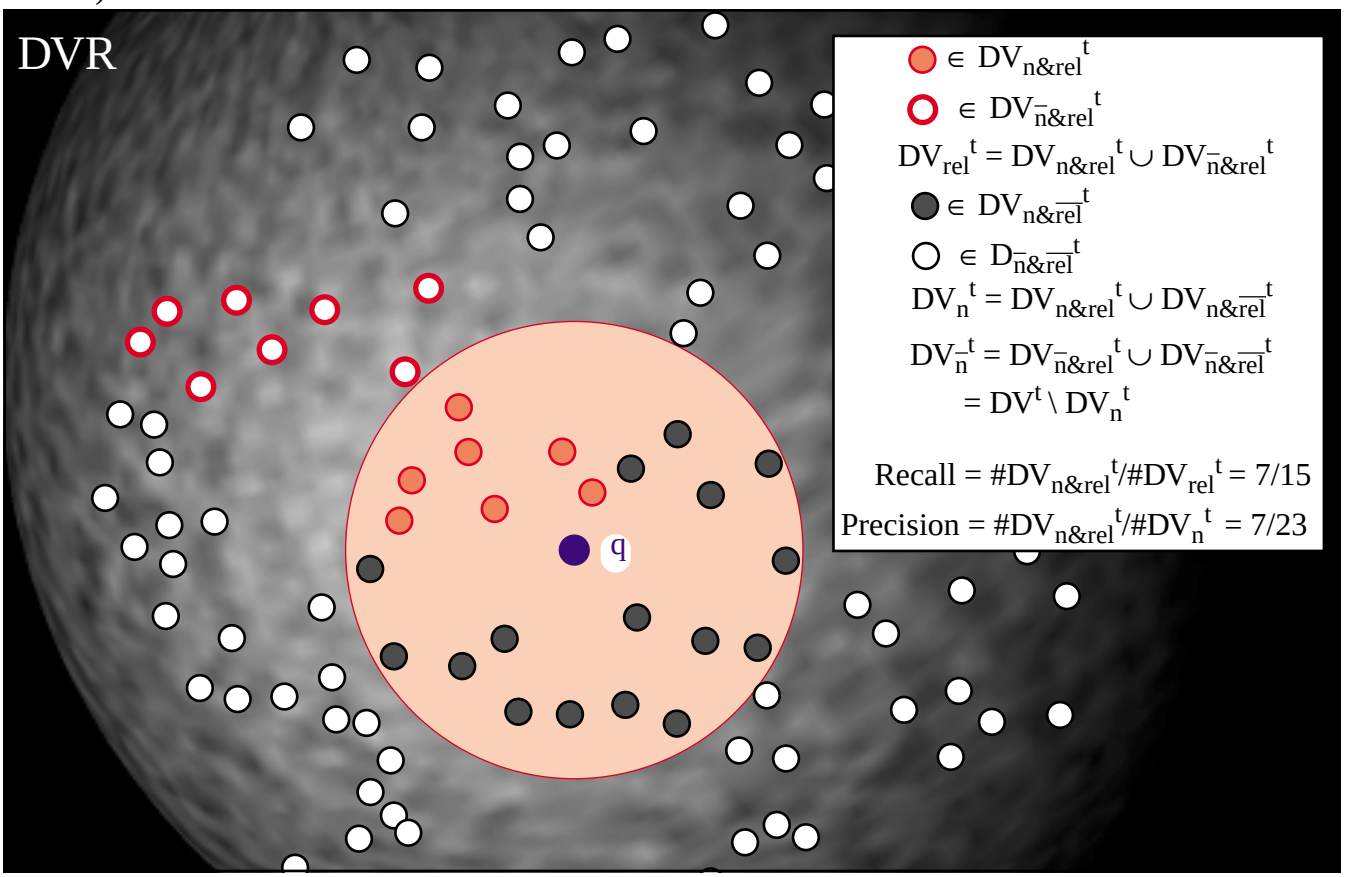
Die Bildung von Hypothesen über zukünftige Nutzer hat nicht nur Folgen für das Interfacedesign, sondern auch für die Wahl von Indexierungs- und Retrievalfunktionen. Der Versuch diese so zu wählen, dass einem „durchschnittlichen“ Agent geholfen wird, bestimmte Problemklassen zu lösen, ist eine sinnvolle Strategie, wenn von einer homogenen Agentenverteilung ausgegangen werden kann. Ist die Gesamtheit der Agenten jedoch bezüglich einer Vielzahl von Variablen divergent, und kann erwartet werden, dass das globale Leitbild der Individualisierung zu einer ständig größer werdenden Varianz in der Agentenverteilung führen wird, so ist die Auswahl einer einzelnen Indexierungs- und Retrievalfunktion und somit einer einzelnen Repräsentation von Informationsobjekten innerhalb des IRS keine geeignete Strategie. Diese Argumentation führt direkt zu dem Vorschlag der Polyrepräsentation in IRS (siehe Abschnitt 1.6) und Kapitel 3)).

1.2.6) Mehrziel-Anforderungen

Performancemessungen von IRS bezüglich einer Query werden durch Maße wie z.B. Recall als Quotient aus der Anzahl der nachgewiesenen relevanten Dokumente ($D_{n\&rel}^{t}$) und der Anzahl der gesamten relevanten Dokumente (D_{rel}^{t}) und durch Precision als Quotient aus der Anzahl der nachgewiesenen, relevanten Dokumente und der Anzahl der nachgewiesenen Dokumente (D_{n}^{t}) bestimmt. In Abb. 1) werden die Dokumentvektoren anstatt der korrespondierenden Dokumente verwendet, um die benötigten Mengen und die Definition von Recall und Precision zu illustrieren. Recall und Precision stellen komplementäre Maße dar, d.h. die Maximierung des einen Maßes minimiert das andere Maß. Die daraus ableitbare Generalisierung, dass die Optimierung eines IRS ein Mehr-Ziel-Optimierungs-Problem (Rosenthal (1985[290]), Steuer (1986[324]), Ringuest (1992[281]), Dasgupta et al. (1999[90]), Veldhuizen (1999[347]), Zitzler (1999[375])) darstellt, wurde in der IR-Literatur jedoch nicht explizit dargestellt.

Die Mehr-Ziel-Sichtweise gilt insbesondere, wenn über Recall und Precision hinaus zusätzliche Performancemaße verwendet werden, die jeweils bestimmte Charakteristika von IRS beschreiben, und die teilweise Überlappungen untereinander bezogen auf die gemessenen Charakteristika besitzen. Erwähnt werden kann z.B. die Utility als Differenz der gewichteten Anzahl der relevanten Dokumente und der gewichteten Anzahl der nicht-relevanten Dokumente (siehe Vogt (1999: 11[352])). Weiterhin können Performancemaße verwendet werden, die auf Rang-Korrelationen basieren (Bartell (1994[28])) und Maße, die sich aus der Signal-Detection-Theory (Egan (1975[107]), Swets (1996[329])) ableiten (siehe Vogt (1999:11f[352])).

Abb. 1) Definition von Recall und Precision



Ohne eine Menge von Performancemaßen ergibt sich eine Mehr-Ziel-Anforderung an IRS direkt aus den restlichen fünf beschriebenen Problemstellungen. D.h. das Handling hochdimensionaler, nicht-linearer und dynamischer Zusammenhänge unter dem Vorliegen von Unsicherheit und Vagheit in einem IRS ist bereits eine Mehr-Ziel-Anforderung.

1.3) Methodentransfer

Die vorgestellten Einzelprobleme und ihre Kombination sind nicht ausschließlich für das IR spezifisch, d.h. es existieren andere Untersuchungsbereiche, in denen ähnliche Problemfelder auftauchen. Z.B. zeichnet sich ein Produktionplanungssystem mit einer Menge unterschiedlicher Maschinen und einer zeitlich geordneten Liste an Jobs durch einen hochdimensionalen, diskreten Suchraum aus, in dem durch Schedulingverfahren nach optimalen bzw. suffizienten Maschinenbelegungsplänen gesucht werden muss

(Blazewicz et al. (1996[50])). Die damit verbundenen Probleme sind nicht-linear und nicht-stationär, wenn der Ausfall von Maschinen und der Ausfall bzw. die neue Aufnahme von Jobs disponiert werden müssen. Durch die Nicht-Beeinflussbarkeit dieser Faktoren durch das System ist es an die offene Welt gekoppelt, was Unsicherheiten und Vagheiten erzeugt. Durch zeitliche Constraints und monetäre Constraints entstehen zudem Mehr-Ziel-Anforderungen an die Leistung eines solchen Systems, sodass quasi alle aufgezeigten Problemfelder ebenfalls vorliegen.

Eine andere Domäne, in der Problemfelder dieser Art auftreten, ist die menschliche Kognition und ihre Modellierung durch kognitive Architekturen (Anderson (1990[8], 1991[9], 1993[10]), Newell (1990[233]), VanLehn (1991[345])). Dies dürfte nicht überraschen, da die Prozesse des IR, d.h. Indexierung und Retrieval, in Bibliotheken von Menschen lange vor der Verwendung von Computern durchgeführt wurden und immer noch durchgeführt werden. Wird das IR als Problem betrachtet, das von einem kognitiven System hinreichend gut gelöst werden kann, so liegt der Schluss nahe, kognitive Architekturen direkt als IRS zu verwenden. Dies ist jedoch mit einer Vielzahl von Problemen verbunden, die alle darin begründet sind, dass eine Menge von kognitiven Architekturen vorgeschlagen wurde, jedoch keine Architektur bezüglich Effektivität und Effizienz in den relevanten Problemstellungen dominierend ist. Doch selbst bei der Verwendung einer der Architekturen, die man bezogen auf Effektivitätskriterien als eine gute Architektur bezeichnen kann, ist ihre Leistungsfähigkeit gegenüber einer menschlichen Kognition immer noch bescheiden. Auf die kognitive Psychologie zu warten, um damit die Probleme des IR zu lösen, wäre eine vergleichbar verfehlte Strategie der IR-Forschung wie das Warten auf die KI in den 1980'er Jahren.

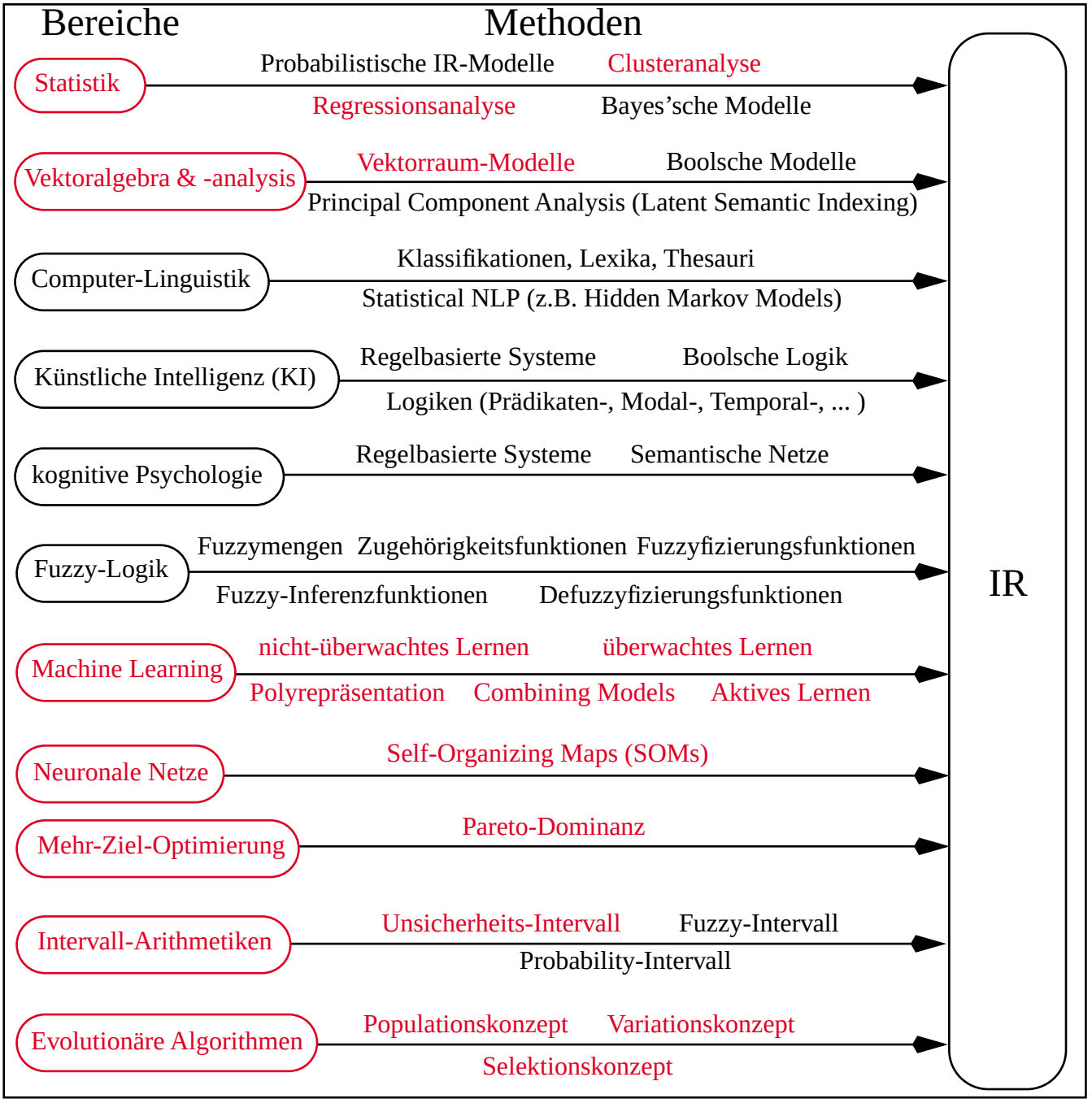
Es existiert eine Reihe weiterer Forschungsbereiche, welche mit gleichen bzw. ähnlichen Einzelproblemen konfrontiert sind, wie z.B. Machine Learning, Statistik, Optimierung und Control-Theory. In diesen Forschungsbereichen wurden eine Vielzahl von Methoden entwickelt, die zur Lösung der bereichsspezifischen Probleme eingesetzt werden. Durch die Erkenntnis, dass es sich um gleiche bzw. ähnliche Einzelprobleme handelt, wird ein Methodentransfer initiiert (siehe Abb. 2)), der über die gesamte Forschungshistorie des IR verfolgt werden kann. D.h. eine häufig beobachtbare Forschungslogik im IR besteht im Transfer von Lösungsmethoden aus Bereichen, in denen ähnliche Problemstellungen auftreten.

Betrachtet werden die Forschungsbereiche und die transferierten Methoden in Bezug auf die beschriebenen Problemfelder des IR, wobei Interdependenzen zwischen den Forschungsbereichen nicht detailliert beschrieben werden sollen wie z.B. Statistik und Computerlinguistik mit statistischen Natural-Language-Processing-Methoden (siehe jedoch Sprachlernen mit Evolutionären Algorithmen (Witten & Smith (1996[361]))).

Statistik ((Overbeck-Larisch & Dolejsky (1998[242]), Schuchmann & Sanns (1999a[303], b[304]), Abell & Braselton (1999[2])) beschreiben das Auftreten und die Eigenschaften von nicht-deterministischen Prozessen, d.h. es werden Unsicherheiten mit Hilfe von Konzepten wie z.B. Verteilungen bei der parametrischen Statistik beschrieben. Die direkte Methodenübertragung sind probabilistische Modelle des IR (Robertson (1977a[283], b[284], c[285], 1979[286]), Panyr (1986c[249]), Harter (1975[159]), Wong & Yao (1995[371]); Crestani et al. (1998[81])), bei denen z.B. für Merkmale die Wahrscheinlich-

keit berechnet werden, dass sie ein Dokument hinreichend gut beschreiben, oder bei denen für Dokumente die Wahrscheinlichkeit berechnet wird, dass sie bezogen auf eine Query relevant sind. Verbunden ist damit das „Probability Ranking Principle“ (Robertson (1977c[285])), bei dem die Dokumentergebnismenge in eine geordnete Liste überführt wird, indem als Ordnungsprinzip die fallende Wahrscheinlichkeit der Relevanz eines Dokumentes bezüglich der aktuellen Query verwendet wird.

Abb. 2) Methodentransfer ins Informations Retrieval



Die Zuordnung von Relevanzwahrscheinlichkeiten bei probabilistischen Modellen verwendet unterschiedliche Verfahren der schließenden Statistik wie Inferenz-Netze (Turtle & Croft (1990[341], 1991[342]), Turtle (1991[344]) oder Inferenzstrategien auf der Basis von Bayes-Verfahren (allg. siehe Martignon & Laskey (1995[211]); Turtle & Croft (1990[341], 1991[342]), Cooper (1991[78]), Kwok (1995[194]), Larkey & Croft (1996[196]), Henrion et al. (1997[161])).

Clusteranalysen (Eckes & Rossbach (1990[103]), Jain et al. (1999[175]), McLachlan & Basford (1987[214])) gehören in den Bereich der nicht-parametrischen Statistik (Schuchmann & Sanns (1999a[303])) und erzeugen aus einer Gesamtmenge von Objekten wie Dokumenten oder Merkmalen entsprechend der Repräsentation dieser Objekte Teilmengen (Cluster), die flach oder hierarchisch strukturiert werden können. Wie bereits beschrieben, werden Clustermethoden im IR im Kontext der hochdimensionalen Repräsentationen der Informationsobjekte eingesetzt (curse-of-dimensionality, Bellmann (1967), Warwick & Kárny (1997[357])).

Mit Regressionsmodellen (Ryan (1997[292]), Rawlings et al. (1998[273])) wird versucht, einen funktionalen Zusammenhang zwischen zwei oder mehr Variablen zu beschreiben, mit denen Objekte repräsentiert sind. Im Rahmen dieser Arbeit nimmt die Relevanz-Regression als Spezialfall der Relevanz-Approximation eine entscheidende Stellung ein (siehe Abschnitt 1.7) und Kapitel 4), siehe auch Cooper et al. (1992[79]); Gey (1994); Robertson & Walker (1994[287])). Die Zuordnung von Relevanzwerten zu Dokumenten bezüglich einer Query durch Regressionsmodelle ist eine alternative Vorgehensweise zu der Zuordnung von Relevanzwahrscheinlichkeiten in probabilistischen IR-Modellen, wobei Relevanzwerte bei Regressionsverfahren aus dem Intervall [0, 1] auch als Wahrscheinlichkeiten eines probabilistischen Modells interpretierbar sind.

Das Vektorraum-Modell des IR (Salton (1971[293]), Panyr (1987a[251])), auf das sich die vorliegende Arbeit bezieht, baut auf Begriffen wie Vektorraum, Distanzfunktion und Metrik auf, die aus der Vektoralgebra und Vektoranalysis (Strampp (1996[325]), Trostel (1993[339], 1997[340])) stammen. Verfahren des Latent-Semantic-Indexing (Dumais et al. (1988[100]), Furnas et al. (1988[136]), Bartell et al. (1992[27]), Soboroff et al. (1998[315]), Hofmann (1999[165]), Ando (2000[11])) basieren ebenso auf dem Vektorraummodell wie konnektionistische Verfahren des IR (Scholtes (1993[301]), Bachelier (1995[14]), Cunningham et al. (1997a[85], b[86])).

Boolsche Modelle des IR (Waller & Kraft (1979[354])) sind die ältesten Modellierungen, die Beziehungen zum Vektorraum-Modell sowie zu logischen IR-Modellen besitzen (Boolsche Logik). Grundlage von Queries sind Terme, die mit den Operatoren „Und“ ($\wedge$), „Oder“ ($\vee$), „Nicht“ ($\neg$) verknüpft werden, wobei Klammern hinzu kommen können, um strukturierte, hierarchische Ausdrücke zu erzeugen. Dokumente werden im einfachsten Fall als Binärvektoren repräsentiert, d.h. jedem Term wird ein Komponent des Binärvektors zugeordnet, wobei die Komponente mit 1 belegt wird, wenn das Merkmal in dem zu repräsentierenden Informationsobjekt auftritt bzw. die Komponente wird mit 0 belegt, wenn das Merkmal nicht auftritt. Eine Retrievalfunktion prüft, ob ein Dokument bezüglich der Query passend ist, d.h. ob bestimmte Terme enthalten sind oder ob bestimmte Terme nicht auftreten. Nachgewiesene Dokumente können nicht unterschieden werden, sodass als Ergebnis eine Menge und keine nach einem Rankingkriterium geordnete Liste vorliegt. Erweiterte Boolsche Modelle versuchen ein Ranking zu erzeugen, indem sie Gewichte der Terme in den Dokumenten bzw. in der Query verwenden, wenn diese berechnet werden bzw. vom Agenten spezifiziert werden (Lee (1994[198]), Grossman & Frieder (1999:58ff[154])). Dies kennzeichnet einen Übergang zu vektorraumbasierten Modellen, da eine Distanzfunktion zwischen einer Query und einem Dokument auf der Basis der Terme in der Query und den Gewichten als Distanz in einem metrischen Vektorraum betrachtet werden kann.

Computer-Linguistik allgemein versucht die Spracherkennung, das Sprachverstehen (Ram & Moorman (1999[270])) und die Spracherzeugung beim Menschen zu modellieren und zu implementieren (siehe z.B. Batori et al. (1989[29])), wobei eine Vielzahl symbolischer sowie subsymbolischer, d.h. konnektionistischer Ansätze (z.B. Scholtes (1993[301])) vorgeschlagen wurden. Zu den symbolischen Ansätzen gehören die Verwendung von Klassifikationen, Lexika und Thesauri, während statistische Methoden der Verarbeitung natürlicher Sprache (Statistical Natural Language Processing (Manning & Schütze (1999[210])) die Schnittmenge zwischen den symbolischen und konnektionistischen Ansätzen darstellen. Zu den statistischen Verfahren gehören insbesondere Hidden Markov Modelle (Buchholz (1991[60]), Sinclair (1992[311]), Davis (1993[92]), Buchholz et al. (1994[61]), Elliott et al. (1995[109])), für die auch Anwendungen in IR existieren (Miller et al. (1999[219])). Da in dieser Arbeit symbolische Ansätze wie die Verwendung von Lexika und Thesauri nicht betrachtet werden, besteht ein Anknüpfungspunkt bei den statistischen und konnektionistischen Verfahren, wobei insbesondere der Ansatz zu erwähnen ist, dass Dokumente, Queries und Merkmale (Features) als Zeichenketten eines endlichen Alphabetes betrachtet werden. Etwas periphär werden im Rahmen der Polyrepräsentation Sprachgenerierungsmodelle betrachtet, die aus einem Dokument oder einer Query eine Menge von Dokumenten bzw. Queries erzeugen, die nachfolgend indexiert werden. Dabei werden insbesondere Zeichenketten-Resampling-Verfahren betrachtet wie das Moving-Blocks-Verfahren (siehe Abschnitt 2.2.3)).

Künstliche Intelligenz (KI) (Görz (1993[151]), Bundy (1997[64])) versucht allgemein Hard- und Software zu entwickeln, die Leistungen erbringen, für welche beim Menschen das Attribut intelligent verwendet würde. Dabei wird keine biologische oder psychologische Plausibilität wie bei Neuronalen Netzen oder Modellen der kognitiven Psychologie gefordert, sondern es werden ausschließlich Effektivität und Effizienz als Performancemaße verwendet. Die KI hat seit Mitte der 50’er Jahre eine große Anzahl von Repräsentationsformen und Lernformen entwickelt, von denen einige im Kontext des IR eingesetzt wurden, wobei regelbasierte Systeme und logikbasierte Systeme angesprochen werden sollen.

Regelbasierte Systeme (Produktionssysteme; Newell (1990[233])) besitzen eine Wissensbasis in Form einer Menge von Wenn-Dann-Regeln. Kombimiert wird diese Wissensbasis mit einen Regelinterpreter, der in Abhängigkeit von einer Menge momentan aktiver Wissenselemente in einem Arbeitsspeicher (Working-Memory) ermittelt, welche Regeln anwendbar sind, indem er die Wissenselemente mit den Wenn-Teilen (If-Teile) der Regeln vergleicht (matched). Je nach der Art des Interpreters führt er alle anwendbaren Regeln aus, eine Teilmenge oder die erste passende Regel, die er findet. Der Dann-Teil (Then-Teile) der Regeln kann Wissenselemente im Arbeitsspeicher modifizieren, löschen, neue erzeugen, externe Aktionen auslösen u.a. Die Regeln in der Wissensbasis werden meist von Menschen erzeugt, bzw. es werden Lernmechanismen implementiert, die neue Regeln erzeugen können, was den Bereich des Machine Learnings berührt.

Der Einsatz von regelbasierten Systemen im IR überschneidet sich mit der Argumentation des Einsatzes von Modellen der kognitiven Psychologie im IR. Die Hypothese der kognitiven Psychologie, dass kognitive Prozesse durch Produktionssysteme adäquat modellierbar sind (Anderson (1990[8], 1991[9], 1993[10]), Newell (1990[233])), ist der Grund, warum diese Systeme im IR eingesetzt werden. Die Indexierung wurde und wird von menschlichen Experten durchgeführt, bevor Computersysteme sich an dieser Aufgabe versuchten. Ziel ist es in diesem Kontext, die Indexierungsregeln zu modellieren, von denen

angenommen wird, dass die Experten sie verwenden, um so Dokumenten Deskriptoren zuordnen zu können, und sie somit zu klassifizieren. Diese Regeln matchen mit ihrem If-Teil in einem Dokument bestimmte Strukturen, wie das Auftreten von einzelnen Termen bzw. die Kombination von Termen. Werden entsprechende Strukturen gematched, so wird durch den Then-Teil der Regel Deskriptoren und Gewichtungen zugeordnet (Tong et al. (1983[333]), Tong & Shapiro (1985[334]), Jüttner & Güntzer (1988[178])).

Die Hauptprobleme von regelbasierten Systemen im IR bestehen in dem Aufwand der Erzeugung der Regeln durch Experten, sowie die ad hoc Definition von Funktionen, die ein Gewicht eines Merkmals in einem Informationsobjekt bestimmt. Weiterhin liegen nach Turtle & Croft (1997: 208[343]) keine Erfahrung bezüglich der Eignung regelbasierter Systeme für große, heterogene Dokumentmengen vor. Insbesondere die dynamischen Zusammenhänge und die Diversität von Agenten und ihren Zielen ist durch eine geschlossene Menge von Regeln nicht modellierbar, sodass Regellernen ein besonderer Schwerpunkt sein muss (Fung et al. (1990[135])).

Bei dem Einsatz klassischer Formen der Logik (Gabbay et al. (1993[137]), Owsnicki-Klewe et al. (1993[243])) wie Prädikatenlogik erster Stufe im IR werden Dokumente und Queries als eine Menge von Aussagen (Propositionen) betrachtet. Dabei wird eine Ähnlichkeit zwischen Dokumenten und einer Query als Funktion bestimmt, wie gut ein Dokument aus einer Query mit Hilfe von Inferenzverfahren abgeleitet werden kann, wobei zusätzlich Domänwissen verwendet werden kann (Chiaramella & Chevalett (1992[67]), Hess (1992[162]), Nie (1992[236])).

Bezüglich einer Wissensdomäne kann vorab nicht festgelegt werden, ob sie durch die Mittel, welche die Prädikatenlogik zur Verfügung stellt, adäquat modellierbar ist. Bestimmte Bereiche, in denen zeitliche und räumliche Aspekte eine dominante Rolle spielen, haben sich als schwierig modellierbar mit diesen Beschreibungsmitteln erwiesen, sodass eine große Anzahl von Erweiterungen entwickelt wurde, um diese Probleme zu überwinden z.B. durch temporale und probabilistische Logiken (Gabbay et al. (1994a[138], b[139], 1995[140])).

Nicht-klassische Logiken werden im IR z.B. von Rijsbergen (1986[280]) vorgeschlagen, wobei mehrwertige Logiken einen Übergang zur Fuzzy-Logik darstellen (Rölleke & Fuhr (1996[288]), Fuhr et al. (1998[133])), während probabilistische Logiken im IR (Nie (1992[236])), welche nicht-deterministische Inferenzstrategien nutzen, direkte Beziehungen zu den probabilistischen Modellen des IR besitzen.

Die Logikansätze besitzen zum einen Probleme hinsichtlich der beschränkten Mächtigkeit der Repräsentationssprache sowie bezüglich der Eigenschaften der Inferenzmechanismen (siehe z.B. Owsnicki-Klewe et al. (1993: 37 f[243])). Es kann nicht ausgeschlossen werden, dass ein Inferenzmechanismus eine exponentielle Anzahl von Variablen erzeugt, sodass damit hoch-dimensionale Suchräume eingeführt werden. Da Dokumente durch eine große Anzahl von Deskriptoren beschrieben werden müssen, um eine ausreichende Differenzierung ähnlicher Inhalte zu erreichen, können deterministische Inferenzmechanismen ohne zusätzliche Heuristiken zu nicht vertretbaren Retrievalzeiten führen.

Die kognitive Psychologie (Anderson (1990[8], 1991[9], 1993[10]), Strube et al. (1993[326])) versucht kognitive Prozesse beim Menschen zu modellieren, um deren Charakteristika zu untersuchen. Dabei sollen auch Fehlleistungen modelliert und untersucht werden, was im Gegensatz zur KI steht, die versucht, intelligentes Verhalten unter Beseitigung von Fehlleistungen durch Computermodelle zu erzeugen. Während in der Frühzeit der kognitiven Psychologie die Modellierung einzelner Fähigkeiten und Fehlleistungen im Vordergrund stand, treten seit Ende der 80'er Jahre ambitionierte Ansätze in den Vordergrund, welche das gesamte kognitive System in einer einheitlichen Weise modellieren wollen, was mit dem Schlagwort der „Unified Theories of Cognition" umschrieben wird (Newell (1990[233])). Vorgeschlagen werden kognitive Architekturen (VanLehn (1991[345])), die symbolisches Wissen aufnehmen, speichern und verarbeiten können, bei denen grob zwischen einem Architektur- und einem Wissenslevel unterschieden wird (eine feinere Strukturierung in vier Level wird von Newell (1990[233]), Anderson (1991[9], 1993[10]) und Pylyshin (1991[266]) vorgeschlagen). Auf dem Architekturlevel finden grundlegende Verarbeitungsoperationen statt, die durch Wissen und Ziele eines Subjektes nicht verändert werden können, wobei der Architekturlevel eines Subjektes jedoch nicht unveränderlich sein muss, da die Möglichkeit einer ontogenetischen Entwicklung existiert, auch wenn diese in entsprechenden kognitiven Architekturen bislang unberücksichtigt bleibt. In höheren biologischen Systemen entspricht der Architekturlevel die Unterteilung in spezialisierte Hirnregionen, die Kommunikation der Regionen sowie die neurologischen Prozesse innerhalb der Regionen, d.h. die Fähigkeit von Neuronen zur Leitung bzw. Hemmung elektrischer Signale und die Ausschüttung von Botenstoffen. Bei gängiger Hardware kann der Architekturlevel mit dem Registerlevel identifiziert werden. Auf dem Wissenslevel werden lernbare Erfahrungen im Hinblick auf die Erreichung von Zielen gespeichert und verarbeit, d.h. Ziele wie Wissen sind variabel.

Im Rahmen der kognitiven Modellierungen werden keine prinzipiell anderen Verfahren eingesetzt als bei der Modellierung von Intelligenz im Bereich der KI, während bei kognitiven Modellen zusätzliche Constraints bezüglich psychologischer Plausibilität hinzu treten. Beispielsweise werden Produktionssysteme (SOAR (Newell (1990[233]), Rosenbloom et al. (1993[289])), Semantische Netze (Collins & Loftus (1975[76]), Schank (1975[298])) oder hybride Architekturen von Produktionssystemen und semantischen Netzen (ACT* oder ACT-R; Anderson (1990[8], 1993[10])) zur kognitiven Modellierung eingesetzt. Da Produktionsysteme im KI-Kontext als regelbasierte Systeme beschrieben wurden, sollen sich die nachfolgenden Darstellungen auf semantische Netze beschränken.

Semantische Netze (Collins & Loftus (1975[76]), Schank (1975[298]), Anderson (1990[8], 1993[10]), Grossman & Frieder (1999:118ff[154]), Lee & Dubin (1999[197])) gehen davon aus, dass menschliches Wissen durch Konzepte und sie verbindende Relationen repräsentiert werden kann. Modelliert werden Konzepte durch Knoten, die durch Verbindungskanten, die Relationen zwischen den Knoten beschreiben, verbunden sind, wobei mehrere Kantentypen wie z.B. „ia_a" und „part_of" unterschieden werden können. Den Knoten können variable Aktivierungen zugeordnet werden und den Kanten sind konstante Gewichte zugeordnet, die zusammen mit einer Regelung, wie sich die Aktivität in Abhängigkeit von den Gewichten über das Netz ausbreitet (Spreading Activation), das Verhalten des Netzes in einem gegebenen Fall deterministisch bestimmt. Die Aktivitätsausbreitung in einem Netz findet sich neben den semantischen Netzen auch in Inferenz-Netzen (Turtle & Croft (1990, 1991), Turtle (1991)) und in Neuronalen

Netzen und wird im Kontext der lateralen Aktivitätsausbreitung in Growing-Neural-Gas-SOMs in Abschnitt 2.1.8) detaillierter beschrieben.

Semantische Netze, Merkmalsgraphen u.ä. Strukturen können im IR zur Query-Expansion bzw. allgemeiner zur Query-Modifikation verwendet werden (Kwok (1991[193], 1995[194]), Efthimiadis (1993[106]), Scholtes (1993[301]), Bachelier (1995[14]), Mostafa et al. (1997[227])). Eine Query überträgt Aktivierungen auf die Merkmale, die in der Query enthalten sind (Initialisierungsknoten), wobei das Maß der Aktivierung eine Funktion der Merkmalsgewichtung in der Query ist, die durch ein unabhängiges Verfahren berechnet wird (s.a. Singhal (1997[312])). Die Aktivierungen breiten sich ausgehend von den Initialisierungsknoten iterativ in dem Graphen aus, bis ein Abbruchkriterium erfüllt ist. Die Aktivierung des gesamten Graphen wird als modifizierte Query betrachtet, indem die Umkehrfunktion der Gewicht-Aktivierungsfunktion angewendet wird.

Die Struktur Semantischer Netze muss in den frühen Ansätzen durch einen Menschen erzeugt werden, sodass diese Ansätze genau wie regelbasierte Systeme kaum geeignet sind, in dynamischen, nicht-stationären Umgebungen adäquate Modellierungen zu liefern. Merkmalsgraphen, die auf der Basis des Vektorraum-Modells bzw. daraus abgeleiteter konnektionistischer Verfahren erstellt werden (Scholtes (1993[301]), Bachelier (1995[14])), besitzen durch adaptive Verfahren den Vorteil, dass die Gewichtsstruktur der Kanten, die Kantenstruktur sowie die Merkmalsknoten dynamisch verändert, neu erzeugt und eventuell gelöscht werden können, ohne dass hierzu externe Eingriffe notwendig werden.

Wie bereits erwähnt, wird der potentielle Einsatz kognitiver Modelle (Daniels (1986[88])) im IR damit begründet, dass natürliche kognitive Systeme, d.h. Menschen, in der Lage sind, die Einzelaufgaben des IR hinreichend gut zu lösen, sodass kognitive Architekturen direkt als IRS anwendbar sein sollten. Für diese Argumentation spricht, dass natürliche, kognitive Systeme mit sehr großen Datenmengen umgehen können, wenn auch in unterschiedlichen Modalitäten verschieden gut, d.h. visuelle Daten können in sehr großem Umfang verarbeitet, wie z.B. klassifiziert werden, während der Umgang mit textuellen Daten erheblich längere Entscheidungszeiten benötigt. Die Argumentation „kognitive Architekturen als IRS“ übersieht jedoch, dass ein definierendes Merkmal kognitiver Architekturen die psychologische Plausibilität ist, worunter auch kognitive Fehlleistungen fallen. Man darf nicht annehmen, dass unterschiedliche menschliche Experten im Indexierungsprozess (oder Abstraktionsprozess (Borko & Bernier (1975[52]), Pfeiffer-Jäger (1980[257]), Mathis & Rush (1985[213]), Bernier (1985[44]), Lancaster (1991[195]), Mani & Maybury (1999[208])) die gleichen Entscheidungen treffen, d.h. es muss keine intersubjektive Reliabilität bezüglich der Einzelaufgaben des IR angenommen werden. Dementsprechend ist auch das Kriterium der Validität problematisch, an dem ein IRS gemessen werden soll, da unterschiedliche, menschliche Experten zu unterschiedlichen Ergebnissen z.B. bezüglich der Deskriptorvergabe kommen werden, und da diese Ergebnisse als einzig verfügbare Benchmarks für IRS verwendet werden können. Würde kaum eine Beschränkung bezüglich der Verfügbarkeit menschlicher Experten bestehen, so könnte ein Expertenkomitee von Indexierern in Verbindung mit einem Delphie-Verfahren (siehe z.B. Fraunhofer-Institut für Systemtechnik und Innovationsforschung (1998[120]) eine reliablere Indexierung erzeugen. Dementsprechend müsste ein Komitee von diversiven kognitiven Architekturen eingesetzt werden, die ein Spektrum von Eigenschaften abdecken, was in den Bereich der Polyrepräsentation und der Combining Models im Machine Learning führt.

Fuzzy-Logik (Höhle & Rodabaugh (1999[166]), Novak & Perfilieva (1999[239]), Nguyen (1999[234])) beschreibt die Vagheit von Objektrepräsentationen und Relationen durch Zugehörigkeitsfunktionen zu Fuzzymengen bzw. vage definierten Konzepten. Auf diese Weise lassen sich unscharfe, linguistische Attribute wie „groß“, „alt“, „schnell“ mathematisch modellieren.

Die Erforschung der Potentiale der Fuzzy-Logik im IR gehen bis in die Mitte der 70’er Jahre zurück (Tahani (1976[330]); Radecki (1979[269])), während in jüngerer Vergangenheit entsprechende Ansätze nur selten zu finden sind, wobei die alleinige Anwendung von Fuzzy-Logik als Indexierungs- und Retrievalverfahren faktisch nicht vorkommen. Z.B. stellen Damiani & Fugini (1995[87]) ein System vor, das in Verbindung mit einem Thesaurus, der Fuzzy-Synonym-Relationen beinhaltet, vage Anfragen bearbeiten kann. Besonders relevant ist die Beziehung zwischen Fuzzy-Sets und Clusteranalyse (Fuzzy-Clusteranalyse, Höppner et al. (1997[167])), da die Clusteranalyse eine lange Tradition innerhalb der IR-Forschung besitzt, sodass eine Fuzzy-Clusteranalyse größeres Anwendungspotential im IR besitzt.

Eine Kritik der Fuzzy-Ansätze findet sich in Panyr (1986d[250])), wobei ein Hauptargument darin besteht, dass es sich bei Fuzzy-basierten IRS um keine neue Modellklasse handelt, sondern dass diese Ansätze äquivalent zu probabilistischen und statistischen Ansätzen im IR sind, wobei es sich im günstigsten Fall um eine elegante Beschreibungssprache handelt. Fuzzy-Operatoren besitzen zudem eine zu geringe Trennschärfe, was zu einer nicht akzeptabel großen Menge nachgewiesener Dokumente in IRS führen kann, sodass auch bei den frühen Ansätzen diese Operatoren verworfen wurden, während das Potential linguistischer Variablen günstig bewertet wird (Panyr (1986d:164[250])). Die zu diesem Zeitpunkt offene Frage der mathematischen Grundlegung der Fuzzy-Logik konnte weitgehend positiv beantwortet werden (Höhle & Rodabaugh (1999[166]), Novak & Perfilieva (1999[239])). Dies bedeutet jedoch nicht, dass sich dadurch Fuzzy-basierten IRS bislang ein breiteres Einsatzfeld eröffnet hätte.

Machine Learning (Morik (1993[226]); Witten & Frank (1999[362]), Witten et al. (1999[363])) wird als Teil der KI verstanden, bei dem ein System aus Erfahrungen lernen soll, d.h. seine Repräsentationen und somit sein Wissen modifizieren soll, um seine Ziele effizienter und effektiver zu erreichen. Geschieht dieses Lernen durch die externe Vorgabe von Beispielen durch einen Lehrer, so wird der Lernprozess als überwacht (superviced) bezeichnet (Wolpert (1995b[369])). Muss das System die zu lernenden Beziehungen selbst suchen, die beim überwachten Lernen in den Beispielen kodiert sind, so wird der Lernprozess als unüberwacht bezeichnet (Kohonen (1989[184], 1995[185]), Hinton & Sejnowski (1999[164])). Beide Lernprozesse besitzen eine direkte Beziehung, da überwachtes Lernen in ein unüberwachtes Lernen und unüberwachtes Lernen in ein überwachtes Lernen überführt werden kann (siehe die Darstellung zum gemeinsamen Framework von überwachtem und unüberwachtem Lernen in Abschnitt 2.1.2.3)).

Einen Überblick über Anwendungen des Machine Learning im IR gibt Cunningham et al. (1997b[86]), wobei im Kontext dieser Arbeit die Combining Models von besonderer Bedeutung sind, die einen Meta-Ansatz zum Machine Learning darstellen. Der grundlegende Ansatz ist die Erzeugung einer Menge von Modellen bezüglich einer Menge überwachter Beispiele. Eine solche Erzeugung kann abhängig sein, z.B. indem für jedes Modell eine gleichverteilt, zufällige Stichprobe aus der Beispielmenge gezogen wird, oder die Erzeugung kann unabhängig sein, indem eine disjunkte Zerlegung der Beispielmenge durchgeführt wird.

Aus jeder Beispielmenge wird ein Modell aufgebaut, wobei zwischen einem Interparadigmen- und einem Intraparadigmen-Combining-Model unterschieden werden kann. Bei einem Interparadigmen-Combining-Model werden unterschiedliche Ansätze des Machine Learnings wie z.B. Regel-Induktion, Entscheidungsbäume (Decision-Trees; Quinlan (1993[267], 1996[268])) und Neuronale Netze verwendet, um die verschiedenen Einzelmodelle zu erzeugen, während bei einem Intraparadigmen-Combining-Model genau ein Ansatz verwendet wird, um eine Modellmenge zu erzeugen.

Sind die Einzelmodelle erzeugt, so werden sie in einer Meta-Architektur kombiniert, wie z.B. durch eine Abstimmungsfunktion (Voting), oder indem alle Einzelmodelle ihren Output in ein hierarchisch darüber gelegenes Modell einbringen, das Gewichtungen lernen kann (Stacking; Wolpert (1990[365], 1993[367]), Edelman (1995[104]), Witten & Ting (1999[364])), wobei die Stacking-Architektur über mehrere Hierarchieebenen konstruiert werden kann.

Combining Models im IR beziehen sich auf die Kombination unterschiedlicher Methoden, um zu einem Gesamtergebnis zu gelangen, was im Kontext des IR oft als Evidenz-Kombination bezeichnet wird (Vogt (1999: 25ff[352]), siehe z.B. Bartell (1994[28]), Lee (1997[199]), Schapire et al. (1998[299]), Baumgarten (1999), Vogt & Cottrell (1998[351]), Kim et al. (2000[181])). Hierfür existieren gewichtige Gründe (s.a. Belkin et al. (1995[38])):

1) Dokument- und Query-Repräsentationen sind nicht exakt.
   Unterschiedliche Modelle behandeln diese Unsicherheiten und Unschärfen jeweils anders, wobei Ergebnisse erzeugt werden, die in anderen Systemen nicht erzeugt werden können, und die bei einer Kombination zur Verfügung stehen.
2) Behandlung von nicht-relevanten Dokumenten
   In verschiedenen Arbeiten wird darauf hingewiesen, dass Modelle, die unterschiedliche Mengen nicht-relevanter Dokumente auf eine Query liefern, gute Kandidaten zur Kombination sind (Lee (1997[199])).
3) Expansion des Parameterraumes
   Unterschiedliche IRS-Typen bzw. Modelltypen verwenden unterschiedliche Parameter, mit denen Relevanzschätzungen durchgeführt werden, sodass die Kombination dieser Modelle auf der Ebene der Ergebnisse implizit eine Kombination der Parameter bedeutet.

In Diamond (1996[95]) (nach Vogt (1999: 26[352])) werden drei Effekte beschrieben, durch welche Kombinationen von Retrievalergebnissen zu einer Verbesserung der Gesamtperformance führen:

1) Skimming-Effekt
   Werden die gleichen Dokumente durch unterschiedliche Modelle repräsentiert, so wird erwartet, dass bei der Verwendung der ersten Rangplätze der Rankings der unterschiedlichen Repräsentationen, eine Recall- und eine Precision-Verbesserung eintritt.
2) Chor-Effekt
   Wird ein Dokument durch verschiedene Modelle als relevant klassifiziert, so ist dies eine stärkere Evidenz, dass das Dokument wirklich relevant ist, als wenn es nur von einem Modell nachgewiesen wird.

3) Dark-Horse-Effekt
   Unterschiedliche Modelle besitzen unterschiedliche Fähigkeiten bezüglich der Relevanzbewertung, sodass eine Kombination unterschiedliche Gewichtungen der einzelnen Modelle berücksichtigen sollte.

In Vogt (1999: 25ff[352]) werden drei Ansätze beschrieben, mit denen eine Evidenz-Kombination im Information Retrieval durchgeführt werden kann:
1) System-Selektion.
2) Daten-Fusion.
3) Kollektion-Fusion.

Bei der System-Selektion wird eine Query einer Menge von IRS zugeführt, die unabhängig voneinander Ergebnisse produzieren. Diese werden analysiert, und es werden die Ergebnisse eines IRS selektiert, die dem Nutzer bzw. Agenten präsentiert werden. Entgegen der intuitiven Annahme, dass die Auswahl des besten IRS einfacher ist als die Kombination von unterschiedlichen IRS, hat sich der System-Selektions-Ansatz bislang nicht bewährt (Savoy et al. (1996[296])).

Methoden zur Daten-Fusion verwenden unterschiedliche Techniken bezüglich einer Dokumentkollektion, indem unterschiedliche IRS unabhängig voneinander eine Query bearbeiten. Die Dokument-Ergebnislisten werden als Input in eine Kombinationsfunktion verwendet, wobei die bekannten Ansätze zur Daten-Fusion als Output genau eine Dokument-Ergebnisliste verwenden, um die Erwartung des Agenten bezüglich eines einzelnen Rankings zu erfüllen. Alternative Outputstrukturen wie mehrere Dokumentmengen, -listen oder -hierarchien haben entscheidenden Einfluss auf die Komplexität der Kombinationsfunktion. D.h. wird der Constraint eines einzelnen Rankings aufgegeben, und werden alternative, graphische Präsentationsformen verwendet, so könnten viele neue Kombinationsfunktionen untersucht werden, die möglicherweise effektiver und effizienter sind als die bislang verwendeten, doch diese Ansätze wurden bislang nicht untersucht.

Eine spezielle Form der Daten-Fusion ist die Query-Kombination (Vogt (1999: 29f[352])), bei der ein Agent eine Query erzeugt, die mehrfach repräsentiert und in Kombination mit genau einer Dokumentkollektion verwendet wird. Denkbar ist, dass ein zugrunde liegendes Informationsbedürfnis durch den Agenten mehrfach und jeweils unterschiedlich formuliert wird, sodass unterschiedliche Queries entstehen, mit denen jeweils ein Retrievalprozess durchgeführt wird, oder die zu einer Meta-Query aggregiert werden.

Während bei der Daten-Fusion unterschiedliche IRS auf genau eine Dokumentkollektion zugreifen, beziehen sich bei der Kollektion-Fusion unterschiedliche IRS auf unterschiedliche Dokumentkollektionen, die entweder disjunkt sind oder eine gewisse Überlappung zeigen. Im Regelfall bezieht sich jede der IRS auf genau eine Dokumentmenge, d.h. es wird von einer 1-zu-1-Beziehung ausgegangen, während eine 1-zu-n-Beziehung eine Mischung von Daten- und Kollektion-Fusion darstellt.

Mit den Combining Models direkt verbunden ist das Konzept der Polyrepräsentation (Ingwersen (1992[171], 1994[172])), das allgemein die mehrfache, unterschiedliche Repräsentation von Wissen bezeichnet, im Gegensatz zur Redundanz, welche die mehrfache, gleichartige Repräsentation bezeichnet. In Analogie zu den Combining Models kann zwischen einer Interparadigmen- und einer Intraparadigmen-Polyrepräsentation unterschieden werden, wobei die erste Form die Repräsentation eines Informationsobjektes in unterschiedlichen Repräsentationsformen bzw. -sprachen bezeichnet. Dies geschieht bei Interparadigmen-Combining-Models quasi automatisch, da z.B. Regel-Induktion, Decision-Trees und Neuronale Netze sehr unterschiedliche Repräsentationsformen besitzen. Intraparadigmen-Polyrepräsentation bezieht sich auf Intraparadigmen-Combining-Models, da in einer Repräsentationssprache ein Informationsobjekt ebenfalls durch unterschiedliche Ausdrucksformen beschrieben werden kann, die jeweils unterschiedliche Aspekte fokussieren können.

Aktives Lernen ist eine Klasse von Methoden im Machine Learning, mit denen der Modellaufbau mit wenigen Beispielen durchgeführt werden soll, was insbesondere dann sinnvoll ist, wenn die Ermittlung der Beispiele einen großen Ressourceneinsatz erfordert, oder sonstigen Begrenzungen unterliegt (Baum & Lang (1991[32]), Baum (1991[31]), Cohn (1994[71], 1995[73]), Cohn et al. (1990[70], 1994[72], 1995[74]), Fedorov (1972[113]), Freund (1993[122]), Freund et al. (1993[123]), MacKay (1992[204]), Paaß & Kindermann (1995a[244], b[245]), Plutowski & White (1993[258]), Plutowski (1994[259]), RayChaudhuri & Hamey (1995[274], 1996[275]), RayChaudhuri (1997[276]), Seung et al. (1992[308]), Sollich (1993[316], 1994[317]), Thrun & Möller (1992[332]); siehe auch Abschnitt 1.8) und Kapitel 5)).

Ein neuronales Netz wird als gerichteter Graph beschrieben, der aus einer Menge von formalen Neuronen als Knoten und gerichteten Verbindungskanten besteht (Brause (1991[58]), Ritter et al. (1991[282]), Schöneburg et al. (1990[302]), Bachelier (1998a[15])). Ein formales Neuron n_j wird allgemein als Tupel mit fünf Komponenten beschrieben, die einen Gewichtsvektor w_j, eine Inputfunktion net_j, eine Aktivierungs- oder Zustandsfunktion z_j, eine Outputfunktion y_j und eine Lernregel Δw_j umfassen (siehe auch Bachelier (1995: 22f[14])). Besteht ein neuronales Netz aus dem gleichen Typ formaler Neurone, so sind alle Komponenten bis auf den Gewichtsvektor konstant und können aus der Beschreibung eines einzelnen Neurons auf die Beschreibungsebene des Netzes ausgelagert werden.

Im Kontext des IR kann ein Informationsobjekt wie ein Dokument und ein Merkmal wie z.B. ein Term durch ein formales Neuron repräsentiert werden, wobei die Beziehung zum Vektorraummodell des IR offensichtlich ist, indem ein Gewichtsvektor eines Neurons als Dokument- oder Termvektor interpretiert wird. Auf diese Weise können neuronale Netze Dokument- und Termmengen repräsentieren und unüberwachte sowie überwachte Lernoperationen auf der Basis der Informationsobjekte durchführen. Es können Dokument- und Term-Cluster gebildet werden, indem man Typen neuronaler Netze verwendet, die unüberwachtes Lernen durchführen, wobei Self-Organizing-Maps (SOMs; Kohonen (1989[184], 1995[185]), siehe auch Abschnitte 2.1.4) bis 2.1.8)) und daraus abgeleitete Modellvarianten häufig für Zwecke des IR verwendet werden (Scholtes (1993[301]), Bachelier (1995[14]), Merkl (1997[215]), Chen et al. (1997[65]), Roussinov et al. (1999[291])). Neben der reinen Repräsentation und der Clusterung, was der Indexierung entspricht, finden sich auch Anwendungen neuronaler Netze in allen anderen Bereichen des IR, z.B. der Query-Modifikation und -Expansion und in Information Filter Systemen (IFS; z.B. Belkin & Croft (1992[36]), Goldberg et al. (1992[147]), Panyr (1994[253]), Allan (1996[5]),

Mostafa et al. (1997[227]), Schapire et al. (1998[299])). Einen Überblick über die Einsatzmöglichkeiten neuronaler Netze in IRS und IFS gibt z.B. Cunningham et al. (1997a[85]).

Eine Mehr-Ziel-Optimierung im Kontext von IR ergibt sich, wenn ein IRS adaptiv seine Repräsentationen verändern soll, wobei mehrere Zielanforderungen gleichzeitig berücksichtigt werden müssen, d.h. wenn die Adaption durch mehrere Performancemaße geregelt werden soll. Von fundamentaler Bedeutung ist dabei das Konzept der Pareto-Dominanz, das durch eine dreiwertige Dominanzfunktion zweier Alternativen operationalisiert wird (Rosenthal (1985[290]), Steuer (1986[324]), Ringuest (1992[281]), Dasgupta et al. (1999[90]), Veldhuizen (1999[347]); Zitzler (1999[375]), siehe auch Abschnitt 2.4.1)). Eine Alternative x_i, der ein Vektor $f(x_i) = (f_k(x_i) \in R \mid k = 1, ...)$ mehrerer Performancemaße zugeordnet ist, wird gegenüber einer Alternative x_j, als Pareto-dominant bezeichnet, wenn alle Einzelmaße $f_k(x_i)$ nicht schlechter sind als das korrespondierende Maß $f_k(x_j)$ und wenn mindestens ein Maß besser ist als die konkurrierende Alternative. Existiert mindestens ein Maß, das besser ist und mindestens ein Maß, das schlechter ist, so sind beide Alternativen nicht-dominant.

Intervalle werden in Bereichen wie Statistik und Fuzzy-Logik verwendet, um Unsicherheiten (Davis (1987[91]), Hyvönen (1992[170]), Wilke et al. (1998[359])) und Vagheit einer Variablen zu repräsentieren. In der Statistik werden Konfidenz-Intervalle verwendet, um eine untere und obere Grenze von geschätzten Variablenwerten bezogen auf ein Signifikanzniveau anzugeben, d.h. es werden Aussagen erzeugt wie „mit 98% Wahrscheinlichkeit liegt der richtige, unbekannte Wert in dem Intervall“ (Abell & Braselton (1999[2]), Schuchmann & Sanns (1999a[303], b[304])). Bei Fuzzy-Sets besitzen die einfachen Zugehörigkeitsfunktionen über einem endlichen Intervall einen positiven Zugehörigkeitswert. Die einfachste Zugehörigkeitsfunktion ist eine Dreiecksfunktion, die links von einer unteren und rechts von einer oberen Intervallgrenze den Zugehörigkeitswert Null besitzt. In dem Intervall steigt der Zugehörigkeitswert linear bis zu einem Maximum an, das bei einer symmetrischen Funktion über dem Intervallmittelpunkt liegt. Rechts vom Maximum fällt die Zugehörigkeitsfunktion linear bis zur rechten Intervallgrenze, und verläuft dann auf der Abzissenachse.

Es existieren Arithmetiken, welche die speziellen Eigenschaften von Intervallen gegenüber reellen Zahlen als Punkt-Intervalle nutzen, wobei Intervall-Arithmetiken (Bauch et al. (1987[30]), Hansen (1992[155]), Adams & Kulisch (1993[3]), Klatte et al. (1993[182]), Baker-Kearfott & Kreinovich (1996[25])) und Fuzzy-Arithmetiken (Buckley & Feuring (1999)) unterschieden werden können. Weiterhin existieren Ansätze Unsicherheit und Vagheit zu kombinieren, indem unpräzise Wahrscheinlichkeiten (Probability-Intervalle) eingeführt werden (Walley (1991[355]), Berleant (1996[43])).

Der Einsatz von Intervall-, Fuzzy- oder Probability-Intervall-Arithmetiken zur Modellierung von Unsicherheiten und Vagheit in IRS wurde bislang vernachlässigt, d.h. kein existierendes IRS, unabhängig von den verwendeten Paradigmen, verwendet Intervall-Arithmetiken zur Indexierung, Clusterung, Retrieval, Query-Modifikation oder einem anderen Teilbereich.

Evolutionäre Algorithmen sind iterative Verfahren, welche durch die Konzepte Population, Variation und Selektion beschrieben werden (Schwefel (1977[305], 1981[306], 1995[307]), Goldberg (1989[146]), Nissen (1994[237]), Rechenberg (1994[277]), Jacob (1997[174]), Bachelier (1998b[16]), siehe auch Abschnitt 2.5)). Begonnen wird mit einer Initialisierungs-Population $P^{t=0}$ von Individuen $a_i^{t=0}$, wobei ein Individuum eine Datenstruktur ist, welche einen Lösungsvorschlag bezüglich einer Fitnessfunktion (Umwelt) kodiert. Im einfachsten Fall ist eine Population eine unstrukturierte Datenstruktur wie eine Menge. In der Reproduktionsphase wird mehrfach eine Selektion zur Reproduktion als Funktion angewendet, welche aus der Population eine Teilmenge auswählt. Wird jeweils genau ein Individuum ausgewählt, so handelt es sich um eine ungeschlechtliche Reproduktion, ansonsten um eine ρ-geschlechtliche Reproduktion, $\rho > 1$. Die Variations-Operation erfolgt polymorph in Abhängigkeit von der Anzahl der Elternteile. Bei einer ungeschlechtlichen Reproduktion wird aus dem einen Elternteil ein Nachkomme erzeugt, indem die Datenstruktur des Elternteils kopiert und auf die Kopie ein Mutations-Operator angewendet wird, der die Datenstruktur durch einen stochastischen Prozess variiert. Bei einer geschlechtlichen Reproduktion wird ein Rekombinations-Operator auf die Elternteile angewendet, der einen Nachkommen erzeugt, indem Kopien der Datenstrukturen der Eltern durch einen stochastischen Prozess neu zusammengefügt werden. Der so erzeugte Nachkomme wird in einem weiteren Schritt einem Mutations-Operator unterzogen.

Nachdem durch diese Reproduktionsoperationen eine Sollanzahl von Nachkommen erzeugt wurde, wird die Reproduktionsphase beendet, und es folgt die Bewertungsphase, in der alle Nachkommen durch eine Fitnessfunktion bewertet werden, d.h. ihnen wird im einfachsten Fall einer Ein-Ziel-Bewertung genau ein Skalarwert zugeordnet. Es folgt die Erzeugung einer Ordnung der Nachkommen eventuell zusammen mit den Elternindividuen entsprechend der Bewertung als Ordnungskriterium, wobei im einfachsten Fall eine geordnete Liste von Individuen erzeugt wird.

Nach der Bewertung erfolgt die Selektion zur Übernahme in die Nachfolgepopulation, indem aus der Ordnungsstruktur der Nachkommen (und Eltern) eine Teilstruktur ausgewählt wird, welche die besten Individuen entsprechend dem zugrunde liegenden Ordnungskriterium beinhaltet. Bei der nach fallenden Fitnesswerten geordneten Liste werden somit die ersten und daher die besten Individuen ausgewählt. Diese Individuen bilden die Nachfolgepopulation, die in die nächste Phase der Selektion zur Reproduktion übergeht, wenn kein Abbruchkriterium greift.

Evolutionäre Algorithmen können zudem als Lernverfahren zum Aufbau von Modellen (überwachtes Lernen) sowie für unüberwachtes Lernen verwendet werden, sodass eine direkte Beziehung zum Machine Learning sowie zu den Neuronalen Netzen besteht (Gerdes et al. (2000[143])). In ähnlicher Weise besteht auch eine Beziehung zur Mehr-Ziel-Optimierung, da Mehr-Ziel-EA (Multi-Objective-EA, MOEA) effektive und effiziente Verfahren ermöglichen, komplexe Mehr-Ziel-Optimierungsaufgaben zu bewältigen, die mit klassischen Verfahren nicht handhabbar sind (Fonseca & Fleming (1993[115], 1995[116]), Binh & Korn (1996[46], 1997[47]), Deb (1998[93]), Veldhuizen (1999[347]), Zitzler (1999[375])).

Evolutionäre Verfahren im Rahmen des IR beziehen sich auf die Adaption von Repräsentationen der betrachteten Informationsobjekte durch evolutionäre Algorithmen unter Verwendung von Relevanzwerten, was als Relevanz-Feedback betrachtet werden kann, wobei als Adaptionsverfahren evolutionäre Algorithmen verwendet werden.

In Gordon (1988[150]) wird eine evolutionäre Adaption der Dokumentrepräsentationen durch einen genetischen Algorithmus auf der Basis von Querymengen und Feedback verfolgt, was somit als evolutionäres Dokumentvektor-Relevanz-Feedback betrachtet werden kann. Es wird eine Population alternativer Dokumentrepräsentationen eines Dokumentes erzeugt, die Reproduktions- und Selektions-Operationen unterzogen werden, wobei die Fitnessfunktion aus den Bewertungen bezüglich der Querymenge abgeleitet wird. Da alle Repräsentationen innerhalb eines Paradigmas stattfinden, handelt es sich um eine Intraparadigmen-Polyrepräsentation der Dokumente, was sich auch dadurch ergibt, dass Rekombinations-Operatoren zwischen Repräsentationen unterschiedlicher Paradigmen ausgeschlossen sind bzw. problematisch sind. Dies besitzt eine direkte Analogie zur Biologie, indem Reproduktions-Operationen, d.h. der Austausch genetischen Materials, zwischen unterschiedlichen Spezies ausgeschlossen bzw. schwer möglich (jedoch nicht ausgeschlossen) ist.

In Blair (1990[49]) und in Yang & Korfhage (1994[374]) wird eine Query-Modifikation durch ein ähnliches Vorgehen unter Verwendung von Genetischen Algorithmen vorgeschlagen, was somit als evolutionäres Queryvektor-Relevanz-Feedback betrachtet werden kann.

Die vielfältigen Ansätze bezüglich Repräsentationsart und Suchverfahren im IR zeigen jedoch, dass sich kein einzelner Ansatz bezüglich unterschiedlichen Dokument- und Querymengen als dominant erwiesen hat. Z.B. zeigen Zobel & Moffat (1998[376]), dass keine Gewichtungsfunktion und keine Distanzfunktion innerhalb des vektorbasierten IR bezüglich verschiedener Dokument- und Querymengen allen anderen überlegen ist. Man kann daraus die Vermutung ableiten, dass eine Chance auf eine signifikante Verbesserung der Performance von IRS nur durch die Integration unterschiedlicher Ansätze erfolgen wird, was der Grundannahme der Interparadigmen Polyrepräsentation nach Ingwersen (1992[171], 1994[172]) entspricht.

1.4) Adaptive Informationssysteme

Adaption ist die Fähigkeit eines Repräsentationssystems (Newell (1990[233])) zur Anpassung seiner Repräsentationen, Repräsentationsarten und Suchverfahren (Problemlösungsstrategien) an externe Rahmenbedingungen. Hierbei werden Repräsentationssysteme als physikalische Systeme betrachtet, die isolierte Zustände von Teilen ihrer Umwelt auf isolierte Zustände in ihnen selbst abbilden können, sodass die internen Zustände die externen Zustände für das System repräsentieren. Durch die Adaption bleibt die grundsätzliche Systemarchitektur erhalten, d.h. eine Adaption verändert nur Strukturen auf der Repräsentations- bzw. Wissensebene, nicht jedoch Strukturen auf der Architekturebene (Anderson (1991[9]), Newell (1990[233]), Pylyshin (1991[266])). Entsprechend dieser Definition von Adaption sind Individuen innerhalb evolutionärer Verfahren (Schwefel (1977[305], 1981[306], 1995[307]), Goldberg (1989[146]), Nissen (1994[237]), Rechenberg (1994[277]), Jacob (1997[174]), Bachelier

(1998b[16]), siehe auch Abschnitt 2.5)) nicht adaptiv, während eine Population als Instanz adaptiv ist. Eine Hybridisierung durch adaptive Individuen in evolutionären Populationen wird als Ansatz betrachtet, Adaptivität auf der Repräsentationsebene und der Architekturebene zu modellieren (Belew & Mitchell (1996[35])).

Es herrscht ein breiter Konsens in Gebieten wie KI, Machine Learning, Neuronale Netze, Evolutionäre Algorithmen, Kognitionswissenschaft u.a., dass nur adaptive Systeme in der Lage sind, Aufgaben und Ziele in einer komplexen Umgebung zu erfüllen, die durch dynamische, veränderliche, nicht-lineare und hochdimensionale Anforderungen beschrieben werden müssen. Unter diesem Gesichtspunkt lassen sich viele der Versuche des Methodentransfers in den Bereich des IR interpretieren, d.h. es wird mit dem Transfer unterschiedlicher Lernverfahren und zugehöriger Datenstrukturen aus den Bereichen Machine Learning, der Neuronalen Netze und evolutionärer Verfahren versucht, adaptive IRS zu generieren.

Dass Lernverfahren im IR eine fundamentale Bedeutung besitzen, wurde schon sehr früh erkannt und in Form von Relevanz-Feedbackverfahren (Borodin et al. (1971[56]), Brauen (1971[57]), Friedman et al. (1971[127]), siehe auch Panyr (1987b[252]) und Abschnitt 3.9)) umgesetzt. Relevanz-Feedbackverfahren sind iterative Retrievalmethoden, bei denen der Agent, der ein Informationsbedürfnis als Anfrage in der Initialisierungsiteration an das IRS gestellt hat, aufgefordert wird, die Relevanz nachgewiesener Informationsobjekte zu bewerten. Mit Hilfe dieser Relevanzbewertungen verändert das IRS seine eigenen Repräsentationen entweder reversibel für die Interaktionszeit mit dem betrachteten Agenten, oder irreversibel für alle zukünftigen Interaktionen mit anderen Agenten. Diese Veränderung der eigenen Repräsentationen auf Grund der externen Rahmenbedingungen, die durch die Relevanzbewertungen des Agenten gegeben sind, machen ein IRS mit einem Relevanz-Feedbackmechanismus zu einem adaptiven IRS. Trotz der langen Tradition von Relevanz-Feedbackverfahren (siehe auch [1]Aalbersberg (1992[1]), Harman (1992[157]), Buckley et al. (1994[62]), Biron & Kraft (1995[48]), Buckley & Salton (1995[63]), Allan (1995[4], 1996[5]), Dunlop (1997[101]), Lundquist et al. (1997[203]), Iwayama (2000[173])) wurde bislang nur ein Ausschnitt des Möglichen beschrieben, indem nur die Repräsentation bestimmter Informationsobjekte adaptiert wurde, wie z.B. beim Queryvektor-Feedback oder Dokumentvektor-Feedback. Insbesondere die Möglichkeiten der Adaption der verwendeten Funktionen wie Indexierungs- und Retrievalfunktion, oder von Metriken in den betrachteten Vektorräumen, wurde nicht oder kaum berücksichtigt. Dies lag bislang möglicherweise daran, dass effektive Verfahren zur Suche in diskreten Funktionsräumen nicht vorhanden bzw. nicht eingesetzt wurden. Eine Ausnahme bildet die Suche nach einer Retrievalfunktion durch Genetic Programming (Koza (1992[188], 1994[189]), Nordin (1997[238]), Banzhaf et al. (1998[26]), Koza et al. (1999[191])) durch Fan et al. (1999[112]), wobei anzunehmen ist, dass im Rahmen der Adaption von Funktionen Genetic Programming in Zukunft eine dominante Rolle spielen wird.

Die Lernverfahren im Rahmen des Relevanz-Feedbacks, wie z.B. die gemischte positive und negative Adaption im Rahmen des Queryvektor-Relevanz-Feedbacks (siehe Abschnitt 3.9.2)), muss jedoch als rudimentäres Lernverfahren im Vergleich zu den Verfahren bewertet werden, die in den vergangenen 30 Jahren in den Disziplinen Machine-Learning, statistisches Lernen und Neuronale Netze entwickelt wurden. Die Einführung unterschiedlicher Lernmethoden z.B. aus dem Bereich der Neuronalen Netze (Scholtes (1993[301]), Bachelier (1995[14]), Merkl (1997[215]), Chen et al. (1997[65]), Roussinov et al.

(1999[291])) und Lernmethoden durch Combining Models (Bartell (1994[28]), Lee (1997[199]), Schapire et al. (1998[299]), Baumgarten (1999[33]), Vogt & Cottrell (1998[351]), Vogt (1999[352]), Kim et al. (2000[181])) im IR können als Versuche interpretierbar werden, komplexere Lernverfahren und somit adaptivere Systeme zu erzeugen.

1.5) Einbettung externer Informationsbeschaffung in ein Modell der allgemeinen Intelligenz

Ein Agent soll im Rahmen dieser Arbeit als rational unter den Constraints seines Wissens betrachtet werden, d.h. wenn er ein Ziel hat und weiß, wie dieses durch eine Aktion erreicht werden kann, dann führt er diese Aktion aus (Newell (1990[233])). Die Entscheidung, mit einem IRS zu interagieren, folgt aus einem Bedürfnis der externen Informationsbeschaffung, das wiederum eingebettet sein muss in den Prozess des Versuches, Ziele zu erreichen. Im weiteren soll dargestellt werden, wie das Bedürfnis nach externer Informationsbeschaffung aus den kognitiven Prozessen der Zielerreichung unter Verwendung des grundlegenden Ansatzes der allgemeinen Intelligenz nach Newell (1990[233]) und der Operationalisierung des Rationalitätsprinzips nach Anderson (1990[8], 1991[9], 1993[10]) hergeleitet wird. Die von diesen Autoren darauf aufbauenden Annahmen sollen jedoch weder dargestellt noch übernommen werden.

Ein Verhaltenssystem ist ein physikalisches System (Agent), das Aktivitätssequenzen gerichtet auf seine Umwelt oder auf andere Subsysteme von sich abgeben kann. Systemaktivität wird als Veränderung der Konfiguration eines Teilsystems betrachtet, die eine physikalische Wirkung in die Umwelt oder andere Subsysteme abgibt.

Ein Repräsentationssystem ist ein Verhaltenssystem mit einem konfigurierbaren, physikalischen (Repräsentations-)Medium als Subsystem, wobei Elemente des Mediums externe Objekte repräsentieren und Relationen der Elemente des Mediums externe Relationen repräsentieren. Auf diese Weise können interne, mentale Modelle externer Umweltzustände, -relationen und -transformationen gebildet werden. Das System hat die Möglichkeit durch interne Transformationen Konfigurationen des Mediums in andere Konfigurationen zu überführen. Die Mächtigkeit eines Repräsentationssystems ist durch die Anzahl der Konfigurationen und Anzahl und Art der Transformationsfunktionen bestimmt.

Ohne dass dies von Newell (1990[233]) oder Anderson (1990[8], 1991[9], 1993[10]) spezifiziert wird, soll von einer Umwelt als offene Welt im Sinne von Popper (1994[262], 1995[263]) ausgegangen werden, was Auswirkungen auf den Status von Repräsentationen hat, da nie von einer verifizierbar, adäquaten Repräsentation externer Objekte und Relationen ausgegangen werden kann.

Repräsentationssysteme können einen Ist-Zustand externer Umweltzustände repräsentieren, und mit einer Repräsentation eines Soll-Zustandes externer Umweltzustände vergleichen, sowie mit einer Repräsentation eines Alternativ-Zustandes externer Umweltzustände vergleichen. D.h. es existieren Teilsysteme des Repräsentationssystems, die solche Vergleichs- und Bewertungsfunktionen operationalisieren, wodurch die Funktionen Teil der Systemarchitektur und somit für das System unveränderlich sind. Alter-

nativ könnten die Bewertungsfunktionen im Teilsystem des Mediums repräsentiert werden, was sie zum Teil der Repräsentationsebene macht, sodass das System Einfluss auf die Funktionen besitzt.

Verhaltens- wie Repräsentationssysteme möchten einen externen Ist-Zustand durch ein externes Verhalten in einen externen Soll-Zustand umwandeln. Zur Verfügung steht zunächst jedoch nur eine interne Repräsentation des externen Ist- und Soll-Zustandes. Insbesondere der Soll-Zustand kann durch Constraints beschrieben werden, wenn präzisere Beschreibungen nicht vorliegen.

Unter Wissen wird die interne Repräsentation des Zusammenhanges zwischen internem Ist-Zustand und einer internen Transformationsfunktion verstanden, um einen internen Soll-Zustand zu erreichen, d.h. Wissen bezieht sich im Kern auf interne Modelle, ihre Eigenschaften und Beziehungen. Die interne Transformationsfunktion ist dabei eine Repräsentation eines externen Systemverhaltens, d.h. einer Aktion, mit der ein externer Ist-Zustand in einen externen Soll-Zustand umgewandelt werden soll. Wissen ist in diesem Kontext prozedural und kann als Regel beschrieben werden, sodass die Operationalisierung des Modells allgemeiner Intelligenz in Form des Systems SOAR (Newell (1990[233]), Rosenbloom et al. (1993[289])) naheliegend als Produktionssystem erstellt wurde. Eine Regel als „Wissensatom“ wird beschrieben durch „Wenn Ist-Zustand gegeben und Soll-Zustand angestrebt, dann wende Verhalten an“ oder durch eine Funktionsdarstellung, die einen Ist-Zustand als Input aufnimmt und einen End-Zustand als Output ausgibt „Verhalten (Ist-Zustand, End-Zustand) ≡ V(I,E)“.

Für Newell (1990[233]) ist alles Wissen prozedurales Wissen, d.h. eine Repräsentation, die auf interne Transformation und somit auf Systemverhalten abzielt. Im Gegensatz hierzu unterscheidet Anderson (1990[8], 1991[9], 1993[10]) prozedurales und deklaratives Wissen, wobei deklaratives Wissen Systemzustände selbst beschreibt. Da prozedurales Wissen Ist- und Soll-Zustände umfasst, umfasst es somit auch deklaratives Wissen, wohingegen deklaratives Wissen isoliert für Newell (1990[233]) nicht existiert. Ob deklaratives Wissen in der menschlichen Kognition existiert, ist eine Frage der Kognitionspsychologie und in dem betrachteten Kontext irrelevant, da von allgemeinen Wissenssystemen ausgegangen wird, in denen deklaratives Wissen durch prozedurales Wissen simulierbar ist, sodass ausschließlich prozedurales Wissen betrachtet werden kann.

Die Einführung eines Repräsentationssystems als Subsystem bereitet dem Agenten als Ganzes einen Vorteil, da externes Verhalten an einem internen Modell einer in Nicht-Standard-Situation erprobt werden kann. Das Scheitern einer Aktion, d.h. das sich nicht Einstellen eines Soll-Zustandes des intern repräsentierten, exteren Systems, kann für das Repräsentationssystem keine existenzielle Bedrohung erzeugen. Es können unterschiedliche Aktionen sequentiell getestet und jeweils der sich einstellende interne Zustand mit dem Soll-Zustand verglichen werden. Diese Situation kann als interne Selektion bezeichnet werden (siehe auch Bachelier (1999d[22])), wobei Repräsentationen innerhalb des Agenten selektiert werden, anstatt der externen Selektion, bei der Agenten durch das sich nicht Einstellen eines externen Soll-Zustandes selektiert werden. Dadurch kommt es zu einem größeren Verhaltensspektrum, das für andere Situationen zur Verfügung steht, falls die Kombination aus Ausgangszustand des internen Modells, Aktion und Endzustand des internen Modells gespeichert und wiedergewonnen werden kann, d.h. wenn der Übergang eines Repräsentationssystems zu einem Wissenssystem gelingt.

Ein Wissenssystem ist ein Repräsentationssystem, das sein verfügbares Wissen einsetzen kann, um interne und somit externe Soll-Zustände zu erreichen. Nach Newell (1990[233]) ist ein Wissenssystem ein (erreichbares) Ideal, da gefordert wird, dass das System sein gesamtes Wissen einsetzen muss, d.h. alles Wissen ist verfügbar und wird auf seine Anwendbarkeit geprüft. Mit Hilfe des Wissenssystems wird der Begriff der Intelligenz eines Systems definiert, indem Intelligenz als Grad der Approximation eines Wissenssystems durch ein existierendes Repräsentationssystem betrachtet wird. Diese Definition besitzt zwei bekannte Probleme, da zum einen ein einfaches System mit wenig Wissen und wenigen Zielen perfekt intelligent sein kann, und zum anderen können komplexe Systeme ebenfalls perfekt intelligent sein, wenn sie beliebig viel Zeit besitzen, um die Anwendbarkeit ihres gesamten Wissens zu prüfen. Beide Probleme lassen sich nur unter Berücksichtigung der Umwelteigenschaften lösen, sodass Intelligenz eine Funktion von System- und Umwelteigenschaften und nicht nur von Systemeigenschaften wird. Zum ersten kann gefordert werden, dass die Umwelt einen bestimmten Komplexitätsgrad besitzt, sodass Soll-Zustände nicht durch triviale Handlungen erzeugbar sind, und somit einfache Systeme in diesem Sinne nicht intelligent sein können. Im zweiten Fall muss die Forderung eines Echt-Zeit-Constraints hinzu treten, d.h. das System muss Verhalten auf der Basis seines Wissens in angemessenen Zeitspannen generieren, die von der Umwelt vorgegeben werden. Der Echt-Zeit-Constraint ist somit eine relative und keine absolute Forderung bezüglich einer physikalischen Zeit, da ein System in einer langsam interagierenden Umwelt den Echt-Zeit-Constraint auch dann erfüllen kann, wenn es viele physikalische Zeiteinheiten zur Verhaltensgenerierung benötigt.

Wird ein Wissenssystem durch die Anwendung von Wissen beschrieben, so stellt sich die Frage, wie das System an das Wissen gelangt. In einer einfachen und stationären Umgebung könnten Wissenssysteme, die sich reproduzieren, a priori Wissen besitzen und vererben, während komplexere und nicht stationäre Umgebungen den Erwerb von Wissen durch Lernen erfordern. Der grundlegende Mechanismus wird von Newell (1990[233]) mit „Generate and Test“ umschrieben, d.h. durch einen (oder mehrere) noch zu spezifizierende Erzeugungsmechanismen wird eine Repräsentation eines internen Verhaltens erzeugt, das die interne Repräsentation eines Ist-Zustandes in einen Soll-Zustand überführen soll. Die Funktion aus internem Verhalten, Ist- und Soll-Zustand ist zunächst ein hypothetisches Wissen (Belief). Der interne Test bezeichnet die interne Anwendung des Verhaltens auf den internen Ist-Zustand und den Vergleich des sich ergebenden Zustandes mit dem Soll-Zustand. Erst wenn ein internes Verhalten gefunden wird, das den internen Ist- in den Soll-Zustand überführt, wird das Verhalten extern ausgeführt und getestet. Zugrunde liegt die Annahme der Rationalität, d.h. ein Wissensssystem wird ein Verhalten extern nicht ausführen, wenn das Verhalten sich an dem internen Modell nicht bewährt hat. Ein Verhalten wird unter der Annahme der Rationalität nur dann extern erzeugt, wenn es intern verifiziert wurde bzw. im Kontext des internen Modells und Performance-Constraints wie der Echt-Zeit-Forderung, nicht falsifiziert werden konnte.

Der externe Test umfasst zunächst die Ausführung des externen Verhaltens, gefolgt von der Beobachtung und Repräsentation des sich einstellenden, externen End-Zustandes und dem internen Vergleich des repräsentierten End-Zustandes mit dem internen Soll-Zustand. Unter Berücksichtigung der sensorischen und repräsentatorischen Constraints des Wissenssystems liefert der Vergleich ein Feedback, das zur Evidenzverstärkung des Wissens führt, wenn beobachteter End-Zustand und Soll-Zustand hinreichend ähnlich sind. Es wird eine Evidenzverstärkung verwendet, da eine externe Verifikation in einer offenen Welt

prinzipiell nicht möglich ist (Popper (1994[262], 1995[263])). Demgegenüber kann das Feedback Anlass zur Modifikation der Repräsentation des internen Verhaltens werden, wenn externer, beobachteter End-Zustand und Soll-Zustand voneinander abweichen, d.h. wenn das Wissen extern falsifiziert wurde.

Zur Definition eines Wissenssystems gehört, dass alles Wissen für das System verfügbar ist und auf Anwendbarkeit, bezogen auf die momentane Ist-Situation, geprüft wird. Allgemein wird anwendbares Wissen durch den übereinstimmenden Ist-Zustand und ein Verhalten definiert, das diesen Ist-Zustand als Input verwendet, unabhängig von einem End- bzw. Soll-Zustand, d.h. des Outputs, was durch einen Platzhalter „$$“ in der Wissensstruktur V(I,$$) beschrieben werden kann. Rational anwendbare Wissensstrukturen ergeben sich durch die Berücksichtigung des angestrebten Soll-Zustandes, da Strukturen V(I,$$) existieren können, die einen Output-Zustand besitzen, der weit von dem angestrebten Soll-Zustand entfernt liegt, bzw. als dessen Negation interpretierbar ist. Rational anwendbare Wissensstrukturen bezeichnen die Teilmenge der anwendbaren Wissensstrukturen, deren Output-Zustand hinreichend ähnlich zu dem angestrebten Soll-Zustand ist, wobei eine adäquate Ähnlichkeitsfunktion operationalisiert werden muss.

In Newell (1990[233]) wird unter Wissenssuche die Suche nach anwendbaren bzw. rational anwendbaren Wissensstrukturen im Langzeitspeicher des Wissenssystems verstanden. Wird ein Wissenssystem als Idealsystem betrachtet, so müssten alle rational anwendbaren Wissensstrukturen in den Kurzzeitspeicher kopiert werden, gefolgt von Selektionsoperationen, die in der Regel genau eine dieser Wissensstrukturen auswählt, da das Wissenssystem per Definition nicht mehrere Verhaltensmuster gleichzeitig durchführen kann, bzw. nicht mehrere bezogen auf ein Aktor-Subsystem. Im Kontext der Wissenssuche wird ein Problem definiert als das fehlende Vorliegen von rational anwendbaren Wissensstrukturen nach einer exhaustiven Suche im Langzeitspeicher. Diese Situation erfordert eine Problemsuche, in der Wissensstrukturen erzeugt und getestet (Generate & Test) werden, um rational anwendbare Wissensstrukturen zu erhalten.

Bei der Problemsuche werden kombinatorische Suchräume gebildet, von denen angenommen werden kann, dass einzelne Punkte des Suchraumes rational anwendbare Wissensstrukturen, bezogen auf die momentane Situation, repräsentieren. Ziel der Problemsuche ist es, ausgehend von einem Anfangs-Zustand in dem Raum, solche Punkte mit Hilfe geeigneter Suchverfahren zu finden. Im Rahmen eines Suchverfahrens werden bestimmte Punkte als Hypothesen ausgewählt und evaluiert, d.h. es wird getestet, ob die Wissensstruktur, die von einem solchen Punkt repräsentiert wird, ein rational anwendbares Wissen ist. In Newell (1990[233]) werden sequentielle Suchverfahren vorausgesetzt, doch es können auch parallele Suchverfahren betrachtet werden, bei denen die Situation eintreten kann, dass eine Menge von mehreren anwendbaren Wissensstrukturen gefunden wird, sodass durch eine Ähnlichkeitsfunktion zwischen den End-Zuständen der gefundenen Funktionen V(I,E) und dem Soll-Zustand aus V(I,S) eine Selektion erfolgt, und die Wissensstruktur mit der größten Ähnlichkeit zum Soll-Zustand ausgewählt wird. Wird der Suchprozess in dem Suchraum durch ein Abbruchkriterium beendet, d.h. wird eine rational anwendbare Wissensstruktur gefunden, so wird die Problemlösung durch einen Prozess namens Chunking in Form eines Wissenselementes dokumentiert und im Langzeitgedächtnis gespeichert (Newell (1990[233])), sodass bei dem erneuten Auftreten des Ist- und Soll-Zustandes eine anwendbare Wissensstruktur durch eine Wissenssuche anstatt eine Problemsuche gefunden werden kann.

Bei einem nicht perfekt intelligenten Repräsentationssystem wird keine exhaustive Suche nach rational anwendbaren Wissensstrukturen durchgeführt, sodass die Notwendigkeit einer Problemsuche diagnostiziert werden kann, auch wenn objektiv rational anwendbares Wissen vorliegt, das jedoch nicht gefunden wurde. Im Kontext der Echt-Zeit-Anforderung muss das System den Aufwand einer Wissenssuche gegenüber einer Problemsuche abwägen, d.h. den Aufwand möglicherweise vorhandenes eigenes Wissen zu suchen, gegenüber dem (kreativen) Nachdenken, als Erzeugung und interne Testung von neuem Wissen, was als Preparation-Deliberation-Trade-Off bezeichnet wird (Newell (1990[233])).

Wird das IR im Kontext eines Problemlösungsprozesses betrachtet, so wird es in den Kontext der Problemsuche gesetzt. Die Problemsuche eines Agenten kann beliebig lange fortgeführt werden, ohne dass er rational anwendbares Wissen durch Nachdenken erzeugen kann. Eine rationale Strategie ist die Einführung einer individuellen Schwellenwertfunktion, die mit der Zeit bzw. der Anzahl der missglückten Versuche, das Problem zu lösen, ansteigt, bis ein Schwellenwert (Frustration) erreicht ist. Unter der Voraussetzung, dass außerhalb des Agenten eine kulturelle Infrastruktur existiert, wird bei Erreichen des Schwellenwertes ein externes Informationsbedürfnis manifest.

Im Rahmen dieser Schwellenwertfunktion wird die Chance bewertet, dass eine rational anwendbare Wissensstruktur durch eigene kognitive Operationen erzeugt werden kann, unter der Kenntnis der bisherigen Fehlversuche. Neben der Chance muss der erwartete Aufwand hinzugezogen werden, um eine rational anwendbare Wissensstruktur zu erzeugen. Die Schwellenwertfunktion kann somit durch die rationale Analyse kognitiver Prozesse formuliert werden (Anderson (1990[8], 1991[9], 1993[10]), d.h. wenn die Chance p eine rational anwendbare Wissensstruktur zu erreichen mal eine Bewertung G des kognitiven Ertrages, der durch diese Struktur dem Agenten zufließt, kleiner den erwarteten kognitiven Kosten C ist, so ist die Bedingung für einen Abbruch der Problemsuche erfüllt. D.h. wenn $p * G < C$ wird, ist ein Abbruch der Problemsuche rational. Dies bedeutet jedoch nicht notwendig, dass die Problemsuche durch den Agenten aufgegeben wird, sondern nur, dass die direkte Suche des Agenten mit seinen unmittelbar zur Verfügung stehenden kognitiven Mitteln abgebrochen wird. Gleichzeitig ist damit die Bedingung der Formulierung eines externen Informationsbedürfnisses erfüllt. Formuliert werden kann ein externes Informationsbedürfnis als Anfrage an ein anderes Wissenssystem wie einen Experten oder ein Informationssystem. Formuliert wird dieses Informationsbedürfnis mit Hilfe der Wissensstrukturen, die während der vorangegangenen Wissenssuche als anwendbare aber nicht rational anwendbare Strukturen nachgewiesen wurden, sowie die Wissensstrukturen, die während des bisherigen Problemlösungsprozesses erzeugt wurden.

Das Experiment als Frage des Wissenssystems an seine externe Umgebung gehört ebenfalls zu den Methoden, mit denen extern Informationen erlangt werden können, die durch Repräsentation und Integration in die vorhandene Wissensstrukturen zu Wissen werden. Damit wird ein Bezug zu aktivem Lernen in Form des Optimal Experiment Design (Fedorov (1972[113]), Cohn (1994[71], 1995[73]), Cohn et al. (1994[72], 1995[74])) hergestellt, da durch die Formulierung von Experimenten das Wissenssystem Einfluss auf die Stimuli besitzt, die es lernen wird. Wissenssysteme, die aktives Lernen beherrschen, d.h. die ihre eigenen internen Modelle analysieren und daraus ableiten, was sie als nächstes lernen wollen, können als reflektive Wissenssysteme (Beyer & Smieja (1994[45]), Mühlenbein (1994[229], 1995[230])) bezeichnet werden. Sie können Stimuli auswählen, von denen sie annehmen, dass mit dem Lernen dieser

Stimuli allgemein die Performance des betrachteten, internen Modells maximiert wird, bzw. spezieller die Unsicherheit operationalisiert durch die Output-Varianz, minimiert wird. Die Auswahl der Stimuli kann auch auf die Maximierung des Wissenszuwachses abzielen, oder auf die Maximierung der Chance, rational anwendbares Wissen zu erzeugen.

Mit dem gleichen (pG-C)-Instrumentarium kann ebenfalls die mehrstufige Interaktion zwischen Agent und einem IS (bzw. Experten) beschrieben werden, die im IR-Kontext als Relevanz-Feedback bezeichnet wird (siehe z.B. Borodin et al. (1971[56]), Brauen (1971[57]), Friedman et al. (1971[127]), Panyr (1987b[252]) und Abschnitt 3.9)). Nachdem eine Dokumentliste vom IRS nachgewiesen wurde, integriert der Agent im Idealfall das darin enthaltene Wissen bzw. eine Teilstruktur des Wissens in seine vorhandene Wissensgesamtstruktur, d.h. nach Newell (1990[233]) würde das Langzeitgedächtnis als Gesamtmenge der prozeduralen Wissenseinheiten in Form von Regeln oder Funktionen V(I,E) um die extrahierten Wissenselemente aus den nachgewiesenen Dokumenten erweitert werden. Effizienter wäre jedoch das Verbleiben der neuen Wissenselemente im Kurzzeitspeicher, da somit direkt die Anwendbarkeit bzw. rationale Anwendbarkeit des Wissens bezüglich des zu lösenden Problems und somit des angestrebten Soll-Zustandes an dem internen Modell geprüft werden kann.

Die Relevanzbewertung bezieht sich auf die Bewertung des extrahierten Wissens, das als Teilmenge des repräsentierten Wissens in den nachgewiesenen Dokumenten betrachtet wird. D.h. Relevanzbewertungen durch den Agenten beziehen sich nicht auf den vollständigen Inhalt eines Dokumentes, wobei die Wissensteile, auf die sich die Relevanzbewertung bezieht, dem IRS unbekannt bleiben. Eine explizite Aufdeckung des extrahierten Teilwissens kann nur dann erfolgen, wenn der Agent seine Interaktion mit dem IRS erweitert, indem er z.B. Teile des Dokumentes als relevant kennzeichnet, wodurch man zu Formen des Passagen-Retrievals gelangt (siehe z.B. Kaszkiel & Zobel (1997[180])). Alternativen wären Formen der Reformulierung bzw. der Nacherzählung des extrahierten Wissens durch den Agenten, gefolgt von der Indexierung des auf diese Weise entstandenen Textes durch das IRS. Ein effektives Vorgehen besteht darin, dem Dokumentvektor dieses Textes die Relevanzbewertung zuzuordnen, anstatt dem ursprünglichen Dokumentvektor. Aufgrund des erhöhten Aufwandes dieser Interaktion ist diese Vorgehensweise jedoch bei menschlichen Agenten kaum anwendbar, da die kognitiven Kosten C dadurch erheblich steigen, und somit ein Interaktionsabbruch früher erfolgt. Doch selbst bei einer Anwendung ist sie nicht fehlerfrei, da der menschliche Agent sein Wissen in einer Sprache formulieren muss, d.h. es werden Repräsentationselemente aus dem Problemlösungsareal auf Repräsentationsobjekte aus dem Sprachsynthesareal abgebildet, sodass die Adäquatheit von der Anzahl der verfügbaren Objekte in den beiden Arealen und der Eigenschaft der Abbildung abhängt.

Der zweite Aspekt der Relevanzbewertung bezieht sich auf die Frage, welchen Beitrag das extrahierte Wissen zur Lösung des Problems geleistet hat. Die Beantwortung dieser Frage setzt jedoch die Lösung des Problems, die Rekonstruktion des Lösungsweges, sowie die Analyse der Beiträge des externen Wissens im Gegensatz zu den Beiträgen der vorhandenen, internen Wissenselementen voraus. Von keinem dieser drei Faktoren kann jedoch während eines Problemlösungsprozesses ausgegangen werden. Somit kann eine Relevanzbewertung durch einen menschlichen Agenten ebenfalls nur als Schätzung betrachtet werden. Eine detailliertere Beschreibung der Relevanzproblematik im Kontext der Problemlösung bzw. Problemsuche findet sich in Abschnitt 3.9.1) (siehe z.B. auch Panyr (1986b[250])).

1.6) Polyrepräsentation

Unter Redundanz wird in einem Repräsentationssystem (Newell (1990[233])) die mehrfache und jeweils gleiche Repräsentation von Informationsobjekten verstanden. Kontrollierte Redundanz wird z.B. in Datenbanksystemen eingesetzt, um eine Verbesserung von Zugriffszeiten zu erreichen, oder um eine höhere Ausfallsicherheit zu erzeugen, wodurch der Speicheraufwand jedoch steigt, d.h. eine Verbesserung der Zeiteffizienz wird auf Kosten der Speichereffizienz durchgeführt.

Im Gegensatz zur Redundanz soll unter Polyrepräsentation die mehrfache und jeweils unterschiedliche Repräsentation eines Informationsobjektes verstanden werden. Es wird dadurch eine Effektivitätssteigerung angestrebt, da bei jeder der Einzelrepräsentationen unterschiedliche Perspektiven fokussiert werden können, die durch ein einzelnes Modell nicht bzw. nicht in diesem Umfang repräsentierbar sind. Weiterhin soll damit im Kontext eines IRS Unsicherheiten bezüglich der Repräsentationen und die Diversität von Agenten modellierbar und somit handhabbar werden.

In natürlichen, physikalischen Repräsentationssystemen sind Redundanz und Polyrepräsentation verknüpft, bzw. man kann Redundanz durch Polyrepräsentation substituieren. Betrachtet werden soll die Redundanz, bezogen auf die Sensoren einer Modalität, d.h. es existieren mehrere Sensoren des gleichen Typs, die sich notwendig an unterschiedlichen, räumlichen Positionen des physikalischen Systems befinden. Dadurch kommt es natürlicherweise zu einer Polyrepräsentation, da die einzelnen Sensoren durch ein externes Ereignis unterschiedlich in ihrer Intensität und zu unterschiedlichen Zeitpunkten aktiviert werden. Die Redundanz der Sensoren bewirkt somit eine mehrfache, unterschiedliche Repräsentation bei dem gleichen Repräsentationsverfahren, was durch den gleichen Sensortyp gegeben ist, sodass dies als Intraparadigmen-Polyrepräsentation verstanden werden kann. Aus diesen räumlichen, intensitätsmäßigen und temporären Unterschieden können in nachfolgenden Schritten komplexe Eigenschaften durch Repräsentationen höherer Ordnung konstruiert bzw. rekonstruiert werden, indem die Einzelrepräsentationen durch unterschiedliche Verfahren integriert werden. Ein Beispiel ist die Rekonstruktion von Entfernungsdaten aus den beiden unterschiedlichen Winkeln, mit denen ein Objekt von den beiden Augen eines menschlichen Agenten wahrgenommen wird, die hier als redundante Sensoren verstanden werden.

Bei einem Ausfall einzelner Sensoren kann in Abhängigkeit von dem Grad der Redundanz die Gesamtsystemleistung erhalten bleiben. Wie im Bereich der Neuronalen Netze (Brause (1991[58]), Ritter et al. (1991[282]), Schöneburg et al. (1990[302]), Bachelier (1998a[15])) gilt hier eine „gracefull degradation“, d.h. bei zunehmendem Ausfall von Sensoren nimmt die Systemleistung über einen längeren Zeitraum nicht oder nur wenig ab, bis ein Schwellenwert der Schädigung erreicht ist, nach dem die Systemleistung rapide abnimmt.

In natürlichen Repräsentationssystemen, die durch unterschiedliche Typen von Sensoren an ihre Umwelt gekoppelt sind, ist Polyrepräsentation im Zusammenhang mit der Multimodalität der sensorischen Verarbeitung eine gängige Strategie, da Objekte zunächst modalitätsspezifisch in eigenen Arealen repräsentiert werden. Durch die unterschiedlichen Eigenschaften der Sensoren bilden diese Repräsentationen unterschiedliche Eigenschaften eines externen Objektes ab. Diese primäre Polyrepräsentation wird erweitert um die Möglichkeiten der Kombination von Einzelrepräsentationen, indem mindestens zwei modalitäts-

spezifische Areale in ein anderes Areal projizieren, in dem eine multi-modale Repräsentation gebildet wird. Auf diese Weise lassen sich multi-modale Repräsentationen zweiter und höherer Ordnung erzeugen, bis bei n primären Polyrepräsentationen eine Repräsentation n'ter Ordnung gebildet wird. Die Gesamtheit aller Repräsentationen erster bis n'ter Ordnung bilden wiederum eine Polyrepräsentation, da jeweils unterschiedliche Aspekte und Eigenschaften fokussiert werden können.

Die Polyrepräsentation steht im Zusammenhang mit dem Bemühen, unterschiedliche Formen von Unsicherheit und Vagheit beim Erkennen und Repräsentieren zu beherrschen. Entsprechend der grundlegenden Aufgabenstellung des maschinellen Lernens als Induktion einer Funktion (siehe z.B. Gama (1999:30f[141])), kann im allgemeinen Fall nicht erwartet werden, dass mit einer endlichen Menge an Lernstimuli, die z.B. durch Sensoren gewonnen werden, ein vollständiges Modell eines externen Phänomens gebildet werden kann. Dies ergibt sich aus der Unterdeterminiertheit einer endlichen Lernmenge bezogen auf die überabzählbar große Menge von Funktionen, die ein Phänomen hinreichend, d.h. bezogen auf bestimmte Anforderungen, beschreiben. Unabhängig wie groß die zu lernende Stimulusmenge ist, es bleibt immer ein Maß von Unsicherheit, ob das gelernte Modell bei noch unbekannten Stimuli in gleicher Weise die zugrunde liegende Funktion approximieren kann. Das gleiche Ergebnis liefert der Lösungsvorschlag für das Induktionsproblem in einer offenen Welt durch Karl Popper (Popper (1994[262], 1995[263]), siehe auch Mühlenbein (1994[229], 1995[230])).

Durch die Festlegung der Art von verwendeten Sensoren und von Repräsentationssprachen wird jeweils eine Form von verfahrenseigenem Fehler (Bias, siehe Abschnitt 2.3)) eingeführt, der durch dieses Verfahren selbst nicht reduziert werden kann. Eine Strategie besteht darin, verschiedene Verfahren, d.h. Aquisitionsfunktionen wie Sensoren oder Repräsentationssprachen, mit unterschiedlichen Formen von Fehlern zu verwenden, die in einem nachfolgenden Schritt aggregiert werden, wodurch Combining-Model-Systeme eingeführt werden (Wolpert (1990[365], 1993[367]), Brodley (1994[59]), Drucker et al. (1994[98]), Freund (1995[124]), Freund & Schapire (1995[125], 1996[126]), Kong & Dietterich (1995[186]), Drucker & Cortes (1996[99]), Quinlan (1993[267], 1996[268]), Skalak (1996[313]), Merz (1998[216]), Gama (1999[141]), Witten & Ting (1999[364])). Die Unterschiedlichkeit ist das Hauptargument für eine Polyrepräsentation im Rahmen der Modellbildung beim maschinellen Lernen, da jede der Einzelrepräsentationen eine Fehler- und Unsicherheitsverteilung mit eigenen Eigenschaften besitzt. Eine Analyse der Modelle und ihrer Fehler eröffnet die Möglichkeit durch eine Aggregation der Modelle den Gesamtfehler im Vergleich zu einzelnen Werten zu verkleinern.

1.6.1) Polyrepräsentation in IS und IRS

Entsprechend der Quin-Tupel-Definition (A, W, Q, I, E) besteht ein IS aus einer Anzahl von Datenstrukturen (W, Q, E) und aus einer Anzahl von Funktionen (A,I). Von einer Polyrepräsentation in einem IS kann dann gesprochen werden, wenn mindestens eine der IS-Datenstrukturen (W, Q, E) bzw. eine der IS-Funktionen (A, I) polyrepräsentiert sind. Analog zur Definition Polyrepräsentation in einem IS liegt eine Polyrepräsentation in einem IRS vor, wenn mindestens eine der IRS-Repräsentationsarten (D^t, DVM^t, F^t, $DVM(q_i^t)$, ...) bzw. eine der IRS-Funktionen ($A_{IR(D)}$, $A_{IR(Q)}$, ret(.)) polyrepräsentiert sind.

Eine mehrfache Polyrepräsentation von Komponenten aus dem Tupel (D^t, DVM^t, F^t, $DVM(q_i^t)$, ..., $A_{IR(D)}$, $A_{IR(Q)}$, ret(.)) führt zu einer kombinatorischen Explosion beim Retrieval, sodass im Rahmen dieser Arbeit schwerpunktmäßig eine Polyrepräsentation jeweils bezüglich genau einer Komponente betrachtet wird. D.h. wird eine Polyrepräsentation der Dokumentvektorenmengen unterstellt, d.h. eine Polyrepräsentation von $A_{IR(D)}$, so wird eine Monorepräsentation der Query-Indexierungsfunktion und der Retrievalfunktion verwendet. Umgekehrt wird von einer Monorepräsentation der Dokumentvektorenmengen ausgegangen, wenn die Fälle der Polyrepräsentation von $A_{IR(Q)}$ oder der Retrievalfunktion untersucht werden sollen.

Eine Polyrepräsentation der IS- oder IRS-Funktionen dient zur Erzeugung der Polyrepräsentation einer Datenstruktur, d.h. eine Polyrepräsentation z.B. der Query-Indexierungsfunktion $A_{IR(Q)}$ beim Vorliegen einer Query-Monorepräsentation erzeugt eine Queryvektor-Polyrepräsentation. Demgegenüber kann eine Polyrepräsentation einer Datenstruktur auch ohne eine Polyrepräsentation der IS-Funktionen erzeugt werden, indem z.B. stochastische Operationen wie Resampling-Operationen, (insbesondere Bootstrap-Operationen; siehe Mammen (1992[206]), Efron & Tibshirani (1993[105]), Hijorth (1994[163]), Shao & Tu (1995[309])) angewendet werden. Dies geschieht ebenfalls durch eine Funktions-Polyrepräsentation, da eine Resampling-Operation ebenfalls als Funktion definiert wird, doch diese Funktionen sind nicht Teil des regulären IR-Prozesses, d.h. nicht Teil der IRS-Definition als n-Tupel. Sie werden als externe Funktions-Polyrepräsentation bezeichnet, sodass allgemein gesagt werden kann, dass Funktions-Polyrepräsentation Datenstruktur-Polyrepräsentationen erzeugen, unabhängig ob die Funktionen intern oder extern bezüglich des IS bzw. IRS sind. Externe Funktions-Polyrepräsentation wird schwerpunktmäßig bei der Polyrepräsentation von Relevanz-Approximationsmodellen betrachtet, indem Resampling-Operationen auf eine Stützpunktmenge angewendet werden (siehe Kapitel 4)).

1.6.2) Inter- und Intraparadigmen-Polyrepräsentation

Interparadigmen-Polyrepräsentation (Ingwersen (1992[171], 1994[172])) bezeichnet die Verwendung unterschiedlicher Repräsentationssprachen bzw. Repräsentationsarten, um ein Objekt mehrfach zu repräsentieren. Beispielsweise liegt eine Query-Interparadigmen-Polyrepräsentation vor, wenn eine Query Q_i als Zeichensequenz durch verschiedene Indexierungsfunktionen auf unterschiedliche Repräsentationssprachen, wie vektorraumbasierte, probabilistische, linguistische, logische, ... abgebildet wird. Intraparadigmen-Polyrepräsentation bezeichnet die mehrfache, unterschiedliche Repräsentation eines Objektes in der gleichen Repräsentationssprache.

Intra- und Interparadigmen-Polyrepräsentation steht in direkter Beziehung zu den Sensoren in natürlichen Verhaltens- und Repräsentationssystemen. Intraparadigmen-Polyrepräsentation bezieht sich auf den gleichen Typ von Sensor, der mehrfach vorhanden ist und jeweils eine eigene Repräsentation erzeugt, die in der gleichen Repräsentationssprache vorliegt, jedoch durch zusätzlich kodierte, räumliche und zeitliche Aspekte unterschiedlich sind. Demgegenüber bezieht sich die Interparadigmen-Polyrepräsentation auf mehrere unterschiedliche Sensortypen, die unterschiedliche Arten physikalischer Wirkung eines Phänomens repräsentieren können, und dies in unterschiedlichen Repräsentationssprachen durchführen.

Innerhalb einer Intraparadigmen-Polyrepräsentation kann unterschieden werden zwischen einer Datenstruktur-Polyrepräsentation und einer Methoden-Polyrepräsenatation. Bei einer Datenstruktur-Polyrepräsentation wird eine Menge unterschiedlicher Repräsentationen des gleichen Typs für ein Informationsobjekt wie eine Query, ein Queryvektor, ein Dokument und ein Dokumentvektor erzeugt. Die Erzeugung kann durch das wiederholte Anwenden genau einer Methode mit unterschiedlichen Parametern erfolgen, oder durch die Anwendung einer Menge von Methoden, wodurch man zu einer Methoden-Polyrepräsentation gelangt. D.h. eine Methoden-Polyrepräsentation ist ein Weg zu einer Datenstruktur-Polyrepräsentation zu gelangen. Methoden im IR werden durch Funktionen wie Indexierung, Retrieval, Clusterung oder Modifikationen von Datenstrukturen im Rahmen des Relevanz-Feedbacks beschrieben, sodass eine Polyrepräsentation der entsprechenden Funktionen zu einer Polyrepräsentation der zugrunde liegenden Datenstrukturen führen.

Es soll festgelegt werden, dass von einem Polyrepräsentations-IRS gesprochen wird, wenn an mindestens einer Stelle im IR-Prozess eine Datenstruktur-Polyrepräsentation erzeugt wird, die im weiteren IR-Prozess genutzt wird.

Dokumente, Queries und Merkmale wie Terme oder n-Gramme, werden im Rahmen dieser Arbeit allgemein als endliche Zeichensequenz bezüglich eines endlichen Alphabetes betrachtet, d.h. als diskrete Datenstrukturen. Demgegenüber stehen Datenstrukturen in einem Kontinuum wie reelle Vektoren, auf die diskrete Datenstrukturen im Rahmen der Indexierung abgebildet werden.

Polyrepräsentation kann auf diskrete wie kontinuierliche Datenstrukturen angewendet werden, wobei die Polyrepräsentation durch externe Festlegungen oder durch stochastische Prozesse erzeugt werden kann. Eine externe Festlegung wäre z.B. die Auswahl einer Menge von Indexierungsfunktionen mit unterschiedlichen Ähnlichkeits- und Gewichtungsfunktionen. Einen Überblick über Indexierungsfunktionen, die im IR verwendet werden, sowie die kombinatorische Verknüpfung der darin enthaltenen Ähnlichkeits- und Gewichtungsfunktionen gibt Zobel & Moffat (1998[376]). Demgegenüber stehen stochastische Prozesse, die jedoch ebenfalls nicht gänzlich ohne externe Festlegungen bezüglich der verwendeten Parameter auskommen. Stochastische Prozesse bezogen auf eine diskrete Datenstruktur eines Dokumentes oder einer Query sind Resampling-Operationen wie z.B. Bootstrap-Operationen (Mammen (1992[206]), Efron & Tibshirani (1993[105]), Hijorth (1994[163]), Shao & Tu (1995[309]); siehe Abschnitt 2.2)), bei denen künstliche Dokumente bzw. Queries durch gleichverteilte Ziehungen mit Zurücklegen von Teilstrukturen wie Terme oder n-Gramm-Sequenzen aus der ursprünglichen Datenstruktur erzeugt werden. Stochastische Prozesse bezogen auf eine kontinuierliche Datenstruktur eines Dokument- oder Queryvektors können durch die additive Überlagerung mit einer Menge von Zufallsvektoren erzeugt werden.

Eine Resampling-Operation und eine Überlagerung mit Zufallsvektoren sind nicht Teil des regulären IR-Prozesses, sodass diese Datenstruktur-Polyrepräsentation als externes Verfahren betrachtet werden muss. Demgegenüber ist z.B. die Erzeugung einer Menge von Indexierungsfunktionen durch einen stochastischen Prozess eine Methoden-Polyrepräsentation, die zu einer Datenstruktur-Polyrepräsentation führt, indem die Indexierungsfunktionen auf die Monorepräsentationen, Dokument oder Query parallel und unabhängig angewendet werden. Da die Indexierung ein Teil des regulären IR-Prozesses ist, ist die

Erzeugung einer Menge von Dokumentvektoren aus einem Dokument durch eine Menge von Indexierungsfunktionen ein internes Verfahren zur Erzeugung einer Datenstruktur-Polyrepräsentation.

Fasst man dies zusammen, so ist die Anwendung von Methoden, die Teil des regulären IR-Prozesses sind, die einzigen Möglichkeiten, eine Datenstruktur-Polyrepräsentation zu erzeugen, die als internes Verfahren, bezogen auf den IR-Prozess, bezeichnet werden kann.

1.6.3) Gründe für das Vektorraummodell als Intraparadigmen-Polyrepräsentation

Menschliche Kognition operationalisiert in einer hierarchischen Struktur eine Vielzahl von Repräsentationsformen wie Mikronemes, Neuronale Netze, K-Lines, Semantische Netze, Frames, Transframes, Scripts (nach Minsky (1986[220], 2000[221])). Repräsentationen einer Ebene müssen in Repräsentationen der darunter und darüberliegenden Ebene transformiert werden können, ohne dass beweisbar wäre, dass die entsprechende Abbildung bijektiv ist, d.h. dass alle Aspekte einer Repräsentationsform in die andere eineindeutig überführbar ist, und dass die Abbildung umkehrbar ist. Faktisch fehlen Erfahrungen mit künstlichen Repräsentationssystemen, die unterschiedliche Repräsentationsformen umfassen und integrieren. Wie problematisch es ist, verschiedene Repräsentationen ineinander zu transformieren, zeigt sich z.B. bei den Versuchen räumliche und zeitliche Aspekte mit Mitteln der Prädikatenlogik zu repräsentieren.

In dieser Arbeit wird eine Intraparadigmen-Polyrepräsentation verwendet, da die funktionelle Äquivalenz verschiedener IR-Ansätze gezeigt werden konnte, was zur Folge hat, dass eine Interparadigmen-Polyrepräsentation dieser Ansätze keine prinzipiell mächtigere Repräsentation ergibt als die Interparadigmen-Polyrepräsentation bezüglich eines dieser Ansätze. Kwok (1995[194]) konnte die Abbildung des probabilistischen IR-Ansatzes (Robertson (1977a[283], b[284], c[285], 1979[286]), Panyr (1986c[249]), Harter (1975[159])) auf ein Feedforward-Netz (Brause (1991[58]), Ritter et al. (1991[282]), Schöneburg et al. (1990[302])) zeigen, und in Bachelier (1995[14]) wurde die Abbildung des IR-Vektorraummodells (Salton (1971[293]), Panyr (1987a[251])) auf ein Feedforward-Netz beschrieben. Somit folgt die funktionale Äquivalenz des probabilistischen IR-Ansatzes, des IR-Vektorraummodells und konnektionistischer IR-Ansätze auf der Basis von Feedforward-Netzen.

Das Vektorraummodell besitzt Repräsentationsformen, die in unüberwachten Lernverfahren mit erweiterten SOMs (Ritter et al. (1991[282]), Kohonen (1989[184], 1995[185]), Bachelier (1998a[15])) direkt verwendet werden können. In Bachelier (1995[14]) wurde damit Indexierung und Clusterung durch SOMs beschrieben, während in der vorliegenden Arbeit komplexe Retrievalstrategien beschrieben werden, in denen erweiterte SOMs zu überwachten Lernverfahren gemäß einem gemeinsamen Framework von unüberwachtem und überwachtem Lernen ausgebaut werden. Auf diese Weise gelangt man zu Relevanz-Approximationsmodellen (siehe Abschnitt 1.7) und Kapitel 4)). Entsprechend dieses Planes ist es sinnvoll, sich auf das Vektorraummodell zu beschränken und die probabilistischen IR-Ansätze auszuklammern.

Andere Ansätze des IR wie computerlinguistische Ansätze mit Lexika und Thesauri, sowie logikbasierte Ansätze mit domänspezifischen Wissensbasen, lassen sich demgegenüber nicht bzw. nicht direkt aufeinander abbilden, da im Vektorraummodell kein externes Weltwissen und deren Formalisierung in Wissensbasen notwendig ist. Weltwissen kann jedoch in vielfältiger Weise in das Vektorraummodell integriert werden, z.B. durch die Auswahl von Indexierungsmerkmalen und deren Vernetzung in einem Assoziationsgraphen, mit dem z.B. Queryvektor-Modifikationen (Queryvektor-Expansion) durchführbar werden. Eine Analogie zwischen symbolischen und subsymbolischen Systemen erscheint hier möglich, indem Ansätze mit Hilfe eines Thesaurus und Wissensbasen als symbolisch und das Vektorraummodell als subsymbolischer Ansatz interpretiert wird. Letzteres gilt insbesondere dann, wenn subsymbolische Merkmale wie kurze n-Gramme als Repräsentationsgrundlage verwendet werden. Die Klärung der Beziehung von subsymbolischen und symbolischen Strukturen, sowie deren Integration in einer gemeinsamen Architektur im Kontext der kognitiven Modellierung und im Konnektionismus (Dorffner (1991[96])) wurde öfters versucht (siehe z.B. Paaß & Kurfeß (1993[246]), Sun & Bookman (1995[327])), bislang jedoch ohne zufriedenstellende Ergebnisse. Aufgrund dieser ungeklärten Grundlagen sollen symbolische Ansätze im Rahmen dieser Arbeit ausgeklammert werden.

1.6.4) Gründe für Polyrepräsentation

1.6.4.1) Beschränkung endlicher Lernmengen

Entsprechend der grundlegenden Aufgabenstellung des maschinellen Lernens als Induktion einer Funktion (siehe z.B. Gama (1999:30f[141])) kann im allgemeinen Fall nicht erwartet werden, dass mit einer endlichen Menge an Lernstimuli, die z.B. durch Sensoren gewonnen werden, ein vollständiges Modell eines externen Phänomens gebildet werden kann. Dies ergibt sich aus der Unterdeterminiertheit einer endlichen Lernmenge, bezogen auf möglicherweise überabzählbar große Menge von Funktionen, die ein Phänomen hinreichend, d.h. bezogen auf bestimmte Anforderungen, beschreiben. Unabhängig wie groß die zu lernende Stimulusmenge ist, es bleibt immer ein Maß von Unsicherheit, ob das gelernte Modell bei noch unbekannten Stimuli in gleicher Weise die zugrunde liegende Funktion approximieren kann. Zum gleichen Ergebnis gelangt der Lösungsvorschlag für das Induktionsproblem in einer offenen Welt durch Karl Popper (Popper (1994[262], 1995[263]), siehe auch Mühlenbein (1994[229], 1995[230])).

Resampling-Verfahren, angewendet auf eine begrenzte Menge von Lernstimuli, haben sich als nützlich erwiesen, um eine Modellmenge zu erzeugen (Modell-Polyrepräsentation), wodurch das Potential besteht, Fehler und Unsicherheiten einzelner Modelle und der gesamten Modellmenge zu quantifizieren. Eine Modell-Polyrepräsentation eröffnet zudem die Möglichkeit rationaler Aggregation von Modellen, indem zunächst die Fähigkeiten und Schwächen einzelner Modelle explizit ermittelt werden. Architekturen, die explizite (Meta-)Modelle ihrer Fähigkeiten und Schwächen erstellen, und diese zur Erreichung von Zielen nutzen, werden als reflektive Architekturen bezeichnet. Damit lassen sich Aggregations-Architekturen erzeugen (Combining-Models (Wolpert (1990[365], 1993[367]), Freund (1993[122], 1995[124]), Freund & Schapire (1995[125], 1996[126]), Quinlan (1993[267], 1996[268]), Skalak (1996[313]), Merz (1998[216]), Gama (1999[141]), Witten & Ting (1999[364])), Mixture-of-Experts (McLachlan & Basford (1987[214])), welche die Schwächen einzelner Modelle mit den Stärken anderer

Modelle kompensieren. Somit können Gesamtmodelle mit geringeren Fehlern und Unsicherheiten erzeugt werden.

Durch eine Modell-Polyrepräsentation wird die prinzipielle Problematik endlicher Lernmengen und das Induktionsproblem nicht gelöst, doch in Bezug konkreter Anwendungen wird die Problematik in der Weise gemildert, wie der Einsatz von Resamplingverfahren es bislang erlaubt.

Zur relativen Bedeutung von Search-NFL- und Representation-NFL-Theorem für die Modellbildung, d.h. induktives Lernen, wird bereits in Geman et al. (1992[142]) darauf hingewiesen, dass Modellbildung wesentlich mehr von der Wahl der Repräsentationssprache beeinflusst wird, als von der Wahl eines Suchverfahrens.

Bezogen auf die Theoriebildung in der Wissenschaft ist eine Modell-Polyrepräsentation der Normalfall. Popper (1994[262], 1995[263]) sieht die Theoriebildung als Versuchs-Irrtums-Prozess (siehe auch Generate & Test bei Newell (1990[233])) auf der Basis einer Menge alternativer, konkurrierender Modelle. Die Evolution einer Menge von Theorien (Theorie-Polyrepräsentation) durch die Kopplung an die offene Welt durch Experimente, könnte auf der Basis unterschiedlicher Combining-Modell-Architekturen simulierbar werden und tiefere Einblicke in die Erkenntnis- und Wisenschaftstheorie ermöglichen. Die Kopplung an die offene Welt kann dabei durch Verfahren des Optimal-Experiment-Designs erfolgen (Fedorov (1972[113]), Cohn (1994[71], 1995[73]), Cohn et al. (1994[72], 1995[74])), als einem Spezialfall des direkten, aktiven Lernens.

1.6.4.2) Fundamentale Beschränkung aller Repräsentationssprachen

Für die Polyrepräsentation existiert ein fundamentales Argument, das alle Systeme betrifft, die Repräsentationen und Suchstrategien verwenden (Repräsentations- und Wissenssysteme (Newell (1990[233])). Es wird unter den Bezeichnungen „No-Free-Lunch for Search (Search-NFL)“ (Wolpert (1992[366]), Macready & Wolpert (1996[205]), Wolpert & Macready (1996[370])) und „Conservation Law for Generalization Performance (CLGP)“ (Schaffer (1994[297]), Rao et al. (1995[271])) diskutiert. Beide Theoreme beschreiben, dass alle Verfahren, die nach Extrema einer Kostenfunktion suchen, die gleiche Gesamtperformance besitzen, wenn die Performance über alle möglichen Kostenfunktionen gemittelt wird. Wenn ein Verfahren A einem anderen Verfahren B bezüglich einer bestimmten Anzahl von Kostenfunktionen überlegen ist, dann existiert die gleiche Anzahl von anderen Kostenfunktionen, bei denen B A überlegen ist. CLGP bezieht sich auf induktives Lernen, wobei es besagt, dass positive Performance eines Lernverfahrens in einigen Lernsituationen ausgeglichen wird durch negative Performance in anderen Lernsituationen.

Aus dem Search-NFL-Theorem kann ein Representation-NFL-Theorem abgeleitet werden, d.h. es existiert keine Repräsentationssprache, die bezüglich allem Repräsentierbaren einer anderen Repräsentationssprache überlegen ist, bzw. alle Repräsentationssprachen besitzen die gleiche Gesamtperformance, wenn die Performance über allem Repräsentierbaren gemittelt wird. Die Operationalisierung von „besser-als“ kann dabei durch die Repräsentationsmächtigkeit durchgeführt werden, d.h. dasjenige, was in einer Repräsentationssprache beschrieben werden kann. Die Operationalisierung kann auch durch die

Adäquatheit bezogen auf eine Aufgabe, d.h. ein Suchprozess, beschrieben werden.
Die Beziehung der beiden Theoreme ergibt sich in natürlicherweise daraus, dass die im Search-NFL-Theorem betrachteten Kostenfunktionen in einer bestimmten Repräsentationssprache formuliert werden müssen, bevor eine Suche in der Funktionslandschaft nach Extrema durchgeführt wird. Eine Kostenfunktion kann in unterschiedlichen Repräsentationssprachen formuliert werden, was jeweils zu einer unterschiedlichen Funktionslandschaft führt, die bezogen auf ein Suchverfahren, unterschiedliche Eigenschaften besitzen kann. D.h. in unterschiedlichen Landschaften kann ein Suchverfahren es unterschiedlich schwer haben, Extrema zu finden. Es existieren eine Reihe von Maßen, mit denen Eigenschaften von Funktionslandschaften (Fitnesslandschaften) beschrieben werden können (siehe Lipsitch (1991[202]), Manderick et al. (1991[207]), Hordijk (1995[168]), Horn (1995[169]), Jones & Forrest (1995[177]), Stadler (1995[321])). Ziel dieser Untersuchungen ist es, anhand von Eigenschaften der Funktionslandschaft die Eignung eines Suchverfahrens zu prognostizieren, in dieser Funktionslandschaft Extrema zu finden.

Zur Bedeutung von Search-NFL- und Representation-NFL-Theorem für die Modellbildung, d.h. induktives Lernen, wird bereits in Geman et al. (1992[142]) darauf hingewiesen, dass Modellbildung wesentlich mehr von der Wahl der Repräsentationssprache beeinflusst wird, als von der Wahl eines Suchverfahrens.

Bezieht man sich auf die angesprochene Unterscheidung zwischen subsymbolisch und symbolischer Repräsentation, so erscheint es im Kontext der fundamentalen Beschränkung aller Repräsentationssprachen sinnvoll, bezogen auf ein Problem aus einer grundlegenden subsymbolischen Repräsentation durch Adaptionsprozesse eine Menge symbolischer Repräsentationen zu erzeugen (Symbol-Grounding-Polyrepräsentation).

Aus der Nicht-Überlegenheit einzelner Repräsentationssprachen und Suchverfahren folgen Strategien, die zu den Combining-Models (Wolpert (1990[365], 1993[367]), Brodley (1994[58]), Freund (1995[124]), Freund & Schapire (1995[125], 1996[126]), Drucker & Cortes (1996[99]), Quinlan (1993[267], 1996[268]), Skalak (1996[313]), Merz (1998[216]), Gama (1999[141]), Witten & Ting (1999[364])) eine direkte Beziehung besitzen:
1) Verwende in einer Repräsentationssprache eine Menge unterschiedlicher Suchverfahren und kombiniere deren Ergebnisse.
2) Verwende eine Menge von Repräsentationssprachen mit jeweils einem Suchverfahren und kombiniere deren Ergebnisse.
3) Verwende eine Menge von Repräsentationssprachen mit jeweils einer Menge von Suchverfahren und kombiniere deren Ergebnisse.

Bei diesen Strategien stellt sich jedoch die Frage nach der Auswahl der Repräsentationssprachen und Suchverfahren. In Rao et al. (1995[271]) wird gezeigt, dass CLGP nur in einem gleichverteilt zufälligen Universum und unter bestimmten Symmetrievoraussetzungen gilt. Da lokale Ausschnitte der Welt jedoch stark heterogen sein können, bedeutet dies, dass für das induktive Lernen innerhalb dieser Weltausschnitte bestimmte Suchverfahren und Repräsentationssprachen gegenüber anderen Verfahren vorteilhaft sind. Auch diese Erkenntnisse fordern wie die Betrachtung im Kontext eines Combining g-Models eine adaptive Auswahl von Repräsentationssprachen und Suchverfahren. D.h. eine Mehr-Ziel-Suche nach

geeigneten Repräsentationssprachen (Problemsuchraum (Newell (1990[233])), nach geeigneten Suchverfahren in den Problemsuchräumen und nach Extrema in der Funktionslandschaft, die durch ein Problemsuchraum als Inputraum in Verbindung mit einer Bewertungsfunktion gebildet werden. Suchaufgaben diesen Typs eignen sich besonders für Evolutions-Algorithmen mit einer hierarchischen Struktur (z.B. Multi-Populations-ES (Rechenberg (1994[277]), Jacob (1997[174]), Bachelier (1998b: 94ff[16])), bei denen auf der untersten Ebene Individuen vorliegen, auf der nächsten Ebene Populationen, gefolgt von Meta-Populationen wachsender Ordnung, wobei ein Objekt einer Ebene aus einer Menge von Objekten der darunter liegenden Ebene zusammengesetzt ist. Variations- und Selektionsoperatoren ziehen sich über die gesamte Hierarchie, d.h. aus einer Menge von z.B. Populationen auf der zweiten Ebene von unten werden neue Populationen durch Rekombination und Mutation erzeugt, und es werden Populationen durch Selektionsprozesse gelöscht.

1.6.4.3) Modellierung von Unsicherheit in Bezug zur Diversität der Agenten

Mit jeder der Komponenten aus IS = (A, W, Q, I, E) sind Unsicherheiten verbunden, wie z.B. Unsicherheit der adäquaten Indexierung von Informationsobjekten und Relevanzbewertungen oder Unsicherheit der adäquaten Definition einer Retrievalfunktion (s.a. Turtle & Croft (1997:191f)). Adäquatheit ist nur sinnvoll in Bezug zu konkreten Aufgabenstellungen zu sehen, was im Kontext von IS die Interaktion mit einem konkreten Agenten und seinen Informationsbedürfnissen und Relevanzbewertungen bedeutet. Die Unsicherheit der Adäquatheit ist eine Funktion der Diversität der Grundgesamtheit von Agenten, die als potentielle IS-Nutzer gelten. Konventionelle IS-Architekturen, d.h. Architekturen mit einer Monorepräsentation bezüglich aller IS-Komponenten, gehen von einem Agententyp aus, der als Mittelwert der Agentenverteilung betrachtet wird. Bei einer heterogenen Agenten-Grundgesamtheit, d.h. einer Verteilung mit einer großen Varianz ist diese Vorgehensweise jedoch nicht mehr adäquat. Eine Möglichkeit heterogene Agentenverteilungen zu bewältigen sind adaptive IS, wobei adaptiv allgemein bedeutet, dass das IS seine internen Repräsentationen entsprechend externen Anforderungen reversibel modifizieren kann. D.h. ausgehend von einem Basis-IS werden individuelle, agentenspezifische IS durch die Agent-IS-Interaktion erzeugt. Damit wird eine Beziehung zwischen Polyrepräsentation und Adaption deutlich, da die Menge der erzeugten, individuellen Repräsentationen eine Polyrepräsentation darstellen, wenn sie gespeichert werden und wenn sie von dem IS für bestimmte Ziele operationalisiert werden können.

Es existieren deutliche Anzeichen für eine Zunahme der mentalen Diversität der Agenten, verursacht durch eine Reihe sozialer Prozesse. Jedes Individuum besitzt eine eigene Entwicklungsgeschichte, und die fortschreitende, selbstdefinierte Diversifizierung von Individuen wird, zumindest in unserem Kulturkreis, toleriert (passiver Pluralismus) bzw. gefördert (aktiver Pluralismus). Durch die Globalisierung und die exponentielle Ausbreitung des Internets kommen zudem ständig neue, bereits existierende kulturelle Eigenheiten in die Wahrnehmung lokal entfernter Individuen, die dadurch beeinflusst werden können. Es entstehen ständig neue Mikro-Kulturen (Tribes), zu denen Personen, unabhängig von ihrer physischen Lokalisierung, gehören können, wobei einzelne Personen auch einer Vielzahl von unterschiedlichen, vernetzten Tribes angehören können. Phänomene dieser Art werden durch Atomisierung des Individuums und gleichzeitige multiple, virtuelle Agglomeration beschrieben.

Die Transformation einer industriellen Gesellschaft zu einer Wissensgesellschaft ist begleitet von einer fortschreitenden Diversifizierung der Wissenschaften in Spezialgebiete und der Beschleunigung der Wissensproduktion. Dies ergibt sich, da immer mehr Individuen an diesen Produktionsprozessen beteiligt werden, und da immer mehr Datenerfassungsgeräte in Verbindung mit Data-Mining-Operationen Wissen produzieren (Witten & Frank (1999[362])). Mit dem erzeugten Wissen werden zwar bestehende Fragen und Probleme gelöst, gleichzeitig entstehen dadurch aber mehr neue Fragen, Probleme und Ziele, was in den Wissenschaften ein offensichtliches Phänomen ist. Dadurch steigt die Diversität der Probleme und Ziele, durch die externe Informationsbedürfnisse entstehen. Es kann vermutet werden, dass es in Zukunft nicht nur zu einer ständig steigenden mentalen Diversität der Agenten kommt, sondern dass sich die Diversitätsausweitung noch beschleunigt (positive zweite Ableitung). Diese Beschleunigung führt zu einer Unsicherheit zweiter Ordnung, da nicht nur die Diversität, sondern auch das Veränderungsverhalten der Diversität modelliert werden müsste.

Der Versuch, eine solche mentale und inhaltliche Diversität mit monolithischen Systemen zu modellieren, ist nicht adäquat. Im Vergleich zu der beschleunigten Dynamik der zugrunde liegenden Phänomene ist der Aufwand und der Zeitbedarf zur Anpassung monolithischer Systeme viel zu groß, sodass ein Auseinanderdriften von Anspruch und Wirklichkeit dieser Systemarchitekturen sicher ist. Nicht stationäre, sich beschleunigende Phänomene sind nur durch viele einfache, adaptive, verteilte und kooperative Systemarchitekturen adäquat modellierbar, d.h. durch Architekturen, in denen Polyrepräsentation ein wesentlicher Bestandteil ist.

Unsicherheit, Vagheit und heterogene Strukturen können durch Cluster und Cluster-Prototypen modelliert werden, wobei dies einer Polyrepräsentation entspricht, wenn die Einzelrepräsentationen Cluster-Prototypen entsprechen. Implizit ist eine Polyrepräsentation in den bekannten Modellen zur Modellierung von Unsicherheit und Unschärfe bereits enthalten, da die Zuordnung einer Verteilungsfunktion (Probabilistik) oder einer Zugehörigkeitsfunktion (Fuzzy-Sets; Höhle & Rodabaugh (1999[166]), Novak & Perfilieva (1999[239]), Nguyen (1999[234])) zu einem Objekt eine Form der multiplen Repräsentation der Eigenschaften dieses Objektes ist. Hervorzuheben ist, dass Verteilungs- und Zugehörigkeitsfunktionen meist als reelle Funktionen modelliert werden, die aus einer endlichen Anzahl von Ereignissen approximiert werden. Dies entspricht dem Sonderfall einer stetigen Polyrepräsentation, d.h. einer Polyrepräsentation mit einer überabzählbaren Menge von Einzelrepräsentationen. Durch die endliche Anzahl der bekannten Ereignisse ist ein stetiger Zusammenhang jedoch nur eine Idealisierung, die zum effizienteren Umgang mit dem zugrunde liegenden Phänomen eingeführt wurde, sodass der allgemeine Fall einer diskreten Polyrepräsentation mit einer endlichen Menge von Einzelrepräsentationen in dieser Arbeit unterstellt wird.

1.7) Relevanz-Approximationsmodelle

Im Rahmen der Approximation (siehe Abschnitt 2.1.2)) wird versucht, eine Ursprungsfunktion durch eine andere Funktion zu ersetzen, wobei die neue Funktion (Approximationsfunktion) bestimmte Eigenschaften in Bezug zu der Ursprungsfunktion besitzen muss. Bei diesen Eigenschaften kann es sich um Effektivitäts-Eigenschaften handeln, d.h. es wird ein oder mehrere Qualitätsmaße definiert, welche die

Ähnlichkeit der Ursprungs- und der Approximationsfunktion beschreiben. Ziel ist dabei, die Ähnlichkeit beider Funktionen zu maximieren bzw. größer als ein Schwellenwert werden zu lassen. Es können zusätzlich Effizienz-Eigenschaften festgelegt werden, d.h. die Approximationsfunktion soll mit einem geringeren Rechen- und/oder Speicheraufwand berechnet werden können als die Ursprungsfunktion, wenn ein Schwellenwert eines Effektivitätsmaßes gegeben ist.

Die Ursprungsfunktion kann dabei in einer geschlossenen Form bekannt oder unbekannt sein, wobei im letzteren Fall eine endliche Anzahl von Input-Output-Tupeln (x, f(x)) vorliegt, wovon im weiteren Verlauf der Arbeit ausgegangen werden soll. Approximation beim Vorliegen einer endlichen Anzahl von (x, f(x))-Tupeln kann in zwei Fälle unterschieden werden. Zum einen kann die Approximationsfunktion so gewählt werden, dass an allen Punkten im Inputraum die Approximationsfunktion nicht von der Ursprungsfunktion abweicht, was mit Interpolation beschrieben wird (siehe auch Göppert (1997:4[149])). Dies bedeutet jedoch nicht, dass Ursprungs- und Approximationsfunktion gleich sind, da sie zwischen den verwendeten Punkten im Inputraum abweichen kann. Demgegenüber steht die Regression als zweite Unterteilung der Approximation, bei der der Funktionswert der Ursprungs- und Approximationsfunktion an den vorgegebenen Punkten im Inputraum abweichen dürfen.

Die Verwendung der Regression als Approximationsverfahren ist dann sinnvoll, wenn die Funktionswerte der Ursprungsfunktion bzw. des Ursprungsprozesses einer gewissen Unsicherheit unterliegen, d.h. wenn die Funktionswerte nicht genau bestimmbar sind. In diesem Fall nützt es nichts, eine Approximationsfunktion zu erzeugen, die an diesen Stellen genau ist. Diese Situation ergibt sich bei jeder Form von Messung, da alle Messinstrumente einen Messfehler besitzen, d.h. alle Messpunkte müssen interpretiert werden als Mittelpunkt eines Intervalls, dessen untere und obere Grenze durch den Messfehler gegeben ist, der vor der Messung durch eine Kalibrierung des Messinstrumentes ermittelt wird. Diese Situation ergibt sich jedoch auch in allen Fällen der Bewertung und Selbstaussagen durch einen Agenten, unabhängig, ob dessen interne Architektur direkt einsehbar ist oder nicht. Ist die interne Architektur einsehbar, so ist dies eine externe Messung, während Selbstausagen eine Form von interner Messung von Zuständen eines Subsystems der Architektur durch ein anderes Subsystem ist, sodass in beiden Fällen unsichere Funktionswerte gewonnen werden. Bei einem menschlichen Agenten, der im betrachteten Kontext des IR vereinfacht durch ein Problemlöse-, Sprachanalyse- und Sprachsyntheseareal modelliert wird, ergibt sich die Unsicherheit aus den Funktionen, die zwischen dem Sprachanalyse- und dem Problemlöseareal und zwischen dem Problemlöse- und dem Sprachsyntheseareal Repräsentationen abbilden.

Aus diesen Gründen ist es sinnvoll, Regressionsmodelle als Spezialisierung des Konzeptes der Approximation im Kontext des IR zu verwenden.

Relevanz-Approximations- bzw. -Regressionsmodelle können als ein Spezialfall eines Nutzermodells betrachtet werden, wobei Nutzermodell allgemein als eine Repräsentation von Eigenschaften bzw. Fähigkeiten eines Nutzers/Agenten betrachtet wird. Im Kontext der Relevanzmodelle bedeutet dies, dass ein entsprechendes Modell die Fähigkeit eines Agenten approximiert, Relevanzurteile bezüglich Dokumenten zu treffen, unter dem impliziten Constraint des Problemlösungsprozesses, in dem der Agent sich befindet. Für das Modell ist das Problem bzw. eine Struktur aus Teilproblemen, aus der das externe Informationsbedürfnis abgeleitet wurde, im IR-Standardfall nicht explizit bekannt, d.h. der Agent wird nicht

vor der Queryabgabe aufgefordert, die für ihn motivierende Problemstruktur zu beschreiben. Aus der Wahl der Query durch den Agenten kann jedoch zumindest auf Problemfelder geschlossen werden.

Für die Erzeugung eines Nutzermodells durch ein IRS, sowie für bestimmte Verfahren des Relevanz-Feedbacks, wie dem Dokumentvektoren-RF, stellt sich die prinzipielle Problematik des Stabilitäts-Plastizitäts-Dilemmas, das in allen inkrementell und lange lernenden Systemen, inbesondere in Neuronalen Netzen, eine große Bedeutung besitzt. Gemeint ist damit die Veränderung von Repräsentationen und die Geschwindigkeit der Veränderung durch adaptive Verfahren im Kontext von externen Anforderungen einer offenen, nicht-stationären Welt. Stabile Repräsentationen sind in Umgebungen nützlich, die relativ stationär sind, während plastische Repräsentationen, in einer stärker nicht-stationären Umgebung, nützlich sind. Ein Nutzermodell on-scratch aus den Interaktionen eines individuellen Agenten aufzubauen, erscheint weniger effizient als die Anpassung eines Grundmodells durch individuelle Parameter einer Interaktion. Dieses Grundmodell oder Menge von prototypischen Grundmodellen wird über einen langen Zeitraum aus der Interaktion einer großen Anzahl von Agenten gebildet, wobei sich dessen Struktur nur langsam mit den Interaktionen der Agenten verändert (Stabilität der Grundmodelle). Demgegenüber muss die Anpassung eines Grundmodells an einen einzelnen Agenten schnell, d.h. mit Hilfe der Daten aus wenigen Interaktionsakten hergeleitet werden (Plastizität der Grundmodelle).

Genauer betrachtet ist das Relevanzmodell eine Funktion, die einen Punkt im Dokumentvektorenraum (Inputraum) auf einen Punkt in einem Relevanzraum (Outputraum) abbildet, wobei im Rahmen dieser Arbeit angenommen wird, dass es sich um einen genau ein-dimensionalen Relevanzraum handelt (siehe auch Cooper et al. (1992[79]), Gey (1994[144]), Robertson & Walker (1994[287])). Mehrdimensionale Relevanzräume, deren Dimensionen durch verschiedene Typen bzw. Operationalisierungen des Relevanzbegriffes bestimmt sind (Swanson (1987[328]), siehe auch Abschnitt 3.9.1)), sollen nicht betrachtet werden. Das gleiche gilt für mehrdimensionale Relevanzräume, deren Dimensionen einen direkten Bezug zu den Dimensionen des Dokumentvektorenraumes besitzt, da jeder Dimension des Dokumentvektorenraumes, d.h. jedem Merkmal (oder Merkmals-Cluster) ein eigener Relevanzwert zugeordnet werden kann.

Im Rahmen dieser Arbeit werden stützpunktbasierte Approximationsmodelle betrachtet, d.h. es existiert eine Menge von Stützpunkten, die jeweils eine Komponente im Inputraum und eine Komponente im Relevanzraum besitzen. Mit diesen Stützpunkten und einem Approxmationsverfahren kann jedem anderen Punkt im Inputraum eine Relevanzschätzung, d.h. eine Komponente im Relevanzraum, zugeordnet werden. Stützpunktbasierte Modelle können unterschieden werden in instanzbasierte Modelle (z.B. Local-Weighted-Regression-Ansätze; siehe Abschnitt 2.1.3)), die Stimuli aus der Lernmenge direkt als Stützpunkte verwenden, und in prototypbasierte Modelle, bei denen aus den Lernstimuli in einem Zwischenschritt eine geringere Anzahl von Prototypen gelernt werden, die als Stützpunkte verwendet werden (z.B. GNG-SOM-basierte Ansätze; siehe Abschnitte 2.1.5) bis 2.1.8)).

Die Polyrepräsentation von Relevanz-Approximationsmodellen spielt im Rahmen dieser Arbeit eine entscheidende Rolle. D.h. bezogen auf eine Relevanz-Feedback-Interaktion eines Agenten, wird durch das IRS eine Menge von Relevanz-Approximationsmodellen auf der Grundlage der Relevanzbewertungen des Agenten gebildet, um die vielfältigen Quellen der Unsicherheit bei der Relevanzbewertung durch den

Agenten modellierbar zu machen. Diese Relevanzmodelle werden als Elemente in einem Combining-Model (Wolpert (1990[365], 1993[367]), Brodley (1994[58]), Drucker et al. (1994[98]), Freund (1995[124]), Freund & Schapire (1995[125], 1996[126]), Kong & Dietterich (1995[186]), Drucker & Cortes (1996[99]), Quinlan (1993[267], 1996[268]), Skalak (1996[313]), Merz (1998[216]), Gama (1999[141]), Witten & Ting (1999[364])) eingesetzt, indem sie noch nicht nachgewiesene Dokumentvektoren bewerten. Auf diese Weise erzeugt jedes Modell eine geordnete Liste von Dokumentvektoren (Ranking), gefolgt von der Aggregation der Listen durch ein Fusion-Verfahren (Vogt (1999: 25ff[352])) bzw. der Aggregation der Relevanzbewertungen, um genau eine Dokumentvektorenliste zu erzeugen, deren korrespondierende Dokumente dem Agenten in der gleichen Reihenfolge präsentiert werden.

Es existiert eine Reihe von Möglichkeiten, wie eine Polyrepräsentation von Relevanz-Approximationsmodellen in Abhängigkeit der vorliegenden Repräsentationen erzeugt werden kann. Schwerpunktmäßig werden statistische Erzeugungsmöglichkeiten durch Resamplingverfahren (siehe Abschnitt 2.2)) in Kombination mit den stützpunktbasierten Approximationsmodellen betrachtet. Die Einzelmodelle werden erzeugt, indem auf die Gesamtmenge der Dokumentvektoren, die über den Umweg der korrespondierenden Dokumente vom Agenten eine Relevanzbewertung erhalten haben, Resampling-Operationen angewendet werden, wodurch künstliche Stimulusmengen erzeugt werden. Diese Mengen aus bewerteten Dokumentvektoren werden verwendet, um unterschiedliche Stützpunktmengen direkt (bei instanzbasierten LWR-Ansätzen) oder indirekt (bei prototypbasierten GNG-SOM-Ansätzen) zu spezifizieren. Jede dieser Stützpunktmengen bildet zusammen mit einem Approximationsverfahren ein Relevanz-Approximationsmodell, deren Ergebnisse in einem nachfolgenden Schritt aggregiert werden.

1.8) Aktives Lernen

1.8.1) Passives und aktives Lernen

Überwachtes Lernen wird allgemein in dieser Arbeit als die Bildung eines Approximationsmodells mit Hilfe einer Menge von Stimuli mit der Struktur m = (x, f(x)) verstanden. Passives Lernen bezeichnet Lernsituationen, bei denen ein Lernender keinen Einfluss darauf besitzt, was er wie lernt, da das Lernverfahren, sowie die Stimulusmenge extern gegeben ist. Aktives Lernen bezeichnet allgemein Lernsituationen, in denen der Lernende Einfluss auf die Komponenten des Lernprozesses besitzt, d.h. was gelernt wird, in welcher Reihenfolge gelernt wird, welche Methoden verwendet werden, u.ä.

Eine solche Einflussnahme kann als rational im Sinne eines rationalen Agenten bezeichnet werden, d.h. es liegt ein rationales aktives Lernen vor, wenn der Lernende damit explizit Ziele verfolgt, wobei die Verbesserung der Performance des Lernenden im Vordergrund steht. Verbunden sind damit Effektivitäts- und Effizienz-Ziele, d.h. der Lernende will durch seine Einflussnahme z.B. geringere Fehler bezüglich einer Lerndomäne machen, in der er bereits Erfahrung besitzt, oder er will durch seine Einflussnahme neue Stimuli schneller lernen. Wird Lernen allgemein als Aufbau eines Modells betrachtet, so bedeutet dies, dass durch aktives Lernen das Modell eine geringere Fehlerbewertung erhält, bzw. dass der Modellaufbau mit weniger Aufwand durchgeführt wird. Operationalisiert werden kann der Aufwand durch die Anzahl der Stimuli oder der Stimuluspräsentationen, die zum Aufbau eines Modells mit vorgegebenen

Performancewerten notwendig sind. Rationales, aktives Lernen ist somit eine heterogene Klasse von Verfahren, mit denen Approximationsmodelle effizienter und/oder effektiver erzeugt werden können.

Im Kontext des Machine Learnings wird die Einflussnahme in der Regel eingeschränkt auf überwachte Lernsituationen, in denen der Lernende Einfluss auf die Stimuli besitzt, die er lernt, d.h. auf das was gelernt wird und in welcher Reihenfolge gelernt wird. Relevant ist insbesondere die Minimierung der Anzahl der Stimuli, um ein Modell mit vorgegebenen Performancewerten zu erzeugen, wenn Lernstimuli zunächst erzeugt werden müssen, d.h. wenn nicht eine große Stimulusmenge vorhanden ist, aus der ausgewählt werden kann. Ist der Aufwand groß, um Lernstimuli zu erzeugen, so ist es sinnvoll, so wenig wie möglich Stimuli erzeugen zu müssen, wobei die Festlegung, welche Stimuli erzeugt und gelernt werden, Aufgabe des Lernenden im Rahmen des aktiven Lernens ist.

Rationales, aktives Lernen, das die Modellperformance des Lernenden maximieren soll, kann auf einer Analyse des Modells basieren, indem seine Stärken und insbesondere Schwächen explizit beschrieben werden. Eine systematische Beschreibung der Stärken und Schwächen eines Modells kann als Metamodell interpretiert werden, d.h. es können Fehler- und Unsicherheitsmodelle eines Ursprungsmodells gebildet werden (siehe Abschnitt 2.3.5)), die zum Zweck des aktiven Lernens benutzt werden. Lernende, welche diese Formen von Metamodellen erster oder höherer Ordnung bilden können und die Metamodelle zum Zweck der Einflussnahme auf den weiteren Lernprozess benutzen, können als reflektive Lernende bezeichnet (Beyer & Smieja (1994[45]), Mühlenbein (1994[229], 1995[230])) werden.

1.8.2) Geschlossene und offene Lernmenge

Im Kontext des IR ist aktives Lernen bei einer geschlossenen Lernmenge relevant (siehe Abschnitt 5.1.2)), d.h. es existiert eine endliche Menge von potentiellen Lernstimuli, in deren Datenstruktur nur die Inputkomponente, nicht jedoch die Outputkomponente belegt ist. Aus dieser Grundmenge soll eine kleine Teilmenge ausgewählt werden, zu der die Outputkomponente extern ermittelt wird. Für das IR bedeutet dies, dass aus der Grundmenge der bislang nicht nachgewiesenen Dokumentvektoren, d.h. der Dokumentvektoren, zu denen keine Relevanzbewertungen vorliegen, eine kleine Teilmenge ausgewählt werden soll. Die Dokumente, die zu dieser Ergebnis-Dokumentvektorenmenge korrespondieren, werden dem Agenten präsentiert, der Relevanzbewertungen für die Dokumente abgibt. Diese Relevanzbewertungen werden als Bewertungen für die Dokumentvektoren übernommen, sodass nun Dokumentvektoren mit Relevanzbewertungen vorliegen, die als vollständige Lernstimuli für einen überwachten Lernprozess verwendet werden, z.B. für den Aufbau oder die Erweiterung eines Relevanz-Approximationsmodells. Die Auswahl der Dokumentvektoren soll einem Performance-Ziel bezüglich des Relevanz-Approximationsmodells dienen, d.h. es werden solche Dokumentvektoren ausgewählt, die zu einer Maximierung eines Performancekriteriums des Approximationsmodells führen, wenn diesen Stimuli Relevanzwerte durch den Agenten zugeordnet werden, und das Modell mit diesen neuen Stimuli nachadaptiert wird.

Aktives Lernen bei einer offenen Lernmenge beschreibt die Situation, dass alle Punkte des Inputraumes, zu denen noch kein Punkt im Outputraum zugeordnet wurde, als potentielle Lernstimuli infrage kommen können (siehe Abschnitt 5.1.4)). Hierbei werden stetige Suchverfahren wie ein Gradientenverfahren sinn-

voll, wenn ein Suchkriterium als differenzierbare Funktion formulierbar ist und wenn pro Iteration genau ein Stimulus erzeugt werden soll (siehe Cohn (1994[71], 1995[73]), Cohn et al. (1994[72], 1995[74])). Als Alternative können populationsbasierte Verfahren, wie Evolutions-Algorithmen, verwendet werden, wenn pro Iteration mehrere Lernstimuli erzeugt werden sollen.

Im Kontext des IR besitzt eine offene Lernmenge keine direkte Anwendung, da nur die Punkte des Dokumentvektorenraumes eine Bedeutung besitzen, die zu einem Dokument korrespondieren. Würde durch ein stetiges Suchverfahren ein Punkt im DVR als nächster Lernstimulus selektiert, zu dem kein Dokument korrespondiert, so müsste durch einen nachfolgenden Schritt der am nächsten liegende Dokumentvektor mit einem korrespondierenden Dokument ausgewählt werden, damit dieses Dokument letztendlich dem Agenten präsentiert werden kann.

Im Rahmen dieser Arbeit wird ausschließlich aktives Lernen mit einer geschlossenen Lernmenge betrachtet, da dies der Situation des IR am besten entspricht.

1.8.3) Direkte und indirekte Verfahren des aktiven Lernens

Bei indirekten Verfahren zum aktiven Lernen werden Eigenschaften der Stimuli in Abhängigkeit der momentan verwendeten Approximationsmodelle ermittelt, wie z.B. die Output-Varianz auf der Basis einer Modellmenge, wobei eine solche Strategie als Query-by-Disagreement bezeichnet wird (RayChaudhuri & Hamey (1995[274], 1996[275]), RayChaudhuri (1997[276])), d.h. das „Disagreement“ wird durch die Output-Varianz operationalisiert. Hierzu notwendig ist eine Menge von Modellen (Committee of Experts), sodass die Strategie Query-by-Disagreement als Spezialisierung der Strategieklasse Query-by-Committee (Seung et al. (1992[308]), Freund et al. (1993[123])) betrachtet werden kann. Diese Verfahrensklasse wird als indirekt bezeichnet, da beim aktiven Lernen ein Modellaufbau erfolgt, sodass Modelleigenschaften anstatt Stimuluseigenschaften betrachtet werden müssten. Bei den indirekten Verfahren wird eine Korrelation zwischen Stimulus- und Modelleigenschaften unterstellt, indem z.B. angenommen wird, dass der Stimulus mit der größten Output-Unsicherheit, operationalisiert durch die Output-Varianz, zu dem Modell mit der größten Verringerung der Unsicherheit führen wird, wenn dem Stimulus ein Outputwert (Relevanzwert) extern zugeordnet wird, und das Modell mit diesem neuen Stimulus nachadaptiert wird.

Die direkten Verfahren zum aktiven Lernen sind mit der Fragestellung der Modellauswahl (Model Selection; siehe z.B. Hijorth (1994[163]), Cohn (1994[71], 1995[73]), Cohn et al. (1994[72], 1995[74])) direkt konfrontiert, d.h. es wird in einem Iterationsschritt eine Menge von alternativen Modellen gebildet, aus der ein Modell auf Grund seiner Eigenschaften ausgewählt werden muss. Ausgehend von einer Grundmenge potentieller Stimuli, die jeweils keine Output-Komponente besitzen, werden Teilmengen festgelegt, wobei meist von der einfachsten Möglichkeit ausgegangen wird, dass in jeder Teilmenge genau ein Element aus der Grundmenge enthalten ist. Die Menge der Lernstimuli, welche das momentane Approximationsmodell bildet, wird jeweils unabhängig um eine Stimulusteilmenge erweitert, wobei dem darin enthaltenen potentiellen Stimulus eine Outputschätzung zugeordnet wird. Das momentane Approximationsmodell wird auf der Basis der neuen Lernmenge nachadaptiert, gefolgt von der Bestim-

mung der Modellqualität durch eine Reihe möglicher Qualitätsmaße, wie z.B. eine Schätzung des Output-Varianz- bzw. Bias-Integrals (Optimal-Experiment-Design; s.a. Fedorov (1972[113]), Cohn (1994[71], 1995[73]), Cohn et al. (1994[72], 1995[74])). Nachdem für alle Stimulusteilmengen ein eigenes Modell gebildet wurde, können diese durch die Qualitätsmaße untereinander verglichen werden, wobei das Modell mit der besten Qualität ausgewählt wird. Für das Element aus der korrespondierenden Stimulusmenge wird nun extern der richtige Output bestimmt, der die Outputschätzung ersetzt, gefolgt von der Nachadaption des ursprünglichen Approximationsmodells auf der Basis der neuen Lernstimulusmenge und der Bestimmung der neuen Modellqualität.

1.8.4) Effektivitäts- und Effizienz-Vergleich direkter und indirekter Verfahren

Direkte und indirekte Verfahren unterscheiden sich insbesondere durch ihre Effektivitätsbegründung und ihren Berechnungsaufwand. Indirekte Verfahren beruhen auf der Annahme der Korrelation zwischen Stimuluseigenschaften und Modelleigenschaften, wenn der Stimulus in die Lernmenge des Ursprungsmodells aufgenommen wird. Bezüglich der Unsicherheit, operationalisiert durch die Output-Varianz, mag diese Korrelation begründet sein, doch bezüglich anderer Modelleigenschaften kann eine solche Korrelation nicht unterstellt werden, d.h. sie müsste für einen Anwendungsfall zunächst geprüft werden. Direkte Verfahren besitzen keine solche Probleme bezüglich der Effektivitätsbegründung, da Modelle faktisch erzeugt und getestet werden, doch gerade hierin liegt das Effizienzproblem direkter Verfahren, da Nachadaption und insbesondere die Bestimmung der Modellqualität um die Alternativen differenzieren zu können, sehr große Rechenressourcen erfordert.

Der Aufwand von indirekten Verfahren wird dominiert von der Anzahl der verwendeten Modelle und dem Aufwand der Erzeugung bzw. Aktualisierung der einzelnen Modelle. Werden Resampling-Verfahren (Efron & Tibshirani (1993[105])) als Quelle der Polyrepräsentation verwendet, so ist die Anzahl der Modelle typischerweise hoch. Werden Approximationsmodelle auf der Basis von GNG-SOMs bzw. SC-GNG-SOMs verwendet (Fritzke (1998[132]), Bachelier (1998a[15], c[17], 1999d[22])), so existieren einige Strategien, den Aufwand der Modellaktualisierung sehr klein zu halten, z.B. durch Nachadaption bei SC-GNG-SOMs (Bachelier (1998c: 58ff[17])), Modellpoolerzeugung durch Resamplingverfahren (Bachelier (1999c: 35ff[21])), oder Verwendung von Modellerzeugungs-Historien (Bachelier (1999d: 117ff[22])).

1.8.5) Relevanz- und Modell-Maximierungskriterium

Im Kontext des IR und der Relevanz-Approximationsmodelle wird ein aktiv lernendes IRS Einfluss darauf nehmen, welchen Dokumentvektoren ein Relevanzwert zugeordnet werden soll. Dies entspricht bereits der Grundsituation des Relevanz-Feedbacks, da durch die Relevanzwerte das IRS bestimmte interne Repräsentationen, wie den Queryvektor oder Dokumentvektoren so verändert, dass mit Hilfe der gleichen Retrieval-Funktion andere Dokumentvektoren ausgewählt und bewertet werden. Die Auswahl von Dokumentvektoren beim Relevanz-Feedback kann jedoch nicht als rationales, aktives Lernen betrachtet werden, da der Lernende damit kein explizites Ziel bezüglich des zu erzeugenden Modells verbindet. Die Zielsetzung beim Relevanz-Feedback besteht in dem Relevanz-Maximierungskriterium, d.h.

es werden die Dokumentvektoren ermittelt, von denen das IRS annimmt, dass sie mit den höchsten Relevanzwerten durch den Agenten bewertet werden. Wird ein Relevanz-Approximationsmodell im Kontext des Relevanz-Feedbacks verwendet, so muss das Modell die daraus resultierenden Stimuli lernen, unabhängig von der Schwierigkeit in Verbindung mit der bestehenden Modellstruktur. Auf diese Weise wird das Modell spezialisiert auf die Schätzung von guten Relevanzwerten, da tendenziell keine Stimuli mit schlechten Relevanzwerten als Lernstimuli zur Verfügung stehen. Die Zielsetzung bei einem Modellaufbau im Kontext des aktiven Lernens besteht in dem Modell-Maximierungskriterium, d.h. es werden Dokumentvektoren mit erwarteten guten wie schlechten Relevanzwerten nachgewiesen, mit dem Zweck, ein Relevanzmodell aufzubauen, welches das gesamte Spektrum der Relevanzbewertungen des Agenten adäquat modellieren kann (siehe Abschnitt 5.2)).

Ein ausschließliches aktives Lernen im Kontext des IR kann somit als eine Form des Relevanz-Feedbacks mit einem abweichenden Optimierungskriterium bezeichnet werden, d.h. es wird ein Modell-Maximierungskriterium anstatt einem Relevanz-Maximierungskriterium verwendet.

Das Relevanz- und das Modell-Maximierungskriterium führen zu einem offensichtlichen Konflikt, der durch den begrenzten Willen eines menschlichen Agenten verursacht ist, Interaktionszeit zu investieren und Relevanzbewertungen vorzunehmen. Diese begrenzte Ressource führt dazu, dass beim Standard-Relevanz-Feedback so schnell wie möglich Dokumente mit den höchsten Relevanzschätzungen nachgewiesen werden sollen, d.h. dass das Relevanz-Maximierungskriterium angewendet wird. Dies hat jedoch Nachteile für das zu erstellende Relevanz-Approximationsmodell, da ein systematischer Fehler (Bias) erzeugt wird, indem vorzugsweise gute Stimuli zur Verfügung stehen und somit gelernt werden können. Dies führt zu einer mangelnden Diskriminationsfähigkeit zwischen guten und schlechten Dokumentvektoren und zu einer systematisch zu guten Schätzung von Dokumentvektoren. Das sich ergebende Approximationsmodell, als Spezialfall eines Nutzermodells, besitzt dadurch eingeschränkte Anwendungsmöglichkeiten, da selbst bei dem gleichen Agenten jedoch einem leicht abweichenden, zukünftigen Informationsbedürfnis dieses Modell nur eingeschränkt als Initialisierungsmodell eingesetzt werden kann, welches zu Beginn der Interaktion zur Relevanzschätzung verwendet wird.

Wird jedoch der Schwerpunkt zu stark auf die Modellbildung gelegt, so werden mehr Interaktionsakte in einem Feedbackprozess benötigt, um ein gutes Modell zu erzeugen, welches gute und schlechte Dokumentvektoren unterscheiden kann. Insbesondere muss der Agent deutlich mehr schlechte Dokumentvektoren, d.h. korrespondierende Dokumente, bewerten, was möglicherweise über dessen Frustrationsgrenze geht, mit der Folge, dass die Interaktion abgebrochen wird, und die vorliegenden Lernstimuli nicht für ein hinreichend gutes Modell ausreichen.

Zusammengefasst bedeutet dies, dass der Modellaufbau unter Verwendung des aktiven Lernens als ein Zwei-Ziel-Optimierungsproblem mit dem Relevanz- und Modell-Maximierungskriterium interpretiert werden muss, sodass eine Integrationsstrategie notwendig wird. Dabei sind als Basisstrategien die Optimierung mit Hilfe des Paretokriteriums oder Gewichtungsverfahrens anwendbar. Bei letzterem wird z.B. zu Beginn der Interaktion der Anteil der Dokumentvektoren höher bewertet, die durch das Modell-Maximierungskriterium ermittelt werden, als der Anteil der Dokumentvektoren, die durch das Relevanz-Maximierungskriterium ermittelt werden. D.h. es werden mehr Dokumentvektoren durch das erste als

durch das zweite Kriterium ausgewählt. Dies hat zur Folge, dass schnell ein gutes Relevanz-Approximationsmodell aufgebaut wird, welches in den folgenden Interaktionsakten genutzt werden kann, um effektive Relevanzschätzungen durchzuführen. Die verbesserte Modellqualität kann letztendlich zu weniger Fehlschätzungen und somit letztlich zu weniger nachgewiesenen Dokumenten führen, welche vom Agenten mit einem geringen Relevanzwert bewertet werden. Die Investition des Aufbaus eines guten Modells zu Beginn der Interaktion zahlt sich bezogen auf die gesamte Interaktion aus, indem die Anzahl der nachgewiesenen und nicht relevanten Dokumente sinkt und die Precision somit steigt.

1.8.6) Modell-Polyrepräsentation beim aktiven Lernen

Die Polyrepräsentation von Relevanz-Approximationsmodellen spielt bei den indirekten Ansätzen zum aktiven Lernen eine entscheidende Rolle, da bei einer Query-by-Disagreement-Strategie eine Modellmenge vorhanden sein muss, wobei jedes Modell für einen potentiellen Lernstimulus eine Output-Schätzung abgibt. Aus diesen Einzelschätzungen wird eine Statistik abgeleitet, wie die Bestimmung des arithmetischen Mittelwertes und der Varianz der Output-Schätzungen. Ausgewählt wird z.B. der potentielle Stimulus mit der maximalen Output-Varianz-Schätzung, da durch die implizite Korrelation von Stimulus- und Modelleigenschaft angenommen wird, dass die Integration dieses Stimulus in die Lernmenge und die Nachadaption der Modelle zu einer maximalen Verringerung der Unsicherheit der Modelle führt.

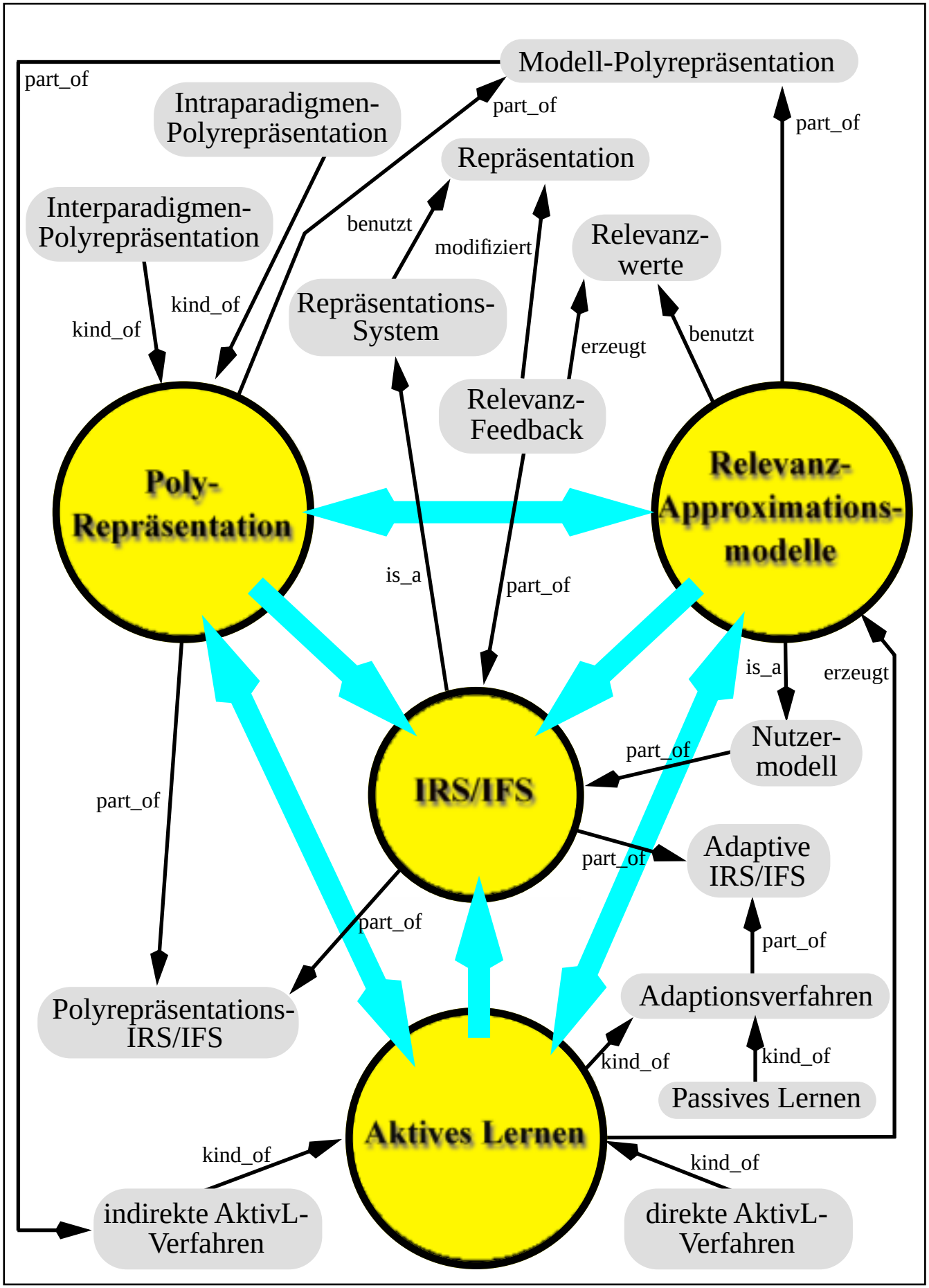
Bei den direkten Verfahren muss keine Modell-Polyrepräsentation verwendet werden, es können jedoch für bestimmte Aufgaben, wie die Bestimmung von effektiveren Output-Schätzungen und Bias-Schätzungen, Modell-Polyrepräsentationen sinnvoll genutzt werden (siehe auch Cohn (1995:3f[73])).

1.9) Semantisches Netz der Beziehungen der verwendeten Ansätze

Die Beziehungen der drei grundlegenden Konzepte Polyrepräsentation, Relevanz-Approximationsmodell und aktives Lernen werden in dem nachfolgenden Beziehungsnetz zusammengefasst. Wird mit der Polyrepräsentation begonnen, so besteht die Beziehung zu den IRS/IFS in der Verwendung einer oder mehrerer polyrepräsentierter Komponenten des Definitions-Tupels von IRS/IFS, wie z.B. eine Query-Indexierungsfunktions-Polyrepräsentation.

Die Beziehung zwischen Polyrepräsentation und den Approximationsmodellen besteht in der Modell-Polyrepräsentation, d.h. in der Verwendung einer Menge von Approximationsmodellen, deren Elemente in einer Combining-Model-Architektur strukturiert sind. Alle Modelle, bzw. die Modelle der ersten Ebene erhalten den gleichen Inputvektor und liefern einen Output, der über eine oder mehrere Ebenen aggregiert wird. Wird als Lernmenge der Einzelmodelle Stimuli verwendet, deren Inputkomponente einen Dokumentvektor und deren Outputkomponente einen Relevanzwert enthält, so handelt es sich um Relevanz-Approximationsmodelle. Diese Stimuli werden durch einen Relevanz-Feedback-Prozess erzeugt, wodurch die erste Beziehung zwischen Relevanz-Approximationsmodellen und Information-Retrieval bzw. Information-Filtering beschrieben wird. Die zweite Beziehung ergibt sich daraus, dass Relevanz-Approximationsmodelle eine Form von Nutzermodellen sind, die Teil eines IRS/IFS sein können.

Abb. 3) Beziehungsnetz der drei grundlegenden Konzepte



Die Beziehung zwischen Relevanz-Approximationsmodellen und aktivem Lernen ergibt sich daraus, dass aktives Lernen, kontrastierend zu passivem Lernen, eine Klasse von Verfahren darstellt, mit denen Approximationsmodelle erzeugt werden. Passives wie aktives Lernen verwendet Adaptionsverfahren, die ebenfalls von adaptiven IRS/IFS verwendet werden, was eine Beziehung zwischen aktivem Lernen und IRS/IFS beschreibt.

Alle drei Konzepte werden für die indirekten Verfahren zum aktiven Lernen benötigt bzw. für direkte Verfahren, welche Modell-Polyrepräsentation zur Schätzung von Eigenschaften für Kandidaten- und Stützpunkt-Stimuli verwenden.

2) Methodische Grundlagen

2.1) Basis-Verfahren der stützpunktbasierten Approximation

2.1.1) Vektorraum und Metrik

Im Rahmen dieser Arbeit wird das Vektorraummodell des Information-Retrievals verwendet, d.h. alle Operationen finden in Vektorräumen statt, in denen eine Metrik verwendet wird, deren Grundlagen im weiteren kurz dargestellt werden soll.

Gegeben sei ein n-dimensionaler Vektorraum $X \subseteq R^n$, der im Rahmen der Approximation als Inputraum bezeichnet werden soll. Eine Funktion $d(x_k, x_j)$: X x X $\rightarrow$ X wird als Distanzmaß bezeichnet, wenn die folgenden Axiome erfüllt sind:

$$\begin{gathered} d(x_k, x_j) = d(x_j, x_k), \\ d(x_k, x_j) \geq 0, \\ d(x_k, x_j) \geq d(x_k, x_k) = 0. \end{gathered} \tag{3}$$

Ein Distanzmaß wird als Quasimetrik bezeichnet, wenn gilt:

$$\forall x_k, x_i, x_j \in X: d(x_k, x_j) \leq d(x_k, x_i) + d(x_i, x_j). \tag{4}$$

Eine Quasimetrik wird als Metrik bezeichnet, wenn $\forall x_k \in X$ gilt:

$$d(x_i, x_j) = 0 \Rightarrow d(x_i, x_k) = d(x_j, x_k). \tag{5}$$

Es wird gefordert, dass solche Objekte die gleiche Position in dem Vektorraum einnehmen. Die beiden Objekte müssen jedoch nicht gleich sein, d.h. $D_i \neq D_k$.

In einem Vektorraum können unendlich viele Distanzmaße, Quasimetriken und Metriken bezüglich einer Punktmenge definiert werden. Wird genau eine Metrik verwendet, so soll diese mit $d_X(., .)$ bezeichnet werden, während bei der Verwendung mehrerer Metriken in einem gemeinsamen Kontext ein zusätzlicher Index $d_{X(h)}(., .)$ hinzu kommt.

Häufig wird eine Spezifizierung der parametrisierten Minkowski-Metrik verwendet, die definiert ist als v'te Wurzel über der Summe der Differenzen der Komponenten der beiden Vektoren x_i und x_k, wobei die Beträge der Differenzen gebildet werden, die zur v'ten Potenz erhoben wird:

$$d_v(x_i, x_k) = \sqrt[v]{\sum_{j=1->p} |x_{ij} - x_{kj}|^v}. \quad (6)$$

Spezialfälle sind die City-Block-Metrik mit v = 1

$$d_1(x_i, x_k) = \sum_{j=1->p} |x_{ij} - x_{kj}|, \quad (7)$$

und die Euklidische Metrik mit v = 2, die dem Skalarprodukt der beiden Vektoren entspricht:

$$d_2(x_i, x_k) := d_E(x_i, x_k) = ||x_i - x_k|| = \sqrt{\sum_{j=1->p} |x_{ij} - x_{kj}|^2}. \quad (8)$$

Ähnlichkeitsmaße können als eine Form der Umkehrung von Distanzmaßen verstanden werden, auch wenn es sich dabei nicht um Umkehrfunktionen handelt. Eine Funktion $s(x_k, x_j)$: X x X → X wird als Ähnlichkeitsmaß bezeichnet, wenn die folgenden Axiome erfüllt sind:

$$\begin{gathered} s(x_k, x_j) = s(x_j, x_k) \text{ (Symmetrie)}, \\ s(x_k, x_j) \geq 0, \\ s(x_k, x_j) \leq s(x_k, x_k), \\ s(x_k, x_k) = 1 \text{ (Reflexivität)}. \end{gathered} \quad (9)$$

Es existieren einige umkehrbare Transformationsfunktionen, die ein Ähnlichkeitsmaß in ein Distanzmaß umwandeln können (vgl. Panyr (1986a: 56[247])). Plausibel sind:

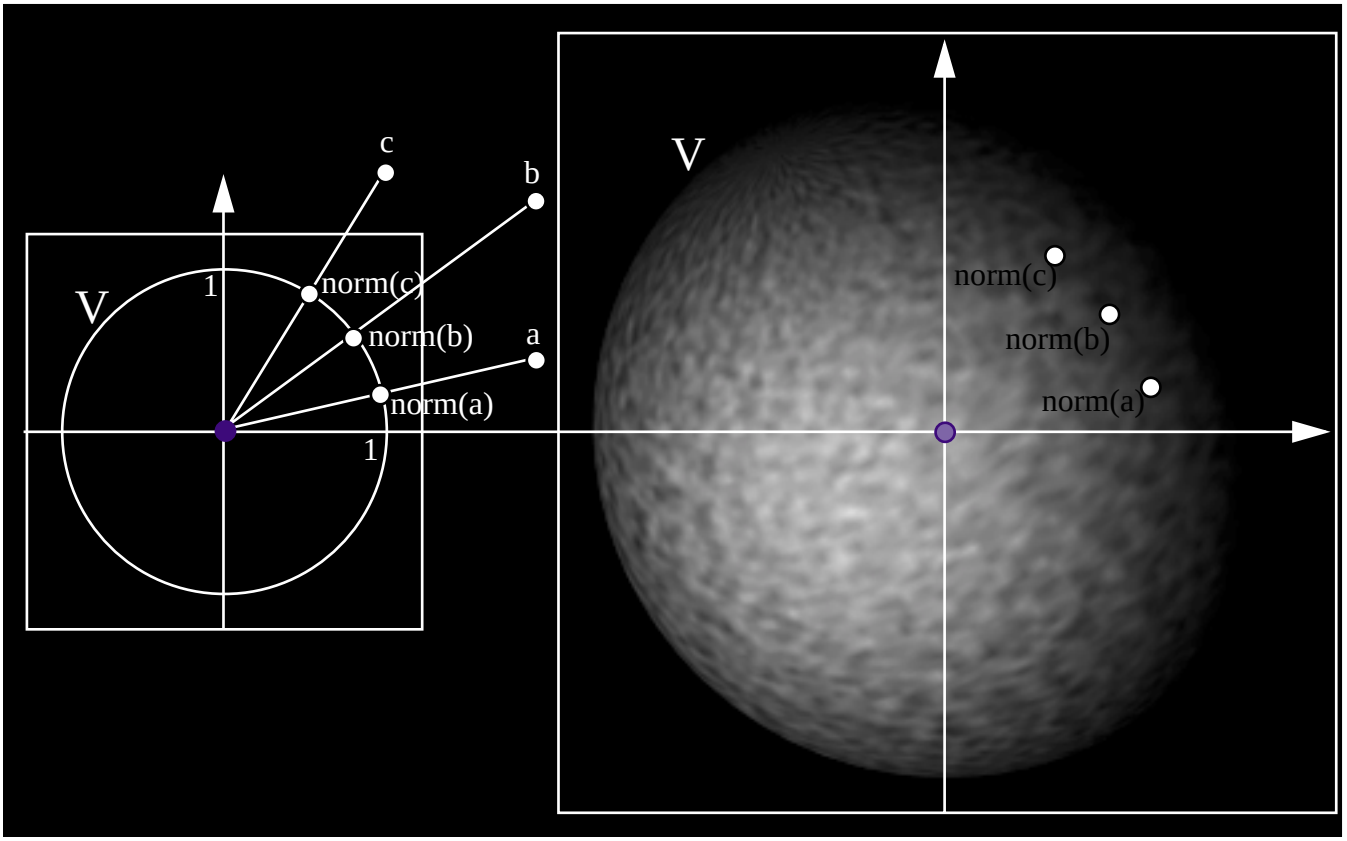
$$\begin{gathered} d(x_k, x_k) = 1 - s(x_k, x_k), \\ d(x_k, x_k) = \sqrt{(1 - s(x_k, x_k))}, \\ d(x_k, x_k) = - \log(s(x_k, x_k)), \\ d(x_k, x_k) = - \mathrm{ld}(s(x_k, x_k)). \end{gathered} \quad (10)$$

Vektoren werden oft in einer (auf Eins) normierten Form verwendet, d.h. sie werden alle so skaliert, dass sie die Länge Eins besitzen. Vektoren werden normiert, indem um den Ursprung des Koordinatensystems zunächst eine Hyperkugel mit dem Radius Eins erzeugt wird. Dann werden die Schnittpunkte der einzelnen Vektoren (a, b, c) mit der Oberfläche der Hyperkugel ermittelt (norm(a), norm(b), norm(c)), die dann als normierte Vektoren verwendet werden (siehe Abb. 4)).

2.1.2) Approximation, Interpolation, Regression

Gegeben sei ein n-dimensionaler Vektorraum $X \subseteq R^n$, der als Inputraum bezeichnet werden soll, und ein m-dimensionaler Vektorraum $Y \subseteq R^m$, der als Outputraum bezeichnet werden soll. Es wird angenommen, dass eine Funktion f(x): X → Y existiert, die durch eine Funktion f(x)^: X → Y ersetzt werden soll. Weiterhin soll ein Qualitäts- bzw. Fehlermaß $Q_{App}(f(x), f(x)^)$ existieren, welches die Ähnlichkeit bzw. Distanz der beiden Funktionen f(x) und f(x)^ durch eine reelle Zahl quantifizieren kann. Das Qualitätsmaß kann als Mittelwert einer endlichen Menge ausgewählter Inputvektoren bezüglich der beiden Funktionswerte definiert sein, oder als Integral, bei dem über den gesamten Inputraum die Abweichungen der beiden Funktionen bestimmt wird.

Abb. 4) Darstellung der Vektor-Normierung



Die Funktion f(x)^ ist eine ε-Approximation von f(x), wenn ein Fehlermaß Q_{App}(f(x), f(x)^) kleinergleich einem kleinen, positiven, reellen Wert ε ist. Faktisch bedeutet dies, dass eine Approximation ein Suffizienzproblem in einem Funktionsraum ist, der alle möglichen Funktionen f(x)^ enthält, wobei nach Funktionen f(x)^ gesucht wird, die eine ε-Approximation von f(x) darstellen. Die Suche nach der effektivsten Approximation ist demgegenüber eine Optimierungsaufgabe, bei der die Funktion f(x)^ gesucht wird, bei der ein gegebenes Fehlermaß minimiert wird. Dies kann in Konflikt mit einer ε-Approximation stehen, wenn das kleinste Fehlermaß größer als ε ist, wenn angenommen werden kann, dass f(x) kein Element des Suchraums ist.

Soll eine Approximationsfunktion in einer bestimmten Klasse von Funktionen gesucht werden, die durch eine Menge von Parametern beschrieben wird, so kann das Approximationsproblem als Suche in einem Parameterraum W reinterpretiert werden. Diese Suche wird auch als Adaption bezeichnet, die durch ein bestimmtes Adaptionsverfahren durchgeführt wird (s.a. Göppert (1997:4[149])).

Ein Fehlermaß kann auch als Extremwert ermittelt werden, wie die maximale Abweichung zweier Funktionswerte f(x) und f(x)^, bzw. es lassen sich diskrete oder stetige Häufigkeits-Verteilungen von Fehler- bzw. Unsicherheitswerten bestimmen, sodass die Definition einer Approximation angepasst werden muss, worauf hier jedoch nicht weiter eingegangen werden soll.

Ist die Funktion f(x) bekannt, so besteht die Aufgabe einer ε-Approximation in einer kombinierten Effektivitäts-Effizienz-Leistung, d.h. in der Suche nach einer ε-Approximationsfunktion, deren Evaluationskosten in Form von Rechenleistung und/oder Speicherplatz kleiner sind. Ist f(x) unbekannt, so muss sie

durch eine Menge von Input-Output-Paaren als Beispiele spezifiziert werden, die von einer ε-Approximationsfunktion rekonstruiert werden sollen. In diesem Fall kann eine Fehlerfunktion nicht über alle möglichen Funktionswerte, sondern nur über die vorliegenden Werte bestimmt werden. Da die richtige Funktion f(x) unbekannt ist, kann auch keine Aussage über eine Effizienz-Leistung der Approximation gemacht werden. Es muss jedoch davon ausgegangen werden, dass die Input-Output-Paare durch physische Experimente, Messung, oder ein deskriptives Modell erzeugt wurden, da die Input-Output-Paare ansonsten ebenfalls nicht verfügbar wären. In diesen Fällen können die Kosten der Ermittlung von Input-Output-Paaren in Relation zu den Kosten der Erstellung eines Approximationsmodells und der Berechnung des Outputs zu gewünschten Inputvektoren gesetzt werden. Das Approximationsmodell wird somit zu einem konstruktiven Modell, dessen Aufgabe in einer Effizienzleistung besteht, die dann erfüllt ist, wenn der Aufwand zur Ermittlung eines Outputs, kleiner ist, als dies durch ein physikalisches Experiment, Messung oder Simulation möglich wäre.

Die Input-Output-Paare sollen im weiteren als Stimuli $m_j = (x_j, f(x_j))$ bezeichnet werden, die in einer Stimulusmenge M zusammengefasst werden:

$$M = \{m_j = (x_j, f(x_j)) \mid j = 1, ..., \mu\}. \tag{11}$$

Liegt eine Stimulusmenge M vor, so besteht das Problem der ε-Approximation in der Suche einer geeigneten Funktion, welche ein Fehlermaß auf der Basis von M kleiner-gleich ε werden lässt. Durch die Festlegung einer endlichen Stimulusmenge ist die Funktion f(x) in jedem Fall unterdeterminiert, d.h. es existieren unendlich viele Funktionen, die durch die gleiche Stimulusmenge beschrieben werden können.

Approximationsfunktionen lassen sich bezüglich der Rekonstruktion der Stimuli in zwei Klassen unterteilen:

1) Interpolation

   Eine Interpolationsfunktion rekonstruiert alle Stimuli exakt bzw. exakt im Rahmen der verwendeten Rechen- bzw. Maschinengenauigkeit, d.h.

$$\forall m_j \in M: f(x_j)^\wedge = f(x_j). \tag{12}$$

2) Regression

   Eine Regressionsfunktion rekonstruiert die Stimuli nicht notwendigerweise exakt, d.h. es existiert mindestens ein Stimulus, dessen Approximation von dem richtigen Outputwert abweicht,

$$\exists_{\geq 1} m_j \in M: f(x_j)^\wedge \neq f(x_j). \tag{13}$$

   wobei das Zusatzziel der Regression in der Erzeugung einer glatten Approximationsfunktion besteht (s.a. Göppert (1997:5[149])).

Durch die Unterdeterminiertheit einer Approximationsfunktion durch eine endliche Stimulusmenge stellt sich die Frage, wie sich eine Approximationsfunktion in den Bereichen verhält, die nicht durch Stimuli abgedeckt sind. Um diese Eigenschaft zu bewerten, wird das Konzept des Generalisierungsfehlers eingeführt. Die gesamte Stimulusmenge M wird hierzu in eine Lernmenge LM und eine kleinere Testmenge TM zerlegt. Die Approximationsfunktion wird durch einen Adaptionsprozess auf der Basis der Lern-

menge erzeugt, während der Generalisierungsfehler durch die Testmenge bestimmt wird, indem mit einem Qualitätsmaß die Outputvektoren der Approximationsfunktion mit den richtigen Outputvektoren der Stimuli aus der Testmenge verglichen werden.

Die disjunkte bzw. überlappende Bildung von Teilmengen der Testmenge bzw. die Bildung unterschiedlicher Zerlegungen der Stimulusmenge in Lern- und Testmengen bietet durch die Analyse der jeweiligen Generalisierungfehler weitere Möglichkeiten, um das Verhalten einer Approximationsfunktion in unbekannten Regionen des Inputraumes abzuschätzen. Erweiterungen dieser Ansätze führen zu Resampling-Verfahren, mit denen Mengen von Lern- und Teststimuli erzeugt werden (siehe Abschnitt 2.2)).

2.1.2.1) Symbolische und stützpunktorientierte Approximation

Approximationsmodelle lassen sich bezüglich ihrer Struktur in zwei Klassen einteilen:

1) Symbolische Approximation

   Bei einer symbolischen Approximation besteht die Approximationsfunktion in einem Symbolausdruck, d.h. in einer Struktur, deren Elemente aus einer Menge von Symbolen stammen. Ein Beispiel ist die symbolische Regression durch Genetic Programming (Koza (1992[188], 1994[189]), Koza et al. (1999[191]), Banzhaf et al. (1998[26]), Nordin (1997[238]), Francone et al. (1996[119])), bei der z.B. eine Baumstruktur erzeugt wird, deren Verzweigungen Funktionsoperatoren sind, oder bei der eine Sequenz von Maschinenbefehlen erzeugt wird (Nordin (1997[238])). Ein Verfahren aus der Klasse des Genetic Programming dient als Adaptionsverfahren, um im Raum der möglichen Programme einzelne Punkte zu spezifizieren. Diese Form der Approximation soll nicht weiter dargestellt werden, da sie keinen Schwerpunkt dieser Arbeit bildet.

2) Stützpunktorientierte Approximation

   Hierbei werden Stützpunkte $m_{s(i)}$ erzeugt bzw. ausgewählt, d.h. es existieren im Inputraum eine Menge von Vektoren $x_{s(i)}$ und im Outputraum eine Menge korrespondierender Vektoren $y_{s(i)}$, die zusammen einen Stützpunkt im carthesischen Input-Outputraum (X x Y) bilden.

2.1.2.2) Instanz- und prototypbasierte stützpunktorientierte Approximation

Die Approximationsstützpunkte $m_{s(i)} = (x_{s(i)}, y_{s(i)})$ müssen nicht notwendig mit den Stimuli zusammenfallen, sondern können eigene Positionen in (X x Y) belegen, die von einem Adaptionsverfahren aus den Positionen der Stimuli abgeleitet werden. Handelt es sich bei den Approximationsstützpunkten um Stimuli aus M bzw. LM, so wird das stützpunktorientierte Approximationsverfahren als instanzbasiert bezeichnet. Werden aus den Stimuli neue Positionen in (X x Y) erzeugt, so sollen diese Stützpunkte als Prototypen, und das Verfahren als prototypbasierte Approximation bezeichnet werden, die den Schwerpunkt der vorliegenden Arbeit bilden soll.

Ein prototypbasiertes Approximationsmodell AM(x) lässt sich als Tripel beschreiben, mit einer Adaptionsfunktion ad(M), die aus der Stimulusmenge M die Stützpunktmenge S erzeugt, der Stützpunktmenge S selbst und einer Approximationsfunktion f(x | AM(x)), die einen Output zu einem gegebenen Input mit Hilfe der Menge S generiert:

$$AM(x) = (ad(M), S, f(x \mid AM(x))). \quad (14)$$

Ist ein Inputvektor x_R gegeben, zu dem der richtige Output y_R nicht bekannt ist, so kann mit Hilfe des prototypbasierten Approximationsmodells AM(x) eine Schätzung $y_R\hat{}$ des Outputs abgegeben werden:

$$y_R\hat{} = f(x_R \mid AM(x)). \quad (15)$$

Schwerpunkt der weiteren Darstellungen ist die Verwendung eines n-dimensionalen Inputvektors $x \in R^n$ und eines ein-dimensionalen Outputs $y \in R$.

2.1.2.3) Framework des überwachten und unüberwachten Lernens

Eine Approximationsfunktion wird als Abbildung von einem Inputraum X in einen Outputraum Y definiert. Eine Adaptionsfunktion, die aus Stimuli Stützpunkte generiert, kann in diesem Fall als ein überwachtes Lernverfahren betrachtet werden, da mit den Stimuli $m_j = (x_j, f(x_j))$ Lehrervorgaben gegeben sind. Lernen in einem überwachten Zusammenhang besteht in der Erzeugung von Prototypen, die je einen Stützpunkt in X und einen korrespondierenden Stützpunkt in Y besitzen, wobei diese Prototypen bestimmte Fehlermaße wie den mittleren, quadratischen Fehler MSE minimieren sollen.

Ein unüberwachter Lernprozess besitzt demgegenüber nur Zugang zu einem Raum, d.h. es existieren keine Zwei-Tupel, die als Lehrervorgaben verwendet werden könnten, sondern nur Punkte in einem Raum, aus denen Prototypen erzeugt werden sollen, wobei diese Prototypen bestimmte Fehlermaße wie den Quantifizierungsfehler oder den Verteilungsfehler minimieren sollen (Bachelier (1998d[18])).

Überwachtes und unüberwachtes Lernen können ineinander transformiert werden, d.h. eine überwachte Lernsituation kann in eine unüberwachte Lernsituation umgewandelt werden und umgekehrt. Wird im Rahmen des überwachten Lernens das carthesische Produkt der beiden Räume Z = X x Y gebildet, so wandeln sich die zwei-komponentigen Stimuli $(x_j, f(x_j))$ in einen Vektor $z_j \in Z$. Das überwachte Lernen in X x Y wird somit zu einem unüberwachten Lernen in Z transformiert. Auf dieser Transformation basieren die parametrischen Selbstorganisierenden Karten P-SOMs (Walter (1996[356]), Bachelier (1998a[15])).

Eine Form der Umwandlung einer unüberwachten Lernsituation in eine überwachte Lernsituation, ist die Zerlegung eines Vektorraums Z in zwei Vektorräume X und Y, wobei X als Inputraum und Y als Outputraum verwendet wird. Die Zuordnung von Input- und Outputraum zu den beiden Teilräumen ist nicht zwingend, d.h. Y kann ebensogut wie X als Inputraum verwendet werden. In Walter (1996: 46ff[356]) wird gezeigt, dass durch eine diagonale Projektionsmatrix ein Vektorraum Z in zwei beliebige Vektorräume X und Y zerlegt werden kann, wobei eine Abbildung von X nach Y durch ein Verfahren gegeben ist, das auf der Minimierung einer euklidischen Distanz in Z basiert, und das als quasi-kontinuierliche, assoziative Vervollständigung bezeichnet wird (siehe auch Bachelier (1998a: 73f[15])).

Eine andere, für diese Arbeit wichtige Form der Ersetzung einer überwachten Lernsituation, zerlegt das überwachte Lernen in einen unüberwachten Lernprozess im Inputraum und einen nachfolgenden Prozess, der als Batch-Lernen bezeichnet werden kann (siehe auch Bachelier (1998c[17])). Zunächst werden

im Inputraum Prototypstützpunkte erzeugt, d.h. Vektoren $w_i \in X$, die keinen der Positionen x_j der Stimuli entsprechen, was einer Vektorquantifizierung in X entspricht (z.B. Kohonen (1989[184], 1995[185])). In einem Zwischenschritt werden die Stimuli m_j aufgrund ihrer Vektoren x_j klassifiziert, d.h. sie werden Prototypen zugeordnet. Wird eine disjunkte Klassifikation durchgeführt, so wird ein Stimulus m_j genau dem Prototyp w_i zugeordnet, dessen Distanz $d_X(w_i, x_j)$ zu x_j kleiner ist als zu allen anderen Prototypen, mit $d_X(.,.)$ als der Distanzmetrik, die in X verwendet werden soll. Im dritten Schritt wird jedem Prototypen ein Punkt im Outputraum Y zugeordnet, wobei die Outputvektoren der Stimuli genutzt werden, die dem Prototypen zugeordnet wurden, indem eine Aggregationsfunktion verwendet wird. Im einfachsten Fall wird der arithmetische Mittelwert der Outputvektoren der zugeordneten Stimuli verwendet, es sind aber prinzipiell alle Regressions- und Interpolationsverfahren anwendbar. Erweiterte Formen dieser Aggregation verwenden nicht nur die Stimuli eines Prototypen, sondern zusätzlich die Stimuli, die benachbarten Prototypen zugeordnet wurden.

2.1.3) Local-Weighted-Regression

Verfahren, die der Local-Weighted-Regression (LWR) zugeordnet sind, verwenden eine Aggregationsfunktion, die als gewichtete Summe der Outputwerte $y_{s(i)}$ der Stützpunkte berechnet wird. Die Gewichte werden durch die Lage der Inputvektoren der Stützpunkte und des Recall-Inputs festgelegt, d.h. das Gewicht $h(x_R)_{s(i)} \equiv h(x_R, x_{s(i)})$, das einem Stützpunkt $m_{s(i)}$ im Fall des Recall-Inputs x_R zugeordnet wird, ist eine Funktion des Abstandes zwischen x_R und $x_{s(i)}$, wobei eine Distanzmetrik $d_X(.,.)$ gegeben ist. Werden alle Stützpunkte in die Aggregation einbezogen, so gilt für die Schätzung des Outputs an der Stelle des Inputs x_R:

$$y_R\hat{} = agg(h(x_R)_{s(i)}, y_{s(i)} \mid \forall m_{s(i)} \in S) = agg(d(x_R, x_{s(i)}), y_{s(i)} \mid \forall m_{s(i)} \in S). \quad (16)$$

Im weiteren wird die Kernel-Regression als Beispiel für LWR-Verfahren kurz beschrieben (siehe Cohn et al. (1995[74])), während das darauf aufbauende LOESS-Verfahren (Cleveland et al. (1988[68])) im Rahmen dieser Arbeit unberücksichtigt bleiben soll, da in den Experimenten in Verbindung mit aktivem Lernen von Cohn et al. (1995[74]) gezeigt wurde, dass LOESS keine wesentlichen Effektivitätsverbesserungen bietet, gleichzeitig jedoch konzeptuell aufwendiger ist und deutlich mehr Rechenressourcen benötigt.

Bei der Kernel-Regression wird der geschätzte Output $y_R\hat{}$ als Mittelwert aller Outputwerte $y_{s(i)}$ in der Stützpunktmenge bestimmt, gewichtet mit einer radialen Kernel-Funktion, die in x_R ihren Mittelpunkt besitzt. Je weiter ein Stützpunkt $m_{s(i)} \in S$ mit seinem Inputvektor $x_{s(i)}$ von dem zu schätzenden Inputvektor x_R entfernt ist, desto geringer ist der Beitrag seines Outputs $y_{s(i)}$ zur $y_R\hat{}$-Schätzung. Die Form der Kernel-Funktion ist ein Verfahrensparameter, wobei in Cohn et al. (1995[74]) eine Gauss-Funktion verwendet wird:

$$h(x_R)_{s(i)} = \exp(-k * (x_R - x_{s(i)})^2), \quad (17)$$

mit k als einer Konstanten. LOESS (Cleveland et al. (1988[68])) führt eine lineare Regression mit den Stützpunkten durch, gewichtet mit einer radialen Kernel-Funktion, die im zu schätzenden Input ihren Mittelpunkt besitzt.

Die radiale Kernel-Funktion, die in beiden Ansätzen verwendet wird, kann so gewählt werden, dass sie einen Grenzwert größer-gleich Null besitzt, d.h. alle Stützpunkte werden in die Gewichtung einbezogen, unabhängig wie weit ihr Inputvektor vom Vektor x_R entfernt liegt. Je nach der verwendeten Präzision der verwendeten Maschinenzahlen und dem Umgang mit Rundungen kann somit der Fall eintreten, dass eine Vielzahl von Stützpunkten mit einem minimalen Gewicht in die Aggregationsfunktion eingehen, die faktisch keinen Einfluss mehr auf den Wert der Schätzung besitzen, was zu einem hohen Aufwand führt. Beispiel für eine derartige Kernel-Funktion mit dieser Eigenschaft ist die obige Gauss-Funktion. Eine Lokalisierung lässt sich durchführen, indem eine Entfernungsschranke festgelegt wird, außerhalb derer die Stützpunkte ausgeschlossen werden:

$$\forall m_{s(i)} \in S: d(x_R, x_{s(i)}) > d_{max} \Rightarrow h(x_R)_{s(i)} := 0. \quad (18)$$

Bei der Verwendung einer Gauss-Funktion kann die Standardabweichung σ_X im Inputraum oder ein Vielfaches der Standardabweichung für diesen Zweck verwendet werden, wie z.B.:

$$\forall m_{s(i)} \in S: d(x_R, x_{s(i)}) > 2 * \sigma_X \Rightarrow h(x_R)_{s(i)} := 0. \quad (19)$$

Für den weiteren Verlauf müssen die Input- und Output-Mittelwerte, -Varianzen berechnet werden, während bei LOESS zusätzlich die Covarianz und bedingte Varianz explizit angeben werden müssen (Cohn et al. (1995[74])). Es sei ν_R ein Normierungsfaktor mit

$$\nu_R = \Sigma_i h(x_R)_{s(i)}, \quad (20)$$

so berechnet sich der Input-Mittelwert $\mu(x_R)_X$ und Output-Mittelwert $\mu(x_R)_Y$ zu:

$$\mu(x_R)_X := \mu_X = 1/\nu_R \Sigma_i h(x_R)_{s(i)} * x_{s(i)},$$
$$\mu(x_R)_Y = \mu_Y = 1/\nu_R \Sigma_i h(x_R)_{s(i)} * y_{s(i)}. \quad (21)$$

Die Input-Varianz $\sigma(x_R)_X^2$ und Output-Varianz $\sigma(x_R)_Y^2$ berechnen sich zu:

$$\sigma(x_R)_X^2 := \sigma_X^2 = 1/\nu_R \Sigma_i h(x_R)_{s(i)} * (x_{s(i)} - \mu_X)^2,$$
$$\sigma(x_R)_Y^2 := \sigma_Y^2 = 1/\nu_R \Sigma_i h(x_R)_{s(i)} * (y_{s(i)} - \mu_Y)^2. \quad (22)$$

Wird ein Input x_R dem Kernel-Regressions--System präsentiert, so wird ein Output geschätzt:

$$f(x_R \mid KR) := y_R\hat{} = \mu_Y. \quad (23)$$

Die Output-Varianz an der Stelle des geschätzten Outputs $y_R\hat{}$ kann für die Kernel-Regression explizit angegeben werden mit:

$$\sigma(y_R\hat{})^2 = \sigma_Y^2/\nu_R. \quad (24)$$

2.1.4) Sensorische-SOM (S-SOM)

Bei den sensorischen Kohonen-Karten (S-SOM, vgl. z.B. Kohonen (1989[184], 1995[185]), Ritter et al. (1991: 67ff[282]), (Bachelier (1998a: 11ff[15])), handelt es sich um ein unüberwachtes Lernverfahren, bei dem eine endliche Menge M von Stimuli m_i gegeben ist, die durch einen n-dimensionalen Vektor x_i spezifiziert sind:

$$M = \{m_i = (x_i) \mid x_i \in R^n, i = 1, ..., \mu_M\}. \quad (25)$$

Weiterhin sei eine Menge von μ Neuronen gegeben, denen jeweils ein Gewichtsvektor w_i und eine Position k_i auf einer Kohonen-Karte zugeordnet wird, die im weiteren als zwei-dimensional angenommen wird, wobei die Positionen auf der Karte diskreten Gitterpunkten entsprechen, und jeder Gitterpunkt durch ein Neuron besetzt ist:

$$N = \{n_i = (w_i, k_i) \mid w_i \in R^n, k_i \in N^2, i = 1, ..., \mu_N\}. \quad (26)$$

Der Lern- bzw. Abbildungsprozess wählt aus der Menge der Stimuli zum Präsentationszeitpunkt t ein Element $m_j^t = (x_j^t)$ aus. Dieser Stimulus legt auf der Karte ein Neuron n_s^t fest, das als Erregungszentrum bezeichnet wird. Es handelt sich dabei um genau das Neuron, dessen Gewichtsvektor w_s^t den geringsten Abstand im Raum V zu dem Stimulusvektor x_j^t besitzt, wobei eine Metrik $d_W(.,.)$ des Inputraumes gegeben sei:

$$d_W(x_j^t , w_s^t) = \min\{d_W(x_j^t, w_i^t) \mid i = 1, ..., \mu_N\}. \quad (27)$$

Nachdem n_s^t ermittelt wurde, werden alle Neurone n_i^t bestimmt, die neben dem Erregungszentrum ihre Gewichtsvektoren anpassen dürfen. Es handelt sich dabei um die Neurone, deren Entfernung $d_A(k_s, k_i)$ auf der Karte nicht größer ist als ein zeitabhängiger Schwellenwert, der als Entfernungsreichweite σ^t bezeichnet wird. Diese Neurone werden in einer Menge N^{+t} zusammengefasst mit:

$$N^{+t} = \{n_i^t = (w_i^t, k_i) \mid d_A(k_s, k_i) \leq \sigma^t\}. \quad (28)$$

Im folgenden Adaptionsschritt wird auf alle Neurone aus N^{+t} ein Lernschritt angewendet, der die Gewichtsvektoren verändert. Der Lernschritt ist interpretierbar als eine Verschiebung der Gewichtsvektoren in Richtung des Stimulusvektors (siehe Abb. 5)).

Es wird entsprechend dem Modell von Ritter et al. (1991[282]) dabei die folgende Adaptionsregel verwendet:

$$w_i^{t+1} = w_i^t + \varepsilon^t h_{si}^t (x_j^t - w_i^t), \text{ mit} \quad (29)$$

den zeitabhängigen Parametergleichungen ε^t und h_{si}^t, die festgelegt werden als:
1) Die zeitabhängige Lernrate ε^t:

$$\varepsilon^t = \varepsilon_{start} * (\varepsilon_{end}/\varepsilon_{start})^{t/t(max)}, \text{ mit} \quad (30)$$

der Startlernrate ε_{start} und ε_{end} als der Lernrate zum Ende des Verfahrens, d.h. nach t_{max} Stimuluspräsentationen.

2) Die zeitabhängige Entfernungsgewichtungsfunktion h_{si}^t:

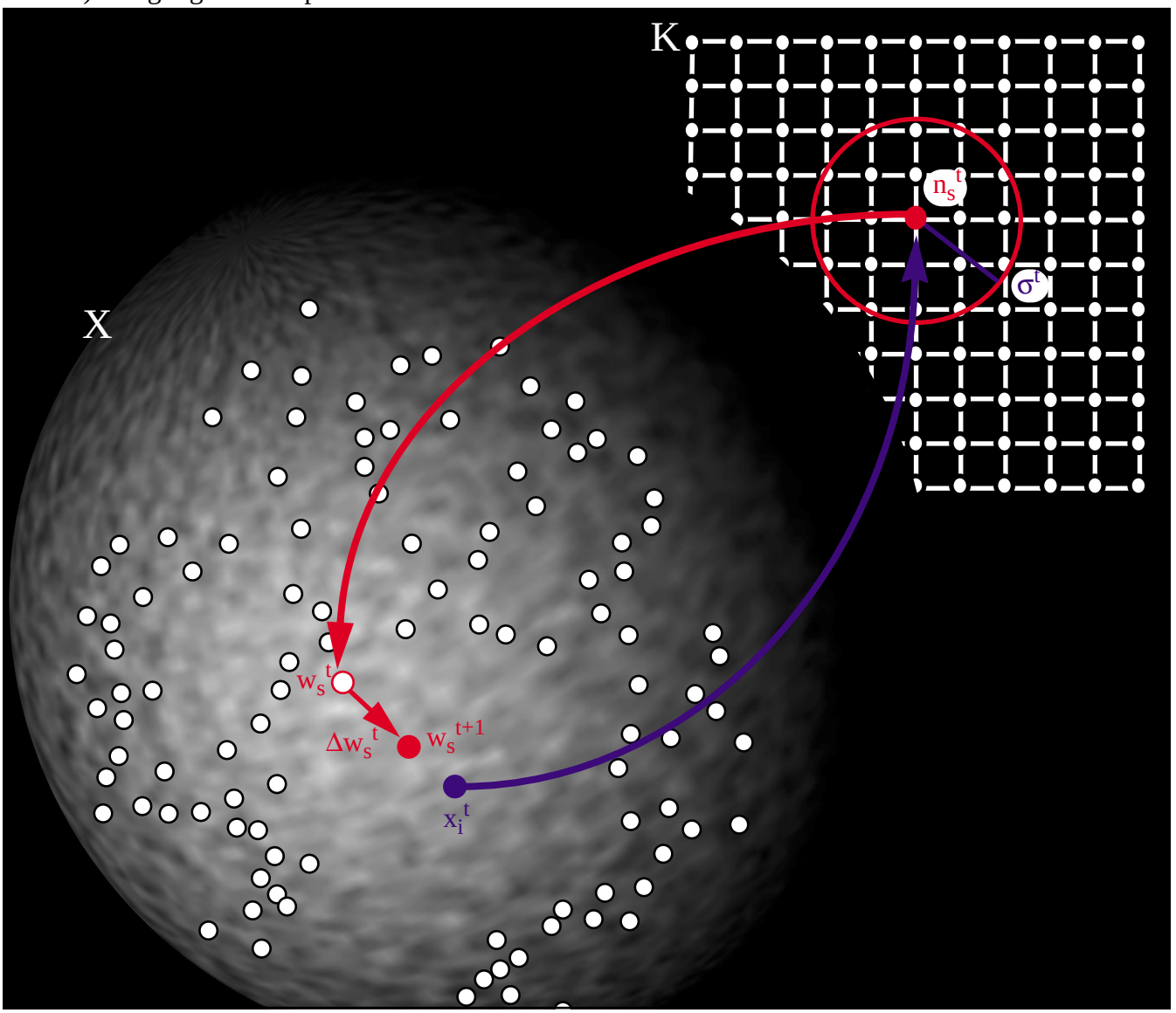
$$h^t_{si} = \exp(-d_A(k_s,k_i)^2/2\sigma^{t2}), \text{ mit} \quad (31)$$

σ^t als dem Nachbarschafts- oder Adaptionsradius um das Gewinner-Neuron auf der Karte:

$$\sigma^t = \sigma_{start} * (\sigma_{end}/\sigma_{start})^{t/t(max)}, \text{ mit} \quad (32)$$

dem Adaptionsradius σ_{start} zum Anfang des Verfahrens, und σ_{end} als dem Adaptionsradius zum Ende des Verfahrens.

Abb. 5) Erregung und Adaption beim sensorischen Kohonen-Modell



Damit eine topologie-erhaltende Abbildung entsteht, müssen zwei Faktoren berücksichtigt werden:

a) Die topologische Nachbarschaft h^t_{si} um das Erregungszentrum muss anfangs groß gewählt werden und im Laufe des Verfahrens verkleinert werden.

b) Die Adaptionsstärke ε^t muss ausgehend von einem großen Wert im Laufe des Verfahrens auf einen kleinen Restwert sinken.

2.1.5) Growing-Neural-Gas (GNG-SOM)

Bei der Growing-Neural-Gas-Self-Organizing-Map (GNG-SOM, Fritzke (1995[130], 1996[131], 1998[132]), Bachelier (1998a[15])) handelt es sich um ein prototypbasiertes unüberwachtes Lernverfahren, das zu einem überwachten Verfahren erweitert werden kann. Überwachte Varianten sind beispielsweise die RBF-GNG-SOM (Radial-Basis-Function-GNG-SOM) oder die LLM-GNG-SOM (Local-Linear-Mapping-GNG-SOM). Im weiteren soll zunächst das zugrundeliegende unüberwachte Verfahren beschrieben werden.

Die GNG-SOM gehört wie die GCS-SOM (Growing-Cell-Structures-SOM, Fritzke (1992[128])) zu den wachsenden Modellen, d.h. die Neuronenanzahl ist nicht wie bei einer S-SOM als externer Parameter gegeben, sondern es findet ein Wachstumsprozess statt, der durch lokale Fehlerwerte E_i^t geregelt wird, die jedem Neuron zugeordnet sind. Nach einer Adaptionsphase, in der eine gewisse Anzahl von Stimuluspräsentationen mit Gewichtsvektoradaptionenen durchgeführt werden, wird eine Wachstums- oder Einfügephase eingeschoben, in der standardmäßig genau ein neues Neuron erzeugt wird, indem die einzelnen Komponenten wie Gewichtsvektor, Verbindungsstruktur und Fehlerwert initialisiert werden.

Bei der GNG-SOM wird ein Verbindungsgraph zwischen den Neuronen auf der Basis der Gewichtsvektoren aufgebaut (Fritzke (1995[130], 1996[131], 1998[132])), wobei den einzelnen Verbindungen ein Alter zugeordnet werden muss. Jedem Neuron n_i^t wird ein Verbindungsvektor C_i^t zugeordnet, der beschreibt, mit welchen anderen Neuronen eine Verbindung besteht, wobei ein Wert von $c_{ij}^t = 0$ bedeutet, dass die beiden Zellen nicht verbunden sind, und $c_{ij}^t > 0$ bedeutet, dass sie seit c_{ij}^t Zeiteinheiten verbunden sind. Weiterhin soll gelten, dass das Verbindungsalter nicht älter als eine Schwelle T sein darf, bei deren Überschreitung die Verbindung gelöscht wird, d.h. der Verbindungsvektor eines Neurons n_i^t besitzt die Struktur

$$C_i^t = (c_{ij}^t \mid c_{ij}^t \in \{0, 1, ..., T\}, j = 1, ..., \mu_N^t). \tag{33}$$

Die Neuronenstruktur besitzt bei einer GNG-SOM zum Zeitpunkt t somit die folgende Darstellung:

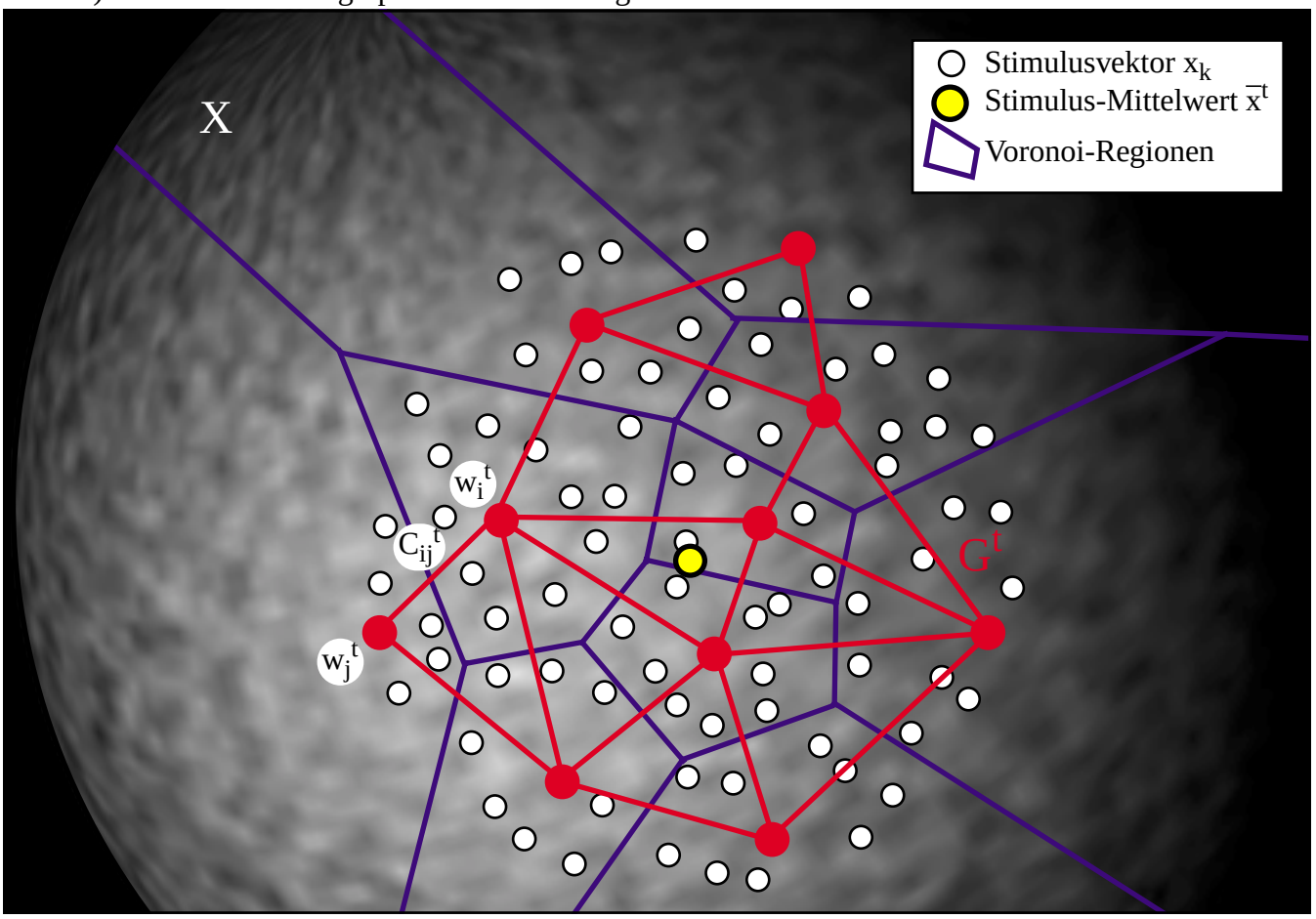
$$N^t = \{n_i^t = (w_i^t, C_i^t, E_i^t) \mid w_i^t \in X \subseteq R^n, C_i^t = (c_{ij}^t \mid j = 1, ..., \mu^t), E_i^t \in R^+; i = 1,..., \mu_N^t\}. \tag{34}$$

Durch das Adaptionsverfahren der GNG-SOM entsteht im Vektorraum X ein Gewichtsvektorengraph G^t, der alle Gewichtsvektoren w_i^t und Verbindungskanten C_{ij}^t umfasst, und der als Delaunay-Triangulation interpretiert wird (Martinetz (1992:136ff[212]), siehe Abb. 6)). Eine Karte mit eigener Nachbarschaftsstruktur wie bei den S-SOM existiert demgegenüber bei den GNG-SOM nicht.

Korrespondierend zu der Delaunay-Triangulation ist die Voronoi-Zerlegung von X, als einer disjunkten Clusterung der Punkte aus X. Eine Voronoi-Region $\tilde{R}_i^t$, die zu dem Gewichtsvektor w_i^t korrespondiert, wird definiert als die Menge der Punkte x aus X, die zu w_i^t einen kleiner-gleichen Abstand besitzen als zu allen anderen Gewichtsvektoren der Neurone n_q^t aus N^t:

$$\tilde{R}_i^t = \{x \in X \mid d_X(w_i^t, x) \leq d_X(w_q^t, x), \forall n_q^t \in N^t\}. \tag{35}$$

Abb. 6) Gewichtsvektorengraph und Voronoi-Regionen einer GNG-SOM



Initialisiert wird der Graph G^t mit zwei Zellen, deren Gewichtsvektoren zufällig initialisiert sind, die unverbunden sind und deren Fehlerwert Null ist. Nach der Stimuluswahl wird das Gewinner-Neuron und das zweitbeste Neuron ermittelt, gefolgt von der Korrektur der Verbindungsstruktur, d.h. die beiden Zellen werden verbunden, bzw. ihr Alter auf 1 gesetzt. Nach der Adaption des Gewinner-Neurons und seiner verbundenen Nachbarn, sowie der Anpassung des Fehlerwertes des Gewinner-Neurons, wird geprüft, ob nach einer bestimmten Anzahl von Adaptionsschritten eine Einfügephase begonnen werden soll. Ist dies der Fall, dann wird das Neuron n_q^t mit dem größten Fehlerwert als primäres Einfügezentrum und dessen Nachbar n_f^t mit dem ebenfalls größten Fehlerwert als sekundäres Einfügezentrum ermittelt. Ein neues Neuron n_r^t wird eingefügt, indem zuerst der Gewichtsvektor w_r^t als Mittelwert der beiden ermittelten Zellen initialisiert wird (siehe Abb. 7)). Eine Reihe von alternativen Wachstumsmethoden eines Gewichtsvektorengraphen findet sich in Bachelier (1999c: 66ff[21]), worauf hier nicht näher eingegangen werden soll.

Danach wird die Verbindungsstruktur aktualisiert, indem die Dimension des Verbindungsvektors bei allen Neuronen um Eins erhöht wird, die Verbindung zwischen w_q^t und w_f^t gelöscht wird, und neue Verbindungen zwischen w_q^t - w_r^t, w_f^t - w_r^t sowie zwischen w_r^t und allen gemeinsam verbundenen Gewichtsvektoren zwischen w_q^t und w_f^t aufgebaut werden (siehe Abb. 8)).

Abb. 7) Einfügen eines neuen Gewichtsvektors in den Gewichtsvektorengraphen

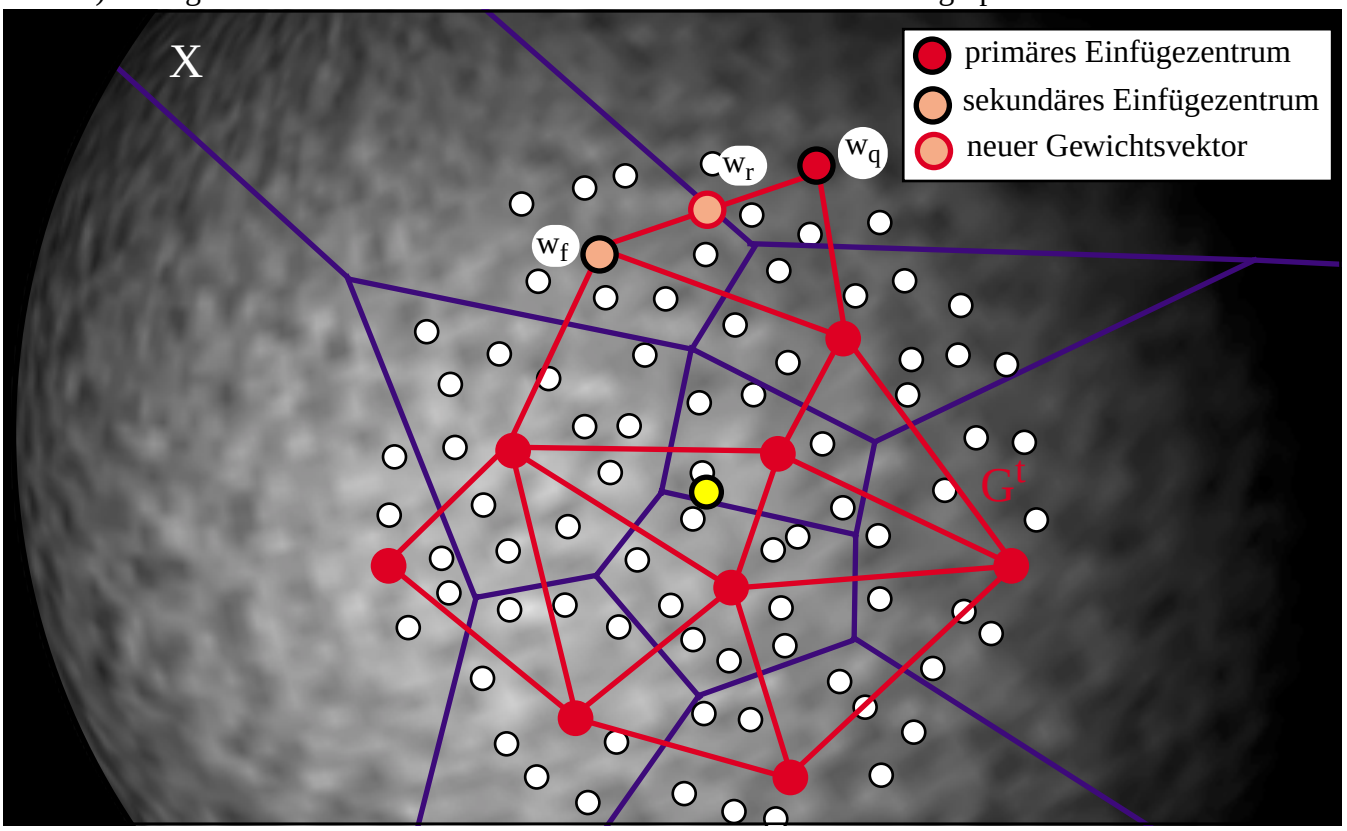
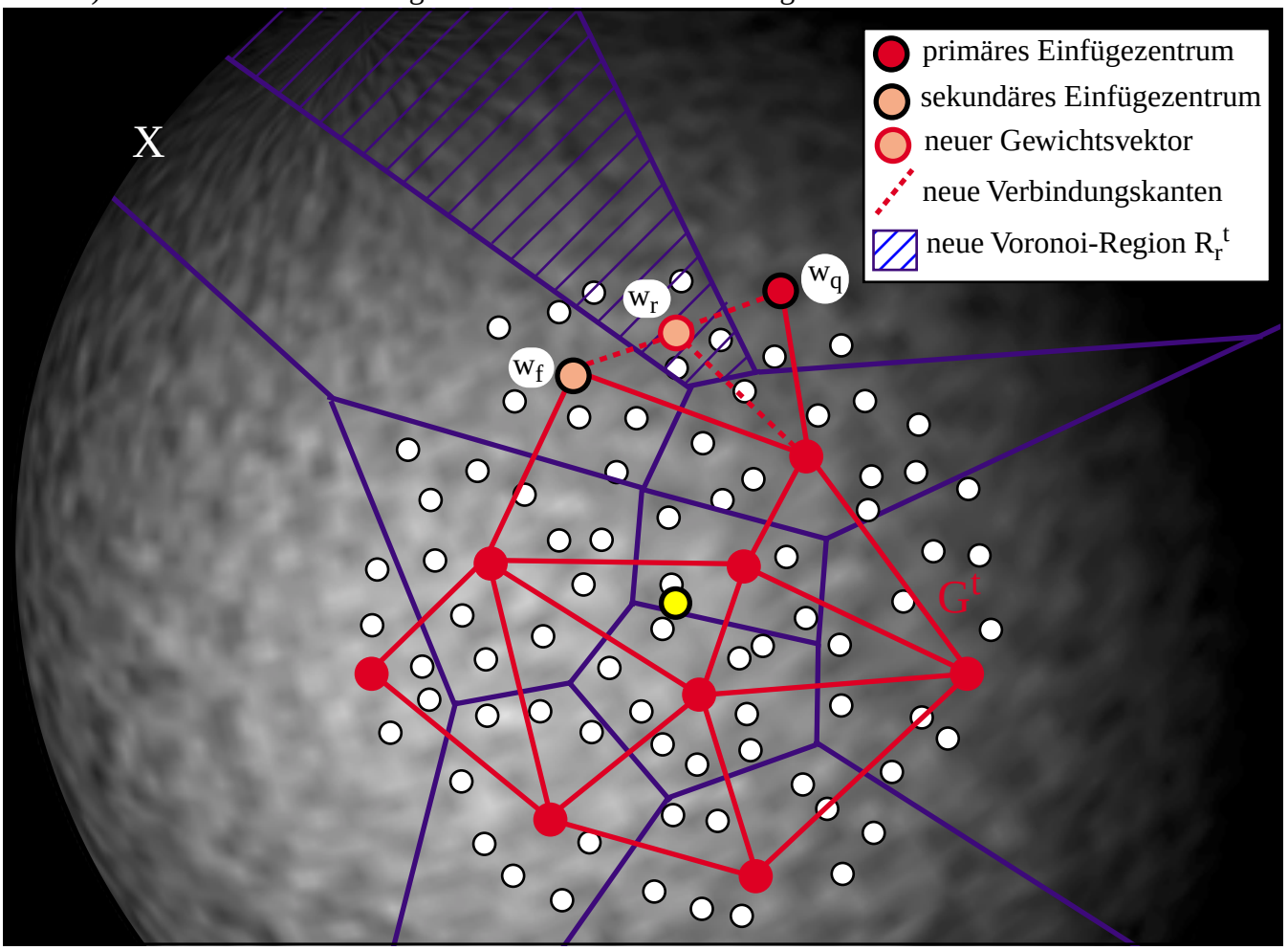


Abb. 8) Korrektur der Verbindungsstruktur und der Voronoi-Regionen



Danach werden die Fehlerwerte aktualisiert, indem E_q^t und E_f^t mit einem konstanten Faktor $\alpha \in]0, 1[$ multipliziert werden, und E_r^t mit dem neuen Wert von E_q^t(neu) initialisiert wird, wobei in Fritzke (1995[130]) der Wert $\alpha = 0{,}5$ gegeben ist. Unabhängig, ob eine Einfüge-Phase durchgeführt wurde oder nicht, werden danach die Fehlerwerte aller existierenden Neurone verringert, indem sie mit einem konstanten Faktor $\delta \in]0, 1[$ multipliziert werden, der in Fritzke (1995[130]) als $\delta = 0{,}0995$ angegeben wird. Ist ein Abbruchkriterium nicht erfüllt, dann wird der nächste Stimulus ausgewählt.

Das GNG-Modell unterscheidet sich von dem GCS-Modell (Fritzke (1992[128])) dadurch, dass auf die komplexe Umverteilung der Fehlerwerte (Signalzähler) mit Hilfe der Schätzungen der Voronoi-Volumina vor und nach der Adaption verzichtet wird. Statt dessen wird der Fehlerwert der beiden ausgewählten Zellen einfach durch Multiplikation mit einem konstanten Faktor verringert, und der Fehlerwert der neuen Zelle einfach mit dem von n_q^t initialisiert.

Weiterhin wird die Auswahl des sekundären Einfügezentrums w_f^t nicht nach der weitesten Entfernung von w_q^t, sondern nach dem maximalen Fehlerwert getroffen. In Fritzke (1994[129]) wird erwähnt, dass das Auswahlkriterium des zweiten Neurons eine untergeordnete Bedeutung besitzt, und dass selbst eine zufällige Auswahl keine negativen Auswirkungen besitzt, da sich unterschiedliche Einordnungen nach einigen Adaptionsschritten angleichen.

Der Graph G^t, der nach dem Abbruch der Adaption und des Wachstums vorliegt, ist eine topologische Modellierung der Stimulusvektorenverteilung. Im Gegensatz zu einer S-SOM findet jedoch keine explizite Dimensionsreduzierung statt, da keine Karte K mit einer vordefinierten, im Vergleich zu R^n kleinen Dimension, vorliegt. Der Graph G^t liegt in Form einer Verbindungsmatrix vor, wobei die Knoten ohne den Bezug zu den Gewichtsvektoren in X ein rein topologisches Modell ohne Bezug zu einem metrischen Raum wie X ist. Das rein topologische Modell ist wesentlich speichereffizienter, da die Kodierung der n-dimensionalen Gewichtsvektoren entfällt. Eine implizite Dimensionsreduzierung kann durchgeführt werden, indem das topologische Modell durch ein Graph-Drawingverfahren in einen nieder-dimensionalen, metrischen Raum abgebildet wird.

2.1.6) Stimulus-Cluster-GNG-SOM (SC-GNG-SOM)

Die Grundidee der Stimulus-Cluster besteht darin, während der Lernphase einen Stimulus $x_j^t \in M$ temporär dem Neuron $n_{s(1)}^t$ zuzuordnen, das als Gewinner-Neuron ermittelt wird (Bachelier (1998c[17])). Auf diese Weise wird einem Neuron n_i im Verlauf einer Lernphase ein oder mehrere Stimuli zugeordnet, die in einem Stimulus-Cluster M_i^t zusammengefasst werden. Die Anzahl der Stimuli pro Cluster ist dabei abhängig von der Anzahl der Neurone und Stimuli, sowie der momentanen Position des Gewichtsvektors und der Stimulus-Verteilung.

Durch diese temporäre Clusterung lassen sich lokale Lernverfahren anwenden, nachdem alle vorhandenen Stimuli mindestens einmal präsentiert und in eine Lernmenge eingefügt wurden. Mit diesen lokalen Lernverfahren wird dem größten Effizienzproblem der SOM begegnet, und zwar dem Aufwand, den die Suche nach dem Gewinnerneuron $n_{s(1)}^t$ verursacht. Standardmäßig wird eine enumerative Suche nach

dem Gewinnerneuron in der jeweils vorliegenden Neuronenmenge durchgeführt, wobei für jedes Neuron die Distanz seines Gewichtsvektors zu dem Inputvektor des Stimulus berechnet werden muss, was in Abhängigkeit von der Metrik $d_X(.,.)$ und der Anzahl der Dimensionen des Inputraumes X sehr aufwendig sein kann. Durch die Einführung eines wachsenden Modells wird eine erhebliche Effizienzverbesserung erreicht, da die Anzahl der Neurone erst langsam ansteigt, doch in der Endphase der Adaption sowie in der Recallphase ergeben sich auch bei wachsenden Modellen die gleichen Probleme, wenn eine globale Präsentation der Stimuli verwendet wird.

Bei einem GNG-Modell gibt es demgegenüber die Möglichkeit, die Verbindungsstruktur für eine Graphensuche zu nutzen (Bachelier (1998a: 44ff[15])), bei der zu jedem Bezugspunkt die unmittelbare Nachbarschaft nach einem näher liegenden Neuron abgesucht wird. Auf diese Weise existiert das Potential, das bei einer Präsentation nicht alle Neurone geprüft werden müssen. Es besteht jedoch das Problem, dass je nach der Art des Graphensuchverfahrens ein lokales Distanzminimum gefunden und verwendet wird, d.h. ein Neuron, in dessen unmittelbarer Nachbarschaft, die durch die Graphen-Distanz $d_G = 1$ angegeben wird, sich keine näherliegende andere Neurone befinden, wobei jedoch in einer größer gewählten Nachbarschft ein oder mehrere andere, näher liegende Neurone existieren. Sinnvoll sind Graphensuchverfahren nur bei einer nicht zu großen Verbindungsdichte anzuwenden, da bei einer großen Verbindungsdichte ein großer Prozentsatz von Neuronen als Nachbarn pro Bezugspunkt geprüft werden müssen, sodass wenige Bezugspunktwechsel genügen, um die gesamte Neuronenmenge zu prüfen.

In Bachelier (1998c[17]) wurden einige Varianten der SC-GNG-SOMs beschrieben, wobei sich die weiteren Darstellungen auf ein Basismodell beziehen, das als Mischverfahren einer Grob- und einer Feinadaption beschrieben wird, wobei die Datenstruktur der Neurone durch die Einführung lokaler Stimulusmengen M_i^t gegenüber einer GNG-SOM erweitert wird zu:

$$N^t = \{n_i^t = (w_i^t, M_i^t, C_i^t, E_i^t) \mid M_i^t = \{x_j \mid j = 1, ..., \mu_{S,i}^t\}) \mid i = 1, ..., \mu_N^t\}. \quad (36)$$

Es soll von einer disjunkten Stimulus-Clusterung ausgegangen werden, d.h. ein Stimulus ist zu jedem Zeitpunkt Element höchstens einer Stimulusmenge eines Neurons:

$$\forall n_i^t, n_j^t \in N^t: M_i^t \cap M_j^t = \varnothing. \quad (37)$$

Es soll von einem zweiphasigen Verfahrensaufbau ausgegangen werden. Bei der ersten Phase handelt es sich um die Grobadaption, bei der ein GNG-SOM-Modell aufgebaut wird, indem die einzelnen Stimuli zufällig je einmal global präsentiert und in die Lernmenge des Gewinners aufgenommen werden. Begonnen wird mit zwei Initialisierungsneuronen, wobei ein schnelles Wachstum durchgeführt wird, d.h. nach einer, im Vergleich zu der Stimulusanzahl geringen Anzahl von globalen Präsentationen mit Adaptionsoperationen, wird eine Einfügephase eingeschoben, in der ein neues Neuron initialisiert wird. Durch das schnelle Wachstum soll nach dem Abbruch der Grobadaption, d.h. wenn alle Stimuli einem Neuron zugeordnet wurden, eine hinreichend große Neuronenmenge und Verbindungsstruktur vorliegen, damit die nachfolgende Feinadaption schwerpunktmäßig die Zuordnung der Stimuli zu Neuronen und die Adaption der Gewichtsvektoren durchführen kann, ohne dass größere Veränderungen in der Verbindungsstruktur und größere Wachstumsprozesse berücksichtigt werden müssen.

Bei der Feinadaption werden die Stimuli in der Lernmenge eines Bezugspunktneurons lokal präsentiert, d.h. aus den Neuronen in einer bestimmten Nachbarschaft wird das Gewinnerneuron ermittelt. Kann von einer hinreichend großen Neuronenanzahl und einer hinreichend dichten Verbindungsstruktur ausgegangen werden, so kann als Größe der Nachbarschaft einer lokalen Präsentation insbesondere die unmittelbare Nachbarschaft $N(d_G \leq 1 \mid N^t)$ des Bezugspunktes gewählt werden, wovon bei dem betrachteten Basismodell ausgegangen wird.

Bezugspunkte können durch lokale und globale Verfahren bestimmt werden, wobei jeweils ein Fehlermaß, wie der Quantifizierungsfehler des Neurons (siehe auch Bachelier (1998d: 30f[18])), bezogen auf seine lokale Lernmenge, als Entscheidungskriterium verwendet wird. Bei einem lokalen Verfahren wird als nachfolgender Bezugspunkt das Neuron in der unmittelbaren Nachbarschaft des momentanen Bezugspunktes mit dem größten Fehlerwert gewählt, sodass die Feinadaption als Trajektorie oder Bewegung von benachbarten Bezugspunkten innerhalb der Verbindungsstruktur betrachtet werden kann. Bei einem globalen Verfahren wird demgegenüber jeweils das Neuron mit dem größten Fehler in der gesamten Neuronenmenge ausgewählt. Als Basismodell soll das globale Verfahren verwendet werden.

Während der Feinadaption kann es zu einem neuen Verbindungsaufbau kommen, wenn eine Standard-Präsentation auf der Nachbarschaftsmenge durchgeführt wird, d.h. wenn zwei Gewinnerneurone ermittelt werden, die bislang noch unverbunden sind. Ein Verbindungsaufbau ist jedoch dann ausgeschlossen, wenn für ein Stimulus geprüft wird, ob es aus der Lernmenge des momentanen Bezugspunktes in eine der Lernmengen der Neurone aus der betrachteten Nachbarschaft umverteilt werden soll. Der Aufwand der beiden Präsentationen ist vergleichbar, da in beiden Fällen für einen Stimulus $\#N(d_G \leq 1 \mid N^t)$ Distanzbestimungen durchgeführt werden müssen. Bei der lokalen Standard-Präsentation müssen zusätzlich die beiden ersten Plätze der Distanzrangfolge bestimmt werden, während beim zweiten Präsentationstyp $\#N(d_G = 1 \mid N^t)$ Größenvergleiche von je zwei Distanzen durchgeführt werden müssen. Da bei der lokalen Standard-Präsention die prinzipielle Möglichkeit des Verbindungsaufbaus besteht, wird sie im betrachteten Basismodell gewählt.

Nachdem alle Stimuli in der Lernmenge eines Bezugspunktes geprüft, und einige möglicherweise umverteilt wurden, wird eine Summenadaption durchgeführt, indem für alle Neurone, denen mindestens ein Stimulus neu zugeteilt wurde, der Mittelwert der Inputvektoren der zugeordneten Stimuli gebildet wird, der als Adaptionszentrum für die Adaption des Gewichtsvektors verwendet wird. D.h. der Gewichtsvektor wird entsprechend der GNG-Adaptionsgleichung in Richtung des Mittelwertvektors verschoben.

Wachstumsprozesse können in der Feinadaptionsphase ausgeklammert werden, bis eine fehlerfreie Zuordnung von Stimuli zu Lernmengen erfolgt ist, sodass ein entsprechendes SC-GNG-SOM-Verfahren optional dreiphasig wird, wenn nach der Feinadaptions- eine Wachstumsphase durchgeführt wird. Ist in diesem Fall ein Abbruchkriterium, wie das Maximum der Quantifizierungsfehler aller vorhandener Neurone noch größer als ein Schwellenwert, so wird durch Einfügeoperationen versucht, die Quantifizierungsleitung der Gewichtsvektorenverteilung zu verbessern.

Ein Übergang von der Feinadaptions- zur Wachstumsphase kann durch eine Umverteilungsvariable UV^t geregelt werden, die bei jedem Lernschritt t ermittelt, ob mindestens eines der Stimuli in der Lernmenge $LM_m{}^t$ des Bezugspunktes $n_m{}^t$ umverteilt wurde. Existiert ein Nachbarneuron $n_l{}^t$, dessen Stimuluszwischenmenge Z_l nach der lokalen Präsentation aller Stimuli aus $LM_m{}^t$ nicht leer ist, so wurde mindestens ein Stimulus umverteilt, und UV^t wird auf 1 gesetzt, sonst auf Null. Wenn während der letzten p_{max} Bezugspunktwechel keine Umverteilungen durchgeführt wurden, dann soll zur Wachstumsphase gewechselt werden, wobei in nachfolgendem Algorithmus genau eine Einfügeoperation durchgeführt werden soll. Teil der Einfügeoperation ist auch die Prüfung der Umverteilung der Stimuli in den Lernmengen der Neurone, die mit dem neuen Neuron verbunden werden, gefolgt von entsprechenden Summenadaptionen.

Nach der Einfüge-, Umverteilungs- und Adaptionsoperationen in der Wachstumsphase, wird wiederum das Abbruchkriterium geprüft. Ist der Quantifizierungsfehler des Neurons mit dem global größten Fehler kleiner als ein Schwellenwert, so wird das Gesamtverfahren beendet, ansonsten wird wieder in die Feinadaption umgeschaltet, bzw. es wird eine neue Einfügeoperation durchgeführt. Es sind auch Metastrategien verwendbar, die nach einer Einfügeoperation solange einen Wechsel zwischen Wachstums- und Feinadaptionsphase durchführen, bis das Abbruchkriterium erfüllt ist. Auf diese Verfahren soll jedoch nicht weiter eingegangen werden.

2.1.7) Batch-Lernen in einer SC-GNG-SOM

Durch die SC-GNG-SOM wird eine Menge von Prototypen in Form von Neuronen $n_i{}^t$ erzeugt, denen durch die Vektorquantifizierung in X Positionen $w(x)_i{}^t := w_i{}^t$ und eine Menge $M_i{}^t$ von Stimuli $m_{ij}{}^t$ zugeordnet wurden:

$$N^t = \{n_i{}^t = (w(x)_i{}^t, M_i{}^t, C_i{}^t) \mid i = 1, ..., \mu_N{}^t\}. \tag{38}$$

D.h. im Inputraum X werden die Stimuli $m_{ij}{}^t$ durch ihre Inputvektoren $x_{ij}{}^t$ in Cluster gruppiert. Besitzen die Stimuli zusätzlich eine Outputkomponente $y_{ij}{}^t$ als Element eines Outputraumes Y, d.h. wird als Ziel ein überwachtes Lernen verfolgt, so kann die Neuronenstruktur N^t derart erweitert werden, dass daraus ein stützpunktbasiertes, prototypbasiertes Approximationsmodell entsteht. Faktisch bedeutet diese Vorgehensweise eine Zerlegung des überwachten Lernens in einen unüberwachten und einen überwachten Anteil, wie dies als Batch-Lernen in dem gemeinsamen Framework von überwachtem und unüberwachtem Lernen in Abschnitt 2.1.2.3) dargestellt wurde.

Entsprechend dem Ansatz des Batch-Lernens muss jedem Neuron neben seinem Stützpunkt $w(x)_i{}^t$ in X ein Stützpunkt $f(w(x)_i{}^t)\hat{}$ in Y zugeordnet werden, wobei hierzu eine Reihe von Verfahren verwendet werden können, die jeweils die lokal zugeordneten Stimuli in $M_i{}^t$ nutzen:
1) Verwendung des arithmetischen Mittelwertes der Outputkomponenten der Stimuli in $M_i{}^t$:

$$f(w(x)_i{}^t \mid MW, M_i{}^t) = f(w(x)_i{}^t)\hat{} = 1/\#M_i{}^t \sum_j y_{ij}{}^t, \forall m_{ij}{}^t \in M_i{}^t. \tag{39}$$

2) Verwendung einer LWR wie z.B. der Kernel-Regression (KR):

$$f(w(x)_i^t \mid KR, M_i^t) = f(w(x)_i^t)\hat{} = 1/\nu_i \sum_j \exp(-\alpha\,(x_{ij}^t - w(x)_i^t)^2) * y_{ij}^t, \text{ mit}$$
$$\nu_i = \sum_j \exp(-\alpha\,(x_{ij}^t - w(x)_i^t)^2),\ \forall m_{ij}^t \in M_i^t. \quad (40)$$

3) Verwendung einer diskreten oder stetigen y-Häufigkeitsverteilung.
Zunächst wird die lokale Stimulusmenge M_i^t analysiert, wobei vorausgesetzt sein soll, dass darin genügend Elemente enthalten sind. Im ersten Schritt wird der Stimulus mit dem kleinsten y-Wert $y_{i,min}^t$ und der Stimulus mit dem größten y-Wert $y_{i,max}^t$ bestimmt, die zusammen ein y-Intervall IM_i^t der Stimulusmenge M_i^t bilden:

$$IM_i^t = [y_{i,min}^t, y_{i,max}^t], \text{ mit}$$
$$y_{i,min}^t = \min\{y_{ij}^t \mid \forall m_{ij}^t \in M_i^t\},\ y_{i,max}^t = \max\{y_{ij}^t \mid \forall m_{ij}^t \in M_i^t\}. \quad (41)$$

Das Gesamtintervall wird in γ gleich breite Teilintervalle $IM(l)_i^t$, $l = 1, ..., \gamma$, zerlegt, das den Durchmesser Δy_i^t besitzt:

$$\Delta y_i^t = (y_{i,max}^t - y_{i,min}^t)/\gamma. \quad (42)$$

Es ergibt sich die geordnete Liste der Teilintervalle:

$$IML_i^t = (IM(l)_i^t \mid l = 1, ..., \gamma), \text{ mit}$$
$$IM(1)_i^t = [y_{i,min}^t, y_{i,min}^t + \Delta y_i^t[,$$
$$IM(2)_i^t = [y_{i,min}^t + \Delta y_i^t, y_{i,min}^t + 2 * \Delta y_i^t[,$$
$$...,$$
$$IM(\gamma)_i^t = [y_{i,min}^t + (\gamma\text{-}1)\,\Delta y_i^t, y_{i,max}^t]. \quad (43)$$

Für jedes der Teilintervalle $IM(l)_i^t$ wird die Menge $M(l)_i^t$ der Stimuli aus M_i^t ermittelt, deren y-Wert innerhalb des Teilintervalls liegt:

$$M(l)_i^t = \{m_{ij}^t \in M_i^t \mid y_{ij}^t \in IM(l)_i^t\}. \quad (44)$$

Die relative Anzahl der Individuen in einem Teilintervall $IM(l)_i^t$ soll mit s(.) bezeichnet werden, d.h. die absolute Anzahl $\#M(l)_i^t$ wird durch die Gesamtzahl der Elemente in M_i^t dividiert:

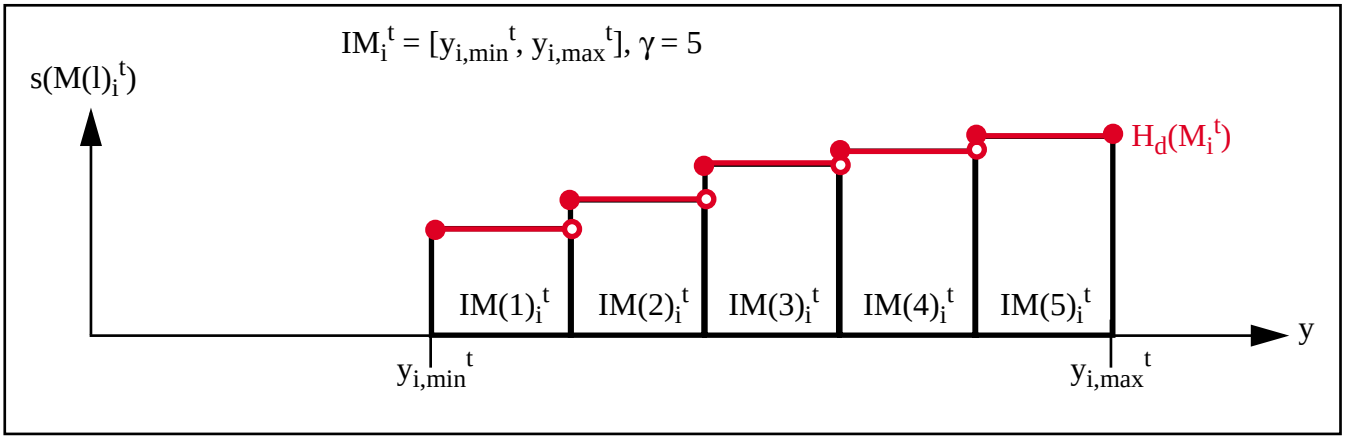
$$s(M(l)_i^t) := \#M(l)_i^t/\#M_i^t. \quad (45)$$

Die Häufigkeitsfunktion $H_d(.)$ ordnet jedem Teilintervall $IM(l)_i^t$ die relative Anzahl $s(M(l)_i^t)$ der Individuen in der entsprechenden Teilstimulusmenge $M(l)_i^t$ zu:

$$H_d(M_i^t): IM(l)_i^t \rightarrow s(M(l)_i^t). \quad (46)$$

Wird jedem Punkt eines Intervalls der s(.)-Wert zugeordnet, so gelangt man zu einer punktweise nicht stetigen Funktion, wobei die Grenzen so gewählt sind, wie die Teilintervalle $IM(l)_i^t$ in der geordneten Liste IML_i^t (siehe Abb. 9))

Abb. 9) Punktweise nicht stetige y-Häufigkeitsfunktion

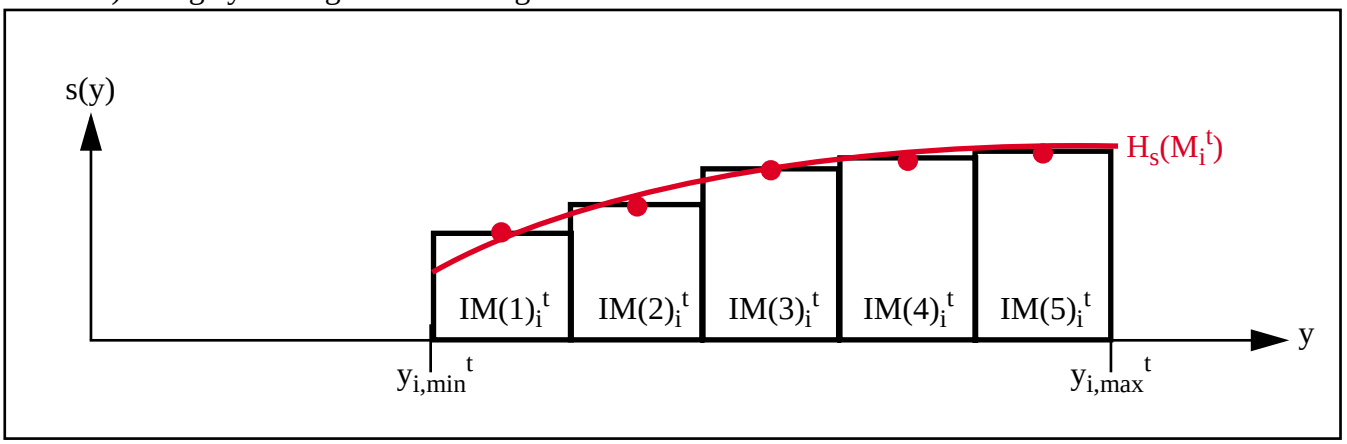


Durch die Verwendung der relativen Anzahl $s(M(l)_i^t)$ der Elemente bezogen auf die Gesamtanzahl $\#M_i^t$ ist das Integral über $H_d(M_i^t)$ in den Grenzen $y_{i,min}^t$ und $y_{i,max}^t$ gleich Eins, sodass die Häufigkeitsfunktion als Wahrscheinlichkeitsdichte interpretiert werden kann.

Eine punktweise nicht stetige Dichtefunktion wird in eine geglättete, über der gesamten Definitionsmenge $[y_{i,min}^t, y_{i,max}^t]$ stetige Dichtefunktion umgewandelt, indem Stützpunkte ausgewählt werden, die als Input in ein Glättungs- oder ein Regressionsverfahren verwendet werden. D.h. zunächst wird eine Anzahl von Outputwerten aus dem Intervall $[y_{i,min}^t, y_{i,max}^t]$ spezifiziert, denen jeweils ihr s(.)-Wert zugeordnet wird. Sinnvoll ist es, aus jedem Teilintervall genau ein Wert zu spezifizieren, da alle Punkte innerhalb eines Teilintervall den gleichen s(.)-Wert besitzen. Denkbar wäre es, die Mittelpunkt der Teilintervalle auszuwählen und ihnen den s(.)-Wert des Teilintervalls zuzuordnen (siehe Abb. 10)). Die Mitte eines y-Intervalls $IM(l)_i^t$ soll mit $IM(l)_i^t/2 = y_{i,min}^t + (l-1) * \Delta y_i^t + 1/2 * \Delta y_i^t = y_{i,min}^t + (l-1/2) * \Delta y_i^t$ bezeichnet werden sodass dem Intervall $IM(l)_i^t$ der Stützpunkt $(IM(l)_i^t/2, s(M(l)_i^t))$ zugeordnet wird. Mit Hilfe der γ Stützpunkte und einem Regressionsverfahren wird eine stetige Dichtefunktion $H_s(M_i^t)$ erzeugt, die jedem Outputwert y aus dem Intervall IM_i^t einen relativen Häufigkeitswert s(y) zuordnet, wobei gelten soll, dass die Dichtefunktion monoton steigend oder monoton fallend über dem Intervall $IM_i^t = [y_{i,min}^t, y_{i,max}^t]$ sein soll:

$$H_s(M_i^t): y \rightarrow s(y),\ y \in IM_i^t. \tag{47}$$

Abb. 10) Stetige y-Häufigkeitsverteilung



Der wichtigste Constraint bei der Regression ist jedoch die Forderung, dass das Integral der Regressionsfunktion $H_s(M_i^t)$ in den Grenzen $[y_{i,min}^t, y_{i,max}^t]$ ebenfalls gleich bzw. näherungsweise gleich Eins sein muss. Auf diese Weise muss ein Optimierungs-Verfahren angewendet werden, um die Regressionsparameter diesem Constraint anzupassen, was diese Vorgehensweise sehr aufwendig macht.

Der arithmetische Mittelwert einer stetigen, monoton steigenden Outputfunktion $H_s(M_i^t)$ ergibt sich mit Hilfe des Integrals über dem Gesamtintervall $[y_{i,min}^t, y_{i,max}^t]$, wobei dieser Wert als Schätzung $f(w(x)_i^t)\wedge$ des unbekannten Output-Stützpunktes $f(w(x)_i^t)$ verwendet werden kann:

$$f(w(x)_i^t)\wedge = 1/[y_{i,max}^t - y_{i,min}^t] * \int_{y(i,min) \rightarrow y(i,max)} H_s(M_i^t)\, dy. \tag{48}$$

4) Verwendung eines Resampling-Verfahrens wie dem Paar-Bootstrap.
Als Vorgriff auf Abschnitt 2.2) soll ein Resampling-Verfahren angedeutet werden, mit dem man die Outputkomponente an der Stelle eines Gewichtsvektors $w(x)_i^t$ schätzen kann, wobei ein einfacher Paar-Bootstrap verwendet werden soll. Aus der Stimulusmenge M_i^t werden hierzu $\delta := \#M_i^t$ künstliche Stimuluslisten M_{ik}^t durch Ziehen mit Zurücklegen aus M_i^t erzeugt, wobei die Anzahl der Elemente in den Listen der Anzahl der Elemente in M_i^t entspricht. Die entstehenden Bootstraplisten können in der Menge B_i^t zusammengefasst werden:

$$B_i^t = \{M_{ik}^t \mid k = 1, ..., \delta\}, \text{ mit}$$
$$M_{ik}^t = (m_{ikj} \in M_i^t \mid j = 1, ..., \#M_i^t). \tag{49}$$

Die Verwendung eines Resampling-Verfahrens erfordert ein Verfahren, mit dem die y-Werte der Stimuli in einer Bootstrapliste zu einem Schätzwert aggregiert werden, d.h. faktisch handelt es sich dabei um kein eigenständiges Verfahren, sondern um ein Meta-Verfahren, das zusammen mit einem der drei vorangegangenen Verfahren verbunden werden muss. Wird die y-Schätzung, die auf der Basis der Bootstrapliste erzeugt wird als $f(w(x)_i^t)_k\wedge$ bezeichnet, so wird im darauf folgenden Schritt eine Aggregation dieser Schätzwerte zu einer Bootstrap-Schätzung mit Hilfe des arithmetischen Mittelwertes durchgeführt:

$$f(w(x)_i^t \mid B_i^t) = f(w(x)_i^t)\wedge = 1/\delta \sum_j f(w(x)_i^t)_k\wedge, \forall M_{ik}^t \in B_i^t. \tag{50}$$

Eine Bootstrap-Schätzung kann allgemein verbessert werden, indem man eine bias-korrigierte Schätzung durchführt, bei der zunächst eine Schätzung durch das Gesamtmodell durchgeführt wird. Im konkreten Fall bedeutet dies, dass alle Elemente aus einer lokalen Stimulusmenge M_i^t eine Schätzung $f(w(x)_i^t \mid M_i^t)$ erzeugen, indem ein arithmetischer Mittelwert oder ein LWR-Verfahren angewendet wird. Die Bias an der Stelle eines beliebigen Punktes wird definiert als Schätzung durch das Gesamtmodell minus dem richtigen Wert, wobei jedoch im Fall des Gewichtsvektors kein richtiger Wert $f(w(x)_i^t)$ bekannt ist:

$$bias(w(x)_i^t) = f(w(x)_i^t \mid M_i^t) - f(w(x)_i^t). \tag{51}$$

Aus diesem Grunde muss eine Bias-Schätzung durchgeführt werden, wobei im nachfolgenden Vorschlag eine Schätzung mit Hilfe aller Stimuli aus M_i^t erfolgen soll. Für jeden Stimulus m_{ij}^t kann ein richtiger Biaswert bestimmt werden, indem zunächst eine Schätzung $f(x_{ij}^t)\wedge$ durchgeführt wird, z.B. mit Hilfe aller Stimuli aus $M_i^t \setminus \{m_{ij}^t\}$ und einer Kernel-Regression:

$$f(x_{ij}^t \mid KR, M_i^t \setminus \{m_{ij}^t\}) = f(x_{ij}^t)\hat{} = 1/\nu_{ij} \sum_k \exp(-\alpha\,(x_{ij}^t - x_{ik}^t)^2) * y_{ik}^t, \text{ mit}$$
$$\nu_{ij} = \sum_k \exp(-\alpha\,(x_{ij}^t - x_{ik}^t)^2),\ \forall m_{ik}^t \in M_i^t \setminus \{m_{ij}^t\}. \quad (52)$$

Die Bias des Stimulus ergibt sich somit zu:

$$bias(x_{ij}^t) = f(x_{ij}^t \mid KR, M_i^t \setminus \{m_{ij}^t\}) - f(x_{ij}^t). \quad (53)$$

Eine Möglichkeit der direkten Bias-Schätzung an der Stelle des Gewichtsvektors $w(x)_i^t$ besteht nun darin, eine Kernel-Regression auf der Basis der Biaswerte der Stimuli aus M_i^t durchzuführen:

$$bias(w(x)_i^t)\hat{} = 1/\nu_i \sum_j \exp(-\alpha\,(x_{ij}^t - w(x)_i^t)^2) * bias(x_{ij}^t), \text{ mit}$$
$$\nu_i = \sum_j \exp(-\alpha\,(x_{ij}^t - w(x)_i^t)^2),\ \forall m_{ij}^t \in M_i^t. \quad (54)$$

Mit dieser Bias-Schätzung kann die Bootstrap-Schätzung $f(w(x)_i^t \mid B_i^t)$ korrigiert werden, indem eine Bias-korrigierte Bootstrap-Schätzung durchgeführt wird (Efron & Tibshirani (1993[105])). Hierzu wird die Differenz von Bootstrap-Schätzung $f(w(x)_i^t \mid B_i^t)$ und Bias-Schätzung $bias(w(x)_i^t)\hat{}$ gebildet:

$$f(w(x)_i^t \mid B_i^t)_{bias} = f(w(x)_i^t \mid B_i^t) - bias(w(x)_i^t)\hat{}. \quad (55)$$

Ein alternativer Ansatz erzeugt zwei unterschiedliche Schätzungen und verwendet diese für die Bootstrap-Schätzung und den richtigen Wert $f(w(x)_i^t)$, wobei eine 0,632-Korrektur durchgeführt werden sollte (Efron & Tibshirani (1993[105])). Wird als Bootstrap-Schätzung $f(w(x)_i^t \mid B_i^t)$ verwendet und als Substitut für den richtigen Wert $f(w(x)_i^t)$ eine Schätzung durch alle verfügbaren Stimuli, d.h. $f(w(x)_i^t \mid KR, M^t)$, so ergibt sich als Bias-Schätzung:

$$bias(w(x)_i^t)_{0,632}\hat{} = (f(w(x)_i^t \mid B_i^t) - f(w(x)_i^t \mid KR, M^t))/0{,}632. \quad (56)$$

Diese führt zu einer 0,632-bias-korrigierten Schätzung:

$$f(w(x)_i^t \mid B_i^t)_{0,632bias} = f(w(x)_i^t \mid B_i^t) - bias(w(x)_i^t)_{0,632}\hat{}. \quad (57)$$

Unabhängig von dem verwendeten Verfahren zur Output-Schätzung $f(w(x)_i^t)\hat{}$ an der Stelle des Gewichtsvektors $w(x)_i^t$ erweitert sich die erweiterte Neuronenstruktur zu:

$$N^t = \{n_i^t = (w(x)_i^t, f(w(x)_i^t)\hat{}, M_i^t, C_i^t) \mid i = 1, ..., \mu_N^t\}. \quad (58)$$

Soll ein Recall-Stimulus $m_k = (x_k, \_)$ mit einem unbekannten Outputwert y_k verarbeitet werden, so wird für ihn eine Output-Schätzung $f(x_k \mid KR)$ erzeugt. Bei der KR auf der Basis der Gewichtsvektoren werden die Neurone aus N^t als Stützpunkte verwendet, wobei alle Neurone an der Regression beteiligt werden:

$$f(x_k \mid KR, N^t) = 1/\nu_k \sum_i \exp(-\alpha(w(x)_i^t - x_k)^2) * f(w(x)_i^t)\hat{}, \text{ mit}$$
$$\nu_k = \sum_i \exp(-\alpha(w(x)_i^t - x_k)^2),\ \forall n_i^t \in N^t. \quad (59)$$

Eine Form der Lokalisierung ergibt sich, wenn beispielsweise die ersten k Plätze der Distanz-Rangfolge der Neurone ihre Gewichtsvektoren in die Regression einbringen, wobei diese Neurone in die Gewinnerliste $G(x_k^t \mid k, N^t)$ der Präsentation von x_k überführt werden:

$$f(x_k \mid KR, G(x_k^t \mid k, N^t)) = 1/\nu_\kappa \Sigma_i \exp(-\alpha(w(x)_i^t - x_k)^2) * f(w(x)_i^t)\wedge,$$
$$\nu_k = \Sigma_i \exp(-\alpha(w(x)_i^t - x_k)^2), \forall n_i^t \in G(x_k^t \mid k, N^t). \quad (60)$$

Die Bezeichnung LWR-SC-GNG-SOM kann auf zwei Verfahrenstypen beschränkt werden:
1) Verfahren, die für die y-Schätzungen der Gewichtsvektoren und für die y-Schätzung von Recall-Stimuli auf der Basis der Gewichtsvektoren ausschließlich LWR-Verfahren einsetzen.
2) Verfahren, die für die y-Schätzung von Recall-Stimuli auf der Basis der Gewichtsvektoren ausschließlich LWR-Verfahren einsetzen, unabhängig wie die y-Schätzungen der Gewichtsvektoren erzeugt werden.

2.1.8) Aktivitätsausbreitung in GNG-Graphen

Im weiteren soll der Fall einer Aktivitätsausbreitung in einem GNG-Graphen G^t betrachtet werden, wobei G^t als ein Feedback-Netz betrachtet wird, in dem sich Aktivierungen entsprechend der Verbindungsstruktur lateral ausbreiten, im Gegensatz zu Spreading-Activation-Prozessen in Semantischen Netzen (siehe z.B. Collins & Loftus (1975[76]), Lee & Dubin (1999[197])).

Lateral bedeutet in diesem Zusammenhang, dass alle Operationen bis auf die Initialisierung und das Auslesen der sich ergebenden Aktivierungen nach einer Anzahl von Iterationen innerhalb der Verbindungsstruktur ablaufen. Bei nicht-lateralen Operationen würden Aktivitätswerte von Aussen zugeführt werden. Wird dies einmal durchgeführt, so besitzt diese Operation den Charakter eines Feedforward-Netzes. Die Präsentation eines Stimulus und die Ermittlung des Gewinner-Neurons kann als ein Beispiel einer nicht-lateralen, nicht-lokalen Operation interpretiert werden, indem alle Neurone in Abhängigkeit von ihrer Distanz zu dem Stimulusvektor ihre Aktivität berechnen. Diese ist bei SOMs gleich der Distanz, d.h. es wird die Indentitätsfunktion zwischen Distanz und Aktivität verwendet. Diese Aktivitätsberechnung ist nicht-lokal, da alle Neurone sie gleichzeitig durchführen, unabhängig wie die laterale Verknüpfung der Neurone ist. Diese Operation ist auch nicht-lateral, da keine Ausbreitung über die Verbindungsstruktur erfolgt, bei der die Neurone mehrmals ihren Aktivitätszustand ändern können. Die Aktivitätsverteilung der Neurone kann als Repräsentation des Stimulus verwendet werden, was jedoch erst interessant wird, wenn andere Aktivitätsfunktionen anstatt der Indentitätsfunktion verwendet werden.

Nachfolgend soll demgegenüber die Aktivitätsausbreitung als eine laterale Operation betrachtet werden, die in der Recall-Phase stattfindet, d.h. nach dem Abbruch der Wachstums- und Adaptionsprozesse. Durch die Aktivitätsausbreitung soll die Struktur des Graphen nicht verändert werden, d.h. die Positionen der Gewichtsvektoren und somit der Verbindungskanten, sowie die Verbindungsvektoren der einzelnen Neurone bleiben konstant. Als Neuronentyp sollen Neurone aus den SC-GNG-SOM betrachtet werden, die während des Wachstums und der Adaption die Struktur $n_i^t = (w_i^t, M_i^t, C_i^t, E_i^t)$ besitzen. Nach dem Abbruch der Wachstums- und Adaptionsprozesse wird der lokale Fehlerwert E_i^t auf der Basis der Sti-

muli, die sich in der lokalen Stimulusmenge M_i^t befinden, nicht mehr benötigt, sodass von einer Grundstruktur $n_i^t = (w_i^t, M_i^t, C_i^t)$ ausgegangen wird. An die Stelle des Fehlerwertes tritt ein lokaler Aktivitätswert z_i^t, der im Rahmen der Aktivitätsausbreitung verändert wird, und der als reeller Wert definiert sein soll, d.h. es werden positive wie negative Aktivitätswerte zugelassen:

$$N^t = \{n_i^t = (w_i^t, M_i^t, C_i^t, z_i^t) \mid M_i^t = \{x_j \mid j = 1, ..., \mu_{S,i}^t\}, z_i^t \in R) \mid i = 1, ..., \mu_N^t\}. \quad (61)$$

Vor der Phase der Aktivitätsausbreitung sollen alle Aktivitätswerte den Wert von Null bzw. eine Mindestaktivität besitzen, die nicht weitergeleitet werden kann. Eine Mindestaktivität hätte in Verbindung mit der Definition von z_i^t auf R den Nachteil, dass das Vorzeichen beachtet werden muss, da eine Mindestaktivität als positiver wie als negativer Wert verwendet werden kann. Aus diesem Grunde soll als Grundzustand vor der Aktivitätsausbreitung Aktivitätswerte von Null vorliegen.

Initialisiert wird eine Aktivitätsausbreitung indem einigen oder allen Neuronen aus N^t ein von Null abweichender Aktivitätswert zugeordnet wird. Die Ausbreitung von Aktivitäten erfolgt ausschließich über die gegebenen Verbindungskanten, d.h. es wird nur eine lokale und schrittweise Ausbreitung erlaubt. Die Verbindungen und die Verbindungsstärken sollen vereinfachend als symmetrisch betrachtet werden, d.h. eine Übertragung von w_i^t nach w_j^t über die Verbindungskante C_{ij}^t soll die gleiche Wirkung besitzen, wie eine gleich starke Übertragung von w_j^t nach w_i^t über die Verbindungskante C_{ji}^t.

Wie oben angedeutet, soll die Aktivitätsausbreitung in einem GNG-Graphen als Operation in einem Feedback-Netz betrachtet werden, sodass den einzelnen Neuronen zusätzliche Eigenschaften zugeordnet werden müssen. Ein formales Neuron n_i kann als 5-Tupel dargestellt werden, mit dem Gewichtsvektor w_i, der Inputfunktion net_i, der Aktivitätsfunktion z_i, der Outputfunktion $S(z_i) = y_i$ und einer Lernregel Δw_i (siehe auch Bachelier (1995: 23[14])):

$$n_i = (w_i, net_i, z_i, S(z_i), \Delta w_i). \quad (62)$$

Im Zusammenhang mit einem GNG-Graphen muss diese Darstellung angepasst werden. Zunächst muss festgehalten werden, dass diese allgemeine Darstellung eines Neurons den Fall einschließt, dass alle Neurone andere Input-, Aktivitäts- und Outputfunktionen besitzen können. Bei einer Aktivitätsausbreitung in G^t sollen alle Neurone die gleiche Funktionsform besitzen, sodass die Funktionsbeschreibungen net, z(net) und S(z) Komponenten sind, die dem Netz N^t nicht, aber einzelnen Neuronen n_i^t zugeordnet sind. Auf die Funktionsformen wird weiter unten eingegangen. Zu unterscheiden ist die Funktion von Werten, d.h. jedem einzelnen Neuron ist zu einem Zeitpunkt ein Inputwert net_i^t, ein Aktivitätswert z_i^t und ein Outputwert $S(z_i^t)$ zugeordnet. Als Inputfunktion wird standardmäßig

$$net_i^{t+1} = \sum_j c_{ij} * S(z_{ij}^t). \quad (63)$$

verwendet, wohingegen die Aktivierungsfunktion einen Schwellenwert T_i verwendet:

$$z_i^{t+1} = net_i^{t+1} - T_i. \quad (64)$$

Es können eine Reihe von Outputfunktionen verwendet werden, wie z.B. die binäre Outputfunktion des McCulloch & Pitts-Modells

$$S(z_i^{t+1}) = \{1, \text{wenn } z_i^{t+1} \leq 0, \quad \{0, \text{wenn } z_i^{t+1} > 0, \tag{65}$$

oder eine sigmoide Outputfunktion mit dem Parameter k:

$$S(z_i^{t+1}) = 1/[1 + \exp(k * z_i^{t+1})]. \tag{66}$$

Der Gewichtsvektor w_i in der obigen Darstellung muss uminterpretiert werden, da er nicht die Position innerhalb des Vektorraums X darstellt, sondern einen Vektor von Gewichtungsfaktoren, die jeweils eine Verbindungskante zu einem benachbarten Neuron repräsentieren. Der Verbindungsvektor C_i soll Distanzgewichtungsfaktoren c_{ij} enthalten, die eine monoton fallende Funktion von $d_X(w_i^t, w_j^t)$ darstellen, d.h. je länger die Kante ist, desto weniger Aktivität wird übertragen, sodass eine Form von Widerstand modelliert wird, der bei einer elektrischen Leitung auftritt. Die entsprechende Funktion cd_X soll für alle Neurone aus N^t gleich und im Zeitverlauf konstant sein, sodass diese Funktionsbeschreibung Teil von N^t nicht aber Teil der einzelnen Neurone ist. Da sich die Verbindungsstruktur von G^t während der Aktivitätsausbreitung nicht verändern soll, bleiben die Distanzfaktoren c_{ij} ebenfalls konstant, d.h. anstatt einer iterationsabhängigen Darstellung C_i^t ist eine iterationsunabhängige Darstellung C_i sinnvoll. Die Distanzgewichtungsfunktion cd_X kann grundsätzlich z.B. als Potenzfunktion $c_{ij} = a * d_X^{-b}$ gebildet werden, wobei beachtet werden muss, dass eine Distanz von Null ausgeschlossen wird. Es kann auch die Möglichkeit betrachtet werden, eine Distanzgewichtungsfunktion für einen speziellen Graphen G^t aufzustellen, der eine duchschnittliche Gewichtsvektorendistanz von $\overline{d}_{X,max}$ mit einer Varianz von $var(d_X)$, und eine maximale Gewichtsvektorendistanz von $d_{X,max}$ besitzt.

Die gleiche iterationsunabhängige Darstellung ergibt sich für den Gewichtsvektor w_i aus X und die Stimulusmenge M_i, da beide Komponenten während der Recall-Phase, in der die Aktivitätsausbreitung stattfindet, konstant bleiben. Aus dem gleichen Grunde soll eine Lernregel Δw_i in der Neuronenstruktur nicht vorhanden sein.

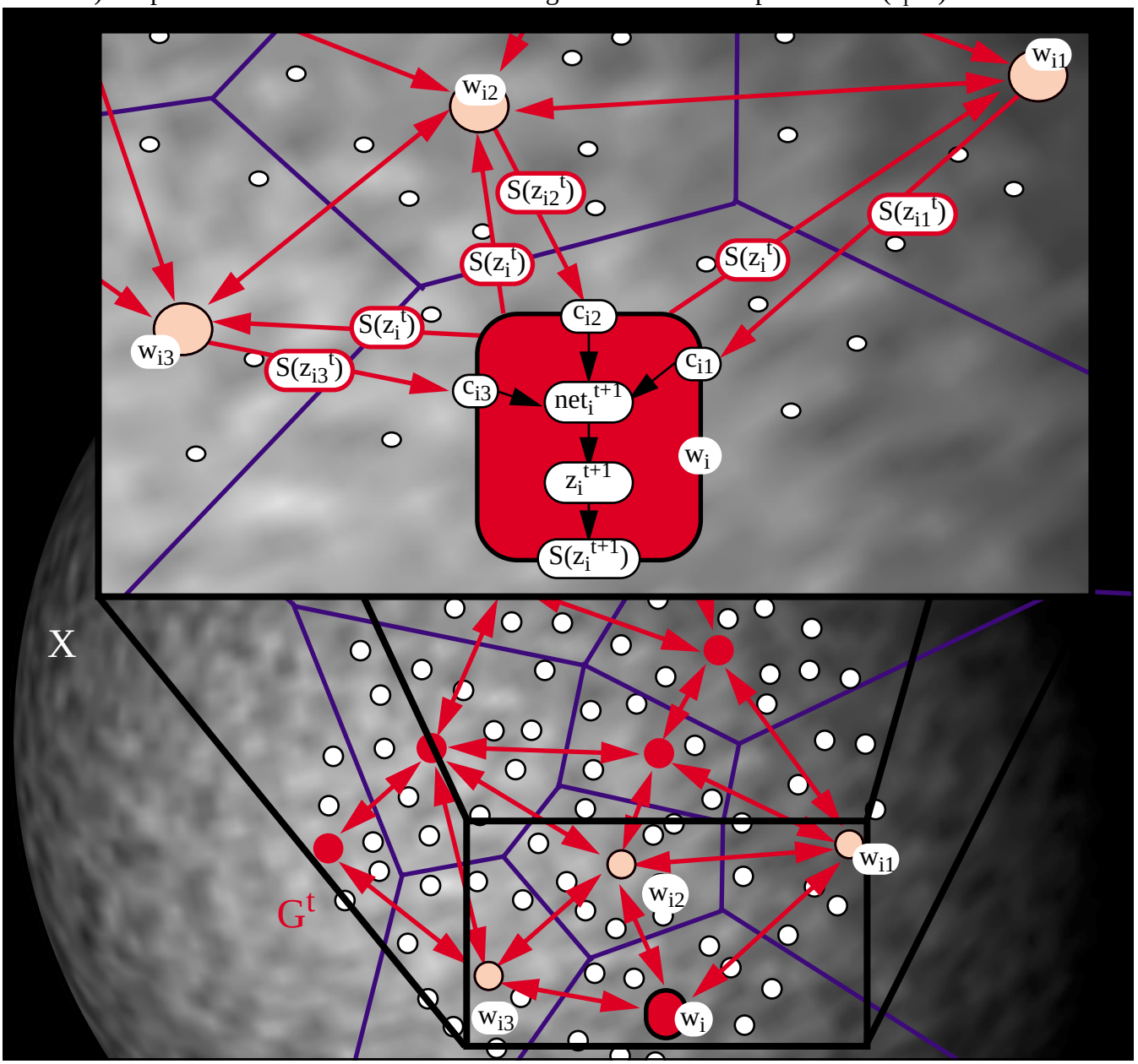
Um ein Verfahren zur Aktivitätsausbreitung in G^t abschließend zu spezifizieren, muss ein Abbruchkriterium festgelegt werden, wie z.B. eine maximale Anzahl t_{max} der Iterationsschritte bzw. der Abbruch bei Erreichen eines Punkt-Attraktors, d.h. wenn weitere Iterationen keine Veränderung an den Aktivitätswerten aller Neurone ergibt. Wird eine sigmoide Outputfunktion und t_{max} Iterationsschritte verwendet, so ergibt sich eine Neuronenstruktur für ein Netz N_{AA}^t, dessen GNG-Graph G^t laterale Aktivitätsausbreitungen durchführen kann, wobei der Index „AA“ für Aktivitätsausbreitung stehen soll:

$$N_{AA}^t = \{n_i^t = (w_i, M_i, C_i, net_i^t, T_i, z_i^t, S(z_i^t)) \mid cd_X, net, z(net), k, S(z), t_{max}; i = 1, ..., \mu_N\}. \tag{67}$$

In Abb. 11) wird eine Iteration der Aktivitätsausbreitung aus der Sicht des Gewichtsvektors w_i^t dargestellt, d.h. der GNG-Graph soll sich in der Iteration t+1 befinden, in der allen Gewichtsvektoren Aktivitätswerte z_i^t zugeordnet sind, und die Werte z_i^{t+1} sollen berechnet werden. Der Gewichtsvektor w_i^t besitzt drei verbundene Nachbarn, w_{i1}^t, w_{i2}^t , w_{i3}^t, von denen er je einen Outputwert $S(z_{i1}^t)$, $S(z_{i2}^t)$, $S(z_{i3}^t)$ erhält, d.h. der Inputvektor von w_i ist drei-dimensional. Mit Hilfe der Inputfunktion net(.) wird der Inputvektor zu einem skalaren Wert net_i^{t+1} umgewandelt, wobei die einzelnen Inputwerte mit den Distanzfaktoren c_{ij}^t gewichtet und aufsummiert werden. Der net_i^{t+1}-Wert geht danach in die Aktivie-

rungsfunktion ein, die daraus den skalaren Wert z_i^{t+1} erzeugt, der wiederum in die Outputfunktion eingeht, die daraus $S(z_i^{t+1})$ erzeugt. Parallel zu dem Empfangen der Outputwerte der verbundenen Nachbarn sendet n_i^t seinen Outputwert $S(z_i^t)$ der vorangegangenen Iteration an seine drei Nachbarn, die zusammen mit den anderen Werten ihrer Nachbarn ihre Aktivitätswerte zu z_{i1}^{t+1}, z_{i2}^{t+1}, z_{i3}^{t+1} aktualisieren.

Abb. 11) Outputwerte-Austausch und Bestimmung des aktuellen Outputwertes $S(z_i^{t+1})$



Bei der Aktivitätsausbreitung im Rahmen einer SC-GNG-SOM müssen zwei Szenarien unterschieden werden, wobei angenommen werden soll, dass eine Stimulus-Clusterung vorliegt, d.h. in den Voronoi-Regionen der einzelnen Gewichtsvektoren liegen in der Regel mehrere Inputvektoren von Stimuli, anstatt einer Triangulation der Inputvektoren, bei der jedem Stimulusvektor ein Gewichtsvektor mit der gleichen Position zugeordnet wurde:

1) Im Rahmen der Initialisierung wird den Neuronen eine Aktivierung zugeordnet, die sich über den GNG-Graphen ausbreiten soll.

2) Im Rahmen der Initialisierung wird den Stimuli eine Aktivierung zugeordnet, die sich über den GNG-Graphen ausbreiten soll.

Dem ersten Szenario entspricht Abb. 11), d.h. ein Neuron n_i^t besitzt seine Stimulusmenge M_i, die während den Prozessen der Aktivitätsausbreitung konstant bleibt, und die in der Regel mehr als ein Element beinhaltet. Es stellt sich die Frage, wie das Verfahren der Aktivitäzsausbreitung ablaufen soll, wenn bei der Initialisierung den Neuronen nicht direkt ein Aktivierungswert $z_i^{t=0}$ zugeordnet wird, sondern wenn den Stimuli ein Aktivierungswert zugeordnet wird. Die Beantwortung dieser Frage ist notwendig, da dies den Normalfall betrifft, wenn Informationsobjekte wie Merkmale oder Dokumente durch SC-GNG-Modelle repräsentiert werden (siehe Abschnitt 3.8)).

Gelöst werden soll dieses Problem mit den vorhandenen Mitteln der Aktivitätsausbreitung, d.h. mit den vorhandenen Funktionen cd_X, net, z(net) und S(z), indem eine virtuelle Verbindung zwischen den Stimulivektoren x_{ij} und dem Gewichtsvektor w_i eingeführt wird, und indem die Stimuli quasi als Neurone behandelt werden, die auf globaler Ebene eine Input-, Aktivitäts- und Outputfunktion besitzen. Konkret bedeutet dies, dass ein Stimulus m_j eine Position x_j in X besitzt und eine Referenz zu dem Neuron, in dessen Voronoi-Region $\tilde{R}_i$ x_j liegt. Weiterhin muss eine Verbindungsstärke $c(x)_{ij}$ zwischen dem Stimulus und dem Neuron n_i vorliegen, das durch die bekannte Funktion cd_X aus der Distanz $d_X(x_j, w_i)$ berechnet wird. Wird eine externe Anregung durchgeführt, so muss jedem Stimulus-Neuron ein Gewichtungsfaktor zugeordnet werden, mit dem der externe, skalare Input gewichtet wird. Im Gegensatz zu den Neuronen aus N_{AA}^t sollen bei den Stimulus-Neuronen keine individuellen Gewichtungen betrachtet werden, sodass ein Gewichtungsfaktor c(x) global und für alle Stimuli konstant verwendet werden soll, der zudem den Wert 1 besitzen kann. Die Analogie zu einem Neuron aus N_{AA}^t erfordert einen Inputwert $net(x)_j^t$, einen Aktivitätswert $z(x)_j^t$ und einen Outputwert $S(z(x)_j^t$. Der Schwellenwert T_x zur Berechnung der Aktivität soll ebenfalls global und für alle Stimuli konstant verwendet werden, wobei er den Wert 0 besitzen kann. Die Outputfunktion S(z) kann vereinfachend als Identitätsfunktion $S(z(x)_j^{t=0}) = z(x)_j^{t=0}$ oder als normale sigmoide Funktion $S(z(x)_j^{t=0}) = 1/[1 + \exp(k * z(x)_j^{t=0})]$ definiert werden, genau so wie bei den Neuronen aus N_{AA}^t. Es ergibt sich somit die Struktur der μ_M'elementigen Stimulusmenge nach deren disjunkter Zerlegung durch den GNG-Graphen G^t:

$$M_{AA}^t = \{m_j = (x_j, n_i, c(x)_{ij}, net(x)_j^t, z(x)_j^t, S(z(x)_j^t) \mid x_j \in \tilde{R}_i) \mid cd_X, c(x), net(x), T_x, z(net(x)), k, S(z(x)); j = 1, ..., \mu_M\}. \quad (68)$$

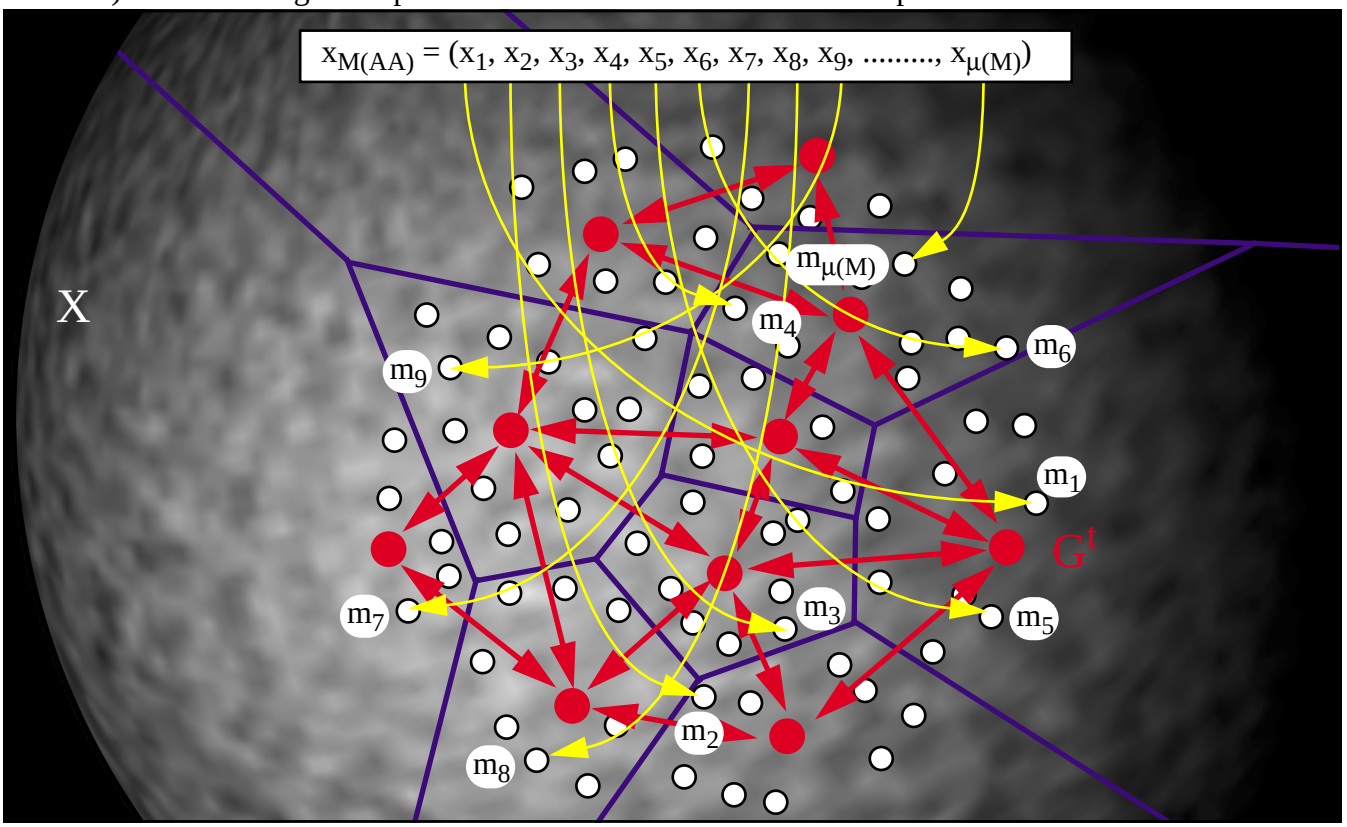
Die Initialisierung der Aktivitätsausbreitung kann nun durch die Sequenz zweier Prozesse operationalisiert werden, wobei das Neuron $n_i \in N_{AA}^t$ und die zugeordneten Stimulus-Neurone $m_{ij} \in M_i$ betrachtet werden sollen:

1) Operationen innerhalb der Stimulus-Neurone m_{ij}.
2) Output-Weiterleitung zu den Neuronen und Operationen innerhalb der Neurone.

Der erste Prozess wird initialisiert als ein Input in Stimulus-Neurone von aussen, wobei genau ein skalarer Input pro Stimulus-Neuron von aussen zugeführt wird. D.h. es existiert ein Inputvektor, der mit $x_{M(AA)}$ bezeichnet werden soll, und der μ_M Komponenten besitzt, die jedoch nicht notwendig alle mit einem von Null abweichenden Wert besetzt sein müssen. Jede Komponente des externen Inputvektors

wird als skalarer Input in einem der Stimulus-Neurone verwendet (siehe Abb. 12)). Gewichtet wird dieser Input mit c(x), wodurch sich der Inputwert $net(x)_{ij}^{t=0}$ ergibt, von dem der Schwellenwert T_x abgezogen wird, wodurch sich der Aktivitätswert $z(x)_{ij}^{t=0}$ ergibt. Entsprechend der Funktion S(z(x)) wird der Output $S(z(x)_{ij}^{t=0})$ berechnet, wodurch der erste Prozess abgeschlossen wird.

Abb. 12) Initialisierung der Inputwerte von Stimuli durch externen Inputvektor



Der zweite Prozess beginnt mit der Weiterleitung der Outputwerte $S(z(x)_{ij}^{t=0})$ an das Neuron n_i, indem die Verbindungsstärken $c(x)_{ij}$ genutzt werden, d.h. für das Neuron n_i wird der Initialisierungs-Inputwert $net_i^{t=0}$ durch die gewichtete Summe gebildet (siehe Abb. 13)):

$$net_i^{t=0} = \sum_j c(x)_{ij} * S(z_i^{t=0}), \forall m_{ij} \in M_i. \quad (69)$$

Im nächsten Schritt wird die Initialisierungs-Aktivierung

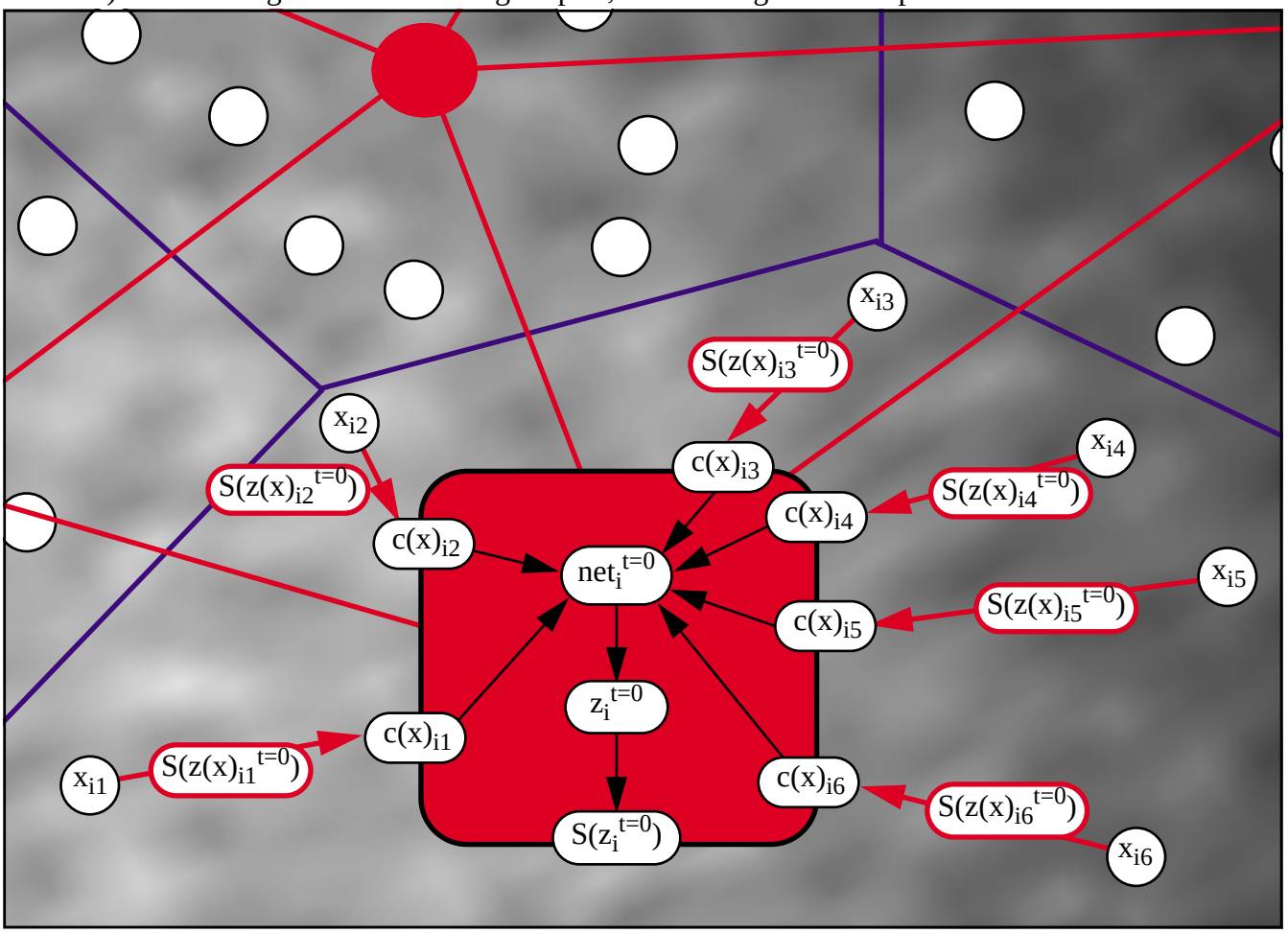
$$z_i^{t=0} = net_i^{t=0} - T_i, \quad (70)$$

und dann der Initialisierungs-Output von n_i berechnet:

$$S(z_i^{t=0}) = 1/[1 + \exp(k * z_i^{t=0})]. \quad (71)$$

Der Initialisierungs-Output $S(z_i^{t=0})$ wird im weiteren für die Aktivitätsausbreitung im GNG-Graphen genutzt, indem er an die Neurone weitergegeben wird, die verbundene Gewichsvektoren besitzen, und indem diese Neurone ihren Initialisierungs-Output an n_i abgeben.

Abb. 13) Bestimmung des Initialisierungs-Input-, Aktivierungs- und Outputwertes



Werden Initialisierungs-Aktivitäten an Stimuli gegeben, so ergeben sich zwei Szenarien bezüglich der Aktivierungen nach dem Abbruch der Aktivierungsausbreitung im GNG-Graphen:
1) Als Endergebnis werden die Aktivierungen an der Stelle der Neurone verwendet.
2) Als Endergebnis sollen Aktivierungen an der Stelle der Stimuli verwendet werden.

Bei dem ersten Szenario ist der Gesamtprozess beendet, wenn das Abbruchkriterium der Ausbreitung in G^t greift, was entsprechend der obigen Definition von $N_{AA}{}^t$ gilt, wenn t_{max} Iterationen durchgeführt wurden. Bei dem zweiten Szenario muss ein abschließender Prozess durchgeführt werden, indem der Output $S(z_i{}^{t(max)})$ eines Neurons n_i an seine Stimuli $m_{ij} \in M_i$ weitergeleitet wird. Dieser Prozess ist symmetrisch zur Initialisierung der Stimuli-Aktivierungen, bei der die Outputs $S(z_{ij}{}^{t=0})$ der Stimuli $m_{ij} \in M_i$ an das Neuron n_i weiter geleitet werden, sodass eine analoge Vorgehensweise verwendet werden soll:

$$net(x)_{ij}{}^{t(max)} = \sum_j c(x)_{ij} * S(z_i{}^{t(max)}),$$
$$z(x)_{ij}{}^{t(max)} = net(x)_{ij}{}^{t(max)} - T_x,$$
$$S(z(x)_{ij}{}^{t(max)}) = 1/[1 + \exp(k * z(x)_{ij}{}^{t(max)})], \forall m_{ij} \in M_i. \quad (72)$$

Prinzipiell kann eine Aktivitätsübertragung auch mit Hilfe von LWR-Verfahren durchgeführt werden, da die dargestellte Vorgehensweise mit Input-, Aktivitäts- und Outputfunktion im Rahmen von Kernel-Funktionen und Radial-Basis-Funktionen reformulierbar ist. So kann beispielsweise die Initialisierungs-

aktivität $z_i^{t=0}$ eines Neurons n_i durch die Aktivitäten $z(x)_{ij}^{t=0}$ der Stimuli m_{ij} in seiner lokalen Stimulusmenge M_i^t bestimmt werden durch die folgende Kernel-Regression, wobei die Kernel-Funktion eine Funktion der Distanz $d_X(x_j, w_i)$ zwischen dem Gewichtsvektor und den Stimulusvektoren ist:

$$z_i^{t=0} = 1/\nu_i \sum_j h(x_j, w_i) * z(x)_{ij}^{t=0}, \text{ mit}$$
$$\nu_i = \sum_j h(x_j, w_i), \forall m_{ij} \in M_i^t. \quad (73)$$

Die Aktivität z_i^t eines Neurons n_i in der t'ten Iteration der Aktivitätsausbreitung in dem GNG-Graphen G_{AA} kann mit Hilfe der Nachbarschaftsmenge $N(d_G=1 \mid N_{AA})_i$ bestimmt werden, d.h. mit den unmittelbar verbundenen Neuronen. Es werden hierzu die Aktivierungen dieser Neurone in der vorangegangenen Iteration t-1 benutzt, um die Aktivierung des betrachteten Neurons in der Iteration t zu berechnen:

$$z_i^t = 1/\nu_i \sum_j h(w_j, w_i) * z_j^{t-1}, \text{ mit}$$
$$\nu_i = \sum_j h(w_j, w_i), \forall n_j \in N(d_G=1 \mid N_{AA})_i. \quad (74)$$

Auch die Aktivitätsausbreitung zum Ende der lateralen Operationen, bei der die Aktivität $z_i^{t(max)}$ eines Neurons n_i auf die Stimuli m_{ij} in seiner lokalen Lernmenge M_i^t übertragen wird, kann durch eine solche KR-Formulierung durchgeführt werden, indem die Aktivierung des Neurons mit dem Wert $h(x_j, w_i)$ der Kernel-Funktion gewichtet wird:

$$z(x)_{ij}^{t(max)} = h(x_j, w_i) * z_i^{t(max)}, \forall m_{ij} \in M_i^t. \quad (75)$$

2.2) Basis-Verfahren des Resamplings

Gegeben ist die Menge der Stimuli $M = \{m_j = (x_j, y_j) \mid j = 1, ..., \mu_M\}$, mit denen künstliche Stimuluslisten bzw. -multimengen B_k erzeugt werden, die jeweils dazu verwendet werden, ein stützpunktbasiertes Approximationsmodell AM_k zu erzeugen (Efron & Tibshirani (1993[105]), Shao & Tu (1995[309])). Das Jackknife-Verfahren, sowie Resamplingverfahren höherer Ordnung, bei denen Bootstrapschätzungen als Stimuli höherer Ordnung verwendet werden, sollen hier nicht dargestellt werden (siehe hierzu Bachelier (1999d[22])).

Liegt eine genügend große Menge von Modellen AM_k vor, so kann die Outputkomponente eines unbekannten, unvollständigen Recallstimulus $m_{R(i)} = (x_{R(i)}, \_)$ durch eine Aggregationsfunktion aller Bootstrapschätzungen $y_{R(i)}^\wedge = f(x_{R(i)} \mid AM_k)$ und der Schätzung des Gesamtmodells $y_{R(i)}^\wedge = f(x_{R(i)} \mid AM)$ erzeugt werden.

2.2.1) Stimulus-Bootstrap

Bei einem Stimulus-Bootstrap werden δ künstliche Multimengen bzw. Listen B_k durch Ziehen mit Zurücklegen aus einer Grundmenge M erzeugt, wobei die Anzahl der Elemente in den B_k der Anzahl der Elemente in M entspricht. Die entstehenden Bootstraplisten B_k können in der Menge B zusammengefasst werden:

$$B = \{B_k \mid k = 1, ..., \delta\}, \text{ mit}$$
$$B_k = (m_{kj} = (x_{kj}, y_{kj}) \in M \mid j = 1, ..., \mu_M). \quad (76)$$

Zu jeder Liste B_k korrespondiert ein Modell AM_k, das für einen unbekannten, unvollständigen Recallstimulus $m_{R(i)} = (x_{R(i)}, \_)$ eine Schätzung $f(x_{R(i)} \mid AM_k)$ liefert, die zu einer Bootstrapschätzung aggregiert werden, indem der arithmetische Mittelwert der einzelnen Schätzungen gebildet wird:

$$f(x_{R(i)} \mid B) = 1/\delta * \sum_{k=1->\delta} f(x_{R(i)} \mid AM_k). \quad (77)$$

Zusammen mit der Schätzung des Gesamtmodells $f(x_{R(i)} \mid AM)$ kann die Bias der Bootstrapschätzung als normale bzw. 0,632-Bias ermittelt werden:

$$bias(x_{R(i)}) = f(x_{R(i)} \mid B) - f(x_{R(i)} \mid AM), \text{ bzw.}$$
$$bias(x_{R(i)})_{0,632} = (f(x_{R(i)} \mid B) - f(x_{R(i)} \mid AM))/0,632. \quad (78)$$

Eine bias-korrigierte Bootstrapschätzung ergibt sich als:

$$f(x_{R(i)} \mid B)_{bias} = f(x_{R(i)} \mid AM) - bias(x_{R(i)}), \text{ bzw.}$$
$$f(x_{R(i)} \mid B)_{bias} = f(x_{R(i)} \mid AM) - bias(x_{R(i)})_{0,632}. \quad (79)$$

2.2.2) Restwert-Bootstrap

Ein Restwert-Bootstrap erzeugt für alle Stimuli $m_j \in M$ eine Schätzung $f(x_j \mid AM)$ mit Hilfe des Gesamtmodells AM, und erzeugt danach die Restwerte $\Delta y_j = y_j - f(x_j \mid AM)$, die in einer Menge RW_M gesammelt werden:

$$RW_M = \{\Delta y_j = y_j - f(x_j \mid AM) \mid j = 1, ..., \mu_M\}. \quad (80)$$

Aus dieser Menge werden neue Stimulusmengen $M_{\Delta k}$ gebildet, indem μ_M mal ein Element aus RW_M mit Zurücklegen gezogen wird, und der Restwert Δy_{jk}, der bei der j'ten Ziehung gezogen wird, wird dem richtigen Output y_j hinzu-addiert:

$$M_{\Delta k} = \{(x_{jk}, y_{\Delta jk} = y_j + \Delta y_{jk}) \mid j = 1, ..., \mu_M\}. \quad (81)$$

Nachem δ Stimulusmengen $M_{\Delta k}$, k = 1, ..., δ, erzeugt wurden, wird jede der Stimulusmengen zum Aufbau eines Modells AM_k verwendet, die jeweils für einen unbekannten, unvollständigen Recallstimulus $m_{R(i)} = (x_{R(i)}, \_)$ eine Schätzung $f(x_{R(i)} \mid AM_k)$ liefern. Die Bootstrapschätzung $f(x_{R(i)} \mid B)$ wird wiederum als arithmetischer Mittelwert berechnet, und es kann mit Hilfe von $f(x_{R(i)} \mid AM)$ eine bias-korrigierte Schätzung $f(x_{R(i)} \mid B)_{bias}$ abgegeben werden.

2.2.3) Moving-Blocks-Bootstrap

Ein Moving-Blocks-Bootstrap (Efron & Tibshirani (1993:99ff[105])) ist die Anwendung des Resamplingprinzips auf eine Zeitreihe, die sich dadurch auszeichnet, dass die x-Werte als Zeitpunkte interpretiert werden, sodass anstatt der x-Werte der Index j selbst gewählt werden kann, wenn die Ermittlung der

einzelnen y-Werte in gleichen Zeitintervallen erfolgt. Durch die zeitliche Ordnung können die Stimuli nicht mehr in einer Menge vorliegen, sondern sie werden in eine geordnete Liste eingetragen:

$$ML = (m_j = (x_j, y_j) \mid x_j \in R^+, x_j < x_{j+1}, y_j \in R, j = 1, ..., \mu_M), \text{ bzw.}$$
$$ML = (m_j = (j, y_j) \mid y_j \in R, j = 1, ..., \mu_M). \quad (82)$$

Für die weiteren Betrachtungen soll die zweite Darstellung $ML = ((1, y_1) , (2, y_2) , (3, y_3) , (4, y_4) , ..., (\mu_M, y_{\mu(M)}))$ verwendet werden.

Beim Moving-Blocks-Bootstrap muss ein Parameter $l \in \{1, 2, ... , \mu_M\}$ festgelegt werden, der die Blockgröße beschreibt. Aus der Stimulusliste ML werden im nächsten Schritt alle möglichen zusammenhängende Teillisten ML_p mit jeweils l Elementen erzeugt, wobei $\mu_M - l + 1$ Teillisten möglich sind:

$$ML_1 = (m_1, m_2, ..., m_l),$$
$$ML_2 = (m_2, m_3, ..., m_{l+1}),$$
$$...$$
$$ML_{\mu(M)-l+1} = (m_{\mu(M)-l} , m_{\mu(M)-l+1}, ..., m_{\mu(M)}). \quad (83)$$

Die einzelnen Teillisten werden in die Bootstrap-Grundelementeliste BML eingefügt:

$$BML = (ML_p \mid p = 1, ..., \mu_M - l + 1). \quad (84)$$

Die Verwendung einer Liste bzw. einer Multimenge anstatt einer einfachen Menge führt dazu, dass Teillisten, die mehrfach in der Stimulusliste ML auftreten, auch mehrfach in BML auftreten, sodass eine erhöhte Wahrscheinlichkeit besteht, dass diese Elemente bei den nachfolgenden Ziehungen mit Zurücklegen gezogen werden. Auf diese Weise werden Charakteristika der ursprünglichen geordneten Liste ML stochastisch bei den Bootstraplisten BML_k erhalten.

Die Anzahl μ_B der Ziehungen einer Teilliste zur Erzeugung einer Bootstrapliste BML_k ist abhängig von der Anzahl der Stimuli μ_M und dem Blockgrößenparameter l, da das Produkt aus μ_B und l größer als μ_M sein muss, d.h. es wird die Anzahl μ_B gewählt, bei der $\mu_B * l$ zum ersten Mal größer als μ_M wird. Sei $round_+(.)$ die Rundungsoperation, die auf die nächst größere natürliche Zahl aufrundet, so ergibt sich die Anzahl der Ziehungen aus dem Quotient von μ_B und dem Blockgrößenparameter l:

$$\mu_B = round_+(\mu_M/l). \quad (85)$$

Soll eine Zeitreihe mit der gleichen Länge erzeugt werden, so wird μ_B auf diese Weise bestimmt, und die überschüssigen Elemente $\mu_B - \mu_M$ werden entfernt, sodass in jeder Bootstrapliste BML_k genau μ_M Stimuli vorliegen:

$$BML_k = (m_{jk} \mid j = 1, ..., \mu_M). \quad (86)$$

Wie üblich werden δ Bootstraplisten auf diese Weise erzeugt und in die Menge B der Bootstraplisten eingefügt, mit der die weiteren Analysen durchgeführt werden.

$$B = (BML_k \mid k = 1, ..., \delta). \quad (87)$$

Mit dem Verfahren der Ziehung mit Zurücklegen aus der Bootstrap-Grundelementeliste BML lassen sich

Stimulussequenzen beliebiger Längen erzeugen, indem entsprechend oft gezogen wird, in Verbindung mit der eventuellen Entfernung überschüssiger Stimuli. D.h. es lassen sich insbesondere längere Sequenzen erzeugen, als die ursprüngliche Liste ML, was im Rahmen der Erzeugung von künstlichen Dokumenten und Queries für die Polyrepräsentation an Bedeutung besitzt (siehe Kapitel 3)).

Die einfache Moving-Block-Strategie lässt sich erweitern, indem eine Stimulusliste ML unabhängig durch verschiedene Blockgrößen verarbeitet wird, sodass die Abhängigkeit von der Auswahl von l verringert wird. Für verschiedene Blockgrößen l wird ML in Teillisten $ML(l)_1$, $ML(l)_2$, ... zerlegt, wobei für jedes l eine Bootstrap-Grundelementeliste BML(l) gebildet wird. Die Listen BML(l) werden in einer Liste BML zusammengefasst, wobei in einem Beispiel Blockgrößen der Länge l_{min} bis l_{max} verwendet werden sollen:

$$BML = (BML(l) \mid l = l_{min}, ..., l_{max}), \text{ mit}$$
$$BML(l) = (ML(l)_p \mid p = 1, ..., \mu_M - l + 1). \quad (88)$$

Obwohl auf diese Weise zwei Parameter l_{min} und l_{max} anstatt einem Parameter l festgelegt werden müssen, ist die Abhängigkeit von der Auswahl der Parameter in dem Maße verringert, indem die Spannbreite der beiden Parameter gewählt wird. Im Extremfall $l_{min} = 1$ und $l_{max} = \mu_M$ muss keine externe Parameterfestlegung getroffen werden, sodass die Blockgröße als Parameter faktisch entfällt.

Die Erzeugung einer künstlichen Bootstrap-Stimulusliste BML_k wird durch einen zweistufigen Ziehungsprozess durchgeführt, indem zunächst aus BML ein Element BML(l) gleichverteilt zufällig mit Zurücklegen gezogen wird. In der zweiten Stufe wird aus der gezogenen Liste BML(l) ein Element $ML(l)_p$ ebenfalls gleichverteilt zufällig mit Zurücklegen gezogen. Die l Stimuli m_j, die sich in der gezogenen Liste $ML(l)_p$ befinden, bilden mit der gleichen Reihenfolge die nächsten l Komponenten in der Stimulussequenz von BML_k. Wird eine Gesamtzahl von Stimuli in BML_k von μ_M zum ersten Mal überschritten, so wird das Verfahren abgebrochen, und die überzähligen Elemente werden entfernt bzw. die sich ergebende Sequenz wird beibehalten, wenn die Länge keinen Einfluss auf die weiteren Operationen hat, wie bei der Indexierung einer Zeichensequenz, als einer Abbildung von einem Zeichensequenzraum in einen metrischen Vektorraum (siehe Abschnitt 3.4)).

Wird die Erzeugung einer diversiven Liste B der Bootstraplisten BML_k als Ziel betrachtet, so existiert eine Alternative, bei der zunächst wieder eine Liste BML(l) gleichverteilt zufällig mit Zurücklegen aus BML gezogen wird. Mit Hilfe dieser Liste wird eine künstliche Stimulusliste $BML(l)_k$ erzeugt, indem Teillisten $ML(l)_p$ aus BML(l) gleichverteilt zufällig mit Zurücklegen gezogen werden, bis die Anzahl der Stimuli zum ersten Mal größer μ_M wird. $BML(l)_k$ wird in die Liste B eingefügt, und es wird unabhängig von der zuvor gezogenen Liste eine neue aus BML gezogen, bis die Sollanzahl δ von Bootstrap-Stimuluslisten erreicht wird, die jeweils durch unterschiedliche Blockgrößen erzeugt wurden:

$$B = (BML(l)_k \mid l \in \{l_{min}, ..., l_{max}\}, k = 1, ..., \delta). \quad (89)$$

2.3) Modellqualität

Grundlage des Aufbaus eines Modells AM(x) sowie der Bestimmung der Modellqualität f(AM(x)) ist die Stimulusmenge $M = \{(x_j, f(x_j)) \mid j = 1, ..., \mu_M\}$, die vor dem überwachten Lernen standardmäßig in eine Lernmenge LM und eine Testmenge TM disjunkt zerlegt wird:

$$M = LM \cup TM,\ LM \cap TM = \varnothing, \text{ mit}$$
$$LM = \{(x_{L,j}, f(x_{l,j})) \mid j = 1, ..., \mu_L\},$$
$$TM = \{(x_{T,j}, f(x_{T,j})) \mid j = 1, ..., \mu_T\},\ \mu_L + \mu_T = \mu_M. \quad (90)$$

Die Modellqualität wird in den meisten Ansätzen auf der Basis der Testmenge ermittelt, sodass die Bezeichnung f(AM(x) | TM) passender wäre.

Im weiteren werden SC-GNG-SOM-Modelle mit ihren Erweiterungen zu überwachten Lernmodellen betrachtet, d.h. es wird von einem Batch-Lernen ausgegangen, bei dem zunächst ein unüberwachtes SC-GNG-SOM-Modell erzeugt wird, gefolgt von einer Erweiterung zu einem stützpunktbasierten Approximationsmodell, indem jedem Neuron als Prototypen ein Stützpunkt im Outputraum in Form einer Schätzung f(w)^ oder f(w(x))^ zugeordnet wird. Diese Vorgehensweise erfordert, dass zwei Typen von Bewertungen betrachtet werden müssen:

1) Modellqualität des unüberwachten Teils, d.h. der Gewichtsvektorenverteilung des SC-GNG-SOM-Modells.
2) Modellqualität des überwachten Teils, d.h. die Bewertung f(AM(x)) des Approximationsmodells auf der Basis einer Stimulusmenge.

Duch die Verwendung eines vorgeschalteten unüberwachten Modells kann eine andere Strategie als der oben dargestellten Zerlegung von M in LM und TM durchgeführt werden, da am Aufbau der Gewichtsvektorenverteilung in X die Inputkomponenten aller Stimuli sinnvoll verwendet werden können.

2.3.1) Unüberwachte SC-GNG-SOM-Qualität durch lokale Quantifizierungsfehler

Im Rahmen von GNG-SOM- und SC-GNG-SOM-Modellen werden lokale Fehler E_i^t auf Neuronenebene verwendet, um Einfügezentren während des Wachstumsprozesses und um Adaptionsbezugspunkte für die lokale Umverteilung von Stimuli zu identifizieren. Das grundlegende SC-GNG-SOM-Modell ist unüberwacht, wobei als Fehlerart meist der Quantifizierungsfehler QF_i^t verwendet wird. Besitzt ein Neuron n_i^t eine lokale Stimuluslernmenge M_i^t, so wird dessen Quantifizierungsfehler bestimmt durch (s.a. Bachelier (1998d[18]) für andere Qualitätsmaße unüberwachter SOM-Modelle):

$$QF_i^t = 1/\mu_i^t \sum_j (w(x)_i^t - x_j^t)^2, \text{ bzw. allgemein}$$
$$QF_i^t = 1/\mu_i^t \sum_j d_X(w(x)_i^t, x_j^t),\ \forall (x_j^t, f(x_j^t)) \in M_i^t. \quad (91)$$

Der Gesamtfehler des SC-GNG-SOM-Modells ergibt sich als Funktion der lokalen Quantifizierungsfehler der μ^t Neurone aus N^t, z.B. als Mittelwert der lokalen Quantifizierungsfehler:

$$QF(N^t) = 1/\mu^t \sum_i QF_i^t,\ \forall n_i^t \in N^t. \quad (92)$$

Die Bewertung des unüberwachten Modells kann auch durch komplexere Funktionen ermittelt werden, indem beispielsweise die Varianz der lokalen Quantifizierungsfehler berücksichtigt wird:

$$\mathrm{var}_{QF}(N^t) = 1/\mu^t \sum_i (QF(N^t) - QF_i^t)^2, \forall n_i^t \in N^t. \quad (93)$$

Mit $QF(N^t)$ als Mittelwert und $\mathrm{var}_{QF}(N^t)$ als Varianz kann dem Modell ein Konfidenz-Intervall der Quantifizierungsfehler zugeordnet werden, das als komplexere Modellbewertung dient, mit α als einem Skalierungsparameter des Intervalls:

$$KI_{QF}(N^t) = [QF(N^t) - \alpha * \mathrm{var}_{QF}(N^t), QF(N^t) + \alpha * \mathrm{var}_{QF}(N^t)]. \quad (94)$$

Die beiden Werte $QF(N^t)$ und $\mathrm{var}_{QF}(N^t)$ können auch im Rahmen einer Zwei-Ziel-Vergleichsoperation betrachtet werden, wenn Modelle verglichen werden sollen, wobei das Pareto-Kriterium verwendet werden kann, sodass eine Spezifizierung einer Aggregationsfunktion, die den Mittelwert und die Varianz zu einem Funktionswert zusammenfasst, oder die Spezifizierung eines Skalierungs-Parameters für $KI_{QF}(N^t)$ entfallen kann (siehe Näheres zum Pareto-Kriterium und dessen Anwendung in einer Mehr-Ziel-Evolutionsstrategie in Bachelier (1998b:141ff[16]), Bachelier (1999d:22ff[22]), siehe auch Abschnitt 2.4) und 2.5)). Ein Modell N_1^t dominiert entsprechend dem Pareto-Kriterium ein Modell N_2^t genau dann, wenn der Fehlermittelwert und die Varianz des ersten Modells kleiner sind als die des zweiten Modells, bzw. wenn einer der beiden Werte kleiner ist, und der andere nicht größer:

$$\begin{aligned} &N_1^t \text{ dominiert } N_2^t <=> \\ &(QF(N_1^t) < QF(N_2^t) \wedge \mathrm{var}_{QF}(N_1^t) \leq \mathrm{var}_{QF}(N_2^t)) \vee \\ &(QF(N_1^t) \leq QF(N_2^t) \wedge \mathrm{var}_{QF}(N_1^t) < \mathrm{var}_{QF}(N_2^t)). \end{aligned} \quad (95)$$

2.3.2) Überwachte SC-GNG-SOM-Qualität durch lokale MSE-Werte

Wird der Quantifizierungsfehler verwendet, um die Gewichtsvektoren in X zu plazieren, so bleiben die Outputfehler unberücksichtigt, was während dem Wachstumsprozess keine Rolle spielt. Andererseits kann das Abbruchkriterium der SC-GNG-Erzeugung teilweise auf einem Outputfehler basieren, insbesondere, wenn eine Feinadaptionsphase beendet werden soll. In jedem Fall muss nach dem Abbruch des unüberwachten Teils des Lernens ein Outputfehler wie der mittlere quadratische Fehler (MSE), verwendet werden, um ein Approximationsmodell AM(x) zu bilden, indem für jeden Gewichtsvektor $w(x)_i^t$ eine Outputschätzung $f(w(x)_i^t)\wedge$ gebildet wird. Dies kann lokal durch die Stimulus $(x_j^t, f(x_j^t)) \in M_i^t$ und beispielsweise eine Kernel-Regression erfolgen:

$$\begin{aligned} f(w(x)_i^t)\wedge &= 1/\nu_i \sum_j h(d_X(w(x)_i^t, x_j^t)) * f(x_j^t), \text{ mit} \\ \nu_i &= \sum_j h(d_X(w(x)_i^t, x_j^t)), \forall (x_{jk}^t, f(x_{jk}^t)) \in M_i^t, \end{aligned} \quad (96)$$

Es entsteht ein AM(x), da jedes Neuron als Prototyp einen Stützpunkt im Input- und im Outputraum besitzt. Um die Modellqualität zu bestimmen, muss jedem Stimulus eine Outputschätzung mit dem zu bewertenden Modell zugeordnet werden. Bei einer globalen Nutzung der Prototypen durch eine Kernel-Regression ergibt sich die Outputschätzung $f(x_j^t \mid AM(x)^t) = f(x_j^t)\wedge$ für den Inputvektor x_j^t:

$$f(x_j^t \mid AM(x)^t) = 1/\nu_j \sum_i h(d_X(x_j^t, w(x)_i^t)) * f(w(x)_i^t)\wedge, \text{ mit}$$
$$\nu_j = \sum_i h(d_X(x_j^t, w(x)_i^t)), \forall n_i^t \in N^t. \quad (97)$$

Der lokale MSE des Neurons n_i^t auf der Basis der lokalen Stimulusmenge bestimmt sich aus den μ_i^t darin enthaltenen Stimuli zu:

$$MSE(n_i^t \mid AM(x)^t) = 1/\mu_i^t * \sum_j (f(x_{ij}^t) - f(x_{ij}^t \mid AM(x)^t))^2, \forall (x_{ij}^t, f(x_{ij}^t)) \in M_i^t. \quad (98)$$

Der Gesamtfehler des überwachten SC-GNG-SOM-Modells ergibt sich wieder als Funktion der lokalen MSE-Werte der μ^t Neurone aus N^t, z.B. als Mittelwert:

$$MSE(AM(x)^t) = 1/\mu^t \sum_i MSE(n_i^t \mid AM(x)^t), \forall n_i^t \in N^t. \quad (99)$$

Zur Bewertung kann auch der lokale Fehlermittelwert und Fehlervarianz herangezogen werden, wobei $MSE(n_i^t \mid AM(x)^t)$ als Fehlermittelwert verwendet wird, und die lokale Varianz sich aus den quadratischen Fehlerwerten SE_{ij}^t ergibt:

$$var(n_i^t \mid AM(x)^t) = 1/(\mu_i^t - 1) * \sum_j (SE_{ij}^t - MSE(n_i^t \mid AM(x)^t))^2,$$
$$SE_{ij}^t = (f(x_{ij}^t) - f(x_{ij}^t \mid AM(x)^t))^2, \forall (x_{ij}^t, f(x_{ij}^t)) \in M_i^t. \quad (100)$$

Eine globale Varianz kann durch den globalen Mittelwert $MSE(N^t \mid AM(x)^t)$ und die lokalen Mittelwerte $MSE(n_i^t \mid AM(x)^t)$ gebildet werden:

$$var_{MSE}(AM(x)^t) = 1/\mu^t \sum_i (MSE(n_i^t \mid AM(x)^t) - MSE(AM(x)^t))^2, \forall n_i^t \in N^t, \quad (101)$$

bzw. es kann die mittlere, globale Varianz gebildet werden:

$$\overline{var}_{MSE}(AM(x)^t) = 1/\mu^t \sum_i var(n_i^t \mid AM(x)^t), \forall n_i^t \in N^t. \quad (102)$$

Analog der Vorgehensweise bei dem Quantifizierungsfehler lässt sich auch mit dem MSE und einem abgeleiteten Varianzwert dem Modell $AM(x)^t$ ein Konfidenz-Intervall zuordnen:

$$KI_{MSE}(AM(x)^t) = [MSE(AM(x)^t) - \alpha * var_{MSE}(AM(x)^t), MSE(AM(x)^t) + \alpha * var_{MSE}(AM(x)^t)]. \quad (103)$$

2.3.3) MSE-Integral und Bias-Varianz-Zerlegung

Wird der MSE über einer bekannten Verteilung P(x) der Inputvariablen definiert, d.h. die Wertebereiche der einzelnen Variablen sind bekannt, so lässt er sich darstellen als (Geman et al. (1992[142]), Cohn (1995[73]), Cohn et al. (1995[74])):

$$f(AM(x)^t) = \int_x E[(f(x \mid AM^t) - f(x))^2] * P(x)\, dx, \quad (104)$$

mit E[.] als dem Erwartungswert über P(x) und der Stimulusmenge M^t. Der Erwartungswert lässt sich zerlegen zu (Cohn (1995[73]), Cohn et al. (1995[74])):

$$E[(f(x \mid AM(x)^t) - f(x))^2 \mid x, M^t] =$$
$$E[(f(x) - E[f(x) \mid x])^2]$$
$$+ (E_{LM(t)}[f(x \mid AM(x)^t)] - E[f(x) \mid x])^2$$
$$+ E_{LM(t)}[(f(x \mid AM(x)^t) - E_{LM(t)}[f(x \mid AM(x)^t)])^2]. \quad (105)$$

Der erste Term gibt die Varianz des Outputs y = f(x) bei einem gegebenen x an, was dem Rauschen in der zu lernenden Funktion entspricht. Dieser Term ist unabhängig von den Eigenschaften des Lerners und der Art wie Stimuli ausgesucht werden. Der zweite Term entspricht der quadratischen Bias des Lerners, wobei die Bias das grundsätzliche Fehlerniveau des Lerners beschreibt, das durch den gegebenen Lern-Algorithmus und die gegebene Lernmenge verursacht wird. Der dritte Term entspricht der Varianz des Lerners bei einer gegebenen Stimulusmenge.

Es existieren weitere MSE-Zerlegungen (siehe Wolpert (1995a[368])), sodass nicht von einer einfachen additiven Zerlegung in Bias und Varianz ausgegangen werden kann. Wird die Zerlegung auf die beiden Komponenten Bias und Varianz beschränkt, so muss im allgemeinen Fall von einem nicht-linearen und nicht-monotonen Zusammenhang ausgegangen werden. Der Zusammenhang muss als nicht-monoton bezeichnet werden, da alle Zerlegungstypen von einem Tradeoff ausgehen, d.h. wird eine der beiden Komponenten verkleinert, so existiert mindestens ein Wertebereich, in dem sich die andere Komponente ebenfalls verkleinert, und es existiert mindestens ein Wertebereich, in dem die andere Komponente ansteigt. Der Zusammenhang muss als nicht-linear bezeichnet werden, da Wertebereiche existieren, in denen die korrespondierende sowie die gegenläufige Veränderung der Komponenten nicht proportional verläuft. D.h. es existiert beispielsweise ein Wertebereich, in dem eine Biasverringerung mit einem überproportionalen Varianzanstieg erkauft werden muss. Der nicht-lineare und nicht-monotone Zusammenhang kann durch eine zweikomponentige Funktion beschrieben werden:

$$E[(f(x \mid AM(x)^t) - f(x))^2] = g(bias(AM(x)^t)^2, var(AM(x)^t)). \quad (106)$$

Die Minimierung des MSE kann somit durch eine Minimierung der Bias und der Varianz eines Modells erreicht werden, d.h. eine 2-Ziel-Minimierung beider Terme führt zu einer Minimierung des MSE. Werden die Modellbias und -varianz als Komponenten einer Modellbewertung betrachtet, so ergibt sich ein zwei-komponentiger Bewertungsvektor, der durch eine Zwei-Ziel-Optimierungsoperation minimiert werden kann:

$$f(AM(x)^t \mid M^t) = (bias(AM(x)^t \mid M^t)^2, var(AM(x)^t \mid M^t)) \rightarrow \min. \quad (107)$$

2.3.4) Modellbewertung durch Varianz- und Bias-Integrale

Entgegen der ein-dimensionalen Integration über ein Varianz- bzw. Bias-Intervall im Rahmen von Häufigkeitsverteilungen kann eine n-dimensionale Integration innerhalb des Objektvariablenraumes X durchgeführt werden, da jedem Punkt $x \in X$ eine Output-Varianzschätzung und eine Schätzung des Biasquadrates zugeordnet werden kann (Cohn (1995[73]), Cohn et al. (1995[74])):

$$\text{var}(AM(x)^t) = \int_x \sigma(f(x))^2 * P(x)\, dx,$$
$$\text{bias}(AM(x)^t) = \int_x \text{bias}(x)^2 * P(x)\, dx. \quad (108)$$

Die Output-Varianz $\sigma(f(x))^2$ muss durch $\sigma(f(x))^{2\wedge}$ geschätzt werden, da der richtige Outputwert f(x) nur für die vorliegenden Stimuli bekannt ist, sodass der geschätzte Output $f(x \mid AM(x)^t)$ verwendet werden muss. Im Rahmen eines Kernel-Regressionsmodells auf der Basis der Gewichtsvektoren des Modells $AM(x)^t$, kann dies explizit angegeben werden (Cohn (1995[73]), Cohn et al. (1995[74])):

$$\sigma(f(x))^{2\wedge} = \sigma(f(x \mid AM(x)^t))^{2\wedge}/\nu_x, \text{ mit}$$
$$\sigma(f(x \mid AM(x)^t))^{2\wedge} = 1/\nu_x \sum_i h(d_X(w(x)_i^t, x)) * (f(w(x)_i^t)^\wedge - f(x \mid AM(x)^t))^2,$$
$$f(x \mid AM(x)^t) = 1/\nu_x \sum_i h(d_X(w(x)_i^t, x)) * f(w(x)_i^t)^\wedge,$$
$$\nu_x = \sum_i h(d_X(w(x)_i^t, x)),\ \forall n_i^t \in N^t. \quad (109)$$

In Cohn (1995[73]) wird mit den cross-validated-estimates ein Regressionsmodell für die Biasquadrat-Schätzung beliebiger Punkte x durch die Biasquadrate $\text{bias}(x_j^t \mid AM(x)^t)^2$ der verfügbaren Stimuli $m_j^t \in M^t$ angegeben. Hierzu werden für alle Stimuli der Restwert $f(x_j^t \mid AM(x)^t) - f(x_j^t)$ bestimmt, der gleich dem Biaswert des Modells $AM(x)^t$ an dieser Stelle ist. Die Wertepaare $(x_j^t, \text{bias}(x_j^t \mid AM(x)^t)^2)$ werden als Stützpunkte einer Kernel-Regression verwendet, wobei $\text{bias}(x_j^t \mid AM(x)^t)^2$ die Stützpunkte in einem ein-dimensionalen Biasraum sind, der als Outputraum behandelt wird. Wird die Biasquadrat-Schätzung $\text{bias}(x \mid AM(x)^t)^2$ eines Integrationspunktes x gesucht, so wird diese aus den μ Stimulusstützpunkten berechnet zu:

$$\text{bias}(x \mid AM(x)^t)^2 = 1/\nu_j \sum_j h(d_X(x, x_j^t)) * \text{bias}(x_j^t \mid AM(x)^t)^2,$$
$$\nu_j = \sum_j h(d_X(x, x_j^t)),\ \forall m_j^t \in M^t. \quad (110)$$

In analoger Weise lässt sich auch ein Varianz-Modell auf der Basis der Wertepaare $(x_j^t, \sigma(f(x_j^t))^2)$ erzeugen, das als Alternative zu den oben dargestellten Varianzschätzungen verwendet werden kann, indem die Varianzwerte als Stützpunkte in einem ein-dimensionalen Varianzraum betrachtet werden. Wird die Outputvarianz $\sigma(f(x))^2$ eines Integrationspunktes x gesucht, so wird diese durch eine Kernel-Regression auf der Basis der μ Stimulusstützpunkten erzeugt:

$$\text{var}(x \mid AM(x)^t) = \sigma(f(x))^2 = 1/\nu_j \sum_j h(d_X(x, x_j^t)) * \sigma(f(x_j^t))^2,$$
$$\nu_j = \sum_j h(d_X(x, x_j^t)),\ \forall m_j^t \in M^t. \quad (111)$$

2.3.5) Output-, Bias- und Varianz-Approximationsmodelle

Eine effizientere Vorgehensweise anstatt den cross-validated-estimates auf der Basis der Stimuli lässt sich dann durchführen, wenn jedem Gewichtsvektor ein Output-Varianzwert bzw. ein Biasquadrat zugeordnet werden könnte, da für eine Varianz- bzw. Biasquadrat-Schätzung eines Integrationspunktes in diesem Fall μ_N Komponenten anstatt μ Komponenten berechnet werden müssen. Die Neuronenstruktur wird sich in diesem Fall erweitern zu:

$$N^t = (n_i^t = (w(x)_i^t, f(w(x)_i^t)\hat{}, \sigma(f(w(x)_i^t))^2\hat{}, bias(w(x)_i^t)^2\hat{}, C_i^t, M_i^t, QF_i^t) \mid i = 1, ..., \mu_N^t). \quad (112)$$

Die Schätzung des Outputs, der Output-Varianz und des Biasquadrates jedes Punktes $x \in X$ lässt sich durch ein LWR-Verfahren auf der Basis der Menge der Neurone in N^t berechnen, wodurch nun drei Approximationsmodelle vorliegen:
1) $AM(x)^t$: Output-Approximationsmodell auf der Basis der Outputwerte $f(w(x)_i^t)\hat{}$.
2) $AM(\sigma_f^2)^t$: Output-Varianz-Approximationsmodell auf der Basis der Varianzwerte $\sigma(f(w(x)_i^t))^2$.
3) $AM(b^2)^t$: Bias-Approximationsmodell auf der Basis der Biasquadrat-Schätzungen $bias(w(x)_i^t)^2$.

Die Output-Varianz und die Biasquadrat-Schätzung setzen jedoch voraus, dass für die Gewichtsvektoren $w(x)_i^t$ jeweils der richtige Outputwert $f(w(x)_i^t)$ anstatt einer Schätzung $f(w(x)_i^t)\hat{}$ durch die Stimuli $(x_j^t, f(x_j^t))$ aus M^t oder M_i^t, zur Verfügung steht. Sollen diese Werte aufgrund des hohen externen Aufwandes nicht bestimmt werden, so kann beispielsweise eine verbesserte biaskorrigierte 0,632-Bootstrap-Schätzung verwendet werden, die auf Stimulus-Bootstrapmengen basieren. Es werden δ Bootstrap-Stimulusmengen $M_{B(k)}^t$, $k = 1, ..., \delta$, durch μ_L^t'faches Ziehen mit Zurücklegen erzeugt, die jeweils die Neuronenmenge N^t zu $N_{B(k)}^t$ rekonfigurieren. Die Rekonfiguration kann durch zwei Verfahren durchgeführt werden:
1) Vollständige Rekonfiguration im Input- und Outputraum.
2) Vereinfachte Rekonfiguration im Outputraum.

Bei der vollständigen Rekonfiguration werden die Positionen der Gewichtsvektoren im Inputraum neu festgelegt, d.h. ein Gewichtsvektor $w(x)_i^t$ wird durch $w(x)_{B(k),i}^t$ ersetzt, indem alle Stimuli aus $M_{B(k),i}^t$ verwendet werden, z.B. indem der Mittelwertsvektor aller $x_{B(k),ij}^t$ gebildet wird. Im Anschluss wird die Outputschätzung $f(w(x)_i^t)\hat{}$ durch $f(w(x)_{B(k),i}^t \mid M_{B(k),i}^t)\hat{}$ ersetzt, indem z.B. eine Kernel-Regression auf der Basis der Stimuli aus $M_{B(k),i}^t$ und $w(x)_{B(k),i}^t$ gebildet wird:

$$f(w(x)_{B(k),i}^t \mid M_{B(k),i}^t)\hat{} = 1/\nu_{B(k),i} \sum_j h(d_X(w(x)_{B(k),i}^t, x_{B(k),ij}^t)) * f(x_{B(k),ij}^t), \text{ mit}$$
$$w(x)_{B(k),i}^t = 1/\mu_i^t \sum_j x_{B(k),ij}^t,$$
$$\nu_{B(k),i} = \sum_j h(d_X(w(x)_{B(k),i}^t, x_{B(k),ij}^t)), \forall (x_{B(k),ij}^t, f(x_{B(k),ij}^t)) \in M_{B(k),i}^t. \quad (113)$$

Eine vereinfachte Rekonfiguration behält die Stützpunkte im Inputraum für alle Neuronenmengen $N_{B(k)}^t$ konstant, und verändert nur die Outputschätzung $f(w(x)_i^t)\hat{}$ durch $f(w(x)_i^t \mid M_{B(k),i}^t)\hat{}$, indem z.B. eine Kernel-Regression auf der Basis der Stimuli aus $M_{B(k),i}^t$ und $w(x)_{B(k),i}^t$ gebildet wird.

$$f(w(x)_i^t \mid M_{B(k),i}^t)\hat{} = 1/\nu_{B(k),i} \sum_j h(d_X(w(x)_i^t, x_{B(k),ij}^t)) * f(x_{B(k),ij}^t), \text{ mit}$$
$$\nu_{B(k),i} = \sum_j h(d_X(w(x)_i^t, x_{B(k),ij}^t)), \forall (x_{B(k),ij}^t, f(x_{B(k),ij}^t)) \in M_{B(k),i}^t. \quad (114)$$

Die δ Bootstrap-Outputschätzungen werden zu einer Bootstrap-Schätzung durch die Bildung eines Mittelwertes aggregiert, wobei sich bei der vereinfachten Rekonfiguration ergibt:

$$f(w(x)_i^t \mid M_{B(k),i}^t, k = 1, ..., \delta)\hat{} = 1/\delta \sum_i * (f(w(x)_i^t \mid M_{B(k),i}^t)\hat{}. \quad (115)$$

Eine erste Varianzschätzung lässt sich bereits an dieser Stelle durchführen mit:

$$\text{var}(w(x)_i^t) = 1/(\delta\text{-}1) \sum_i * (f(w(x)_i^t \mid M_{B(k),i}^t)\text{^} - f(w(x)_i^t \mid M_{B(k),i}^t, k = 1, ..., \delta)\text{^})^2. \quad (116)$$

Zusammen mit der vorhandenen Outputschätzung $f(w(x)_i^t \mid M_i^t)$^ durch das Gesamtmodell, d.h. durch die gesamten lokalen Lernmengen M_i^t, wird die 0,632-Bias $\text{bias}(w(x)_i^t)_{0,632}$ bestimmt:

$$\text{bias}(w(x)_i^t)_{0,632} = [f(w(x)_i^t \mid M_i^t)\text{^} - f(w(x)_i^t \mid M_{B(k),i}^t, k = 1, ..., \delta)\text{^}]/0{,}632, \quad (117)$$

mit der die Bias-Korrektur zu $f(w(x)_i^t \mid M_i^t)_{0,632\text{bias}}$^ durchgeführt werden kann. Auf diese Weise besitzt man bereits an der Stelle des Gewichtsvektors $w(x)_i^t$ eine Biasschätzung mit $\text{bias}(w(x)_i^t)$, und mit $f(w(x)_i^t \mid M_i^t)_{0,632\text{bias}}$^ als Schätzung des richtigen Outputwertes $f(w(x)_i^t)$ lässt sich eine Bootstrap-Varianzschätzung durchführen:

$$\sigma(f(w(x)_i^t))^2\text{^} = 1/(\delta\text{-}1) \sum_i * (f(w(x)_i^t \mid M_{B(k),i}^t)\text{^} - f(w(x)_i^t \mid M_i^t)_{0,632\text{bias}}\text{^})^2. \quad (118)$$

Um eine Output-Varianz nach Cohn et al. (1995[74]) an der Stelle eines Gewichtsvektors zu bestimmen, ist die Outputschätzung $f(w(x)_i^t \mid AM(x)^t)$ durch das Modell $AM(x)^t$ an der Stelle des Gewichtsvektors notwendig. Dies kann durch eine Kernel-Regression auf der Basis der restlichen Gewichtsvektoren erfolgen:

$$f(w(x)_i^t \mid AM(x)^t) := 1/\nu_i \sum_k h(d_X(w(x)_i^t, w(x)_k^t)) * f(w(x)_k^t)\text{^},$$
$$\nu_i = \sum_k h(d_X(w(x)_i^t, w(x)_k^t)), \forall n_k^t \in N^t \setminus \{n_i^t\}. \quad (119)$$

Für die Output-Varianz an der Stelle des Gewichtsvektors $w(x)_i^t$ ergibt sich somit die Schätzung:

$$\sigma(f(w(x)_i^t))^2\text{^} = 1/\nu_i \sum_k h(d_X(w(x)_i^t, w(x)_k^t)) * (f(x_j^t \mid AM(x)^t) - f(w(x)_i^t \mid M_i^t)_{0,632\text{bias}}\text{^})^2,$$
$$\nu_i = \sum_k h(d_X(w(x)_i^t, w(x)_k^t)), \forall n_k^t \in N^t \setminus \{n_i^t\}. \quad (120)$$

Die Bootstrapschätzungen sind in Abhängigkeit von der Anzahl δ der Bootstraplisten aufwendig, sodass auch nach Alternativen zur Bestimmung von $\sigma(f(w(x)_i^t))^2$^ und $\text{bias}(w(x)_i^t)^2$^ gesucht werden soll. Diese beiden Werte könnten durch ein Regressionsmodell aus den Stimuli $(x_j^t, f(x_j^t))$ aus M^t oder M_i^t in der gleichen Weise erzeugt werden, wie die Outputschätzung $f(w(x)_i^t)$^ durch die richtigen Outputwerte $f(x_j^t)$. Dies ergibt sich daraus, dass aufgrund der richtigen Outputwerte jedem Stimulus seine Output-Varianz und seine Bias berechnet werden kann, wodurch eine Stimulusstruktur erzeugt wird:

$$m_j^t = (x_j^t, f(x_j^t), \sigma(f(x_j^t))^2, \text{bias}(x_j^t \mid AM(x)^t)^2). \quad (121)$$

Wird die Regression auf die Stimuli in der Lernmenge M_i^t eines Neurons n_i^t beschränkt, so wird für jeden Gewichtsvektor die folgenden Berechnungen angestellt, wobei der Faktor ν_i genau einmal bestimmt werden muss:

$$f(w(x)_i^t)\text{^} = 1/\nu_i \sum_i h(d_X(w(x)_i^t, x_{ij}^t)) * f(x_{ij}^t),$$
$$\sigma(f(w(x)_i^t))^2\text{^} = 1/\nu_i \sum_i h(d_X(w(x)_i^t, x_{ij}^t)) * \sigma(f(x_{ij}^t))^2,$$

$$\text{bias}(w(x)_i^t)^{2\wedge} = 1/\nu_i \, \Sigma_i \, h(d_X(w(x)_i^t, x_{ij}^t)) * \text{bias}(x_{ij}^t \mid AM(x)^t)^2,$$
$$\nu_i = \Sigma_i \, h(d_X(w(x)_i^t, x_{ij}^t)), \forall m_{ij}^t \in M_i^t. \quad (122)$$

Um die Schätzungen weiter zu verbessern, können diese Berechnungen jeweils um ein Bootstrap-Verfahren erweitert werden, indem δ Stimulus-Bootstraplisten erzeugt werden, die jeweils eine Schätzung ergeben, die durch den arithmetischen Mittelwert zu einer Bootstrap-Gesamtschätzung aggregiert werden, worauf jedoch nicht näher eingegangen werden soll.

2.3.6) Suche nach Inputvektoren mit extremalen Outputwerten

Wird der Generalisierungsfehler durch eine gegebene Stimulusmenge bestimmt, so sind die Ergebnisse der Qualitätsbestimmung eines Approximationsmodells AM(x) abhängig von der Zusammensetzung dieser Stimulusmenge. Würde die Möglichkeit bestehen, neue Stimuli zu erheben und das betrachtete Modell AM(x) anhand dieser Stimuli zu testen, so könnte nach der globalen Spannweite des Qualitätsmaßes gesucht werden, indem durch zwei Optimierungsverfahren nach dem Stimulus mit dem kleinsten und dem größten Qualitätsmaß, bezogen auf AM(x), gesucht wird.

Diese Vorgehensweise ist jedoch in der Regel nicht praktikabel, da die Erhebung von Stimuli, d.h. die Bestimmung des richtigen Outputs zu einem Punkt des Inputraumes X einen Aufwand einer bestimmten Größe erfordert, und für die beiden Optimierungsläufe eine Vielzahl von Stimuli benötigt werden. Weiterhin könnten diese neuen Stimuli in einen neuen Lernprozess des Modells einbezogen werden, um den Aufwand der Stimuluserzeugung in einen Ertrag in Form eines besseren Modells umzusetzen.

Besteht jedoch die Möglichkeit, auf der Basis eines gegebenen Modells jedem Inputvektor eine Fehlerschätzung, bzw. eine Bias- und eine Outputvarianz-Schätzung zuzuordnen, da müssen keine neuen Stimuli erzeugt werden, um einen Optimierungsprozess durchzuführen, der nach der Bias- bzw. der Output-Varianz-Spannbreite, oder nach den beiden Spannbreiten sucht.

Weiterhin können die Schätzungen, die während der Suchprozesse ermittelt wurden, in Form von diskreten Häufigkeitsverteilungen umgesetzt werden, aus denen stetige Verteilungen abgeleitet werden können. Je nach den verwendeten parametrischen Modellen, die der Erzeugung einer stetigen Verteilung zugrunde liegen, können maximal erwartete Fehler als Schnittpunkte mit der Qualitätsachse ermittelt werden, oder asymptotische Aussagen über die Wahrscheinlichkeit sehr hoher Fehlerwerte getroffen werden.

2.3.7) Modellbewertung durch Momente höherer Ordnung

Die Zerlegung des MSE in eine Biasquadrat- und eine Varianzkomponente durch Geman et al. (1992[142]), sowie ein anderer Ansatz durch Wolpert (1995a[368]), der Covarianzen berücksichtigt, legt nahe, dass weitere Zerlegungsmöglichkeiten existieren, die zur Modellbewertung herangezogen werden können. Eine parametrische Form der Modellbewertung kann durch Momente höherer Ordnung erfolgen. Liegt eine Liste $M = (x_i \mid i = 1, ..., \mu_M)$ von Werten oder Vektoren vor, so berechnet sich das allge-

meine Moment r‘ter Ordnung bezogen auf einen Wert oder Vektor c durch:

$$mom(r,c) = 1/\mu_M * \sum_i (x_i - c)^r,\ r \in N_0. \quad (123)$$

Von besonderem Interesse ist das zentrale Moment $mom(r,\bar{x})$ r‘ter Ordnung, das sich auf den Mittelwert oder Mittelwertsvektor bezieht. Der Mittelwert kann auch als Moment mom(1,0) erster Ordnung mit c = 0 beschrieben werden, sodass insgesamt gilt:

$$mom(r,\bar{x}) = 1/\mu_M * \sum_i (x_i - \bar{x})^r, \text{ mit}$$
$$\bar{x} = mom(1,0) = 1/\mu_M * \sum_i (x_i - 0)^1 = 1/\mu_M * \sum_i x_i. \quad (124)$$

Die Varianz var(x) der Liste M bestimmt sich als zentrales Moment zweiter Ordnung durch:

$$var(x) = mom(r = 2,\bar{x}) = 1/\mu_M * \sum_i (x_i - \bar{x})^2. \quad (125)$$

2.3.7.1) Output- und Fehler-Moment bei einem Modell

Die Bewertung eines Modells AM(x) soll bezüglich einer Stimulusliste M durchgeführt werden, das die folgende Struktur besitzt:

$$M = (m_i = (x_i, f(x_i), f(x_i \mid AM(x)), \Delta f_i^2) \mid$$
$$x_i \in R^n;\ f(x_i) \in R;\ f(x_i \mid AM(x)) \in R;\ \Delta f_i^2 = (f(x_i) - f(x_i \mid AM(x)))^2;\ i = 1, ..., \mu_M). \quad (126)$$

Die Momente beziehen sich bei einer Modellbewertung allgemein auf die Outputwerte bzw. auf Funktionen der Outputwerte, wie dem quadratischen Fehler als einer Fehlerfunktion. Es sollen zentrale Momente r‘ter Ordnung betrachtet werden, wobei für den Mittelwert $\bar{c}$ gelten kann:

$$mom(r, M, \bar{c}) \text{ , mit } \bar{c} \in \{\bar{f}(x), \bar{f}(x)\hat{}, \Delta\bar{f}^2\}, \text{ mit}$$
$$\bar{f}(x) = 1/\mu_M * \sum_i f(x_i),$$
$$\bar{f}(x)\hat{} = 1/\mu_M * \sum_i f(x_i \mid AM(x)),$$
$$\Delta\bar{f}^2 = 1/\mu_M * \sum_i \Delta f_i^2. \quad (127)$$

Werden Einzelwerte und Mittelwerte des gleichen Typs verwendet, d.h. Outputwerte und deren Mittelwert, Outputschätzungen und deren Mittelwert, sowie Fehlerwerte und deren Mittelwert, so ergibt sich allgemein bzw. speziell:

$$mom(r, M, \bar{c}) = 1/\mu_M * \sum_i (c_i - \bar{c})^r, \text{ bzw.}$$
$$mom(r, M, \bar{f}(x)) = 1/\mu_M * \sum_i (f(x_i) - \bar{f}(x))^r,$$
$$mom(r, M, \bar{f}(x)\hat{}) = 1/\mu_M * \sum_i (f(x_i \mid AM(x)) - \bar{f}(x)\hat{})^r,$$
$$mom(r, M, \Delta\bar{f}^2) = 1/\mu_M * \sum_i (\Delta f_i^2 - \Delta\bar{f}^2)^r. \quad (128)$$

Die Momentbildung $mom(r, M, \bar{f}(x)\hat{})$ ist unabhängig von der Kenntnis der richtigen Outputwerte $f(x_i)$, sodass $mom(r, M, \bar{f}(x)\hat{})$ für beliebige Punktmengen in X berechnet werden können, unabhängig von

einer Stimulusmenge M. Dies gilt nicht für die beiden anderen Momenttypen, da die richtigen Outputwerte direkt oder indirekt erforderlich sind.

2.3.7.2) Output- und Fehler-Moment bei einer Modellmenge

Existiert eine Menge von Modellen AM = $\{AM(x)_k, k = 1, ..., \delta\}$, so kann das r'te Moment bezüglich eines einzelnen Stimulus mit seinem Inputvektor bzw. seinem Outputwert angegeben werden. Gegeben sei der Stimulus $m_i = (x_i, f(x_i))$, für den die Modelle die Outputschätzungen $f(x_i)_k\hat{} = f(x_i \mid AM(x)_k)$ liefern. Zunächst wird der Mittelwert $\bar{f}(x_i)_{AM}\hat{}$ der Outputschätzungen gebildet, der als Bezugspunkt der Moment-Bildung dient:

$$\bar{f}(x_i)_{AM}\hat{} = 1/\delta * \sum_{k=1->\delta} f(x_i \mid AM(x)_k). \quad (129)$$

Das zentrale r'te Output-Moment bezüglich eines Stimulus m_i und einer Menge von Modellen $AM(x)_k$, k = 1, ..., δ, ergibt sich somit:

$$mom(r, m_i, \bar{f}(x_i)_{AM}\hat{}) = 1/\delta * \sum_{k=1->\delta} (f(x_i \mid AM(x)_k) - \bar{f}(x_i)_{AM}\hat{})^r. \quad (130)$$

Diese Momentbildung ist unabhängig von der Kenntnis des richtigen Outputwertes $f(x_i)$, sodass diese Momente für alle Punkte des Inputraumes X berechnet werden können, unabhängig von einer Stimulusmenge.

Anstatt ein Output-Mittelwert zu verwenden kann die Momentbildung wie oben beschrieben auch auf der Basis eines Fehlerwertes $\Delta f(x_i)_k^2 = (f(x_i \mid AM(x)_k) - f(x_i))^2$ durchgeführt werden. Hierzu wird zunächst der Mittelwert der Fehler über alle δ Schätzungen gebildet:

$$\Delta\bar{f}(x_i)^2 = 1/\delta * \sum_{k=1->\delta} \Delta f(x_i)_k^2. \quad (131)$$

Das zentrale r'te Fehler-Moment bezüglich eines Stimulus m_i und einer Menge von Modellen $AM(x)_k$, k = 1, ..., δ, ergibt sich somit:

$$mom(r, m_i, \Delta\bar{f}(x_i)^2) = 1/\delta * \sum_{k=1->\delta} (\Delta f(x_i)_k^2 - \Delta\bar{f}(x_i)^2)^r. \quad (132)$$

Liegt eine Stimulusmenge M und eine Modellmenge AM vor, so sollen letztlich Vergleiche zwischen Modellen $AM(x)_k$ und $AM(x)_p$ durchgeführt werden. Jedem Modell werden durch die Bildung von n Momenten ein n'komponentiger Momentverktor bei zugeordnet, der i mit $C(n, M, AM(x)_k)$ bezeichnet werden soll, wobei die Verwendung von Fehlerwerten unterstellt wird:

$$C(n, M, AM(x)_k) = (mom(r, M, \Delta\bar{f}(x_i)^2) \mid r = 1, ..., n), \text{ bzw.}$$

$$C(n, M, AM(x)_k) = (c(0, M, \Delta\bar{f}(x_i)^2) := 0, c(r, M, \Delta\bar{f}(x_i)^2) \mid r = 1, ..., n). \quad (133)$$

Bei der Verwendung von Fehlerwerten kann der Vergleich zweier solcher Momentvektoren $C(n, M, AM(x)_k)$ und $C(n, M, AM(x)_p)$ als Mehr-Ziel-Minimierungsaufgabe interpretiert werden, die mit Hilfe des Pareto-Kriteriums gelöst werden kann (siehe Abschnitt 2.4)). D.h. Momente gleicher Ordnung r werden mit Hilfe einer Dominanzfunktion verglichen, wobei ein Modell das andere dominiert, wenn sein

Fehler-Moment kleiner ist als die des Konkurrenten. Für die Dominanzfunktion ergibt sich mit $c(r)_k \in \{mom(r, M, \Delta\bar{f}(x_i)^2), c(r, M, \Delta\bar{f}(x_i)^2)\}$, r = 1, ..., n:

$$dom(r)_{k,p} := dom(c(r)_k, c(r)_p) = \begin{cases} -1, & \text{wenn } c(r)_k > c(r)_p, \\ 0, & \text{wenn } c(r)_k = c(r)_p, \\ 1, & \text{wenn } c(r)_k < c(r)_p. \end{cases} \quad (134)$$

Werden alle n Komponenten der Momentvektoren $C(n, M, AM(x)_k)$ und $C(n, M, AM(x)_p)$ auf diese Weise verglichen, so ergibt sich ein Dominanzvektor $dom_{k,p} := dom(AM(x)_k, AM(x)_p)$, der in Verbindung mit dem Pareto-Kriterium weiter ausgewertet wird. Ein Modell $AM(x)_k$ dominiert $AM(x)_p$, wenn in keiner Komponente von $dom_{k,p}$ ein Wert -1 auftritt und mindestens einmal ein Wert 1. Treten beide Werte aus {-1, 1} auf, so sind beide Modelle nicht dominant. Es ergibt sich somit eine Pareto-Dominanzfunktion $dom_{k,p}$ bezüglich der Modelle $AM(x)_k$ und $AM(x)_p$:

$$dom_{k,p} = \begin{cases} -1, & \text{wenn } [\exists_{\geq 1} dom(r)_{k,p}: dom(r)_{k,p} = -1 \wedge \neg\exists dom(r)_{k,p}: dom(r)_{k,p} = 1], \\ 0, & \text{wenn } [\exists_{\geq 1} dom(r)_{k,p}: dom(r)_{k,p} = 1 \wedge \exists_{\geq 1} dom(r)_{k,p}: dom(r)_{k,p} = -1], \\ 1, & \text{wenn } [\neg\exists dom(r)_{k,p}: dom(r)_{k,p} = -1 \wedge \exists_{\geq 1} dom(r)_{k,p}: dom(r)_{k,p} = 1]. \end{cases} \quad (135)$$

Aus der Menge der Modelle kann mit Hilfe der Matrix aller paarweisen Dominanzwerte die Paretomenge gebildet werden, in welcher die Modelle liegen, die von keinem anderen Modell dominiert werden und untereinander nicht dominant sind. Mit den Dominanzwerten lässt sich auch eine Pareto-Hierarchie bzw. eine Pareto-Wettkampf-Hierarchie aufbauen (siehe Abschnitt 2.4.2)).

2.3.7.3) Schätzung von Output- und Fehler-Momenten

Output-Momente können für beliebige Punkte x_i in X erstellt werden, da für sie die Kenntnis einer Menge von Outputschätzungen ohne die Kenntnis des richtigen Outputwertes $f(x_i)$ notwendig ist. Die instanzbasierten Outputschätzungen $f(x_i \mid AM(x \mid M^t))$ oder die prototypbasierten Schätzungen $f(x_i \mid AM(x \mid N^t))$ erfordern jeweils einen Aufwand, der proportional mit der Anzahl der Stützpunkte wächst, die mit μ_L bezeichnet werden soll. Von Bedeutung ist weiterhin die Anzahl der Outputschätzungen, was beim Vorliegen einer Modellmenge $AM(x) = \{AM(x)_k \mid k = 1, ..., \delta\}$ gleich der Anzahl δ der Modelle ist. Ist die Anzahl der Stützpunkte mit μ_L^t pro Modell für alle verwendeten Modelle gleich, so ergibt sich eine Zeitkomplexität von $O(\mu_L^t * \delta)$ für die Berechnung der Voraussetzungen eines Output-Momentes. Für die Berechnung eines Momentes werden δ Summanden $f(x_i \mid AM(x)_k)$ bzw. $(f(x_i \mid AM(x)_k) - \bar{f}(x_i)_{AM^\wedge})^r$ erfordert, sodass sich $O(\mu_L^t * \delta + \delta)$ ergibt. Als letzter Faktor kommt die Anzahl der Momente mit r > 1 hinzu, sodass sich eine Gesamtkomplexität von $O(\mu_L^t * \delta + r * \delta)$ für genau einen Punkt $x_i \in X$ ergibt. Während einer Iteration t wird jedoch nicht nur von einem Punkt eine Menge von Momenten ermittelt, sondern es liegt eine Kandidatenmenge KM^t mit μ_K^t Elementen vor, sodass eine Komplexität von $O(\mu_K^t * (\mu_L^t * \delta + r * \delta))$ anfällt.

Eine erhebliche Reduktion könnte durchgeführt werden, wenn ein Output-Moment an der Stelle eines Punktes $x_i \in X$ nicht auf der Basis von δ Outputschätzungen berechnet werden, sondern wenn eine stützpunktbasierte Approximation durchgeführt wird. Dabei wird eine Teilmenge $KM^{t'}$ von $\mu_K^{t'}$ Kandidaten selektiert, für welche richtige Output-Momente auf der Basis von δ Outputschätzungen berechnet wer-

den, während die Restmenge $KM^t \setminus KM^{t\text{‘}}$ mit $\mu_K{}^t - \mu_K{}^{t\text{‘}}$ Kandidaten eine stützpunktbasierte Approximation der Momente erhält. Gegeben sei die Menge KM^t von Kandidaten $m_{K,i}{}^t$, denen ein richtiger Outputwert $f(x_{K,i}{}^t)$ fehlt:

$$KM^t = \{m_{K,i}{}^t = (x_{K,i}{}^t, \_) \mid i = 1, ..., \mu_K{}^t\}. \tag{136}$$

Die Zerlegung der Kandidatenmenge in $KM^{t\text{‘}}$ und $KM^t \setminus KM^{t\text{‘}}$ ist für die Zeitkomplexität am wichtigsten, da je kleiner die Anzahl der Kandidaten ist, denen Output-Momente konventionell zugeordnet wird, desto größere Einsparungen sind möglich. Andererseits sinkt mit der Anzahl der Stützpunkte die Qualität der Approximation, sodass genügend Kandidaten in $KM^{t\text{‘}}$ liegen sollen. Verbunden mit der Anzahl ist das Selektionskriterium von Kandidaten aus KM^t die in $KM^{t\text{‘}}$ überführt werden. Es können zufallsbasierte Selektionsverfahren verwendet werden, wie das zufällige Ziehen ohne Zurücklegen. Es können alternativ deterministische Verfahren durchgeführt werden, die Cluster von Kandidaten verwenden, die auf der Basis ihres Inputvektors $x_{K,i}{}^t$ gebildet werden, z.B. durch eine SC-GNG-SOM (siehe Abschnitte 2.1.5) und 2.1.6)). Aus jedem der Cluster kann ein Prototyp verwendet werden, z.B. der Kandidat, dessen Inputvektor den geringsten Abstand von dem Gewichtsvektor der entsprechenden Klasse bzw. des Neurons besitzt. Die externe Vorgabe der Anzahl $\mu_K{}^{t\text{‘}}$ wird in diesem Kontext ersetzt durch die externe Vorgabe eines Qualitätsmaßes wie dem Quantifizierungsfehler der Klassifikation, d.h. eine SC-GNG-SOM wird solange aufgebaut und adaptiert, bis alle Cluster einen Quantifizierungsfehler kleiner-gleich dem externen Schwellenwert besitzen.

Nachdem $KM^{t\text{‘}} = \{m_{K,i}{}^t = (x_{K,i}{}^t, \_) \mid i = 1, ..., \mu_K{}^{t\text{‘}}\}$ erzeugt wurde gibt jedes der Modelle aus $AM(x) = \{AM(x)_k \mid k = 1, ..., \delta\}$ eine Outputschätzung für jedes Element aus $KM^{t\text{‘}}$ ab, gefolgt von der Mittelwertbildung $\bar{f}(x_{K,i}{}^t)_{AM}\text{^}$, sodass sich die erweiterte Datenstruktur ergibt:

$$KM^{t\text{‘}} = \{m_{K,i}{}^t = (x_{K,i}{}^t, \_, f(x_{K,i}{}^t \mid AM(x)_k), \bar{f}(x_{K,i}{}^t)_{AM}\text{^}) \mid k = 1, ..., \delta) \mid i = 1, ..., \mu_K{}^{t\text{‘}}\}. \tag{137}$$

Es folgt der Aufbau der $r = 2, ..., r_{max}$ Output-Momente, wobei die vereinfachte Bezeichnung $mom(r, x_{K,i}{}^t) := mom(r, m_i, \bar{f}(x_i)_{AM}\text{^})$ verwendet werden soll:

$$KM^{t\text{‘}} = \{m_{K,i}{}^t = (x_{K,i}{}^t, \_, f(x_{K,i}{}^t \mid AM(x)_k), \bar{f}(x_{K,i}{}^t)_{AM}\text{^}, mom(r, x_{K,i}{}^t)) \mid k = 1, ..., \delta; r = 2, ..., r_{max}) \mid i = 1, ..., \mu_K{}^{t\text{‘}}\}. \tag{138}$$

Damit existieren die Stützpunkte $(x_{K,i}{}^t, mom(r, x_{K,i}{}^t))$ für ein Approximationsmodell, das durch ein Verfahren wie die Kernel-Regression vollständig spezifiziert wird, und das bezeichnet werden soll mit $AM(mom(r,x) \mid KM^{t\text{‘}})$. Da für jedes r ein solches Modell existiert, wird durch die Datenstruktur von $KM^{t\text{‘}}$ eine Modellmenge AM(mom) mit r_{max} Elementen festgelegt:

$$AM(mom) = \{AM(mom(r,x) \mid KM^{t\text{‘}}) \mid r = 2, ..., r_{max}\}. \tag{139}$$

Im weiteren werden jedem Kandidaten $m_{K,j}{}^t$ aus der Restmenge $KM^t \setminus KM^{t\text{‘}}$ r_{max} Output-Moment-Schätzungen $mom(r, x_{K,j}{}^t)\text{^}$ zugeordnet, wobei eine globale Kernel-Regression unterstellt wird:

$$KM^t \setminus KM^{t\text{‘}} = \{m_{K,j}^t = (x_{K,j}^t, \_, mom(r, x_{K,j}^t)^\wedge \mid r = 2, ..., r_{max}) \mid j = 1, ..., \mu_K^t - \mu_K^{t\text{‘}}\}, \text{ mit}$$

$$mom(r, x_{K,j}^t)^\wedge = mom(r, x_{K,j}^t \mid AM(mom(r,x) \mid KM^{t\text{‘}})) = 1/\nu_{K,j}^t \sum_i h(d_X(x_{K,j}^t, x_{K,i}^t)) * mom(r, x_{K,i}^t),$$

$$\nu_{K,j}^t = \sum_i h(d_X(x_{K,j}^t, x_{K,i}^t)), \forall m_{K,i}^t \in KM^{t\text{‘}}. \quad (140)$$

Bezogen auf die Zeitkomplexität ergibt diese Vorgehensweise, dass im ersten Schnitt der Bestimmung der Stützpunkte eine Komplexität von $O(\mu_K^{t\text{‘}} * (\mu_L^t * \delta + r_{max} * \delta))$ anfält. Im zweiten Schritt werden $\mu_K^t - \mu_K^{t\text{‘}}$ Kandidaten r_{max} Output-Moment-Schätzungen zugeordnet, was eine Komplexität von $O((\mu_K^t - \mu_K^{t\text{‘}}) * (r_{max} * \delta))$ erfordert, sodass sich eine Gesamtkomplexität von $O(\mu_K^{t\text{‘}} * (\mu_L^t * \delta + r_{max} * \delta) + (\mu_K^t - \mu_K^{t\text{‘}}) * (r_{max} * \delta))$ ergibt.

Im Gegensatz zu den Output-Momenten stehen die Fehler-Momente, bei denen die Kenntnis des richtigen Outputs an der Stelle des betrachteten Punktes bzw. der Punktmenge aus X neben den Outputschätzungen notwendig ist. Mit Hilfe von stützpunktbasierten Approximationsverfahren ist es jedoch möglich, Fehlermomente an beliebigen Punkte x in X direkt oder indirekt zu schätzen. Gegeben sei eine Menge $AM(x) = \{AM(x)_k \mid k = 1, ..., \delta\}$ von Modellen sowie eine Menge $M^t = \{m_i^t = (x_i^t, f(x_i^t) \mid i = 1, ..., \mu_M^t\}$ von Stimuli, für die jeweils der richtige Outputwert $f(x_i^t)$ bekannt ist. Die Modelle $AM(x)_k$ könnten ihre Stützpunkte aus M^t durch ein Resampling-Verfahren beziehen. Weiterhin sei eine Kandidatenmenge $KM^t = \{m_{K,j}^t = (x_{K,j}^t, \_) \mid j = 1, ..., \mu_K^t\}$ von Stimuli gegeben, denen jeweils Fehler-Moment-Schätzungen zugeordnet werden sollen.

Bei der <u>indirekten Schätzung</u> wird an der Stelle $x_{K,j}^t$ eine Outputschätzung $f(x_{K,j}^t \mid AM(x)_k)$ durch alle verfügbaren Modelle $AM(x)_k$ durchgeführt. Die Einzelschätzungen werden zu einer Gesamtschätzung $f(x_{K,j}^t \mid AM(x))$ aggregiert, die als Substitution für den richtigen aber unbekannten Wert $f(x_{K,j}^t)$ verwendet wird. Als Aggregationsfunktion kann der arithmetischen Mittelwert

$$f(x_{K,j}^t \mid AM(x)) = 1/\delta \sum_k f(x_{K,j}^t \mid AM(x)_k), \quad (141)$$

verwendet werden oder komplexere Vorgehensweisen auf der Basis von Resamplingverfahren. Für jede der Schätzungen $f(x_{K,j}^t \mid AM(x)_k)$ wird ein Fehlerwert $\Delta f_{k,K,j}^{t2} = (f(x_{K,j}^t \mid AM(x)) - f(x_{K,j}^t \mid AM(x)_k))^2$ berechnet, die zu einem arithmetischen Mittelwert $\Delta\bar{f}_{K,j}^{t2}$ aggregiert werden, der als Basis für die zentralen Momente $mom(r, \Delta\bar{f}_{K,j}^{t2})$ dient:

$$mom(r, \Delta\bar{f}_{K,j}^{t2}) = 1/\delta * \sum_{k=1->\delta} (\Delta f_{k,K,j}^{t2} - \Delta\bar{f}_{K,j}^{t2})^r. \quad (142)$$

Für die Datenstruktur der Kandidaten ergibt sich für die indirekten Schätzung:

$$KM^t = \{m_{K,i}^t = (x_{K,i}^t, \_, f(x_{K,j}^t \mid AM(x)_k), f(x_{K,j}^t \mid AM(x)), \Delta f_{k,K,j}^{t2}, \Delta\bar{f}_{K,j}^{t2}, mom(r, \Delta\bar{f}_{K,j}^{t2}) \mid$$
$$k = 1, ..., \delta; r = 2, ..., r_{max}) \mid i = 1, ..., \mu_K^t\}. \quad (143)$$

Bei der <u>direkten Schätzung</u> wird die Datenstruktur der Stimuli aus M^t erweitert, die als Stützpunktmenge der Approximation verwendet werden. Im ersten Schritt bildet jedes instanzbasierte Modell $AM(x)_k$ für jeden Stimulus eine Outputschätzung, aus der ein Fehlerwert bestimmt wird, wobei ein Gewinnerlisten-Verfahren verwendet werden soll (siehe Abschnitt 2.3.8)), ohne dass dies hier für den weiteren Ablauf von Bedeutung ist. Es ergibt sich die erweiterte Datenstruktur der Stimuli:

$$M^t = \{m_i^t = (x_i^t, f(x_i^t), f(x_i^t \mid AM(x)_k), \Delta f_{k,i}^{t2} \mid k = 1, ..., \delta) \mid i = 1, ..., \mu_M{}^t\}, \text{ mit}$$
$$\Delta f_{k,i}^{t2} = (f(x_i^t) - f(x_i^t \mid AM(x)_k))^2. \quad (144)$$

Der nächste Schritt besteht darin, den einzelnen Stimuli Fehler-Momente zuzuordnen, wobei zunächst der Mittelwert $\Delta\bar{f}_i^{t2}$ gebildet wird, der als Bezugspunkt der zentralen Momente $mom(r, x_i^t)$ r'ter Ordnung verwendet wird:

$$\Delta\bar{f}_i^{t2} = 1/\delta \sum_k \Delta f_{k,i}^{t2}. \quad (145)$$

Für die Datenstruktur der Stützpunkte ergibt dies die abschlissende Struktur:

$$M^t = \{m_i^t = (x_i^t, f(x_i^t), f(x_i^t \mid AM(x)_k), \Delta f_{k,i}^{t2}, \Delta\bar{f}_i^{t2}, mom(r, x_i^t) \mid k = 1, ..., \delta; r = 2, ..., r_{max}) \mid$$
$$i = 1, ..., \mu_M{}^t\}, \text{ mit}$$
$$mom(r, \Delta\bar{f}_i^{t2}) = 1/\delta * \sum_k (\Delta f_{k,i}^{t2} - \Delta\bar{f}_i^{t2})^r. \quad (146)$$

Die Schätzung des r'ten Fehler-Momentes an der Stelle eines Kandidaten in X, zu dem kein richtiger Outputwert bekannt ist, ergibt sich somit durch die LWR-Schätzung der r'ten Fehler-Momente aller Stimuli m_i^t aus M^t, wobei der zentrale Wert $\Delta\bar{f}_{K,j}^{t2}$ des Kandidaten unbekannt bleibt und für die Berechnung selbst nicht benötigt wird:

$$mom(r, \Delta\bar{f}_{K,j}^{t2}) = 1/\nu_{K,j}^t \sum_i h(d_X(x_i^t, x_{K,j}^t)) * mom(r, \Delta\bar{f}_i^{t2}),$$
$$\nu_{K,j}^t = \sum_i h(d_X(x_i^t, x_{K,j}^t)), \forall m_i^t \in M^t. \quad (147)$$

Für die Datenstruktur der Kandidaten ergibt sich für die direkte Schätzung:

$$KM^t = \{m_{K,i}^t = (x_{K,i}^t, \_, mom(r, \Delta\bar{f}_{K,j}^{t2}) \mid r = 2, ..., r_{max}) \mid i = 1, ..., \mu_K{}^t\}. \quad (148)$$

2.3.8) Modellqualität durch Gewinnerlisten-Verfahren

Ist die Ermittlung von Stimuli (x, f(x)) eine sehr aufwendige Operation, so ist es sinnvoll, so viele Stimuli wie möglich aus einer gegebenen Grundmenge in den Aufbau des Modells zu investieren. Entsprechend der klassischen Sichtweise muss jedoch immer eine Testmenge erzeugt werden, wie klein diese auch sei, wobei die Arbeiten von Amari et al. (1996a[6], b[7]) zeigen, dass die Testmenge nur wenige Prozent der Gesamtstimuli enthalten muss, um effektive Aussagen über die Modellqualität zu treffen. Im weiteren soll ein Vorschlag für eine Verfahrensklasse gemacht werden, die als Gewinnerlisten-Verfahren bezeichnet werden soll, und bei der alle Stimuli zum Lernen verwendet werden können, und trotzdem kann dem stützpunktorientierten Modell ein Qualitätsmaß zugeordnet werden, wobei ein LWR-Verfahren für die Durchführung der Schätzungen verwendet wird. Das Prinzip dieser Verfahren wurde in Bachelier (1999c[21]) eingeführt, um Anwendungsregionen von Approximationsmodellen mit Hilfe der Stimuli in einer gegebenen Generation schätzen zu können, wobei der Ursprung der Verfahrensklasse auf Darstellungen im Zusammenhang mit den Interpolierenden SOM zurückgeht (I-SOMs, Göppert (1997[149]), Bachelier (1998b[16])).

Die Haupteigenschaft der Gewinnerlisten-Verfahren besteht darin, dass alle Stimuli unter bestimmten Bedingungen als Lernstimuli, aber auch als Teststimuli verwendet werden können. D.h. allen Stimuli kann eine Outputschätzung durch das Modell bzw. ein modifiziertes Modell zugeordnet werden, die im nächsten Schritt zu Einzelfehlern, und zu einem Modellfehler aggregiert werden. Die Modifikation dient dazu, das Overfitting des Modells an den Stützpunkten zu umgehen, da sonst die Modellqualität viel zu gut geschätzt wird, was daraus resultiert, dass die Abweichungen zwischen Istwert und Schätzungen an den Stützpunkten bei den Regressionsverfahren besonders klein sind. Die Verwendung eines Regressionsverfahrens ist hier nicht unbedingt notwendig, da bei einem Interpolationsverfahren das Modell auf die gleiche Weise modifiziert werden kann, sodass Abweichungen zwischen den beiden Werten erzeugt werden können.

Durch die Modifikation des zu bewertenden Modells wird ein neues Qualitätsmaß erzeugt, mit dem alternative Modelle für eine Selektionsoperation bewertet werden können, oder mit dem ein Abbruchkriterium geprüft werden kann, wenn ein entsprechend modifizierter Schwellenwert verwendet wird. Ein Abbruchkriterium kann ebenfalls über den Fortschritt der Modellbildung formuliert werden, sodass die Verwendung eines originalen oder eines modifizierten Qualitätsmaß irrelevant wird.

Bei einem gobalen, instanzbasierten LWR-Verfahren dient die Bestimmung der Modellqualität keinem dieser Zwecke, da faktisch kein Lernen durchgeführt wird, weil alle Stützpunkte bekannt sind und verwendet werden, wenn das Schätzverfahren eine extern gegebene Kernelfunktion mit festgelegten Parametern besitzt. Denkbar wäre jedoch ein Lernverfahren, das in diesem Fall den oder die Parameter der Kernelfunktion ermittelt, bei einem gegebenen Typ der Funktion. Die Zuordnung von f(x)-Schätzwerten zu den Instanzen ermöglicht es den Instanzen Bias- und Output-Varianzwerte zuzuordnen, sodass ein LWR-Bias- und Output-Varianz-Approximationsmodell gebildet werden kann (siehe Abschnitt 2.3.5)).

Es soll im weiteren der Fall eines instanzbasierten Modells betrachtet werden, d.h. zur Schätzung des Outputs f(x) werden die Input- und Outputkomponenten von Stimuli verwendet, wobei davon ausgegangen werden soll, dass ein globales Verfahren verwendet wird, d.h. alle vorliegenden Stimuli werden zur Schätzung herangezogen. Werden prototypbasierte Modelle verwendet, bei denen die Stimuli richtige Outputwerte besitzen, und den Prototypen sollen ebenfalls richtige Outputwerte zugeordnet werden, so können die Schätzungen der Prototyp-Outputwerte durch die Stimuli oder durch die anderen Prototypen erfolgen, wobei die zuletzt genannte Möglichkeit der Vorgehensweise entspricht, dass eine Stimulus-Outputschätzung durch die anderen Stimuli erfolgt.

Wird ein Stimulus $(x_j, f(x_j)) \in M$ betracht, dem eine Schätzung und somit ein Fehlerwert zugeordnet werden soll, so wird dieser Stimulus bei einem Gewinnerlisten-Verfahren behandelt, als wenn er der gesamten Stützpunktmenge M präsentiert wird, d.h. es werden die Distanzen $d_X(x_j, x_k)$ zwischen x_j und den Inputvektoren aller Stimuli aus M gebildet. Die μ Stimuli aus M werden nach steigender Distanz in einer Gewinnerliste $G(x_j \mid M)$ geordnet, sodass das erste Element der präsentierte Stimulus selbst ist:

$$G(x_j \mid M) = (m_{s(i|j)} \in M \mid d_X(x_j, x_{s(i|j)}) < d_X(x_j, x_{s(i+1|j)}), i = 1, ..., \mu), \text{ mit}$$
$$m_{s(1|j)} = m_j. \tag{149}$$

Bei der Standard-Vorgehensweise einer globalen LWR würden alle Stützpunkte aus $G(x_j \mid M)$ für die Schätzung heran gezogen werden, während die erste Instanz aus der Klasse der Gewinnerlisten-Verfahren das erste Element aus $G(x_j \mid M)$ ignoriert, und die Schätzung mit den verbleibenden Stützpunkten durchführt, d.h. es wird eine spezielle Form der lokalen LWR-Schätzung durchgeführt. In diesem Fall müsste faktisch keine geordnete Gewinnerliste gebildet werden, da alle Stützpunkte außer dem präsentierten Stützpunkt verwendet werden, d.h. alle Elemente aus $M \setminus \{m_j\}$:

$$f(x_j)^\wedge = 1/\nu_j \sum_k h(d_X(x_j, x_k)) * f(x_k),$$
$$\nu_j = \sum_k h(d_X(x_j, x_k)), \forall m_k \in M \setminus \{m_j\}. \quad (150)$$

Um die einzelnen Instanzen der Gewinnerlisten-Verfahren differenzieren zu können, soll die Bezeichnung $G(x_j \mid M, i_{start}, i_{end})$ eingeführt werden, die eine zusammenhängende Teilliste aus $G(x_j \mid M)$ beschreibt, mit i_{start} als dem ersten Stimulus in der geordneten Liste $G(x_j \mid M)$, der einbezogen wird, und i_{end} als dem letzten Stimulus, der in die gewichtete Summe der Kernel-Regression einbezogen wird. Gewinnerlisten-Verfahren, die keine zusammenhängende Teilliste verwenden, wie z.B. die Zufallsauswahl von Stimuli, sollen im weiteren nicht betrachtet werden. Weiterhin sollen ausschließlich die Fälle betrachtet werden, bei denen der zweite Parameter i_{end} gleich μ gesetzt wird, und nur der erste Parameter i_{start} variiert. Die oben dargestellte Schätzung lässt sich mit Hilfe dieser Bezeichnung beschreiben als

$$f(x_j \mid i_{start}, i_{end})^\wedge = f(x_j \mid 2, \mu)^\wedge = 1/\nu_j \sum_k h(d_X(x_j, x_k)) * f(x_k),$$
$$\nu_j = \sum_k h(d_X(x_j, x_k)), \forall m_k \in G(x_j \mid M, 2, \mu). \quad (151)$$

Wird der erste Stimulus aus $G(x_j \mid M)$ in die Schätzung nicht mit einbezogen, so wird dadurch der Fall simuliert, dass zwischen dem Inputvektor eines präsentierten Stimulus und dem Inputvektor des nächsten Stützpunktes eine deutliche Distanz liegt. Würde der erste Stimulus, also m_j selbst einbezogen werden, so würde das Gewicht $h(d_X(x_j, x_j))$ die gesamte gewichtete Summe dominieren, und die anderen $\mu - 1$ Summanden würden nur wenig beitragen, was durch die Form der Kernelfunktion verstärkt oder abgeschwächt werden kann. Ist h(.) eine Expotentialfunktion, so wird die Dominanz von m_j wesentlich verstärkt, wenn m_j einbezogen wird.

Die einfachste Nutzung dieses Gewinnerlisten-Ansatzes besteht darin, jeden Stimulus m_j aus M mit genau einer Schätzung $f(x_j \mid 2, \mu)^\wedge$ zu bewerten, und jeweils den Deltawert $\Delta f(x_j)$ zu erzeugen, der nicht weiter als $\Delta f(x_j \mid 2, \mu)^\wedge$ spezifiziert werden muss, wenn jeweils nur eine Schätzung verwendet wird:

$$\Delta f(x_j)^2 = (f(x_j) - f(x_j \mid 2, \mu)^\wedge)^2. \quad (152)$$

Die Stimuli besitzen nach dieser Operation die Form

$$M = (m_j = (x_j, f(x_j), f(x_j \mid 2, \mu)^\wedge, \Delta f(x_j)^2) \mid j = 1, ..., \mu). \quad (153)$$

Die Bewertung des instanzbasierten Modells AM(x) mit M als Stützpunktmenge kann als MSE aus den quadrierten Deltawerten gebildet werden:

$$MSE(AM(x) \mid M) = 1/\mu * \sum_j \Delta f(x_j)^2, \forall m_j \in M. \quad (154)$$

Ausgehend von dieser Basisvorgehensweise existiert eine Anzahl von Varianten, die im weiteren angedeutet werden sollen. Wird die unveränderte Struktur $m_j = (x_j, f(x_j), f(x_j \mid 2, \mu)^\wedge, \Delta f(x_j)^2)$ unterstellt, so können Resampling-Verfahren angewendet werden, wobei ein einfaches Paar-Bootstrap-Verfahren dargestellt werden soll. Ausgehend von M werden δ Bootstraplisten B_k erzeugt, indem jeweils μ mal mit Zurücklegen ein Individuum gezogen wird. Für jede der Bootstraplisten B_k wird ein Einzel-Bootstrap-MSE-Wert gebildet, die zu einem Gesamt-Bootstrap-MSE-Wert aggregiert werden:

$$\begin{aligned} MSE(AM(x) \mid B) &= 1/\delta * \sum_k MSE(AM(x) \mid B_k), \forall B_k \in B, \\ MSE(AM(x) \mid B_k) &= 1/\mu * \sum_j \Delta f(x_{kj})^2, \forall m_{kj} \in B_k, \\ B &= \{B_k = (m_{kj} \mid j = 1, ..., \mu) \mid k = 1, ..., \delta\}. \end{aligned} \quad (155)$$

Andere Verfahrensvarianten berechnen die Bias- und Output-Varianzwerte an der Stelle eines Stimulus, wobei die Summe bei der Outputvarianz alle Stimuli m_k aus M außer dem betrachteten Stimulus m_j verwendet. Um zu kennzeichnen, dass die Bias- und Output-Varianzwerte im Rahmen einer Gewinnerliste $G(x_j \mid M, 2, \mu)$ bestimmt werden, sollen die Bezeichnungen $bias(x_j \mid 2, \mu)$ und $\sigma(f(x_j \mid 2, \mu))^2$ verwendet werden:

$$\begin{aligned} bias(x_j \mid 2, \mu) &= f(x_j \mid 2, \mu) - f(x_j), \\ \sigma(f(x_j \mid 2, \mu))^2 &= 1/(\mu\text{-}2) \sum_k (f(x_j \mid 2, \mu) - f(x_k))^2, \forall m_k \in G(x_j \mid M, 2, \mu), \end{aligned} \quad (156)$$

sodass sich die folgende Stimulusstruktur ergibt:

$$M = (m_j = (x_j, f(x_j), f(x_j \mid 2, \mu)^\wedge, bias(x_j \mid 2, \mu), \sigma(f(x_j \mid 2, \mu))^2 \mid j = 1, ..., \mu). \quad (157)$$

Die Nutzung der Bias und der Output-Varianz ist analog zu der Nutzung des quadratischen Fehlers, d.h. die Werte können über alle Stimuli aggregiert werden, um einen Bias- und einen Output-Varianzwert für das Modell AM(x) zu erhalten. Die gleiche Vorgehensweise kann bei der Verwendung von zentralen Momenten höherer Ordnung durchgeführt werden.

Eine andere große Verfahrensklasse zeichnet sich dadurch aus, dass den einzelnen Stimuli aus M eine Menge von Schätzungen $f(x_j \mid i_{min}, \mu)^\wedge$, $i_{min} = 2, ...$, zugeordnet werden, die in einem Zwischenschritt zu einer Schätzung aggregiert werden, mit welcher der Deltawert gebildet wird. Die quadrierten Deltawerte werden dann zu einem MSE-Wert aggregiert. Diese Verfahrensklasse ist unabhängig von der vorangegangenen Resampling-Klasse, da Bootstraplisten von IStimuli gebildet werden können, die selbst Schätzungsmengen enthalten. Von besonderem Interesse ist dabei die Verwendung der Sequenz der μ-1 Listen $G(x_j \mid M, 2, \mu)$ bis $G(x_j \mid M, \mu, \mu)$, mit denen jeweils eine Schätzung $f(x_j \mid i_{min}, \mu)^\wedge$ erzeugt wird, sodass sich eine Stimulusstruktur mit $\mu + 1$ Komponenten ergibt:

$$M = (m_j = (x_j, f(x_j), f(x_j \mid i_{min}, \mu)^\wedge \mid i_{min} = 2, ..., \mu) \mid j = 1, ..., \mu). \quad (158)$$

Die Sequenz der Schätzungen repräsentiert die Fähigkeit des Modells immer weiter entfernte Stimuli zu schätzen, da die Distanz zwischen x_j und dem Inputvektor $x_{s(i(min)|j)}$ des ersten Stützpunktes mit zunehmendem Wert i_{min} zunimmt. Da bei jeder dieser Schätzungen immer ein Stützpunkt weniger verwendet wird, steigt jedoch die Unsicherheit ebenfalls mit zunehmender Indexzahl i_{min}, sodass es sinnvoll ist, zusätzlich Qualitätsmaße zu verwenden, welche parallel die Unsicherheit der Schätzungen bewerten.

Es existieren verschiedene Formen der Aggregation dieser Werte, wie die horizontale Aggregation, bei der alle Schätzungen eines Stimulus aggregiert werden, wobei das Problem dieser Vorgehensweise in der wachsenden Unsicherheit der Schätzungen besteht. Sinnvoll ist daher eine gewichtete Aggregation, die weiter entfernten Stützpunkten ein geringeres Gewicht zuordnet. Denkbar ist in diesem Zusammenhang ein LWR-Verfahren wie z.B.:

$$f(x_j \mid i_{min} = 1,...,\mu, \mu)^\wedge = 1/\nu_j \sum_{i(min)=2->\mu} h(d_X(x_j, x_{s(i(min)|j)})) * f(x_j \mid i_{min}, \mu)^\wedge,$$
$$\nu_j = \sum_{i(min)=2->\mu} h(d_X(x_j, x_{s(i(min)|j)})), \forall f(x_j \mid i_{min}, \mu)^\wedge \in m_j. \quad (159)$$

Alternativ dazu existiert die vertikale Aggregation, bei der die Schätzung einer Indexzahl über alle Stimuli aggregiert werden, d.h. $f(x_j \mid i_{min}, \mu)^\wedge$, für alle Stimuli aus M:

$$f(M \mid i_{min}, \mu)^\wedge = 1/\mu \sum_j f(x_j \mid i_{min}, \mu)^\wedge, \forall m_j \in M. \quad (160)$$

Die vertikalen Aggregationen sind direkt interpretierbar als Qualitätsmaße des instanzbasierten Modells. Ein Maß $f(M \mid i_{min}, \mu)^\wedge$ ist als eine indirekte Form von einem distanzabhängigem Qualitätsmaß interpretierbar, da mit größer werdendem Indexwert i_{min} die Distanz in X zwischen dem Inputvektor des zu schätzenden Stimulus und dem am nächsten liegenden Stützpunkt größer wird. Indirekt ist diese Beziehung dadurch, dass das Maß rangbasiert ist, sodass benachbarte Ränge große oder kleine Distanzen in X besitzen können. Eine direkte Beziehung könnte eingeführt werden, indem eine Schätzung $f(x_j \mid i_{min}, \mu)^\wedge$ zusammen mit der Distanz $d_X(x_j, x_{s(i(min)|j)})$ ermittelt und verwendet wird, sodass z.B. zwei Mittelwerte $f(M \mid i_{min}, \mu)^\wedge$ und $d_X(M, x_{s(i(min)|j)})$ gebildet werden, mit

$$d_X(M, x_{s(i(min)|j)}) = 1/\mu \sum_j d_X(x_j, x_{s(i(min)|j)}), \forall m_j \in M. \quad (161)$$

Dies ist interpretierbar als ein Fehlermaß des Modells für Stimuli, deren Inputvektor eine durchschnittliche Distanz von $d_X(M, x_{s(i(min)|j)})$ von dem Inputvektor des Stützpunktes besitzen, der am nächsten liegt. Zu dieser Vorgehensweise sind eine Reihe von Varianten denkbar, wie z.B. die Verwendung der mittleren Distanz der Stützpunkte anstatt der kleinsten Distanz, die im Rahmen der Schätzung von expliziten Anwendungsregionen durch generationsinterne Stimuli in Bachelier (1999c[21]) detailliert dargestellt wurden, und auf die hier nicht weiter eingegangen werden soll.

2.4) Hierarchische Strukturierung bei Mehr-Ziel-Optimierungen

Ein Mehr-Ziel-Optimierungsproblem wird durch eine Menge von n Inputvariablen $x_i = (x_{ij} \mid j = 1, ..., n) \in X \subseteq R^n$, einer Menge von m Zielfunktionen $f_k(x)$, k = 1, ..., m, sowie einer Menge von p Constraints $g_l(x)$, l = 1, ..., p, beschrieben (siehe z.B. Binh & Korn (1996[46], 1997[46]), Veldhuizen (1999[347]), Zitzler (1999[375])). Ein Inputvektor x_i wird auch als Entscheidungsvektor (decision vector), und ein Ergebnis- oder Outputvektor $f(x_i)$ wird auch als Zielvektor (objective vector) bezeichnet. Der n-dimensionale Teilraum X des Inputraumes R^n wird auch als Entscheidungsraum und der m-dimensionale Teilraum Y des Outputraumes R^m wird auch als Zielraum bezeichnet. Formuliert wird ein Mehr-Ziel-Minimierungsproblem durch:

Minimiere:	$y = f(x) = (f_k(x) \mid k = 1, ..., m),$
bezüglich der Constraints:	$g(x) = (g_l(x) \mid l = 1, ..., p) \leq 0,$
mit	$x = (x_j \mid j = 1, ..., n) \in X,$
	$y = (y_k \mid j = 1, ..., m) \in Y.$ (162)

Die Menge von Inputvektoren, welche die Constraints erfüllen, wird als anwendbare Menge (feasible set) X_f bezeichnet:

$$X_f = \{x \in X \mid g(x) \leq 0\}. \quad (163)$$

Die Abbildung der Menge X_f in den Outputraum wird als anwendbare Region Y_f im Zielraum bezeichnet (Zitzler (1999:6[375])):

$$Y_f = \bigcup_{x \in X(f)} \{f(x)\}. \quad (164)$$

2.4.1) Paretokriterium und Paretomenge

Bei einer Ein-Ziel-Optimierung ist die Menge X_f vollständig bezüglich der verwendeten Funktion f(x) geordnet, d.h. für alle Elemente $x_i, x_j \in X_f$ gilt $f(x_i) \geq f(x_j)$ oder $f(x_i) \leq f(x_j)$. Bei einer Mehr-Ziel-Optimierung sind die Elemente aus X_f jedoch nur partiell geordnet, wodurch man zu dem Begriff der Pareto-Dominanz gelangt (Pareto (1896[254]), nach Zitzler (1999[375])). Ein Element x_i dominiert x_j im Kontext einer Minimierung aller Funktionen $f_k(x)$, wenn keine Bewertung $f_{k(1)}(x_i)$ existiert, die größer ist als die korrespondierende Bewertung $f_{k(1)}(x_j)$, und wenn mindestens eine Bewertung $f_{k(2)}(x_i)$ existiert, die kleiner ist als die korrespondierende Bewertung $f_{k(2)}(x_j)$:

$$x_i, x_j \in X_f: x_i \text{ dominiert } x_j:$$
$$\neg\exists\, f_{k(1)}(x): f_{k(1)}(x_i) > f_{k(1)}(x_j) \wedge \exists_{\geq 1}\, f_{k(2)}(x): f_{k(2)}(x_i) : < f_{k(2)}(x_j). \quad (165)$$

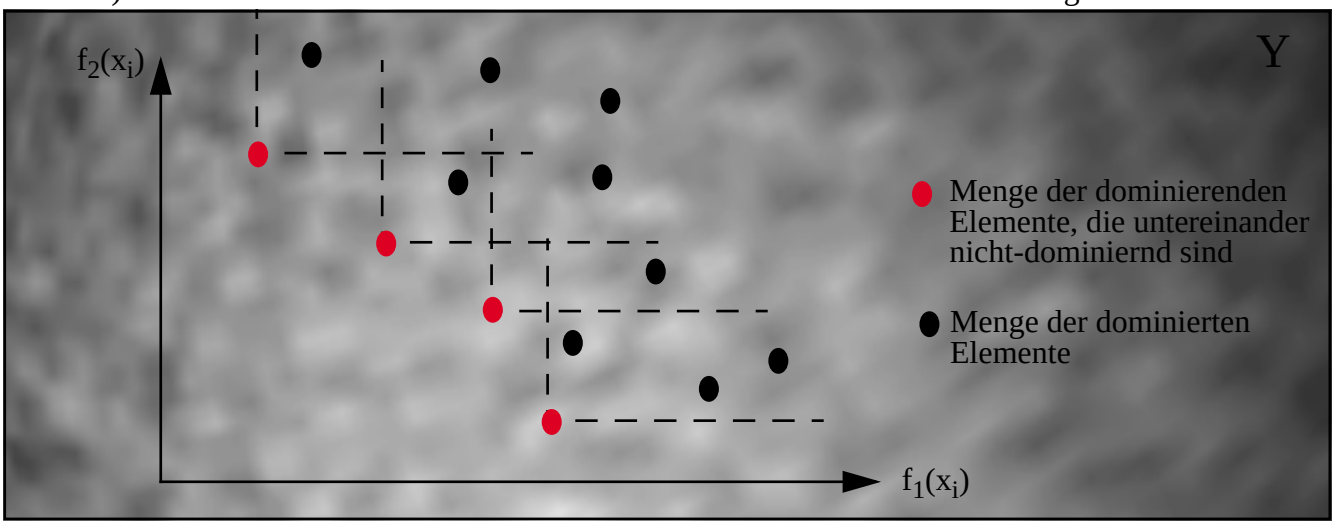
Zwei Elemente $x_i, x_j \in X_f$ werden im Kontext einer Minimierung als nicht-dominand bezeichnet, wenn Bewertungen $f_{k(1)}(x_i)$ existieren, die größer ist als die korrespondierende Bewertung $f_{k(1)}(x_j)$, und wenn Bewertungen $f_{k(2)}(x_i)$ existiert, die kleiner sind als die korrespondierenden Bewertungen $f_{k(2)}(x_j)$ (siehe Abb. 14)):

$$x_i, x_j \in X_f: x_i \text{ ist nicht-dominant bezüglich } x_j:$$
$$\exists_{\geq 1}\, f_{k(1)}(x): f_{k(1)}(x_i) > f_{k(1)}(x_j) \wedge \exists_{\geq 1}\, f_{k(2)}(x): f_{k(2)}(x_i) : < f_{k(2)}(x_j). \quad (166)$$

Es kann eine drei-wertige Dominanz-Funktion eingeführt werden, welche die möglichen Dominanzrelationen zweier Elemente aus x_i, x_j aus X_f beschreibt:

$$\text{dom}(x_i, x_j) = \begin{cases} 1, & \text{wenn } x_j \text{ von } x_i \text{ dominiert wird.} \\ 0, & \text{wenn } x_i \text{ und } x_j \text{ nicht-dominiert sind.} \\ -1, & \text{wenn } x_i \text{ von } x_j \text{ dominiert wird.} \end{cases} \quad (167)$$

Abb. 14) Dominierende und dominierte Elemente bei einer Zwei-Ziel-Minimierung



Die Stellung der beiden Elemente in der Funktion dom(.,.) ist nicht vertauschungsinvariant, wobei gilt:

$$dom(x_i, x_j) = - dom(x_j, x_i). \tag{168}$$

Weiterhin kann eine n-komponentige Dominanzliste $DL(x_i, x_j)$ eingeführt werden, in der die Ergebnisse der Einzel-Vergleiche zwischen korrespondierenden Komponenten der Bewertungsvektoren $f_k(x_i)$ und $f_k(x_j)$ zusammengefasst werden:

$$DL(x_i, x_j) = (d_1, ..., d_k, ..., d_m), \tag{169}$$

mit

$$d_k = \begin{cases} 1, & \text{wenn } f_k(x_i^t) < f_k(x_j^t) \\ 0, & \text{wenn } f_k(x_i^t) = f_k(x_j^t), \\ -1, & \text{wenn } f_k(x_i^t) > f_k(x_j^t). \end{cases} \tag{170}$$

Mit Hilfe der Komponenten der Dominanzliste können die drei alternativen Werte der Dominanz-Funktion reformuliert werden:

$$dom(x_i, x_j) = \begin{cases} 1, & \text{wenn } [\forall d_k: d_k = 1] \vee [\forall d_k: d_k = 0 \wedge \exists_{\geq 1} d_k: d_k = 1]. \\ 0, & \text{wenn } \exists_{\geq 1} d_k = 1 \wedge \exists_{\geq 1} d_k = -1. \\ -1, & \text{wenn } [\forall d_k: d_k = -1] \vee [\forall d_k: d_k = 0 \wedge \exists_{\geq 1} d_k: d_k = -1]. \end{cases} \tag{171}$$

Anstatt zunächst die gesamte Dominanzliste zu erstellen und die Festlegung des Wertes von $dom(x_i, x_j)$ nach dem Abschluss der Einzel-Vergleiche durchzuführen, ist durch die Dominanz-Definition mit Hilfe der Existenz-Quantoren die Möglichkeit offen, die Prüfung zu einem potentiell früheren Zeitpunkt abzubrechen. Die Nicht-Dominanz kann festgelegt werden, wenn mindestens ein $d_k = 1$ und mindestens ein $d_k = -1$ ist, d.h. frühestens nach zwei Einzelvergleichen. Demgegenüber kann die Dominanz bzw. Unterlegenheit nur bei einem Vergleich aller m Komponenten entschieden werden, da der Fall eintreten kann, dass bis auf die letzte Komponente sich eine Dominanz bzw. Unterlegenheit ergibt, und der letzte Vergleich ergibt $d_m = -1$ bzw. $d_m = 1$, sodass die beiden Individuen doch als nicht-dominant einzustufen sind.

Ein Element $x_i \in X_f$ kann bezüglich einer Teilmenge $A \subseteq X_f$ nicht-dominant sein, wenn kein Element $a \in A$ x_i dominiert:

$$\neg\exists\, a \in A\text{: } a \text{ dominiert } x_i. \tag{172}$$

Als Pareto-optimal wird ein Element $x_i \in X_f$ bezeichnet, wenn in der gesamten anwendbaren Menge X_f kein Element existiert, das x_i dominiert:

$$\neg\exists\, a \in X_f\text{: } a \text{ dominiert } x_i. \tag{173}$$

Die Menge aller Pareto-optimalen Elemente aus einer Menge A werden in der Menge der nicht-dominante Entscheidungsvektoren zusammengefasst:

$$PM(A) = \{a \in A \mid a \text{ ist nicht-dominant bezüglich } A\}. \tag{174}$$

Die Menge der nicht-dominante Entscheidungsvektoren bezüglich der gesamten anwendbaren Menge X_f wird als Pareto-Menge $PM(X_f) = X_{PM}$ bezeichnet. Die korrespondierenden Vektoren aus Y werden als pareto-optimale Front bzw. Oberfläche $Y_{PM} \subset Y_f$ bezeichnet.

Weiterhin kann die Unterscheidung zwischen globaler und lokaler Pareto-optimaler Menge eingeführt werden (Deb (1998[93])). Eine Menge $A \subseteq X_f$ wird als lokal Pareto-optimale Menge bezüglich den Metriken $d_X(.,.)$ und $d_Y(.,.)$ sowie den Parametern ε_X, $\varepsilon_Y > 0$ bezeichnet, wenn gilt:

$$\forall a \in A\text{: } \neg\exists\, x \in X_f\text{: } x \text{ dominiert } a \wedge d_X(x, a) < \varepsilon_X \wedge d_Y(f(x), f(a)) < \varepsilon_Y. \tag{175}$$

Eine Menge $A \subseteq X_f$ wird als global Pareto-optimale Menge bezeichnet, wenn gilt:

$$\forall a \in A\text{: } \neg\exists\, x \in X_f\text{: } x \text{ dominiert } a. \tag{176}$$

Eine global Pareto-optimale Menge muss nicht alle existierenden Pareto-optimalen Lösungen enthalten, und alle globalen Pareto-optimalen Mengen sind gleichzeitig lokal Pareto-optimal.

2.4.2) Pareto-Hierarchien

Liegt eine Menge A von Vektoren aus der anwendbaren Menge X_f vor, so kann als eine einfache Form der Strukturierung die Pareto-optimale Menge PM(A) gebildet werden, sodass die binäre Strukturierung von A in PM(A) und $A \setminus PM(A)$ entsteht. Im weiteren sollen Verfahren vorgestellt werden, mit der eine Vektorenmenge $A \subseteq X_f$ mit Hilfe des Konzeptes der Pareto-Dominanz komplexer strukturiert werden kann, wobei eine hierarchische Struktur entstehen soll. Diese wird als Liste von Vektorenteilmengen repräsentiert, wobei die erste Liste die oberste Hierarchiestrufe entspricht. Andere Formen der Strukturierung, wie die Bildung einer flachen Vektorenclusterung, sollen hier nicht betrachtet werden.

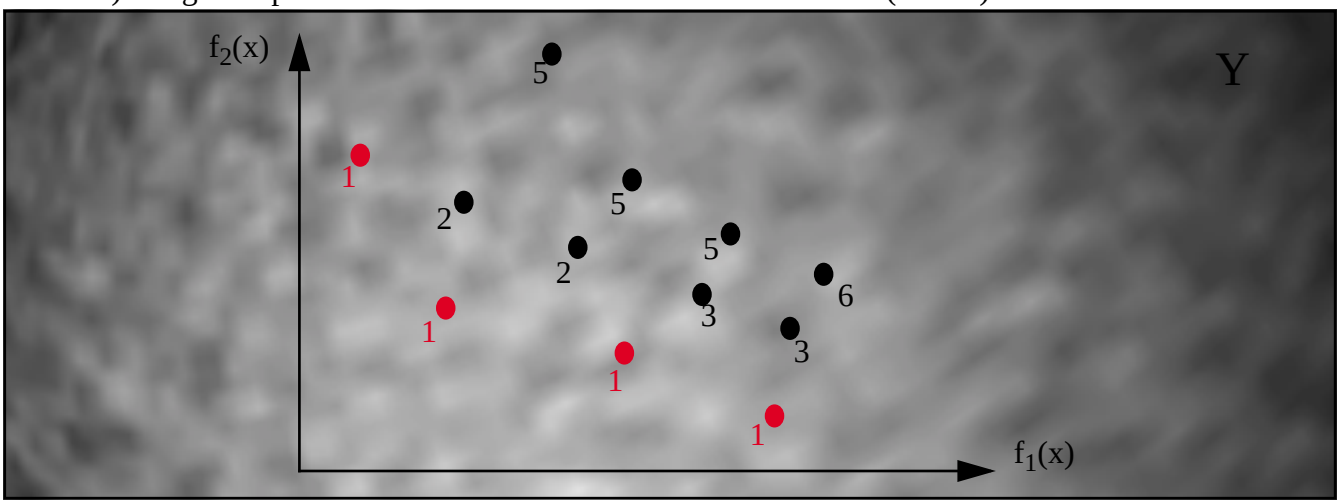
2.4.2.1) Dominanz-Ranking

In Fonseca & Fleming (1993[115]) wird ein Genetischer Algorithmus (GA) für die Mehr-Ziel-Optimierung vorgestellt (siehe Abschnitt 2.5)) für Details zu Evolutionären Algorithmen; siehe auch Schwefel (1977[305], 1981[306], 1995[307]), Goldberg (1989[146]), Nissen (1994[237]), Rechenberg

(1994[277]), Jacob (1997[174]), Bachelier (1998b[16])), der mit einem Ranking der Individuen arbeitet. Im weiteren sollen sich die Darstellung allgemein auf Vektoren und Teilmengen aus X_f beziehen. Der Rang $r(x_i \mid A)$ eines Vektors x_i aus einer endlichen Vektorenmenge $A \subseteq X_f$ berechnet sich durch die Anzahl p_i der Vektoren aus A, die x_i dominieren, wobei in Abb. 15) der Fall einer Zwei-Ziel-Minimierung dargestellt wird:

$$r(x_i \mid A) = 1 + p_i. \tag{177}$$

Abb. 15) Ränge entsprechen der Anzahl der dominierten Individuen (+ Eins)



Die Menge A von Vektoren wird auf diese Weise vollständig hierarchisch strukturiert, indem Rangmengen $PRM(A)_k$ erzeugt werden, wobei $PRM(A)_{k=1}$ der Menge der nicht-dominierten Vektoren entspricht, d.h. $PRM(A)_{k=1} = PM(A)$. Die Liste dieser disjunkten Rangmengen bildet die Pareto-Hierarchie PH(A):

$$PH(A) = (PRM(A)_k \mid k = 1, ...), \text{ mit}$$
$$PRM(A)_k \cap PRM(A)_p = \emptyset, \bigcup_k PRM(A)_k = A. \tag{178}$$

In Fonseca & Fleming (1993[115]) wird den einzelnen Rängen eine Selektionswahrscheinlichkeit zugeordnet, die von der Position des Ranges in der Rangordnung, jedoch nicht von der Anzahl der Individuen innerhalb eines Ranges abhängt. Wie bei einem GA üblich, wird die Population P^t strukturiert, um diese Struktur für die Selektion zur Reproduktion sel_R zu nutzen, wohingegen die gleiche Form der Strukturierung bei einer ES genutzt werden kann, um die Zwischenpopulation ZP^t zu strukturieren, um damit die Selektion sel_N zur Übernahme in die Nachfolgepopulation durchzuführen.

Das Problem dieser Form der Strukturierung durch die Anzahl der dominierten Individuen besteht darin, dass Ränge bzw. Hierarchieebenen existieren, die mit keinem Individuum belegt sind, wie der fehlende Rang „4“ in Abb. 15), d.h. es können Mengen innerhalb der Hierarchie PH(A) existieren, die leer sind.

2.4.2.2) Sukzessive Deaktivierung von Paretomengen

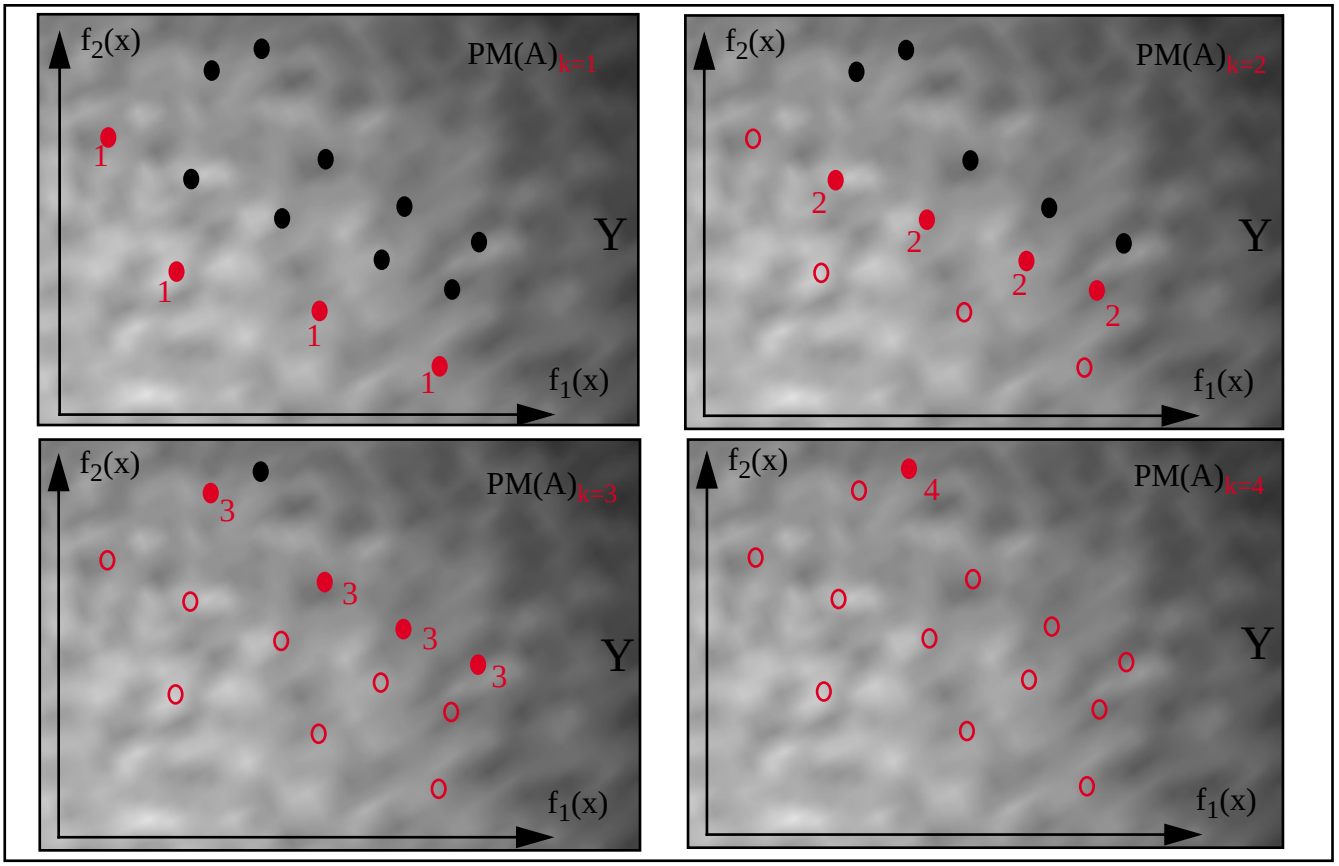
Soll jede Rangmenge einer Hierarchie PH(A) mit mindestens einem Vektor $x \in A$ besetzt werden, dann muss eine alternative Rang-Definition verwendet werden. Im einfachsten Fall werden leere Rangmengen

entfernt und die rechts davon liegenden Rangmengen erhalten einen um Eins kleineren Index k. Auf diese Weise werden jedoch Informationen vernichtet, da aus der sich ergebenden Hierarchie keine Rückschlüsse von Rang und Anzahl der dominierten Vektoren mehr durchführbar ist.

Eine Alternative ergibt sich, wenn die Menge der nicht-dominierten Vektoren PM(A) deaktiviert wird, d.h. wenn die Restmenge aus der Gesamtmenge A und der Menge der nicht-dominierten Vektoren PM(A) gebildet wird, wobei auf der Basis der Restmenge A \ PM(A) wiederum die Menge der nicht-dominierten Vektoren gebildet wird. Diese Vorgehensweise kann sukzessiv weiter geführt werden, bis die Restmenge leer wird. $RM(A)_1$ sei die Restmenge der ersten Iteration, d.h. $RM(A)_1 = A \setminus PM(A)$, so wird in der zweiten Iteration $PM(RM(A)_1)$ gebildet. Wird A als Initialisierung gleich $RM(A)_0$ gesetzt, so ergibt sich allgemein für die Restmengenbildung (siehe Abb. 16)):

$$RM(A)_k = RM(A)_{k-1} \setminus PM(RM(A)_k). \tag{179}$$

Abb. 16) Ränge durch sukzessive Deaktivierung von Paretomengen



2.4.2.3) Pareto-Wettkampf-Hierarchien

Die Paretomenge basiert auf dem Konzept der Dominanz, die durch eine Vergleichs-Operation zwischen zwei Vektoren ermittelt wird. Grundlage der Wettkampf-Selektion ist ebenfalls eine Vergleichs-Operation zwischen zwei Individuen, sodass die Kombination beider Konzepte zu einer Pareto-Wettkampf-Hierarchie PWH naheliegend ist.

2.4.2.3.1) Wettkampfoperation

Eine Wettkampf-Hierarchie wird durch die wiederholte Anwendung eines Wettkampf-Verfahrens aufgebaut, das im weiteren dargestellt werden soll. Eine allgemeines Wettkampf-Verfahren baut auf einer Grundmenge A von Vektoren auf. Ein Wettkampf-Verfahren besteht aus der Anwendung von Wettkampfoperationen, bei denen eine Teilmenge von Elementen aus A ausgewählt wird, und ein Wettkampf durchgeführt wird, bei dem eine Anzahl von Individuen aus der Teilmenge als Gewinner spezifiziert werden. Um dieses zu operationalisieren, wird ein Parameter $\zeta \in N$ eingeführt, der die Anzahl der Elemente spezifiziert, die aus A ausgewählt werden, und zwischen denen der Wettkampf ausgetragen wird. Die sich ergebende Teilmenge soll mit W_k bezeichnet werden:

$$W_k = \{x_{k(i)} \in A \mid i = 1, ..., \zeta\}. \tag{180}$$

Wie eine Teilmenge W_k aus A ausgewählt wird, soll durch ein Verfahren spezifiziert werden, das als Selektion zum Wettkampf sel_W bezeichnet werden kann, und das im einfachsten Fall eine gleichverteilte Zufallsauswahl aus A ist. Allgemein wird die Bezeichnung verwendet:

$$sel_W(A) = W_k. \tag{181}$$

Ein zweiter Parameter $\xi \in N$ mit $\xi < \zeta$ spezifiziert die Anzahl der Elemente, die bei einem Wettkampfgeschehen als Sieger hervorgehen dürfen. Ein (ζ=2, ξ=1)-Wettkampf ist somit die einfachste Form, bei der zwei Individuen aus einer allgemeinen Grundmenge A ausgewählt werden, einem Wettkampf auf der Basis ihrer Bewertungen unterliegen, bei dem genau ein Individuum als Sieger hervorgeht. Die Wettkampf-Hierarchie im Kontext eines EA, die in Angeline & Pollack (1993[12]) dargestellt wird, basiert auf diesen beiden Parameterwerten.

Der eigentliche Wettkampf wird zwischen den Elementen einer Menge W_k ausgetragen. Hierzu muss eine Dominanzfunktion auf der Basis der Vektoren x definiert werden, die W_k als Input verwendet, und eine Siegermenge als Output erzeugt, die mit WS_k bezeichnet werden soll:

$$dom(W_k \mid x) = WS_k = \{x_{k(i)} \in W_k \mid i = 1, ..., \xi\}. \tag{182}$$

Die Funktion $dom(W_k \mid x)$ kann als binäre Klassifikationsfunktion betrachtet werden, die W_k in zwei disjunkte Teilmengen zerlegt, wobei die Teilmenge WS_k die ξ Sieger-Elemente enthält, und WV_k die ζ - ξ Verlierer-Elemente:

$$W_k = WS_k \cup WV_k,$$
$$WS_k \cap WV_k = \varnothing. \tag{183}$$

Nachdem WS_k und WV_k festgelegt wurden, muss geklärt werden, was mit den Gewinner- und Verlierer-Individuen geschehen soll. Dies soll durch die Operationen sel_S und sel_V geschehen, wobei Elemente wieder in A zurückgelegt werden können, oder aus den weiteren Wettkämpfen ausgeschlossen werden können. Beispielsweise kann sel_V das vollständige oder teilweise Zurücklegen der Verlierer regeln, mit der Folge, dass sie eine weitere Chance erhalten. Werden die Verlierer von weiteren Auswahloperationen ausgeschlossen, so operationalisiert sel_V zusammen mit sel_W eine Auswahl ohne Zurücklegen. Es sind auch sel_S-Operationen definierbar, welche die Gewinner wieder in A zurückführen, wobei ein Gewinnerzähler die Anzahl der Zuordnungen zu der Gewinnermenge aufsummiert. Werden die Gewin-

ner und Verlierer einer Wettkampfoperation nicht zurückgelegt, sondern in zwei separate Mengen eingefügt, in die alle Elemente aus A durch eine Sequenz von Wettkampfoperation eingefügt werden, so wird dadurch eine einfache Zerlegung von A in eine Gewinner- und eine Verliererklasse durchgeführt.

Zusätzlich zu den dargestellten Operationen muss eine Abbruchbedingung spezifiziert werden, da bestimmte Wettkampf-Verfahren auch bei einer endlichen Grundmenge Abeliebig lange durchgeführt werden können, wenn sel_S und sel_V Zurücklegekomponenten besitzen. Eine natürliche Abbruchbedingung ist beispielsweise eine Auszehrung von A durch sel_W, indem durch ein fehlendes Zurücklegen bei jeder sel_W-Operation A kleiner wird, bis es weniger als ζ Elemente enthält, sodass keine vollständige Teilmenge W_k mehr gebildet werden kann, wobei geregelt werden muss, ob die verbleibenden Individuen in A zur Gewinner- bzw. Verliermenge gezählt werden sollen. Wird die Abbruchbedingung über die verbleibende Anzahl in A definiert, so ist sie abhängig von der Selektion zum Wettkampf, sodass die Abbruchbedingung als Funktion $end(sel_W)$ definiert wird.

Zusammenfassend kann ein Wettkampf-Verfahren durch einen 8-Tupel beschrieben werden:

$$W = (A, \zeta, sel_W, \xi, dom(W_k \mid x), sel_S, sel_V, end(sel_W)). \tag{184}$$

2.4.2.3.2) Wettkampf-Hierarchie

Eine Wettkampf-Hierarchie ist eine spezielle Form des Wettkampfes, bei der durch die Operationen sel_S und sel_V eine Hierarchie aus A aufgebaut wird, anstatt eine binäre Klassifikation in Siger und Verlierer durchzuführen. Die Selektionsoperation sel_W greift dabei im Verfahrensverlauf nicht mehr auf A sondern auf Individuen einer Hierarchieebene zu.

Wird eine Wettkampf-Hierarchie durch ein (ζ=2, ξ=1)-Wettkampf-Verfahren erzeugt, wie dies bei Angeline & Pollack (1993[12]) beschrieben wird, so wird zunächst die Gesamtmenge A als unterste Hierarchieebene betrachtet, aus der bei jeder Wettkampfoperation zwei Elemente durch sel_W gleichverteilt zufällig ohne Zurücklegen ausgewählt werden, d.h.

$$sel_W(A) = W_k = \{x_i, x_j\};\ A(neu) = A(alt) \setminus W_k. \tag{185}$$

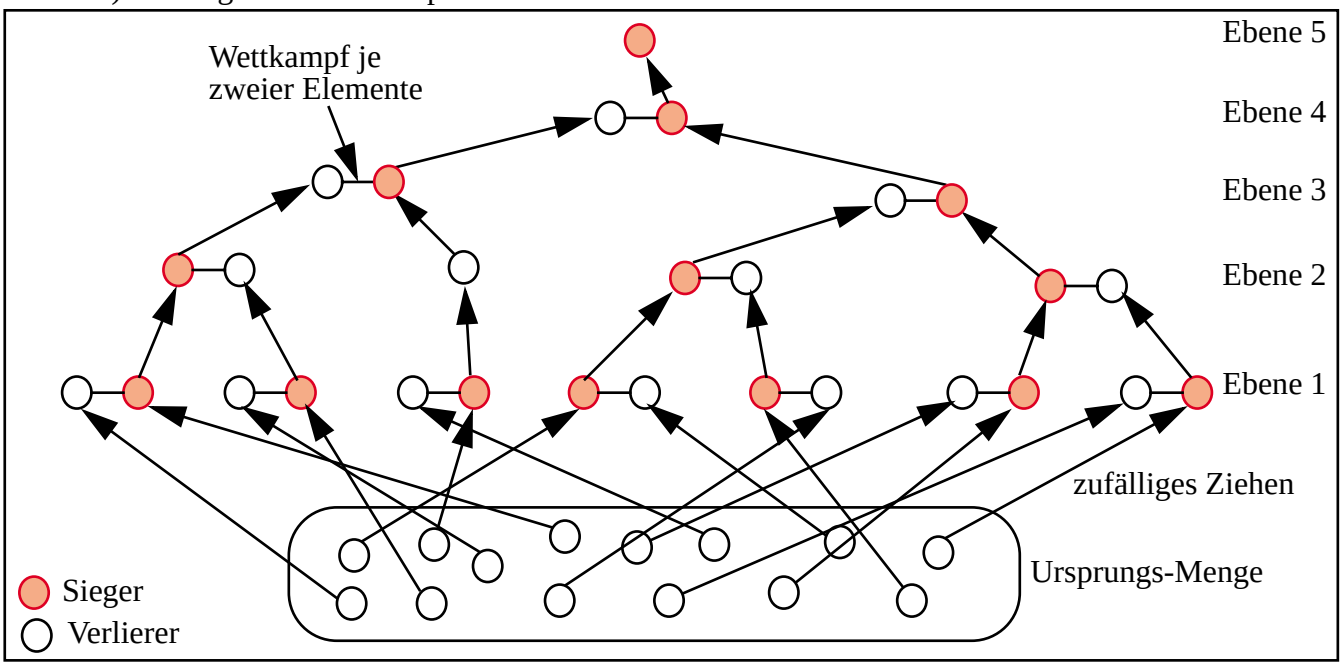
Als Dominanzfunktion wird der Vergleich der beiden Funktionswerte $f(x_i)$ und $f(x_j)$ durchgeführt. Bei einer Minimierung liefert die Dominanzfunktion die Siegermenge von genau einem Vektor, wobei gilt, dass zufällig mit einer Wahrscheinlichkeit von 1/2 eines der beiden Individuen ausgewählt wird, wenn beide Funktionswerte gleich sein sollten:

$$dom(W_k \mid x) = WS_k = \begin{cases} \{x_i, & \text{wenn } f(x_i) < f(x_j), \\ \{x_j, & \text{wenn } f(x_i) > f(x_j), \\ \{p(x_i) = 1/2, & \text{wenn } f(x_i) = f(x_j). \end{cases} \tag{186}$$

Die Auflösung eines Unentschieden dient zur strikten Erzeugung einer Hierarchie, in der eine Ebene k genau halb so viele Elemente enthalten soll, wie die darunterliegende Ebene k-1 (siehe Abb. 17)). Das

gleiche gilt für eine Grundmenge A, die aus einer ungeraden Anzahl von Individuen besteht, d.h. ein verbleibender Vektor wird automatisch auf die nächste Hierarchieebene transferiert.

Abb. 17) Bildung einer Wettkampf-Hierarchie



Besitzt die Ursprungsmenge A π Elemente, so wird beim ersten Durchgang $\pi/2$ Wettkämpfe durchgeführt, wobei die $\pi/2$ Verlierer die Wettkampf-Hierarchieebene WH(1 | P) bilden, d.h.

$$WH(1 \mid P) = \cup_{k=1->\pi/2} WV_k. \tag{187}$$

Die Gewinner werden auf die nächst höhere Ebene transferiert, auf der im nächsten Durchgang $\pi/4$ Wettkämpfe durchgeführt werden. Während der Bildung aller Hierachieebenen werden somit c Vergleiche durchgeführt (Angeline & Pollack (1993: 266[12])), mit

$$c = \sum_{i=1->\log(\pi)} \pi/(2^i) = \pi - 1. \tag{188}$$

Bei der Erzeugung einer Wettkampf-Hierarchie aus π Elementen ergibt sich eine Anzahl von ε Ebenen

$$\varepsilon = \text{IntegerPart}[ld(\pi)] + 2, \tag{189}$$

mit IntegerPart[.] als der Funktion, die den ganzzahligen Anteil einer reellen Zahl liefert.

2.4.2.3.3) Pareto-Wettkampf-Hierarchie als Spezialisierung einer Wettkampf-Hierarchie

Die Pareto-Wettkampf-Hierarchie ist eine Spezialisierung der Wettkampf-Hierarchie, bei der die Dominanzfunktion nicht auf einem Bewertungsskalar, sondern auf einem Bewertungsvektor basiert, d.h. es werden die beiden Outputvektoren $f(x_i)$, $f(x_j) \in Y_f \subseteq R^m$ entsprechend dem Paretokriterium verglichen.

Beginnend mit einer Kopie der Gesamtmenge A werden bei einem (ζ=2, ξ=1)-Wettkampf-Verfahren zwei Vektoren x_i und x_j gleichverteilt zufällig ohne Zurücklegen aus A gezogen. Der Wettkampf besteht

im Vergleich der beiden Outputvektoren $f(x_i)$, $f(x_j) \in Y_f \subseteq R^m$ entsprechend dem Kriterium der Pareto-Dominanz. Das dominierende Element wird in die Ebene $PWH(A)_2$ verschoben, während der Verlierer in die Ebene $PWH(A)_1$ verschoben wird. Tritt der Fall ein, dass die zwei Vektoren nicht-dominand sind, so ist dies mit $f(x_i) = f(x_j)$ im Fall einer Ein-Ziel-Bewertung vergleichbar, mit der Folge, dass ein Vektor zufällig ausgewählt wird und in die höhere Ebene $PWH(A)_2$ verschoben wird, während der verbleibende Vektor nach $PWH(A)_1$ verschoben wird. Diese Zufallsoperation wird verwendet, um die Hierarchie-Struktur strikt aufzubauen, nach der eine Ebene k genau halb so viele Elemente besitzen darf, wie die darunter liegende Ebene k-1.

Es folgt die nächste Auswahl zweier Elemente aus der Restmenge $A(neu) = A(alt) \setminus \{x_i, x_j\}$, bis die Restmenge leer ist, oder genau ein Element enthält, das direkt nach $PWH(A)_2$ verschoben wird. Danach sind die Operationen bezüglich der Ebene $PWH(A)_1$ beendet, und die Operationen bezüglich des Aufbaus von $PWH(A)_3$ werden begonnen. Als Grundmenge dient nun $PWH(A)_2$, wobei eine Zwischenmenge $PWH(A)_2' = \emptyset$ eingeführt wird. Aus $PWH(A)_2$ werden nun wiederum jeweils zwei Vektoren gleichverteilt zufällig ohne Zurücklegen gezogen, gefolgt von der Wettkampfoperation, dem Verschieben des Gewinners nach $PWH(A)_3$ und des Verlieres nach $PWH(A)_2'$, bis $PWH(A)_2$ leer wird. Die Zwischenmenge $PWH(A)_2'$ wird danach als reguläre Hierarchieebene $PWH(A)_2$ verwendet, wodurch die Operationen bezüglich der zweit untersten Ebene abgeschlossen sind.

Auf diese Weise werden iterativ aus A eine Pareto-Wettkampf-Hierarchie PWH(A) mit IntegerPart[ld(π)] + 2 Ebenen erzeugt, die untereinander disjunkt sind:

$$PWH(A) = (PWH(A)_k \mid k = 1, ..., IntegerPart[ld(\pi)] + 2), \text{ mit}$$
$$PWH(A)_k \cap PWH(A)_p = \emptyset, \bigcup_k PWH(A)_k = A. \qquad (190)$$

2.4.3) Abbruchkriterium bei Mehr-Ziel-Optimierung

Das grundlegende Ziel einer Mehr-Ziel-Optimierung besteht in der Ermittlung einer Menge nicht-dominanter Objektvektoren, die im Outputraum von der Paretofront einen möglichst geringen Abstand besitzen. Weiterhin sollen sie die Paretofront hinreichend gut repräsentieren, d.h. sie sollen nicht auf einen lokalen Teil der Front sondern über die gesamte Front verteilt sein. Ein schwieriges Problem bei der Mehr-Ziel-Optimierung ist die Formulierung eines Abbruchkriteriums, da die Paretomenge selbst unbekannt ist, und die iterationsübergreifende Menge der nicht-dominanten Objektvektoren von Iteration zu Iteration starken Fluktuationen ausgesetzt sein kann. Im Extremfall kann in einer neuen Iteration ein Objektvektor gefunden werden, der einen Großteil oder alle Elemente in der iterationsübergreifenden Menge dominiert, sodass alle diese Elemente gelöscht werden müssen.

In Zitzler (1999:43ff[375]) werden Maße definiert, welche die Qualität einer Menge nicht-dominanter Elemente beschreiben bzw. die zwei Mengen vergleichen. Das Ziel dieser Darstellungen ist der Vergleich unterschiedlicher Verfahren der Mehr-Ziel-Optimierung, indem die Ergebnismengen nicht-dominanter Elemente verglichen werden. Da bei jeder Iteration eines Mehr-Ziel-Optimierungsverfahrens eine Ergebnismenge erzeugt wird, können die Mengen aufeinander folgender Iterationen bzw. die Mengen der Ite-

rationssequenz verglichen werden, mit dem Ziel, daraus ein Abbruchkriterium abzuleiten. Zitzler (1999:33[375]) selbst spezifiziert bei seinem evolutionären Mehr-Ziel-Optimierungsverfahren „Strength Pareto Evolutionary Algorithm (SPEA)“ ein Abbruchkriterium nur unpräzise, wobei er im einfachsten Fall von einer maximalen Iterationsanzahl als externem Parameter ausgeht, d.h. er verwendet seine eigenen Performance-Maße nicht zur Definition eines Abbruchkriteriums.

Unterschieden wird zwischen skalierungs-invarianten und skalierungs-varianten Maßen (Zitzler (1999:43ff[375]), wobei erstere Maßse unabhängig von der verwendeten Distanzmetrik $d_X(.,.)$ im Objektvariablenraum $X \subseteq R^n$ und und $d_Y(.,.)$ Outputraum $Y \subseteq R^m$ sind.

Das erste invariante Maß beschreibt die Größe des dominierten Raumes bezogen auf eine einzelne Menge $A = \{x_i \mid i = 1, ..., \mu\} \subseteq X_f$ nicht-dominanter Objektvektoren, wobei dieses Maß jedoch nicht zum Vergleich zweier Mengen A, $B \subseteq X_f$ herangezogen werden kann, sondern ein ergänzendes Maß ist. Gegegen ist ein in allen m Dimensionen beschränkter Outputraum, d.h. bei einer Minimierung existiert für jede Dimension ein maximaler Outputwert, die in dem Vektor $f_{max} = (f_{1,max}, ..., f_{m,max})$ zusammengefasst werden. Die erlaubte Region Y_f wird angenommen als der Hyperkubus bzw. das Rechteck bei m = 2 dessen Diagonale durch den Urspung und den Punkt f_{max} gegeben ist

$$Y_f = [0, f_{k,max}]^m. \tag{191}$$

Der dominierte Raumes $Y_{f,A}$ einer Menge A wird definiert als Vereinigungsmenge der dominierten Räume aller Elemente aus A (Zitzler (1999:43[375])). Betrachtet wird ein Element x_i von A, dem im Outputraumes Y_f ein Punkt $f(x_i) \in Y_f$ zugeordnet wird. Der von $f(x_i)$ dominierte Raum $Y_{f,x(i)}$ ist definiert durch den Hyperkubus bzw. das Rechteck bei m = 2 dessen Diagonale durch die beiden Punkte $f(x_i)$ und f_{max} beschrieben wird (siehe Abb. 18)):

$$Y_{f,x(i)} = [f_k(x_i), f_{k,max}]^m. \tag{192}$$

Für den dominierten Raumes $Y_{f,A}$ einer Menge A gilt somit (siehe Abb. 19)):

$$Y_{f,A} = \bigcup_{i=1->\mu} Y_{f,x(i)}. \tag{193}$$

Das zweite invariante Maß beschreibt die Überdeckung zweier Mengen $A = \{x_{A,i} \mid i = 1, ..., \mu_A\}$ und $B = \{x_{B,i} \mid i = 1, ..., \mu_B\}$, wobei die Elemente einer Menge untereinander nicht-dominant sind (Zitzler (1999:43f[375])). Die Funktion c(A,B) bildet die geordnete Liste (A,B) auf einen Wert im Intervall [0, 1] ab, indem die relative Anzahl der Elemente aus B ermittelt wird, die von Elementen aus A schwach dominiert werden:

$$c(A,B) = \#\{x_{B,i} \in B \mid \exists x_{A,j} \in A: x_{A,j} \text{ dominiert } x_{B,i} \text{ oder } x_{A,j} \text{ ist nicht-dominiert zu } x_{B,i}\}/\mu_B. \tag{194}$$

Abb. 18) Dominierter Raum eines nicht-dominanten Punktes im Outputraum

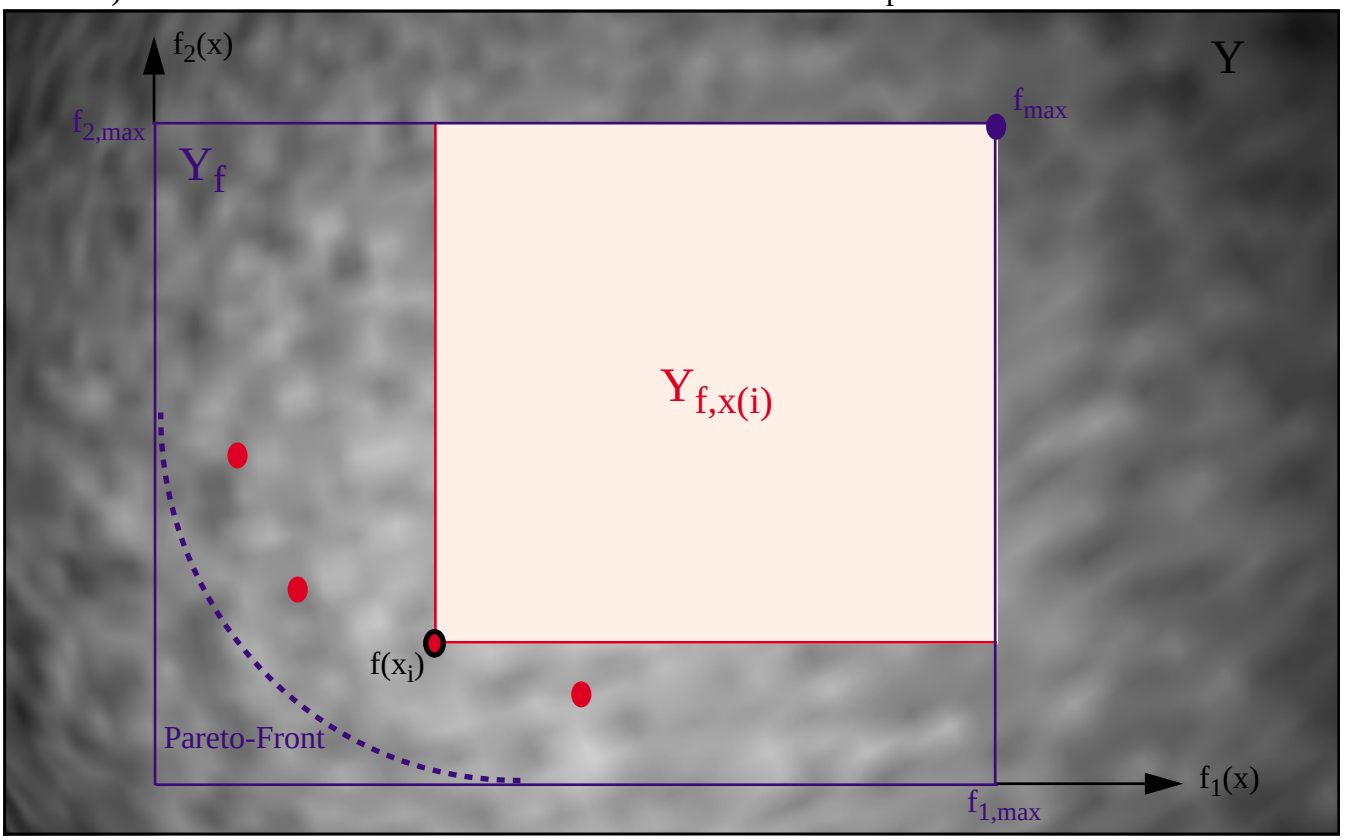
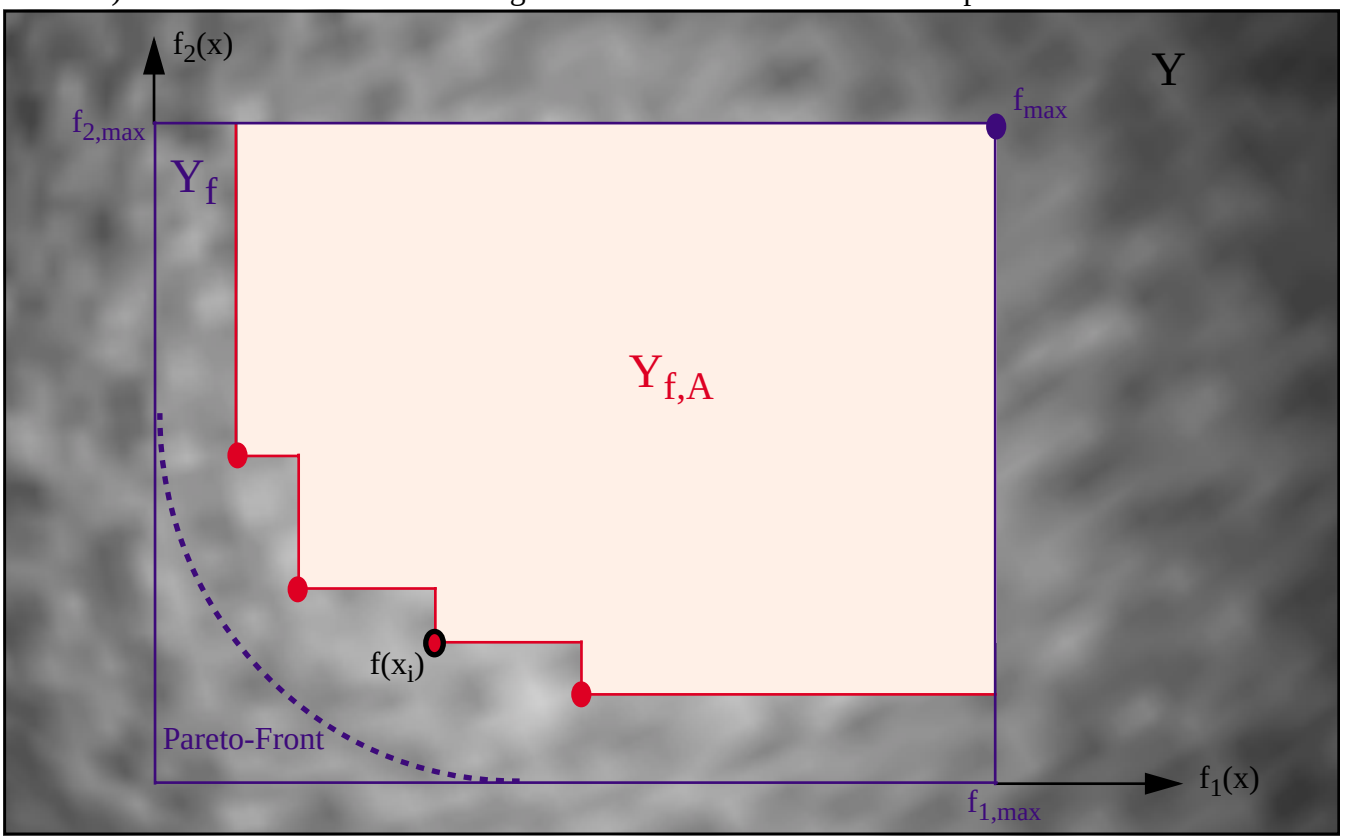


Abb. 19) Dominierter Raum einer Menge nicht-dominanter Punkte im Outputraum



Der Wert c(A,B) = 1 bedeutet, dass alle Elemente aus B durch Elemente aus A schwach dominiert werden. Das Maß c(.,.) ist vertauschungs-variant, d.h. c(A,B) ≠ 1 - c(B,A), sodass beide Richtungen c(A,B) und c(B,A) betrachtet werden müssen.

Ein potentielles Problem mit dem Maß c(A,B) ergibt sich durch die Eigenschaft der Skalierungs-Unabhängigkeit, und dadurch, dass die Dominanz hier als eine binäre Funktion definiert ist. Es kann vorkommen, dass eine Front (Front A) in Y näher an der unbekannten Paretofront im Fall einer Minimierung liegt als eine andere (Front B), und trotzdem werden beide Fronten durch das c-Maß als gleichwertig betrachtet (siehe Abb. 20) und Abb. 21)). Von Zitzler (1999:44f[375]) wird ein Maß auf der Basis des s-Maßes eingeführt, das zwischen Mengen A und B in einer solchen Situation differenzieren kann, wobei dies für den Fall zweier Maximierungen beschrieben wird, sodass zunächst dieser Fall beschrieben werden soll (siehe Abb. 20)). Gegeben sind zwei Mengen A und B aus untereinander nicht-dominierenden Objektvektoren, wobei nun Front B näher an der Pareto-Front liegt als Front A. Unterschieden wird zwischen dem Raumgebiet $Y_{f,A}$, das durch A dominiert wird $Y_{f,A}$, dem Raumgebiet $Y_{f,B}$, das durch B dominiert wird, sowie dem Raumgebiet $Y_{f,A+B}$, das durch A oder B dominiert wird, als der Vereinigungsmenge der beiden voran gegangenen Raumgebiete:

$$Y_{f,A\cup B} = Y_{f,A} \cup Y_{f,B}. \tag{195}$$

Die Region, die durch A und durch B dominiert wird, wird durch die Schnittmenge von $Y_{f,A}$ und $Y_{f,B}$ beschrieben:

$$Y_{f,A\cap B} = Y_{f,A} \cap Y_{f,B}. \tag{196}$$

Weiterhin kann unterschieden werden zwischen der Region $Y_{f,A-B}$, die durch A aber nicht durch B dominiert wird, und der Region $Y_{f,B-A}$, die durch B aber nicht durch A dominiert wird:

$$\begin{aligned} Y_{f,A-B} &= Y_{f,A} \setminus Y_{f,B} = Y_{f,A\cup B} \setminus Y_{f,B}, \\ Y_{f,B-A} &= Y_{f,B} \setminus Y_{f,A} = Y_{f,A\cup B} \setminus Y_{f,A}. \end{aligned} \tag{197}$$

Das von Zitzler (1999:44f[375]) vorgeschlagene Maß beschreibt die Räume $Y_{f,A-B}$ und $Y_{f,B-A}$, wobei in Abb. 20) ersichtlich ist, dass $Y_{f,B-A} > Y_{f,A-B}$ ist, was interpretiert wird, dass im Fall der dargestellten Maximierung die Front B näher an der Pareto-Front liegt als Front A, und somit als besser bewertet wird.

Die korrespondierende Situation bei einer Minimierung wird in Abb. 21) dargestellt, wobei hier Front A näher an der Pareto-Front liegt als Front B, während beide Fronten durch das Maß c(A,B) und c(b,A) als gleichwertig betrachtet werden. Auch hier wird die Region $Y_{f,A-B}$ bestimmt, die durch A jedoch nicht durch B dominiert wird, sowie die Region $Y_{f,B-A}$, die durch B jedoch nicht durch A dominiert wird. Es zeigt sich dass $Y_{f,A-B} > Y_{f,B-A}$ ist, d.h. die Front A wird durch das Maß als besser bewertet als Front B, sodass eine geeignete Unterscheidungsfähigkeit vorliegt.

Abb. 20) Dominierte Raumgebiete bei Maximierung

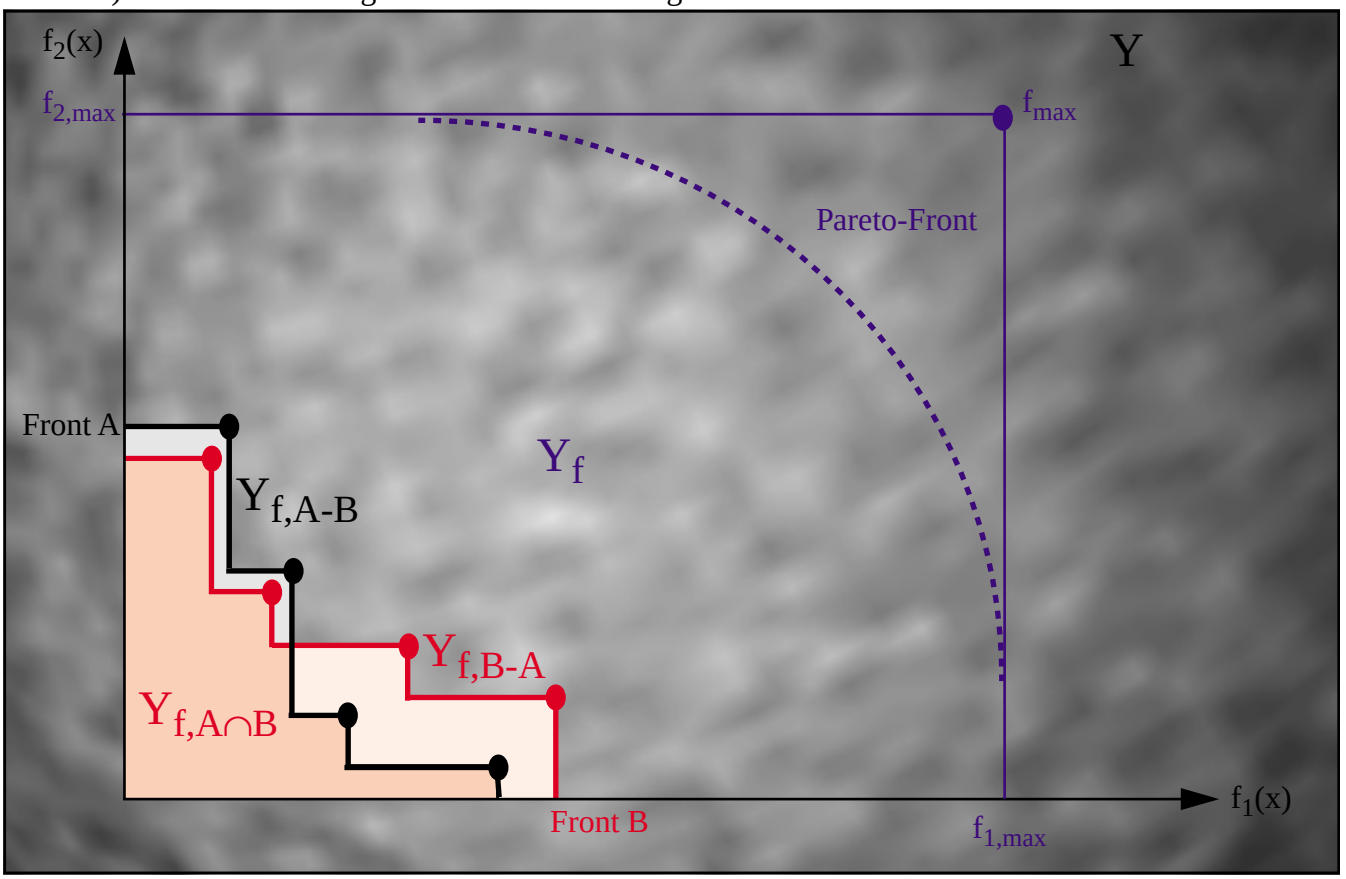
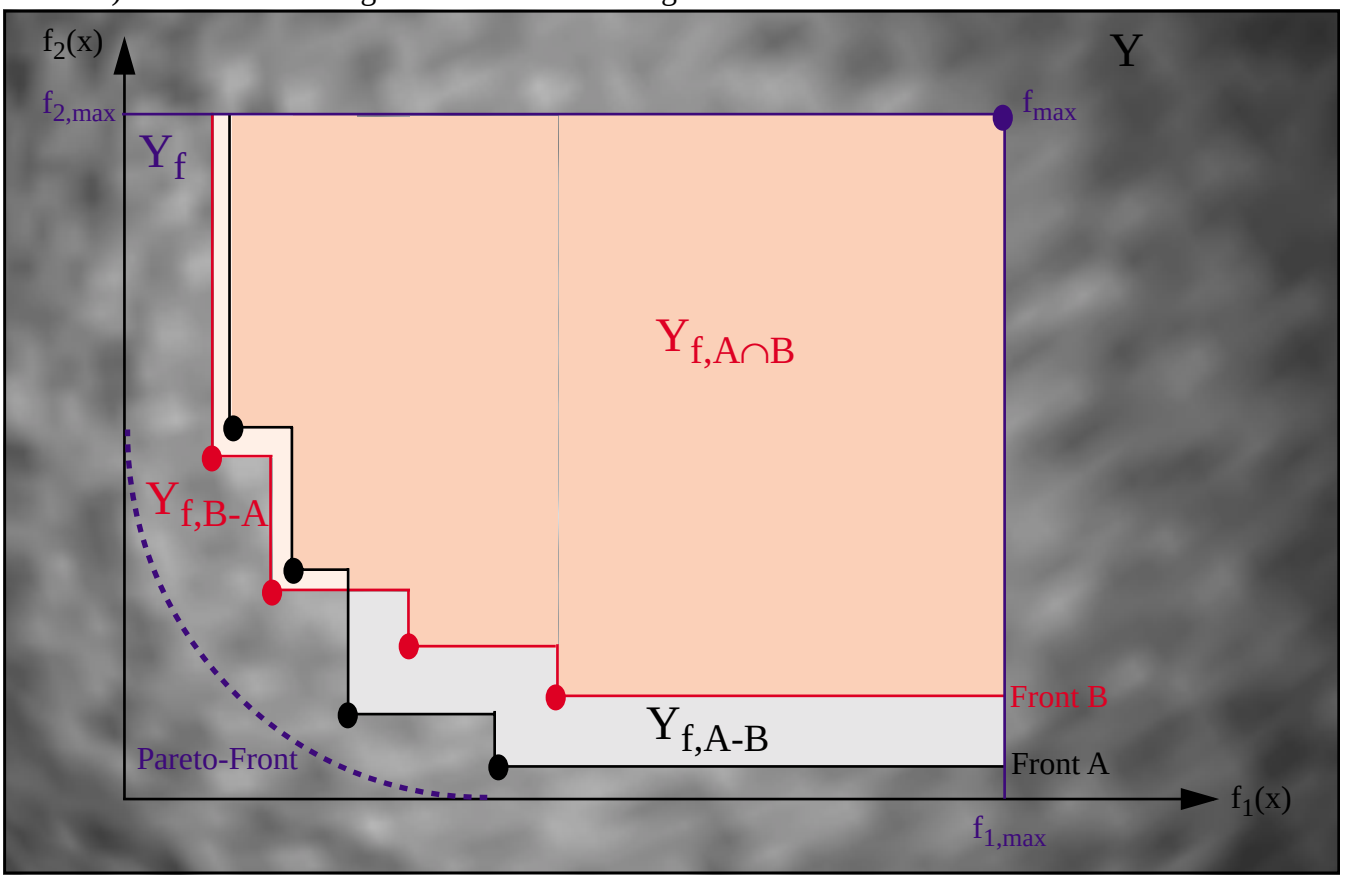


Abb. 21) Dominierte Raumgebiete bei Minimierung



In Zitzler (1999:45[375]) wird die Normierung von Raumgebieten durch das Volumen V_f des möglichen Raumgebietes in Y verwendet:

$$V_f = \prod_{i=1->n}(f_{i,max} - f_{i,min}). \quad (198)$$

Insgesamt werden die folgenden vier Maße bestimmt, um die beiden Mengen A und B zu vergleichen:
1) $Y_{f,A}/V_f$: Relative Größe der Region im Outputraum, der von A schwach dominiert wird.
2) $Y_{f,B}/V_f$: Relative Größe der Region im Outputraum, der von B schwach dominiert wird.
3) $Y_{f,A-B}/V_f$: Relative Größe der Region im Outputraum, der von A und nicht von B schwach dominiert wird.
4) $Y_{f,B-A}/V_f$: Relative Größe der Region im Outputraum, der von B und nicht von A schwach dominiert wird.

Neben den skalierungs-invarianten Maßen werden in Zitzler (1999:45f[375]) skalierungs-variante Maße vorgestellt, die von einer Metrik $d_X(.,.)$ bzw. $d_Y(.,.)$ abhängig sind. Die vorgestellten Maße bewerten ausschlisslich eine einzelne Menge $A = \{x_{A,i} (x_{A,ij} \mid j = 1, ..., n) \mid i = 1, ..., \mu_A\} \subseteq X$ mit untereinander nicht-dominanten Objektvektoren, d.h. Maße, die zwei Mengen involvieren wie $Y_{f,A-B}$ und $Y_{f,B-A}$ werden im Bereich der skalierungs-varianten Maße nicht definiert.

Das erste Maß gibt die durchschnittliche Entfernung der Elemente von A zu der Pareto-optimalen Menge X_P an:

$$M(A)_1 = 1/\mu_A * \sum_i \min\{d_X(x_{A,i}, x) \mid x \in X_P\}. \quad (199)$$

Das zweite Maß verwendet die Verteilung in Kombination mit der Anzahl nicht-dominanter Lösungen indem die Elemente aus A ermittelt werden, die von anderen Elementen aus A eine Distanz größer als ein Schwellenwert δ_X besitzen:

$$M(A)_2 = 1/(\mu_A-1) * \sum_i \#\{x_{A,l} \in A \mid d_X(x_{A,i}, x_{A,l}) > \delta_X\} \in [0, \mu_A]. \quad (200)$$

Dieses Verteilungsmaß gibt die Anzahl der δ_X-Nichen an, wobei ein höherer Wert für einen geeigneten, externen Parameter δ_X besser ist als ein kleinerer Wert. Das Maximum bedeutet, das für kein Element aus A ein anderes Element in einer δ_X-Nachbarschaft existiert.

Das dritte Maß berechnet die Spannweite der Menge A, indem zunächst die maximale Distanz zwischen den Elementen aus A bezüglich jeder der n Dimensionen von X berechnet wird. Diese maximalen Werte werden summiert und es wird die zweite Wurzel daraus gezogen:

$$M(A)_3 = \sqrt{\sum_{j=1->n} \max\{d_X(x_{A,ij}, x_{A,lj}) \mid x_{A,i}, x_{A,l} \in A\}}. \quad (201)$$

In analoger Weise werden drei Maße für den Outputraum ermittelt, mit der Metrik $d_Y(.,.)$ und dem Schwellenwert δ_Y:

$$M(f(A))_1 = 1/\mu_A * \sum_i \min\{d_Y(f(x_{A,i}), y) \mid y \in Y_P\},$$
$$M(f(A))_2 = 1/(\mu_A\text{-}1) * \sum_i \#\{f(x_{A,l}) \in f(A) \mid d_Y(f(x_{A,i}), f(x_{A,l})) > \delta_Y\} \in [0, \mu_A],$$
$$M(f(A))_3 = \sqrt{\sum_{k=1->m}} \max\{d_Y(f(x_{A,ik}), f(x_{A,lk})) \mid f(x_{A,i}), f(x_{A,l}) \in f(A)\}. \quad (202)$$

Das Verteilungsmaß $M(f(A))_2$ gibt die Anzahl der δ_Y-Nichen an, wobei das Maximum bedeutet, das alle Vektoren aus der Outputmenge f(A) eine Mindestdistanz von δ_Y besitzen. D.h. die Vektoren sind über die nicht-dominante Front in Abhängigkeit von dem Parameter δ_Y optimal verteilt.

2.5) Verwendete Modelle der Evolutions-Strategien

2.5.1) Ein-Ziel-Optimierung durch ρ-geschlechtliche (μ,λ)- und (μ+λ)-ES

Evolutions-Strategien sind eine Klasse von Evolutions-Algorithmen, die besonders geeignet sind, reelle Objektvariablen zu optimieren (Schwefel (1977[305], 1981[306], 1995[307]), Goldberg (1989[146]), Rechenberg (1994[277]), Jacob (1997[174]), Bachelier (1998b[16])). In einer Generation t wird eine Population P^t von μ Eltern-Individuen betrachtet, die durch Reproduktionsoperationen eine Menge von λ > μ Nachkommen -Individuen erzeugen. Bei einer (μ,λ)-ES bilden die λ > μ Nachkommen die Zwischenpopulation ZP^t, während bei einer (μ+λ)-ES die μ Eltern- und die λ Nachkommen-Individuen die Zwischenpopulation bilden. D.h. bei einer (μ,λ)-ES besitzt ein Individuum eine Lebenszeit von genau einer Generation, während bei einer (μ+λ)-ES Individuen in Abhängigkeit von ihrer Bewertung durch eine Fitnessfunktion länger existieren können. Der Verfahrensparameter ρ < μ gibt die Anzahl der Eltern an, die bei einer Reproduktionsoperation aus P^t zufällig und mit Zurücklegen gezogen werden, um einen Nachkommen zu erzeugen. In den meisten Fällen wird ρ = 2 verwendet, was den biologischen Standardfall einer zweigeschlechtlichen Reproduktion modelliert, während ρ > 2 in der Biologie als Spezialfall gilt, der nur bei der Reproduktion von Viren eine Rolle spielt.

Eltern-Individuen a_i^t bestehen bei der betrachteten Klasse von ES aus einem n-dimensionalen Objekvektor $x_i^t = (x_{ij}^t \in R \mid j = 1, ..., n) \in R^n$, der Fitnessbewertung im Fall einer Ein-Ziel-Optimierung $f(x_i^t) \in R$ und einem Strategievektor $\sigma_i^t = (\sigma_{ij}^t \in R \mid j = 1, ..., n) \in R^n$, sodass die Eltern-Population beschrieben wird durch:

$$P^t = \{a_i^t = (x_i^t, f(x_i^t), \sigma_i^t) \mid i = 1, ..., \mu\}. \quad (203)$$

Die Aufnahme des Strategievektors, der Mutationsschrittweiten σ_{ij}^t für jeden der n Objektvariablen x_{ij}^t kodiert, in die Struktur eines Individuums, war ein wichtiger Schritt in der Entwicklung der ES, da auf diese Weise die Möglichkeit der Selbstadaption der Strategieparameter eingeführt wird (Schwefel (1981[306])). Anstatt σ^t durch eine externe Heuristik zu regeln, werden die Strategieparameter als Teil der genetischen Information eines Individuums betrachtet, auf den Rekombinations- und Mutations-Operatoren zugreifen können. Auf diese Weise entwickeln sich bessere Strategieparameter parallel zu besser werdenden Objektvariablen, d.h. es findet durch die Selbstadaption eine Evolution der Lernfähigkeit statt.

Zu Beginn der Evolution wird eine Ausgangspopulation $P^{t=0}$ durch einen Zufallsprozess erzeugt, der zunächst gleichverteilt zufällig innerhalb der erlaubten Region des n-dimensionalen Objektvariablen- und des n-dimensionalen Strategievariablenraumes μ Positionen spezifiziert, die zusammen μ Initialisierungs-Individuen $a_i^{t=0}$ festlegen. Danach wird auf der Basis des Objektvektors die Fitness $f(x_i^{t=0})$ der Individuen bestimmt, womit die Initialisierungsgeneration t=0 abgeschlossen ist. Es folgt die erste Generation t=1, in der Reproduktionsoperationen durchgeführt werden, indem λ mal nacheinander ρ Eltern-Individuen aus P^t mit Zurücklegen gezogen werden, wobei im weiteren vereinfachend ρ = 2 verwendet wird.

Die Reproduktionsoperation unterteilt sich in zwei Rekombinations- und zwei Mutationsoperationen, bei denen zwei Eltern $a_{E(1,k)}^t$ und $a_{E(2,k)}^t$ einen Nachkommen $a_{N(k)}^t$ erzeugen. Die Auswahl der Eltern wird als Selektion sel_R zur Reproduktion bezeichnet und ist bei der Standard-ES durch ein gleichverteilt Zufälliges Ziehen aus der Elternpopulation P^t operationalisiert:

$$sel_R(P^t) = E_{N(k)}^t = \{a_{E(j,k)}^t \mid j = 1, ..., \rho = 2\}, \tag{204}$$

Andere Selektionsfunktionen können Eigenschaften der Eltern berücksichtigen, wie z.B. ihren Fitnesswert oder die Distanz zwischen den Objektvektoren $d_X(x_{E(1,k)}^t, x_{E(2,k)}^t)$, was insbesondere bei einer strukturierten Elternpopulation sinnvoll wird (Bachelier (1998b[16])). Komplexere Selektionsfunktionen werden möglich, wenn in der Datenstruktur der Eltern Fitness-Approximationsmodelle integriert sind, mit denen sie die Fitness potentieller Nachkommen bewerten können und entsprechend diesen Schätzungen entscheiden, ob eine bestimmte Partnerwahl günstig ist oder nicht. Eine Viehlzahl von Selektionsfunktionen mit Hilfe individueller und populationsspezifischer Fitness-Approximationsmodelle findet sich in Bachelier (1999d[22]), worauf in diesem Kontext jedoch nicht näher eingegangen werden soll.

Im ersten Schritt der Erzeugung von $a_{N(k)}^t$ werden die Strategievektoren $\sigma_{E(1,k)}^t$ und $\sigma_{E(2,k)}^t$ der Eltern zu $\sigma_{N(k)}^t$ rekombiniert, was allgemein modeliert wird durch:

$$\sigma_{N(k)}^t = rec(\sigma_{E(1,k)}^t, \sigma_{E(2,k)}^t). \tag{205}$$

Diese Darstellung spezifiziert noch nicht die konkrete Rekombinationsfunktion, die zwei n-dimensionale Vektoren als Input verwendet und einen n-dimensionalen Vektor als Output liefert. Die beiden häufigsten Rekombinationsfunktionen sind die intermediäre und die diskrete Rekombination. Bei der intermediären Rekombination wird der arithmetische Mittelwert der beiden Elternvektoren erzeugt:

$$\sigma_{N(k)}^t = rec(\sigma_{E(1,k)}^t, \sigma_{E(2,k)}^t)_{inter} = (\sigma_{N(k),i}^t = 1/2*(\sigma_{E(1,k),i}^t + \sigma_{E(2,k),i}^t) \mid i = 1, ..., n). \tag{206}$$

Bei der diskreten Rekombination wird eine der beiden Elternkomponenten gleichverteilt zufällig ausgewählt. Hierzu wird ein Zufallsexperiment mit der Ergebnismenge {0, 1} durchgeführt, wobei die Komponente $\sigma_{E(1,k),i}^t$ des ersten Elternteils $a_{E(1,k)}^t$ verwendet wird, wenn das Zufallsexperiment „0“ liefert, und $\sigma_{E(2,k),i}^t$ wenn das Experiment „1“ liefert:

$$\sigma_{N(k)}^t = rec(\sigma_{E(1,k)}^t, \sigma_{E(2,k)}^t)_{disk} = (\sigma_{N(k),i}^t \mid i = 1, ..., n), \text{ mit} \tag{207}$$

$$\sigma_{N(k),i}^{t} = \begin{cases} \{\sigma_{E(1,k),i}^{t}, & \text{wenn } E = \{0\}, \\ \{\sigma_{E(2,k),i}^{t}), & \text{wenn } E = \{1\}. \end{cases} \quad (208)$$

Nach der Rekombination der Strategievektoren wird $\sigma_{N(k)}^{t}$ einer Selbst-Mutation unterzogen, was allgemein modelliert wird durch:

$$\sigma_{N(k)}^{t'} = m(\sigma_{N(k)}^{t} \mid \sigma_{N(k)}^{t}). \quad (209)$$

Die Mutation eines Strategievektors und eines Objektvektors werden durch jeweils andere Mutationsfunktionen durchgeführt, wobei für die Strategievektor-Mutation eine multiplikative Funktion verwendet wird. Zur Mutation einer Komponente $\sigma_{N(k),i}^{t}$ wird zunächst ein Zufallswert aus einer $N(0, \sigma_{N(k),i}^{t})$-Gauss-Verteilung erzeugt, welcher der Exponentialfunktion exp[.] unterzogen wird und mit der Komponente $\sigma_{N(k),i}^{t}$ multipliziert wird. Für den mutierten Strategievektor des Nachkommens ergibt sich:

$$\sigma_{N(k)}^{t'} = (\sigma_{N(k),i}^{t'} = \sigma_{N(k),i}^{t} * \exp[N(0, \sigma_{N(k),i}^{t})]) \mid i = 1, ..., n). \quad (210)$$

In Schwefel (1981[306]) und Bäck & Schwefel (1993[23]) finden sich alternative Ansätze zur Strategievektor-Mutation, die hier jedoch nicht dargestellt werden sollen.

Im nächsten Schritt werden die Objektvektoren $x_{E(1,k)}^{t}$ und $x_{E(2,k)}^{t}$ der Eltern rekombiniert, wobei hierbei oft die intermediäre Rekombinationsfunktion verwendet wird. Allgemein gilt:

$$x_{N(k)}^{t} = rec(x_{E(1,k)}^{t}, x_{E(2,k)}^{t}). \quad (211)$$

Der letzte Schritt in der Erzeugung eines Nachkommens besteht in der Mutation des rekombinierten Objektvektors des Nachkommen durch den selbst-mutierten Schrittweitenvektor $\sigma_{N(k)}^{t'}$:

$$x_{N(k)}^{t'} = m(x_{N(k)}^{t} \mid \sigma_{N(k)}^{t'}). \quad (212)$$

Die Mutation eines Objektvektors wird durch eine additive Mutationsfunktion durchgeführt, indem für eine Komponente $x_{N(k),i}^{t}$ ein Zufallswert aus einer $N(0, \sigma_{N(k),i}^{t'})$-Gauss-Verteilung erzeugt wird, der zu der Objektvektor-Komponente addiert wird. Für den mutierten Objektvektor des Nachkommens ergibt sich:

$$x_{N(k)}^{t'} = (x_{N(k),i}^{t'} = x_{N(k),i}^{t} + N(0, \sigma_{N(k),i}^{t'})) \mid i = 1, ..., n). \quad (213)$$

Der Nachkomme $a_{N(k)}^{t}$ wird mit der noch unvollständigen Struktur $a_{N(k)}^{t} = (x_{N(k)}^{t'}, \_, \sigma_{N(k)}^{t'})$ in die Zwischenpopulation ZP^{t} aufgenommen:

$$ZP^{t}(neu) = ZP^{t}(alt) \cup \{a_{N(k)}^{t}\}. \quad (214)$$

Nachdem alle λ Nachkommen in der Zwischenpopulation ZP^{t} aufgenommen wurden, wird die Reproduktionsphase der Generation t beendet, und es wird die Bewertungsphase eingeleitet, in der alle Nachkommen einen Fitnesswert $f(x_{N(k)}^{t'}) \in R$ erhalten, sodass sie eine vollständige Struktur $a_{N(k)}^{t} = (x_{N(k)}^{t'}, f(x_{N(k)}^{t'}), \sigma_{N(k)}^{t'})$ besitzen.

Der Bewertungsphase folgt die Selektion sel_N zur Übernahme in die Nachfolgepopulation, bei der eine

Teilmenge von ZP^t ausgewählt wird, die als Population P^{t+1} der nächsten Generation verwendet wird, falls kein Abbruchkriterium greift:

$$sel_N(ZP^t) = P^{t+1}. \tag{215}$$

Bei den ES wird standardmäßig eine deterministische Selektion nach dem Fitnesswert der Nachkommen bei einer (μ,λ)-ES bzw. der Nachkommen und der Eltern bei einer (μ+λ)-ES durchgeführt, d.h. die Zwischenpopulation als Menge wird bei einer Minimierung in eine Liste nach steigenden Fitnesswerten geordnet, aus der die ersten μ Elemente ausgewählt werden. Komplexere Selektionsfunktionen verwenden zusätzlich weitere Eigenschaften der Individuen, wie ihren Objektvektor, indem z.B. nur die besten unähnlichen Individuen in die Nachfolgepopulation übernommen werden dürfen, was eine Form von Clusteranalyse der Individuen in ZP^t voraussetzt. Besitzen die Individuen erweiterte Strukturen, wie z.B. zusätzlich ein Fitness-Approximationsmodell, so kann die sel_N-Operation als Zwei-Ziel-Optimierungsverfahren modelliert werden, das die Individuen mit dem besten Objektvektor und dem besten Approximationsmodell auswählt (siehe Bachelier (1999d[22])).

Die übernommenen Individuen bilden die Eltern-Population P^{t+1}, wobei eine nachfolgende Generation t+1 erst begonnen wird, wenn ein Abbruchkriterium nicht erfüllt ist. Bei einer Minimierung kann ein Schwellenwert $S_{f(min)}$ extern vorgegeben werden, wobei ein Abbruch erfolgt, wenn zum ersten mal ein Individuum einen Fitnesswert kleiner $S_{f(min)}$ erreicht. Alternativ können Abbruchkriterien verwendet werden, die den Optimierungsverlauf bewerten. Sollte sich das Fitnessniveau der Population über mehrere Generationen nicht verbessern, d.h. stagniert die Optimierung, so kann ein Abbruch erfolgen, wenn keine weitere Verbesserung erwartet werden kann (siehe zur Bewertung des Populationsfortschrittes Bachelier (1999b[20])).

Start(ρ-geschlechtliche_(μ,λ)-ES)

1) Generiere Ausgangspopulation $P^0 = \{a_i^0 = (x_i^0, \sigma_i^0, \_) \mid i = 1, ..., \mu\}$ der Generation t = 0.
2) Bewerte Individuen der Ausgangspopulation: $P^0 = \{a_i^0 = (x_i^0, \sigma_i^0, f(x_i^0)) \mid i = 1, ..., \mu\}$.
3) Reproduktionsphase: Erzeuge Zwischenpopulation ZP^t durch ρ-geschlechtliche Reproduktion.
    3.0) Initialisierung: $ZP^t := \varnothing$.
    3.1) k = 1.
    3.2) Selektion zur Reproduktion: $sel_R(P^t) = E_{N(k)}^t = \{a_{E(j,k)}^t \mid j = 1, ..., \rho \le \mu\}$.
    3.3) Reproduktion: $rep(E_{N(k)}^t) = a_{N(k)}^{t'}$.
        3.3.1) Strategievektoren-Rekombination: $rec(\sigma_{E(j,k)}^t \mid j = 1, ..., \rho) = \sigma_{N(k)}^t$.
        3.3.2) Strategievektoren-Mutation: $m(\sigma_{N(k)}^t \mid \sigma_{N(k)}^t) = \sigma_{N(k)}^{t'}$.
        3.3.3) Objektvektoren-Rekombination: $rec(x_{E(j,k)}^t \mid j = 1, ..., \rho) = x_{N(k)}^t$.
        3.3.4) Objektvektoren-Mutation: $m(x_{N(k)}^t \mid \sigma_{N(k)}^{t'}) = x_{N(k)}^{t'}$.
    3.4) Einfügen des Nachkommen in ZP^t: $ZP^t(neu) = ZP^t(alt) \cup \{a_{N(k)}^{t'} = (x_{N(k)}^{t'}, \sigma_{N(k)}^{t'}, \_)\}$.
    3.5) Wenn $k + 1 \le \lambda$, dann setze k = k + 1 und gehe zu 3.2), sonst gehe zu 4).
4) Bewerte Individuen der Zwischenpopulation: $ZP^t = \{a_{N(k)}^{t'} = (x_{N(k)}^{t'}, \sigma_{N(k)}^{t'}, f(x_{N(k)}^{t'})) \mid i = 1, ..., \lambda\}$.
5) Selektion zur Übernahme in die Nachfolgepopulation: $sel_N(ZP^t) = P^{t+1}$.
6) Ist Abbruchbedingung erfüllt, dann Ende, sonst setze t = t + 1 und gehe zu 3).

Ende.

Der Unterschied zwischen einer solchen Komma-ES ((μ,λ)-ES) und einer Plus-ES ((μ+λ)-ES) besteht nur darin, dass bei einer Plus-ES die Individuen aus P^t zusätzlich in die Zwischenpopulation aufgenommen werden, wodurch Eltern mit ihren Nachkommen im Rahmen von sel_N um die Übernahme in die Nachfolgepopulation konkurrieren. Bei der deterministischen sel_N-Strategie folgt daraus, dass Super-Individuen, d.h. Individuen mit einem sehr guten Fitneßwert, faktisch beliebig lange existieren können, und so mit ihren Nachkommen die Population dominieren werden, was zu einer frühzeitigen Konvergenz führt.

2.5.2) Mehr-Ziel-ES

Bei einer Mehr-Ziel-Optimierungsproblem (siehe Abschnitt 2.4)) wird eine Menge von m Fitnessfunktionen $f_k(x)$, k = 1, ..., m, und eventuell eine Menge von p Constraints $g_l(x)$, l = 1, ..., p, verwendet. Das Ziel eines Optimierungsverfahrens ist es, eine Menge von nicht-dominanten Individuen zu erzeugen, die einen möglichst kleinen Abstand zur Pareto-Front besitzen, und die eine große Diversität besitzen, d.h. alle Abschnitte der Pareto-Front sollen durch Individuen in geeigneter Weise repräsentiert werden (Veldhuizen (1999[347]), Zitzler (1999[375])).

Pareto-ES, als eine Form von Mehr-Ziel-ES, orientieren sich an dem grundsätzlichen Ablauf eines (μ+λ)-ES, wobei insbesondere die Selektion zur Übernahme in die Nachfolgepopulation durch die Verwendung der Pareto-Dominanz modifiziert wird. Durch die Verwendung einer „+“-Strategie gelangt man zu einem Elite-Verfahren (elitism sheme), das bei Mehr-Ziel-EA eine besonders wichtige Rolle spielt (Zitzler (1999: 26, 39f[375])). In Fonseca & Fleming (1995[116]) wird beschrieben, dass Pareto-EA, die sich ausschließlich auf die Pareto-Dominanz beziehen und keine weiteren Konzepte aus dem Repertoire der EA nutzen, bei einer großen Anzahl von Fitnessfunktionen keine hinreichend gute Performance besitzen. Entsprechende Verfahren werden im weiteren vorgestellt, wobei insbesondere Elite-Konzepte als zusätzliche Verfahrenskomponenten verwendet werden, die zur gleichen Verfahrens-Klasse wie der Genetic-Load gehören (Jaquard (1974[176]), Born (1978[53]), Born & Bellmann (1983[54]), Born et al. (1992: 190[55]), Bachelier (1998b: 103ff[16])).

Bezüglich der Selektion zur Übernahme in die Nachfolgepopulation stellt sich bei einer Pareto-ES die Frage, ob nur nicht-dominante Individuen übernommen werden sollen. Das damit verbundene Problem ist die konstante Anzahl der Individuen in der Elternpopulation P^t mit μ und Zwischenpopulation ZP^t mit λ Individuen. Bei der Reproduktion kann in keiner Weise davon ausgegangen werden, dass bei λ Individuen in ZP^t mindestens μ untereinander nicht-dominant sind, wenn eine (μ,λ)-Strategie verwendet wird. Werden konstant λ Individuen in ZP^t erzeugt, und seien $\mu_{\overline{dom}}^t$ davon nicht-dominant, so ergeben sich folgende Möglichkeiten zur Selektion in Abhängigkeit der Anzahl von $\mu_{\overline{dom}}^t$:

1) $\mu_{\overline{dom}}^t < \mu$ sind nicht-dominant
   1.1) Auswahl aller nicht-dominanten Individuen, mit der Folge, dass in P^{t+1} mehr als μ Individuen vorliegen werden.

1.2) Zusätzliche Auswahl dominanter Individuen, mit der Folge, dass in P^{t+1} genau μ Individuen vorliegen werden.

2) $\mu_{\overline{dom}}^{t} > \mu$ sind nicht-dominant

2.1) Auswahl eines (zufälligen) Teils der nicht-dominanten Individuen, mit der Folge, dass in P^{t+1} genau μ Individuen vorliegen werden.

2.2) Auswahl aller nicht-dominanten Individuen, mit der Folge, dass in P^{t+1} mehr als μ Individuen vorliegen werden.

3) $\mu_{\overline{dom}}^{t} = \mu$ sind nicht-dominant

Auswahl aller nicht-dominanten Individuen, mit der Folge, dass in P^{t+1} genau μ Individuen vorliegen werden.

Diese Möglichkeiten gehen davon aus, dass nicht-dominante Individuen einer Generation in der nachfolgenden Generation auch reproduktiv wirksam sind. Als Alternative bietet sich an, alle nicht-dominanten Individuen, die im Generationsverlauf erzeugt wurden, in einer generationsübergreifenden Paretomenge $PM^{1->t}$ zu sammeln, und aus dieser Menge eine Eltern-Population zusammenzustellen. Diese Vorgehensweise gehört zur gleichen Verfahrens-Klasse wie der Genetic-Load (Jaquard (1974[176]), Born (1978[53]), Born & Bellmann (1983[54]), Born et al. (1992: 190[55]), Bachelier (1998b: 103ff[16])), bei dem ebenfalls Genmaterial existiert, das nicht, nicht immer, oder nicht mit der gleichen Wahrscheinlichkeit am Reproduktionsprozess teilnimmt. Der Unterschied besteht darin, dass beim Genetic-Load alternative Objektvektoren innerhalb eines Individuums verwendet werden, während bei dieser Version der Pareto-ES alternative Individuen verwendet werden. Es sind dabei zwei Fälle zu unterscheiden:

1) $\#PM^{1->t} \leq \mu$

In diesem Fall, der zu Beginn einer Mehr-Ziel-Optimierung eintritt, kann die gesamte Pareto-Menge als Eltern-Population verwendet werden, aus der die Selektion zur Reproduktion Eltern für eine geschlechtliche Reproduktion auswählt.

2) $\#PM^{1->t} > \mu$

In diesem Fall, können zwei-stufige Selektions-Operationen verwendet werden, die zunächst aus $PM^{1->t}$ eine Eltern-Population P^t erzeugen, aus der die Selektion zur Reproduktion Eltern für eine geschlechtliche Reproduktion auswählt. Eine gleichverteilte Zufalls-Auswahl bei beiden Stufen entspricht jedoch einer ein-stufigen Auswahl, sodass eine nicht-gleichverteilte Auswahl z.B. den Erzeugungszeitpunkt eines nicht-dominanten Individuums berücksichtigen könnte.

Eine solche Vorgehensweise hat gewisse Ähnlichkeit zu einer „+“-ES in Verbindung mit einer tendenziell steigenden Elternzahl. Im Gegensatz zu einer Ein-Ziel-ES, die bei einer Plus-Strategie anfällig für lokale Minima ist, muss dies bei einer Mehr-Ziel-Optimierung keinen Nachteil bedeuten, da das Pareto-Kriterium durch die Und-Verknüpfung der einzelnen Vergleiche viele Individuen in $PM^{1->t}$ zulässt, sodass ein großes Differenzierungs-Spektrum erhalten bleibt.

Im Algorithmus Pareto-ES_1 wird zunächst eine hinreichend große zufällige Ausgangspopulation $P^{t=0}$ mit μ Elementen erzeugt, aus der die Paretomenge $PM^{t=0}$ gebildet wird, die mehr als ein Element enthalten soll. Im nächsten Schritt werden λ Nachkommen erzeugt und in die Zwischenpopulation ZP^t eingefügt, wobei eine 2-geschlechtliche Reproduktion unterstellt wird. Die beiden Eltern werden aus der Paretomenge $PM^{1->t}$ durch die Selektions-Operation sel_R ausgewählt, d.h. es werden immer nur unter-

einander nicht-dominante Individuen zur Reproduktion herangezogen. Nachdem die Zwischenpopulation ZP^t erzeugt wurde, werden die λ Elemente bewertet, und aus ZP^t wird in die generationsinterne Paretomenge $PM(ZP^t)$ erzeugt, d.h. die Menge der nicht-dominanten Individuen aus ZP^t. Danach wird die neue generationsübergreifende Paretomenge $PM^{1->t+1}$ gebildet, indem die Paretomenge aus der Vereinigungsmenge $PM^{1->t} \cup PM(ZP^t)$ gebildet wird.

Start(Pareto-ES_1)

1) Generiere zufällige Ausgangspopulation $P^{t=0} = \{a_i^{t=0} = (x_i^{t=0}, \sigma_i^{t=0}, \_) \mid i = 1, ..., \mu\}$.
2) Bewerte Individuen der Ausgangspopulation: $P^{t=0} = \{a_i^{t=0} = (x_i^{t=0}, \sigma_i^{t=0}, f(x_i^{t=0})) \mid i = 1, ..., \mu\}$.
3) Bilde Paretomenge $PM^{t=0}$ aus $P^{t=0}$: $PM(P^{t=0})$, wobei $\#PM^{t=0} > 1$ angenommen wird.
4) Erzeuge Zwischen-Population ZP^t durch 2-geschlechtliche Reproduktion.
    4.0) Initialisierung: $ZP^t := \varnothing$.
    4.1) $k = 1$.
    4.2) $sel_R(PM^t) = E_{N(k)}^t = \{a_{E(i,k)}^t \mid i = 1, 2\}$.
    4.3) $rep(E_{N(k)}^t) = a_{N(k)}^t$.
    4.4) Einfügen von $a_{N(k)}^t = (x_{N(k)}^{t'}, \sigma_{N(k)}^{t'}, \_)$ in ZP^t: $ZP^t(neu) = ZP^t(alt) \cup \{a_{N(k)}^t\}$.
    4.5) $i + 1 \leq \lambda$, dann setze $i = i + 1$ und gehe zu 4.2), sonst gehe zu 5).
5) Bewerte Individuen aus ZP^t: $ZP^t = \{a_{N(k)}^t = (x_{N(k)}^{t'}, \sigma_{N(k)}^{t'}, f(x_{N(k)}^{t'})) \mid f(x_{N(k)}^{t'}) \in R^m, i = 1, ..., \lambda\}$.
6) Bilde generationsinterne Paretomenge aus ZP^t: $PM(ZP^t)$.
7) Aktualisiere generationsübergreifende Paretomenge: $PM^{1->t+1} = PM(PM^{1->t} \cup PM(ZP^t))$.
8) Ist Abbruchbedingung erfüllt, dann Ende, sonst setze $t = t + 1$ und gehe zu 4).

Ende.

Die Vorgehensweise in Pareto-ES_1 kann weiter vereinfacht werden, wenn ein Steady-State-Prinzip verwendet wird. Zunächst wird genau ein Individuum zufällig erzeugt, das als Element der Initialisierungs-Paretomenge $PM^{t=0}$ verwendet wird. Mit Hilfe der ungeschlechtlichen Reproduktions-Operation, d.h. mit einer Mutation von Strategie- und Objektvektor, wird genau ein Nachkomme erzeugt und bewertet. Sind Elternteil und Nachkomme nicht-dominant, dann wird die Paretomenge aktualisiert, und es wird zu einer geschlechtlichen Reproduktion auf der Basis der jeweiligen Paretomenge übergegangen, bei der jeweils genau ein Nachkomme erzeugt und direkt bewertet wird. Nach der Bewertung wird die neue Paretomenge aus der Vereinigung der alten Paretomenge und dem Nachkommen gebildet, wobei dieser aufgenommen wird, wenn er nicht-dominant gegenüber allen verbleibenden Elementen in der alten Paretomenge ist.

Start(Pareto-ES_2)

1) Generiere zufällige Ausgangspopulation $P^{t=0} = \{a_i^{t=0} = (x_i^{t=0}, \sigma_i^{t=0}, \_) \mid i = 1\}$.
2) Bewerte Individuum der Ausgangspopulation: $P^{t=0} = \{a_i^{t=0} = (x_i^{t=0}, \sigma_i^{t=0}, y(x_i^{t=0})) \mid i = 1\}$.
3) Bilde Paretomenge $PM^{t=0}$ aus $P^{t=0}$: $PM(P^{t=0})$, wobei $\#PM^{t=0} > 1$ angenommen wird.
4) Erzeuge Nachkomme durch ungeschlechtliche Reproduktion.
    4.1) $sel_R(PM^t) = E_{N(k)}^t = \{a_{E(i,k)}^t \mid i = 1\}$.

4.2) $rep(E_{N(k)}^t)_{ungeschl} = a_{N(k)}^t = (x_{N(k)}^t, \sigma_{N(k)}^t, \_)$.

4.2.1) $m(\sigma_{E(i,k)}^t \mid \sigma_{E(i,k)}^t) = \sigma_{N(k)}^t$.

4.2.2) $m(x_{E(i,k)}^t \mid \sigma_{N(k)}^t) = x_{N(k)}^t$.

4.3) Bewerte Nachkomme $a_{N(k)}^t = (x_{N(k)}^t, \sigma_{N(k)}^t, f(x_{N(k)}^t))$.

4.4) Bilde generationsübergreifende Paretomenge $PM^{1->t+1}$ aus $PM^{1->t} \cup \{a_{N(k)}^t\}$.

4.5) Wenn $\#PM^{1->t+1} > 1$, dann gehe zu 5), sonst gehe zu 4).

5) Erzeuge Nachkomme durch 2-geschlechtliche Reproduktion.

5.1) $sel_R(PM^t) = E_{N(k)}^t = \{a_{E(i,k)}^t \mid i = 1, 2\}$.

5.2) $rep(E_{N(k)}^t)_{geschl} = a_{N(k)}^{t'}$.

5.3) Bewerte Nachkomme $a_{N(k)}^t = (x_{N(k)}^{t'}, \sigma_{N(k)}^{t'}, f(x_{N(k)}^{t'}))$.

5.4) Bilde generationsübergreifende Paretomenge $PM^{1->t+1}$ aus $PM^{1->t} \cup \{a_{N(k)}^t\}$.

6) Ist Abbruchbedingung erfüllt, dann Ende, sonst setze $t = t + 1$ und gehe zu 5).

Ende.

Aus einer Zwischenpopulation ZP^t von λ Nachkommen kann die generationsinterne Paretomenge $PM(ZP^t)$ erzeugt werden, aus der die Elemente bestimmt werden können, die zu den Elementen in der generationsübergreifenden Paretomenge $PM^{1->t}$ nicht-dominant sind, wobei diese Teilmenge als $PM(ZP^t \mid PM^{1->t})$ bezeichnet werden soll. Die dazu komplementäre Teilmenge ist $PM(ZP^t) \setminus PM(ZP^t \mid PM^{1->t})$. In der Nachfolgegeneration müssen die Eltern nicht direkt aus $PM^{1->t+1}$ selektiert werden, sondern es kann eine Elternpopulation P^{t+1} mit konstanten μ Individuen verwendet werden, die aus den drei Mengen $PM(ZP^t)$, $PM(ZP^t \mid PM^{1->t})$ und $PM^{1->t}$ durch unterschiedliche Operationen zusammengestellt werden kann. Faktisch handelt es sich dabei um eine Spezifikation der Selektion zur Übernahme bzw. zur Erzeugung der Nachfolgepopulation, die als Population der potentiellen Eltern verwendet wird:

$$P^{t+1} = sel_N(PM(ZP^t), PM^{1->t}, PM(ZP^t \mid PM^{1->t})). \tag{216}$$

Beispiele dieser Elternmengen-Selektion sind:

1) Verwende eine zufällige Auswahl von μ Individuen aus $PM^{1->t+1} = PM^{1->t} \cup PM(ZP^t \mid PM^{1->t})$.
2) Verwende eine zufällige Auswahl von μ Individuen aus $PM^{1->t} \cup PM(ZP^t)$.
3) Verwende eine zufällige Auswahl von μ_1 Individuen aus $PM(ZP^t)$ und μ_2 aus $PM^{1->t}$, mit $\mu = \mu_1 + \mu_2$.
4) Verwende eine zufällige Auswahl von μ_1 Individuen aus $PM(ZP^t \mid PM^{1->t})$ und μ_2 aus $PM^{1->t}$, mit $\mu = \mu_1 + \mu_2$.
5) Wenn $PM(ZP^t) < \mu$, dann verwende $PM(ZP^t)$ und eine zufällige Auswahl von $\mu - \#PM(ZP^t)$ Individuen aus $PM^{1->t}$.
6) Wenn $PM(ZP^t \mid PM^{1->t}) < \mu$, dann verwende $PM(ZP^t \mid PM^{1->t})$ und eine zufällige Auswahl von $\mu - \#PM(ZP^t \mid PM^{1->t})$ Individuen aus $PM^{1->t}$.

Die Beispiele 1), 4) und 6) verwenden als Eltern ausschließlich Individuen, die untereinander nicht-dominant sind, während die anderen Beispiele durch die Verwendung von $PM(ZP^t)$ auch Individuen zulassen, die von Individuen aus $PM^{1->t}$ dominiert werden.

Ausgehend von einer zufälligen Anfangspopulation $P^{t=0}$ als einer Liste mit μ Individuen wird die

Anfangs-Paretomenge $PM^{t=0}$ gebildet. Nach der Initialisierung soll für t = 1 die Paretomenge als Elternmenge verwendet werden, wohingegen in allen späteren Generationen die Population $P^{t=0}$ als Elternmenge verwendet wird. Nach der Festlegung der Elternmenge wird eine Zwischen-Population ZP^t als eine Menge mit $\lambda > \mu$ Individuen durch 2-geschlechtliche Reproduktion erzeugt, aus der die Paretomengen $PM(ZP^t)$ und $PM(ZP^t \mid PM^{1->t})$ gebildet werden.

Durch die Selektion sel_N zur Übernahme in bzw. Erzeugung der Nachfolgepopulation wird aus den Mengen $PM(ZP^t)$, $PM^{1->t}$, $PM(ZP^t \mid PM^{1->t})$ die Nachfolgepopulation P^{t+1} zusammengestellt. Dabei soll gelten, dass durch die Listenstruktur Mehrfach-Nennungen erlaubt sind, da insbesondere zu Beginn des Verfahrens, die Paretomenge $PM^{1->t}$ möglicherweise nicht genug Elemente besitzt, um zusammen mit den anderen Mengen μ Elemente zu erzeugen.

Treten Mehrfach-Nennungen auf, so besteht die Möglichkeit, dass bei einer Zufallsauswahl von zwei Eltern $a_{E(1,k)}^t$ und $a_{E(2,k)}^t$ aus P^t die beiden gleichen Individuen ausgewählt werden, was auf eine ungeschlechtliche Reproduktion hinauslaufen würde. Um dies zu verhindern, wird bei der Selektion zur Reproduktion gefordert, dass beide Eltern unterschiedliche Individuen sind.

Start(Pareto-ES_3)

1) Erzeuge und bewerte Individuen-Initialisierungsliste: $P^{t=0} = (a_i^{t=0} = (x_i^{t=0}, \sigma_i^{t=0}, f(x_i^{t=0}) \mid i = 1, ..., \mu)$.
2) Bilde Paretomenge $PM^{t=0}$ aus $P^{t=0}$ und verwende $PM^{t=0}$ als Elternmenge P^t.
3) Erzeuge Zwischenpopulation ZP^t durch 2-geschlechtliche Reproduktion aus der Elternmenge P^t.
    3.0) Initialisierung: $ZP^t := \varnothing$.
    3.1) k = 1.
    3.2) $sel_R(P^t) = E_{N(k)}^t = \{a_{E(i,k)}^t \mid i = 1, 2\}$, mit $a_{E(1,k)}^t \neq a_{E(2,k)}^t$.
    3.3) $rep(E_{N(k)}^t) = a_{N(k)}^t$.
    3.4) Einfügen des Nachkommen $a_{N(k)}^t = (x_{N(k)}^t, \sigma_{N(k)}^t, \_)$ in ZP^t: $ZP^t(neu) = ZP^t(alt) \cup \{a_{N(k)}^t\}$.
    3.5) $i + 1 \leq \lambda$, dann setze i = i + 1 und gehe zu 3.2), sonst gehe zu 4).
4) Bewerte Individuen der Zwischenpopulation: $ZP^t = \{a_{N(k)}^t = (x_{N(k)}^t, \sigma_{N(k)}^t, f(x_{N(k)}^t)) \mid k = 1, ..., \lambda\}$.
5) Bilde generationsinterne Paretomenge $PM(ZP^t)$ aus ZP^t.
6) Bilde generationsübergreifende Paretomenge $PM^{1->t+1} = PM^{1->t} \cup PM(ZP^t \mid PM^{1->t})$.
7) Erzeuge Nachfolgepopulation: $P^{t+1} = sel_N(PM(ZP^t), PM^{1->t}, PM(ZP^t \mid PM^{1->t}))$.
8) Ist Abbruchbedingung erfüllt, dann Ende, sonst gehe zu 3).

Ende.

Ein schwieriges Problem ist die Formulierung eines Abbruchkriteriums innerhalb einer Pareto-ES, da die Paretomenge nicht stetig wächst, sondern der Fall eintreten kann, dass durch ein neues Individuum ein großer Teil der vorhandenen Elemente in der generationsübergreifenden Paretomenge gelöscht wird. Aus diesem Grunde ist es problematisch, die Entwicklung der Paretomenge über den Generationsverlauf zu verfolgen, und ein Abbruchkriterium herzuleiten.

2.6) Intervall-Selektions-Operatoren

Sollen Rankings auf der Basis von externen Messwerten oder von numerisch erzeugten Eigenschaften wie z.B. Modellqualitäten erzeugt werden, so besitzen diese effektiv betrachtet nicht die Struktur eines Punktes, sondern die Struktur eines Intervalls, da im ersten Fall Messunsicherheiten und im zweiten Fall systematische Fehler (Bias) und Varianzen einflissen (siehe auch Bauch et al. (1987[30])). Das gleiche gilt, wenn die Eigenschaften durch Intervall-Arithmetiken erzeugt werden, oder wenn die grundlegende Eigenschaft, auf die sich eine Bewertung bezieht, selbst ein Intervall oder ein Intervall-Vektor ist. Werden Dokumentvektoren Eigenschaftsintervalle zugeordnet, so ist die Spezifikation einer Dominanzfunktion notwendig, die entscheidet, welcher Dokumentvektor besser ist.

Im weiteren soll eine Kandidatenmenge KM betrachtet werden, deren Elemente m_j aus einem Punktvektor $x_j \in R^n$ und aus einer intervallbasierten Bewertung $I_f(x_j)$ besteht:

$$KM = \{m_j = (x_j, I_f(x_j)) \mid j = 1, ..., \mu_K\}. \tag{217}$$

Der Punktvektor x_j beschreibt eine Eigenschaft und das Intervall $I_f(x_j)$ ordnet der Eigenschaft eine Bewertung in Form eines Intervalls zu. Analog könnte die Eigenschaft selbst als Intervall $I(x_j)$ beschrieben werden, sodass die folgende Datenstruktur erzeugt wird:

$$KM = \{m_j = (I(x_j), I_f(x_j)) \mid j = 1, ..., \mu_K\}. \tag{218}$$

Die Verwendung von x_j bzw. $I(x_j)$ macht bezüglich der nachfolgenden Beschreibungen jedoch keinen Unterschied macht, da Selektions- und Ranking-Operationen ausschlisslich auf dem Bewertungs-Intervall $I_f(x_j)$ basieren sollen.

Das Intervall $I_f(x_j)$ ist durch eine reelle untere Grenze $\underline{f}(x_j)$ und eine reelle obere Grenze $\bar{f}(x_j)$ definiert, mit $\underline{f}(x_j) < \bar{f}(x_j)$, wobei die Grenzen Element des Intervalls sein sollen, d.h. $I_f(x_j) = [\underline{f}(x_j), \bar{f}(x_j)]$. Analog zur Menge der reellen Zahlen wird eine Menge aller reellen Intervalle I gebildet, in der alle Intervalle liegen, deren linke Grenze kleiner-gleich der rechten Grenze ist. Die Menge der reellen Zahlen ist Teilmenge von I, da die rellen Intervalle, deren linke und rechte Grenze übereinstimmt, reelle Zahlen bzw. Punkt-Intervalle sind.

Der allgemeine Fall, dass ein Intervall-Bewertungsvektor $I_f(x_j) = (I_f(x_j)_k \in I \mid k = 1, ..., m) \in I^m$ vorliegt, kombiniert die Darstellungen zur Mehr-Ziel-Optimierung und der Intervall-Selektions-Operatoren und soll hier nicht weiter beschrieben werden.

Ziel ist es im weiteren, aus der Menge KM eine Liste KL zu generieren, in der die Elemente m_j entsprechend ihrer Intervall-Bewertung $I_f(x_j) \in I$ geordnet sind. Alternativ kann als Ziel die Auswahl der besten Elemente ohne ein Ranking durchgeführt werden, wobei die Generierung eines Rankings eine anspruchsvollere Aufgabe ist. Die besten Elemente bilden eine Menge, sodass diese Aufgabe einer binären Klassifikation von KM entspricht, wobei eine Rangfolge der Elemente innerhalb dieser beiden Teilmengen irrelevant ist. Bei einem Ranking spielt die Rangfolge der Elemente bzw. ausgewählten Elemente jedoch die entscheidende Rolle.

Liegen überlappende Intervalle vor, so würde die effektivste Vorgehensweise darin bestehen, die Intervalle dieser Elemente neu und genauer zu bestimmen, wobei Intervalle mit einem kleineren Durchmesser erzeugt werden, die zu keiner Überlappung führen. Eine solche Möglichkeit besteht z.B. dann, wenn die Bewertung durch eine externe Messung gewonnen wird, die nochmals bzw. mehrmals durchgeführt werden kann. Wird die Bewertung durch einen stochastischen Prozess bestimmt, so ist eine mehrmalige Durchführung der Bewertung nicht optional sondern notwendig. Mit Hilfe des Mittelwertes E(x) und der Varianz var(x) lässt sich ein Konfidenzintervall KV(x) = [E(x) - √var(x), E(x) + √var(x)] erzeugen, auf das Operationen der Intervallarithmetik angewendet werden können.

Wird die Bewertung durch eine numerische Approximation gewonnen, die Abhängig ist von der Anzahl von Approximationsstützpunkten, so kann durch Erhöhung der Stützpunktanzahl eine bessere Approximation und somit ein schmaleres Bewertungsintervall erzeugt werden. Analoges gilt bei der Anwendung eines Resampling-Verfahrens im Rahmen einer Bewertung, da mit der Erhöhung der Anzahl der Bootstrap-Einzelschätzungen die Approximation der Bewertung ebenfalls besser wird. Sind genügend Einzelschätzungen erzeugt worden, so kann ein Bootstrap-Konfidenzintervall berechnet oder approximiert werden (Efron & Tibshirani (1993[105])).

Bei allen Methoden, die mehrere Intervalle pro Eigenschaft x_i liefern, kann die Durchschnittseigenschaft der Intervallarithmetik herangezogen werden (Bauch et al. (1987:30[30])). Wird ein wahrer Wert bei einer Messung oder Berechnung in einem Intervall $I_f(x_i)_1$ lokalisiert, und bei einer weiteren Messung in $I_f(x_i)_2$, so bedeutet dies, dass der wahre Wert in der Schnittmenge der beiden Intervalle, d.h. in $I_f(x_i)_1 \cap I_f(x_i)_2$ liegen muss. Liegen Einzelintervalle $I_f(x_i)_k$, k = 1, ... vor, so ergibt sich das aggregierte Intervall zu:

$$I_f(x_i) = \bigcap_k I_f(x_i)_k. \tag{219}$$

Dieses aggregierte Intervall kann durch mehrmaliges Messen oder Berechnen nie größer werden, sondern höchstens kleiner. Dies führt jedoch nicht notwendig zu einer Verbesserung, da im Fall eines Intervalls, das innerhalb aller anderen bekannten Intervalle liegt, durch die Aggregation keine Verbesserung gegenüber diesem Intervall eintritt.

2.6.1) Rangfolge durch Ordnen nach einem ausgewählten Punkt im Intervall

Zunächst können die Elemente m_j aus KM auf einer Bewertungsskala angeordnet werden (siehe Abb. 22) mit 20 Elementen), wobei die Dreiteilung der Skala nur der besseren Übersicht dient.

Dadurch liegt noch keine Rangordnung vor, da unterschiedliche Merkmale der Intervalle zu unterschiedlichen Rangfolgen führen, wobei Merkmale der Intervalle ausgewählte Punkte des Intervalls sind, wie die untere oder obere Grenze, oder der Intervallmittelpunkt. In Abb. 23) wird die untere Grenze als Ordnungskriterium verwendet.

Abb. 22) Anordnung von Elementen auf einer Bewertungsskala

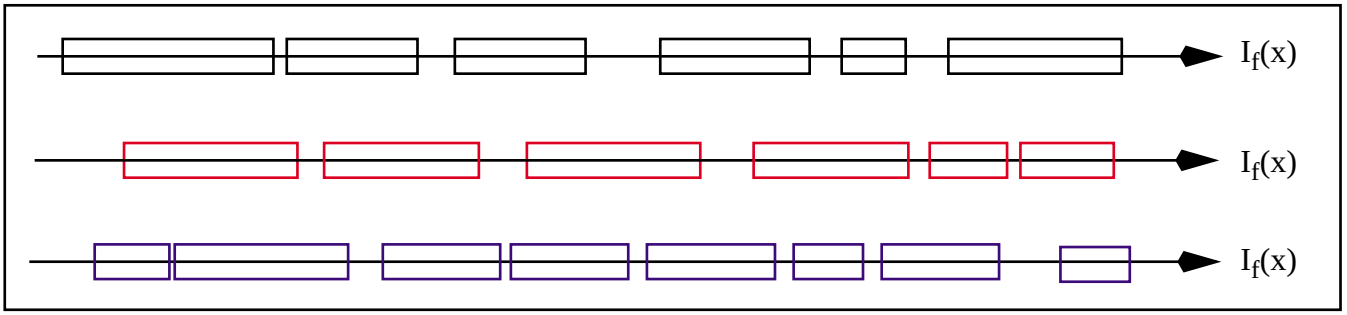
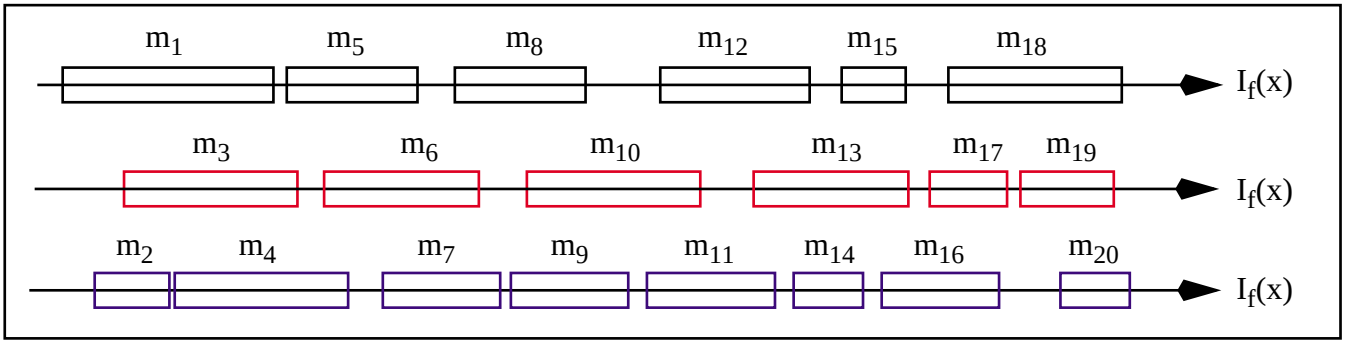
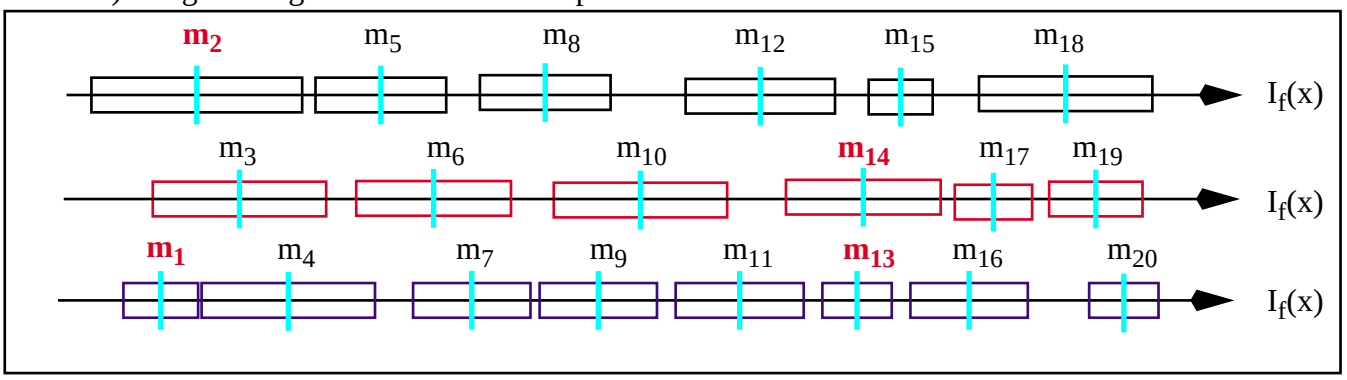


Abb. 23) Rangplätze durch untere Intervallgrenze



In Abb. 24) wird der Intervallmittelpunkt verwendet, wobei sich zeigt, dass bei überlappenden Bewertungsintervallen unterschiedliche Rangfolgen erzeugt werden. Im dargestellten Beispiel tauschen die beiden ersten Plätze sowie die Plätze 13 und 14 ihre Position.

Abb. 24) Rangordnung durch Intervallmittelpunkte



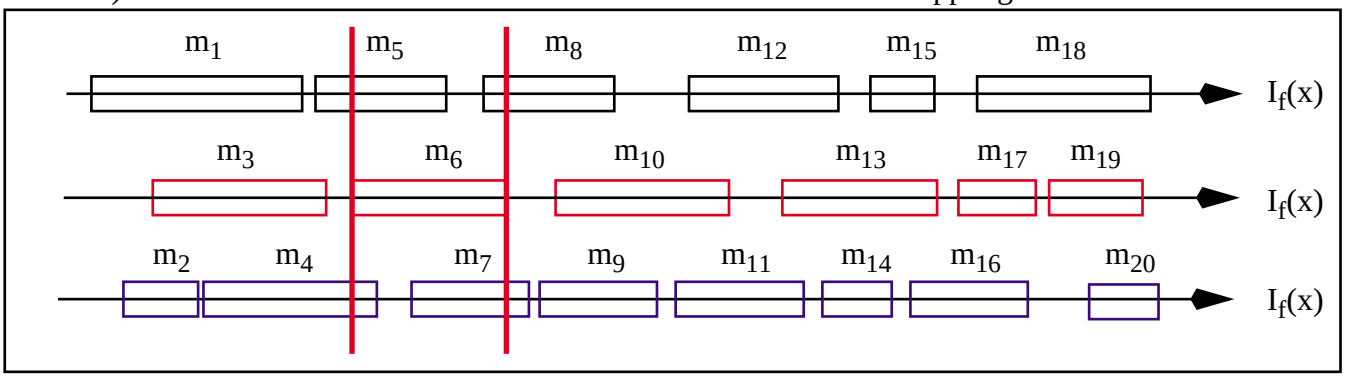
Problematisch werden Selektionsoperationen, wenn überlappende Intervalle vorliegen, wovon im Regelfall ausgegangen werden soll. Sollen z.B. die besten 6 Elemente bei einer Minimierung ausgewählt werden (siehe Abb. 25)), so zeigt sich, dass das sechte Element eine Überlappung mit dem 7'ten und 8'ten Element besitzt. Das Problem besteht darin, dass die richtige, jedoch unbekannte Bewertung der Elemente m_4 bis m_8 so liegen, dass eine Auswahl entsprechend dem Rang, der durch die untere Grenze festgelegt wurde, irreführend gegenüber dem Rang auf der Basis der richtigen Bewertungen sein kann.

Entsprechend der Zielvorgabe der Auswahl der 6 besten Elemente und der Erzeugung eines Rankings kann die Teilmenge KM_{6+} aller Elemente aus KM ermittelt werden, deren Intervall vollständig links von der unteren Grenze $\underline{f}(x_6)$ von $I_f(x_6)$ liegt, d.h. die obere Grenze der Elemente ist kleiner als die untere

Grenze von $I_f(x_6)$:

$$KM_{6+} = \{m_j \in KM \mid \bar{f}(x_j) < \underline{f}(x_6)\}. \quad (220)$$

Abb. 25) Auswahl von 6 aus 20 Elementen aus KM mit Intervall-Überlappungen



In Abb. 25) entspricht dies der Menge $KM_{6+} = \{m_1, m_2, m_3\}$. Weiterhin kann die Menge KM_{6-} gebildet werden, welche diejenigen Elemente enthält, deren Bewertungs-Intervall vollständig rechts von der oberen Grenze $\bar{f}(x_6)$ von $I_f(x_6)$ liegt, d.h. deren untere Grenze größer ist als die obere Grenze von $I_f(x_6)$:

$$KM_{6-} = \{m_j \in KM \mid \underline{f}(x_j) > \bar{f}(x_6)\}. \quad (221)$$

In Abb. 25) entspricht dies der Menge $KM_{6-} = \{m_9, ..., m_{20}\}$. Weiterhin kann die Menge $KM_{6\pm}$ aller Elemente gebildet werden, die mit $I_f(x_6)$ eine Überlappung besitzen, d.h. deren untere oder obere Grenze innerhalb von $I_f(x_6)$ liegt:

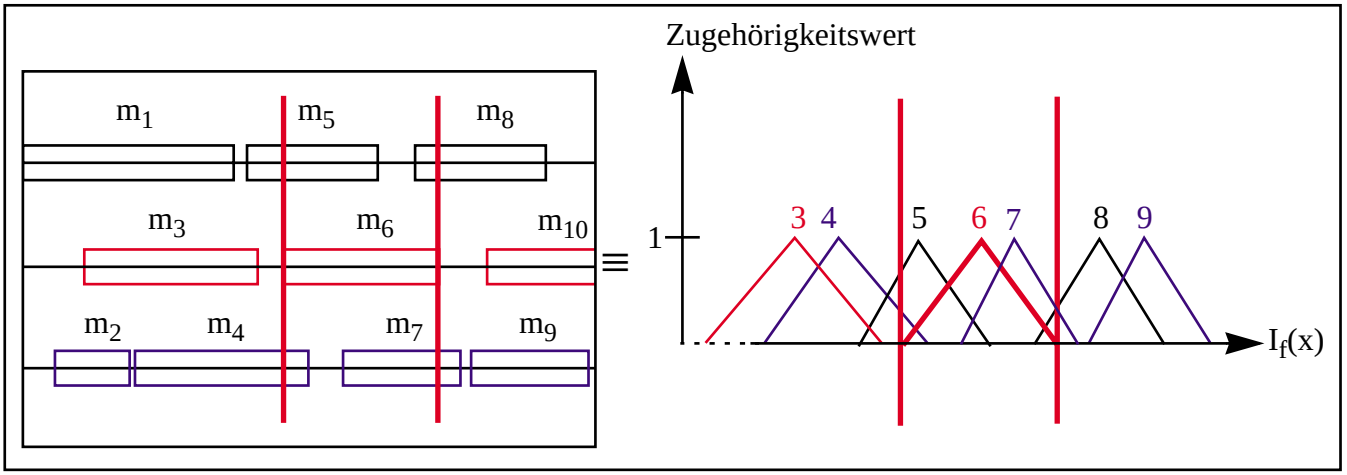
$$KM_{6\pm} = KM \setminus \{KM_{6+} \cup KM_{6-}\} = \{m_j \in KM \mid \underline{f}(x_j) \in I_f(x_6) \vee \bar{f}(x_j) \in I_f(x_6)\}. \quad (222)$$

Mit den drei Mengen KM_{6+}, KM_{6-} und $KM_{6\pm}$ können nun unterschiedliche Selektionsstrategien formuliert werden. Einfache Heuristiken wäre anwendbar, wenn die Angabe der Anzahl der auszuwählenden Elemente eine Soll- und keine Muss-Anzahl ist, d.h. wenn mehr oder weniger Elemente ausgewählt werden könnten. In disem Fall könnte KM_{6+} oder $KM_{6+} \cup KM_{6\pm}$ als Ergebnismenge verwendet werden. Wird die zusätzliche Aufgabe eines Rankings gestellt, so können die Elemente aus KM entsprechend der verwendeten Ordnung bezogen auf die Intervallgrenze oder den Intervallmittelpunkt übernommen werden.

2.6.2) Selektion durch Zugehörigkeitsfunktionen

Andere Entscheidungsverfahren bezüglich der Auswahl von Elementen aus $KM_{6\pm}$ lassen sich durch die Interpretation der Intervalle als Zugehörigkeitsfunktionen herleiten. Die sinnvollste Zuordnung ist eine dreieckige Zugehörigkeitsfunktion mit dem Intervallmittelpunkt als Zugehörigkeitswert 1 und den Intervallgrenzen als Zugehörigkeitswert 0, wobei rechts der oberen und links der unteren Intervallgrenze der konstante Zugehörigkeitswert 0 vorliegen soll (siehe Abb. 26)). Die Entscheidung wird in Verbindung mit einem der bekannten Fuzzy-Aggregations- bzw. -Inferenz-Mechanismen stattfindet (Höhle & Rodabaugh (1999[166]), Novak & Perfilieva (1999[239]), Nguyen (1999[234])), worauf jedoch nicht detaillierter eingegangen werden soll.

Abb. 26) Interpretation von Intervallen durch Zugehörigkeitsfunktionen



2.6.3) Dominanzfunktion auf der Basis von Intervallen

Bei einer Wettkampf-Selektion und insbesondere bei der Erzeugung einer Wettkampf-Hierarchie muss eine Dominanzfunktion auf der Basis der Bewertungsintervalle vorliegen, die zwei Intervalle wie $f(I_f(x_i)$ und $I_f(x_j)$ als Input aufnimmt, und ein Skalar als Output liefert. Es wurde gezeigt, dass die Intervallarithmetik eine Arithmetik über einer drei-wertigen Mengenlehre ist (Bauch et al. (1987: 33f[30])), sodass die Dominanzfunktion zwei Intervalle in ihrer einfachsten Form dargestellt werden kann als:

$$dom(I_f(x_i), I_f(x_j)) \in \{-1, 0, 1\}. \quad (223)$$

Dies lässt sich spezifizieren, indem die obere Grenze $\bar{f}(x_i)$ von $I_f(x_i)$ und die untere Grenze $\underline{f}(x_j)$ von $I_f(x_j)$ verglichen wird:

$$dom(I_f(x_i), I_f(x_j)) = \begin{cases} 1, & \text{wenn } \bar{f}(x_i) \leq \underline{f}(x_j), \\ -1, & \text{wenn } \bar{f}(x_i) \geq \underline{f}(x_j), \\ 0, & \text{sonst.} \end{cases} \quad (224)$$

Diese Dominanzfunktion beschreibt somit in den beiden Fällen $dom(I_f(x_i), I_f(x_j)) = \{1, -1\}$ zwei überlappungsfreie Intervalle, denen eine eindeutige Dominanz zugeordnet werden kann, und den Fall der nicht-dominanz mit $dom(I_f(x_i), I_f(x_j)) = 0$, wenn zwei sich überlappende Intervalle vorliegen.

In Bauch et al. (1987: 33f[30]) werden die beiden Relationen „$\leq_+$: stark kleiner-gleich“ und „$\leq_{++}$: schwach kleiner-gleich“ eingeführt, mit:

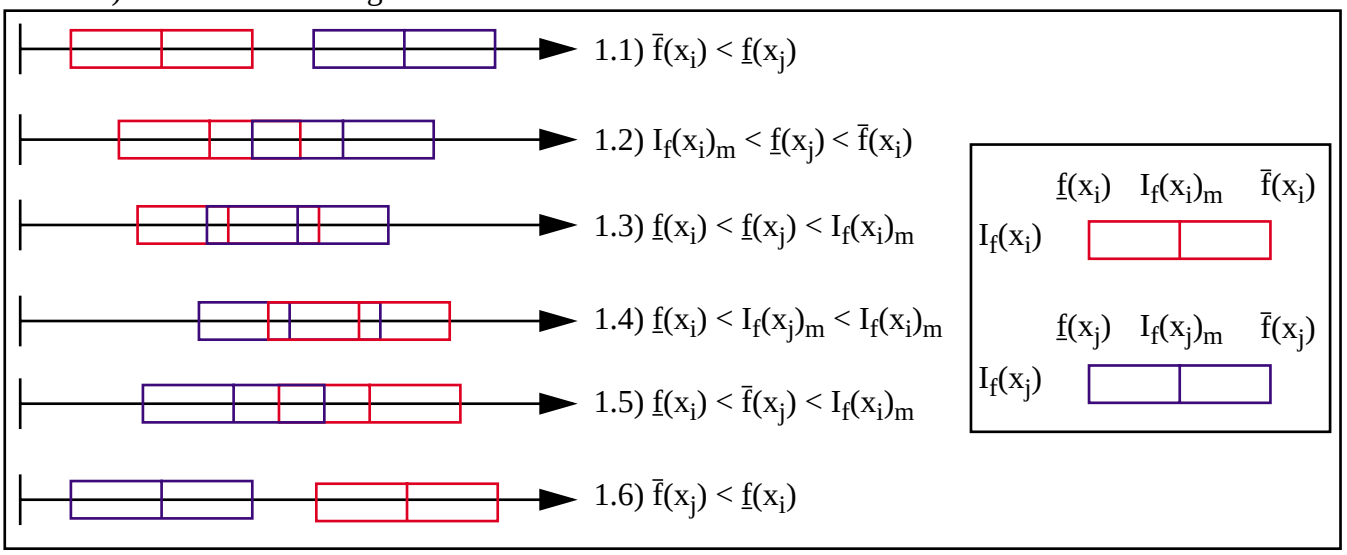
$$I_f(x_i) \leq_+ I_f(x_j) <=> \bar{f}(x_i) \leq \underline{f}(x_j);$$
$$I_f(x_i) \leq_{++} I_f(x_j) <=> \underline{f}(x_i) \leq \bar{f}(x_j). \quad (225)$$

Mit Hilfe der Relation „stark kleiner-gleich“ lässt sich die erste Dominanzfunktion reformulieren zu:

$$dom(I_f(x_i), I_f(x_j)) = \begin{cases} 1, & \text{wenn } I_f(x_i) \leq_+ I_f(x_j), \\ -1, & \text{wenn } I_f(x_i) \geq_+ I_f(x_j), \\ 0, & \text{sonst.} \end{cases} \quad (226)$$

Berücksichtigt man die Unter- und die Obergrenze, sowie den Mittelpunkt eines Intervalls, so können zwei Intervalle $I_f(x_i) = [\underline{f}(x_i), \bar{f}(x_i)]$ und $I_f(x_j) = [\underline{f}(x_j), \bar{f}(x_j)]$ gleicher Breite, d.h. $\bar{f}(x_i) - \underline{f}(x_i) = \bar{f}(x_j) - \underline{f}(x_j)$, sechs Positionen zueinander einnehmen (siehe Abb. 27)).

Abb. 27) Positionen zweier gleich-breiter Intervalle zueinander



In Tabelle 1) werden zwei mögliche Dominanzfunktionen angegeben, welche die unterschiedlichen Situationen bewerten, wobei die erste Dominanzfunktion dem Paretokriterium entspricht, d.h. eine beliebige Überdeckung der beiden Intervalle führt zu einer Bewertnung „nicht-dominant“, die mit „0“ kodiert ist.

Tabelle 1) Position zweier Intervalle bei Minimierung und mögliche Bewertungsfunktionen

$\bar{f}(x_i) - \underline{f}(x_i) = \bar{f}(x_j) - \underline{f}(x_j)$	Dominanzfunktion 1	Dominanzfunktion 2
$\bar{f}(x_i) < \underline{f}(x_j)$	$dom(I_f(x_i),I_f(x_j)) = 1$	$dom(I_f(x_i),I_f(x_j)) = 1$
$I_f(x_i)_m < \underline{f}(x_j) < \bar{f}(x_i)$	$dom(I_f(x_i),I_f(x_j)) = 0$	$dom(I_f(x_i),I_f(x_j)) = 2/3$
$\underline{f}(x_i) < \underline{f}(x_j) < I_f(x_i)_m$	$dom(I_f(x_i),I_f(x_j)) = 0$	$dom(I_f(x_i),I_f(x_j)) = 1/3$
$\underline{f}(x_i) < \underline{f}(x_j) < I_f(x_i)_m$	$dom(I_f(x_i),I_f(x_j)) = 0$	$dom(I_f(x_i),I_f(x_j)) = 0$
$\underline{f}(x_i) < I_f(x_j)_m < I_f(x_i)_m$	$dom(I_f(x_i),I_f(x_j)) = 0$	$dom(I_f(x_i),I_f(x_j)) = -1/3$
$\underline{f}(x_i) < \bar{f}(x_j) < I_f(x_i)_m$	$dom(I_f(x_i),I_f(x_j)) = 0$	$dom(I_f(x_i),I_f(x_j)) = -2/3$
$\bar{f}(x_j) < \underline{f}(x_i)$	$dom(I_f(x_i),I_f(x_j)) = -1$	$dom(I_f(x_i),I_f(x_j)) = -1$

Die in der zweiten Dominanzfunktion dargestellte Vorgehensweise lässt sich verallgemeinern, um differenziertere Dominanzbewertungen zu erhalten, indem die Intervalle nicht einmal, sondern durch eine ungerade Anzahl von Unterteilungen zerlegt werden. Bei 2n-1 Unterteilungen ergeben sich 2n Intervallabschnitte, deren Lage zueinander beschrieben werden können.

Der Normalfall bei einem Vergleich von Intervallen ist das Vorliegen von abweichenden Intervalldurchmessern und Mittelpunkten. Im Rahmen der Intervallarithmetik gilt, dass Intervalle mit einem kleineren Durchmesser vorzuziehen sind. Wird eine Minimierung angenommen, so sind Intervalle besser, die einen kleineren Referenzpunkt wie z.B. die Intervalluntergrenze oder den Mittelpunkt besitzen. Allgemein kann der Vergleich zweier Intervalle im Kontext einer 2-Ziel-Minimierungsproblem interpretiert werden, mit den äquivalenten Formulierungen:
1) Minimierung der Untergrenze und Minimierung des Durchmessers.
2) Minimierung der Untergrenze und Minimierung der Obergrenze.

Ist es möglich, eine Distanzmetrik für Intervalle zu definieren, so lassen sich Elemente aus KM entsprechend ihren Bewertungsintervallen durch die Distanz zum Ursprung des Intervallraumes sortieren, sodass die Metrik die Funktion einer 1-Ziel-Minimierung übernimmt. Als Beispiel sei die Intervallmetrik dargestellt, die auf der Norm von Intervallvektoren I(x) basieren (Bauch et al. (1987: 23[30])). Sei

$$I_f(x) = (I_f(x_i) = [\underline{f}(x_i), \bar{f}(x_i)] \mid i = 1, ..., n) \in I^n \tag{227}$$

ein Intervallvektor, so berechnet sich die Norm $|I_f(x)|$ des Intervallvektors $I_f(x)$ als Supremum über allen Punktvektornormen $|x_i|$, die innerhalb $I_f(x)$ liegen:

$$|I_f(x)| = \sup\{|x_i| \mid x_i \in I_f(x)\}. \tag{228}$$

Wird eine Distanz als Norm, und wird $I_f(x)$ als Hyperquader interpretiert, so ist die Norm eines Intervallvektors gleich dem größten Abstand zweier Punkte in dem Hyperquader, d.h. einer Diagonallänge.

Eine Distanzmetrik $d_I(I_f(x)_1, I_f(x)_2)$ zwischen zwei Intervallvektoren $I_f(x)_1, I_f(x)_2 \in I^n$

$$\begin{aligned} I_f(x)_1 &= (I_f(x_i)_1 = [\underline{f}(x_i)_1, \bar{f}(x_i)_1] \mid i = 1, ...,n; \\ I_f(x)_2 &= (I_f(x_i)_2 = [\underline{f}(x_i)_2, \bar{f}(x_i)_2] \mid i = 1, ..., n \end{aligned} \tag{229}$$

lässt sich definieren durch den Betrag der Differenzen korrespondierender unterer und oberer Grenzen. D.h. für jede der n Komponenten der Vektoren wird die Differenz $\underline{f}(x_i)_1 - \underline{f}(x_i)_2$ zwischen den beiden unteren Grenzen und den beiden oberen Grenzen $\bar{f}(x_i)_1 - \bar{f}(x_i)_2$ gebildet. Der Betrag aus den beiden Differenzen liefert die beiden Werte $|\underline{f}(x_i)_1 - \underline{f}(x_i)_2|$ und $|\bar{f}(x_i)_1 - \bar{f}(x_i)_2|$, gefolgt von der Bildung des Maximums aus den beiden Werten bezüglich der Komponente i der beiden Vektoren. Die Distanz zwischen den beiden Intervallvektoren wird als Maximum dieser Werte über alle n Komponenten gebildet (Bauch et al. (1987: 23[30])):

$$d_I(I_f(x)_1, I_f(x)_2) = \max\{\max\{|\underline{f}(x_i)_1 - \underline{f}(x_i)_2|, |\bar{f}(x_i)_1 - \bar{f}(x_i)_2|\} \mid i = 1, ..., n\}. \tag{230}$$

3) Mono- und Polyrepräsentation im vektorraumbasierten Information-Retrieval

In diesem Kapitel wird dargestellt, was Polyrepräsentation innerhalb eines einzelnen Paradigmas wie der Verwendung eines Vektorraummodells bedeuten kann, d.h. es wird die Intraparadigmen-Polyrepräsentation im Bezug zum Vektorraummodell betrachtet. Auf die Darstellung von Information-Filter-Systeme wird verzichtet, da diese als komplementäre Struktur zum Information-Retrieval betrachtet werden (Belkin & Croft (1992[36]), Goldberg et al. (1992[147]), Panyr (1994[253]), Allan (1996[5]), Mostafa et al. (1997[227]), Schapire et al. (1998[299])), sodass eine Intraparadigmen-Polyrepräsentation für Information-Filter-Systeme in ähnlicher Weise abgeleitet werden können.

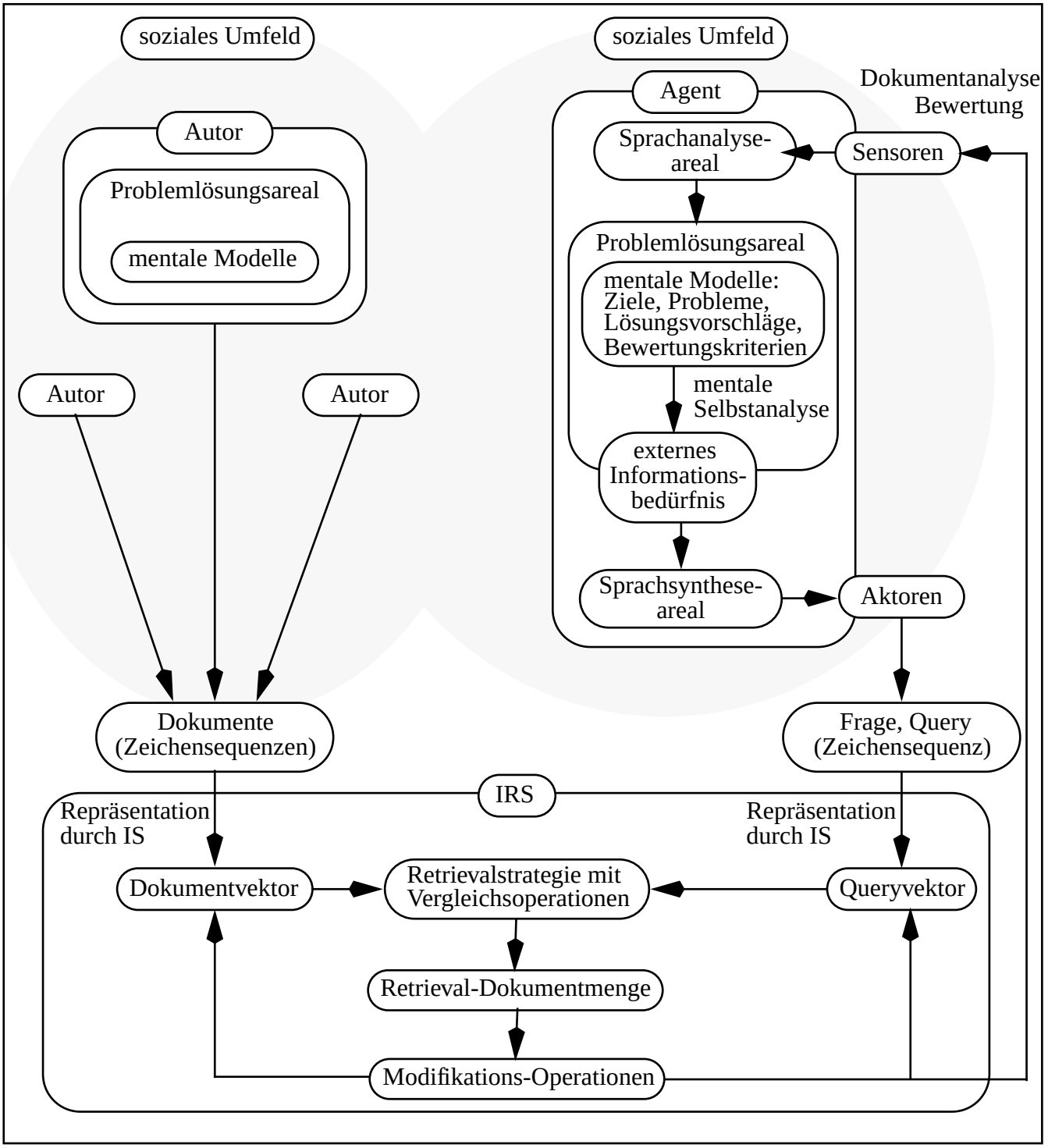
3.1) Information Retrieval Systeme

Belkin & Croft (1992[36]) haben ein allgemeines Modell des Information-Retrievals und des Information-Filterings entworfen, das im Rahmen dieses Kapitels verwendet und um kognitive bzw. neuro-kognitive Komponenten erweitert werden soll. Diese Erweiterung betreffen mentale Prozesse im Nutzer bzw. Agenten, die auch für die Diskussion der Relevanzproblematik zentral sind (siehe auch Abschnitt 3.9.1)). Der Agent wird als kognitiver Agent modelliert, d.h. er besitzt eine interne kognitive Struktur, die bei einer neuro-kognitiven Sichtweise mit unterschiedlichen Hirnarealen korreliert, die jeweils verschiedene kognitive Funktionen erfüllen, mit jeweils eigenen Repräsentationsformaten und kognitive Operationen. Beachtet werden muss jedoch, dass die internen Zustände des Agenten für das Information-Retrieval-System (IRS) nicht direkt zugänglich sind, sondern nur über das Verhalten des Agenten, das er über Aktoren abgibt, z.B. die Bedienung einer Tastatur oder akustische Sprachäusserungen.

Betrachtet wird ein Sprachanalyseareal, ein Problemlösungsareal und ein Sprachsyntheseareal (siehe Abb. 28) und Belkin & Croft (1992:31[36])). Es existieren gerichtete neuronale Verbindungen zwischen diesen Hirnarealen, d.h. ein Areal A projiziert in ein Areal B, ohne dass auf den gleichen Verbindungen eine Rückprojektion stattfindet, wobei eine solche Projektion nicht nur eine Weiterleitung von Daten ist, sondern als eine Transformation verstanden werden muss, welche Repräsentationsformate verändert. Eine solche Transformation kann als nicht umkehrbare Funktion verstanden werden, da verschiedene Zustände auf einen Zielzustand abgebildet werden können (siehe auch Darstellungen im Abschnitt 1.2.4)).

Da das Information-Retrieval im Kontext eines Problemlösungsprozesses behandelt werden soll, sind die kognitiven Modelle und Operationen im Problemlösungsareal von besonderer Bedeutung. Für einen Problemlösungsprozess muss eine Repräsentation des Gesamtproblems (bzw. mehrere alternative Repräsentationen bei einer mentalen Polyrepräsentation) vorliegen. Handelt es sich um ein komplexes Problem, so kann es nur bearbeitet werden, wenn das Gesamtproblem in eine Struktur von Teilproblemen zerlegt wird, wobei die Struktur sowie die Anzahl und Beschreibung der Teilprobleme im Prozess der Problemlösung ständig modifiziert wird. Die einfachste Struktur ist eine lineare Anordnung von Teilproblemen bzw. den korrespondierenden Zielen, das jeweilige Teilproblem zu lösen, was im Rahmen kognitiver Modelle auch als Goal-Stack bezeichnet wird (Anderson (1990[8], 1993[10]), Newell (1990[233])).

Abb. 28) Allgemeines Modell des Information-Retrieval



Neben Repräsentationen des Problems sollen auch Repräsentationen von Problemlösungsvorschlägen zu Teilproblemen innerhalb des Problemlösungsareals enthalten sein. Weiterhin müssen Bewertungs- und Entscheidungskriterien vorhanden sein, mit denen die Güte eines Lösungsvorschlages bewertet werden kann. Je nach der Bewertung wird entschieden, ob ein Vorschlag als Lösung für ein Teilproblem akzeptiert wird, sodass die Struktur der verbleibenden Teilprobleme entsprechend verändert wird, wie z.B. die Löschung des obersten Zieles in einem Goal-Stack, wenn dieses Ziel erreicht wurde. Die individuellen Bewertungs- und Entscheidungskriterien sind bezüglich des Agenten subjektiv, d.h. es besteht die Möglichkeit, dass durch eine externe Instanz aus dem sozialen Kontext des Agenten eine subjektiv akzeptierte

Lösung überschrieben wird, was der Situation entspricht, dass der Agent eine Lösungsvorschlag präsentiert, der von einer externen Instanz z.B. einer Arbeitsgruppe nicht in dieser Form akzeptiert wird.

Im Repertoire der kognitiven Operationen des Problemlösungsareals muss eine Meta-Operation existieren (Selbstanalyse), die entscheidet, ob ein Problemlösungsprozess durch interne Operationen weitergeführt werden soll, oder ob externe Informationen notwendig werden, um die mentalen Strukturen weiterzuentwickeln (siehe Darstellungen zur Einbettung externer Informationsbeschaffung in ein Modell der allgemeinen Intelligenz im Abschnitt 1.5)). Würde eine solche Meta-Operation nicht existieren, so würden immer weitere, interne Problemlösungsversuche durchgeführt, ohne dass es zu einer Formulierung eines externen Informationsbedürfnisses kommen würde. Die Formulierung eines externen Informationsbedürfnisses in Form einer Frage/Query ist eine Transformation von mentalen Zuständen aus dem Problemlösungsareal in sprachliche Konstrukte, die als Synthese-Transformation bezeichnet werden soll. Sie ist eine wesentliche Quelle von Unsicherheit und Unschärfe auf der Seite des Agenten, was sich daraus ergibt, dass eine Projektion aus dem Problemlösungsareal in das Sprachsyntheseareal notwendig ist, bei der Repräsentationsformate überführt werden. Die inhaltliche Abbildung ist von den individuell verfügbaren Sprachkonstrukten des Agenten abhängig, d.h. von seinem aktiven Wortschatz und den individuellen semantischen Beziehungen zwischen den vorhandenen Begriffen, d.h. von einem individuellen, semantischen Assoziationsnetz. Eine mentale Struktur, die auf zwei unterschiedliche Assoziationsnetze abgebildet wird, ergibt zwei unterschiedliche Queries, die von einem IRS verarbeitet werden müssen.

Liegt eine Aktivierung des Sprachsyntheseareals vor, so werden damit Aktoren wie Hand- und Fingerbewegungen beim Schreiben geregelt. Diese Aufschlüsselung ist notwendig um eine explizit vorliegende Zeichensequenz zu erhalten, die als Input in das IRS verwendet werden kann, da interne Zustände, unabhängig ob sie im Problemlösungs- oder im Sprachsyntheseareal vorliegen, vom IRS nicht einsehbar sind. Die sich ergebende Zeichensequenz wird im Standardfall des Information-Retrieval als Frage (Query) interpretiert, wenn es sich um eine sprachliche Übertragung einer mentalen Struktur handelt, die das externe Informationsbedürfnis repräsentiert. Es sind auch andere Textformen übertragbar wie Problembeschreibungen, Lösungsvorschläge u.a., was im Abschnitt 3.9.7) im Zusammenhang mit der Reformulierungs-Relevanz betrachtet wird.

Die vom Agenten erzeugte Zeichensequenz, unabhängig von der Textart, wird als Input für das IRS verwendet, das die Zeichensequenz indexiert, und sie somit in einen metrischen Vektorraum abbildet, was als Indexierung bzw. Indexierungsfunktion bezeichnet wird. Auf diese Weise wird standardmäßig ein Queryvektor erzeugt, wenn es sich bei dem Texttyp um eine Frage handelte.

Im Ansatz von Belkin & Croft (1992: 33[36]) stehen neben dem Agenten eine Anzahl von Autoren als Produzenten von Zeichensequenzen, die als Dokumente im IRS gespeichert und indexiert werden, d.h. ihnen wird ebenfalls ein Punkt in einem metrischen Vektorraum zugeordnet. Es wird davon ausgegangen, dass alle Zeichensequenzen in dem gleichen Vektorraum repräsentiert werden, sodass Vergleiche mit Hilfe der Metrik in dem Vektorraum möglich werden.

Die Autoren besitzen prinzipiell die gleichen mentalen Strukturen wie der Agent, doch liegt der Schwerpunkt beim Information-Retrieval auf der Interaktion zwischen Agent und IRS, und nicht auf der Textproduktion der Autoren mit einer eventuellen Interaktion mit dem IRS, sodass die entsprechenden mentalen Strukturen unspezifiziert bleiben sollen. Dies bedeutet auch, dass keine Interaktion zwischen Autor und Agent mit Hilfe des IRS als Vermittler zustande kommt bzw. dass dies nicht schwerpunktmäßig vom Information-Retrieval behandelt wird.

Die Vergleichsoperationen der Vektoren der beiden Herkunftsarten ist Teil der Retrievalstrategie des IRS, die eine Menge von Dokumentvektoren liefert, wobei die korrespondierenden Dokumente die Retrieval-Dokumentenmenge ist, die als Output auf den Query-Input vom IRS geliefert und dem Agenten präsentiert wird, womit eine Interaktionsiteration eines Information-Prozesses beendet ist. Ob weitere folgen ist abhängig von der Bewertung der Ergebnismenge durch den Agenten und davon, ob eine Relevanz-Feedback-Komponente innerhalb des IRS existiert.

Die präsentierten Dokumente werden vom Agenten analysiert, wobei spezielle Textteile wie Abstract und Zusammenfassung bei einem strukturierten Text und bei zeitlichen Constraints zuerst analysiert werden (On-Line-Analyse). Liegen keine Zeit-Constraints vor, so kann der Agent die Dokumente ausführlich analysieren, wozu er entsprechende Zeit benötigt, sodass eine Interaktionsunterbrechung mit dem IRS sinnvoll ist (Off-Line-Analyse), wobei der Stand der Interaktion vom IRS bis zum nächsten Interaktionsschritt zwischengespeichert wird.

Bei der Dokumentanaylse sind zunächst die Sensoren und das Sprachanalyseareal beteiligt, gefolgt von der Integration des Inhaltes, der dem Agenten zugänglich ist, in seine mentalen Strukturen im Problemlösungsareal. Die Umkehrung der Transformationsrichtung wird im Rahmen der Relevanzbewertung von Dokumenten notwendig, indem zunächst Dokumentteile wie der Titel, das Abstract oder die Zusammenfassung sprachlich analysiert werden müssen, was in anderen Hirnarealen als die Sprachproduktion durchgeführt wird. Durch die Trennung der Areale für Sprachanalyse und Sprachsynthese kann das Textverstehen nicht als Umkehrung der Textformulierung verstanden werden, was sich allein in der Existenz eines passiven und aktiven Wortschatzes zeigt, die sich erheblich unterscheiden. Somit ist die Transformation von kognitiven Zuständen aus dem Problemlösungsareal in sprachliche Konstrukte des Sprachsyntheseareals keine Umkehrfunktion der Transformation von sprachlichen Konstrukten aus dem Sprachanalyseareal in mentale Zustände des Problemlösungsareals (Analyse-Transformation). Genau wie bei der Synthese-Transformation ist die Analyse-Transformation eine weitere wichtige Quelle von Unsicherheit und Unschärfe. Zum einen kann es zu einem fehlerhaften Verstehen von Textinhalten im Rahmen der Spachanalyse kommen, zum anderen besteht die Möglichkeit, dass die extern ermittelten Informationen fehlerhaft in die mentalen Strukturen integriert werden, die im Problemlösungsareal repräsentiert sind, bzw. überhaupt nicht integriert werden können.

Tragen die integrierten Informationsobjekte dazu bei, Teilprobleme zu lösen, so wird die Struktur der Teilprobleme verändert. Weiterhin können Problemlösungsvorschläge neu bewertet, verworfen, oder modifiziert werden. Wird das momentane, externe Informationsbedürfnis durch diese mentalen Operationen befriedigt, so finden keine weiteren Interaktionen mit dem IRS statt.

Ist der Agent noch nicht zufrieden oder ergeben sich andere externe Informationsbedürfnisse, so wird die Interaktion fortgesetzt, wobei folgende Alternativen standardmäßig betrachtet werden:
1) Reformulierung der Query durch den Agenten, bezogen auf das gleiche Informationsbedürfnis.
2) Relevanzbewertung der präsentierten Informationsobjekte durch den Agenten bei einem Relevanz-Feedback-Verfahren, bezogen auf das gleiche Informationsbedürfnis.
3) Formulierung einer neuen Query, bezogen auf ein neues Informationsbedürfnis.

Im Rahmen eines Relevanz-Feedback-Verfahren werden aus den Relevanzbewertungen des Agenten und mit Hilfe von Modifikationsoperatoren verschiedene Repräsentationen des IRS verändert, wobei meist die Queryvektor- und die Dokumentvektor-Modifikation betrachtet wird. Im Rahmen der Darstellungen zum Relevanz-Feedback in diesem Kapitel werden Vorschläge zur Modifikation weiterer Repräsentationsformen innerhalb des IRS gemacht, die sich nicht an anderer Stelle finden (siehe Abschnitt 3.9)).

3.2) Dokument als Zeichensequenz

3.2.1) Dokument-Monorepräsentation

Ein Dokument D_i soll im Rahmen dieser Arbeit allgemein als endliche Sequenz von Zeichen d_i aus einem endlichen Alphabet Θ verstanden werden:

$$\Theta = (d_k \mid k = 1, ..., |\Theta| \in N). \tag{231}$$

Die Menge aller Dokumente, die mit Hilfe des Alphabetes Θ gebildet werden können, und die eine Sequenzlänge von $n \in N$ besitzen, soll mit $D(\Theta \mid n)$ bezeichnet werden. Im Rahmen von IRS müssen Dokumente repräsentiert werden, die sehr unterschiedliche Sequenzlängen besitzen, sodass es sinnvoll ist, die Menge aller Dokumente mit einer Sequenzlänge von kleiner-gleich n Zeichen zu definieren, die mit $D(\Theta \mid \leq n)$ bezeichnet werden soll. Ein Dokument D_i mit der Sequenzlänge $\#D_i$ wird als Element der Dokumentmenge $D(\Theta \mid \leq n)$ bezeichnet, wenn es als Zeichensequenz wie folgt beschrieben werden kann:

$$D_i = (d_{ij} \mid \forall d_{ij} \in \Theta, j = 1, ..., \#D_i \leq n) \in D(\Theta \mid \leq n). \tag{232}$$

Ist die Dokumentenlänge irrelevant, z.B. weil im Rahmen der Indexierung der Dokumentvektor normiert wird, d.h. Dokumente mit sehr unterschiedlichen Längen werden in dem gleichen Dokument-Vektorraum repräsentiert, so kann verallgemeinert die Menge aller möglichen Dokumente bezogen auf das Alphabet Θ mit $D(\Theta)$ bezeichnet werden.

Entsprechend dieser allgemeinen Definition sind nicht nur natürlich-sprachliche Texte, sondern auch andere „Codes“ wie Byte-Code oder genetischer Code eingeschlossen, auf die verschiedene Operationen wie Speichern, Indexieren, Clustern, Retrieval, u.ä. angewendet werden können. Dass Indexierung und Retrieval von genetischem Code innerhalb Gen-Datenbanken sinnvoll eingesetzt werden kann, hat z.B. Williams (1998[360]) gezeigt. In dem Maße, wie biologische „Texte“ in Form von Genen verfügbar werden, werden auch entsprechende IS an Bedeutung gewinnen, wobei Indexierungs-, Cluster- und Retrieval-Funktionen aus IRS hierbei Anwendung finden werden, da viele der biologischen Fragestellungen probabilistischer Natur sind, und somit dem IR näher sind als dem Fakten-Retrieval, entsprechend der Abgrenzung von Fakten-Retrieval und IR (siehe Rijsbergen (1979[279])).

Was eine Zeichensequenz zum Code macht, ist die Existenz eines Interpreter-Systems oder eines Automaten, der Zeichen oder Teilsequenzen als Befehle interpretiert, was im Zusammenhang mit einer kurzen Darstellung von Automaten beschrieben werden soll (siehe z.B. Siegelmann (1999: 4ff[310])). Das mathematische Objekt „Automat“ oder „Maschine“M wird definiert als 5-Tupel:

$$M = (Q, I, Y, f, h), \text{ mit} \tag{233}$$

- Q als der Menge aller Zustände des Automaten (Zustandsraum),
- I als der Menge aller möglichen Inputs (Inputraum),
- Y als der Menge aller möglichen Outputs (Outputraum),
- f als der „Next-State“-Funktion: $f: Q \times Q \rightarrow Q$,
- h als der Output-Funktion: $h: Q \rightarrow Y$.

Ein Dokument als Zeichensequenz wird als Input in einen geeigneten Automaten interpretiert, der damit seine inneren Zustände verändert, und einen Output oder eine Sequenz von Outputs erzeugt. Beim genetischen Code werden DNA-Sequenzen, die als eigene „Dokumente“ durch Start- und Stop-Zeichen gekennzeichnet sind, durch eine Transkriptionsfunktion h in eine Aminosäuresequenz umgesetzt, die sich durch Faltungsprozesse in Proteine selbstorganisierend entwickelt. Natürlich-sprachliche Texte werden in diesem Kontext als Programme für individuelle, mentale Automaten, d.h. für Subjekte, interpretiert, die ihren inneren Zustand durch die Verarbeitung und Interpretation von Texten verändern, indem mentale Modelle verändert oder aufgebaut werden, und die einen Output in Form von Verhalten generieren, der sprachlich oder haptisch sein kann.

3.2.2) Dokument-Polyrepräsentation

In dem vorangegangenen Abschnitt wurde ein Dokument D_i als genau eine Zeichensequenz $(d_{ij} \mid \forall d_{ij} \in \Theta; j = 1, ..., \#D_i)$ dargestellt, was dem Standardfall einer Mono-Repräsentation entspricht. Eine solche Darstellung entspricht dem Original-Dokument, da dieses unabhängig von einem IRS vorliegt, sodass man zunächst nicht von einer Repräsentation sprechen kann, da das Original-Dokument Teil des Definitionsbereiches der Abbildung darstellt, während das repräsentierte Dokument Teil des Wertebereiches der Abbildung ist.

Im Hinblick auf die Indexierungsfunktion und bezüglich der Zielvorstellung einer Polyrepräsentation kann es trotzdem sinnvoll sein, polyrepräsentierte Dokumente zu betrachten, wobei diese aus dem Original-Dokument erzeugt werden, und somit künstliche Dokumente darstellen. In diesem Fall von Polyrepräsentation zu sprechen ist konsistent, da das Original-Dokument auf ein künstliches Dokument durch eine Repräsentationsfunktion abgebildet wird.

Ziel ist hierbei die Erzeugung von künstlichen Dokumentlisten BD_k^t, die neben der Original-Dokumentliste D^t verwendet werden:

$$BD^t = \{D^t, BD_k^t \mid k = 1, ..., \delta\}. \tag{234}$$

Hierbei sind zwei Strategien anwendbar:
1) Erzeugung künstlicher Dokumentlisten durch Original-Dokumente.
2) Erzeugung künstlicher Dokumentlisten durch künstliche Dokumente.

Die erste Möglichkeit lässt sich durch Resamplingverfahren wie dem Paar-Bootstrap realisieren, indem aus D^t m^t mal ein Dokument mit Zurücklegen gezogen wird, um eine künstliche Dokumentliste $BD_k{}^t$ zu erzeugen. Diese Variante soll im weiteren jedoch nicht näher betrachtet werden, da die Erzeugung künstlicher Dokumente auch im Hinblick auf die spätere Erzeugung von künstlichen Queries ein weiter tragender Ansatz ist.

Künstliche Dokumente treten auch im Rahmen der Erstellung von automatischen Abstracts (siehe z.B. Borko & Bernier (1975[52]), Pfeiffer-Jäger (1980[257]), Mathis & Rush (1985[213]), Bernier (1985[44]), Lancaster (1991[195])), Extrakten oder Zusammenfassungen (Summarization; Mani & Maybury (1999[208])) auf (siehe z.B. Goldstein et al. (1999[148])). Im Rahmen der automatischen Zusammenfassung oder Extraktion werden Worten, Sätzen und Abschnitten eines Textes Gewichte zuzuordnen, die ihre Bedeutung innerhalb des Gesamttextes repräsentieren, analog zu den Termgewichtungen beim IR. Die Zuordnung von Gewichten kann mit den gleichen Mitteln der deskriptiven Statistik geschehen, wie bei den vektorraumbasierten IRS (siehe Abschnitt 3.5)). Nach der Zuordnung von Gewichten kann eine sukzessive Kondensation erfolgen, indem bei jedem Iterationsschritt der am wenigsten relevante Satz oder Abschnitt entfernt wird. Zusätzlich kann die Reihenfolge der jeweils vorhandenen Sätze verändert werden, indem eine Rangfolge der Sätze nach fallenden Gewichtungen gebildet wird. Auf diese Weise entsteht eine geordnete Liste künstlicher Dokumente, die aus dem Original-Dokument abgeleitet sind, und deren Dokumentlänge mit zunehmender Position in der Liste abnimmt, bis im Extrem das kleinste künstliche Dokument aus genau einem Satz besteht bzw. aus einer Liste von Termen (Ultra-Summarization).

Bei allen künstlichen Dokumenten soll die Bedeutung des Textes im Vergleich mit dem Original-Dokument weitgehend erhalten bleiben. Die Anforderung der weitgehenden Bedeutungserhaltung kann im Rahmen vektorraumbasierten IRS spezifiziert werden, indem gefordert wird, dass die Dokumentvektoren der einzelnen künstlichen Dokumente im Dokumentraum DR möglichst nahe beieinander liegen sollen.

Ein künstliches Dokument $D_{ik}{}^t$, das aus $D_i{}^t$ erzeugt wurde, soll in dem Kontext der Polyrepräsentation eine Bedeutungsvariante bzw. eine abweichende Bedeutungsfacette darstellen, die mit statistischen Mitteln erzeugt und analysiert werden soll. Es wird vorgeschlagen, diese Repräsentation mit Hilfe des Moving-Blocks-Bootstrap durchzuführen (Efron & Tibshirani (1993:99ff[105])), wobei dieses Verfahren ursprünglich als Anwendung des Resamplingprinzips auf eine Zeitreihe konzipiert wurde (siehe Abschnitt 2.2.3)). Eine Zeitreihe wird jedoch als Zeichensequenz kodiert, wobei die sequentielle Ordnung durch die Zeit vorgegeben ist, sodass das Moving-Block-Verfahren auf beliebige Zeichensequenzen verallgemeinbar ist, so auch auf ein Original-Dokument als Zeichen-Sequenz. Es kann auch die umgekehrte Argumentation angeführt werden, bei der ein Text als Zeitreihe von Zeichen interpretiert wird, da bei der Textgenerierung wie bei der Textrezeption die zeitliche Komponente in der Sequenz der Wort und Satzstruktur kodiert ist. Wird ein Text als Zeitreihe interpretiert, so können dementsprechend Methoden der Zeitreihenanalyse wie der Moving-Blocks-Bootstrap angewendet werden.

Ziel der Anwendung des Moving-Block-Verfahrens auf die Original-Dokumentliste D^t ist die Erzeugung von δ geordneten Dokumentlisten BD_k^t, die zusammen mit D^t die Bootstrapmenge BD^t von Dokumentlisten bilden, die durch die nachfolgende Indexierungsfunktion weiteren Repräsentationsoperationen unterzogen wird. Durchgeführt werden kann dies durch eine unabhängige Erzeugung, d.h. es werden einzelne Dokumente D_i^t betrachtet, aus denen δ künstliche Dokumente D_{ik}^t erzeugt werden, indem die Funktion movB(.) angewendet wird:

$$\text{movB: } D(\Theta) \rightarrow D(\Theta): D_i^t \mid\rightarrow D_{ik}^t, k = 1, ..., \delta. \tag{235}$$

Bei dieser Formulierung werden ausschließlich Strukturelemente aus D_i^t verwendet, um die künstlichen Dokumente D_{ik}^t zu erzeugen, während bei einer abhängigen Erzeugung zunächst Strukturelemente aus allen Original-Dokumenten ermittelt werden, die in einer eigenen Menge gesammelt werden. Aus dieser Strukturelementemenge werden künstliche Dokumente durch sequentielles Ziehen mit Zurücklegen erzeugt. Im weiteren soll demgegenüber von einer unabhängigen Erzeugung ausgegangen werden.

Die wichtigste Entscheidung im Rahmen eines Moving-Block-Verfahrens ist die Festlegung des Blockgrößen-Parameters $l \in N$, der die Anzahl der Zeichen pro Block angibt. Dabei ist zu beachten, dass er nicht zu klein sein darf, damit zusammenhängende semantische Strukturen erfasst werden. Im Rahmen natürlich-sprachlicher Texte kann a-priori-Wissen z.B. über die durchschnittliche Satzlänge in Dokumenten aus D^t und der Standardabweichung der Satzlängen verwendet werden, indem l als Funktion der durchschnittlichen Satzlänge festgelegt wird, angegeben in einer Anzahl von Zeichen.

Nachdem die Block-Größe festgelegt wurde, werden aus einem Original-Dokument D_i^t alle vorliegenden, zusammenhängenden Teillisten DL_{ip}^t mit jeweils l Elementen extrahiert:

$$\begin{gathered} DL_{i1}^t = (d_{i1}^t, d_{i2}^t, ..., d_{il}^t), \\ DL_{i2}^t = (d_{i2}^t, d_{i3}^t, ..., d_{il+1}^t), \\ ... \\ DL_{i,\#D(i)-l+1}^t = (d_{i,\#D(i)-l}^t, d_{i,\#D(i)-l+1}^t, ..., d_{i,\#D(i)}^t). \end{gathered} \tag{236}$$

Die einzelnen Teillisten werden in die Bootstrap-Grundelementliste BDL_i^t eingefügt, wobei zu beachten ist, dass mehrfach auftretende Teilsequenzen aus D_i^t in BDL_i^t auch mehrfach vorliegen, was einen direkten Einfluss auf die Ziehungswahrscheinlichkeiten besitzt:

$$BDL_i^t = (DL_{ip}^t \mid p = 1, ..., \#D_i^t - l + 1). \tag{237}$$

Aus dieser Grundliste werden Teillisten zufällig mit Zurücklegen gezogen, und in der Reihenfolge der Ziehung in eine Zeichenliste eingefügt, die jeweils ein abgeleitetes, künstliches Dokument D_{ik}^t repräsentiert. Werden beispielsweise 8 sequentielle Ziehungen aus einer 9'elementigen Bootstrap-Grundmenge mit Zurücklegen durchgeführt, die zusammen das erste künstliche Dokument D_{i1}^t ergeben, so kann diese die folgende Form besitzen:

$$D_{i1}^t = (DL_{i6}^t, DL_{i2}^t, DL_{i3}, DL_{i7}^t, DL_{i5}^t, DL_{i9}^t, DL_{i4}^t, DL_{i1}^t). \tag{238}$$

Der zweite Parameter μ_B eines Moving-Block-Bootstrap-Verfahrens legt die Anzahl der Ziehungen fest, mit denen aus BDL_i^t eine Teilliste gezogen wird. Diese Anzahl ist abhängig von der Länge $\#D_i^t$ des Original-Dokuments, da ein künstliches Dokument etwa die gleiche Länge besitzen soll, wie das Original-Dokument. Im Gegensatz zu einer Zeitreihe, als spezieller Form einer Sequenz, ist eine genaue Übereinstimmung der Länge nicht notwendig, da durch die nachfolgende Indexierungsfunktion auch unterschiedlich lange Dokumente in den gleichen n-dimensionalen Dokumentraum abgebildet werden. Sei $round_+(.)$ die Rundungsoperation, die auf die nächst größere natürliche Zahl aufrundet, so ergibt sich die Anzahl μ_B der Ziehungen aus dem Quotient von $\#D_i^t$ und dem Blockgrößen-Parameter l:

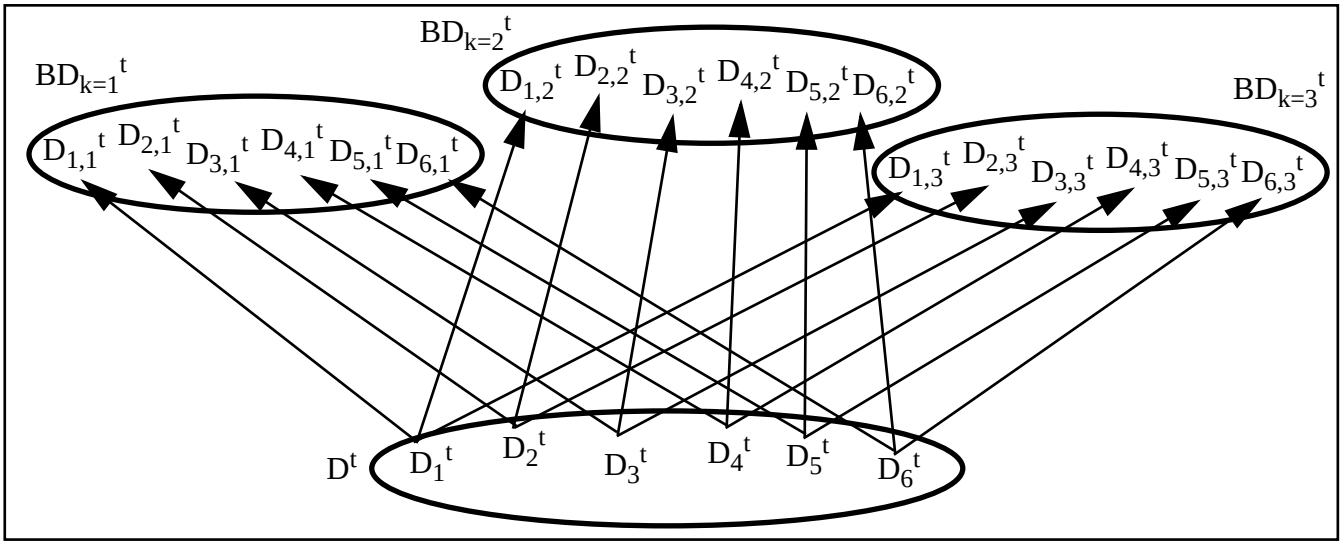
$$\mu_B = round_+(\#D_i^t/l). \tag{239}$$

Die Zeichenanzahl $\#D_{ik}^t$ der künstlichen Dokumente D_{ik}^t besitzt somit einen etwas größeren Wert mit

$$\#D_{ik}^t = \mu_B * l. \tag{240}$$

Der dritte Parameter δ ist die Anzahl der künstlichen Dokumentlisten BD_k^t, die aus D^t erzeugt werden sollen. Nachdem ein künstliches Dokument D_{ik}^t erzeugt wurde, wird es in eine der künstlichen Dokumentmengen BD_k^t eingefügt, d.h. in dieser Menge soll jeweils nur ein künstliches Dokument enthalten sein, das von dem Original D_i^t abgeleitet ist. Die Ableitung der Mengen BD_k^t aus BD_k^t besitzt somit die Struktur, die in Abb. 29) dargestellt wird.

Abb. 29) Ableitung von Bootstrap-Dokumentmengen aus Original-Dokumentmenge



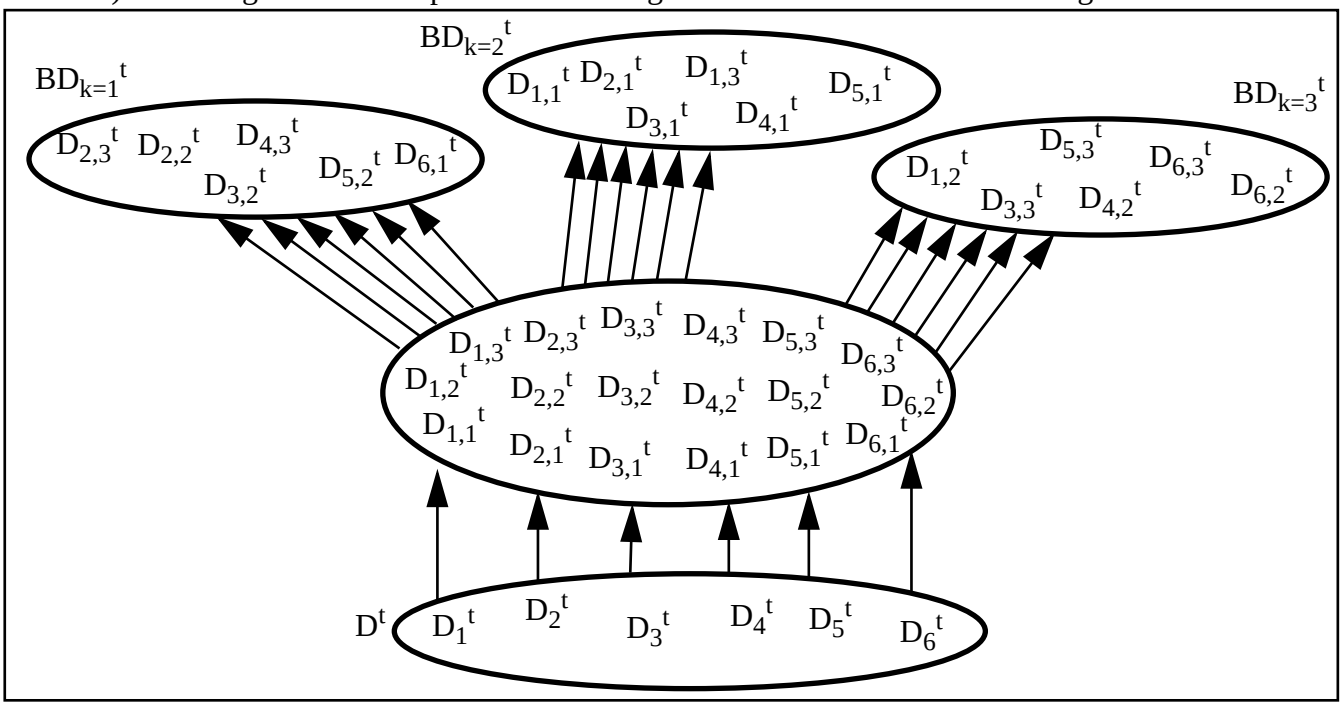
Wird eine erweiterte Variante der Moving-Block-Strategie mit mehreren Blockgrößen-Parametern l verwendet, so kann mit dem Mittelwert und der Standardabweichung der Satzlängen in D^t die beiden Parameter l_{min} und l_{max} festgelegt werden. Es wird davon ausgegangen, dass l_{max} - l_{min} Bootstrap-Grundelementelisten $BDL(l)_i^t$ für ein Dokument D_i^t gebildet werden, die in der Liste BML_i zusammengefasst werden:

$$BDL_i^t = (BDL(l)_i^t \mid l = l_{min}, ..., l_{max}), \text{ mit}$$
$$BDL(l)_i^t = (DL(l)_{ip}^t \mid p = 1, ..., \mu_M - l + 1). \tag{241}$$

Der untere Parameter l_{min} kann z.B. als Mittelwert minus der Standardabweichung der Zeichen in Sätzen festgelegt werden, und l_{max} als Mittelwert plus der Standardabweichung. Ein künstliches Dokument D_{ik}^t wird bei dieser Variante durch ein zweistufiges Verfahren festgelegt, indem zunächst eine Grundelementliste $BDL(l)_i^t$ aus BDL_i^t zufällig und mit Zurücklegen gezogen wird. Im Anschluss wird aus der gezogenen Liste eine Zeichenliste $DL(l)_{ip}^t$ der Länge l gezogen, welche die nächsten Zeichen des künstlichen Dokumentes bildet. Abgebrochen wird die zweistufige Ziehung, wenn zum ersten Mal mehr als μ_M Zeichen in dem künstlichen Dokument D_{ik}^t auftreten.

Es existiert zudem die Möglichkeit, die einzelnen Dokumente D_{ik}^t, k = 1, ..., δ in eine gemeinsame Zwischenmenge einzufügen, in der alle $m^t * \delta$ künstliche Dokumente gesammelt werden (siehe Abb. 30)). Aus dieser Zwischenmenge werden neue Dokumentlisten durch Ziehen ohne Zurücklegen erzeugt. Diese Möglichkeit soll hier jeoch nicht näher betrachtet werden, da durch das zufällige Ziehen Dokumentlisten erzeugt werden können, die von D^t stark abweichen.

Abb. 30) Ableitung von Bootstrap-Dokumentmengen aus Zwischen-Dokumentmenge



3.3) Merkmale als Zeichensequenz

3.3.1) Monorepräsentation von Merkmalen

Im Rahmen des Vektorraummodells werden Dokumentvektoren x_i^t als Punkte in einem n^t-dimensionalen Dokumentvektorraum DVR interpretiert. Jeder einzelnen Dimension von DVR entspricht ein Merkmal oder Feature F_i^t, bei dem es sich beispielsweise um Terme bei natürlich-sprachlichen Texten, oder um Zeichen-n- bzw. oder um Term-n-Gramme handeln kann (vergl. Scholtes (1993[301]), Bachelier (1995: 34ff[14])). D.h. Terme wie n-Gramme sind Zeichensequenzen, die aus dem gleichen Alphabet $\Theta = (d_k \mid k = 1, ..., \#\Theta \in N)$ stammen, wie die Dokumente, damit die Beschreibung von Dokumenten durch Merkmale im Rahmen der Indexierung durchführbar ist. Im Vergleich zu Dokumenten sind Merkmale jedoch

sehr kleine Zeichensequenzen, wobei die Länge bei Termen nicht spezifiziert ist, wohingegen Zeichen-n-Gramme über die gleiche Anzahl n definiert sind.

Die n^t Merkmale bilden zusammen eine Merkmalsliste F^t, die zeitlich variabel sein kann, sodass eine Darstellung durch den Zeitindex t präzisiert wird:

$$F^t = (F_i^t \mid i = 1, ..., n^t). \tag{242}$$

3.3.2) Polyrepräsentation von Merkmalen

Eine Polyrepräsentation von Merkmalen ist demgegenüber nicht so offensichtlich wie bei Dokumenten, da keine künstlichen Merkmale analog zu künstlichen Dokumenten sinnvoll sind, wenn man sich auf den Fall von Zeichen-n-Gramme beschränkt, sodass ein Moving-Block-Bootstrap nicht sinnvoll angewendet werden kann. Dies ergibt sich daraus, dass alle auftretenden Zeichen-n-Gramme in F^t bzw. alle möglichen Zeichen-n-Gramme in F berücksichtigt werden. Die Menge aller möglichen n-Gramme besitzt $\#\Theta^n$ Elemente, sodass die Verwendung von F nur bei kleinen n wie z.B. aus {2, 3, 4} mit den momentan verfügbaren Nicht-Superpositions-Architekturen (siehe z.B. Preskill (1998[265]), Siegelmann (1999[310])) durchführbar ist.

Eine offensichtliche Polyrepräsentation bei der Verwendung von Zeichen-n-Gramme ist die parallele Verwendung von unterschiedlichen n. Die Menge aller auftretenden Merkmale als Zeichensequenzen der Länge k sei mit F_k^t bezeichnet, so ergibt sich eine sinnvolle Merkmals-Polyrepräsentation, wobei die Bezeichung BF^t in Analogie zu der Menge BD^t verwendet wird, ohne hier auf ein Bootstrapverfahren hinzudeuten:

$$BF^t = \{F_k^t \mid k = 2, 3, 4, 5\}. \tag{243}$$

Durch die oben angesprochene kombinatorische Problematik mit großen n-Grammen, besitzt diese Form der Polyrepräsentation keine Skalierungsfähigkeit wie Bootstrapverfahren, bei denen δ mit {50, ...} hinreichend groß gewählt werden kann, um bei der Kombination von Einzelmodellen zu einem Gesamtmodell statistisch signifikante Aussagen zu erreichen. Für eine statistisch erzeugte Polyrepräsentation ist die Variation von n somit nicht geeignet.

Bei der Bildung künstlicher Dokumentlisten BD_k^t wurden zunächst künstliche Dokumente D_{ik}^t aus einem Ursprungsdokument D_i^t durch ein Moving-Block-Bootstrap-Verfahren erzeugt. Auf die Möglichkeit der Erzeugung von künstlichen Dokumentlisten durch Resampling von Ursprungsdokumenten ohne die Erzeugung künstlicher Dokumente, wurde nicht eingegangen, da dadurch eine Polyrepräsentations-Dokumentliste jedoch keine Polyrepräsentations-Dokumente erzeugt werden. Geht man von der Merkmalsliste F^t aus, so könnten Bootstrap-Merkmalslisten F_k^t durch n^t'faches Ziehen von Merkmalen aus F^t gebildet werden, und die Ursprungsdokumentliste D^t könnte mit jedem der Merkmalslisten F_k^t indexiert werden. Gegen ein Resampling von Ursprungsmerkmalen aus F^t spricht das gleiche Argument wie bei dem Resampling von Ursprungsdokumenten aus D^t, da dadurch eine Polyrepräsentation der Merkmalsliste, jedoch keine Polyrepräsentation der Merkmale erzeugt wird.

Grundsätzlich anders sieht die Situation aus, wenn keine Zeichen-n-Gramme verwendet werden, sondern Terme z.B. aus kontrollierten Vokabularien. In einem solchen Kontext erscheint die Erzeugung von künstlichen Merkmalslisten sinnvoll, da jede dieser Listen ein eigenes Vokabular repräsentiert. Geht man von natürlich-sprachlichen Texten aus, so bedeutet dies, dass Dokumente einer Dokumentliste D^t mit Hilfe unterschiedlicher Vokabularien, d.h. Merkmalslisten, indexiert werden. Dies ergibt auch keine Polyrepräsentation von Merkmalen, doch eine sinnvolle Polyrepräsenation der Indexierung, da ein Dokumentvektor eines Dokumentes jeweils bestimmte Aspekte in Abhängigkeit von dem verwendeten Vokabular repräsentiert, das in einem anderen Dokumentvektor des gleichen Ursprungsdokumentes nicht oder nur teilweise vorkommt.

3.4) Indexierung

Die Aquisitionsfunktion A wird allgemein als Funktion verstanden, die eine bestimmte Form von Wissen in eine andere Repräsentationsform überführt, die durch die verwendeten Inferenzfunktionen manipulierbar sind:

$$A: W \rightarrow W. \tag{244}$$

Beispielsweise werden in Expertensystemen natürlich-sprachliche Aussagen von Experten in Regeln überführt, die durch Regelinterpreter manipulierbar werden.

3.4.1) Monorepräsentation der Indexierung

Beim Vektorraummodell des IR wird die Indexierung als spezielle Indexierungsfunktion $A_{IR(D)}$ verwendet, um ein Dokument D_i auf einen Dokumentvektor x_i^t abzubilden, der einem Punkt in einem n-dimensionalen Dokumentvektorraum $DVR \subseteq R^n$ entspricht (siehe Abb. 31)). D.h. $A_{IR(D)}$ wird als Abbildung aus der Menge $D(\Theta)$ aller Dokumente bezüglich eines Alphabetes Θ in einen Dokumentvektorraum DVR definiert als:

$$A_{IR(D)}: D(\Theta) \rightarrow DVR: D_i = (d_{ij} \mid \forall d_{ij} \in \Theta) \mapsto x_i^t = (x_{ij}^t \in R \mid j = 1, ..., n^t). \tag{245}$$

Die Komponenten x_{ij} dieses Dokumentvektors repräsentieren eine Gewichtung des j'ten Merkmals f_j des Dokumentes D_i, wobei unterschiedliche Gewichtungsverfahren verwendet werden können (siehe Abschnitt 3.5)).

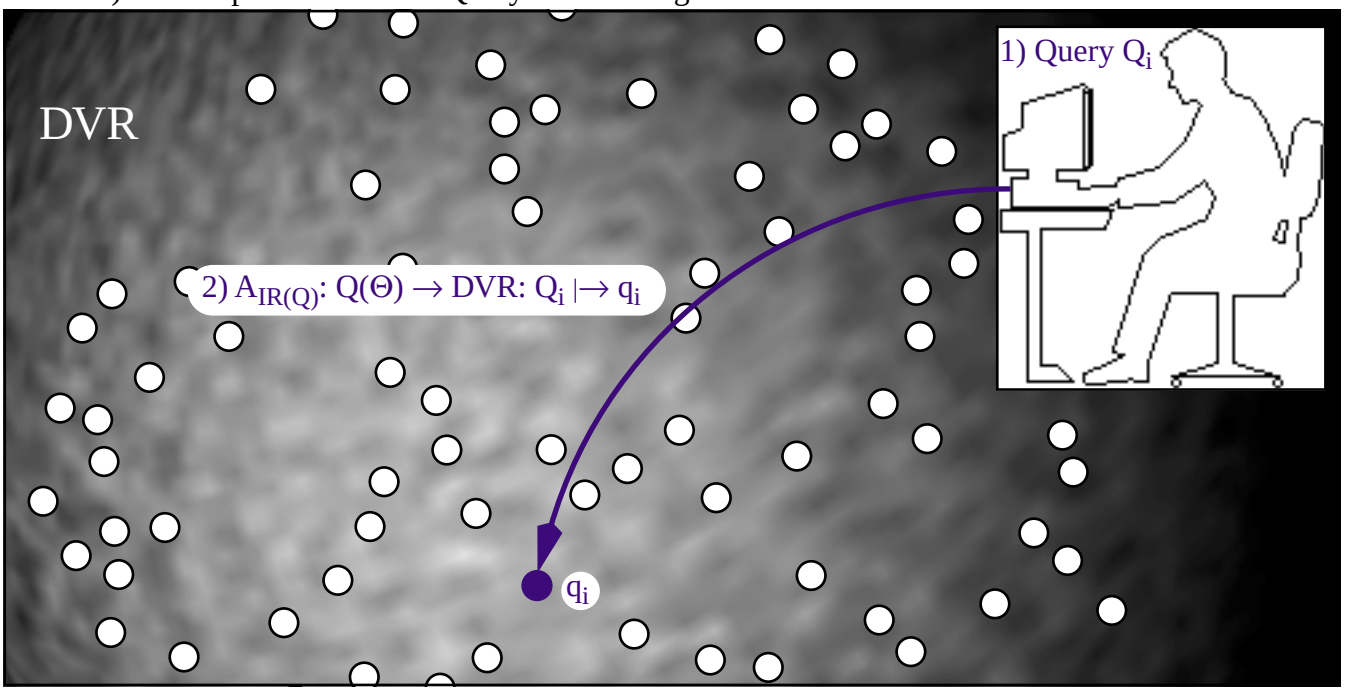
Werden alle Dokumente D_i^t in der momentan vorliegenden Dokumentmenge D^t mit Hilfe der Merkmale aus F^t indexiert, so kann eine Dokument-Merkmals-Matrix DMM^t gebildet werden, in der alle Dokumente bezüglich allen Merkmalen beschrieben werden. Die Zeilen von DMM^t sollen mit den m^t Dokumenten und die Spalten mit den n^t Merkmalen belegt werden:

$$DMM^t = (x_{ij}^t \mid i = 1, ..., m^t; j = 1, ...n^t), \text{ oder} \tag{246}$$

Tabelle 2) Dokument-Merkmals-Matrix DMM^t

D_i^t / F_j^t	F_1^t	F_2^t	F_3^t	...	$F_{n(t)}^t$
D_1^t	x_{11}^t	x_{12}^t	x_{13}^t	...	$x_{1n(t)}^t$
D_2^t	x_{21}^t	x_{22}^t	x_{23}^t	...	$x_{2n(t)}^t$
D_3^t	x_{31}^t	x_{32}^t	x_{33}^t	...	$x_{3n(t)}^t$
...	...	...	...	...	...
$D_{m(t)}^t$	$x_{m(t)1}^t$	$x_{m(t)3}^t$	$x_{m(t)3}^t$	...	$x_{m(t)n(t)}^t$

Abb. 31) Monorepräsentation der Query-Indexierungsfunktion



Die DMM^t kann auch spaltenweise gelesen werden, indem keine Dokumentvektoren x_i^t sondern Merkmalsvektoren f_j^t betrachtet werden. Zur Unterscheidung sollen die Komponenten eines Merkmalsvektors mit f_{ji}^t bezeichnet werden, wobei der konkrete Gewichtungsfaktor gleich x_{ji}^t ist:

$$f_j^t = (f_{ji}^t \mid i = 1, ..., m^t). \tag{247}$$

Diese Darstellung definiert einen m^t-dimensionalen Feature- bzw. Merkmalsvektorraum $FVR \subseteq R^{m(t)}$ in Analogie zu dem n^t-dimensionalen Dokumentvektorraum $DVR \subseteq R^{n(t)}$, in dem die einzelnen Deskriptoren jeweils einen Punkt repräsentieren. Im Rahmen des Vektorraummodells wird (mindestens) eine Metrik in diesem Featureraum verwendet, die mit $d_{FVR}(., .)$ bezeichnet werden soll.

3.4.2) Polyrepräsentation der Indexierung

In dem vorangegangenen Abschnitt wurde ein Dokument D_i^t als genau ein Dokumentvektor x_i^t in DVR durch eine Indexierungsfunktion $A_{IR(D)}$ repräsentiert, was dem Standardfall einer Monorepräsentation entspricht. Soll eine Polyrepräsentation betrachtet werden, so sind zwei Fälle zu unterscheiden:

1) Es existiert bereits eine Poyrepräsentation von Dokumenten z.B. durch ein Moving-Blocks-Bootstrap-Verfahren, das aus D^t die Dokumentlistenmenge $BD^t = \{D^t, BD_k^t \mid k = 1, ..., \delta\}$ erzeugt hat.
2) Es liegt nur eine Monorepräsentation von D^t vor.

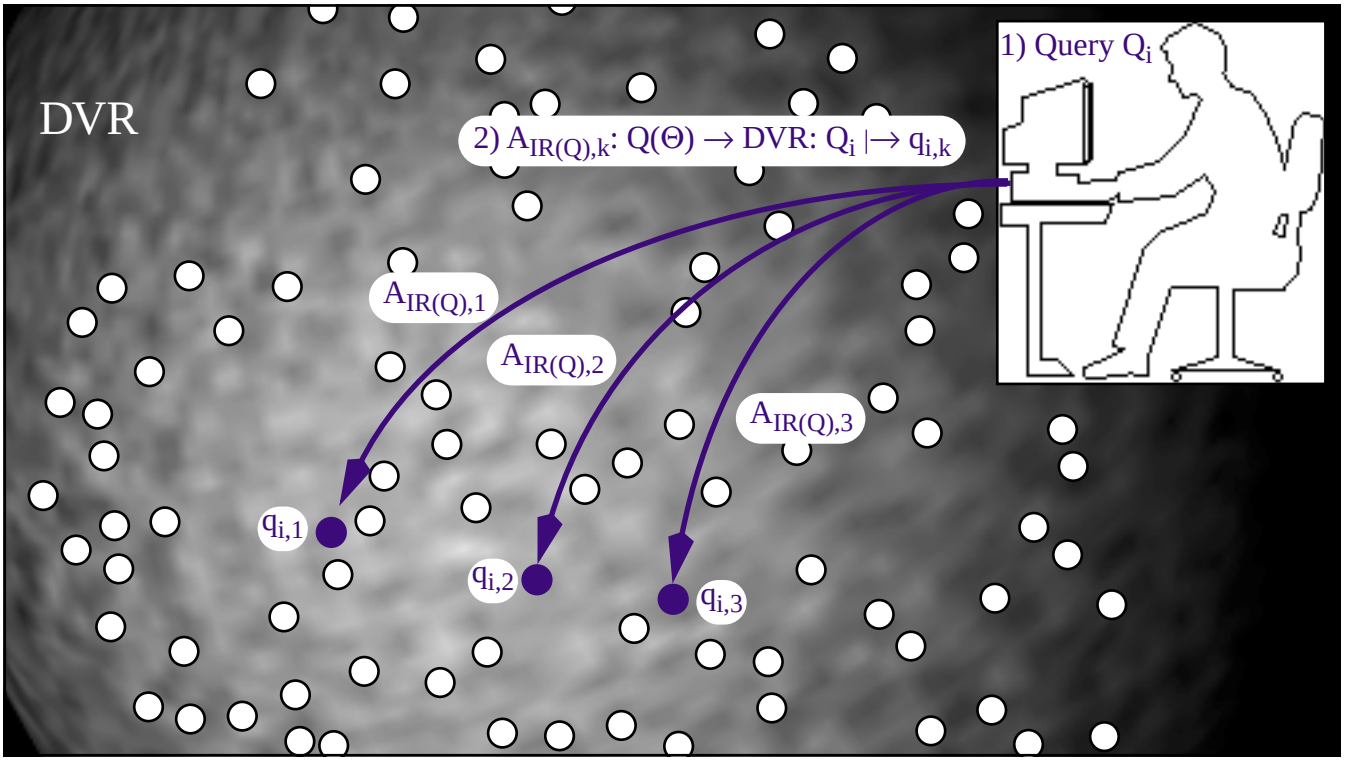
Die Durchführung einer Polyrepräsentations-Indexierung ist im ersten Fall offensichtlich, da jede der Dokumentlisten aus BD^t mit genau einer Indexierungsfunktion $A_{IR(D)}$ auf eine Dokumentvektorenliste abgebildet werden kann:

$$A_{IR(D)}: D(\Theta) \rightarrow DVR:$$
$$D_i^t \mid\rightarrow x_i^t \in R^{n(t)}, \forall D_i^t \in D^t \wedge$$
$$D_{ik}^t \mid\rightarrow x_{ik}^t \in R^{n(t)}, \forall D_{ik}^t \in BD_k^t, k = 1, ..., \delta. \quad (248)$$

Auf diese Weise entstehen zu den $\delta + 1$ Dokumentlisten D^t, BD_k^t korrespondierende Dokumentvektorenlisten DV^t, BDV_k^t:

$$BDV^t = \{DV^t, BDV_k^t \mid k = 1, ..., \delta\},$$
$$DV^t = (x_i^t \in R^{n(t)} \mid i = 1, ..., m^t),$$
$$BDV_k^t = (x_{ik}^t \in R^{n(t)} \mid i = 1, ..., m^t), k = 1, ..., \delta. \quad (249)$$

Abb. 32) Polyrepräsentation der Query-Indexierungsfunktion mit $\delta = 3$



Liegt nur eine Monorepräsentation von D^t vor, so besteht die einzige Möglichkeit der Polyrepräsentations-Indexierung in der Erzeugung von Indexierungsfunktionen $A_{IR(D,k)}$ neben der Original-Indexierungsfunktion $A_{IR(D)}$, die in der Funktionsmenge $IM_{IR(D)}$ zusammengefasst werden sollen :

$$IM_{IR(D)} = \{A_{IR(D)}, A_{IR(D,k)} \mid k = 1, ..., \delta\}. \quad (250)$$

Jedes Dokument D_i^t aus D^t wird unabhängig voneinander durch die Funktionen $A_{IR(D)}$, $A_{IR(D,k)}$ indexiert (siehe Abb. 32)):

$$A_{IR(D)}: D(\Theta) \rightarrow DVR: D_i^t \mid\rightarrow x_i^t \in R^{n(t)},$$
$$A_{IR(D,k)}: D(\Theta) \rightarrow DVR: D_i^t \mid\rightarrow x_{ik}^t \in R^{n(t)}, k = 1, ..., \delta. \quad (251)$$

Zu beachten ist, dass alle Funktionen $A_{IR(D)}$, $A_{IR(D,k)}$ Dokumente in den gleichen Dokumentvektorraum $DVR \subseteq R^{n(t)}$ abbilden, d.h. durch diese Formulierung werden die Polyrepräsentationen ausgeschlossen, die Dokumente durch verschiedene Indexierungsfunktionen in unterschiedliche Dokumentvektorenräume abbilden, die sich durch abweichende Merkmale und Anzahl der Merkmale (Dimension der Dokumenträume) unterscheiden. Beispielsweise können durch unterschiedliche n-Gram-Kodierungen unterschiedliche Merkmale definiert werden, die verschiedene Dokumentvektorenräume erzeugen, worauf jedoch nicht näher eingegangen werden soll.

Es bleibt die Frage, wie unterschiedliche Indexierungsfunktionen $A_{IR(D,k)}$ mit $D(\Theta) \rightarrow DVR$ erzeugt werden können. Eine Möglichkeit sind unterschiedliche Merkmalsgewichtungen (siehe Abschnitt 3.5)), d.h. die absolute Häufigkeit $h(F_i \mid D_j^t)$ des Auftretens des Merkmals F_i im Dokument D_j^t wird in ein Gewicht $w(F_i \mid D_j^t)$ transformiert, das als Komponente x_{ji}^t des Dokumentvektors x_j^t verwendet wird. Für die Transformationsfunktion sind eine Reihe von Modellen erzeugt worden. Interessant sind inbesondere parametrische Modelle, d.h. Modelle, die durch einen oder mehrere freie Parameter spezifiziert werden, da man mit unterschiedlichen Parameterwerten jeweils eine Indexierungsfunktion $A_{IR(D,k)}$ spezifizieren kann.

Unterschiedliche Indexierungsfunktionen $A_{IR(D,k)}$ können auch durch unterschiedliche Merkmalslisten F_k^t vorgenommen werden. Ein Resampling aus einer Grundmenge F^t von Merkmalen wurde im Rahmen der Darstellung von Merkmalen als Zeichensequenzen bereits ausgeschlossen, wohingegen bei der Verwendung von Termen als Merkmale diese Vorgehensweise sinnvoll sein kann, da einzelne Merkmalslisten als Vokabularien interpretierbar sind, die jeweils eigene Aspekte repräsentieren. Zwischen den Vakabularien existiert eine gewisse Überlappung, wenn Resamplingverfahren verwendet werden, die die unterschiedlichen Aspekte und Ausrichtungen statistisch repräsentieren.

Eine vollständig andere Art der Indexierungsfunktions-Polyrepräsentation ergibt sich durch die Einführung stochastischer Indexierungsfunktionen. Bislang wurde ausschließlich der Fall deterministischer Indexierungsfunktionen betrachtet, die sich dadurch auszeichnen, dass bei einem mehrmaligen und unabhängigen Input der gleichen Zeichensequenz immer der gleiche Vektor erzeugt wird, d.h. wird das gleiche Dokument D_i^t mehrmals indexiert, so wird jeweils der gleiche Dokumentvektor x_i^t erzeugt. Bei einer stochastischen Indexierungsfunktion erzeugt jede unabhängige Indexierungsoperation eines Dokumententes einen eigenen Dokumentvektor x_{ik}^t im DVR.

Die einfachste Konstruktion einer stochastischen Indexierungsfunktion besteht in der Verwendung einer deterministischen Indexierungsfunktion, zu der eine Zufallskomponente addiert wird. In diesem Fall existiert ein Dokumentvektor x_i^t, der durch die deterministische Indexierungsfunktion erzeugt wird. Ein stochastisch abgeleiteter Dokumentvektor x_{ik}^t wird erzeugt, indem ein n-dimensionaler Vektor $\sigma_k^t \in R^n$ von Zufallszahlen hinzu addiert wird:

$$A_{IR(D)}: D(\Theta) \rightarrow DVR: D_i^t \mid\rightarrow x_{ik}^t = x_i^t + \sigma_k^t; \sigma_k^t = (\sigma_{kj}^t \in R \mid j = 1, ..., n). \quad (252)$$

Die Addition eines Zufallsvektors kann auch als eine Fom der Mutation interpretiert werden, sodass das Queryvektor-Polyrepräsentations-Verfahren als eine ES-Mutations-Operation interpretiert werden kann (siehe Abschnitt 2.5)):

$$x_{ik}^t = m(x_i^t \mid \sigma_k^t). \quad (253)$$

Die entscheidende Frage in diesem Fall besteht in der Festlegung einer Wahrscheinlichkeitsverteilung, durch welche die einzelnen Komponenten σ_{kj}^t erzeugt werden. Hierbei muss die Topologie und die Metrik des Dokumentvektorenraums berücksichtigt werden. Weiterhin muss die Lokalität erhalten werden, d.h. die Verteilung muss in x_i^t ihr Maximum besitzen, und mit zunehmender Distanz zu x_i^t wird ihr Wert monoton kleiner, eventuell in Verbindung mit einem Distanzschwellenwert, bei dessen Überschreitung der Wahrscheinlichkeitswert auf Null gesetzt wird. Jede Komponente σ_{kj}^t könnte durch eine Normalverteilung mit dem Erwartungswert Null erzeugt werden, und einer Standardabweichung, die eine Funktion der Dokumentvektorenverteilung ist. Z.B. könnte die durchschnittliche Distanz und die Standardabweichung der Distanzen der Dokumentvektoren verwendet werden, um die Standardabweichung der Normalverteilung zu konstruieren, mit der eine Komponente σ_{kj}^t erzeugt wird. Grundlage ist dabei die Dokumentvektorenverteilung, die durch die deterministische Indexierungsfunktion erzeugt wird.

Unabhängig ob die Dokumentvektoren-Polyrepräsentation $BDV^t = \{DV^t, BDV_k^t \mid k = 1, ..., \delta\}$ durch eine Indexierungsfunktion-Monorepräsentation $A_{IR(D)}$ aus einer Dokument-Polyrepräsentation $BD^t = \{D^t, BD_k^t \mid k = 1, ..., \delta\}$ erzeugt wird, oder ob BDV^t durch eine Indexierungsfunktion-Polyrepräsentation $IM_{IR(D)} = \{A_{IR(D)}, A_{IR(D,k)} \mid k = 1, ..., \delta\}$ aus einer Dokument-Monorepräsentation D^t erzeugt wird, jede der Dokumentvektorenlisten lässt sich als Dokument-Merkmals-Matrix darstellen. Auf diese Weise wird neben der Matrix DMM^t, die zu DV^t korrespondiert, eine Menge neuer Matrizen DMM_k^t erzeugt, die zu den Dokumentvektorenlisten BDV_k^t korrespondieren.

Durch diese Darstellung als Dokument-Merkmals-Matrizen ist auch das Problem der Merkmalsvektoren-Polyrepräsentation gelöst. Dies ergibt sich dadurch, dass jedes Merkmal F_i^t in einer Dokument-Merkmals-Matrix als eine Spalte über den vorhandenen m^t Dokumenten kodiert wird, sodass das gleiche Merkmal in $\delta + 1$ unterschiedlichen Dokument-Merkmals-Matrizen jeweils einen eigenen Merkmalsvektor f_i^t, f_{ik}^t, $k = 1, ..., \delta$, besitzt, die zusammen als Merkmalsvektoren-Polyrepräsentationen direkt verwendet werden können. Bei einer gleichen Liste F^t von Merkmalen ergeben sich somit $\delta + 1$ Listen von Merkmalsvektoren, die in einer Menge BFV^t zusammengefasst werden sollen:

$$BFV^t = \{FV^t, BFV_k^t \mid k = 1, ..., \delta\},$$
$$FV^t = (f_i^t \in R^{m(t)} \mid i = 1, ..., m^t), BFV_k^t = (f_{ik}^t \in R^{m(t)} \mid i = 1, ..., m^t). \quad (254)$$

3.5) Merkmalsgewichtungsmodelle

3.5.1) Grundlegende Merkmalsgewichtungsmodelle

Gemeinsames Ziel aller Verfahren, die Merkmalsgewichte erzeugen, ist die Bildung einer Rangfolge von Merkmalen, in der durch Schwellenwerte wichtige von weniger wichtigen Merkmalen getrennt werden können (vgl. Panyr (1986b[248], 1987a[251])). Dies hat den Zweck, die Dokument-Merkmals-Matrix zu verkleinern, da nur die wichtigsten Merkmale zur Indexierung herangezogen werden (vgl. auch Salton & McGill (1987: 65ff[294])), was bereits einer Dimensionsreduzierung entspricht. Einen Überblick über Gewichtungsverfahren und Ähnlichkeitsmaße, die diese Gewichtungen verwenden, wird in Zobel & Moffat (1998[376]) gegeben.

Die Häufigkeiten von Features in Dokumenten und der gesamten Dokumentenmenge D^t spielen dabei eine wesentliche Rolle. Die absolute Häufigkeit des Auftretens des Feature F_i im Dokument D_j wird mit $h(F_i \mid D_j)$, und die absolute Häufigkeit des Auftretens des Feature F_i in der Dokumentliste D^t wird mit $h(F_i \mid D^t)$ bezeichnet. Die absolute Häufigkeit der Features, die im Dokument D_j auftreten, was der Dokumentenlänge entspricht, soll mit $h(F^t \mid D_j) = \#D_j$ bezeichet werden, wenn das mehrfache Auftreten eines Features auch mehrmals in die Summe eingeht. Das Feature, das am häufigsten im Dokument D_j auftritt, wird mit $F_{max|j}$, und das Feature, das am häufigsten in D^t auftritt, wird mit $F_{max|D(t)}$ bezeichnet. Die korrespondierenden absoluten Häufigkeiten sind $h(F_{max|j} \mid D_j)$ und $h(F_{max|D(t)} \mid D^t)$. Die Menge der Features aus F^t, die in einem Dokument D_j enthalten sind, wird als $F(D_j)^t$ bezeichnet. Die Menge der Dokumente aus D^t, in denen ein Feature F_i enthalten ist, wird als $D(F_i)^t$ bezeichnet.

Mit diesen Häufigkeiten wurden verschiedene Gewichtungsmodelle von Merkmalen gebildet, wobei das Gewicht sich auf die Dokumentmenge D^t bezieht, d.h. bei $w(F_i \mid D^t)$ handelt es sich um ein Gewicht eines Merkmals F_i bezogen auf D^t, im Gegensatz zu relativen Gewichtungen $w(F_i \mid D_j^t)$ eines Merkmals F_i bezogen auf ein Dokument D_j^t. Bei $w(F_i \mid D^t)$-Gewichtungen soll die allgemeine Regel verwendet werden, dass die Gewichtung um so geringer sein soll, je häufiger das Auftreten des Merkmals in der gesamten Dokumenmenge ist, sodass diese Gewichtungen auch als inverse Dokumenthäufigkeit oder inverse document frequency (idf) bezeichnet werden. Die $w(F_i \mid D_j^t)$-Gewichtungen verwenden demgegenüber die Regel, dass der Gewichtungsfaktor umso höher sein soll, je häufiger das Auftreten in einem Dokument ist.

Die Gewichtung für binäre Vergleiche wird angegeben mit $w(F_i \mid D^t) = 1$. Die logarithmische Gewichtung und die normalisierte logarithmische Gewichtung werden bestimmt zu

$$w(F_i \mid D^t) = \log_e[1+ m^t/h(F_i \mid D^t)], \text{ bzw.}$$
$$w(F_i \mid D^t) = \log_e[1 + h(F_{max|D(t)} \mid D^t)/h(F_i \mid D^t)]. \quad (255)$$

Die hyperbolische Gewichtung und die normalisierte hyperbolische Gewichtung werden bestimmt zu

$$w(F_i \mid D^t) = 1/h(F_i \mid D^t), \text{ bzw.}$$
$$w(F_i \mid D^t) = \log_e[(m^t - h(F_i \mid D^t))/h(F_i \mid D^t)]. \quad (256)$$

Mit Hilfe des Informationsballastes (Salton & McGill (1987: 70[294])) oder des Rauschens $\nu(F_i \mid D^t)$ und des Signals $sig(F_i \mid D^t)$ wurden weitere Maße definiert:

$$\nu(F_i \mid D^t) = \sum_{D(j)} [-h(F_i \mid D_j)/h(F_i \mid D^t) * \log_2(h(F_i \mid D_j)/h(F_i \mid D^t))], \forall D_j \in D(F_i)^t,$$
$$sig(F_i \mid D^t) = \log_2[h(F_i \mid D^t) - \nu(F_i \mid D^t)]. \quad (257)$$

Zu diesen Maßen gehört die Entropie, sowie drei weitere Maße:

$$w(F_i \mid D^t)_{Entropie} = 1 - \nu(F_i \mid D^t)/\log_2[m^t],$$
$$w(F_i \mid D^t) = sig(F_i \mid D^t),$$
$$w(F_i \mid D^t) = sig(F_i \mid D^t)/\nu(F_i \mid D^t),$$
$$w(F_i \mid D^t) = \nu(F_i \mid D^t)_{max} - \nu(F_i \mid D^t), \text{ mit } \nu(F_i \mid D^t)_{max} = \max\{\nu(F_i \mid D^t) \mid \forall F_i \in F^t\}. \quad (258)$$

Fasst man $w(F_i \mid D^t)$ allgemein als Funktion g(.) einer Menge von Variablen und Konstanten auf, so ergeben sich die folgenden Funktionsdarstellungen:

$$w(F_i \mid D^t) = g(m^t, h(F_i \mid D^t), h(F_i \mid D_j) \mid \forall D_j \in D(F_i)^t), \text{ bzw.}$$
$$w(F_i \mid D^t) = g(m^t, \nu(F_i \mid D^t), s(F_i \mid D^t), \nu(F_i \mid D^t)_{max}). \quad (259)$$

Entsprechend der Vorgehensweise in Zobel & Moffat (1998:22[376]) werden aus den dokumentmengenspezifischen Merkmalsgewichtungen $w(F_i \mid D^t)$ dokumentspezifische Merkmalsgewichtungen $w(F_i \mid D_j^t)$ erzeugt, die als Komponenten x_{ji} der Dokumentvektoren verwendet werden. Dabei werden entweder die dokumentmengenspezifischen Merkmalsgewichtungen direkt übernommen, oder sie werden mit relativen Häufigkeiten $r(F_i \mid D_j)$ multipliziert:

$$x_{ji} := w(F_i \mid D_j^t) = w(F_i \mid D^t), \text{ oder}$$
$$x_{ji} := w(F_i \mid D_j^t) = r(F_i \mid D_j) * w(F_i \mid D^t). \quad (260)$$

Für die relativen Häufigkeiten $r(F_i \mid D_j^t)$ existieren eine Reihe von verschiedenen Formulierungen. Die relative Häufigkeit für einen binären Vergleich wird definiert mit

$$r(F_i \mid D_j) = \begin{cases} 1, & \text{wenn } F_i \in F(D_j)^t, \\ 0, & \text{sonst.} \end{cases} \quad (261)$$

Die Standardhäufigkeit ist gleich der absoluten Häufigkeit:

$$r(F_i \mid D_j) = h(F_i \mid D_j). \quad (262)$$

Die logarithmische Häufigkeit wird definiert durch:

$$r(F_i \mid D_j) = 1 + \log_e(h(F_i \mid D_j)). \quad (263)$$

Normalisierte relative Häufigkeiten werden angegeben mit dem Parameter $K \in [0{,}3, 0{,}5]$:

$$r(F_i \mid D_j) = h(F_i \mid D_j)/h(F_{max|j} \mid D_j), \text{ oder}$$
$$r(F_i \mid D_j) = K + (1 - K) * h(F_i \mid D_j)/h(F_{max|j} \mid D_j). \quad (264)$$

Der Diskriminanzwert bringt zum Ausdruck, wie gut ein Merkmal Dokumente unterscheiden kann. Allgemein wird davon ausgegangen, dass „die Retrievaleffektivität umgekehrt proportional zur Dichte des Dokumentenraumes“ (Panyr (1987a: 14[251])) ist. Dies ist dann der Fall, wenn:
a) die Dokumente separierbar sind,
b) die Dokumente Cluster bilden, die separierbar sind.

Zuerst wird die Dichte der Dokumentmenge D^t im Dokumentvektorraum DVR mit Hilfe einer Ähnlichkeitsfunktion s(.,.) und dem Dokumentvektor $\bar{x}$ des Zentroiden der Dokumentmenge berechnet, die mit $DURAEN^t$ bezeichnet werden soll (vgl. Salton & McGill (1987: 73[294]), Panyr (1987a: 14[251])):

$$DURAEN^t = c * \sum_j s(\bar{x}, x_j), \text{ mit}$$
$$\bar{x} = 1/m^t * \sum_j x_j, \forall D_j \in D^t, \text{ und c als einem Parameter.} \quad (265)$$

Im nächsten Schritt wird ein $(n^t - 1)$-dimensionaler Dokumentvektorraum $DVR_{-i} \subseteq R^{n(t)-1}$ betrachtet, der aus den Merkmalen aus der Menge $F^t \setminus \{F_i\}$ gebildet wird, d.h. alle Dokumentvektoren x_j werden ohne die Komponente x_{ji} betrachtet. In diesem Dokumentvektorraum DVR_{-i} wird ebenfalls die Dichte in Form von $DURAEN_{-i}^t$ berechnet. Der Diskriminanzwert DW_i^t wird als Dichte ohne F_i minus die Gesamtdichte berechnet:

$$DW_i^t = DURAEN_{-i}^t - DURAEN^t. \quad (266)$$

Ein Merkmal F_i ist ein guter Diskriminator, wenn DW_i^t einen Wert größer Null einnimmt ($DW_i^t > 0$), was bedeutet, dass durch die Einfügung dieses Merkmals die Dokument-Raum-Ähnlichkeit zunimmt. Ein schlechter Diskriminator ($DW_i^t < 0$) verdichtet den Dokument-Raum. Entnimmt man einen schlechten Diskriminator, dann wird die Dokument-Raum-Ähnlichkeit kleiner und die Dokumente sind besser zerlegbar (vgl. Panyr (1987a: 15[251])). Der Diskriminanzwert kann zur Berechnung von dokumentspezifischen Merkmalsgewichtungen verwendet werden, indem DW_i^t mit der Häufigkeit $h(F_i \mid D_j)$ von F_i in D_j gewichtet wird:

$$w(F_i \mid D_j) = h(F_i \mid D_j) * DW_i^t. \quad (267)$$

Es lässt sich zeigen, dass Merkmale mit den besten Signal-Werten $s(F_i \mid D^t)$ eine sehr niedrige Dokument- und Gesamthäufigkeit besitzen, d.h. Begriffe mit dem besten Informationswert können nicht gut zwischen verschiedenen Dokumenten differenzieren. Da man den Diskriminanzwert als ein gutes Maß zur Bewertung von Deskriptoren ansehen kann, kann man bei der Verwendung von Informationswerten eine Reihenfolge mit fallender Güte der Deskriptoren dadurch aufstellen, indem man die Deskriptoren mit steigenden Informationswerten auflistet.

Der Variationsquotient V_i^t eines Merkmals F_i bezüglich der Dokumentenmenge D^t wird verwendet, um zwei Mengen von Merkmalen auszuwählen, die als sehr wichtig bzw. wichtig gelten. In Crouch (1972[84]) werden als Merkmale Terme verwendet, wobei die sehr wichtigen Terme als core-terms und die wichtigen Terme als post-terms bezeichnet werden (vgl. Panyr (1986a: 92ff[247])).

Entsprechend der Vorgehensweise bei Crouch (1972[84]) wird zunächst eine zufällige Untermenge aus D^t gebildet, in der die Variationsquotienten der Terme bestimmt werden. Dann werden die Terme nach steigenden Variationsquotienten geordnet und es werden die ersten m_c Terme als core-terms und die Terme (m_c + 1 bis m_p) als post-terms verwendet. D.h. der Variationsquotient V_i^t des Merkmals F_i wird direkt als dokumentspezifisches Merkmalsgewicht verwendet:

$$w(F_i \mid D_j) = V_i^t. \tag{268}$$

Berechnet wird der Variationsquotient V_i^t des Merkmals F_i aus der Varianz var_i^t und der mittleren Häufigkeit $\bar{h}(F_i \mid D^t)$ des Auftretens von F_i in D^t:

$$V_i^t = var_i^t / \bar{h}(F_i \mid D^t) \text{ mit}$$
$$var_i^t = 1/m^t * \sum_j (h(F_i \mid D_j) - \bar{h}(F_i \mid D^t))^2,$$
$$\bar{h}(F_i \mid D^t) = 1/m^t * \sum_j h(F_i \mid D_j), \forall D_j \in D(F_i)^t. \tag{269}$$

Im Verfahrensteil “Kategorisierung” der Indexierung nach Crouch (1972[84]) wird die gesamte Dokumentenmenge D^t mit den core-terms reindexiert und dann geclustert. Im Verfahrensteil “Klassifikation” der Indexierung wird jedem Dokument D_j ein weiterer Kodierungsvektor, der aus den post-terms besteht, zugeordnet. Das Retrieval in einem solchen Ansatz besteht aus den Phasen “Grobrecherche”, die auf den Kategorisierungsvektoren basiert, und aus der “Feinrecherche”, die auf den Klassifikationsvektoren basiert.

3.5.2) Mono- und Polyrepräsentation der Indexierung im Kontext der Merkmalsgewichtungsmodelle

Im Kontext der Merkmalsgewichtungsmodelle kann die Indexierungsfunktion $A_{IR(D)}$ als Verkettung mehrerer Funktionen modelliert werden, welche als Quelle der Polyrepräsentation betrachtet werden. Ausgangspunkt ist eine Dokumentmenge D^t und eine Merkmals- bzw. Featuremenge F^t, wobei für jedes Merkmal F_j eine eindeutige dokumentmengenspezifische, absolute Häufigkeit $h(F_j \mid D^t)$ existiert, welche die Anzahl des Auftretens des Merkmals F_j in der gesamten Dokumentmenge zum Zeitpunkt t beschreibt. Dieser Wert wird durch eine Häufigkeitsfunktion berechnet, die D^t auf eine natürliche Zahl abbildet. Eine Dokumentmenge mit m^t Elementen kann als Teil des diskreten Raumes $D(\Theta)^{m(t)}$ betrachtet werden, sodass sich für die dokumentmengenspezifische, absolute Häufigkeitsfunktion die Definition ergibt:

$$h(F \mid D^t): D(\Theta)^{m(t)} \rightarrow N_0: D^t \mid\rightarrow h(F_j \mid D_i^t). \tag{270}$$

Aus der Häufigkeit $h(F_j \mid D^t)$ wird bei der Monorepräsentation genau eine dokumentmengenspezifische Merkmalsgewichtung $w(F_j \mid D^t)$ berechnet, indem genau eine dokumentmengenspezifische Gewichtungsfunktion $w(F \mid D^t)$ verwendet wird:

$$w(F \mid D^t): N_0 \text{ x } N \rightarrow R: (h(F_j \mid D^t), m^t) \mid\rightarrow w(F_j \mid D^t), \text{ z.B.}$$
$$w(F_j \mid D^t) = \log_e[1 + m^t / h(F_j \mid D^t)]. \tag{271}$$

Die dokumentmengenspezifische Merkmalsgewichtung $w(F_j \mid D^t)$ wird auf eine dokumentspezifische Merkmalsgewichtung $w(F_j \mid D_i^t)$ abgebildet, indem sie mit einer relativen Häufigkeit gewichtet wird. Es wird eine Funktion $w(F \mid D)$ eingeführt, welche das Gewicht eines Merkmals in einem Dokument bestimmt, im Gegensatz zu $w(F \mid D^t)$, die das Gewicht eines Merkmals in der Dokumentmenge D^t bestimmt. $w(F \mid D)$ soll als Funktion des Gewichtes $w(F_j \mid D^t)$ und der relativen Häufigkeit $r(F_j \mid D_i^t)$ des Merkmals in dem betrachteten Dokument definiert werden:

$$w(F \mid D): R \times R^+ \rightarrow R: (w(F_j \mid D^t), r(F_j \mid D_i^t)) \mapsto w(F_j \mid D_i^t), \text{ z.B.}$$
$$w(F_j \mid D_i^t) = w(F_j \mid D^t) * r(F_j \mid D_i^t). \quad (272)$$

Eine relative Häufigkeit $r(F_j \mid D_i^t)$ wird als eine Funktion der absoluten Häufigkeit $h(F_j \mid D_i^t)$ des Merkmals F_j im Dokument D_i^t bestimmt, wobei diese Funktion als $r(F \mid D)$ bezeichnet werden soll:

$$r(F \mid D): N_0 \rightarrow R^+: h(F_j \mid D_i^t) \mapsto r(F_j \mid D_i^t), \text{ z.B.}$$
$$r(F_j \mid D_i^t) = 1 + \log_e(h(F_j \mid D_i^t)). \quad (273)$$

Als unterste Stufe kann die Häufigkeitsfunktion $h(F \mid D)$ beschrieben werden, die einem betrachteten Dokument die Anzahl des Auftretens des Merkmals wie F_j zuordnet:

$$h(F \mid D): D(\Theta) \rightarrow N_0: D_i^t \mapsto h(F_j \mid D_i^t). \quad (274)$$

Die dokumentspezifische Merkmalsgewichtung $w(F_j \mid D_i^t)$ wird als j'te Komponente x_{ij} des Dokumentvektors x_{ij} verwendet.

Die Indexierungsfunktion $A_{IR(D)}$ ist somit allgemein definierbar als Funktion der beiden Funktionen $r(F \mid D)$ und $w(F \mid D^t)$ für alle n^t Merkmale, d.h. ein Dokument aus $D(\Theta)$ und eine Dokumentmenge aus $D(\Theta)^{m(t)}$ werden abgebildet auf einen n^t-dimensionalen Vektor x_i aus DVR, wobei die j'te Komponente x_{ij} gleich der dokumentspezifische Merkmalsgewichtung $w(F_j \mid D_i^t)$ ist:

$$A_{IR(D)} = f(r(F \mid D), w(F \mid D^t)): D(\Theta) \times D(\Theta)^{m(t)} \rightarrow DVR:$$
$$(D_i^t, D^t) \mapsto x_i = (x_{ij} = w(F_j \mid D_i^t) = f(r(F_j \mid D_i^t), w(F_j \mid D^t)) \mid j = 1, ..., n^t). \quad (275)$$

Die beiden Funktionen $r(F \mid D)$ und $w(F \mid D^t)$ besitzen das unmittelbare Potential zur Poyrepräsentation, da in Zobel & Moffat (1998[376]) eine Reihe von alternativen Funktionsbeschreibungen dargestellt werden. Demgegenüber besitzen $h(F \mid D)$ und $h(F \mid D^t)$ kein Potential zur Polyrepräsentation, da es sich dabei um Zählfunktionen handelt, welche die absolute Häufigkeit des Auftretens von Merkmalen ermitteln.

$w(F \mid D)$ als Funktion von $r(F \mid D)$ und $w(F \mid D^t)$ besitzt ein begrenztes Potential zur Polyrepräsentation, da es sich um eine Gewichtungsfunktion mit $r(F_j \mid D_i)$ als konkretem Gewichtungsfaktor und $w(F_j \mid D^t)$ als zu gewichtendem Faktor handelt.

Eine Polyrepräsentation der Indexierungsfunktion kann somit durch die Polyrepräsentation von $r(F \mid D)$ und $w(F \mid D^t)$ erzeugt werden, indem eine Funktion $r(F \mid D)_g$ aus der Menge $R(F \mid D)$ der möglichen bzw. zulässigen Häufigkeitsfunktionen und eine Funktion $w(F \mid D^t)_h$ aus der Menge $W(F \mid D^t)$ der möglichen bzw. zulässigen Gewichtungsfunktionen ausgewählt wird:

$$R(F \mid D) = \{r(F \mid D)_g: N_0 \rightarrow R^+: h(F_j \mid D_i^t) \mid\rightarrow r(F_j \mid D_i^t)_g \mid g = 1, ... \},$$
$$W(F \mid D^t) = \{w(F \mid D^t)_h: N_0 \text{ x } N \rightarrow R: (h(F_j \mid D^t), m^t) \mid\rightarrow w(F_j \mid D^t)_h \mid h = 1, ... \}. \quad (276)$$

Zusammen bilden die beiden ausgewählten Funktionen eine Indexierungsfunktion $A_{IR(D),k}$. Eine Indexierungsfunktions-Polyrepräsentation mit k = 1, ..., δ Funktionen lässt sich somit im Kontext der Merkmalsgewichtungsmodelle allgemein beschreiben als:

$$A_{IR(D),k} = f(r(F \mid D)_g, w(F \mid D^t)_h), r(F \mid D)_g \in R(F \mid D), w(F \mid D^t)_g \in W(F \mid D^t), k = 1, ..., \delta:$$
$$D(\Theta) \text{ x } D(\Theta)^{m(t)} \rightarrow DVR:$$
$$(D_i^t, D^t) \mid\rightarrow x_i = (x_{ij} = w(F_j \mid D_i^t)_k = f(r(F_j \mid D_i^t)_g, w(F_j \mid D^t)_h) \mid j = 1, ..., n^t). \quad (277)$$

3.6) Retrieval

Ziel des Retrievals ist die Identifizierung einer Teilmenge von Dokumenten aus D^t, die den externen Informationsbedürfnissen eines Agenten bezüglich eines speziellen Problems genügen sollen. Zu diesem Zweck wird standardmäßig von dem Agenten eine Anfrage an das IRS gestellt, die als Zeichensequenz formuliert werden soll. Picturale Anfragen, z.B. durch Spezifizierung von Bildteilen, oder akustische Anfragen, sollen nicht explizit behandelt werden, wobei diese Anfrageformen durch eine entsprechende Kodierung ebenfalls in Zeichensequenzen umwandelbar sind. Es existieren jedoch Argumente gegen die Verwendung einer Anfrage als Mittel der Interaktion zwischen Agent und IRS, die kurz dargestellt werden sollen. Trotz dieser Kritikpunkte sollen die Darstellungen des Retrievals in den nachfolgenden Abschnitten auf der Interaktion zwischen Agent und IRS durch Queries basieren.

Ein fundamentaler Kritikpunkt an diesem query-basierten IR-Ansatz ist, dass eine Query und ein Dokument unterschiedliche Texttypen sind, die in einem gemeinsamen Raum repräsentiert werden. Dies ist durch die Indexierungsfunktion mathematisch möglich, doch es stellt sich die Frage, wie Distanzen bzw. Ähnlichkeiten zwischen Repräsentationen unterschiedlicher Texttypen zu interpretieren sind.

Eine Query beschreibt ein externes Informationsbedürfnis, nicht jedoch einen faktischen Inhalt, der durch Aussagen beschrieben wird. Ein Dokument kann demgegenüber als eine zusammengesetzte Aussage interpretiert werden. Es stellt sich die prinzipielle Frage, ob eine indexierte Frage den gleichen Status besitzt wie eine indexierte Aussage, d.h. besitzt ein Queryvektor den gleichen Status wie ein Dokumentvektor.
Die Suche nach ähnlichen Dokumentvektoren in der topologischen Umgebung eines Queryvektors muss daher nicht notwendig so interpretiert werden, dass die nachgewiesenen Dokumente Lösungsvorschläge für das aktuelle Teilproblem darstellen, das zu der Formulierung des expliziten, externen Informationsbedürfnis geführt hat. D.h. die Distanz zwischen einem Queryvektor und einem Dokumentvektor muss nicht ein Maß für die Lösungsfähigkeit des Dokumentes für das zugrundeliegende Problem gewertet werden.

Standardmäßig wird eine Query behandelt wie eine Aussage, was sich durch die Verwendung der gleichen Indexierungsfunktion wie bei den Dokumenten ergibt, d.h. $A_{IR(Q)} := A_{IR(D)}$. Würde der unterschiedliche Status von Frage und Aussage jedoch ernst genommen, so muss der Agent einen anderen Texttyp für seine Interaktion mit einem IRS verwenden, der kompatibel zu dem Aussage-Status von Dokumenten ist. Naheliegend wäre die Spezifizierung von Textteilen aus einem oder mehreren Dokumenten, die dem Agenten bekannt sind. Der Status dieser Textteile wäre gleich dem Status nachgewiesener Dokumente, wobei die Menge an Textteilen einer impliziten Aufforderung wie „mehr zu diesen Themen“ entspricht.

In diesem Zusammenhang ist die Einbettung des IR-Prozesses in einen Problemlösungsprozess wertvoll (siehe auch Abschnitt 1.5)), da Texttypen, die bei einer Problemlösung eine Rolle spielen können, in das Blickfeld treten. Hierzu gehört die Beschreibung des Gesamtproblems, die strukturelle Zerlegung des Gesamtproblems in Teilprobleme und die Beschreibung dieser Teilprobleme. Neben der Beschreibung von Problemen gehört auch die Beschreibung von Problemlösungsvorschlägen, die für unterschiedliche Teilprobleme zu einem Zeitpunkt in einer unterschiedlichen Ausführlichkeit ausgearbeitet sein können. Bezüglich des allgemeinen Prinzips der Polyrepräsentation ist es günstig, für die Probleme jeweils mehrere, diversive Lösungsvorschläge zu erzeugen, die im Verlauf des Problemlösungsprozesses modifiziert und rekombiniert werden. Unter die Klasse „Lösungsvorschläge“ gehören auch Ergebnisse von Kreativtechniken (Facaoaru (1985[111]), Audehm (1995[13]), siehe auch Szenariotechniken (Reibnitz (1987[278]))) wie Mindmaps, Assoziationslisten und Brainstorming-Protokolle, wobei letztere insbesondere eine Rolle spielen, wenn eine Gruppe von Agenten mit der Problemlösung beschäftigt ist. Zu den potentiell vorhandenen Texttypen gehören auch Kommentare und Anmerkungen zu Dokumenten, die einem Agenten bekannt sind, oder annotierte Literaturlisten zu dem betrachteten Problembereich. Ist der Agent Teil einer Problemlösungsgruppe, bzw. einer Gruppe, deren Mitglieder ähnliche Probleme lösen sollen, so können weitere Texttypen wie Diskussionslisten u.ä. herangezogen werden.

Wird ein Gesamtproblem in eine Struktur von Teilproblemen zerlegt, so kann sich die Struktur sowie die Formulierung der einzelnen Teilprobleme im Verlauf des Problemlösungsprozesses ändern. Werden einzelne Teilprobleme endgültig gelöst, so kann zwischen dem Gesamtproblem und dem zu einem Zeitpunkt verbleibendem Restproblem unterschieden werden.

Da Dokumente als Lösungsvorschläge für ein aktuelles Teilproblem verwendet werden sollen, liegt es nahe, explizite Problemlösungsvorschläge bzw. Hypothesen durch den Agenten als Substitut für Queries zu verwenden, da es sich dabei um Aussagen handelt. Die Indexierung eines solchen Lösungsvorschlages erzeugt einen Vektor in DVR, in dessen Umgebung nach Dokumentvektoren mit den bekannten Retrieval-Strategien gesucht wird, wobei die Ähnlichkeit zwischen dem Lösungsvorschlagsvektor und einem Dokumentvektor nun korrekt als Ähnlichkeit zwischen zwei Texten gleichen Typs interpretiert werden kann. D.h. gesucht wird nach einem anderen, ähnlichen Lösungsvorschlag in Form eines Dokumentes, der eventuell eine Bewertung des Lösungsvorschlages durch den Autor des Dokumentes enthält, noch offene Detailfragen behandelt oder weitere Informationen liefert, die dem Agenten bislang unbekannt waren, und die für den Problemlösungsprozess oder den Reformulierungsprozess nützlich sein können.

Der Zweifel an dem gleichen Status von indexierten Fragen und Aussagen bedeutet jedoch nicht, dass eine Verwendung in dem gleichen Vektorraum DVR keine sinnvollen Ergebnisse erzeugen kann. Da alle vektorraumbasierten Verfahren auf dieser Annahme aufbauen, und da diese Verfahrensklasse erfolgreich in IRS eingesetzt wird, bedeutet dies, dass die gemeinsame Verwendung von Query- und Dokumentvektoren in entsprechendem Maße nützlich angewendet werden kann. Dies ergibt sich zum einen daher, dass durch das gleiche Indexierungsverfahren $A_{IR(Q)} := A_{IR(D)}$ die Frage quasi als Aussage behandelt wird. Zum anderen ergibt sich dies daher, dass die Gewichtungsmethoden innerhalb der Indexierungsverfahren eine Form von Soft-Computing implementieren, mit der Unsicherheit und Unschärfe in gewissem Umfang modelliert werden können, was aus der Verwendung von Häufigkeiten und Gewichtungsfaktoren folgt. Insbesondere das Soft-Computing-Argument legt die Vermutung nahe, dass bei der Verwendung einer speziellen Indexierungsfunktion, angewendet auf einen passenden Texttyp wie einen Problemlösungsvorschlag, sich ein IRS mit besseren Performancewerten ergeben könnte, als bei der Verwendung der gleichen Indexierungsfunktion auf einen nicht passenden Texttyp.

Eine andere Richtung der Kritik an der Verwendung von Fragen betrifft die Fähigkeit bzw. eingeschränkte Fähigkeit des Agenten Sachverhalte zu spezifizieren, die ihm unbekannt sind. Eine Frage ist eine Beschreibung dessen, was dem Agenten unbekannt ist, während andere Texttypen wie ein Lösungsvorschlag eine Beschreibung dessen ist, was der Agent kennt, da er sonst den entsprechenden Sachverhalt nicht als Lösung für ein Problem anbieten könnte, wenn man von einer gleichverteilt zufälligen Zuordnung von Lösungsvorschlägen absieht.

3.6.1) Query als Zeichensequenz

3.6.1.1) Query-Monorepräsentation

Vereinfachend soll gelten, dass eine Anfrage oder Query Q_i^t mit Hilfe des gleichen Alphabetes Θ formuliert wird, in denen die Dokumente formuliert sind, sodass gilt:

$$Q_i^t = (d_{ij} \mid \forall d_{ij} \in \Theta; j = 1, ..., \#Q_i^t) \in Q(\Theta). \quad (278)$$

Dabei bezeichnet $Q(\Theta)$ analog wie $D(\Theta)$ bei den Dokumenten die Menge aller Anfragen, die mit Hilfe des Alphabetes Θ formuliert werden können, unabhängig von der Länge der Anfrage.

3.6.1.2) Query-Polyrepräsentation

3.6.1.2.1) Polyrepräsentation durch multiple Queryformulierung eines Agenten

Im Rahmen einer Polyrepräsentation ergibt sich bei der Anfrage die Besonderheit, dass ein Agent durch das IRS aufgefordert werden kann, sein externes Informationsbedürfniss durch mehrere, alternative Fragen zu formulieren, die im weiteren indexiert und zum Retrieval verwendet werden. Dieser Idealfall besitzt zumindest bei menschlichen Agenten eine enge Grenze, da diese Vorgehensweise eine mentale Bereitschaft zur Mehrarbeit erfordert. Analog wie bei der mentalen Bereitschaft zur Bewertung von Dokumenten, die vom IRS als potentiell relevant angegeben werden, im Rahmen des Relevanz-Feedbacks (siehe Abschnitt 3.9)), wird dies nur dann erfolgen, wenn der erwartete Gewinn den damit verbundenen Aufwand übersteigt.

D.h. wenn der Aufwand durch die allgemeine rationale Entscheidung

$$p * G - C > 0 \quad (279)$$

innerhalb des gegebenen Problemlösungsprozesses gerechtfertigt werden kann, mit p als der Wahrscheinlichkeit, dass die Informationsbedürfnisse erfüllt werden, G als dem daraus erwachsenden Erlös, und C als den damit verbundenen Kosten (siehe Anderson (1990[8], 1993[10])). Im Regelfall muss jedoch von genau einer Frage Q_i^t ausgegangen werden, die durch den Agenten formuliert wird, sodass eine Polyrepräsentation ein, durch das IRS durchgeführtes Verfahren, anstatt ein, durch den Agenten durchgeführtes Verhalten erfolgen muss.

3.6.1.2.2) Polyrepräsentation durch kollaborative Queryformulierung einer Agentengruppe

Bei einer kollaborativen Strategie im Rahmen eines IRS wird eine Gruppe AgM von n_{agent} Agenten unterstellt, die das gleiche bzw. ein hinreichend ähnliches Informationsbedürfnis besitzen. Wird das IRS im Kontext eines Problemlösungsprozesses gesehen, so handelt es sich bei dieser Agentenmenge entweder um eine explizite, lokale Arbeitsgruppe oder um eine verteilte Community, welche die gleichen (wissenschaftlichen) Interessen teilen. Eine solche Community kann explizit vorliegen, indem z.B. die Mitglieder einer Mailingliste mit einem speziellen, engen Themenbereich als eine solche Community verstanden wird. Andererseits kann eine Community durch einen unüberwachten Prozess, d.h. eine Clusteranalyse, generiert werden, indem beispielsweise langfristig die Interaktionen mit einem IRS analysiert werden, wobei eine Clusterung der Queryvektoren und somit der sie formulierenden Agenten durchgeführt wird.

Im weiteren soll von einer expliziten, lokalen Arbeitsgruppe ausgegangen werden, was den Vorteil hat, dass eine direkte Motivation bezüglich der Formulierung von Queries und Relevanzbewertungen unterstellt werden kann. Jeder der Agenten Ag_i erzeugt eine eigene Query Q_i^t zu dem gemeinsamen Informationsbedürfnis, wodurch unterschiedliche Präferenzen, Perspektiven, Vorwissen u.ä. eingebracht werden. Analog läuft dies auch für andere Texttypen wie Problembeschreibung, Pröblemlösungsvorschläge u.ä., was im Abschnitt 3.9.1) thematisiert wird. Eine solche Query wird mit Hilfe einer gemeinsamen Indexierngsfunktion $A_{IR(Q)}$ zu dem Queryvektor q_i^t indexiert. Die Querymenge $QM^t = \{Q_i^t \mid i = 1, ..., n_{agent}\}$ wird auf diese Weise auf eine Queryvektorenmenge $QVM^t = \{q_i^t \mid i = 1, ..., n_{agent}\}$ abgebildet.
Diese Form der Polyrepräsentation kann als eine natürliche Polyrepräsentation im Gegensatz zu den stochastisch erzeugten, künstlichen Polyrepräsentationsformen verstanden werden.

3.6.1.2.3) Polyrepräsentation durch Moving-Blocks-Bootstrap

Eine weitere Möglichkeit der Query-Polyrepräsentation kann in Analogie zu der Dokument-Polyrepräsentation durch ein Moving-Blocks-Bootstrap-Verfahren durchgeführt werden, das als Funktion movB definiert ist:

$$\text{movB: } D(\Theta) \rightarrow D(\Theta):$$

$$Q_i^t \mid\rightarrow Q_{ik}^t = (d_{ij} \mid \forall d_{ij} \in \Theta; j = 1, ..., \#Q_{ik}^t) \in Q(\Theta), k = 1, ..., \delta. \quad (280)$$

Die Original-Query wird mit den künstlichen Fragen in einer Bootstrapmenge BQ_i^t gesammelt:

$$BQ_i^t = \{Q_i^t, Q_{ik}^t \mid k = 1, ..., \delta\}. \quad (281)$$

Um die künstlichen Queries zu erzeugen, werden aus der Original-Query Q_i^t zunächst alle vorliegenden, zusammenhängenden Teillisten QL_{ip}^t mit jeweils l Elementen erzeugt:

$$QL_{i1}^t = (d_{i1}^t, d_{i2}^t, ..., d_{il}^t),$$
$$QL_{i2}^t = (d_{i2}^t, d_{i3}^t, ..., d_{il+1}^t),$$
$$...$$
$$QL_{i,\#Q(i)-l+1}^t = (d_{i,\#Q(i)-l}^t, d_{i,\#Q(i)-l+1}^t, ..., d_{i,\#Q(i)}^t). \quad (282)$$

Die einzelnen Teillisten werden in die Bootstrap-Grundelementliste BQL_i^t eingefügt:

$$BQL_i^t = \{QL_{ip}^t \mid p = 1, ..., \#Q_i^t - l + 1\}. \quad (283)$$

Aus dieser Grundliste werden Teillisten zufällig mit Zurücklegen gezogen, und in der Reihenfolge der Ziehung in eine Zeichenliste eingefügt, die jeweils eine abgeleitete, künstliche Query Q_{ik}^t repräsentiert. Die Länge $\#Q_{ik}^t$ der künstlichen Queries ergibt sich allgemein aus der Multiplikation der Anzahl $\mu_{B(Q)}$ der Ziehungen und der Blockgröße:

$$\#Q_{ik}^t = \mu_{B(Q)} * l. \quad (284)$$

Die Anzahl der Ziehungen $\mu_{B(Q)}$ kann aus zwei Szenarien abgeleitet werden:

1) Ableitung aus Original-Query, indem der Quotient aus der Querylänge $\#Q_i^t$ und dem Blockgrößen-Parameter l gebildet wird:

$$\mu_{B(Q)} = \text{round}_+(\#Q_i^t/l). \quad (285)$$

2) Ableitung aus den Längen der Dokumente aus der Dokumentliste D^t, z.B. durch die Verwendung der durchschnittlichen Dokumentlänge $\#\overline{D}^t$:

$$\mu_{B(Q)} = \text{round}_+(\#\overline{D}^t/l), \text{ und}$$
$$\#\overline{D}^t = 1/m^t * \sum_i \#D_i^t, \forall D_i^t \in D^t. \quad (286)$$

Das zweite Szenario ergibt sich aus der prinzipiellen Möglichkeit mit Hilfe des Moving-Blocks-Bootstraps beliebig lange Zeichensequenzen zu erzeugen. Dieser Ansatz ist im Rahmen des vektorraumbasierten IR sinnvoll, da die Query als ein Dokument interpretiert wird, das in den metrischen Dokumentvektorraum DVR abgebildet werden kann. In der Regel ist die Query als Zeichensequenz jedoch wesentlich kürzer als ein Dokument aus D^t, sodass die Interpretation einer Query als Dokument effektiv wird, wenn künstliche Queries erzeugt werden können, die in ihrer Länge den Dokumentlängen vergleichbar sind.

In diesem Zusammenhang stellt sich jedoch die Frage nach der Verwendung Orginal-Query Q_i^t, da deren Länge $\#Q_i^t$ von der Länge der künstlichen Queries stark abweicht. Durch die nachfolgende Indexierung können unterschiedlich lange Zeichensequenzen in den Dokumentraum abgebildet werden, sodass die Original-Query wie oben dargestellt, neben den künstlichen Queries verwendet werden kann, oder es werden nur die künstlichen Queries in der Bootstrapmenge BQ_i^t gesammelt:

$$BQ_i^t = \{Q_i^t, Q_{ik}^t \mid k = 1, ..., \delta\}, \text{ oder}$$
$$BQ_i^t = \{Q_{ik}^t \mid k = 1, ..., \delta+1\}. \quad (287)$$

3.6.1.2.4) Polyrepräsentation durch Mutations-Operationen

Eine andere Möglichkeit aus einer Query Q_i^t eine Menge verwandter Queries abzuleiten ist Überlagerung der Query mit einem Zufallsprozess, mit dem die Unsicherheit in der Frageformulierung modelliert werden soll. Dies kann als Mutations-Operation interpretiert werden, indem einzelne Komponenten der Zeichensequenz durch andere Komponenten ausgetauscht werden. Als Komponente kann im einfachsten Fall die Zeichenebene verwendet werden, sodass eine Mutation der Query einer Punktmutation entspricht, bei der mit einer bestimmten, extern vorgegebenen Wahrscheinlichkeit einzelne Zeichen aus Q_i^t durch ein anderes Zeichen ersetzt werden. Diese Vorgehensweise wird bei der Modellierung der Mutation im Genetic-Programming verwendet (Koza (1992[188], 1994[189]), Koza et al. (1999[191]), Banzhaf et al. (1998[26]), Nordin (1997[238])), unabhängig ob als Datenstruktur eine Zeichensequenz oder eine Baumstruktur verwendet wird.

Als Komponente könnte auch eine Teilsequenz wie ein Block im Rahmen des Moving-Blocks-Bootstrap verwendet werden, d.h. es wird ein oder mehrere Blocks in einer Zeichensequenz selektiert, und durch einen anderen Block aus der Bootstrap-Grundelementliste BQL_i^t ersetzt.

Die Mutations-Operation im Rahmen des Genetic-Programming wird immer als gleichverteilte Zufallsoperation definiert, d.h. wird ein Zeichen zur Mutation ausgewählt und existieren z.B. 20 Zeichen in dem verwendeten Alphabet Θ, so wird das betreffende Zeichen mit der gleichen Wahrscheinlichkeit von 1/20 durch ein beliebiges anderes Zeichen des Alphabetes, inklusive sich selbst, ersetzt. Alternativ zu der gleichverteilten Zufallsoperation kann eine Übergangsmatrix für diesen Zweck verwendet werden, in der einzelne Zellen angeben, wie wahrscheinlich der Übergang eines bestimmten Zeichens in ein anderes Zeichen ist. Die Summe der Zellen jeder Spalte bzw. jeder Zeile ist gleich Eins. Es stellt sich dabei die Frage, wie eine solche Übergangsmatrix erzeugt werden soll, wobei die Interpretation einer Zeichensequenz als Markov-Prozess eine wichtige Rolle spielt (Buchholz (1991[60]), Sinclair (1992[311]), Davis (1993[92]), Buchholz et al. (1994[61]), Elliott et al. (1995[109])).

3.6.1.2.5) Polyrepräsentation durch Markov-Prozesse

Hierzu sind Verfahren anwendbar, die im Zusammenhang mit Markov-Ketten stehen (Buchholz (1991[60]), Sinclair (1992[311]), Davis (1993[92]), Buchholz et al. (1994[61]), Elliott et al. (1995[109])), und einen Bezug zu den Sprachgenerierungsmodellen besitzen, die bei der Dokument-

Polyrepräsentation angesprochen wurden (siehe Abschnitt 3.2)). D.h. eine Möglichkeit der Polyrepräsentation der Query ist die Erzeugung eines Sprachgenerierungsmodells aus der Query Q_i^t und die Erzeugung einer Menge von Zeichensequenzen mit diesem Sprachgenerierungsmodell, die jeweils als künstliche Query Q_{ik}^t verwendet wird. Das Problem besteht jedoch darin, dass eine Query eine viel zu kleine Zeichensequenz ist, um damit eine vollständige Übergangsmatrix zu bestimmen. Für die Vielzahl der unbesetzten Zellen in der Matrix stellt sich die Frage, ob sie mit Null besetzt werden sollen, einem Defaultwert, oder ob eine Wahrscheinlichkeitsverteilung unterstellt werden soll, mit der für jede Zelle ein zufälliger, kleiner Wert generiert werden soll, der für jede Zelle unabhängig durch ein Zufallsexperiment ermittelt wird.

Eine Alternative wäre die Verwendung von anderen Zeichensequenzen des Agenten, die im Kontext seines Problems stehen, und das die Ursache für sein externes Informationsbedürfnis ist. Bei diesen Texten handelt es sich um die Beschreibung des Gesamtproblems und Teilprobleme, Lösungsvorschläge, Ideen, Assoziationen, Kommentare zu anderen Texten, Diskussionen mit anderen Agenten über das Problem u.ä. (siehe auch Abschnitt 3.9.7)). Diese Texte können als Korpus zusammen mit der Query verwendet werden, um eine gemeinsame Übergangsmatrix zu erzeugen, wobei jedoch ebenfalls nicht garantiert werden kann, dass die Matrix voll besetzt ist.

Eine weitere Alternative wäre die Verwendung eines Dokument-Korpus-Modells, das auf der Basis aller verfügbaren Dokumente in D^t erstellt wurde, und das zur Dokument-Polyrepräsentation herangezogen wurden (siehe Abschnitt 3.2)). Unabhängig von der Größe des Korpus kann nicht garantiert werden, dass daraus eine vollständig besetzte Matrix folgt. Dieses Dokument-Korpus-Modell kann durch ein reines Query-Modell überlagert bzw. adaptiert werden, wobei das sich ergebende Modell als gemischtes Query-Modell verwendet wird, um eine Menge von künstlichen Queries zu generieren, die als Query-Polyrepräsentation Verwendung finden (siehe Ponte & Croft (1998[261]) und Song & Croft (1999[318]) für Ansätze der Sprachmodelle im IR).

3.6.1.2.6) Polyrepräsentation durch Rekombinations-Operationen

Liegen bereits mehrere Queries vor, so lässt sich daraus eine größere Menge von Queries erzeugen, indem Rekombinations-Operationen eventuell in Kombination mit Mutations-Operationen angewendet werden, wodurch eine größere Query-Polyrepräsentation erzeugt wird. Wird eine natürliche Polyrepräsentation durch multiple Queryformulierung eines Agenten oder durch kollaborative Queryformulierung einer Agentengruppe unterstellt, so kann davon ausgegangen werden, dass die Anzahl der Queries nicht groß genug ist, um den Anforderungen einer stochastischen Polyrepräsentation zu genügen, wie sie z.B. von Resampling-Verfahren erfüllt werden. In einem solchen Kontext wäre es sinnvoll, aus der natürlichen, kleinen Polyrepräsentation eine künstliche bzw. gemischte, große Polyrepräsentation zu generieren. Dies gelingt z.B. durch die Anwendung von Rekombinations-Operationen auf die Zeichensequenzen der Elemente der vorliegenden Querymenge, die als Elternmenge QEM_i^t bezeichnet werden soll, wobei implizit unterstellt wird, dass es sich um eine Polyrepräsentation durch multiple Queryformulierung eines Agenten Ag_i handelt.

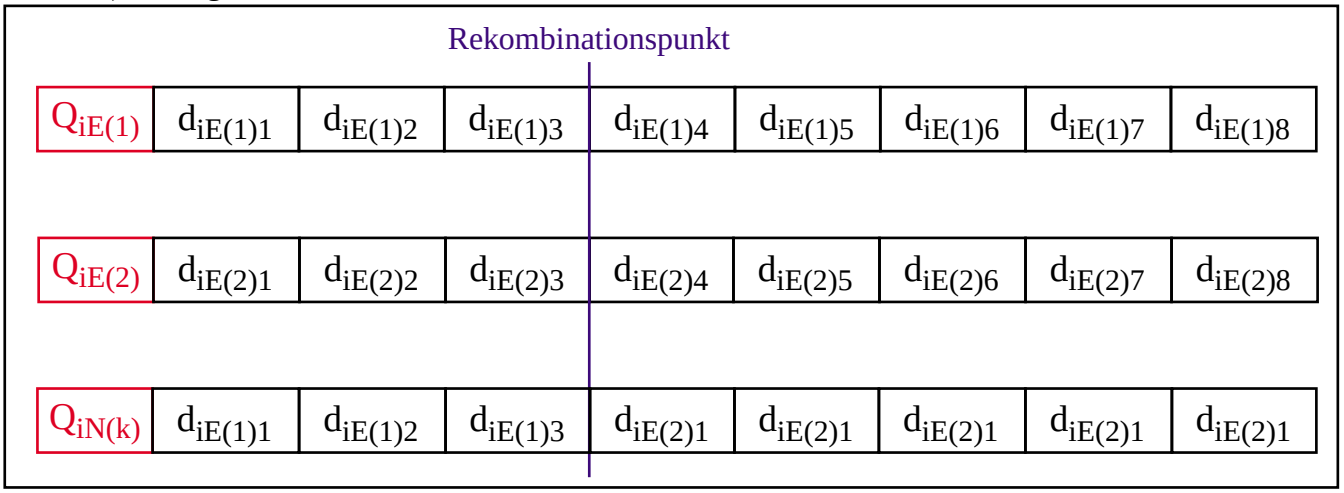
$$QEM_i^t = \{Q_{iE(j)}^t \mid j = 1, ..., \mu_E\}. \quad (288)$$

Vorgestellt wird eine einfache zwei-geschlechtliche Rekombination, bei der zwei Elemente $Q_{iE(1)}{}^t$ und $Q_{iE(2)}{}^t$ aus $QEM_i{}^t$ mit Zurücklegen gezogen werden, gefolgt von der Rekombinations-Operation, sodass eine Nachkommen-Zeichensequenz $Q_{iN(k)}{}^t$ erzeugt wird. Allgemein wird dies formuliert als:

$$Q_{iN(k)}{}^t = rec(Q_{iE(1)}{}^t, Q_{iE(2)}{}^t). \tag{289}$$

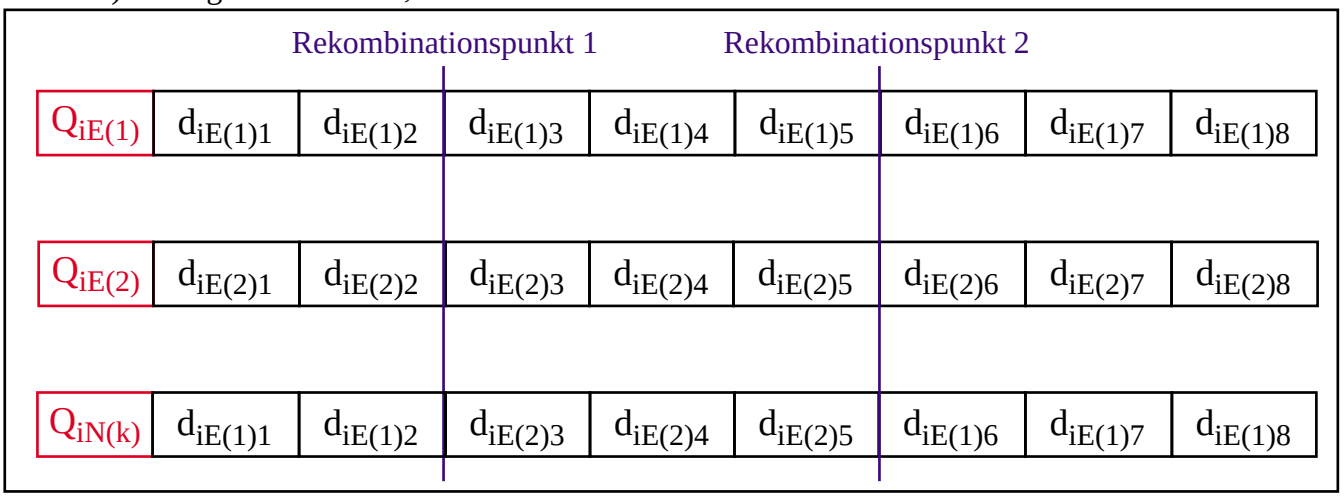
Besitzen die einzelnen Zeichensequenzen $Q_{iE(j)}{}^t$ die gleiche Anzahl von Zeichen, und soll diese Anzahl bei der Rekombination erhalten bleiben, so ist eine diskrete Rekombination sinnvoll. Die einfachste zwei-geschlechtliche diskrete Rekombination verwendet einen 1-Punkt-Mechanismus, d.h. es wird zunächst innerhalb der Zeichensequenz ein Punkt festgelegt, der für beide Sequenzen gilt (siehe Abb. 33)). Danach wird gleichverteilt zufällig entschieden, ob alle Zeichen, die links von dem Rekombinationspunkt liegen, von dem ersten oder zweiten Elternteil stammen sollen. Im betrachteten Beispiel sollen die Zeichen von dem ersten Elternteil stammen, sodass die Zeichen rechts von dem Elternteil von dem zweiten Elternteil übernommen werden.

Abb. 33) Zwei-geschlechtliche, diskrete 1-Punkt-Rekombination



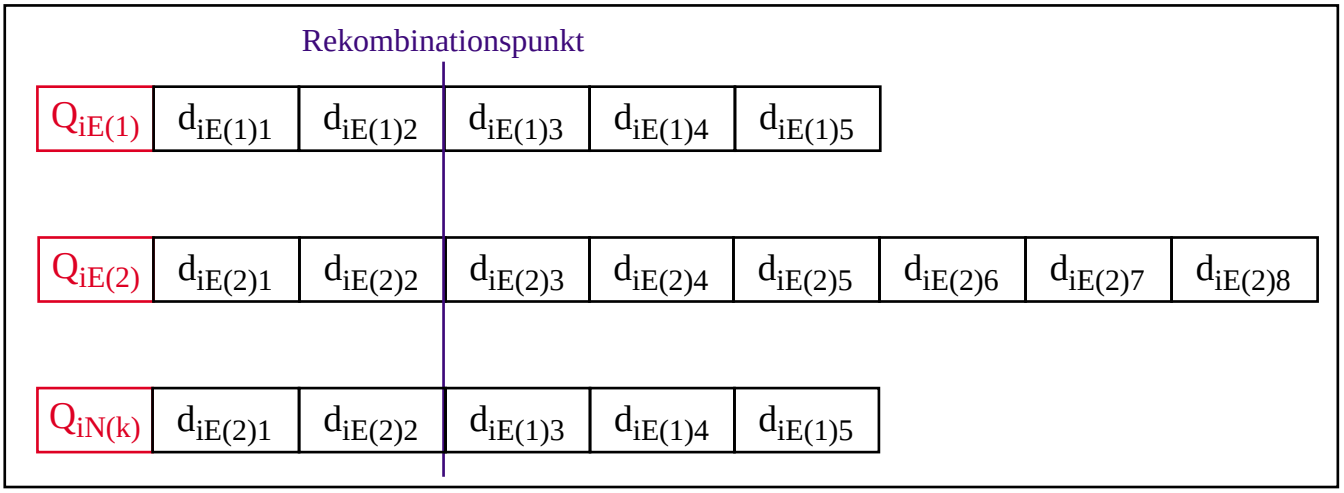
Verallgemeinern lässt sich diese diskrete Rekombinations-Operation durch die Mehr-Punkt-Rekombination (siehe Abb. 34)), wobei wiederum zwei Eltern verwendet werden.

Abb. 34) Zwei-geschlechtliche, diskrete Mehr-Punkt-Rekombination



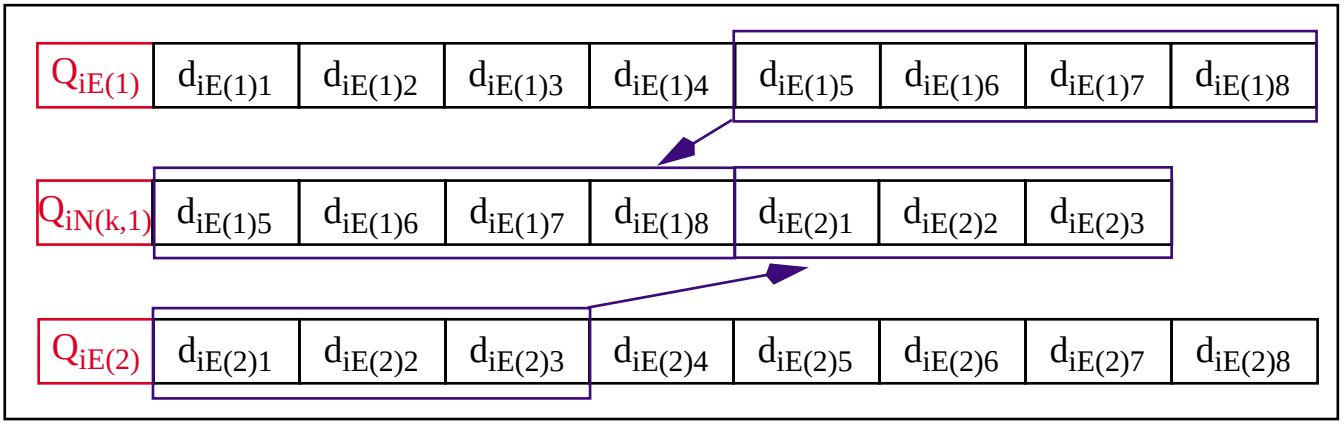
Zunächst werden die d > 1 Rekombinationspunkte bei jeder Rekombinationsoperation zufällig in der Zeichensequenz verteilt, wobei d + 1 Teilsequenzen entstehen. Bei jeder Teilsequenz wird durch ein unabhängiges Zufallsexperiment entschieden, welches Elternteil seine Teilsequenz kopieren darf, wobei im Fall zweier Eltern eine jeweilige Wahrscheinlichkeit von 1/2 für das entsprechende Ereignis verwendet wird. Diese Formen der diskreten Rekombination erzeugen aus Zeichensequenzen gleicher Länge eine neue Zeichensequenz der gleichen Länge. Sind die beiden Elternteile jedoch unterschiedlich lange, wovon bei Queries ausgegangen wird, so soll die freie Wahl des Rekombinationspunktes beschränkt werden auf die kleinere der beiden Sequenzen (siehe Abb. 35)). Als Nachkomme entsteht eine Sequenz, die entweder die Länge des ersten oder zweiten Elternteils besitzt, jenachdem ob der erste oder zweite Elternteil länger ist, und ob die erste oder zweite Teilsequenz des ersten Elternteils als erste Teilsequenz des Nachkommen durch den Zufallsprozess ausgewählt wird.

Abb. 35) Diskrete 1-Punkt-Rekombination bei unterschiedlich langen Sequenzen



Eine 1-Punkt-Crossover-Rekombination kann als diskrete Rekombinations-Operation interpretiert werden, bei der ein Rekombinationspunkt auf den beiden Zeichensequenzen der Eltern unabhängig voneinander zufällig ausgewählt wird (siehe Abb. 36)). Die zweite Teilsequenz des ersten Elternteils wird dann als erste Teilsequenz des Nachommen verwendet, und die erste Teilsequenz des zweiten Elternteils wird als zweite Teilsequenz des Nachkommen verwendet. Bei Crossover-Operationen werden meist bei einer Operation zwei Nachkommen erzeugt, wobei dieser die erste Teilsequenz des ersten Elternteils und die zweite Teilsequenz des zweiten Elternteils erhält.

Abb. 36) Zwei-geschlechtliche 1-Punkt-Crossover-Operation



Wie sich an dem Beispiel zeigt, entstehten durch Crossover-Operationen Nachkommen, die zueinander und im Vergleich zu den Eltern unterschiedlich lange Zeichensequenzen besitzen. Dieser Operationstyp funktioniert in der gleichen Weise, wenn die Eltern unterschiedlich lange Zeichensequenzen besitzen. Beides ist im gegebenem Kontext wichtig, da die Queries standardmäßig als unterschiedlich lange Zeichensequenzen angenommen werden.

Die Rekombinations-Operationen lassen sich auch auf einer aggregierten Ebene anstatt der Zeichenebene durchführen, indem Worte oder bei längeren Texten Sätze als Grundbausteine verwendet werden. Die Rekombinations-Operationen dürfen in diesem Fällen nur zwischen den einzelnen Grundelementen Rekombinationspunkte setzen, d.h. zwischen Wörtern bzw. zwischen Sätzen. Wird eine $(\mu_E+\lambda)$-EA unterstellt, so werden λ Nachkommen erzeugt, die zusammen mit den Eltern die Queryvektorenmenge QVM_i^t bilden:

$$QVM_i^t = \{q_{iE(j)}^t, q_{iN(k)}^t \mid j = 1, ..., \mu_E; k = 1, ..., \lambda\}. \quad (290)$$

Diese Rekombinations-Operationen können durch zusätzliche Punkt-Mutationen ergänzt werden, indem ein einzelnes Zeichen durch ein anderes Zeichen aus dem verwendeten Alphabet ersetzt wird. Wird auf der Ebene der Zeichen keine Struktur zwischen den einzelnen Zeichen angenommen, so wird eine gleichverteilt zufällige Ersetzung durchgeführt. Wird ein Korpus wie die zugrundeliegende Dokumentmenge analysiert, so kann eine Zeichensequenz als Markov-Kette betrachtet werden, bei der nachfolgende Zeichen einer expliziten Übergangsstruktur unterliegen, welche durch Übergangswahrscheinlichkeiten beschrieben werden können. Diese Wahrscheinlichkeiten können für eine Punkt-Mutation genutzt werden, indem Übergangswahrscheinlichkeiten als Ersetzungswahrscheinlichkeiten verwendet werden. Werden Wörter als Grundelemente verwendet, und liegt ein kontrolliertes Vokabular in Form einer Liste erlaubter Wörter vor, so kann auch auf dieser Ebene eine Punkt-Mutation durchgeführt werden, indem ein Wort durch ein anderes ersetzt wird. Auch in diesem Fall können Markov-Ketten und Übergangswahrscheinlichkeiten genutzt werden, wobei die Übergangsmatrix aufgrund der vielen Wörter in dem kontollierten Vokabular sehr groß ist. Dies setzt einen sehr großen Korpus zur Analyse voraus und ist entsprechend aufwendig, ohne dass garantiert werden kann, dass alle Zellen in der Übergangsmatrix besetzt werden können.

Werden Wörter als Merkmale betrachtet, so besteht die Möglichkeit im Merkmalsraum einen Merkmalsgraph z.B. durch ein GNG-SOM-Verfahren zu bilden, der zu einem Spreading-Activation-Graph erweitert werden kann (siehe Abschnitt 2.1.8) zur Aktivitätsausbreitung in GNG-Graphen und Abschnitt 3.8.6)). Die gerichteten Verbindungen zwischen den einzelnen Knoten eines solchen GNG-SOM-Merkmalsgraphen lassen sich auch stochastisch interpretieren, wobei die Übergangswahrscheinlichkeiten im Graph wieder als Ersetzungswahrscheinlichkeiten bei einer Punkt-Mutation verwendet werden können.

3.6.1.2.7) Polyrepräsentation durch GNG-SOM-Merkmalsgraphen

Weitere Möglichkeiten der Polyrepräsentation einer Query ergeben sich, wenn im Merkmalsraum ein Merkmalsgraph z.B. durch ein GNG-SOM-Verfahren gebildet wurde, der zu einem Spreading-Activation-Graph erweitert wird (siehe Abschnitt 2.1.8)), was im Abschnitt 3.8.6) näher dargestellt werden soll.

3.6.2) Query-Indexierung und Queryvektor-Mono- und Polyrepräsentation

3.6.2.1) Queryvektor-Monorepräsentation

Im ersten Arbeitsschritt des Retrievals wird die Zeichensequenz Q_i^t durch eine Funktion auf einen Queryvektor q_i^t abgebildet. Dies bedeutet faktisch eine Query-Indexierung, und die Indexierungsfunktion soll durch $A_{IR(Q)}$ bezeichnet werden. Auf diese Weise wird ein Queryvektorraum $QVR \subseteq R^{n(Q,t)}$ definiert, in dem Q_i durch einen Punkt repräsentiert wird:

$$A_{IR(Q)}: Q(\Theta) \rightarrow QVR: Q_i^t \mid\rightarrow q_i^t = (q_{ij}^t \in R \mid j = 1, ..., n_Q^t). \tag{291}$$

Es soll im weiteren vereinbart werden, dass der Queryvektor q_i^t genau wie die Dokumentvektoren n^t-dimensional sein soll, wobei nicht nur die gleiche Anzahl von Merkmalen verwendet werden, sondern auch die gleichen Einzel-Merkmale und die gleiche Reihenfolge der Merkmale in den entsprechenden Vektoren. Auf diese Weise wird der Queryvektorraum QVR gleich dem Dokumentvektorraum DVR gewählt, und Q_i wird als ein Punkt in DVR repräsentiert:

$$A_{IR(Q)}: Q(\Theta) \rightarrow DVR: Q_i \mid\rightarrow q_i^t = (q_{ij}^t \in R \mid j = 1, ..., n^t). \tag{292}$$

Würden zwei unterschiedliche Alphabete Θ_D und Θ_Q verwendet werden, so wäre eine Bearbeitung dann möglich, wenn eine Indexierungsfunktion $A_{IR(Q)}$ existiert, die eine Q_i-Zeichensequenz in den Dokumentvektorraum DVR abbildet, d.h. an diese Indexierungsfunktion werden andere Anforderungen gestellt als an $A_{IR(D)}$, wohingegen bei der Verwendung des gleichen Alphabetes die beiden Indexierungsfunktionen im wesentlichen gleich sein können:

$$A_{IR(Q)}: Q(\Theta_Q) \rightarrow DVR: Q_i \mid\rightarrow q_i^t. \tag{293}$$

3.6.2.2) Queryvektor-Polyrepräsentation

Im vorangegangenen Abschnitt wurde eine Query Q_i^t als genau ein Queryvektor q_i^t in DVR durch eine Indexierungsfunktion $A_{IR(Q)}$ repräsentiert, wobei unterstellt wird, dass genau ein DVR mit genau einer Dokumentvektorenverteilung DV^t vorliegt. Bei einer Polyrepräsentation soll eine Menge von Queryvektoren erzeugt werden:

$$QVM_i^t = \{q_{ik}^t \mid k = 1, ..., \delta+1\}. \tag{294}$$

Es existieren wiederum unterschiedliche Möglichkeiten, diese Queryvektor-Polyrepräsentation zu erzeugen, wobei in den nächsten Abschnitten verschiedene Klassen unterschieden werden, die sich auf die vorliegende Query bzw. einer bereits vorliegenden Query-Polyrepräsentation beziehen.

3.6.2.2.1) Query-Polyrepräsentation und Indexierungsfunktions-Monorepräsentation

Liegt bereits eine Query-Polyrepräsentation mit einer Querymenge vor, wie z.B. eine Bootstrapmenge BQ_i^t mit einer Original- und δ künstlichen Queries, so kann die Menge QVM_i^t einfach erzeugt werden, indem mit Hilfe genau einer Indexierungsfunktion $A_{IR(Q)}$ die Elemente aus QVM_i^t indexiert werden:

$$A_{IR(Q)}: Q(\Theta) \rightarrow DVR:$$
$$Q_i^t \mid\rightarrow q_i^t \wedge Q_{ik}^t \mid\rightarrow q_{ik}^t, \forall Q_{ik}^t \in BQ_i^t. \quad (295)$$

3.6.2.2.2) Query-Monorepräsentation und Indexierungsfunktions-Polyrepräsentation

Liegt nur die Original-Query vor, so besteht die Möglichkeit der Polyrepräsentation in der Spezifizierung einer Menge $IM_{IR(Q)}$ von Indexierungsfunktionen $A_{IR(Q,k)}$, die Q_i^t mehrfach unabhängig indexieren, ohne dass hier auf die Herkunft bzw. der Ableitung von $A_{IR(Q,k)}$ aus einer Original-Indexierungsfunktion $A_{IR(Q)}$ eingegangen werden soll:

$$A_{IR(Q,k)}: Q(\Theta) \rightarrow DVR: Q_i^t \mid\rightarrow q_{ik}^t, \forall A_{IR(Q,k)} \in IM_{IR(Q)}. \quad (296)$$

3.6.2.2.3) Query-Monorepräsentation und stochastische Indexierungsfunktion

Eine dritte Möglichkeit der Erzeugung von abgeleiteten Queryvektoren q_{ik}^t aus genau einem Queryvektor q_i^t und genau einer Indexierungsfunktion besteht dann, wenn die Indexierungsfunktion, die bislang als deterministischer Prozess modelliert wurde, durch einen stochastischen Prozess ergänzt wird, d.h. indem eine stochastische Indexierungsfunktion eingeführt wird (siehe auch die stochastische Indexierungsfunktion bei der Indexierung von Dokumenten in Abschnitt 3.4.2)). Durch mehrmaliges, unabhängiges Anwenden der Indexierungsfunktion auf Q_i^t wird jeweils ein anderer Queryvektor q_{ik}^t erzeugt, ohne dass ein ausgezeichneter Original-Queryvektor q_i^t existiert.

Die einfachste Konstruktion einer stochastischen Indexierungsfunktion ist die Verwendung einer deterministischen Indexierungsfunktion, zu der eine Zufallskomponente, d.h. ein n-dimensionaler Vektor $\sigma_k^t \in R^n$ von Zufallszahlen, addiert wird:

$$A_{IR(Q)}: Q(\Theta) \rightarrow DVR: Q_i^t \mid\rightarrow q_{ik}^t = q_i^t + \sigma_k^t; \sigma_k^t = (\sigma_{kj}^t \in R \mid j = 1, ..., n). \quad (297)$$

Der entscheidende Punkt bei einer stochastischen Indexierungsfunktion ist die Festlegung einer Wahrscheinlichkeitsverteilung, durch welche die einzelnen Komponenten σ_{kj}^t erzeugt werden. Hierbei muss die Topologie und die Metrik des Dokumentvektorenraums berücksichtigt werden. Weiterhin muss die Lokalität erhalten werden, d.h. die Verteilung muss in q_i^t ihr Maximum besitzen, und mit zunehmender Distanz zu q_i^t wird ihr Wert monoton kleiner, eventuell in Verbindung mit einem Distanzschwellenwert, bei dessen Überschreitung der Wahrscheinlichkeitswert auf Null gesetzt wird.

Im Rahmen dieser Arbeit werden deterministische Indexierungsfunktionen unterstellt, sodass auf eine weitere Darstellung der Konstruktion und Anwendung stochastischer Indexierungsfunktionen verzichtet werden soll.

3.6.2.2.4) Query-Polyrepräsentation und Queryvektor-Reproduktions-Operationen

Die Addition eines Zufallsvektors kann auch als eine Fom der Mutation interpretiert werden, sodass das Queryvektor-Polyrepräsentations-Verfahren als eine ES-Mutations-Operation interpretiert werden kann:

$$q_{ik}^{t} = m(q_i^{t} \mid \sigma_k^{t}). \tag{298}$$

Liegen mehrere Queryvektoren vor, so lässt sich daraus eine größere Menge von Queryvektoren erzeugen, indem Rekombinations-Operationen eventuell in Kombination mit Mutations-Operationen angewendet werden. Beispielsweise könnte eine Query-Polyrepräsentation betrachtet werden, bei der ein Agent mehrere alternative Formulierungen seines Informationsbedürfnisses abgegeben hat, wobei angenommen wird, dass eine geringe Anzahl von Queries vorliegt. Diese Anzahl genügt den Anforderungen einer stochastischen Polyrepräsentation nicht, wie sie durch ein Resampling-Verfahren wie dem Moving-Blocks-Bootstrap erfüllt sind. In einem solchen Fall kann eine Rekombinations-Operation auf der Ebene der Queries (siehe Abschnitt 3.6.1.2.6)) durchgeführt werden, wobei diskrete Rekombinationen und Crossover-Rekombinationen verwendet werden, die der Datenstruktur der Zeichensequenz angepasst sind, die aus dem Bereich der Genetischen Algorithmen und des Genetic Programmings stammen. Im weiteren wird jedoch der Fall betrachtet, dass die Queries durch eine gemeinsame Indexierungsfunktion $A_{IR(Q)}$ indexiert werden und die Rekombinations-Operation auf der Ebene der Queryvektoren angewendet wird. Auf der Ebene der Queryvektoren können die Rekombinations-Operationen der ES angewendet werden, wobei durch die diskrete und intermediäre Rekombination die Dimension der beteiligten Vektoren erhalten bleibt.

Sei $QEVM_i^{t}$ die Menge der Queryvektoren, die durch die Indexierung einer kleinen Querymenge erzeugt wird, und die als Elternmenge der Rekombinations-Operationen verwendet wird:

$$QEVM_i^{t} = \{q_{iE(j)}^{t} \mid j = 1, ..., \mu_E\}. \tag{299}$$

Aus dieser Elternmenge wird eine Zwischenpopulation von Nachkommen erzeugt, die zusammen mit den Eltern als Queryvektor-Polyrepräsentation verwendet werden sollen. Es sei darauf hingewiesen, dass in diesem Kontext noch kein Selektionskriterium formuliert ist, mit dem Elemente aus der Zwischenpopulation selektiert werden können, sodass es sich um kein vollständiges evolutionäres Verfahren handelt, obwohl die Komponenten Population und Reproduktion vorliegen.

Vorgestellt wird eine einfache zwei-geschlechtliche Rekombination, bei der zwei Elemente $q_{iE(1)}^{t}$ und $q_{iE(2)}^{t}$ aus $QEVM_i^{t}$ mit Zurücklegen gezogen werden, gefolgt von der Rekombinations-Operation, sodass ein Nachkommenvektor $q_{iN(k)}^{t}$ erzeugt wird, ohne dass hier das spezielle Rekombinationsverfahren weiter spezifiziert werden soll:

$$q_{iN(k)}^{t} = rec(q_{iE(1)}^{t}, q_{iE(2)}^{t}). \tag{300}$$

Wird eine $(\mu_E+\lambda)$-ES unterstellt, so werden λ Nachkommen erzeugt, die zusammen mit den Eltern die Queryvektorenmenge QVM_i^{t} bilden:

$$QVM_i^{t} = \{q_{iE(j)}^{t}, q_{iN(k)}^{t} \mid j = 1, ..., \mu_E; k = 1, ..., \lambda\}. \tag{301}$$

Diese Darstellung berücksichtigt keine Mutations-Operation als Bestandteil der Reproduktion, sowie keine Selbstadaption eines n-dimensionalen Mutationsvektors. Eine Selbstadaption, die aus einer Selbstmutation, einer Rekombination und einem Selektionsprozess besteht, kann in diesem Kontext jedoch nicht durchgeführt werden, da zum einen der Selektionsprozess fehlt, und zum anderen eine sinnvolle Selbstadaption eines Strategievektors nur über einen längeren Zeitraum, d.h. eine größere Anzahl von Iterationen, durchführbar ist. Bei einem einzelnen Interaktionsakt zwischen Agent und IRS in Form der Formulierung einer oder mehrerer alternativer Queries ist diese Bedingung jedoch nicht erfüllt, was sich im Kontext eines Relevanzfeedbacks jedoch ändern kann, wenn der Agent bereit ist eine längere Sequenz von Feedback-Prozessen durchzuführen.

Trotzdem kann eine Mutationsoperation im Rahmen der Erzeugung des Nachkommenvektors $q_{iN(k)}^{t}$ eingefügt werden, wobei zunächst die Diversivität der Eltern in $QEVM_i^t$ ermittelt wird, indem die Queryvektor-Varianz ermittelt wird:

$$var_{QEVM(i,t)} = 1/\mu_E * \sum_j (\bar{q}_{iE}^{t} - q_{iE(j)}^{t})^2, \text{ mit}$$
$$\bar{q}_{iE}^{t} = 1/\mu_E * \sum_j q_{iE(j)}^{t}. \quad (302)$$

Ein rekombinierter Nachkommenvektor kann mutiert werden, indem diese Varianz direkt verwendet wird, oder indem eine individuelle Varianz durch Selbstmutation erzeugt wird:

$$q_{iN(k)}^{t'} = m(q_{iN(k)}^{t} \mid var_{QEVM(i,t)}) \text{ oder}$$
$$q_{iN(k)}^{t'} = m(q_{iN(k)}^{t} \mid var_{QEVM(i,t)k}) \text{ mit } var_{QEVM(i,t)k} = m(var_{QEVM(i,t)} \mid var_{QEVM(i,t)}). \quad (303)$$

3.6.3) Retrievalstrategien bei einer Dokumentvektor-Monorepräsentation

Der zweite Schritt im Rahmen des Retrievals nach der Query-Indexierung besteht darin, mit Hilfe des Queryvektors q_i^t und der Dokument-Merkmals-Matrix DMM^t bzw. der gesamten Dokumentvektorenmenge DVM^t eine Teilmenge von Dokumentvektoren zu spezifizieren, deren zugehörige Dokumente dem Agenten als Ergebnis der Retrievaloperation betrachtet und dem Agenten präsentiert werden. Sei $DVM_i^t \equiv DVM(q_i^t) \subset DVM^t$ die Ergebnismenge der Dokumentvektoren, die durch ein Distanzmaß oder eine Metrik durch q_i^t im Dokumentvektorraum DVR ermittelt wird. Die Ergebnismenge wird durch eine Retrieval-Funktion ret(.) ermittelt, die allgemein als eine Abbildung der Potenzmenge $P^{DVM(t)}$ der Dokumentvektoren aus DVM^t auf sich selbst definiert wird:

$$ret(.): P^{DVM(t)} \rightarrow P^{DVM(t)}: DVM^t \mid\rightarrow DVM_i^t \equiv DVM(q_i^t). \quad (304)$$

Die Retrieval-Funktion ist in jedem Fall abhängig von der Dokumentvektorenmenge DVM^t, dem Queryvektor q_i^t und dem metrischen Dokumentvektorenraum $DVR \subseteq R^{n(t)}$ mit den Eigenschaften, die ihn definieren, wobei hier insbesondere die Metrik d_{DVR} betrachtet wird. Sei Γ_{DVR} die Menge aller Metriken, die in einem Dokumentvektorenraum DVM angewendet werden können:

$$\Gamma_{DVR} = \{d_{DVR}(.,.): DVM \times DVM \rightarrow DVM \mid$$
$$d_{DVR}(x_k, x_j) = d_{DVR}(x_j, x_k),$$

$$d_{DVR}(x_k, x_j) \geq 0,$$
$$d_{DVR}(x_k, x_j) \geq d(x_k, x_k) = 0,$$
$$d_{DVR}(x_k, x_j) \leq d_{DVR}(x_k, x_i) + d_{DVR}(x_i, x_j),$$
$$d_{DVR}(x_i, x_j) = 0 \Rightarrow d_{DVR}(x_i, x_k) = d_{DVR}(x_j, x_k)\}. \quad (305)$$

Die Retrieval-Funktion kann somit spezifiziert werden als eine Abbildung:

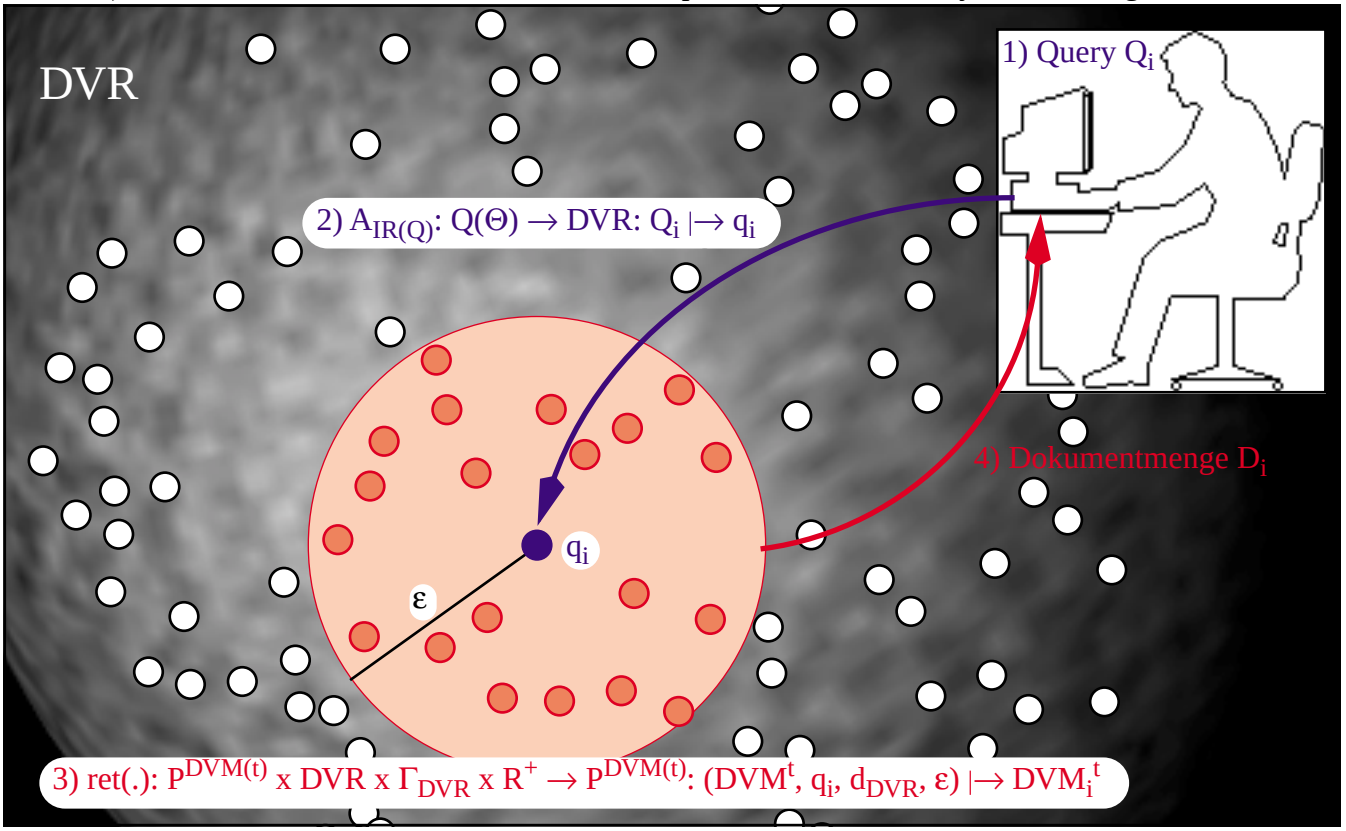
$$ret(DVM^t, q_i^t, d_{DVR}): P^{DVM(t)} \times DVR \times \Gamma_{DVR} \rightarrow P^{DVM(t)}: (DVM^t, q_i^t, d_{DVR}) \mid\rightarrow DVM_i^t. \quad (306)$$

Wird eine einfache Best-Match-Retrievalstrategie betrachtet, bei der alle Dokumentvektoren selektiert werden, deren Abstand von q_i^t kleiner-gleich einer Distanzschwelle $\varepsilon \in R^+$ ist, so ist die Retrieval-Funktion zusätzlich von ε abhängig. Es ergibt sich somit allgemein eine Abhängigkeit von DVM^t, q_i^t, d_{DVR}, und ε (siehe Abb. 37)):

$$ret(DVM^t, q_i^t, d_{DVR}, \varepsilon): P^{DVM(t)} \times DVR \times \Gamma_{DVR} \times R^+ \rightarrow P^{DVM(t)}: (DVM^t, q_i^t, d_{DVR}, \varepsilon) \mid\rightarrow DVM_i^t, \text{ mit}$$
$$DVM_i^t = \{x_{ij}^t \in DVM^t \mid d_{DVR}(q_i^t, x_{ij}^t) \leq \varepsilon\}. \quad (307)$$

Von der Retrieval-Funktion zu unterscheiden ist die Ranking-Funktion, die als eine Abbildung der Dokumentvektorenmenge DVM_i^t auf die geordnete Dokumentvektorenliste DV_i^t definiert wird. Liegt ein Retrieval ohne Relevanz-Feedback vor bzw. wird die Initialisierungs-Iteration eines Relevanz-Feedbacks betrachtet, so werden die Dokumentvektoren x_{ij}^t aus DVM_i^t nach steigenden Distanzen zu dem Queryvektor q_i^t geordnet.

Abb. 37) Retrieval-Funktion im Kontext der Monorepräsentation der Query-Indexierung



Zur Definition der Ranking-Funktion wird in Analogie zur Potenzmenge, als Menge aller Teilmengen, die Potenzliste, als Menge aller geordneter Teillisten, benötigt. Bezogen auf die Dokumentvektorenmenge DVM^t sei die Potenzmenge mit $P^{DVM(t)}$ und die Potenzliste mit $L^{DVM(t)}$ bezeichnet, während die Potenzmenge bezogen auf die Ergebnismenge DVM_i^t mit $P^{DVM(i,t)}$ und die Potenzliste mit $P^{DVM(i,t)}$ bezeichnet wird. Die Ranking-Funktion kann definiert werden als:

$$\begin{aligned} &rank(DVM_i^t, d_{DVR}): P^{DVM(i,t)} \times \Gamma_{DVR} \rightarrow L^{DVM(i,t)}: (DVM_i^t, d_{DVR}) \mid\rightarrow DV_i^t, \text{ mit} \\ &DV_i^t = (x_{ij}^t \in DVM_i^t \mid d_{DVR}(q_i^t, x_{ij}^t) < d_{DVR}(q_i^t, x_{ij+1}^t)). \end{aligned} \tag{308}$$

Der gesamte Retrievalprozess kann somit als Verkettung einer Retrieval- und einer Ranking-Funktion beschrieben werden:

$$\begin{aligned} &rank(DVM_i^t, d_{DVR}) \circ ret(DVM^t, q_i^t, d_{DVR}): \\ &P^{DVM(t)} \times DVR \times \Gamma_{DVR} \rightarrow L^{DVM(t)}: (DVM^t, q_i^t, d_{DVR}) \mid\rightarrow DV_i^t. \end{aligned} \tag{309}$$

Da der Input (DVM^t, q_i^t, d_{DVR}) in die Verkettung gleich dem Input in die Retrieval-Funktion ist, kann eine erweiterte Retrieval-Funktion als Verkettung der einfachen Retrieval-Funktion und der Ranking-Funktion definiert werden.

Die einfache oder erweiterte Retrieval-Funktion kann nicht als eine Form der Umkehrfunktion der Indexierungsfunktion verstanden werden, da mehrere gegebene Dokumentvektoren ermittelt werden sollen. Eine Abbildung $DVR \rightarrow D(\Theta)$ ist demgegenüber unterspezifiziert, da zu einem Punkt im Dokumentvektorenraum viele Dokumente zuordenbar sind, selbst wenn das gleiche Gewichtungsmodell verwendet wird. Dies ergibt sich daraus, dass Häufigkeiten von Merkmalen als Grundlage der Indexierung verwendet werden, sodass unterschiedliche Zeichenketten die exakt gleichen diskreten Häufigkeitverteilungen besitzen können, denen genau der gleiche Punkt im DVR zugeordnet wird. Die Konstruktion von künstlichen Dokumenten $D(x_i)_j$ zu einem Punkt x_i in DVR kann dem gegenüber als eine Optimierungsaufgabe formuliert werden, die bei einer gegebenen Indexierungsfunktion $A_{IR(D)}$ iterativ gelöst werden kann, indem (künstliche) Dokumente gesucht werden, die durch $A_{IR(D)}$ auf Punkte x_{ij} in DVR abgebildet werden, die in einer kleinen ε-Umgebung um den gesuchten Vektor x_i liegen, wobei die Distanz ε im Verlauf der Optimierung gesenkt werden kann:

$$\begin{aligned} &\text{Suche } D(x_i)_j: A_{IR(D)}: D(\Theta) \rightarrow DVR: D(x_i)_j \mid\rightarrow x_{ij}, \\ &\text{mit } d_{DVR}(x_i, x_{ij}) < \varepsilon \text{ bzw. } d_{DVR}(x_i, x_{ij}) \rightarrow 0. \end{aligned} \tag{310}$$

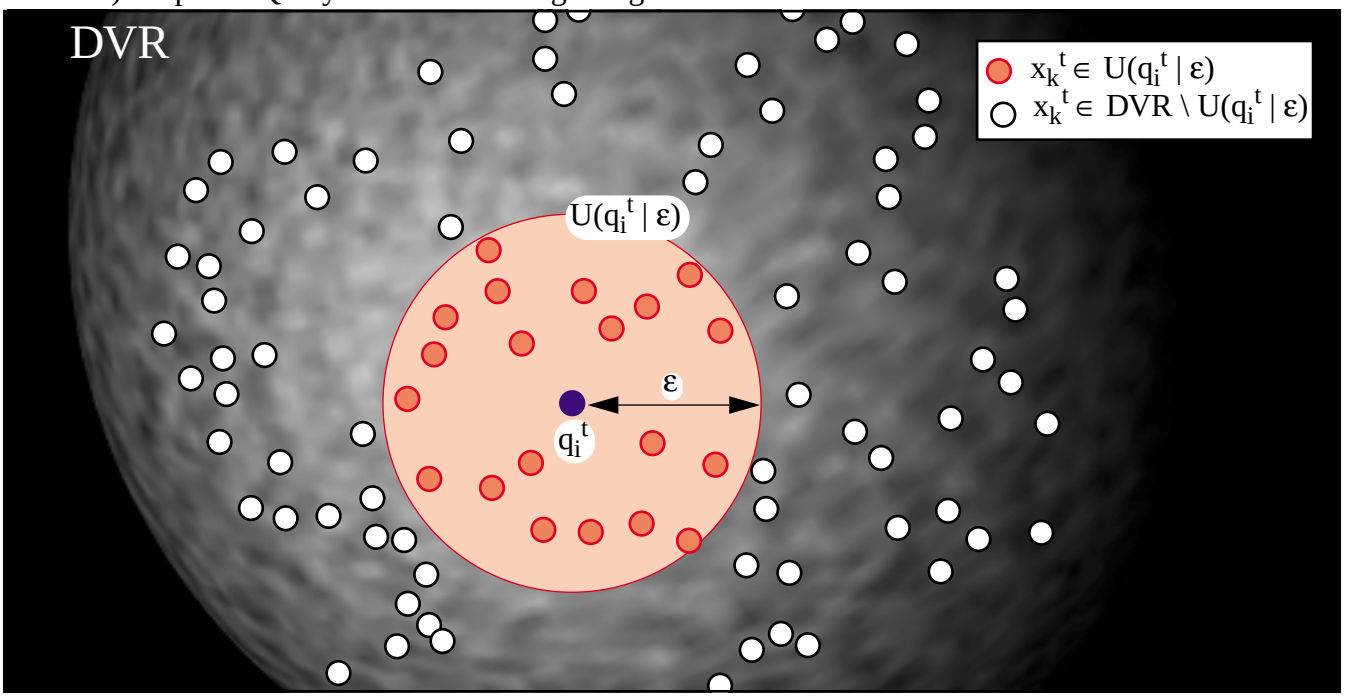
Soll eine Suche nach einer Dokumentmenge mit dieser Randbedingung durchgeführt werden, was dem Polyrepräsentationsprinzip entspricht, so ist ein populationsbasiertes Verfahren, d.h. ein Evolutions-Algorithmus (siehe Abschnitt 2.5)) am besten geeignet, ohne dass hier weiter darauf eingegangen werden soll.

3.6.3.1) Dokumentvektor-Monorepräsentation und Queryvektor-Monorepräsentation

Eine Klasse von Retrieval-Strategien, die im Rahmen dieser Arbeit mehrfach betrachtet wird, basiert auf der Retrieval-Mannigfaltigkeit, als einer zusammenhängenden Region im Dokumentvektorraum DVR um den Queryvektor q_i^t, die unterschiedliche topologische Ausprägungen besitzen kann. Eine einfache Möglichkeit der Definition einer Retrieval-Mannigfaltigkeit ist die Festlegung einer ε-Umgebung um den Queryvektor q_i^t mit Hilfe der Metrik $d_{DVR}(.,.)$ als offene Menge aller Punkte aus DVR, die einen Abstand kleiner ε von q_i^t besitzen (siehe Abb. 38)):

$$U(q_i^t \mid \varepsilon) = \{x \in DVR \mid d_{DVR}(q_i^t, x) < \varepsilon, \varepsilon \in R^+\}. \quad (311)$$

Abb. 38) Einpunkt-Queryvektor mit ε-Umgebung im DVR



Der Parameter ε kann im Rahmen der Modellierung von Unsicherheiten interpretiert werden. Die Query q_i^t wird bei größer werdendem Parameter unspezifischer, d.h. immer mehr Dokumentvektoren werden in die entsprechende Umgebung einbezogen. Der Parameterwert ε ist in jeder der n^t Dimensionen von DVR gleich, d.h. ein großer ε-Wert bezieht für ein Merkmal F_j Dokumentvektoren mit ein, die ausgehend von q_i^t als Mittelpunkt, gössere wie kleinere F_j-, d.h. x_j-Werte besitzt. Somit wird der Umgebungsparameter auch zu dem Radius eines Konfidenzintervalls für jedes der n^t Merkmale mit $q_{ij}^t \in R$ als Mittelpunkt. Würden diese Konfidenzintervalle unabhängig für alle Dimensionen existieren, so würde dies einen n^t-dimensionalen Würfel als Retrieval-Mannigfaltigkeit ergeben.

Die Ergebnisdokumentvektorenliste $DV(q_i^t)$ ergibt sich aus den Dokumentvektoren x_k^t, die innerhalb der ε-Umgebung des Queryvektors liegen, und die entsprechend ihrer steigenden Distanz zu q_i^t geordnet werden:

$$DV(q_i^t) = (x_k^t \in U(q_i^t \mid \varepsilon) \mid d_{DVR}(q_i^t, x_k^t) < d_{DVR}(q_i^t, x_{k+1}^t) < \varepsilon). \quad (312)$$

Der Nachteil dieser Vorgehensweise besteht darin, dass für einen gegebenen Schwellenwert ε die ε-Umgebung leer sein kann, d.h. es können keine Dokumentvektoren ermittelt werden, sodass $DV(q_i^t)$

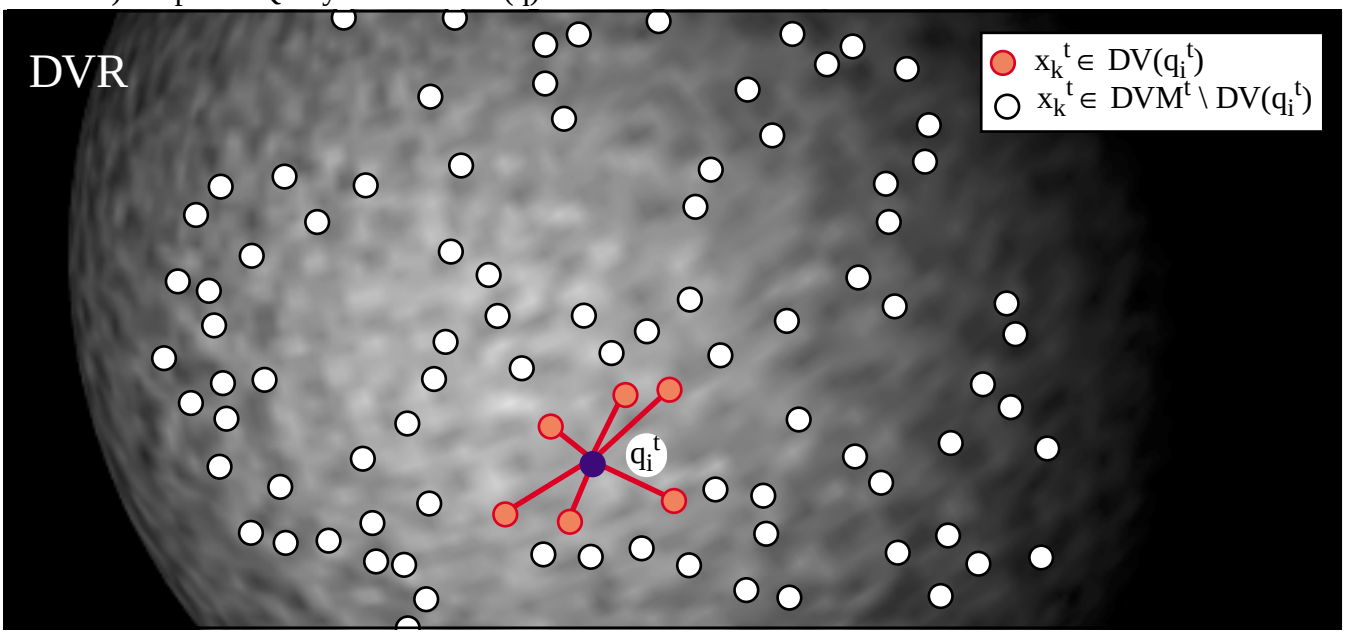
ebenfalls leer ist. Um sicher zu stellen, dass für alle Queryvektoren eine nicht-leere Dokumentvektorenmenge durch die Retrievalfunktion ermittelt werden, muss eine andere Retrievalstrategie angewendet werden, bei der nicht eine Mannigfaltigkeit wie eine ε-Umgebung gegeben ist, sondern bei der eine feste Anzahl von $m(q) << m^t$ Dokumentvektoren ausgegeben wird, welche die geringsten Abstände von q_i^t besitzen (siehe Abb. 39)).

$$DV(q_i^t) = (x_k^t \in DVM^t \mid d_{DVR}(q_i^t, x_k^t) < d_{DVR}(q_i^t, x_{k+1}^t); k = 1, ..., m(q)), \text{ mit}$$
$$d_{DVR}(q_i^t, x_1^t) = \min\{d_{DVR}(q_i^t, x_i^t) \mid \forall x_i^t \in DVM^t\},$$
$$d_{DVR}(q_i^t, x_2^t) = \min\{d_{DVR}(q_i^t, x_i^t) \mid \forall x_i^t \in DVM^t \setminus \{x_1^t\}\},$$
$$...,$$
$$d_{DVR}(q_i^t, x_{m(q)}^t) = \min\{d_{DVR}(q_i^t, x_i^t) \mid \forall x_i^t \in DVM^t \setminus \{x_1^t, ..., x_{m(q)-1}^t\}\}. \quad (313)$$

Der Nachteil dieser Vorgehensweise besteht darin, dass für eine gegebene Anzahl m(q) Dokumentvektoren ermittelt werden, unabhängig wie groß die Distanz zwischen ihnen und dem Queryvektor ist. D.h. liegt q_i^t weit von den nächsten Dokumentvektoren entfernt, so werden diese trotzdem ermittelt.

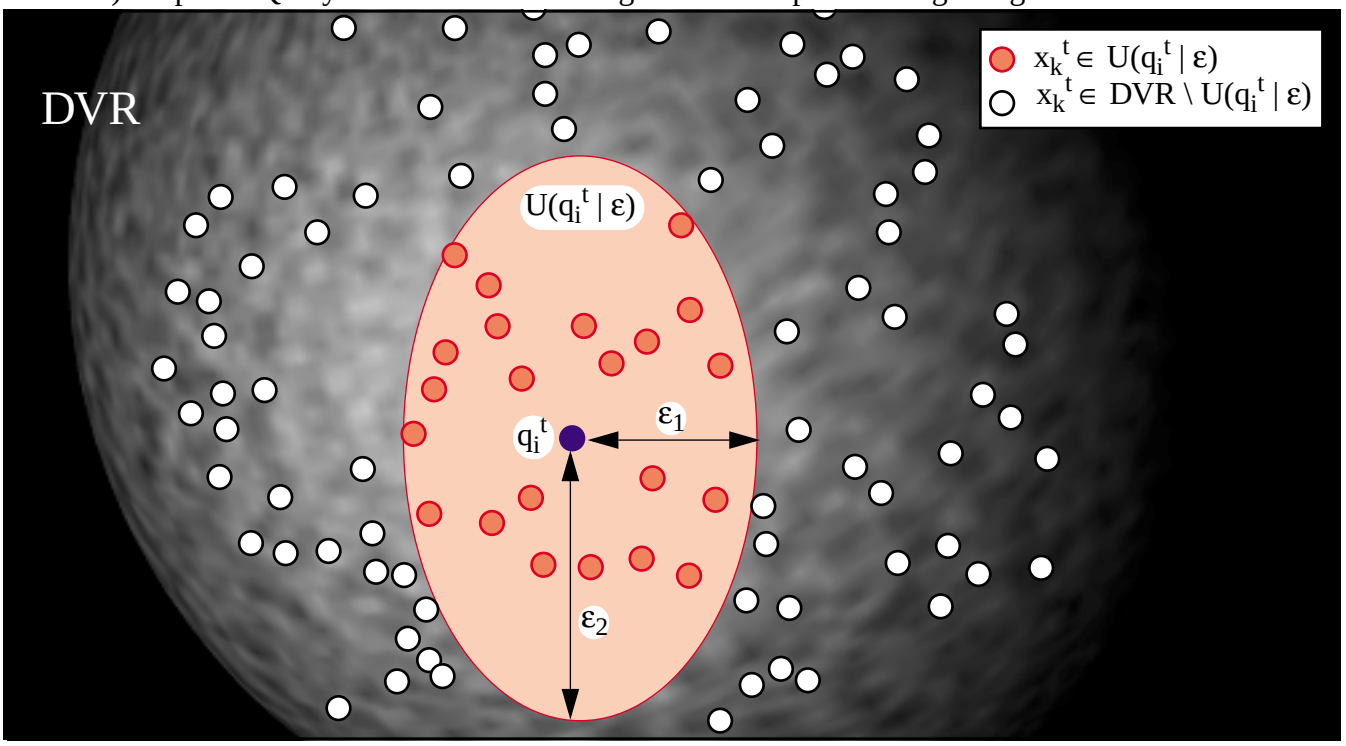
Gemischte Strategien sind sinnvoll, wenn innerhalb einer ε-Umgebung mehr Dokumentvektoren liegen, als einem Agenten präsentiert werden sollen. $DV(q_i^t)$ wird zunächst durch die ε-Umgebung bestimmt, und dann wird geprüft, ob die Anzahl der darin enthaltenen Dokumentvektoren größer ist als ein Schwellenwert wie m(q). Ist dies der Fall, so werden die ersten m(q) Dokumentvektoren übernommen, und die restlichen verworfen bzw. zurückgehalten. Sollten in der ε-Umgebung weniger Dokumentvektoren liegen, so könnte eine gemischte Strategie weitere Dokumentvektoren außerhalb $U(q_i^t \mid \varepsilon)$ entsprechend dem kleinsten Abstand zu q_i^t suchen, bis die Sollanzahl von m(q) erreicht wird.

Abb. 39) Einpunkt-Queryvektor mit m(q) = 6 Retrieval-Dokumentvektoren



Neben der oben dargestellten Retrieval-Mannigfaltigkeit können beispielsweise ellipsoide Umgebungen definiert werden, deren Achsen parallel zu den Koordinatenachesen des Dokumentvektorenraums DVR liegen (siehe Abb. 40)).

Abb. 40) Einpunkt-Queryvektor mit Retrievalregion als ε-ellipsoide Umgebung im DVR



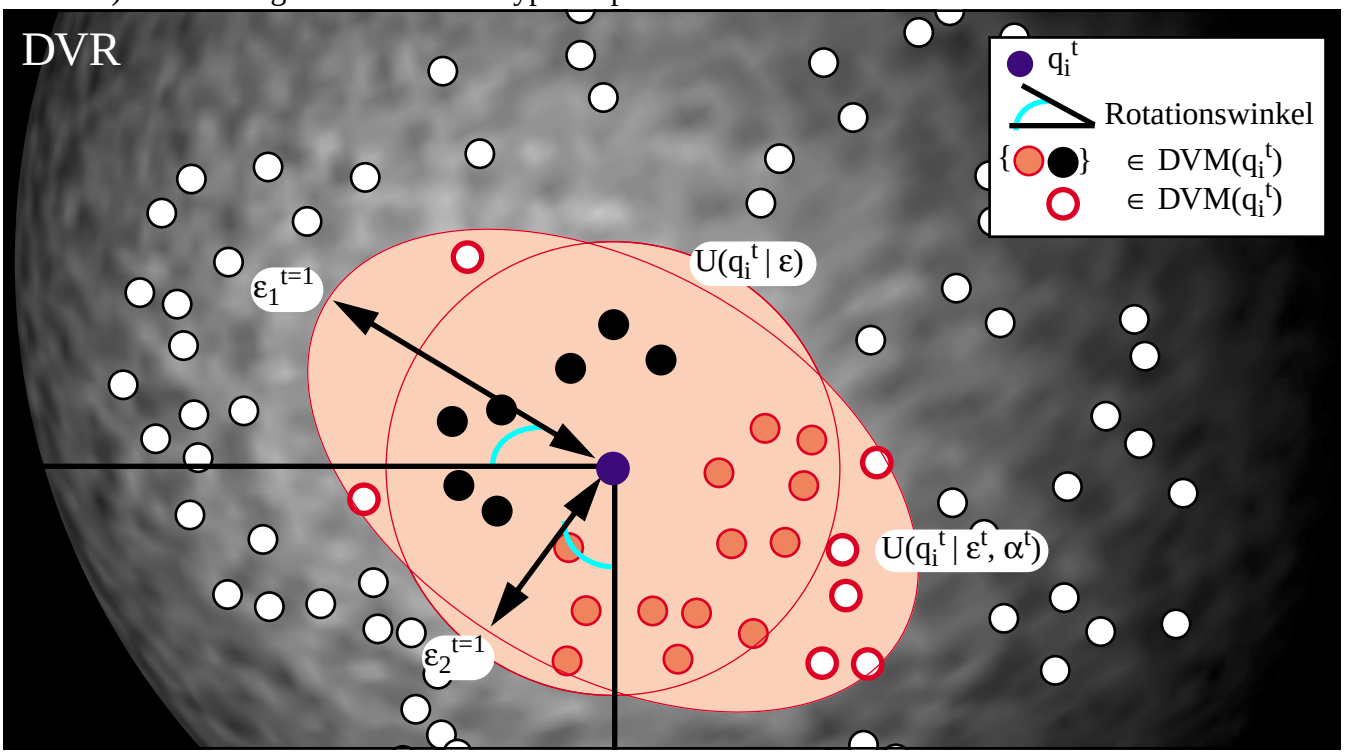
Diese Form besitzt eine Analogie zum Mutationsschrittweitenvektor σ_i^t bei den Evolutions-Strategien (Bachelier (1998b: 49ff[16]), Rechenberg (1994[277]), Jacob (1997[174]), Schwefel (1977[305], 1981[306], 1995[307])), wobei der Nachteil gegenüber einer zentrischen Umgebungsdefinition darin besteht, dass im n-dimensionalen DVR n reelle Zahlen notwendig sind, um eine ellipsoide Umgebung zu beschreiben, und dass die Entscheidung, ob ein Dokumentvektor Element einer solchen Umgebung $U(q_i^t \mid \varepsilon \in R^{+n})$ ist, deutlich aufwendiger wird.

Die Wirkung einer Umgebung $U(q_i^t \mid \varepsilon \in R^{+n})$ besteht darin, dass den unterschiedlichen Merkmalen im DVR unterschiedliche Gewichtungen zugeordnet werden können. Ein großer Parameterwert ε_j bezüglich einem Merkmal F_j führt dazu, dass die Ellipse in dieser Dimension weit gestreckt ist, mit der Folge, dass mehr Dokumentvektoren Element dieser Umgebung werden. Durch die Symmetrie um q_i^t bedeutet dies jedoch auch, dass Dokumentvektoren mit einem kleineren wie größeren F_j-Wert einbezogen werden, sodass ε_j auch als eine Modellierung von Unsicherheit bezüglich des entsprechenden Merkmals betrachtet werden kann. Dies steht ebenfalls in Analogie zu dem Mutationsschrittweitenvektor σ_i^t bei den ES, da dort die Einzelwerte σ_{ij}^t des Vektors als Standardabweichungen in einer Mutationsverteilung wie der Gauss-Verteilung verwendet werden, die als statistisches Maß der Unsicherheit verwendet wird.

Wird die Analogie zu den ES weiter verwendet, so kann die Retrievalregion weiter modifiziert werden, indem zusätzlich der [n(n-1)/2]-dimensionale Strategieparametervektor α der korrelierten Mutationen (Bachelier (1998b: 77ff[16]), Nissen (1994: 155[237])) eingeführt wird, der als Vektor bzw. Matrix von Rotationswinkeln interpretiert wird. Wird, wie oben beschrieben, ein Hyperellipsoid durch einen n-dimensionalen Parametervektor $\varepsilon^t \in R^{+n}$ beschrieben, so verlaufen alle n Achsen des Ellipsoides parallel zu den Achsen des Koordinatensystems im DVR. Ein solcher Hyperellipsoid kann rotiert werden, sodass seine Achsen einen Winkel zu den Koordinatenachsen bilden (siehe Abb. 41)).

Da jede Achse unabhängig rotiert werden kann, ergibt dies n(n-1)/2 Freiheitsgrade, die zusätzlich zu den n Freiheitsgraden der Achsenlängen in ε^t hinzukommen, sodass die Retrievalregion beschrieben wird durch $U(q_i^t \mid \varepsilon, \alpha)$, mit $\varepsilon \in R^{+n}$ und $\alpha \in R^{+[n(n-1)/2]}$.

Abb. 41) Retrievalregion als rotierter Hyperellipsoid



Prinzipiell stellt sich bei einer ellipsoiden Umgebung die Frage, wie die Einzelwerte ermittelt werden sollen, wobei von globalen oder lokalen Umgebungsparametern ausgegangen werden kann, d.h. der ε-Vektor gilt für alle möglichen Queryvektoren oder nur für eine bestimmte Query eines Agenten. Legt man die Interpretation der ε_j-Werte als Unsicherheit zugrunde, so sind lokale Umgebungsparameter sinnvoll, die im Rahmen der Formulierung der Query durch den Agenten erfragt werden müssen, was jedoch im Hinblick auf die sehr hohen mentalen Kosten des Agenten und die prinzipiellen Beschränkungen bezüglich der bewussten Übersetzbarkeit von mentalen Präferenzen in sprachliche Konstrukte, als kaum durchführbar erscheint. Im Rahmen von Feedback-Operationen könnten sich unterschiedliche Gewichtungen von Merkmalen jedoch berechnen lassen, sodass dort auf diese Problematik eingegangen werden soll (siehe Abschnitt 3.9)).

Alle Retrievalmannigfaltigkeiten, die bislang dargestellt wurden, sind deterministisch definiert, d.h. durch einen Umgebungsparameter oder -parametervektor bezogen auf einen Mittelpunkt, ist eine Umgebung eindeutig definiert. Obwohl die deterministische Vorgehensweise im weiteren verwendet wird, sei darauf hingewiesen, dass mit der Angabe eines Umgebungsparameters auch eine stochastische Interpretation einer Umgebung festgelegt werden kann, wobei vereinfachend von genau einem Parameter $\varepsilon \in R^+$ ausgegangen werden soll. Eine stochastische Interpretation ergibt sich in Verbindung mit einer Verteilungsfunktion, wobei vereinfachend eine Gaussfunktion betrachtet werden soll, wobei der Parameter ε als Standardabweichung verwendet wird.

Eine stochastische Retrievalmannigfaltigkeit wird erst durch eine Menge von Zufallsexperimenten zu dem Zeitpunkt erzeugt, in dem sie benötigt wird. Soll eine Umgebung $U(q_i^t \mid \varepsilon)$ spezifiziert werden, so wird in einem n-dimensionalen Raum wie in DVR n mal ein Zufallsexperiment mit einer $N(0, \varepsilon)$-Verteilung durchgeführt, sodass sich n reelle Zufallszahlen ergeben, von denen der Betrag gebildet wird. Diese n Zufallszahlen bilden den Umgebungsparametervektor $\varepsilon^t \in R^{+n}$, der eine deterministische Umgebung $U(q_i^t \mid \varepsilon^t)$ in Form eines Hyperellipsoides um q_i^t bildet. Wird der gleiche Prozess bei einem gleichen Ursprungsparameter ε nochmals durchgeführt, so ergibt sich durch die n Zufallsprozesse ein anderer Parametervektor und somit eine andere Umgebung. Auf die gleiche Weise lässt sich ein stochastischer rotierter Hyperellipsoid erzeugen, indem n Zufallsexperimente für die Achsenlängen und n(n-1)/2 Zufallsexperimente für die Rotationswinkel zwischen den Achsen in dem n-dimensionalen DVR durchgeführt werden.

3.6.3.2) Dokumentvektor-Monorepräsentation und Queryvektor-Polyrepräsentation

Wurden durch den Agenten simultan mehrere Formulierungen seines Informationsbedürfnisses gegeben, oder wurde eine Query durch eine Polyrepräsentation auf mehrere Queryvektoren q_{ik}^t abgebildet, so müssen die Retrievalstrategien, die genau einen Queryvektor als Input verarbeiten, modifiziert werden. Die Menge der verwendeten Queryvektoren sei mit QVM_i^t bezeichnet:

$$QVM_i^t = \{q_{ik}^t \in DVR \mid j = 1, ..., \delta\}. \tag{314}$$

Die einfachste Strategie besteht darin, die einzelnen Queryvektoren zunächst als unabhängige Anfragen zu behandeln, für die jeweils eine eigene Retrieval-Dokumentvektoren-Menge $DVM(q_{ik}^t)$ z.B. durch eine ε-Umgebung gebildet wird, oder eine Retrieval-Dokumentvektoren-Liste $DV(q_{ik}^t)$ erstellt wird (siehe Abb. 42) und Abb. 43)).

Die Retrieval-Dokumentvektoren-Mengen $DVM(q_{ik}^t)$ müssen in einem zweiten Schritt kombiniert oder integriert werden (Combining), um eine einheitliche Retrievalmenge zu generieren, wobei unterschiedlichste Strategien angewendet werden können, die in zwei Klassen unterteilt werden:

1) Strategien, bei denen alle Dokumentvektoren der Einzelmengen in der Gesamtmenge enthalten sind.
2) Strategien, bei denen nur eine Teilmenge der Dokumentvektoren in den $DVM(q_{ik}^t)$ in einer Gesamtmenge übernommen werden.

Werden alle Dokumentvektoren übernommen, so können diese mit unterschiedlichen Gewichten versehen werden, was den Übergang zu einer Retrieval-Dokumentvektoren-Gesamtliste bzw. einem Ranking bedeutet. Eine Gewichtung ist auch die Grundlage für Ausschlussverfahren aus der zweiten Strategieklasse, sodass die Bestimmung von Gewichtungen auch hier eine entscheidende Rolle spielt. Eine einfache Grundlage für Gewichtungen ist die Häufigkeit der Zugehörigkeit eines Dokumentvektors zu den verschiedenen Umgebungen $U(q_{ik}^t \mid \varepsilon)$, wobei eine monoton steigende Funktion der Gewichtung in Abhängigkeit von der Häufigkeit unterstellt wird (siehe Abb. 44)).

Abb. 42) Retrieval bei einer Queryvektor-Polyrepräsentation

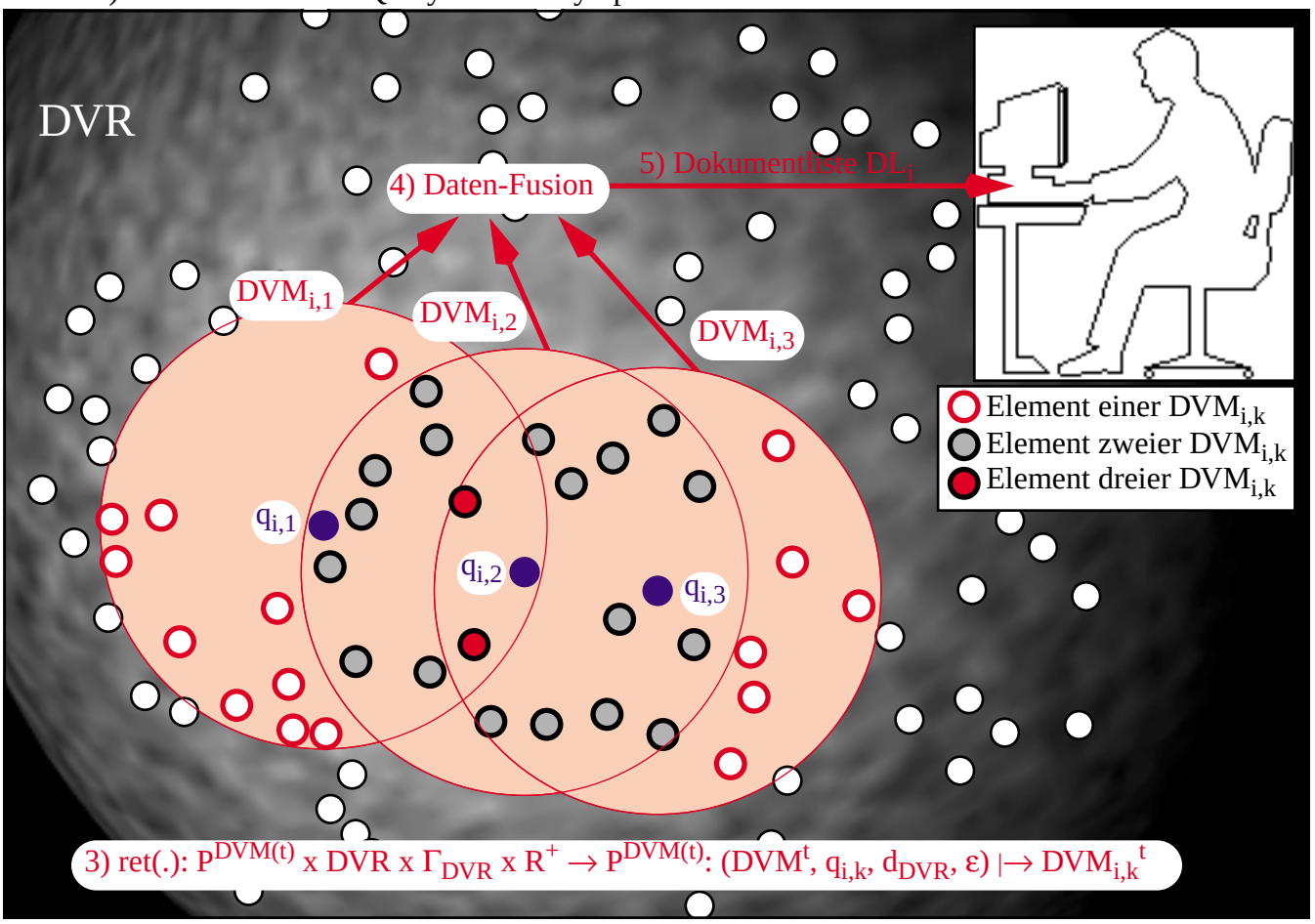
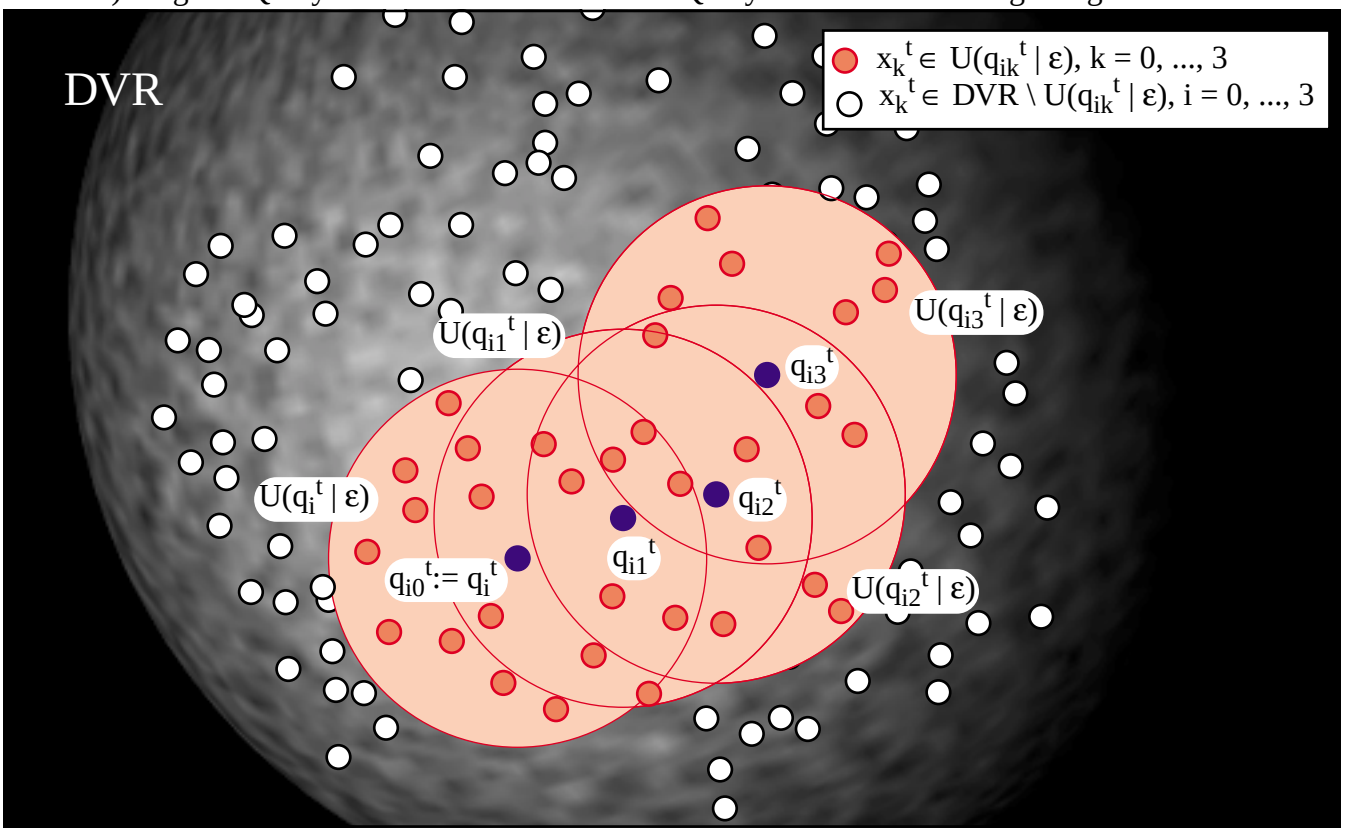


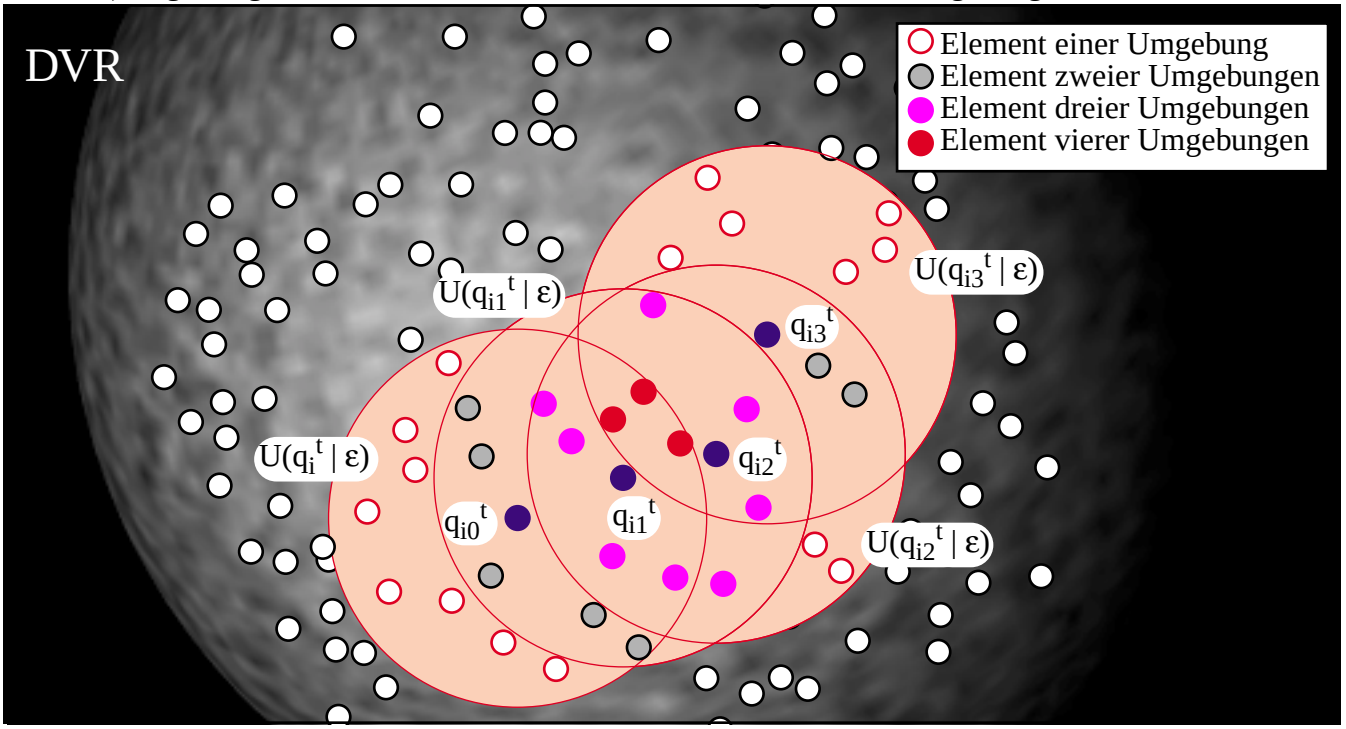
Abb. 43) Original-Queryvektor mit drei künstlichen Queryvektoren und ε-Umgebungen



Sei $DVM(QVM_i^t)$ die Gesamtmenge aller Dokumentvektoren aus den Einzelmengen $DVM(q_{ik}^t)$:

$$DVM(QVM_i^t) = \cup_{k=0->\delta} DVM(q_{ik}^t) = \{x_j^t \mid j = 1, ..., \}. \quad (315)$$

Abb. 44) Zugehörigkeit eines Dokumentvektors zu unterschiedlichen Umgebungen



Sei $h(x_j^t \mid U(q_i^t \mid \varepsilon)_{ges}) \in \{1, ..., \delta + 1\}$ die absolute Häufigkeit des Auftretens des Dokumentvektors $x_j^t \in DVM(q_i^t)_{ges}$ in den Umgebungen $U(q_{i0}^t \mid \varepsilon), ..., U(q_{i\delta}^t \mid \varepsilon)$:

$$h(x_j^t \mid U(q_i^t \mid \varepsilon)_{ges}) = \sum_{k=0->\delta} h(x_j^t \mid U(q_{ik}^t \mid \varepsilon)), \text{ mit} \quad (316)$$

$$h(x_j^t \mid U(q_{ik}^t \mid \varepsilon)) = \begin{cases} 1, & \text{wenn } x_j^t \in DVM(q_{ik}^t), \\ 0, & \text{wenn } x_j^t \notin DVM(q_{ik}^t). \end{cases} \quad (317)$$

Jedem der Dokumentvektoren $x_j^t \in DVM(QVM_i^t)$ wird seine Häufigkeit $h(x_j^t \mid U(q_i^t \mid \varepsilon)_{ges})$ zugeordnet, gefolgt von der Klassifizierung der Dokumentvektoren nach fallender Häufigkeit. In der Menge $DVM(q_i^t \mid 1)_{ges}$ werden die Dokumentvektoren gesammelt, die einen Häufigkeitswert von $\delta + 1$ besitzen, bis in $DVM(q_i^t \mid \delta + 1)_{ges}$ die Dokumentvektoren gesammelt werden, die in genau einer Umgebung liegen, und die somit einen $h(x_j^t \mid U(q_i^t \mid \varepsilon)_{ges})$-Wert von 1 besitzen. Entsprechend der verwendeten Definitionen kann jede der Dokumentvektorenmengen bis auf $DVM(q_i^t \mid \delta + 1)_{ges}$ leer sein, da jeder Dokumentvektor in mindestens einer Umgebung liegen muss, aber kein Dokumentvektor muss in mehr als einer Umgebung liegen. Die zugehörigen Dokumentmengen $DM(q_i^t \mid 1)_{ges}$ bis $DM(q_i^t \mid \delta + 1)_{ges}$ mit Referenzen zu den jeweiligen Dokumenten werden dem Agenten als Retrievalergebnis der polyrepräsentierten Query in dieser Rangfolge angeboten.

Eine Dokumentrangliste der Gesamt-Dokumentvektorenmenge kann gebildet werden, wenn die Mengen $DVM(q_{ik}^t)$ zunächst in Listen $DV(q_{ik}^t)$ umgewandelt werden, und wenn die Ränge dieser Listen über alle

k = 0, ..., δ zusammengefasst werden. D.h. die Dokumentvektoren in der Menge

$$DVM(q_{ik}^t) = \{x_{kj}^t \mid x_{kj}^t \in U(q_{ik}^t \mid \varepsilon)\}, \tag{318}$$

in der jeweils unterschiedlich viele Elemente enthalten sein können, werden nach steigender Distanz zu dem Queryvektor q_{ik}^t sortiert, sodass die geordnete Liste entsteht:

$$DV(q_{ik}^t) = (x_{kj}^t \mid x_{kj}^t \in DVM(q_{ik}^t); d_{DVR}(q_{ik}^t, x_{kj}^t) < d_{DVR}(q_{ik}^t, x_{kj+1}^t)). \tag{319}$$

Die Ränge j über alle Listen $DV(q_{ik}^t)$ werden dann zusammengefasst, wobei Ränge, die in einzelnen Listen unbesetzt sind, berücksichtigt werden:

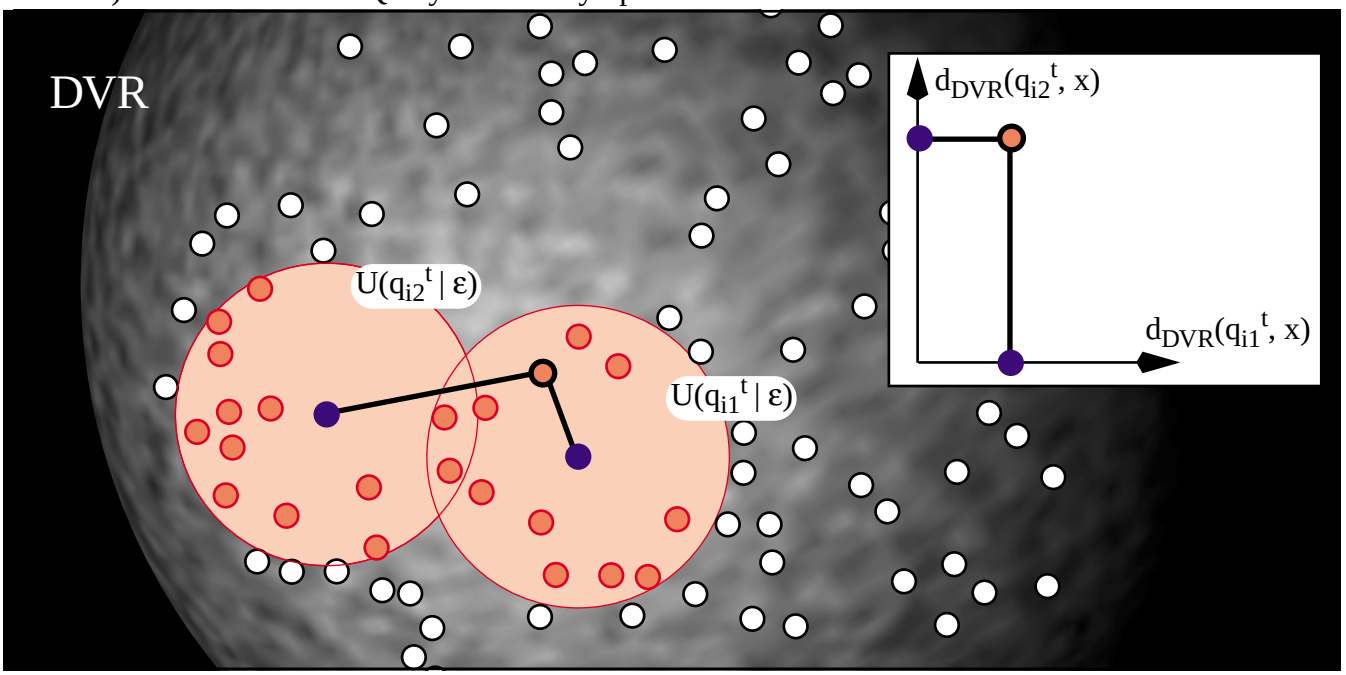
$$DVM(q_i^t \mid j)_{ges} = \{x_{kj}^t \mid x_{kj}^t \in DV(q_{ik}^t), k = 0, ..., \delta\}. \tag{320}$$

Sei j(max) der größte auftretende Rang, der gleich der Anzahl der Elemente in der größten $DVM(q_{ik}^t)$-Menge ist. Die zugehörigen Dokumentmengen $DM(q_i^t \mid 1)_{ges}$ bis $DM(q_i^t \mid j(max))_{ges}$ mit Referenzen zu den jeweiligen Dokumenten werden dem Agenten als Retrievalergebnis der polyrepräsentierten Query in dieser Rangfolge angeboten.

Anstatt ein Ranking auf der Basis von Zugehörigkeiten zu ε-Umgebungen aufzubauen, kann dies auch durch Distanzen von Dokumentvektoren zu den Queryvektoren durchgeführt werden, so wie bei einer Queryvektor-Monorepräsentation ein Ranking auf der Basis der Distanzen zwischen q_i^t und den Dokumentvektoren aus der Ergebnismenge $DV(q_i^t)$ gebildet wird, wobei kleine Distanzen einen kleinen und somit besseren Rang bedeuten. Jedem Dokumentvektor x_j^t aus der Gesamtmenge $DVM(QVM_i^t)$ wird ein δ-komponentiger Vektor d_j^t zugeordnet, der die Distanzen zu den δ Queryvektoren q_{ik}^t aus QVM_i^t enthält (siehe Abb. 45)):

$$d_j^t = (d_{DVR}(x_j^t, q_{ik}^t) \mid \forall q_{ik}^t \in QVM_i^t, k = 1, .., \delta). \tag{321}$$

Abb. 45) Distanzvektor bei Queryvektor-Polyrepräsentation



Mit diesen Vektoren lässt sich ein Ranking durch ein Ein-Ziel- bzw. durch ein δ-Ziel-Entscheidungsverfahren formulieren. Im ersten Fall wird der Mittelwert $\bar{d}_j^t \in R^+$ der Distanzen gebildet, und die Dokumentvektoren werden nach steigendem Distanzmittelwert geordnet, sodass die geordnete Liste $DV(QVM_i^t)$ entsteht:

$$DV(QVM_i^t) = (x_j^t \in DVM(QVM_i^t) \mid \bar{d}_j^t < \bar{d}_{j+1}^t),$$
$$\bar{d}_j^t = 1/\delta * \sum_k d_{DVR}(x_j^t, q_{ik}^t), \forall q_{ik}^t \in QVM_i^t. \quad (322)$$

Im zweiten Fall wird das Pareto-Kriterium in Verbindung mit einem hierarchie-erzeugenden Verfahren wie z.B. der sukzessiven Deaktivierung von Paretomengen verwendet (siehe auch Abschnitt 2.4); Bachelier (1999d:32ff[22])), wobei alle Komponenten der Distanzvektoren minimiert werden sollen. Bei der sukzessiven Deaktivierung von Paretomengen wird aus der Gesamtmenge $DVM(QVM_i^t)$ die Pareto-Menge $PM(DVM(QVM_i^t))$ gebildet, die aus der Grundmenge entfernt wird, sodass sich die erste Restmenge $\Delta DVM(QVM_i^t)_1$ ergibt. Aus dieser wird wiederum die Paretomenge $PM(\Delta DVM(QVM_i^t)_1)$ gebildet, die entfernt wird, solange bis keine Elemente in der Restmenge vorliegen. Die einzelnen Paretomengen bilden die Ränge einer Dokumentvektor-Hierarchie:

$$DVH(DVM(QVM_i^t)) = (PM(DVM(QVM_i^t)), PM(\Delta DVM(QVM_i^t)_p) \mid p = 1, ...,). \quad (323)$$

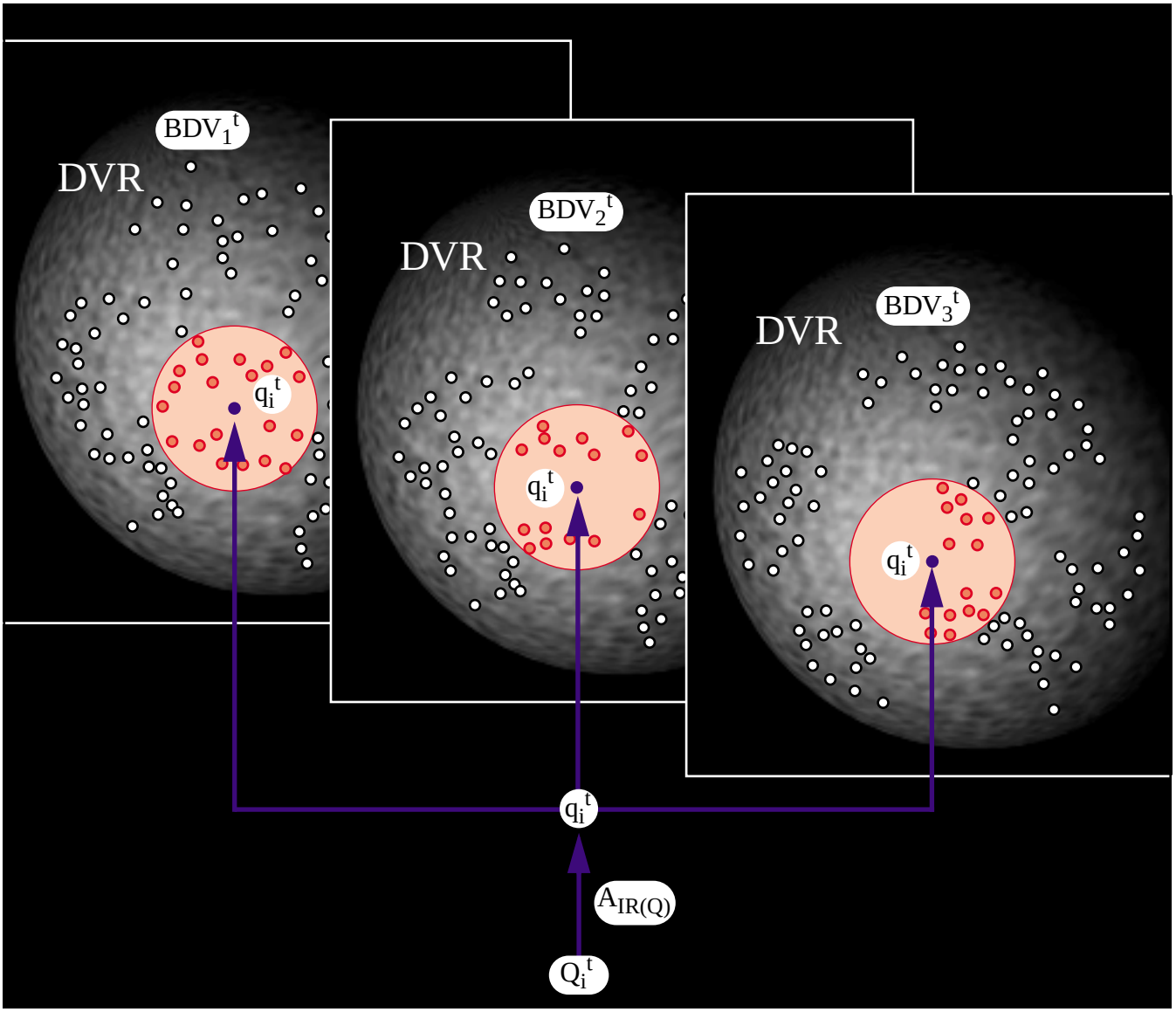
3.6.4) Retrievalstrategien bei einer Dokumentvektor-Polyrepräsentation

In den vorangegangenen Darstellungen wurde von einer Monorepräsentation der Dokumentvektoren, bei einer Mono- bzw. Polyrepräsentation des Queryvektors ausgegangen. Eine Polyrepräsentation der Dokumentvektoren kann daraus abgeleitet werden, indem für jede der Dokumentvektorenlisten BDV_k^t unabhängig eine der beschriebenen Retrievalstrategien angewendet wird, wobei zwischen einer Mono- und einer Polyrepräsentation der Query unterschieden werden kann.

Wie in der Einleitung festgelegt wurde, soll darauf verzichtet werden, eine mehrfache Polyrepräsentation von Komponenten aus dem Tupel (D^t, DVM^t, F^t, $DVM(q_i^t)$, ..., $A_{IR(D)}$, $A_{IR(Q)}$, ret(.)) zu beschreiben, da dies zu einer kombinatorischen Explosion beim Retrieval führt (siehe Abschnitt 1.6.1)). Aus diesem Grunde soll eine Retrieval-Strategie bei einer Dokumentvektor-Polyrepräsentation nur in Verbindung mit einer Queryvektor-Monorepräsentation beschrieben werden, und auf die Kombination mit einer Queryvektor-Polyrepräsentation soll hier verzichtet werden.

Bei einer Query-Monorepräsentation wird von Q_i^t ausgegangen, die durch genau eine Indexierungsfunktion $A_{IR(Q)}$ auf den Queryvektor q_i^t abgebildet wird. Dieser Vektor nimmt genau einen Punkt im Dokumentvektorraum DVR ein, und kann als Mittelpunkt einer ε-Umgebung $U(q_i^t \mid \varepsilon)$ verwendet werden (siehe Abb. 46)).

Abb. 46) Dokumentvektor-Polyrepräsentation und Queryvektor-Monorepräsentation



Für jede der Dokumentvektorenlisten BDV_k^t kann nun eine Retrievalmenge oder -liste an Dokumentvektoren bestimmt werden, die innerhalb der Retrieval-Mannigfaltigkeit liegt:

$$DV(q_i^t)_k = (x_k^t \in BDV_k^t \mid d_{DVR}(q_i^t, x_k^t) < d_{DVR}(q_i^t, x_{k+1}^t) < \varepsilon). \quad (324)$$

Diese Einzelmengen werden in einem weiteren Schritt aggregiert (Combining), wobei die Verfahren verwendet werden können, die bei der Query-Polyrepräsentation in Verbindung mit einer Dokument-Monorepräsentation vorgeschlagen wurden.

3.6.5) Retrievalstrategien bei einer Retrievalregion-Mono- und -Polyrepräsentation

Die bislang dargestellten Retrievalstrategien mit einer Polyrepräsentation gingen von unterschiedlichen Queryvektoren in einer Dokumentvektorenverteilung aus, ohne dass das Verfahren in die Repräsentation einbezogen wurde, mit dem die konkrete Auswahl von Dokumentvektoren erfolgte. Es wurde standardmäßig von einer ε-Umgebung $U(q_i^t \mid \varepsilon)$ ausgegangen, wobei der Umgebungsparameter extern gegeben und konstant gehalten wurde. D.h. diese Vorgehensweisen können jeweils als Retrievalregion-Monorepräsentation betrachtet werden. Im weiteren sollen Strategien in Verbindung mit einer Retrievalregion-Polyrepräsentation betrachtet werden. Eine Retrievalregion-Polyrepräsentation ergibt sich immer dann, wenn innerhalb eines Retrievalprozesses unterschiedliche Umgebungsparameter verwendet werden, unabhängig ob verschiedene Hyperkugeln $U(q_i^t \mid \varepsilon_k \in R^+)$, achsenparallele Hyperellipsoide $U(q_i^t \mid \varepsilon_k \in R^{+n})$ oder rotierte Hyperellipsoide $U(q_i^t \mid \varepsilon_k \in R^{+n}, \alpha_k \in R^{+[n(n-1)/2]})$ verwendet werden. Im weiteren soll stellvertretend $\varepsilon_k \in R^{+n}$ betrachtet werden, d.h. bei einer Retrievalregion-Polyrepräsentation existiert eine Liste von Parametervektoren $(\varepsilon_k \mid k = 1, ..., \delta)$, die für die Spezifikation von Retrievalregionen verwendet werden.

Wiederum soll auf eine detaillierte Beschreibung einer mehrfachen Polyrepräsentation auf grund der kombinatorischen Explosion verzichtet werden (siehe Abschnitt 1.6.1)), zumindest was die Kombination mit einer Dokumentvektoren-Polyrepräsentation anbelangt. D.h. es soll eine Retrievalregion-Polyrepräsentation bei Queryvektor-Mono- und Polyrepräsentation kurz beschrieben werden.

Die erste Kombination ist eindeutig definiert, da um genau einen Queryvektor q_i^t δ unterschiedliche Retrievalmannigfaltigkeiten $U(q_i^t \mid \varepsilon_k)$ festgelegt werden, wobei die Liste $(\varepsilon_k \mid k = 1, ..., \delta)$ externspezifiziert sein soll. Jeder dieser Retrievalmannigfaltigkeiten führt zu einer Retrieval-Dokumentvektorenliste $DV(q_i^t \mid \varepsilon_k)$, die im Rahmen eines Combining-Modells zu einer Gesamtliste aggregiert werden, deren einzelnen Komponenten eventuell mit mehreren Dokumentvektoren besetzt sein kann (siehe Abb. 48)).

Liegen mehrere Queryvektoren und mehrere Umgebungsparameter vor, so muss damit noch keine mehrfache Polyrepräsentation verbunden sein, da eine 1-zu-1-Zuordnung von Queryvektor und Umgebungsparameter möglich ist, insbesondere dann, wenn die Anzahl der Queryvektoren (δ_q) gleich der Anzahl der Parametervektoren (δ_ε) ist, d.h. es gilt $\delta := \delta_q = \delta_\varepsilon$. Insgesamt existieren drei Möglichkeiten bezüglich der Verknüpfung zwischen Queryvektor und Umgebungsparameter:

1) 1-zu-1-Verknüpfung, wobei zu klären ist, welches Zuordnungsverfahren verwendet werden soll (z.B. zufällige Paarung) (siehe Abb. 47)).
2) 1-zu-δ-Verknüpfung, d.h. jeder Queryvektor erzeugt δ Retrievalregionen.
3) 1-zu-d-Verknüpfungen, mit $d \leq \delta$, d.h. einem Queryvektor wird mindestens eine und maximal δ Retrievalregionen zugeordnet, wobei auch hier die Zuordnung von Umgebungsparametern zu Queryvektor geklärt werden muss (z.B. gleichverteilt zufällige Auswahl eines Elementes d aus (1, 2, ..., δ) und Zufallsauswahl ohne Zurücklegen von d Elementen aus $(\varepsilon_k \mid k = 1, ..., \delta)$).

Abb. 47) 1-zu-1-Zuordnung von Queryvektor und Umgebungsparameter im DVR

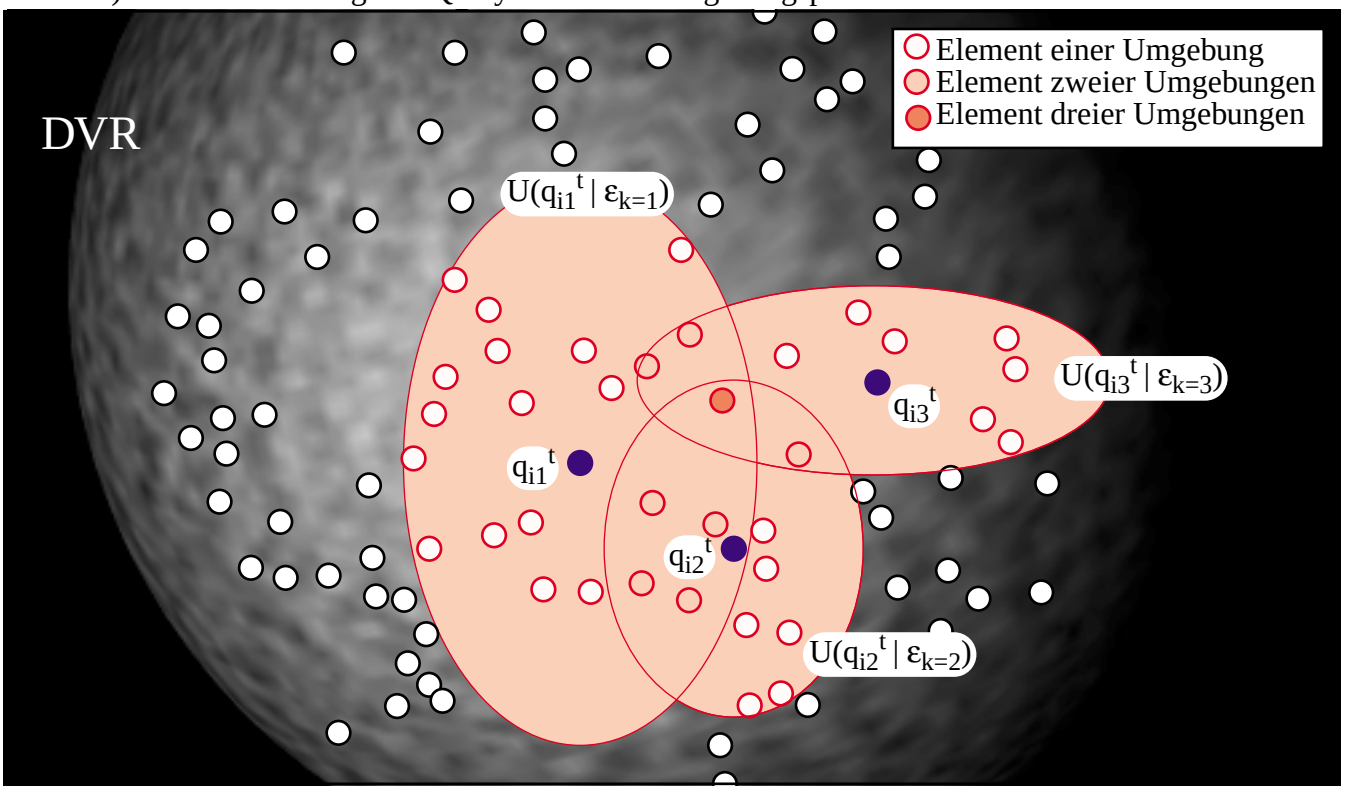
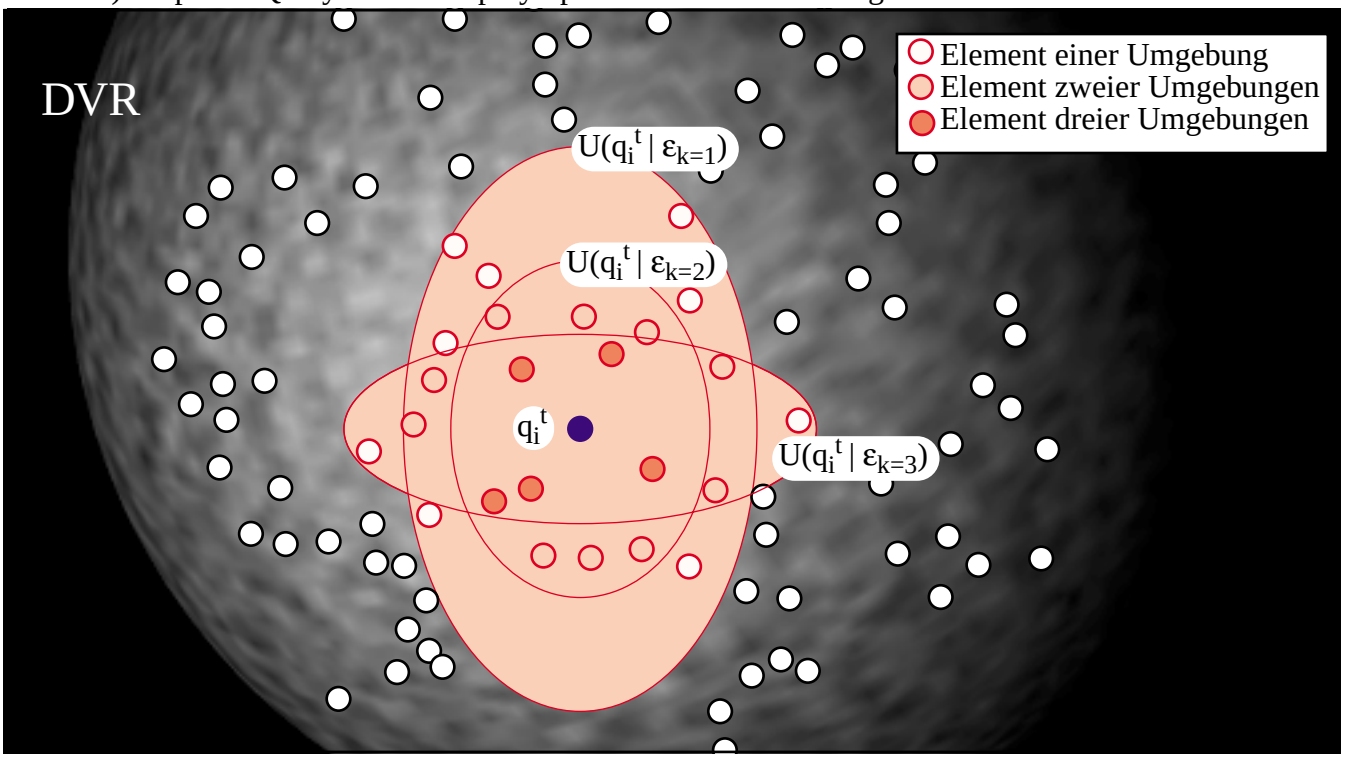


Abb. 48) Einpunkt-Queryvektor mit polyrepräsentierter Retrievalregion im DVR



Die Fälle 2) und 3) stellen eine mehrfache Polyrepräsentation dar, sodass auf eine entsprechende Darstellung verzichtet werden soll.

Prinzipiell stellt sich die Frage nach der Herkunft des oder der Umgebungsparameter bzw. ob und wie diese in einer Sequenz von Relevanzfeedback-Operationen entwickelt werden. Neben der Auswahl von Parametern aus einem konstanten Pool könnten evolutionäre Verfahren erwogen werden, bei denen der Parameter wie bei den ES Teil der zu entwickelnden Datenstruktur ist. Bei einer Queryvektor- und Retrievalregion-Polyrepräsentation mit einer 1-zu-1-Zuordnung existieren somit Individuen der Form $a_i^t = (q_i^t, \varepsilon_i^t)$ in einer Population, d.h. jedes Individuum verkörpert eine Retrievalregion mit einem Queryvektor q_i^t als Zentrum und einer umgebenden Mannigfaltigkeit, die durch den Parameter oder Parametervektor ε_i^t beschrieben wird. Jedes dieser Individuen a_i^t liefert eine Dokumentvektorenmenge $DV(a_i^t)$ als Teilmenge von DV^t als Ergebnis, wobei die Fitnessbewertung eines Individuums von der Relevanzbewertung dieser Dokumentvektoren abhängt. Durch die Einführung von Reproduktions-Operationen, die auf den Query- und den Parametervektoren basieren, werden neue Individuen erzeugt, die in der nächsten Feedback-Iteration verwendet werden. Die Reproduktions-Operationen sind problemlos, da die Rekombination von Queryvektoren und Umgebungsparametern z.B. durch das arithmetische Mittel festgelegt werden kann. Eine Mutation und die Integration einer Mutationsvariablen in die Datenstruktur führt zu einer Erweiterung zu $a_i^t = (q_i^t, \varepsilon_i^t, \delta(q)_i^t, \delta(\varepsilon)_i^t)$ mit $\delta(q)_i^t$ als dem Mutationsparameter der Queryposition und $\delta(\varepsilon)_i^t$ als dem Mutationsparameter des Umgebungsparameters $\varepsilon \in R^+$. Diese Vorgehensweise lässt sich als gleichzeitige Modifikation mehrerer Repräsentationen interpretieren, d.h. als Modifikation des Queryvektors und der Retrievalregion in Form von ε_i^t (siehe hierzu Abschnitt 3.9.8)).

An einer solchen evolutionären Sichtweise sind jedoch einige Punkte nicht unproblematisch. Zunächst muss darauf hingewiesen werden, dass die Anzahl der Feedback-Iterationen bei einem menschlichen Agenten nicht ausreicht, um eine evolutionäre Suche nach einer Region bzw. einer Menge von Regionen durchzuführen. Die Fitnessbewertung muss zudem berücksichtigen, dass die Umgebung um einen Queryvektor nicht beliebig ausgedehnt werden soll, da auf diese Weise eine maximale Anzahl von relevanten Dokumentvektoren erzeugt werden kann. Um dies zu verhindern, muss die relative Anzahl an relevanten Dokumentvektoren verwendet werden, was der Maßzahl „Precision“ entspricht (siehe z.B. Panyr (1986a:303ff[247])). Die Umgebungsparameter können zudem durch Constraints begrenzt werden, d.h. die Umgebungen dürfen ein maximales Volumen nicht überschreiten, bzw. die Einzel-Parameter in einem Parametervektor dürfen nicht größer als ein Maximalwert werden.

Bei einer Queryvektor-Polyrepräsentation ist zudem zu beachten, dass die Beziehung von Individuum und Population spezifiziert werden muss. Ein Individuum kann entweder durch einen Queryvektor mit der ε-Umgebung definiert werden, oder ein Individuum kann als eine Menge von Queryvektoren und Umgebungen definiert werden, wobei eine kompatible Populationsdefinition erfolgen muss. Diese Formulierung ergibt sich daraus, dass bei einer Queryvektor-Polyrepräsentation die Einzel-Ergebnismengen aggregiert werden, sodass eine Queryvektor-Polyrepräsentation quasi als ein Individuum betrachtet werden kann.

3.6.6) Retrievalstrategien mit positiven und negativen Queries und Queryvektoren

In boolschen IRS ist die Verwendung von Merkmalen möglich, die durch ein logisches „Nicht“ oder „Not“ mit anderen Merkmalen verknüpft sind. Eine solche Aussage wird so interpretiert, dass in einem nachgewiesenen Dokument das entsprechende Merkmal nicht auftreten darf. In vektorraumbasierten IRS sind solche Konstrukte in der Regel nicht modellierbar, es sei denn, man führt negative Queries Q_i^{-t} ein, die durch die gleiche Indexierungsfunktion $A_{IR(Q)}$ auf einen Punkt im DVR abgebildet werden, wie die bislang betrachteten Queries, die im folgenden Kontext als positive Queries Q_i^{+t} bezeichnet werden. Die sich ergebenden Vektoren sollen als positive Queryvektoren q_i^{+t} bzw. negative Queryvektoren q_i^{-t} bezeichnet werden.

Insbesondere die Verwendung von Dokumenten bzw. Textteilen, die von einem Agenten als negatives Beispiel oder als Abgrenzung zu einem positiven Informationsbedürfnis bewertet werden, können als negative Query interpretiert und verwendet werden. Weiterhin können auch Dokumente bzw. Textteile, die als irrelevant oder irreführend bewertet werden, hierzu verwendet werden. Diese Relevanzbewertung besitzt somit eine direkte Beziehung zu den Relevanzbewertungen im Rahmen des Relevanz-Feedbacks (siehe Abschnitt 3.9)). Dort werden nachgewiesene Dokumente durch den Agenten bewertet, wobei relevante wie nicht relevante Dokumente einen Adaptionsprozess verursachen, der bei einem Queryvektor-Feedback die Position des Queryvektors verändert.

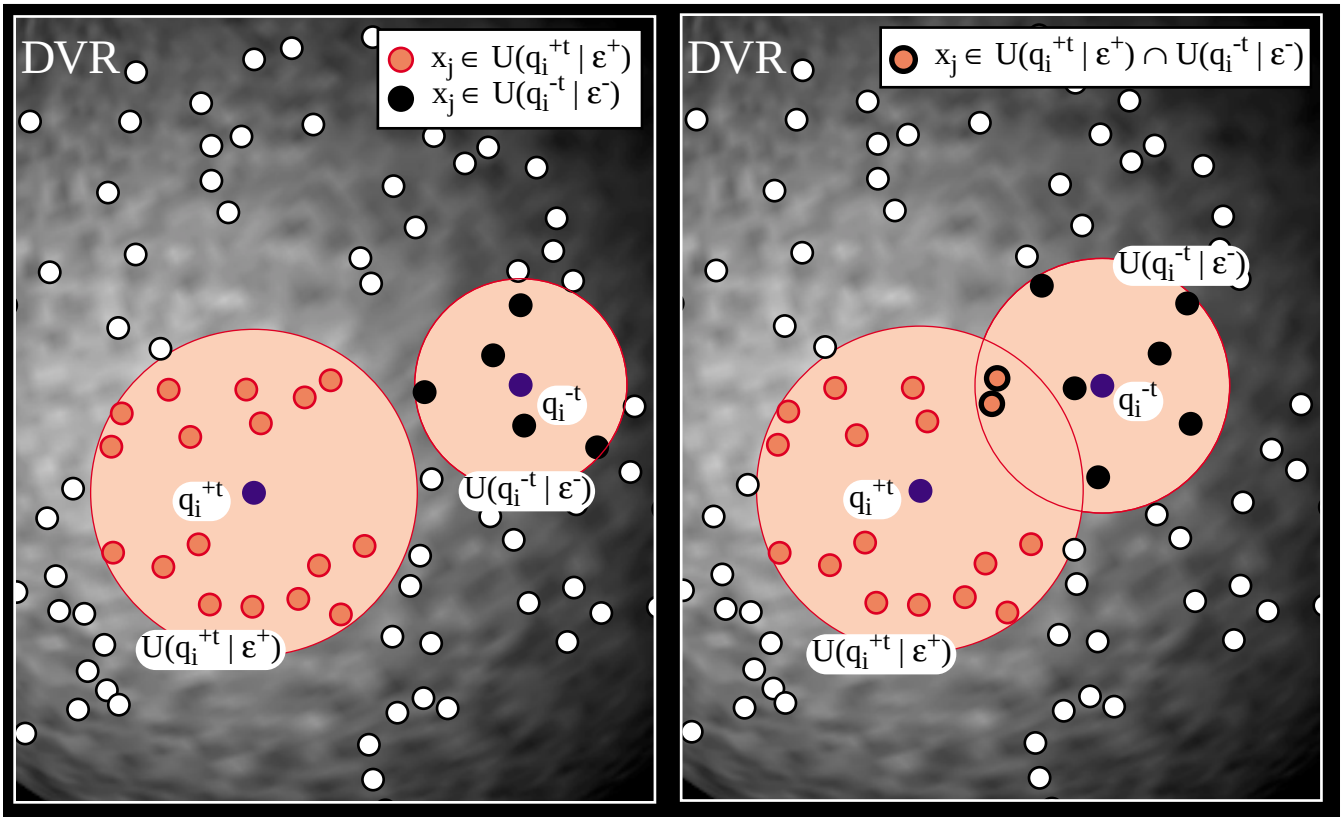
Andererseits besitzt die Unterscheidung zwischen positiven und negativen Queries eine direkte Beziehung zu positiven und negativen Filtern im Rahmen der Information-Filter-Systeme (z.B. Belkin & Croft (1992[36]), Goldberg et al. (1992[147]), Panyr (1994[253]), Allan (1996[5]), Mostafa et al. (1997[227]), Schapire et al. (1998[299])), wobei ein negativer Filter als eine langfristig im IFS gehaltene Repräsentation eines negativen Informationsbedürfnis oder einer Abgrenzung zu einem positiven Informationsbedürfnis betrachtet wird.

3.6.6.1) Monorepräsentation von positiven und negativen Queryvektoren

In den weiteren Darstellungen soll die Betrachtung eines Konstruktes wie einer negativen Query als Abgrenzung zu einem positiven Konstrukt interpretiert werden, da durch diese Interpretation ein topologischer bzw. geometrischer Aspekt eingeführt wird, der im Kontext eines metrischen Vektorraumes wie dem DVR operationalisiert werden kann. Dies kann in der gleichen Weise wie bei einer positiven Query dadurch geschehen, dass um den negativen Queryvektor q_i^{-t} eine ε-Umgebung $U(q_i^{-t} \mid \varepsilon^-)$ erzeugt wird, die als negative Queryregion bezeichnet werden soll, wobei eine Monorepräsentation des positiven und negativen Queryvektors unterstellt werden soll (siehe Abb. 49) und Abb. 50)). Die Umgebungsparameter ε^+ und ε^- sind dabei externe Parameter, die in dem dargestellten Beispiel ungleich ($\varepsilon^+ > \varepsilon^-$) gewählt wurden:

$$U(q_i^{+t} \mid \varepsilon^+) = \{x \in DVR \mid d_{DVR}(q_i^{+t}, x) < \varepsilon^+, \varepsilon^+ \in R^+\},$$
$$U(q_i^{-t} \mid \varepsilon^-) = \{x \in DVR \mid d_{DVR}(q_i^{-t}, x) < \varepsilon^-, \varepsilon^- \in R^+\}. \quad (325)$$

Abb. 49) Positive und negative Queryvektorregion mit leerer bzw. nicht leerer Schnittmenge



Bezüglich der Lage der beiden Umgebungen $U(q_i^{+t} \mid \varepsilon^+)$ und $U(q_i^{-t} \mid \varepsilon^-)$ können folgende Szenarien unterschieden werden wobei 2.2) und 2.3) als Extremfälle nicht weitergehend behandelt werden sollen:

1) $U(q_i^{+t} \mid \varepsilon^+) \cap U(q_i^{-t} \mid \varepsilon^-) = \varnothing$ (siehe Abb. 49)a)).
2) $U(q_i^{+t} \mid \varepsilon^+) \cap U(q_i^{-t} \mid \varepsilon^-) \neq \varnothing$ (siehe Abb. 49)b) bis Abb. 50)b).
   2.1) $U(q_i^{+t} \mid \varepsilon^+) \setminus U(q_i^{-t} \mid \varepsilon^-) \neq \varnothing$ (siehe Abb. 49)b).
   2.2) $U(q_i^{-t} \mid \varepsilon^-) \subset U(q_i^{+t} \mid \varepsilon^+)$ (siehe Abb. 50)a)).
   2.3) $U(q_i^{+t} \mid \varepsilon^+) \subset U(q_i^{-t} \mid \varepsilon^-)$ (siehe Abb. 50)b)).

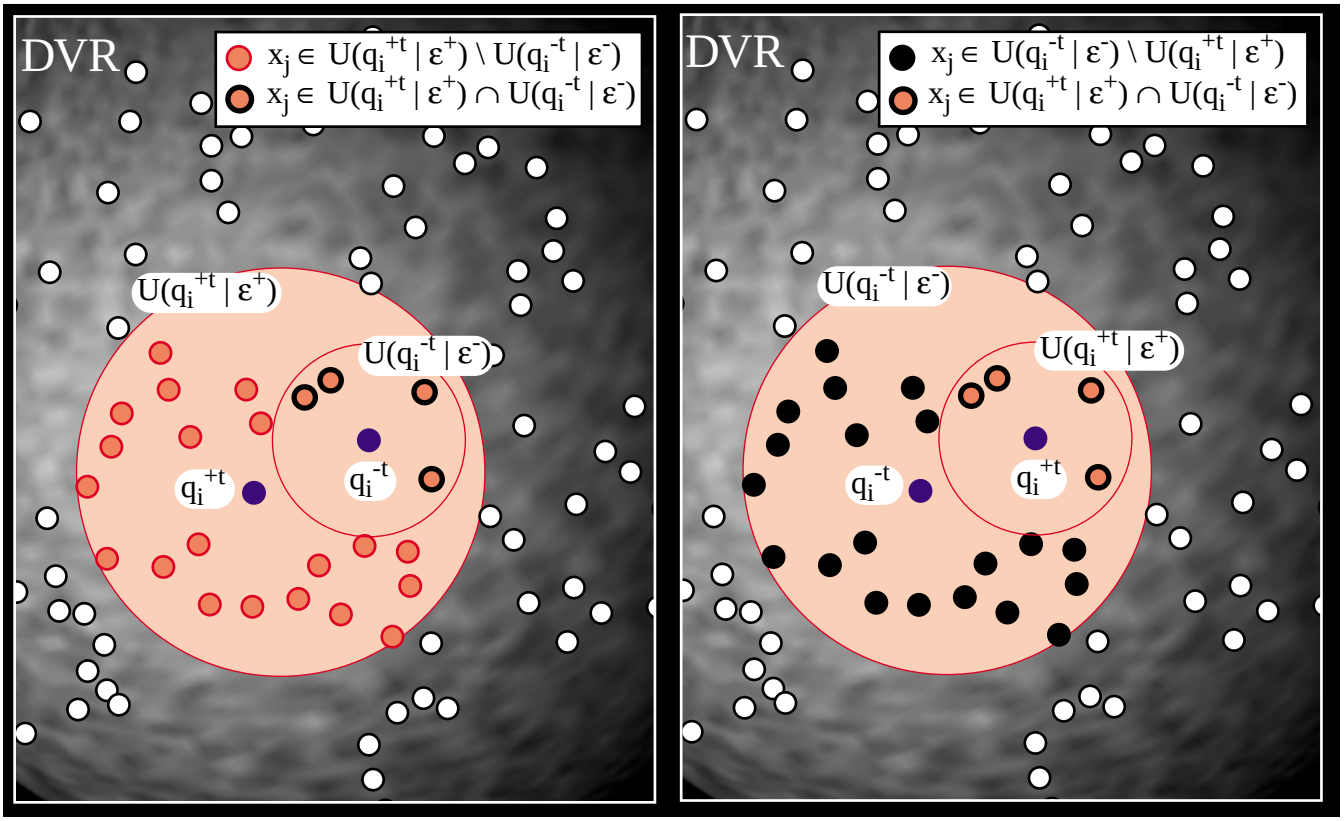
Die Umgebungen werden im Rahmen einer Retrieval-Operation betrachtet, sodass die Dokumentvektorenmengen $DV(q_i^{+t})$ und $DV(q_i^{-t})$ abgeleitet werden sollen:

$$DV(q_i^{+t}) = \{x_j^t \in DV^t \mid d_{DVR}(q_i^{+t}, x_j^t) < \varepsilon^+, \varepsilon^+ \in R^+\},$$
$$DV(q_i^{-t}) = \{x_j^t \in DV^t \mid d_{DVR}(q_i^{-t}, x_j^t) < \varepsilon^-, \varepsilon^- \in R^+\}. \quad (326)$$

Genau wie die Umgebungen können die Dokumenvektorenmengen ebenfalls leere oder nicht leere Schnittmengen bilden, wobei dies im Hinblick auf die gemeinsame Ergebnismenge $DV(q_i^{+t}, q_i^{-t})$, die aus dem gemeinsamen Auftreten des positiven und negativen Queryvektors hervorgeht, relevanter ist, als die Lage der Umgebungen. Problemlos ist der Fall einer leeren Schnittmenge $DV(q_i^{+t}) \cap DV(q_i^{-t}) = \varnothing$, da dies so behandelt werden kann, als wenn keine negative Query vorhanden wäre:

$$DV(q_i^{+t}) \cap DV(q_i^{-t}) = \varnothing: DV(q_i^{+t}, q_i^{-t}) = DV(q_i^{+t}). \quad (327)$$

Abb. 50) Überlagernde positive und negative Queryvektorregion



Liegt eine nicht leere Schnittmenge vor, so ist die Dokumentvektorenteilmenge in jedem Fall einzuschließen, die in $DV(q_i^{+t})$ jedoch nicht in der gemeinsamen Schnittmenge liegen, d.h. in $DV(q_i^{-t})$ jedoch nicht in $DV(q_i^{-t})$:

$$\forall x_j^t \in \{DV(q_i^{+t}) \setminus DV(q_i^{-t})\}: x_j^t \in DV(q_i^{+t}, q_i^{-t}). \tag{328}$$

Demgegenüber können die Dokumentvektoren ausgeschlossen werden, die in $DV(q_i^{-t})$ jedoch nicht in der gemeinsamen Schnittmenge liegen, d.h. die in $DV(q_i^{-t})$ jedoch nicht in $DV(q_i^{+t})$ liegen:

$$\forall x_j^t \in \{DV(q_i^{-t}) \setminus DV(q_i^{+t})\}: x_j^t \notin DV(q_i^{+t}, q_i^{-t}). \tag{329}$$

Zu klären ist somit noch die Zugehörigkeit der Dokumentvektoren in der Schnittmenge $DV(q_i^{+t}) \cap DV(q_i^{-t})$. Werden Entscheidungen ausschließlich auf der Basis von Mengen getroffen, so existieren nur zwei Alternativen, d.h. entweder werden alle Elemente der Schnittmenge verworfen

$$\forall x_j^t \in \{DV(q_i^{+t}) \cap DV(q_i^{-t})\}: x_j^t \notin DV(q_i^{+t}, q_i^{-t}), \tag{330}$$

oder alle Elemente der Schnittmenge werden akzeptiert:

$$\forall x_j^t \in \{DV(q_i^{+t}) \cap DV(q_i^{-t})\}: x_j^t \in DV(q_i^{+t}, q_i^{-t}). \tag{331}$$

Dokumente, deren Dokumentvektor in einer Schnittmenge der Queryvektorregion eines positiven und eines negativen Queryvektors liegen, besitzen Teile, die als relevant bezüglich der positiven Query bewertet werden müssen, da der entsprechende Dokumentvektor sonst nicht in $U(q_i^{+t} \mid \varepsilon^+)$ liegen würde. Da er zusätzlich in $U(q_i^{-t} \mid \varepsilon^-)$ liegt, bedeutet dies, dass das korrespondierende Dokument auch Teile

besitzt, die als irrelevant bewertet werden. Ob der Agent Dokumente mit diesen Eigenschaften entgegen nimmt, wird somit zu einem externen Parameter, der von einem Effizienz-Effektivitäts-Trade-off beeinflusst wird. Aus Effektivitätsgründen wäre die Überprüfung von Dokumenten aus der Schnittmenge $DV(q_i^{+t}) \cap DV(q_i^{-t})$ sinnvoll, doch dies ist im Rahmen des Relevanz-Feedbacks mit einem erhöhten Aufwand von Seiten des Agenten verbunden.

Eine Zergliederung von Dokumenten in Passagen wäre in diesen ambiguen Fällen wünschenswert, da auf diese Weise jede der Passagen als eigenes Dokument behandelt wird, das indexiert wird, und somit einen Punkt im DVR einnimmt. Für die Elemente dieser Punktmenge wird dann die Zuordnung zu $U(q_i^{+t} \mid \varepsilon^+)$ bzw. $U(q_i^{-t} \mid \varepsilon^-)$ bestimmt, wobei durch die Zerlegung die Wahrscheinlichkeit der Zuordnung in $U(q_i^{+t} \mid \varepsilon^+) \cap U(q_i^{-t} \mid \varepsilon^-)$ verkleinert wird. Dies bedeutet jedoch, dass hier nicht garantiert werden kann, dass keine Passagen in der Schnittmenge liegen, sodass die grundsätzliche Entscheidung durch den Agenten ebenfalls notwendig ist. Auf die Möglichkeiten eines Passagen-Retrievals soll hier jedoch nicht weiter eingegangen werden.

Die Aufnahme eines Dokumentes aus der Schnittmenge in die Ergebnismenge kann durch die Einführung eines Rankings relativiert werden, indem diese Elemente auf die hinteren Ränge verwiesen werden. Auf diese Weise kann man die Dokumente dem Agenten präsentieren, der gleichzeitig die Kontrolle darüber besitzt, nach welchem Rankingplatz er die Analyse der Dokumente in der präsentierten Dokument-Sequenz abbricht. Eine entsprechende Rankingstrategie ordnet zunächst alle Elemente aus $DV(q_i^{+t})$ nach steigenden Distanzen zu q_i^{+t} gefolgt von der Ordnung der Elemente in der Schnittmenge $DV(q_i^{+t}) \cap DV(q_i^{-t})$ nach steigenden Distanzen zu q_i^{+t}. D.h. ein Dokumentvektor kann trotz einer kleineren Distanz weiter hinten im Gesamtranking liegen als ein anderer Dokumentvektor, weil er Element der Schnittmenge, während der andere Vektor Element von $DV(q_i^{+t}) \setminus DV(q_i^{-t})$ ist.

3.6.6.2) Polyrepräsentation von positiven und negativen Queryvektoren

Werden durch einen Agenten mehrere positive und negative Queries spezifiziert, die zusammen ein Informationsbedürfnis beschreiben, so handelt es sich dabei um eine natürliche Polyrepräsentation der Queries, im Gegensatz zu künstlichen Polyrepräsentationen, bei denen aus je einer positiven und negativen Query mehrere Queries bzw. Queryvektoren generiert werden.

Bei einer natürlichen Polyrepräsentation existieren zwei Szenarien des Retrievals, während bei einer künstlichen Polyrepräsentation nur das zweite Szenario sinnvoll ist:

1) Aggregation der positiven und der negativen Queries zu je einer Gesamt-Query und Durchführung einer entsprechenden Retrievalstrategie.
2) Verwendung der unaggregierten positiven und negativen Queries in einer Retrievalstrategie, und Aggregation der Ergebnisse im Rahmen eines angepassten Rankingverfahrens.

Unproblematisch sind die Dokumentvektoren, die in der Region von mindestens einem positiven Queryvektor liegen, ohne in der Schnittmenge der Retrievalregion eines negativen Queryvektors zu liegen. Diese Dokumentvektoren können direkt in der Ergebnismenge aufgenommen werden, während Dokumentvektoren, die in der Region eines negativen Queryvektors liegen, ohne dass sie auch gleichzeitig in

der Region eines positiven Queryvektors liegen, direkt verworfen werden. Problematisch sind somit die Fälle, bei denen ein Dokumentvektor in einer oder mehreren positiven, jedoch gleichzeitig in einer oder mehreren negativen Queryvektorregionen liegt. Es soll die prinzipielle Strategie verfolgt werden, dass alle Dokumentvektoren, die in mindestens einer positven Queryvektorregion liegen, akzeptiert werden, und dass die weiteren Attribute wie Anzahl der positiven und negativen zugeordneten Umgebungen oder Distanzwerte zu positiven und negativen Queryvektoren, den Rang innerhalb des Gesamt-Rankings aller akzeptierter Dokumentvektoren bestimmen.

Sei QVM_i^{+t} die Menge der positiven Queryvektoren und QVM_i^{-t} die Menge der negativen Queryvektoren eines einzelnen Agenten Ag_i^t:

$$QVM_i^{+t} = \{q_{ij}^{+t} \in DVR \mid j = 1, ..., m_{iQ+}^t\},$$
$$QVM_i^{-t} = \{q_{ij}^{-t} \in DVR \mid j = 1, ..., m_{iQ-}^t\}. \quad (332)$$

Die Teilmenge der Dokumentvektoren aus DV^t, die in mindestens einer Retrievalregion liegen, soll als $DV(QVM_i^{+-t})$ bezeichnet werden:

$$DV(QVM_i^{+-t}) = \{x_k^t \in DV^t \mid \exists_{\geq 1}\, q_{ij}^{+t}: x_k^t \in U(q_{ij}^{+t} \mid \varepsilon_{ij}^{+}) \vee \exists_{\geq 1}\, q_{ij}^{-t}: x_k^t \in U(q_{ij}^{-t} \mid \varepsilon_{ij}^{-})\}. \quad (333)$$

Jedem Dokumentvektor x_k^t aus $DV(QVM_i^{+-t})$ kann nun ein zwei-komponentiger Vektor zugeordnet werden, der die Anzahl der positiven Queryregionen bzw. negativen Regionen aufzählt, in der x_k^t liegt:

$$(\#U_k^{+t}, \#U_k^{-t}); \#U_k^{+t} \in \{1, ..., m_{iQ+}^t\}, \#U_k^{-t} \in \{1, ..., m_{iQ-}^t\}. \quad (334)$$

Wie oben bereits festgelegt, werden alle Dokumentvektoren aus $DV(QVM_i^{+-t})$ entfernt, die nur in negativen Regionen liegen, d.h. die einen Vektor $(0, \#U_k^{-t})$, $\#U_k^{-t} \in \{1, ..., m_{iQ-}^t\}$ besitzen. Diese Dokumentvektoren sollen in $DV(\neg QVM_i^{+t} \wedge QVM_i^{-t})$ zusammengefasst werden:

$$DV(\neg QVM_i^{+t} \wedge QVM_i^{-t}) =$$
$$\{x_k^t \in DV(QVM_i^{+-t}) \mid \forall q_{ij}^{+t}: x_k^t \notin U(q_{ij}^{+t} \mid \varepsilon_{ij}^{+}) \wedge \exists_{\geq 1}\, q_{ij}^{-t}: x_k^t \in U(q_{ij}^{-t} \mid \varepsilon_{ij}^{-})\}. \quad (335)$$

Alle weiteren Darstellungen beziehen sich auf die Restmenge von $DV(QVM_i^{+-t})$ und $DV(\neg QVM_i^{+t} \wedge QVM_i^{-t})$:

$$DV(QVM_i^{+-t})' = DV(QVM_i^{+-t}) \setminus DV(\neg QVM_i^{+t} \wedge QVM_i^{-t}). \quad (336)$$

Mit Hilfe der zwei-komponentigen Vektoren $(\#U_k^{+t}, \#U_k^{-t})$ kann bereits eine grobe Hierarchisierung der Elemente in $DV(QVM_i^{+-t})'$ vorgenommen werden, wenn das Pareto-Kriterium und ein damit verbundenes Hierachisierungsverfahren, wie die sukzessive Deaktivierung von Pareto-Mengen, durchgeführt wird (siehe Abschnitt 2.4); Bachelier (1999d:32ff[22])). Durch die Verwendung der Vektoren $(\#U_k^{+t}, \#U_k^{-t})$ wird das Problem zu einer Zwei-Ziel-Optimierung mit Maximierung der ersten Komponente und Minimierung der zweiten Komponente. Ein Dokumentvektor $x_{k(1)}^t$ dominiert entsprechend dem Pareto-Kriterium und einer 2-Ziel-Optimierung einen anderen Vektor $x_{k(2)}^t$, wenn er in einer der beiden Komponenten besser und gleichzeitig in der anderen nicht schlechter ist. Mit Hilfe einer drei-wertigen Dominanzfunktion lässt sich die Dominanz bei zwei Attributen und Maximierung des ersten sowie Minimierung des zweiten Attributes beschreiben als:

$$\begin{aligned}
& & \{1, \quad & \text{wenn } (\#U_{k(1)}^{+t} > \#U_{k(2)}^{+t} \wedge \#U_{k(1)}^{-t} \leq \#U_{k(2)}^{t}) \vee \\
& & & \text{wenn } \#U_{k(1)}^{+t} \geq \#U_{k(2)}^{+t} \wedge \#U_{k(1)}^{-t} < \#U_{k(2)}^{t}) \\
\text{dom}(x_{k(1)}^{t}, x_{k(2)}^{t}) = & & \{0, \quad & \text{wenn } (\#U_{k(1)}^{+t} > \#U_{k(2)}^{+t} \wedge \#U_{k(1)}^{-t} > \#U_{k(2)}^{t}) \vee \\
& & & \text{wenn } \#U_{k(1)}^{+t} < \#U_{k(2)}^{+t} \wedge \#U_{k(1)}^{-t} < \#U_{k(2)}^{t}. \\
& & \{-1, \quad & \text{wenn } (\#U_{k(1)}^{+t} < \#U_{k(2)}^{+t} \wedge \#U_{k(1)}^{-t} \geq \#U_{k(2)}^{t}) \vee \\
& & & \text{wenn } \#U_{k(1)}^{+t} \leq \#U_{k(2)}^{+t} \wedge \#U_{k(1)}^{-t} > \#U_{k(2)}^{t}. \qquad (337)
\end{aligned}$$

Die Pareto-Menge als Teilmenge einer Gesamtmenge bestimmt sich aus den untereinander nicht dominanten Elementen, d.h. aus Elementen, die einen Dominanzwert von Null besitzen, und die dominant gegenüber den restlichen Elementen sind.

Soll aus den Elementen der Dokumentvektorenmenge $DV(QVM_i^{+-t})'$ ein Ranking erzeugt werden, so kann mit Hilfe des Paretokriteriums eine hierarchische Clusterung durchgeführt werden, d.h. eine Form der Pareto-Hierarchie, indem eine sukzessive Deaktivierung von Pareto-Mengen durchgeführt wird (siehe Abschnitt 2.4.2.2); Bachelier (1999d:32ff[22])). Für alle Elemente x_k^t aus $DV(QVM_i^{+-t})'$ wird die Pareto-Menge $PM(DV(QVM_i^{+-t})')$ gebildet, die den ersten Rang bzw. die erste Hierarchieebene einnimmt. Danach wird die Pareto-Menge aus $DV(QVM_i^{+-t})'$ entfernt, wodurch die erste Restmenge $\Delta DV(QVM_i^{+-t})'_1$ entsteht, aus der eine neue Pareto-Menge $PM(\Delta DV(QVM_i^{+-t})'_1)$ erzeugt wird. Diese Pareto-Menge bildet den zweiten Rang, gefolgt von weiteren sukzesiven Deaktivierungen, bis die Restmenge leer wird. Dieser Vorgang wird bei einer 2-Ziel-Minimierung in Abb. 16) dargestellt. Auf diese Weise gelangt man zu einer sehr einfach zu erzeugenden, hierarchischen Strukturierung der Dokumentvektoren aus $DV(QVM_i^{+-t})'$, indem eine Dokumentvektor-Hierarchie gebildet wird:

$$DVH(QVM_i^{+-t})' = (PM(DV(QVM_i^{+-t})'), PM(\Delta DV(QVM_i^{+-t})'_p) \mid p = 1, ...,). \qquad (338)$$

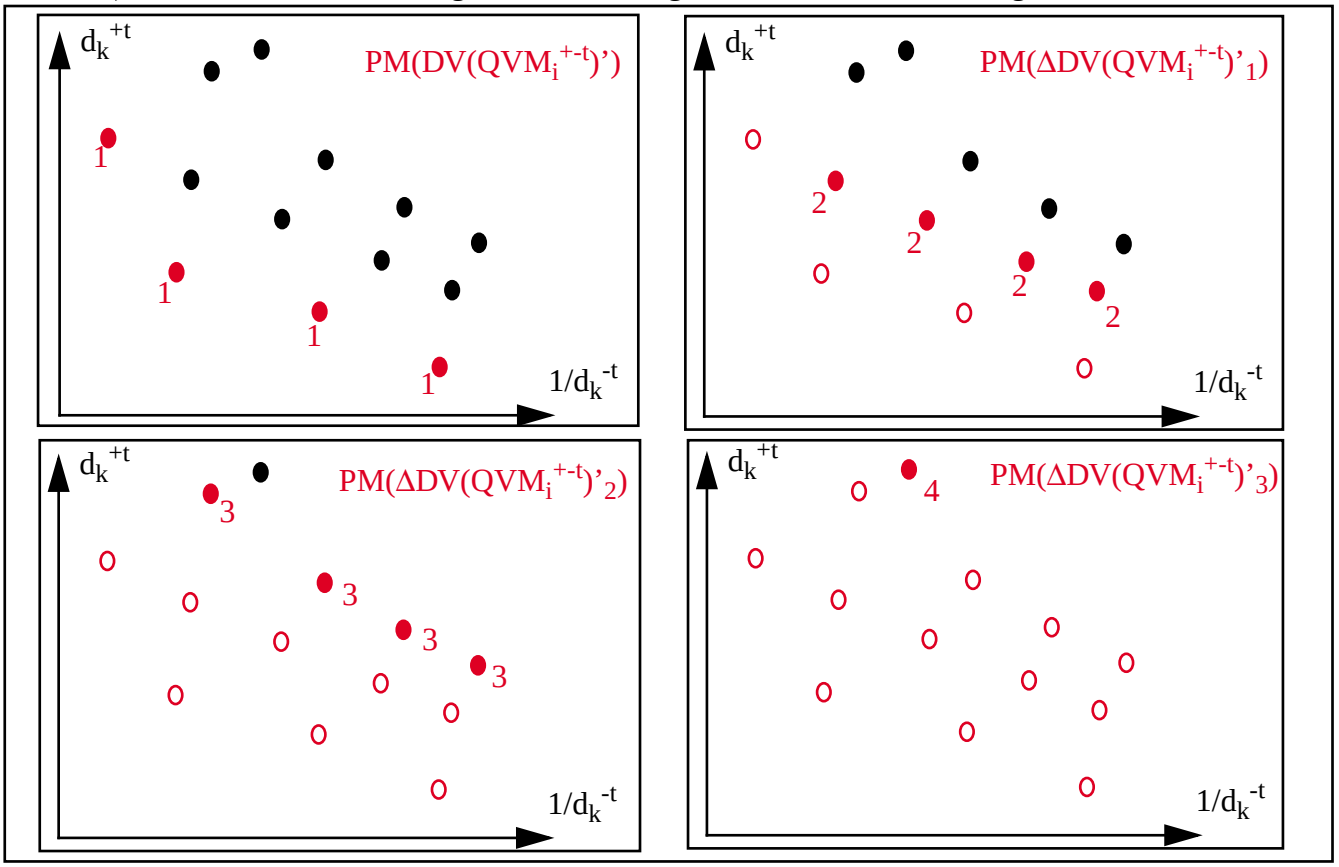
Neben dem groben Ranking auf der Basis der Zugehörigkeit zu positiven und negativen Umgebungen kann ein feineres Ranking auf der Basis von Distanzen zu den positiven und negativen Queryvektoren erzeugt werden. Da der Wertebereich der Distanzen mit R^+ größer ist als der Wertebereich zu Zugehörigkeiten mit $\#U_k^{+t} \in \{1, ..., m_{iQ+}^t\}$ und $\#U_k^{-t} \in \{1, ..., m_{iQ-}^t\}$, gelingt eine bessere Differenzierung der einzelnen Dokumentvektoren aus $DV(QVM_i^{+-t})'$. Ideal wäre der Aufbau einer Pareto-Hierarchie, bei der auf jeder Ebene sich genau ein Element befindet, d.h. jede der sukzessiv erzeugten Pareto-Mengen besitzt genau einen Dokumentvektor. Bei der Verwendung von Distanzen können zwei Strategien verwendet werden:
1) Zwei-Ziel-Optimierung.
2) $(m_{iQ+}^t + m_{iQ-}^t)$-Ziel-Optimierung.

Im ersten Fall wird für ein Dokumentvektor x_k^t aus $DV(QVM_i^{+-t})'$ die Distanzsumme zu allen positiven und allen negativen Queryvektoren gebildet:

$$\begin{aligned}
d_k^{+t} &= \sum_j d_{DVR}(x_k^t, q_{ij}^{+t}), \forall q_{ij}^{+t} \in QVM_i^{+t}, \\
d_k^{-t} &= \sum_j d_{DVR}(x_k^t, q_{ij}^{-t}), \forall q_{ij}^{-t} \in QVM_i^{-t}. \qquad (339)
\end{aligned}$$

Jedem Dokumentvektor x_k^t kann nun der zwei-komponentige Vektor (d_k^{+t}, d_k^{-t}) zugeordnet werden, wobei die erste Komponente minimiert und die zweite Komponente maximiert werden soll. Sollen beide Komponenten minimiert werden, so kann die Transformation zu (d_k^{+t}, $1/d_k^{-t}$) verwendet werden (siehe Abb. 51)).

Abb. 51) Sukzessive Deaktivierung von Paretomengen bei 2-Ziel-Minimierung von Distanzen



Bei der (m_{iQ+}^t + m_{iQ-}^t)-Ziel-Optimierung werden die Distanzen nicht aggregiert, sodass ein (m_{iQ+}^t + m_{iQ-}^t)-komponentiger Vektor ($d_{DVR}(x_k^t, q_{ij}^{+t})$, $d_{DVR}(x_k^t, q_{ij}^{-t}) \mid \forall q_{ij}^{+t} \in QVM_i^{+t}; \forall q_{ij}^{-t} \in QVM_i^{-t}$) entsteht, wobei die ersten m_{iQ+}^t Komponenten minimiert und die verbleibenden m_{iQ-} Komponenten maximiert werden sollen. Diese Option soll zugunsten der Zwei-Ziel-Optimierung hier nicht weiter verfolgt werden. Weitergehende Retrievalstrategien mit positiven und negativen Queryvektoren finden sich im Kontext der Dokumentvektoren-Clusterung (siehe Abschnitt 3.8.3)).

3.7) Clusterung in IRS

Im Zusammenhang mit dem Information Retrieval ist die Clusterung von Dokumenten und Merkmalen eine Funktion, die nach der Indexierungsfunktion $A_{IR(D)}$ durchgeführt wird. Dabei handelt es sich bei der Strukturierung um eine Rekodierungsfunktion der Dokument-Merkmals-Matrix DMM^t, wobei die Clusterung der Dokumentvektoren zu einer (m_C^t x n^t)-Matrix DMM_{DC}^t und die Clusterung der Merkmalsvektoren zu einer (m^t x n_C^t)-Matrix DMM_{FC}^t führt. Eine Integration beider Clusteroperationen führt zu einer (m_C^t x n_C^t)-Matrix DMM_{DFC}^t bzw. DMM_{FDC}^t, je nachdem welche Clusterfunktion zuerst durchgeführt wird.

3.7.1) Allgemeine Objekt- und Objektvektoren-Clusterung

Bei der Clusterung werden Objekte S_i^t zu Teilmengen zusammengefasst, die allgemein als Objekt-Cluster SC bezeichnet werden sollen, wobei die Objekte einer Teilmenge untereinander ähnlicher sein sollen, als im Vergleich zu Elementen, die nicht der gleichen Teilmenge angehören. Grundlage für die Bildung von Cluster sind die Objektvektoren, d.h. die Eigenschaftsvektoren wie die Dokumentvektoren x_i^t der Dokumente D_i^t bzw. die Merkmalsvektoren f_i^t der Merkmale F_i^t zusammen mit der verwendeten Metrik $d_{DVR}(.,.)$ im Dokumentraum bzw. $d_{FVR}(.,.)$ im Merkmalsvektorraum.

Die Menge aller Objekt-Cluster SC_i^t, die aus der Menge S^t der betrachteten Objekte S_j^t gebildet werden, heißt Objekt-Klassifikation OK^t:

$$OK^t = \{SC_i^t \mid i = 1, ..., \#OK^t\}, \text{ mit}$$
$$SC_i^t = \{S_{ij}^t \mid j = 1, ..., \#SC_i^t\}. \quad (340)$$

Die Objekt-Klassifikation wird abgeleitet aus der Objektvektoren-Klassifikation OVK^t, als der Menge der Objektvektoren-Cluster SVC_i^t , die aus den Objektvektoren s_{ij}^t gebildet werden, wobei der Objektvektorenraum SR unterstellt wird (siehe auch Abb. 52)):

$$OVK^t = \{SVC_i^t \mid i = 1, ..., \#OVK^t\}, \text{ mit}$$
$$SVC_i^t = \{s_{ij}^t \mid j = 1, ..., \#SC_i^t\}. \quad (341)$$

Eine Klassifikation OVK^t oder OK^t heißt Zerlegung (auch disjunkte Klassifikation oder Partition), wenn für alle Cluster SVC_i^t, SVC_j^t bzw. SC_i^t, SC_j^t, $i \neq j$ gilt:

$$SVC_i^t \cap SVC_j^t = \varnothing \wedge \bigcup_i SVC_i^t = SV^t, \text{ bzw.}$$
$$SC_i^t \cap SC_j^t = \varnothing \wedge \bigcup_i SC_i^t = S^t. \quad (342)$$

Eine Klassifikation OVK^t oder OK^t heißt Überdeckung, wenn mindestens ein Clusterpaar SVC_i^t, SVC_j^t bzw. SC_i^t, SC_j^t, $i \neq j$ existiert, für das gilt:

$$SVC_i^t \cap SVC_j^t \neq \varnothing \text{ bzw. } SC_i^t \cap SC_j^t \neq \varnothing. \quad (343)$$

Eine Klassifikation OVK^t oder OK^t heißt exhaustiv, wenn jedes Element der Grundmenge SV^t bzw. S^t mindestens einem Cluster zugeordnet wird:

$$\forall s_j^t \in SV^t, \exists_{\geq 1} SVC_i^t: s_j^t \in SVC_i^t,$$
$$\forall S_j^t \in S^t, \exists_{\geq 1} SC_i^t: S_j^t \in SC_i^t. \quad (344)$$

Als arithmetischer Zentroid-Vektor wird der Mittelpunktsvektor der Objektvektoren eines Clusters bezeichnet:

$$\bar{s}_{C(i)}^t = 1/\#SVC_i^t * \sum_j s_{ij}^t, \forall s_{ij}^t \in SVC_i^t. \quad (345)$$

Der Objektvektor, eines Objektes aus SC_i^t wird als Median-Objektvektor $z_{C(i)}^t$ bezeichnet, der den geringsten Abstand vom arithmetischen Zentroid-Vektor besitzt. Das zugehörige Objekt wird als Median-Objekt $Z_{C(i)}^t$ bezeichnet:

$$z_{C(i)}^t: d(z_{C(i)}^t, \bar{s}_{C(i)}^t) = \min\{d(s_{ij}^t, \bar{s}_i^t) \mid \forall s_{ij}^t \in SVC_i^t\}, Z_{C(i)}^t \in SC_i^t. \quad (346)$$

Analog der Einführung einer ε-Umgebung um einen Queryvektor im Rahmen der Retrievalstrategien, kann allgemein im Rahmen der Clusterung von Objektvektoren eine Clusterregion als ε-Umgebung um einen arithmetischen Zentroid-Vektor $\bar{s}_{C(i)}^t$ eingeführt werden:

$$U(\bar{s}_{C(i)}^t \mid \varepsilon) = \{s_j \in SR \mid d_{SR}(\bar{s}_{C(i)}^t, s_j) < \varepsilon, \varepsilon \in R^+\}. \quad (347)$$

Alternativ kann eine Clusterregion auch über den Median-Objektvektor definiert werden:

$$U(z_{C(i)}^t \mid \varepsilon) = \{s_j \in SR \mid d_{SR}(z_{C(i)}^t, s_j) < \varepsilon, \varepsilon \in R^+\}. \quad (348)$$

Eine Clusterregion kann bezüglich eines Clusters SVC_i^t individualisiert werden, indem der Umgebungsparameter ε als eine Funktion der Objektvektoren $s_{ij}^t \in SVC_i^t$ berechnet wird. Beispielsweise kann er als $\varepsilon_{C(i)}^t$ als Abstand zwischen arithmetischen Zentroid-Vektor und dem am weitesten entfernten Objektvektor definiert werden:

$$\varepsilon_{C(i)}^t = d_{SR}(\bar{s}_{C(i)}^t, s_{C(i),max}^t) \text{ mit}$$
$$d_{SR}(\bar{s}_{C(i)}^t, s_{C(i),max}^t) = \max\{d_{SR}(\bar{s}_{C(i)}^t, s_{ij}^t) \mid \forall s_{ij}^t \in SVC_i^t\}, \text{ oder}$$
$$\varepsilon_{C(i)}^t = d_{SR}(z_{C(i)}^t, s_{C(i),max}^t) \text{ mit}$$
$$d_{SR}(z_{C(i)}^t, s_{C(i),max}^t) = \max\{d_{SR}(z_{C(i)}^t, s_{ij}^t) \mid \forall s_{ij}^t \in SVC_i^t\}. \quad (349)$$

Zwei Clusterregionen $U(\bar{s}_{C(i)}^t \mid \varepsilon_{C(i)}^t)$ und $U(\bar{s}_{C(j)}^t \mid \varepsilon_{C(j)}^t)$, $i \neq j$ heißen überlappungsfrei bezüglich der Clusterregionen, wenn gilt:

$$U(\bar{s}_{C(i)}^t \mid \varepsilon_{C(i)}^t) \cap U(\bar{s}_{C(j)}^t \mid \varepsilon_{C(j)}^t) = \varnothing. \quad (350)$$

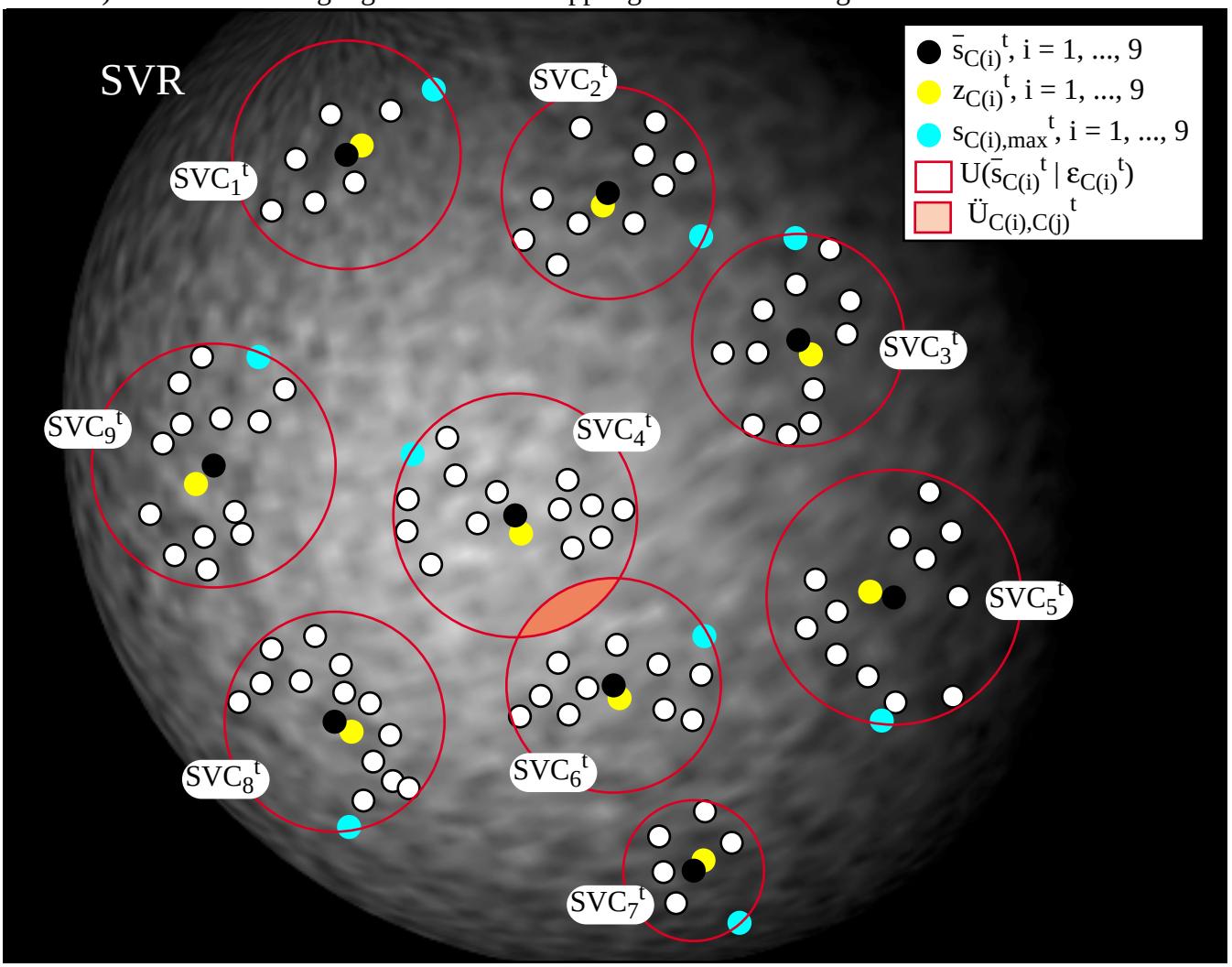
Eine Klassifikation OVK^t heißt überlappungsfrei bezüglich ihrer Clusterregionen, wenn für alle Clusterregionen $U(\bar{s}_{C(i)}^t \mid \varepsilon_{C(i)}^t)$ und $U(\bar{s}_{C(j)}^t \mid \varepsilon_{C(j)}^t)$, $i \neq j$, die Überlappungsfreiheit gilt.

Sind zwei Clusterregionen $U(\bar{s}_{C(i)}^t \mid \varepsilon_{C(i)}^t)$ und $U(\bar{s}_{C(j)}^t \mid \varepsilon_{C(j)}^t)$, $i \neq j$ nicht überlappungsfrei, so besitzen sie eine Überlappungsregion $\ddot{U}_{C(i),C(j)}^t$, die definiert wird als Menge der Punkte des Objektvektorenraums SR, die in beiden Umgebungen liegen:

$$\ddot{U}_{C(i),C(j)}^t = \{s_j \in SR \mid s_j \in U(\bar{s}_{C(i)}^t \mid \varepsilon_{C(i)}^t) \wedge s_j \in U(\bar{s}_{C(j)}^t \mid \varepsilon_{C(j)}^t)\}. \quad (351)$$

In Abb. 52) wird eine exhaustive Zerlegung von Objektvektoren im Raum SR dargestellt, bei der 9 Objektvektoren-Cluster SVC_1^t, ..., SVC_9^t entstehen. Die beiden Clusterregionen $U(\bar{s}_{C(4)}^t \mid \varepsilon_{C(4)}^t)$ und $U(\bar{s}_{C(6)}^t \mid \varepsilon_{C(6)}^t)$ besitzen eine Überlappungsregion, sodass die gesamte Klassifikation als nicht überlappungsfrei bezüglich ihrer Clusterregionen gilt.

Abb. 52) Exhaustive Zerlegung mit nicht überlappungsfreien Clusterregionen



3.7.2) Dokumentvektoren-Clusterung

Bei der Dokumentvektoren-Clusterung werden die Dokumentvektoren $x_i^t = (x_{ij}^t \mid j = 1, ..., n^t)$ der Dokumente D_i^t verwendet, um Teilmengen zu bilden, deren Elemente untereinander geringere Distanzen besitzen, als im Vergleich zu Elementen, die nicht der gleichen Teilmenge angehören, wobei die Distanzmetrik $d_{DVR}(.,.)$ im Dokumentvektorraum DVR verwendet wird. Die Menge aller Dokumentvektoren-Cluster DVC_i^t, soll mit Dokumentvektoren-Klassifikation DVK^t bezeichnet werden:

$$DVK^t = \{DVC_i^t \mid i = 1, ..., m_C^t\}, \text{ mit}$$
$$DVC_i^t = \{x_{ij}^t \mid j = 1, ..., m_{C(i)}^t\}. \quad (352)$$

Mit der Klassifikation DVK^t wird die ursprüngliche Dokument-Merkmals-Matrix DMM^t reformuliert, indem die vorher m^t-dimensionalen Merkmalsvektoren f_i^t zu m_C^t-dimensionalen Merkmalsvektoren $f_{C,i}^t$ durch eine Rekodierungsfunktion $rec(DMM^t)$ umgewandelt werden. D.h. die (m^t x n^t)-Matrix DMM^t wird zu einer (m_C^t x n^t)-Matrix, die mit DMM_{DC}^t bezeichnet werden soll. Aus dem ursprünglich m^t-dimensionalen Merkmalsvektorraum FVR wird somit der m_C^t-dimensionale neue Merkmalsvektorraum FVR_C unter Beibehaltung der Metrik $d_{FVR}(., .)$. Die Dokumentvektoren-Clusterung lässt sich durch die Clusterfunktion cl(.) und die Rekodierungsfunktion rec(.) beschreiben als:

$$cl(DV^t) = DVK^t \rightarrow rec(DMM^t) = DMM_{DC}{}^t. \quad (353)$$

Der Dokumentvektoren-Klassifikation DVK^t entspricht eine Dokument-Klassifikation DK^t :

$$DK^t = \{DC_i{}^t \mid i = 1, ..., m_C{}^t\}, \text{ mit}$$
$$DC_i{}^t = \{D_{ij}{}^t \mid j = 1, ..., m_{C(i)}{}^t\}. \quad (354)$$

Der arithmetische Zentroid-Dokumentvektor $\bar{x}_{C(i)}{}^t$ des Clusters $DVC_i{}^t$ wird berechnet durch:

$$\bar{x}_{C(i)}{}^t = 1/\#DVC_i{}^t * \sum_j x_{ij}{}^t, \forall x_{ij}{}^t \in DVC_i{}^t. \quad (355)$$

Der Dokumentvektor, der $\bar{x}_{C(i)}{}^t$ am nächsten liegt, wird als Median-Dokumentvektor $z_{C(i)}{}^t$ bezeichnet, das zugehörige Dokument als Median-Dokument $Z_{C(i)}{}^t$ bezeichnet:

$$z_{C(i)}{}^t: d_{DVR}(z_{C(i)}{}^t, \bar{x}_{C(i)}{}^t) = \min\{d_{DVR}(x_{ij}{}^t, \bar{x}_i{}^t) \mid \forall x_{ij}{}^t \in DVC_i{}^t\}, Z_{C(i)}{}^t \in DC_i{}^t. \quad (356)$$

Diese Darstellungen können als Monorepräsentation von Dokument- und Dokumentvektoren-Clusterung interpretiert werden, zu denen Polyrepräsentationen erzeugbar sind. Ziel ist der Aufbau einer Dokumentvektoren-Polyklassifikation $BDVK^t$ bzw. Dokument-Polyklassifikation BDK^t , wobei wiederum Bootstrapverfahren unterstellt werden können:

$$BDVK^t = \{DVK^t, BDVK_k{}^t \mid k = 1, ..., \delta\} \text{ bzw.}$$
$$BDK^t = \{DK^t, BDK_k{}^t \mid k = 1, ..., \delta\}. \quad (357)$$

Danach liegen $\delta + 1$ Klassifikationen vor, für die jeweils die ursprüngliche Dokument-Merkmals-Matrix DMM^t reformuliert werden kann. Bei der Rekodierung von DMM^t durch DVK^t soll $DMM_{DC}{}^t$, und bei der Rekodierung von DMM^t durch $BDVK_k{}^t$ sollen die Matrizen $DMM_{DC,k}{}^t$ erzeugt werden. Die Cluster- und Rekodierungsprozesse lassen sich somit umschreiben als:

$$cl(DV^t) = DVK^t \rightarrow rec(DMM^t) = DMM_{DC}{}^t,$$
$$cl(BDV_k{}^t) = BDVK_k{}^t \rightarrow rec(DMM^t) = DMM_{DC,k}{}^t. \quad (358)$$

Genau wie bei der Polyrepräsentation der Indexierung sind die folgenden Fälle zu unterscheiden:
1) Es existiert bereits eine Dokumentvektor-Polyrepräsentation $BDV^t = \{DV^t, BDV_k{}^t \mid k = 1, ..., \delta\}$.
2) Es liegt nur eine Monorepräsentation von DV^t vor.

Liegt eine Polyrepräsentation vor, so können die einzelnen Listen unabhängig in Cluster zerlegt werden, d.h. aus den Bootstraplisten $BDV_k{}^t$ können direkt und parallel die Klassifikationen $BDVK_k{}^t$ gebildet werden. Liegt eine Monorepäsentation vor, so kann eine Polyrepräsentation nur durch verschiedene Clusterungsoperationen durchgeführt werden, beispielsweise indem ein Clusterverfahren mit verschiedenen Parametern unabhängig mehrmals durchgeführt wird. Alternativ kann bei Verfahren wie Single-Pass-Clusterungen die Reihenfolge der Elementauswahl der Objekte zufällig und unabhängig mehrmals durchgeführt wird, wobei jeweils die gleichen sonstigen Verfahrensparameter verwendet werden.

Besonders effizient können Verfahren gestaltet werden, bei denen zunächst die Ursprungsliste DV^t in eine Klassifikation BDV^t überführt wird, d.h. indem genau eine Clusterung durchgeführt wird. Danach werden δ Bootstraplisten ${BDV_k}^t$ der Dokumentvektoren durch Ziehen mit Zurücklegen aus DV^t erzeugt. Die Ursprungsklassifikation wird sodann δ mal kopiert, und die zugeordneten Objekte werden denen der Bootstraplisten angepasst, d.h. fehlende Objekte werden gelöscht, und mehrmals vorhandene Objekte werden neu eingefügt, wobei beachtet werden muss, dass in diesem Szenario Cluster nicht als einfache Mengen, sondern als Multimengen oder Listen definiert werden müssen, um das mehrfache Auftreten von Objekten zu ermöglichen. Im letzten Schritt werden die Zentroid-Vektoren der einzelnen Klassifikationen angepasst, d.h. es findet eine Feinadaption statt. In Bachelier (1999d: 103ff[22]) werden eine Reihe von Verfahren, u.a. auch Resampling-Verfahren, beschrieben, die Modellvarianten aus einem Primärmodell ableiten, wobei es sich bei dem Primär- wie bei den Sekundärmodellen um SC-GNG-SOMs handelt.

3.7.3) Merkmalsvektoren-Clusterung

Bei der Merkmalsvektoren-Clusterung werden die Vektoren $f_i^t = (f_{ij}^t \mid j = 1, ..., m^t)$ der Merkmale F_i^t verwendet, um Teilmengen zu bilden, deren Elemente untereinander geringere Distanzen besitzen, als im Vergleich zu Elementen, die nicht der gleichen Teilmenge angehören, wobei die Distanzmetrik $d_{FVR}(.,.)$ im Merkmalsvektorraum FVR verwendet wird. Die Menge aller Merkmalsvektoren-Cluster FVC_i^t, soll mit Merkmalsvektoren-Klassifikation FVK^t bezeichnet werden:

$$FVK^t = \{FVC_i^t \mid i = 1, ..., {n_C}^t\}, \text{ mit}$$
$$FVC_i^t = \{f_{ij}^t \mid j = 1, ..., {n_{C(i)}}^t\}. \quad (359)$$

Mit der Klassifikation FVK^t wird die ursprüngliche Dokument-Merkmals-Matrix DMM^t reformuliert, indem die vorher n^t-dimensionalen Dokumentvektoren x_i^t zu ${n_C}^t$-dimensionalen Dokumentvektoren $x_{C,i}^t$ durch eine Funktion rec(DMM^t) rekodiert werden. D.h. die (m^t x n^t)-Matrix DMM^t wird zu einer (m^t x ${n_C}^t$)-Matrix, die mit ${DMM_{FC}}^t$ bezeichnet werden soll. Aus dem ursprünglich n^t-dimensionalen Dokumentvektorraum DVR wird somit der ${n_C}^t$-dimensionale neue Dokumentvektorraum DVR_C unter Beibehaltung der Metrik $d_{DVR}(., .)$. Die Merkmalsvektoren-Clusterung lässt sich durch die Clusterfunktion cl(.) und die Rekodierungsfunktion rec(.) beschreiben als:

$$cl(FV^t) = FVK^t \rightarrow rec(DMM^t) = {DMM_{FC}}^t. \quad (360)$$

Die Clusterung der Merkmale führt somit zu einer Einführung eines neuen Dokumentvektorraumes DVR_C mit einer neuen Strukturierung der darin enthaltenen Dokumentvektoren. Alle Queries werden entsprechend den geclusterten Merkmalen indexiert, sodass sie Queryvektoren in DVR_C besitzen, gefolgt von einer Einpunkt- oder einer Mehrpunkt-Retrievalstrategie, wie sie bereits im ungeclusterten Fall beschrieben wurden, da die Dokumentvektoren durch die Clusteroperationen im Merkmalsvektorraum FVR nicht geclustert werden.

Durch eine Polyrepräsentation der Merkmalsvektoren oder durch eine Polyrepräsentation der Clusterfunktion cl(.) wird die Merkmalsvektoren-Polyklassifikation $BDVK^t$ erzeugt, als der (δ+1)'elementigen Menge von Merkmalsvektoren-Klassifikationen, wobei wiederum Bootstrapverfahren unterstellt werden:

$$BFVK^t = \{FVK^t, FDVK_k{}^t \mid k = 1, ..., \delta\}. \quad (361)$$

Die Cluster- und Rekodierungsprozesse lassen sich umschreiben als:

$$cl(FV^t) = FVK^t \rightarrow rec(DMM^t) = DMM_{FC}{}^t,$$
$$cl(BFV_k{}^t) = BFVK_k{}^t \rightarrow rec(DMM^t) = DMM_{FC,k}{}^t. \quad (362)$$

Eine Polyrepräsentation Dokumentvektoren $BDV^t = \{DV^t, BDV_k{}^t \mid k = 1, ..., \delta\}$ ergibt, wie bei der Indexierung dargestellt wurde, durch die Dokument-Merkmals-Matrizen DMM^t und $DMM_k{}^t$ bereits eine Polyrepräsentation der Merkmalsvektoren, sodass für die Clusterung keine weiteren Vorverarbeitungsschritte notwendig werden.

3.7.4) Integrierte Dokumentvektoren und Merkmalsvektoren-Clusterung

In den beiden vorangegangenen Abschnitten wurde die Clusterung der Dokumentvektoren und der Merkmalsvektoren isoliert betrachtet, d.h. die Clusterung im Dokumentraum bzw. die Einführung eines neuen Dokumentvektorraumes DVR_C können als eigenständige Strategien durchgeführt werden. Demgegenüber können auch beide Strategien zusammen in einer einzigen Strategie integriert werden, wobei die folgenden Möglichkeiten existieren:

1) Unabhängige Clusterung.
2) Abhängige Clusterung.

Bei der unabhängigen Clusterung wird bei einer Monorepräsentation DMM^t eine Dokumentvektoren- und eine Merkmalsvektoren-Clusterung durchgeführt, d.h. aus DMM^t wird die Dokumentvektorenliste DV^t und die Merkmalsvektorenliste FV^t extrahiert, wonach eine jeweils unabhängige Clusterung $cl(DV^t)$ und $cl(FV^t)$ durchgeführt wird, mit der DMM^t in eine $(m_C{}^t \times n^t)$-Matrix $DMM_{DC}{}^t$ und eine $(m^t \times n_C{}^t)$-Matrix $DMM_{FC}{}^t$ umgewandelt wird:

$$DMM^t \rightarrow DV^t, FV^t \rightarrow$$
$$cl(DV^t) = DVK^t, cl(FV^t) = FVK^t \rightarrow rec(DMM^t) = DMM_{DC}{}^t, rec(DMM^t) = DMM_{FC}{}^t. \quad (363)$$

Dies bedeutet, dass nach den Clusterprozessen zwei Dokument-Merkmalsmatrizen und somit je zwei Dokument- und Merkmalsräume vorliegen, sodass spezielle Formen des Retrievals eingeführt werden müssen. Beispielsweise wird eine Query durch die ungeclusterte und die geclusterte Merkmalsmenge indexiert, sodass in den beiden Dokumenträumen ein Queryvektor erzeugt wird. Es werden zwei Einpunkt-Query-Retrievaloperationen durchgeführt, deren Ergebnisse durch eines der Verfahren von Mehrpunkt-Query-Retrievaloperationen aggregiert wird.

Komplexer ist die Situation bei einer abhängigen Clusterung, da die Clusterung von Merkmalsvektoren bzw. Dokumentvektoren Auswirkungen auf die jeweilig andere Clusterung besitzt, wenn die Klassifikation einer Vektorenliste zur Reformulierung der anderen Vektorenliste eingesetzt wird, die ihrerseits geclustert werden soll. Es stellt sich die Frage, welche von beiden Vektorenlisten zuerst geclustert werden soll, da die Festlegung der Reihenfolge das Endergebnis wesentlich beeinflusst.

Es ergeben sich bei der sequentiellen Clusterung beider Listen folgende Szenarien, wobei vereinfachend von einer Monorepräsentation ausgegangen werden soll:

1) Beginne mit der Dokumentvektoren-Klassifikation.
Im ersten Schritt werden die m^t Dokumentvektoren DV_i^t zu m_C^t Dokumentvektoren-Clustern DVC_i^t zusammengefasst, was durch die Clusterfunktion $cl(DV^t)$ beschrieben wird. Mit der Klassifikation DVK^t wird die ursprüngliche Dokument-Merkmals-Matrix DMM^t reformuliert, indem die vorher m^t-dimensionalen Merkmalsvektoren f_i^t zu m_C^t-dimensionalen Merkmalsvektoren $f_{C,i}^t$ rekodiert werden. D.h. die (m^t x n^t)-Matrix DMM^t wird zu einer (m_C^t x n^t)-Matrix DMM_{DC}^t. Durch die Rekodierung der Merkmalsvektoren entsteht aus dem ursprünglich m^t-dimensionalen Merkmalsvektorraum FVR^t der neue m_C^t-dimensionale Merkmalsraum FVR_C^t, in dem die Clusterung der Merkmalsvektoren durchgeführt werden soll. Die n^t rekodierten Merkmalsvektoren $f_{C,i}^t$ werden zu n_C^t Merkmalsvektoren-Clustern FVC_i^t zusammengefasst, wobei sich die Klassifikation FVK_C^t ergibt. Damit lässt sich die (m_C^t x n^t)-Matrix DMM_{DC}^t zu einer (m_C^t x n_C^t)-Matrix DMM_{FDC}^t rekodieren, und der ursprünglich n^t-dimensionale Dokumentvektorraum DVR^t wird zu einem n_C^t-dimensionalen neuen Dokumentvektorraum DVR_C^t. Diese Reihenfolge soll beschrieben werden als:

$$cl(DV^t) = DVK^t \rightarrow rec(DMM^t) = DMM_{DC}^t \rightarrow$$
$$cl(FV_C^t) = FVK_C^t \rightarrow rec(DMM_{DC}^t) = DMM_{FDC}^t. \quad (364)$$

2) Beginne mit der Merkmalsvektoren-Klassifikation.
Im ersten Schritt werden die n^t Merkmalsvektoren FV_i^t zu n_C^t Merkmalsvektoren-Clustern FVC_i^t zusammengefasst, was durch eine Clusterfunktion $cl(FV^t)$ beschrieben werden soll. Mit der Klassifikation FVK^t wird die ursprüngliche Dokument-Merkmals-Matrix DMM^t reformuliert, indem die vorher n^t-dimensionalen Dokumentvektoren x_i^t zu n_C^t-dimensionalen Dokumentvektoren $x_{C,i}^t$ rekodiert werden. D.h. die (m^t x n^t)-Matrix DMM^t wird zu einer (m^t x n_C^t)-Matrix, die mit DMM_{FC}^t bezeichnet werden soll. Durch die Rekodierung der Dokumentvektoren entsteht aus dem ursprünglich n^t-dimensionalen Dokumentvektorraum DVR^t der n_C^t-dimensionale neue Dokumentvektorraum DVR_C^t, in dem die Clusterung der Dokumentevektoren durchgeführt werden soll. Die m^t rekodierten Dokumentvektoren $c_{C,i}^t$ werden zu m_C^t Dokumentvektoren-Clustern DVC_i^t zusammengefasst, wobei sich die Klassifikation DVK_C^t ergibt. Damit lässt sich die (m^t x n_C^t)-Matrix DMM_{FC}^t zu einer (m_C^t x n_C^t)-Matrix DMM_{DFC}^t rekodieren, und der ursprünglich m^t-dimensionale Merkmalsvektorraum FVR^t wird zu einem m_C^t-dimensionalen neuen Merkmalsvektorraum FVR_C^t. Diese Reihenfolge soll beschrieben werden als:

$$cl(FV^t) = FVK^t \rightarrow rec(DMM^t) = DMM_{FC}^t \rightarrow$$
$$cl(DV_C^t) = DVK_C^t \rightarrow rec(DMM_{FC}^t) = DMM_{DFC}^t. \quad (365)$$

Die beiden Matrizen DMM_{FDC}^t und DMM_{DFC}^t müssen weder die gleiche Anzahl von Zeilen oder Spalten besitzen, auch wenn in beiden Fällen m_C^t und n_C^t verwendet wird, da durch die Reihenfolge der Clusterung die Anzahl der unüberwacht entstehenden Cluster unterschiedlich sein kann.

3.7.5) Cluster-Retrieval-Strategien

Ziel aller Cluster-Retrieval-Strategien ist es, die Anzahl der Vergleichsoperationen zwischen dem oder den Queryvektoren und anderen Objektvektoren wie den Dokumentvektoren zu verkleinern, um einen Effizienzvorteil gegenüber einer enumerativen Suche zu erzielen. Die Nutzung von Merkmals-Clustern ist in diesem Zusammenhang nur indirekt, da die Query mit Hilfe der Merkmals-Klassifikation indexiert wird, wohingegen hier keine weiteren Operationen im Merkmalsraum betrachtet werden. Eine wesentliche Operationsklasse, die Expansion einer Query oder die Modifikation eines Queryvektors, soll im Zusammenhang mit Clusteroperationen durch SOMs im nachfolgenden Kapitel beschrieben werden.

Ausgangspunkt ist eine Query Q_i^t, die in einen Queryvektor q_i^t überführt wurde, und die in die Dokumentvektorenverteilung DV^t bzw. die Dokumentvektoren-Klassifikation DVK^t eingefügt wird. Ziel der weiteren Betrachtungen ist es, mit q_i^t bzw. unter Hinzunahme der Query-Umgebung $U(q_i^t \mid \varepsilon)$ zunächst Dokumentvektoren-Cluster zu spezifizieren. Die Cluster werden im weiteren ganz in die Retrieval-Dokumentmenge übernommen, oder es wird ein weiterer Selektionsprozess angeschlossen, der aus diesen Clustern Dokumentvektoren mit bestimmten geometrischen oder topologischen Eigenschaften auswählt, welche die Retrievalmenge bilden. Die Selektion der Cluster soll als Primärselektion und die Auswahl von Dokumentvektoren in diesen Clustern soll als Sekundärselektion bezeichnet werden .

Wird zunächst nur der Queryvektor betrachtet, so sind die beiden Szenarien zu unterscheiden:
1) Der Queryvektor liegt außerhalb aller Clusterregionen (siehe Abb. 53)):

$$q_i^t \notin U(\bar{s}_{C(j)}^t \mid \varepsilon_{C(j)}^t),\ \forall DVC_j^t \in DVK^t. \tag{366}$$

2) Der Queryvektor ist Element einer bzw. mindestens einer Clusterregion (siehe Abb. 54)):

$$\exists_{\geq 1} DVC_j^t \in DVK^t: q_i^t \in U(\bar{s}_{C(j)}^t \mid \varepsilon_{C(j)}^t). \tag{367}$$

Die Prüfung, ob q_i^t in einer Clusterregion $U(\bar{x}_{C(j)}^t \mid \varepsilon_{C(j)}^t)$ liegt, kann bei hyperkugelförmigen Regionen durch eine einzige Distanzberechnung $d_{DVR}(\bar{x}_{C(j)}^t, q_i^t)$ oder $d_{DVR}(z_{C(j)}^t, q_i^t)$ und den Vergleich mit dem Umgebungsparameter $\varepsilon_{C(j)}^t \in R^+$ durchgeführt werden:

$$\begin{aligned} &d_{DVR}(\bar{x}_{C(j)}^t, q_i^t) < \varepsilon_{C(j)}^t \Rightarrow q_i^t \in U(\bar{x}_{C(j)}^t \mid \varepsilon_{C(j)}^t), \text{ oder} \\ &d_{DVR}(z_{C(j)}^t, q_i^t) < \varepsilon_{C(j)}^t \Rightarrow q_i^t \in U(z_{C(j)}^t \mid \varepsilon_{C(j)}^t). \end{aligned} \tag{368}$$

Liegt q_i^t innerhalb einer oder mehrerer Clusterregionen, so kann dies als Selektionskriterium für die entsprechenden Dokumentvektoren-Cluster betrachtet werden, d.h. die Vereinigungsmenge dieser Dokumentvektoren bildet die Retrieval-Dokumentvektorenmenge $DVM(q_i^t)$ bezüglich des Queryvektors q_i^t:

$$\begin{aligned} &DVM(q_i^t) = \bigcup_j DVC_j^t,\ q_i^t \in U(\bar{x}_{C(j)}^t \mid \varepsilon_{C(j)}^t) = \\ &\{x_{jk}^t \in DVC_j^t \mid q_i^t \in U(\bar{x}_{C(j)}^t \mid \varepsilon_{C(j)}^t)\}. \end{aligned} \tag{369}$$

Abb. 53) Queryvektor außerhalb aller Clusterregionen

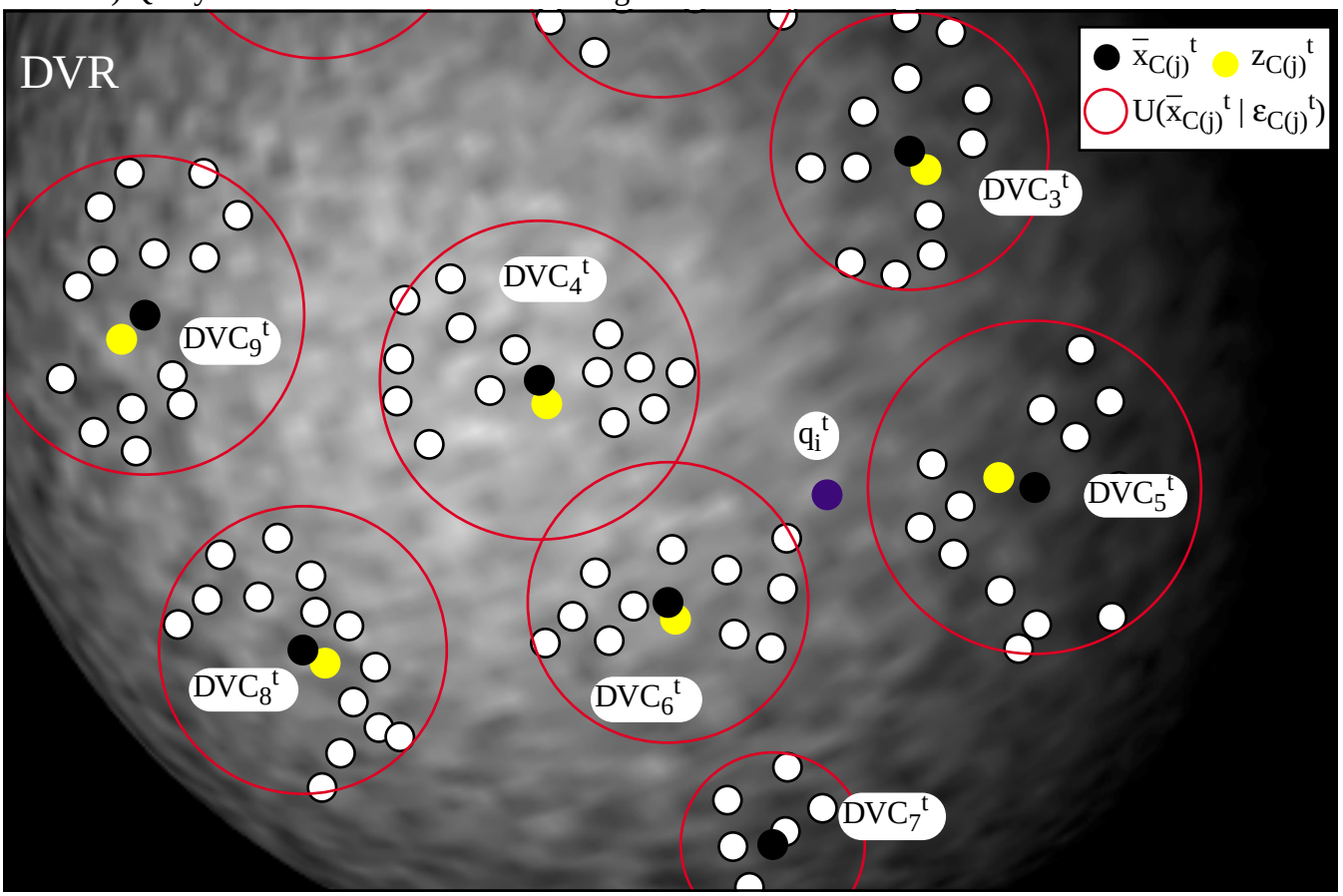
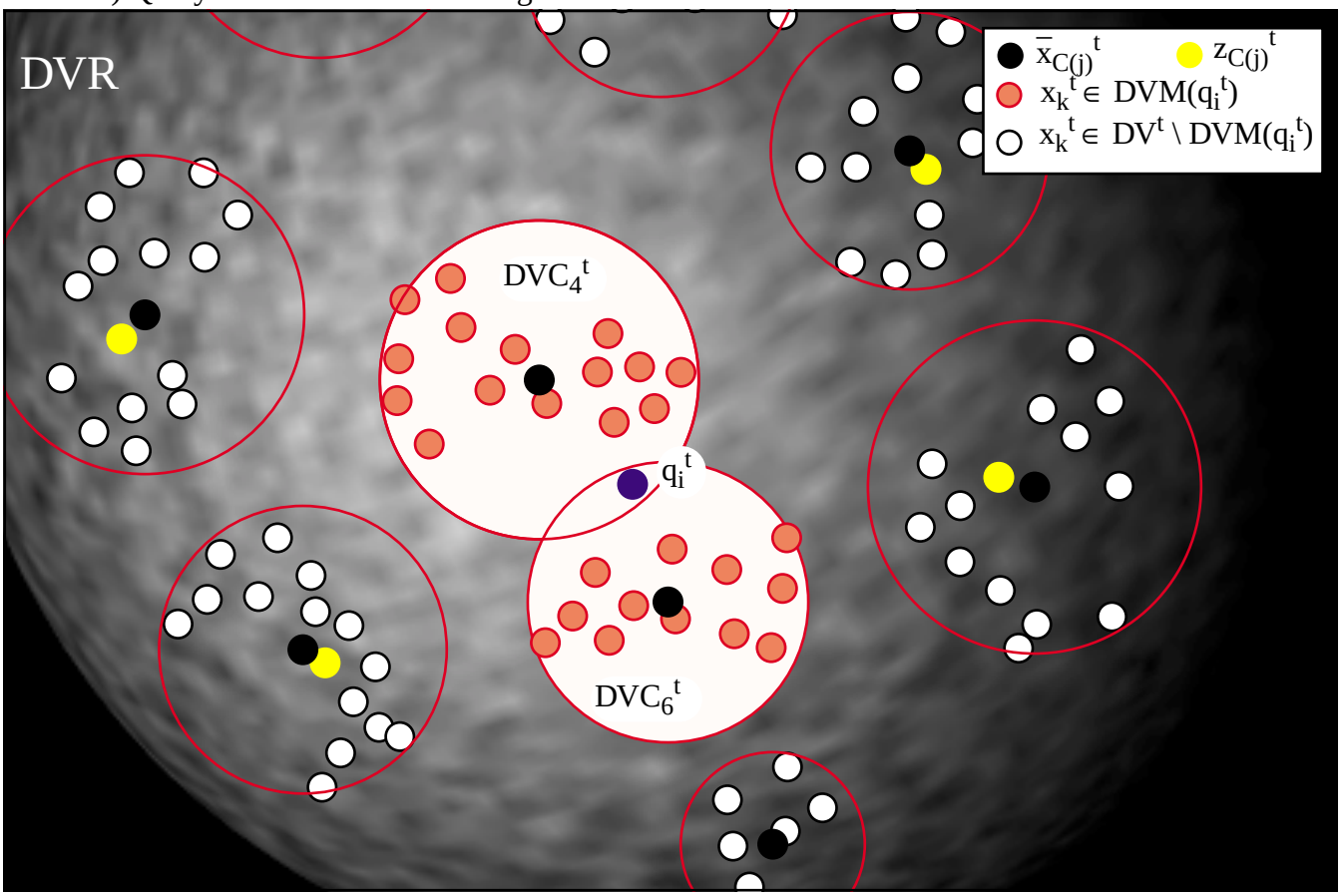


Abb. 54) Queryvektor in zwei Clusterregionen

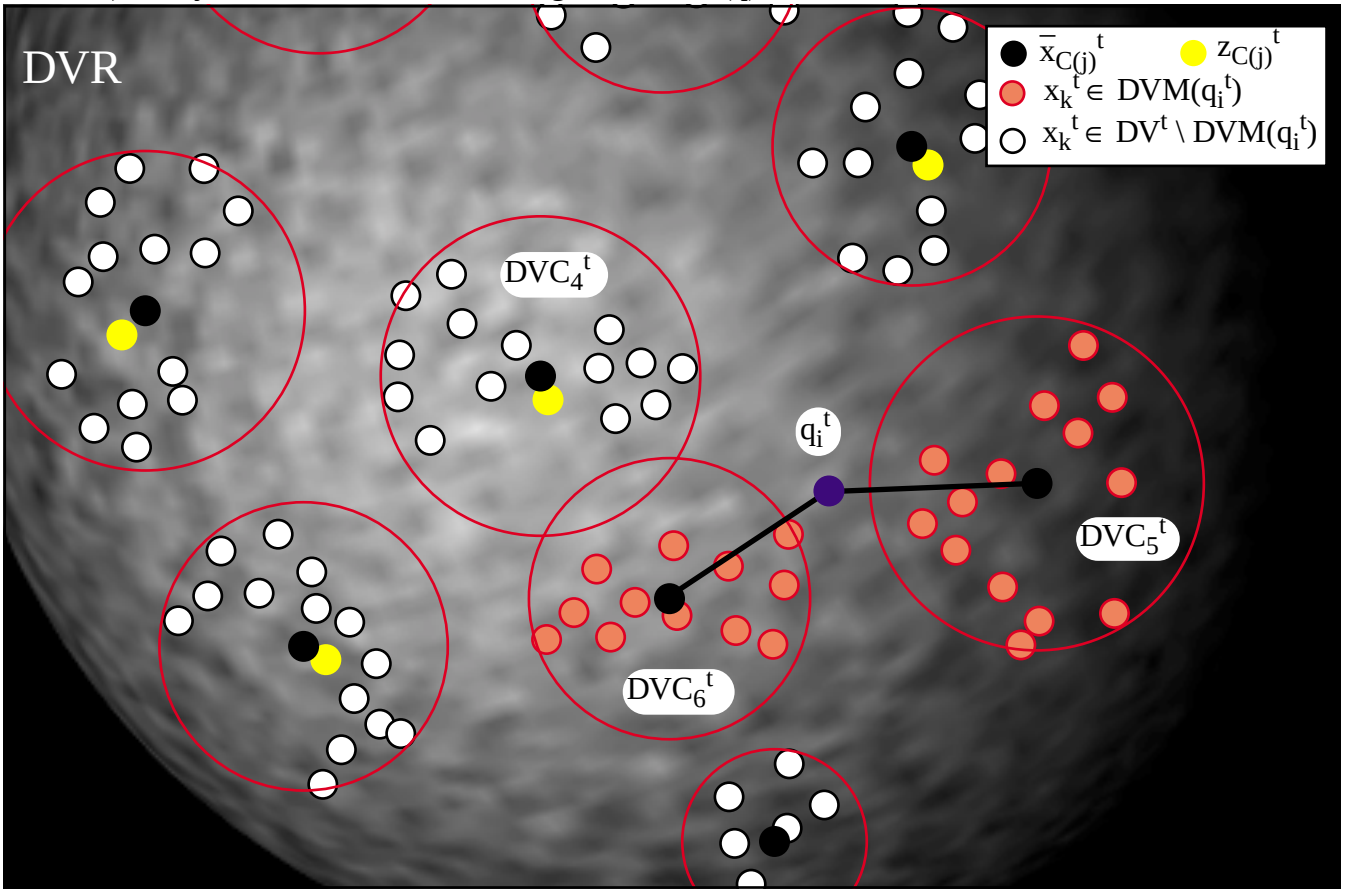


Diese Situation wird am Beispiel von zwei Dokumentvektoren-Clustern in Abb. 54) dargestellt. Die Dokumentvektorenmenge kann durch ein Sortierkriterium in eine Retrieval-Dokumentvektorenliste transformiert werden, indem die Dokumentvektoren entsprechend steigendem Abstand zu q_i^t sortiert werden:

$$DV(q_i^t) = (x_k^t \in DVM(q_i^t) \mid d_{DVR}(q_i^t, x_k^t) < d_{DVR}(q_i^t, x_{k+1}^t)). \quad (370)$$

Liegt q_i^t außerhalb aller Clusterregionen, so muss eine Retrieval-Strategie mit einem anderen Kriterium zur Clusterauswahl verwendet werden. Analog wie bei einer Retrieval-Strategie, bei der kein Dokumentvektor innerhalb einer Query-Umgebung liegt, können die m(q) am nächsten liegenden Dokument-Cluster für diesen Zweck verwendet werden. Dieses Kriterium kann durch die Distanz $d_{DVR}(\bar{x}_{C(j)}^t, q_i^t)$ oder $d_{DVR}(z_{C(j)}^t, q_i^t)$ operationalisiert werden, die bereits vorliegt, da die Distanzwerte für die Entscheidung berechnet werden, ob q_i^t in einer der Clusterregionen liegt. Werden beispielsweise die beiden nächsten Dokumentvektoren-Cluster ausgewählt, d.h. m(q) = 2, und werden alle Dokumentvektoren in die Retrievalmenge übernommen, so ergibt sich eine Beispielsituation, die in Abb. 55) dargestellt ist.

Abb. 55) Queryvektor außerhalb Clusterregionen mit m(q) = 2 Cluster



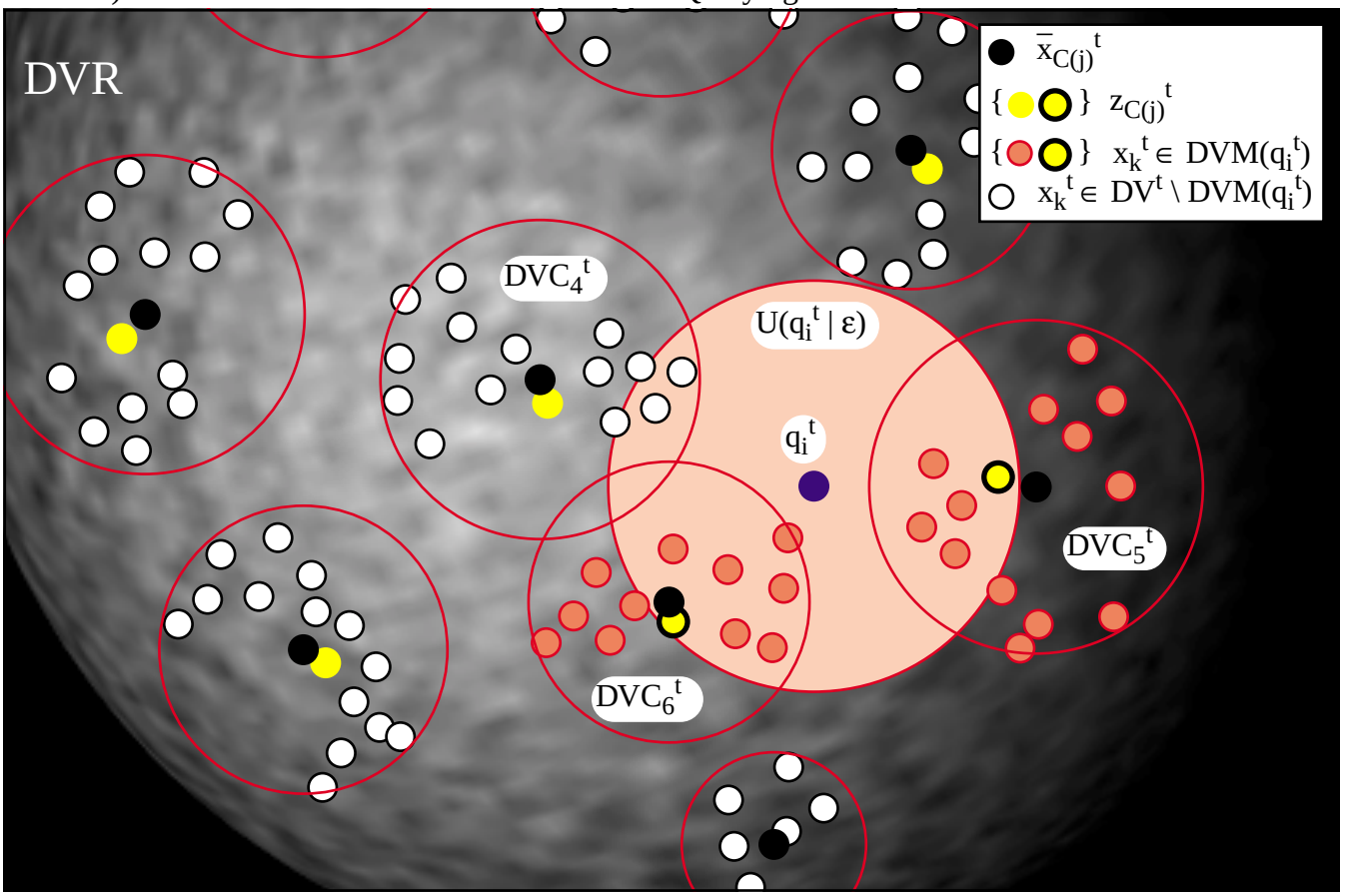
Als Alternative zu der Festlegung der nächsten m(q) Cluster kann die Query-Umgebung $U(q_i^t \mid \varepsilon)$ herangezogen werden, um Cluster für den Fall zu selektieren, dass q_i^t außerhalb aller Clusterregionen liegt. Beispielsweise kann geprüft werden, ob ein Zentroidvektor oder ein Medianvektor in der Query-Umgebung liegt:

$$\exists_{\geq 1} \bar{x}_{C(j)}^{\,t}: \bar{x}_{C(j)}^{\,t} \in U(q_i^{\,t} \mid \varepsilon), \text{ oder}$$
$$\exists_{\geq 1} z_{C(j)}^{\,t}: z_{C(j)}^{\,t} \in U(q_i^{\,t} \mid \varepsilon). \quad (371)$$

In Abb. 56) wird der Fall dargestellt, dass in der Queryregion zwei Medianvektoren und ein Zentroidvektor liegen, und dass die Selektion durch die Medianvektoren erfolgt, sodass die zwei betroffenen Dokumentvektoren-Cluster mit allen ihren Elementen ausgewählt werden.

Werden alle Elemente der Dokumentvektoren-Cluster ausgewählt, so kann der Fall eintreten, dass Dokumentvektoren ausgewählt werden, die nicht in der Query-Umgebung liegen. Andererseits können Dokumentvektoren nicht ausgewählt werden, die zwar in der Query-Umgebung liegen, die aber in einem Cluster liegen, der durch die Primärselektion nicht ausgewählt wurde.

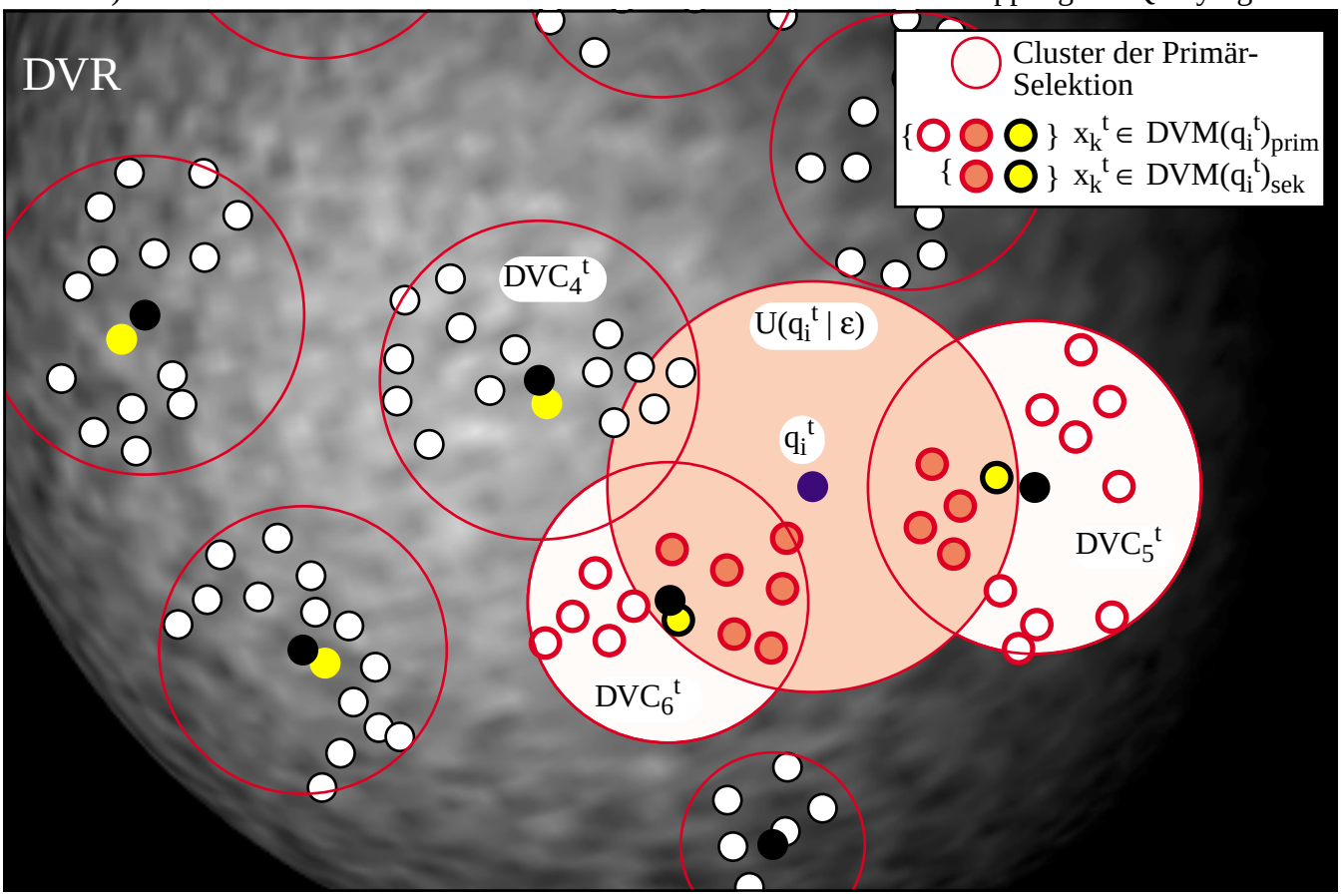
Abb. 56) Zentroid- und Medianvektor innerhalb der Queryregion



Um den ersten Problemfall auszuschließen, kann im Rahmen der Sekundärselektion für alle Elemente aus der Vereinigungsmenge der Cluster, die im Rahmen der Primärselektion ausgewählt wurden, geprüft werden, ob sie Element der Query-Umgebung sind. Zunächst kann eine primäre Retrievalmenge $DVM(q_i^{\,t})_{prim}$ erzeugt werden, als Vereinigungsmenge aller Cluster, deren Repräsentant $\bar{x}_{C(j)}^{\,t}$ innerhalb der Query-Umgebung liegt (siehe Abb. 57)):

$$DVM(q_i^{\,t})_{prim} = \bigcup_j DVC_j^{\,t}, \bar{x}_{C(j)}^{\,t} \in U(q_i^{\,t} \mid \varepsilon). \quad (372)$$

Abb. 57) Primärselektion durch Median und Sekundärselektion durch Überlappung mit Queryregion



Im zweiten Schritt werden die Dokumentvektoren aus $DVM(q_i^t)_{prim}$ ausgewählt, die selbst in der Query-Umgebung liegen, wobei die sich ergebende Menge als sekundäre Retrievalmenge $DVM(q_i^t)_{sek}$ bezeichnet werden soll, und deren korrespondierende Dokumente dem Agenten als Retrievalergebnis präsentiert werden:

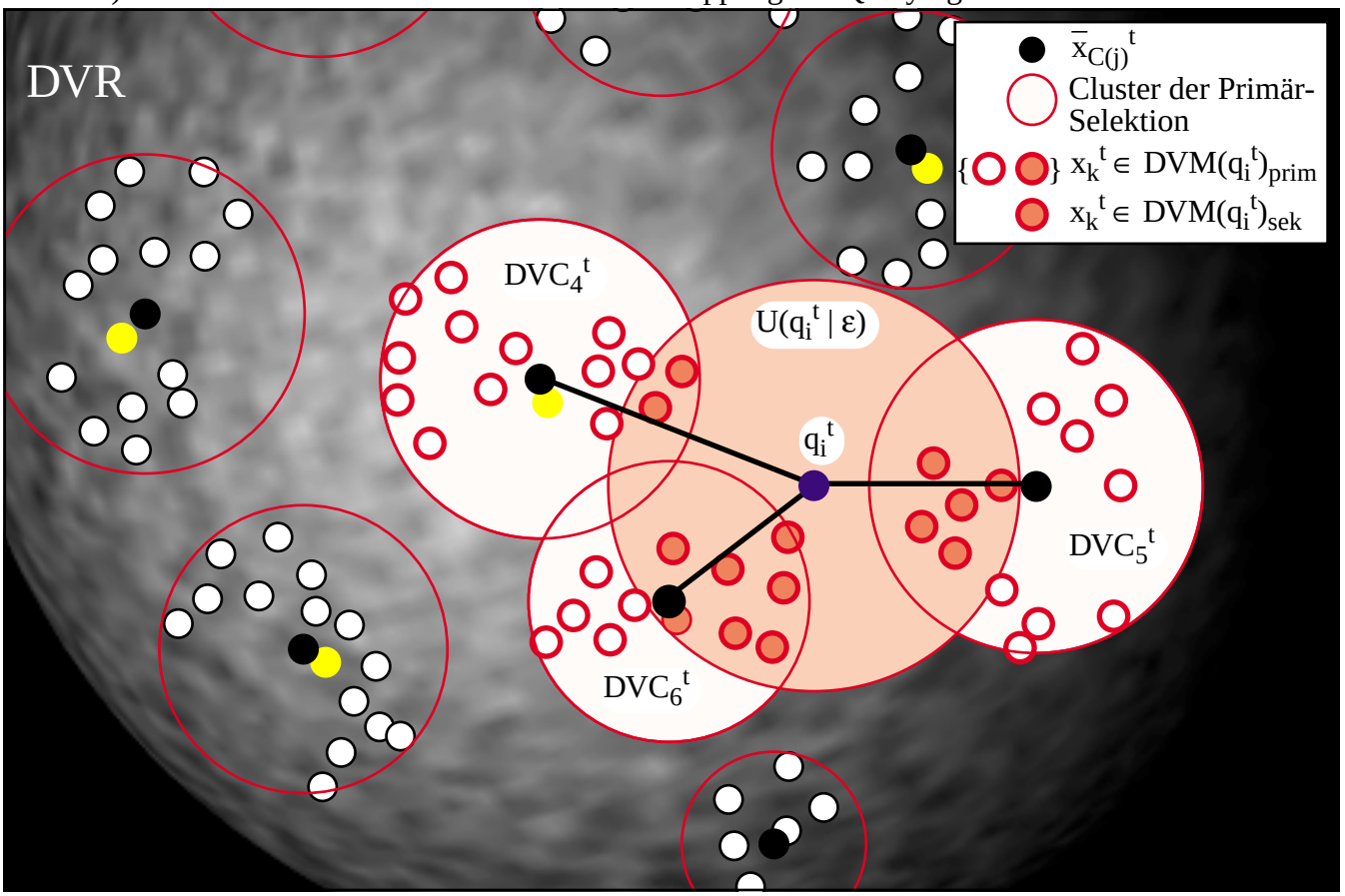
$$DVM(q_i^t)_{sek} = \{x_{jk}^t \in DVM(q_i^t)_{prim} \mid x_{jk}^t \in U(q_i^t \mid \varepsilon)\}. \quad (373)$$

Diese zweistufige Vorgehensweise besitzt einen Effizienzvorteil zu der einstufigen, enumerativen Suche nach Dokumentvektoren in der Query-Umgebung, da weniger als m^t Distanzen $d_{DVR}(q_i^t, x_k^t)$ berechnet werden müssen. Zunächst müssen die m_C^t Distanzen $d_{DVR}(q_i^t, \bar{x}_{C(j)}^t)$ zu den Repräsentanten der Dokumentvektoren-Clustern berechnet werden, gefolgt von den $\sum_j m_{C(j)}^t$ Distanzen $d_{DVR}(q_i^t, x_{jk}^t)$ zu den Elementen x_{jk}^t der selektierten Clustern DVC_j^t.

Das zweite angesprochene Problem, d.h. die Nicht-Auswahl von Dokumentvektoren, die in $U(q_i^t \mid \varepsilon)$ liegen, kann nicht durch eine Änderung der Sekundär- sondern nur durch eine Änderung der Primärselektion angegangen werden. Ein Ansatz ist die Ermittlung aller Cluster, deren Clusterregion $U(\bar{x}_{C(j)}^t \mid \varepsilon_{C(j)}^t)$ eine Überlappung mit der Query-Region besitzen. Die Prüfung der Existenz einer Überlappungsregion $\ddot{U}(U(q_i^t \mid \varepsilon), U(\bar{x}_{C(j)}^t \mid \varepsilon_{C(j)}^t))$ wird durch die Distanz $d_{DVR}(q_i^t, \bar{x}_{C(j)}^t)$ und die beiden Umgebungsparameter $\varepsilon_{C(j)}^t$ und ε durchgeführt, wobei eine Überlappung vorliegt, wenn die Distanz kleiner der Summe der beiden Parameter ist (siehe Abb. 58)):

$$d_{DVR}(q_i^t, \bar{x}_{C(j)}^t) < \varepsilon_{C(j)}^t + \varepsilon \Rightarrow \ddot{U}(U(q_i^t \mid \varepsilon), U(\bar{x}_{C(j)}^t \mid \varepsilon_{C(j)}^t)) \neq \varnothing. \quad (374)$$

Abb. 58) Primär- und Sekundärselektion durch Überlappung mit Queryregion



Wird wiederum eine primäre und eine sekundäre Retrievalmenge erzeugt, so wird $DVM(q_i^t)_{prim}$ als Vereinigung aller Cluster gebildet, die mit $U(q_i^t \mid \varepsilon)$ eine Überlappung besitzen:

$$DVM(q_i^t)_{prim} = \bigcup_j DVC_j^t, \ddot{U}(U(q_i^t \mid \varepsilon), U(\bar{x}_{C(j)}^t \mid \varepsilon_{C(j)}^t)) \neq \varnothing. \quad (375)$$

Die sekundäre Retrievalmenge besitzt die gleiche Definition wie bei der oben dargestellten Version, d.h. aus der primären Retrievalmenge werden die Elemente ausgewählt, die in der Query-Umgebung liegen:

$$DVM(q_i^t)_{sek} = \{x_{jk}^t \in DVM(q_i^t)_{prim} \mid x_{jk}^t \in U(q_i^t \mid \varepsilon)\}. \quad (376)$$

3.8) Indexierung und Retrieval mit GNG-SOM-Modellen am Beispiel unabhängiger Merkmals- und Dokument-Graphen

Im weiteren sollen Verfahrensweisen vorgestellt werden, bei denen Indexierung, Clusterung und Retrieval unter Verwendung von SC-GNG-Graphen durchgeführt wird. Im Gegensatz zu Bachelier (1995[14]) sollen die Verfahren nicht auf den Pseudo-Stetigen-SOM (PS-SOM oder PSS-SOM, siehe auch Bachelier (1999a[19])) basieren, da eine Dimensionsreduzierung zum Zweck der Visualisierung im Rahmen dieser Arbeit nicht im Vordergrund stehen soll. Gemeinsam mit den PS-SOM ist jedoch die Tatsache, dass die

Stimuli zum Aufbau eines GNG-Graphen zum einen Merkmalsvektoren f_i^t und zum anderen Dokumentvektoren x_i^t sind, wobei folgende Szenarien möglich sind:
1) Merkmale und Dokumente werden zusammen in einem Graphen dargestellt.
2) Es wird ein eigener Merkmalsgraph und ein eigener Dokumentgraph verwendet.
3) Es werden mehrere Graphen der Typen Merkmals- und Dokumentgraph verwendet.

Zu beachten ist im ersten Fall, dass x_i^t und f_i^t entsprechend der Dokument-Merkmals-Matrix DMM^t mit n^t und m^t unterschiedliche Dimensionen besitzen, sodass eine Darstellung in einem gemeinsamen metrischen Vektorraummodell nicht ohne weiteres möglich ist. Die Darstellung unterschiedlicher Informationsobjekte in einer integrierten Darstellung findet sich in STEINADLER (Panyr (1986a[247])) und in KOAN (Führing et al. (1994[134])).

Wie in Bachelier (1995[14]) sollen die nachfolgenden Darstellungen ausschlisslich den zweiten Fall behandeln, d.h. dass in einem Merkmalsvektorraum $FVR \subseteq R^{m(t)}$ genau ein Merkmalsgraph G_F^t mit dem zugehörigen GNG-SOM-Modell N_F^t gebildet wird, und dass in einem Dokumentvektorraum $DVR \subseteq R^{n(t)}$ genau ein Dokumentgraph G_D^t mit dem zugehörigen GNG-SOM-Modell N_D^t gebildet wird. Im Rahmen der Polyrepräsentation von Dokumenten, Merkmalen und Queries kann der dritte Fall sinnvoll angewendet werden, worauf jedoch hier hicht fokussiert werden soll.

Das wesentliche Designkriterium des Aufbaus von Merkmalsvektor- und Dokumentvektor-GNG-SOM-Modellen ist die Entscheidung bezüglich der Abhängigkeit der beiden Modelle, wobei hier drei Alternativen formulierbar sind:
1) Unabhängige Merkmals- und Dokument-Graphen
2) Sequentiell abhängige Merkmals- und Dokument-Graphen
3) Rekursiv abhängige Merkmals- und Dokument-Graphen

Bei der unabhängigen Erzeugung wird aus der Dokument-Merkmalsmatrix die Menge Dokumentvektoren und die Menge der Merkmalsvektoren extrahiert, aus denen unabhängig eine Dokumentvektor-GNG-SOM und eine Merkmalsvektor-GNG-SOM aufgebaut wird. Bei der sequentiell abhängigen Formulierung wird zunächst eine GNG-SOM aufgebaut, mit der die Vektoren des anderen Typs reformuliert werden, gefolgt vom Aufbau der zweiten GNG-SOM auf der Basis der reformulierten Vektoren. Das direkt zugängliche Beispiel ist der Aufbau einer Merkmalsvektor-GNG-SOM mit weniger Neurone als Stimuli, was zu einer Clusterung der Merkmale führt. Die Dokumentvektoren werden mit der geringeren Anzahl von Merkmalen reindexiert, gefolgt von dem Aufbau einer Dokumentvektor-GNG-SOM in einem metrischen Raum, dessen Dimensionsanzahl gleich der Anzahl der Merkmalsvektoren-Cluster ist. Die rekursive Formulierung führt die Abhängigkeit noch weiter, indem mehrere Zwischenschritte eingeführt werden, d.h. zunächst wird eine GNG-SOM aus der ursprünglichen Matrix erzeugt, gefolgt von der Reformulierung des zweiten Vektortyps und dem Aufbau der zweiten GNG-SOM auf der Basis der reformulierten Vektoren. Die zweite GNG-SOM wird dann verwendet, um den ersten Vektortyp zu reformulieren, gefolgt von einer Neuerzeugung einer GNG-SOM auf der Basis des dann vorliegenden Vektorraumes. Diese rekursive Formulierung kann theoretisch über beliebige Iterationen fortgeführt werden.

Im weiteren soll sich auf die einfachste Formulierung, d.h. auf den unabhängigen Aufbau beschränkt werden, da die Retrieval-Operationen in den beiden GNG-SOMs unabhängig davon ablaufen, wie diese erzeugt wurden.

Ein anderes Designkriterium ist die Entscheidung, ob mit dem Aufbau der GNG-SOM-Modelle eine Clusterung der jeweiligen Objekte erfolgen soll, oder ob durch den Wachstumsprozess die Graphen so groß werden sollen, dass jedem Neuron genau ein Objekt zugeordnet ist, d.h. dass der Abbruch des Graphenaufbaus dann erfolgt, wenn in der lokalen Stimulusmenge M_i^t jedes Neurons n_i^t genau ein Element enthalten ist, sodass mit μ_N genauso viele Neurone wie Stimuli mit μ_M vorliegen. Wird der Gewichtsvektor $w(x)_i^t$ eines Neurons n_i^t nach dem Verfahrensabbruch durch den Inputvektor des Stimulus ersetzt, das als einziges Element in M_i^t liegt, so handelt es sich bei dem Verfahren letztendlich um die Bildung einer Delaunay-Triangulation auf der Basis der Stimuli, d.h. alle Stimuli-Inputvektoren sind Knoten des Delaunay-Graphen. Der Quantifizierungsfehler des Graphen wird auf diese Weise mit einem Wert von Null minimal. Das Abbruchkriterium des Wachstumsprozesses bei einer Clusterung kann demgegenüber nicht dadurch erreicht werden, indem $\mu_N = \mu_M$ gesetzt wird. Es kann eine maximale Anzahl von Neuronen festgelegt werden, bei deren Erreichung der Wachstumsprozess beendet wird, gefolgt von weiteren Adaptionsprozesses, mit denen versucht wird, ein Bewertungsmaß wie den Quantifizierungsfehler zu minimieren. Umgekehrt kann eine Fehlerschranke vorgegeben werden, wobei durch Wachstums- und Adaptionsprozesse versucht wird, die Schranke zu unterschreiten, woraus der Abbruch folgt. Im weiteren soll von einer Clusterung der Vektoren der jeweiligen Informationsobjekte ausgegangen werden.

Eine notwendige Eigenschaft der Graphen G_F^t und G_D^t soll die Möglichkeit der Aktivitätsausbreitung über die Verbindungsstruktur der Graphen sein (siehe Abschnitt 2.1.8)), da mit den Graphen Prozesse wie Query-Erweiterung bzw. -Modifikation durchgeführt werden soll (siehe Abschnitt 3.8.5)). Informationsobjekte erhalten dabei die Initialisierung der Aktivierung, die an das am nächsten liegende Neuron weitergeleitet wird, gefolgt von einer Aktivitätsausbreitung über die Verbindungsstruktur des GNG-Graphen. Nach t_{max} Iterationsschritten wird die Aktivitätsausbreitung beendet, gefolgt von der Weiterleitung der Aktivierung eines Neurons an die Informationsobjekte, deren Objektvektor in der Voronoi-Region des Neurons liegt.

3.8.1) Aufbau unabhängiger Graphen

Die Grundidee der unabhängigen GNG-Graphen besteht darin, dass der Merkmals- und der Dokument-GNG-Graph aus einer Quelle erzeugt werden, ohne dass zur Erzeugung des einen Graphen Daten oder Zustände des anderen Graphen benötigt werden. Bei der gemeinsamen Quelle handelt es sich um die Dokument-Merkmals-Matrix DMM^t, aus der die Liste der Merkmalsvektoren $FV^t = (f_i^t \in R^{m(t)} \mid i = 1, ..., n^t)$ und die Liste der Dokumentvektoren $DV^t = (x_i^t \in R^{n(t)} \mid i = 1, ..., m^t)$ betrachtet werden. Es soll jeweils ein SC-GNG-Modell unterstellt werden, d.h. Stimuli sollen explizit in lokale Stimulusmengen geclustert werden, sodass die Definition von Stimuli auf der Basis der Stimulusvektoren notwendig wird.

Der Vorteil der unabhängigen Vorgehensweise besteht darin, dass beide Abbildungen prinzipiell parallel ablaufen können, d.h. es existiert ein Effizienzvorteil. Ein weiterer Vorteil besteht in dem einfachen Auf-

bau des Verfahrens, da keine Rekodierungen von Zuständen mit Hilfe von Aktivitätsausbreitungen in einem Vektorraum durchgeführt werden müssen, um einen Input für den anderen Vektorraum zu erhalten.

Der Nachteil besteht darin, dass keine semantischen Beziehungen verwendet werden können, die über die gemeinsame Quelle hinausgehen. Bei den abhängigen Verfahren wird ein GNG-Graph mit Hilfe des anderen aufgebaut, d.h. es werden zusätzlich die semantischen Beziehungen verwendet, die sich durch den Selbstorganisationsprozess, d.h. das unüberwachte Lernen, in einem GNG-Graphen bilden, um den Abbildungsprozess des anderen Graphen zu beeinflussen. Die Beziehungen, die sich durch die Eigenschaft der Selbstorganisation ergeben, sind nur zu einem Teil durch die Eigenschaften der Quelle bestimmt, zum anderen Teil von den Eigenschaften des Abbildungsalgorithmus beeinflusst. Daher geht die Verwendung der Beziehungen eines GNG-Graphen für die Abbildung eines anderen GNG-Graphen über die Beziehungen einer Quelle hinaus.

Ein Merkmals-Stimulus $m_{F,i}^t$ soll aus der Bezeichnung des Merkmals F_i^t und dem Merkmalsvektor f_i^t bestehen, die in einer n^t-elementigen Stimulusliste M_F^t gesammelt sind, die zu der Merkmalsliste $F^t = (F_i^t \mid i = 1, ..., n^t)$ korrespondiert:

$$M_F^t = (m_{F,i}^t = (F_i^t, f_i^t) \mid i = 1, ..., n^t). \tag{377}$$

Ein Dokument-Stimulus $m_{D,i}^t$ soll aus der Bezeichnung des Dokumentes D_i^t und dem Dokumentvektor x_i^t bestehen, die in einer m^t-elementigen Stimulusliste M_D^t gesammelt sind, die zu der Dokumentliste $D^t = (D_i^t \mid i = 1, ..., m^t)$ korrespondiert:

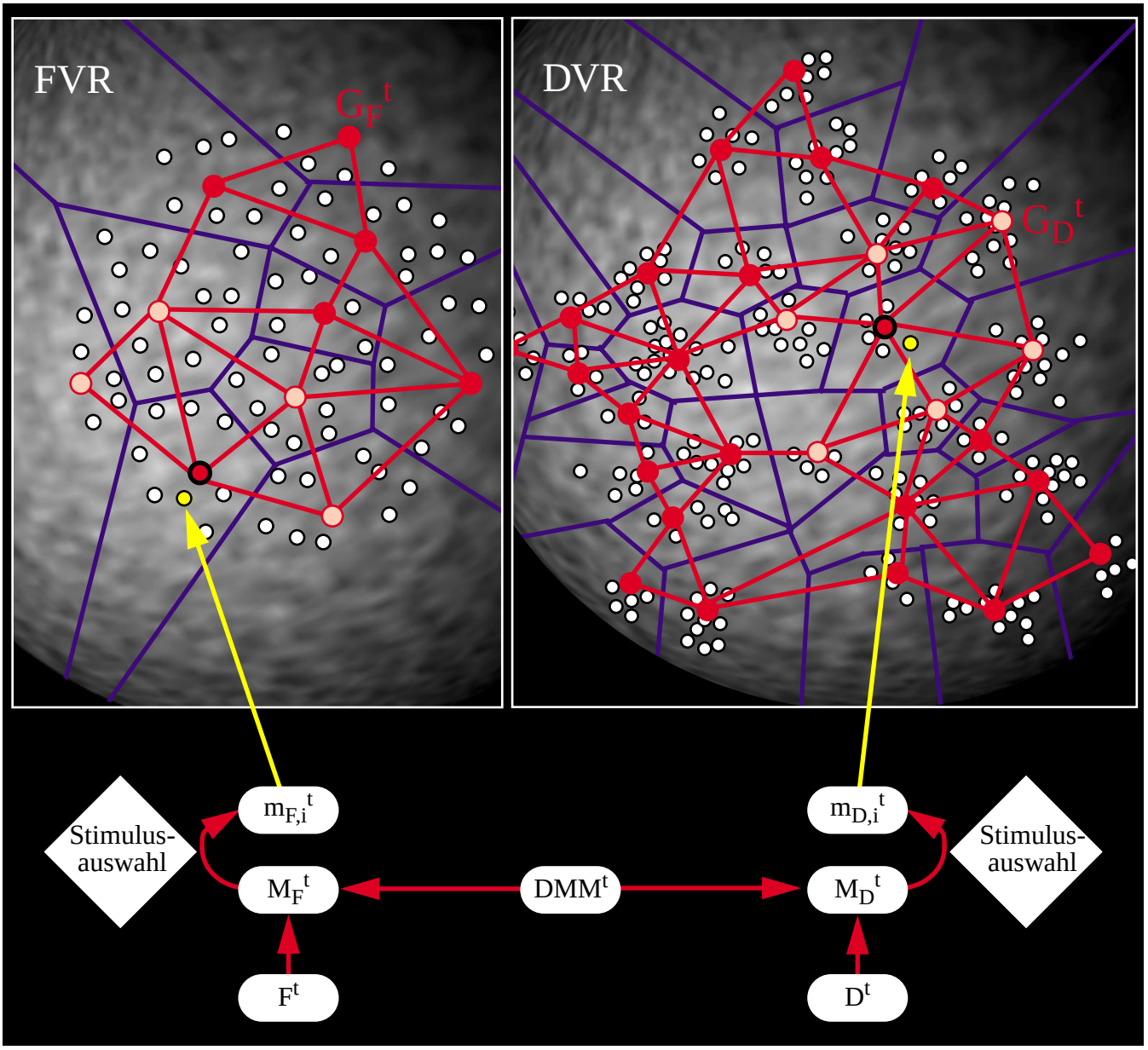
$$M_D^t = (m_{D,i}^t = (D_i^t, x_i^t) \mid i = 1, ..., m^t). \tag{378}$$

Bei den unabhängigen GNG-Graphen wird eine Neuronenstruktur N_F^t mit dem GNG-Graph G_F^t durch die Präsentation der Stimuli aus M_F^t durch Adaptions- und Wachstumsprozesse gebildet, genauso wie N_D^t mit G_D^t durch Präsentationen von Stimuli aus M_D^t gebildet wird:

$$\begin{aligned} N_F^t &= \{n_{F,i}^t = (w(x_F)_i^t, M_{F,i}^t, C_{F,i}^t) \mid i = 1, ..., \mu_{N,F}^t\}, \\ N_D^t &= \{n_{D,i}^t = (w(x_D)_i^t, M_{D,i}^t, C_{D,i}^t) \mid i = 1, ..., \mu_{N,D}^t\}. \end{aligned} \tag{379}$$

In Abb. 59) wird dieses Vorgehen noch einmal dargestellt, indem aus DMM^t unter Referenz auf die beiden Listen F^t und D^t die Stimuluslisten M_F^t und M_D^t erzeugt werden. Der Aufbau der Neuronenmengen N_F^t und N_D^t sowie und der korrespondierenden Graphen G_F^t und G_D^t erfolgt unabhängig und parallel, indem bei jedem Iterationsschritt genau ein Stimulus aus der jeweiligen Stimulusmenge mit Zurücklegen gezogen wird, wobei ein normaler SC-GNG-Aufbau unterstellt wird. Alternativ kann auch Ziehen ohne Zurücklegen verwendet werden, wenn ein SC-GNG-Aufbau mit einer Grob- und einer Fein-Adaptionsphase unterstellt wird. Für den Stimulus wird das Gewinner-Neuron in der jeweiligen Neuronenmenge ermittelt, gefolgt von einer speziellen Adaptions-Operation für das Gewinner-Neuron und einer eigenen Adaptions-Operation für die verbundenen Nachbarn, d.h. für die Nachbarschaftsmengen $N(d_G=1 \mid G_F^t)$ und $N(d_G=1 \mid G_D^t)$.

Abb. 59) Unabhängige Erzeugung des Merkmals- und des Dokumentvektoren-Graphen

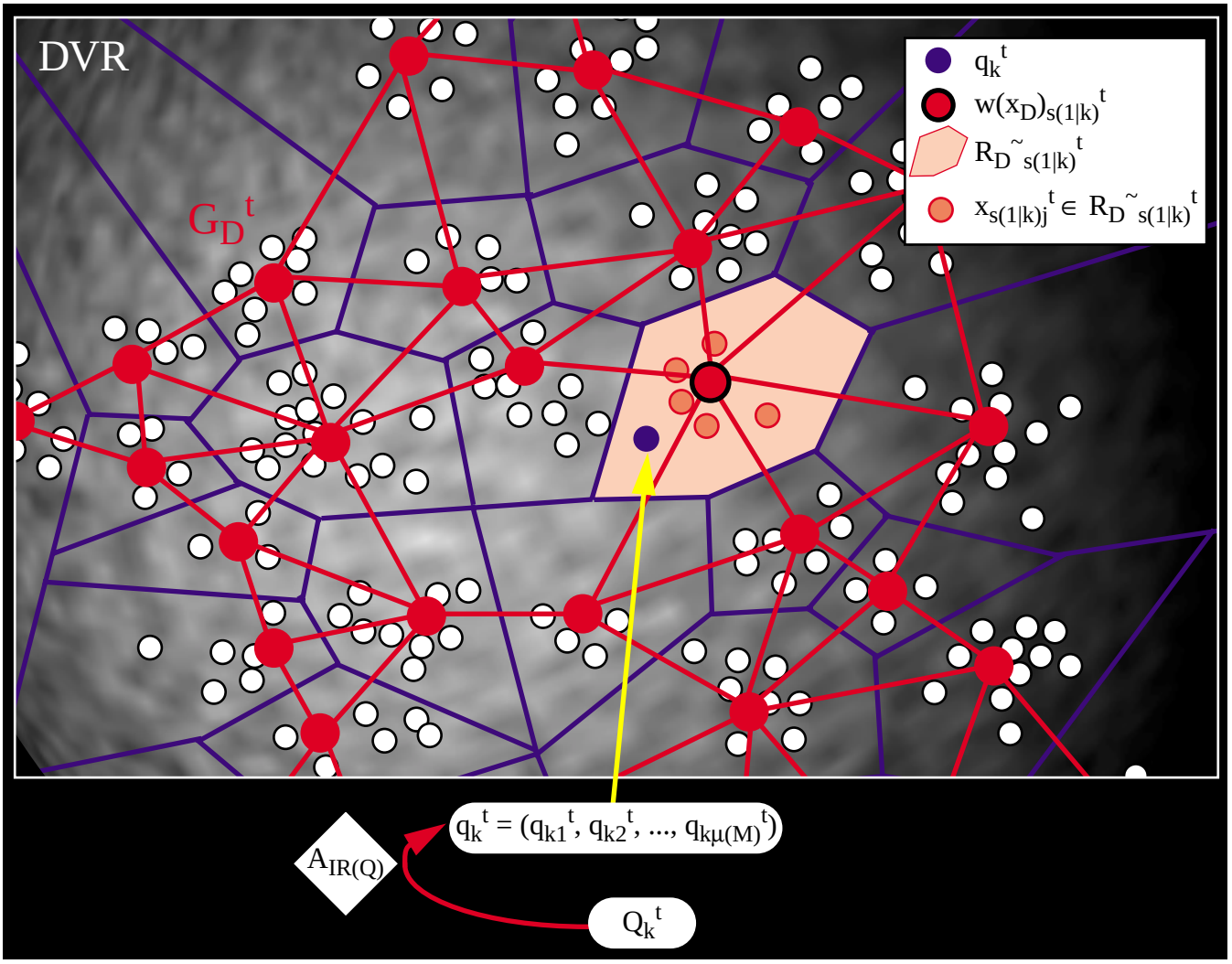


3.8.2) Einfache Cluster-Retrieval-Strategien mit Dokumentvektoren-Graph

Für ein Information-Retrieval mit Hilfe von SC-GNG-Graphen genügt bereits die Erzeugung eines Dokumentvektor-GNG-Graphen, ohne dass ein Merkmalsvektoren-Graph erstellt wird, wobei mit G_D^t bereits einfache Retrieval-Strategien unter Ausnutzung der expliziten Clusterung der Dokumentvektoren-Stimuli durchgeführt werden können.

Im Rahmen einer Monorepräsentation sei eine Query Q_k^t gegeben, aus der durch den Indexierungsprozess $A_{IR(Q)}$ ein Queryvektor q_k^t erzeugt wird (siehe Abb. 60)). Dieser n-dimensionale Queryvektor, mit n = μ_M, nimmt im Dokumentvektorraum DVR genau einen Punkt ein, der durch die Voronoi-Zerlegung des Dokumentraumes durch G_D^t in genau einer Voronoi-Region liegt. Diese Voronoi-Region wird identifiziert durch das Gewinner-Neuron $n_{D,s(1|k)}^t$ bzw. durch das erste Element der Gewinnerliste mit dem Gewichtsvektor $w(x_D)_{s(1|k)}^t$, und diese Region wird als $R_D\tilde{}_{s(1|k)}^t$ bezeichnet.

Abb. 60) Retrieval durch Voronoi-Region des Gewinner-Neurons



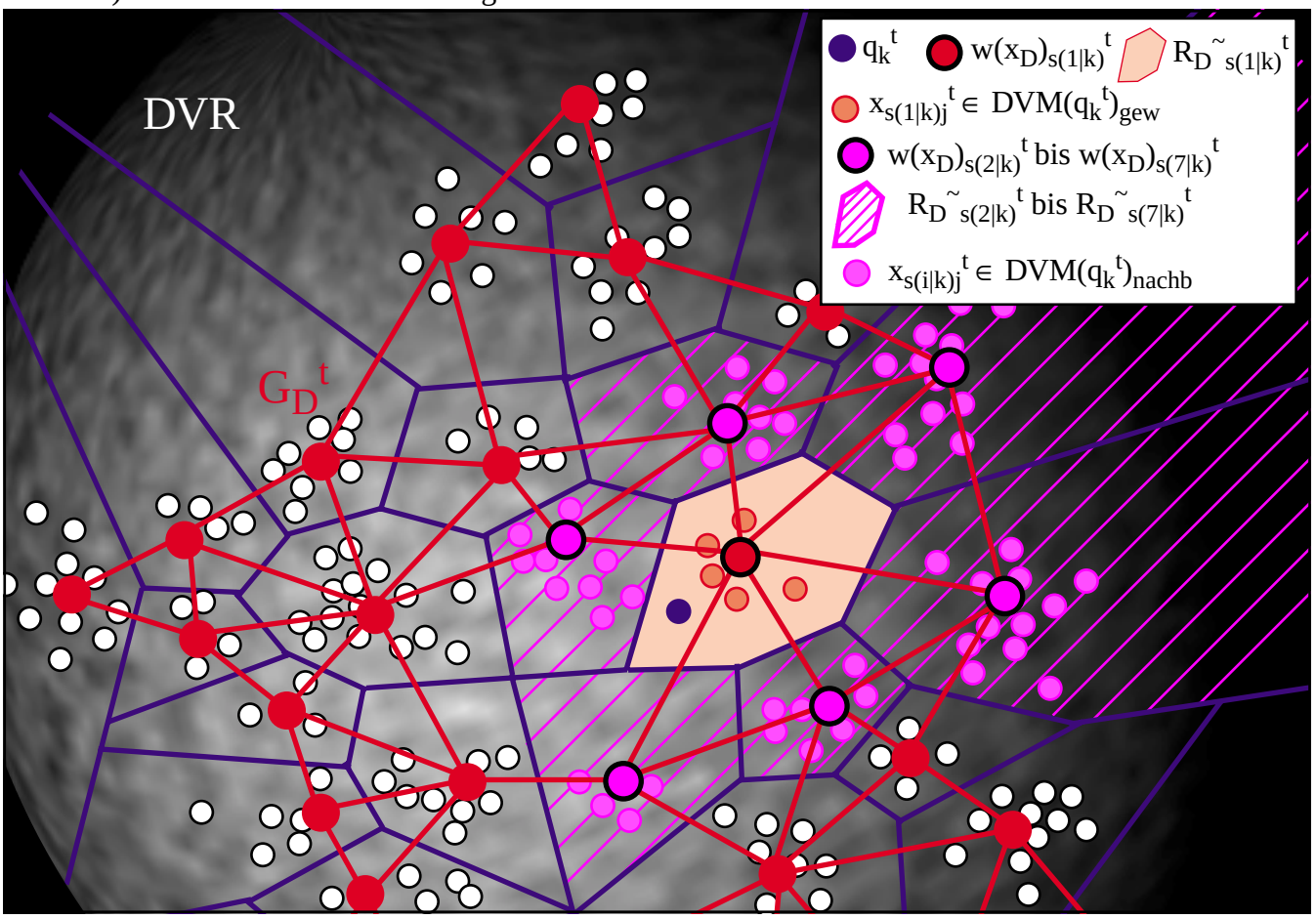
Die einfachste Erzeugung einer Retrieval-Dokumentvektoren-Menge $DVM(q_k^t)$ kann in diesem Kontext aus den Dokumentvektoren $x_{s(1|k)j}^t$ gebildet werden, die innerhalb der Voronoi-Region $R_D\tilde{}_{s(1|k)}^t$ des Gewinner-Neurons liegen. Diese Dokumentvektoren korrespondieren zu den Stimuli $m_{D,s(1|k)j}^t$, die dem Gewinner-Neuron $n_{D,s(1|k)}^t$ zugeordnet sind, und die in der Stimulusmenge $M_{D,s(1|k)}^t$ liegen, wobei diese Retrieval-Strategie somit keine weiteren Parameter wie einen Umgebungsparameter ε erfordert:

$$DVM(q_k^t) = \{x_{s(1|k)j}^t \mid \forall m_{D,s(1|k)j}^t \in M_{D,s(1|k)}^t\}. \tag{380}$$

Eine Erweiterung des Retrievals mit Hilfe von Voronoi-Regionen ergibt sich, wenn neben dem Gewinner-Neuron $n_{D,s(1|k)}^t$ seine Nachbarn aus der Nachbarschaftsmenge $N(d_G=1 \mid G_D^t)_{s(1|k)}^t$ einbezogen werden (siehe Abb. 61)), d.h. aus der Neuronenmenge die direkt mit $n_{D,s(1|k)}^t$ verbunden sind, und somit einen Graphenabstand von $d_G=1$ im Graph G_D^t besitzen. Beispielsweise können zwei Dokumentvektorenmengen gebildet werden, wobei die erste $DVM(q_k^t)_{gew}$ durch die Stimuli aus $M_{D,s(1|k)}^t$, und die andere $DVM(q_k^t)_{nachb}$ durch die Stimuli aus $M_{D,s(i|k)}^t$, für alle $n_{s(i|k)}^t$ aus $N(d_G=1 \mid G_D^t)_{s(1|k)}^t$ gegeben ist:

$$DVM(q_k^t)_{gew} = \{x_{s(1|k)j}^t \mid \forall m_{D,s(1|k)j}^t \in M_{D,s(1|k)}^t\},$$
$$DVM(q_k^t)_{nachb} = \{x_{s(i|k)j}^t \mid n_{s(i|k)}^t \in N(d_G=1 \mid G_D^t)_{s(1|k)}^t, \forall m_{D,s(i|k)j}^t \in M_{D,s(i|k)}^t\}. \tag{381}$$

Abb. 61) Retrieval durch Voronoi-Region des Gewinners und seiner Nachbarn 1



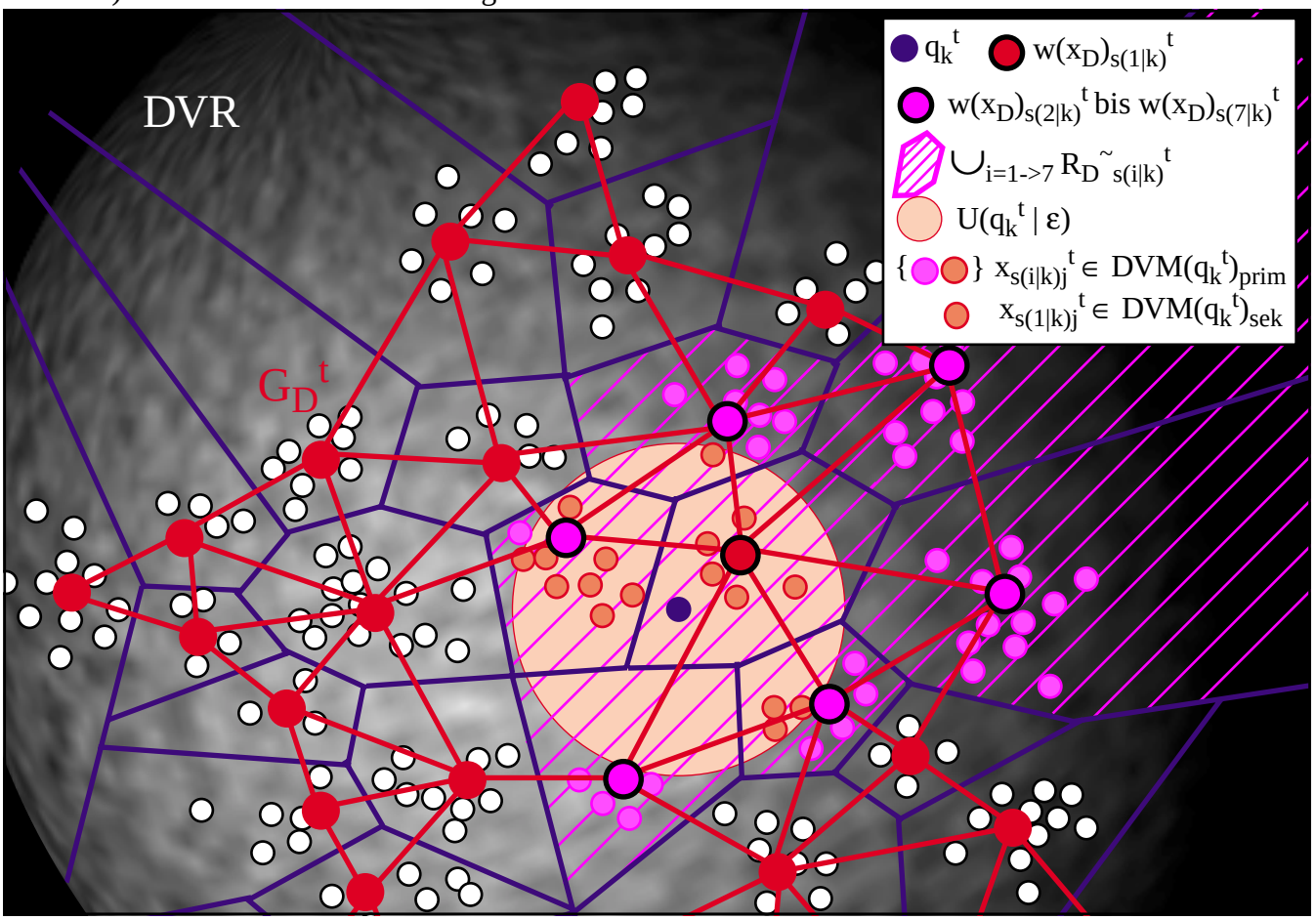
Die Menge $DVM(q_k^t)_{gew} \cup DVM(q_k^t)_{nachb}$ kann auch als Grundmenge für weitere Suchoperationen dienen, indem um q_k^t eine ε-Umgebung $U(q_k^t \mid \varepsilon)$ für Retrieval-Zwecke erzeugt werden soll, analog den Vorgehensweisen bei Cluster-Retrieval-Strategien. Die Nutzung der Verbindungsstruktur in G_D^t besitzt Effizienzvorteile, da im ersten Schritt Distanzen nur zwischen q_k^t und den Gewichtsvektoren $w(x_D)_i^t$ gebildet werden. Im zweiten Schritt werden Distanzen zwischen q_k^t und den vorselektierten Dokumentvektoren gebildet. Die Grundmenge wird somit als primäre Retrievalmenge $DVM(q_k^t)_{prim}$ verwendet:

$$DVM(q_k^t)_{prim} = DVM(q_k^t)_{gew} \cup DVM(q_k^t)_{nachb}. \tag{382}$$

Aus $DVM(q_k^t)_{prim}$ werden die Dokumentvektoren aus der vorselektieren Menge $DVM(q_k^t)_{prim}$ ermittelt, die innerhalb $U(q_k^t \mid \varepsilon)$ liegen, wobei die sich ergebende Menge als sekundäre Retrievalmenge $DVM(q_k^t)_{sek}$ bezeichnet wird:

$$DVM(q_k^t)_{sek} = \{x_{s(i|k)j}^t \mid x_{s(i|k)j}^t \in DVM(q_k^t)_{prim} \wedge x_{s(i|k)j}^t \in U(q_k^t \mid \varepsilon)\}. \tag{383}$$

Abb. 62) Retrieval durch Voronoi-Region des Gewinners und seiner Nachbarn 2



3.8.3) Cluster-Retrieval-Strategien mit positiven und negativen Queryvektoren

Gegeben ist eine Dokumentvektoren-Klassifikation durch eine GNG-SOM N_D^t, wobei von einer Dokumentvektoren-Monorepräsentation ausgegangen werden soll. Weiterhin liegt eine Menge QVM_i^{+t} von positiven Queryvektoren und eine Menge QVM_i^{-t} von negativen Queryvektoren eines einzelnen Agenten Ag_i^t vor, wobei von einer jeweiligen Queryvektor-Polyrepräsentation ausgegangen werden soll, d.h. $m_{iQ+}{}^t > 1$ und $m_{iQ-}{}^t > 1$:

$$QVM_i^{+t} = \{q_{ij}^{+t} \in DVR \mid j = 1, ..., m_{iQ+}{}^t\},\ QVM_i^{-t} = \{q_{ij}^{-t} \in DVR \mid j = 1, ..., m_{iQ-}{}^t\}. \quad (384)$$

Die Queryvektoren liegen in einzelnen Voronoi-Regionen, wobei die Zuordnung durch eine Präsentation mit der Ermittlung des Gewinner-Neurons erzeugt wird. D.h. für jeden Queryvektor q_{ij}^{+t} oder q_{ij}^{-t} kann das Gewinner-Neuron $n_{D,s(1|j+)}{}^t$ bzw. $n_{D,s(1|j-)}{}^t$ ermittelt werden:

$$d_{DVR}(w(x_D)_{s(1|j+)}^t, q_{ij}^{+t}) = \min\{d_{DVR}(w(x_D)_p^t, q_{ij}^{+t}) \mid p = 1, ..., \mu_{N,D}{}^t\},\ \forall q_{ij}^{+t} \in QVM_i^{+t};$$
$$d_{DVR}(w(x_D)_{s(1|j-)}^t, q_{ij}^{-t}) = \min\{d_{DVR}(w(x_D)_p^t, q_{ij}^{-t}) \mid p = 1, ..., \mu_{N,D}{}^t\},\ \forall q_{ij}^{-t} \in QVM_i^{-t}. \quad (385)$$

Die Neuronenstruktur soll erweitert werden, um eine Menge für die positiven und die negativen Queryvektoren aufzunehmen, wobei die Zuordnung temporär für jede Retrieval-Operation und für jeden Agenten durchgeführt wird, d.h. es werden die beiden agentenspezifischen Mengen $M_{Q+,ip}{}^t$ und $M_{Q-,ip}{}^t$ gebildet. Erweiterte Retrievalstrategien, bei denen die Queryvektoren eines Agenten über eine Anzahl

von Retrieval-Operationen gesammelt wird, oder bei denen die Queryvektoren unterschiedlicher Agenten gesammelt werden, um Agentengruppen zu bilden, sollen hier nicht betrachtet werden:

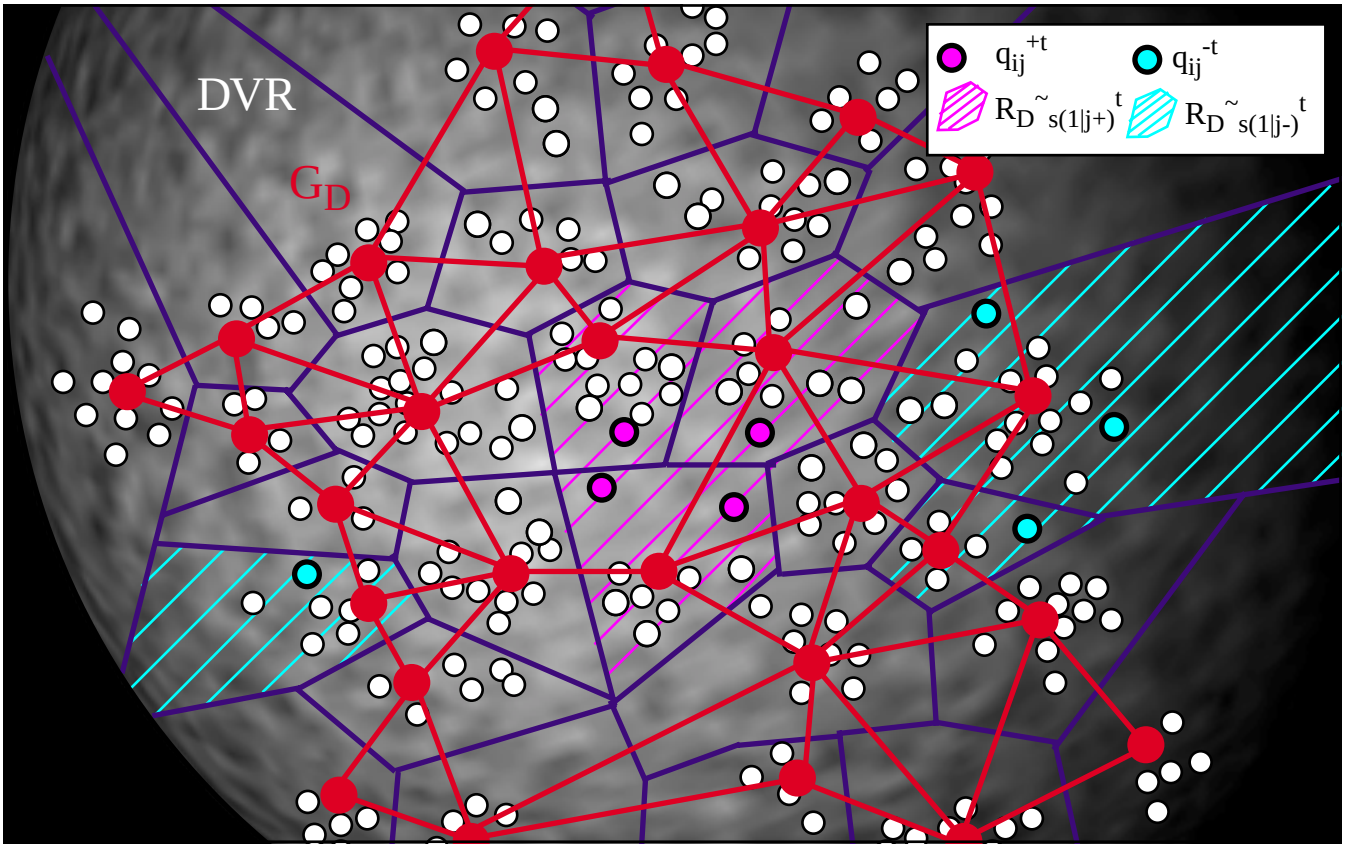
$$N_D^t = \{n_{D,p}^t = (w(x_D)_p^t, M_{D,p}^t, M_{Q+,ip}^t, M_{Q-,ip}^t, C_{D,p}^t) \mid p = 1, ..., \mu_{N,D}^t\}. \quad (386)$$

Die Neurone können entsprechend den Mengen $M_{Q+,ip}^t$ und $M_{Q-,ip}^t$ in vier Klassen zerlegt werden:

$$\begin{aligned} N_{D+}^t &= \{n_{D,p}^t \mid M_{Q+,ip}^t \neq \varnothing, M_{Q-,ip}^t = \varnothing\}, \\ N_{D-}^t &= \{n_{D,p}^t \mid M_{Q+,ip}^t = \varnothing, M_{Q-,ip}^t \neq \varnothing\}, \\ N_{D+-}^t &= \{n_{D,p}^t \mid M_{Q+,ip}^t \neq \varnothing, M_{Q-,ip}^t \neq \varnothing\}, \\ N_{D/}^t &= \{n_{D,p}^t \mid M_{Q+,ip}^t = \varnothing, M_{Q-,ip}^t = \varnothing\}. \end{aligned} \quad (387)$$

Unproblematisch sind alle Fälle, bei denen N_{D+-}^t leer ist, d.h. in keiner der Voronoi-Regionen liegt gleichzeitig ein positiver und ein negativer Queryvektor (siehe Abb. 63)).

Abb. 63) Dokumentvektoren-GNG-SOM mit eindeutiger Zuordnung pos. und neg. Queryvektoren

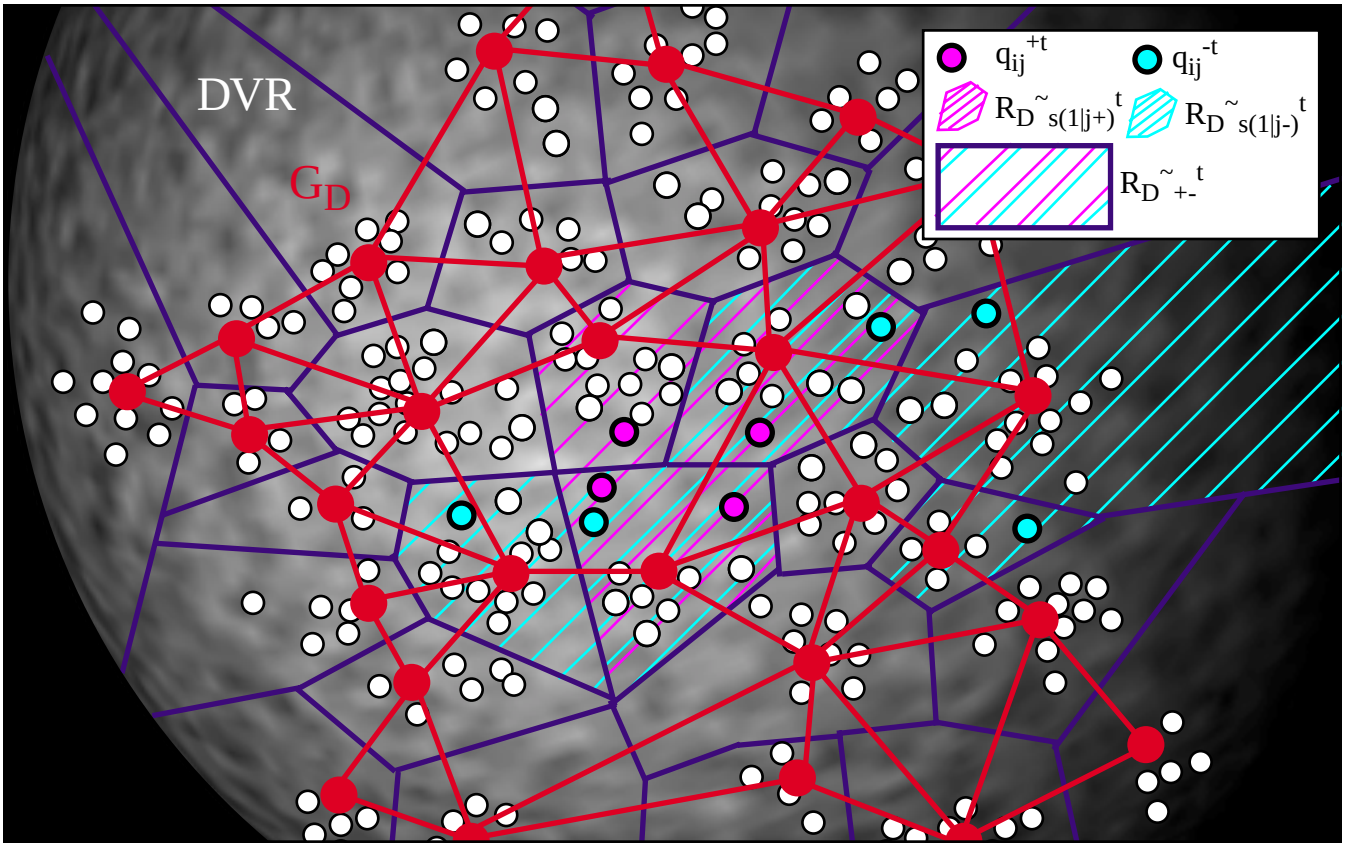


Wird in diesem Fall eine Retrievalregion auf der Basis der Voronoi-Regionen verwendet, so ist dies problemlos, da durch die Zerlegung der Dokumentvektorenmenge durch die GNG-SOM nur leere Schnittmengen zwischen den lokalen Stimulusmengen existieren. Dadurch spielt bei dieser Retrievalstrategie das Vorhandensein negativer Queryvektoren keine Rolle. Die Retrievalmenge $DV(q_{ij}^{+t})$ eines positiven Queryvektors q_{ij}^{+t}, der in der Voronoi-Region $R_D\tilde{}_{s(1|j+)}^t$ liegt, ist $M_{D,s(1|j+)}^t$, sodass sich die Gesamt-Retievalmenge ergibt:

$$DV(QVM_i^{+-t}) := DV(QVM_i^{+t}) = \bigcup_j M_{D,s(1|j+)}^t, \forall q_{ij}^{+t} \in QVM_i^{+t}. \quad (388)$$

Wird die Retrievalregion durch ε-Umgebungen gebildet, so ergibt sich wiederum die Möglichkeit nicht leerer Schnittmengen zwischen positiven und negativen Regionen, wobei die Pareto-Kriterium-Strategien aus Abschnitt 3.6.6) angewendet werden können. Im weiteren soll dies jedoch nicht vertieft werden. Statt dessen soll der Fall betrachtet werden, dass $N_{D+-}^{\ t}$ nicht leer ist, d.h. dass Neurone aus $N_D^{\ t}$ existieren, in deren Querymengen $M_{Q+,ip}^{\ t}$ und $M_{Q-,ip}^{\ t}$ jeweils mindestens ein Element vorliegt (siehe Abb. 64)).

Abb. 64) Dokumentvektoren-GNG-SOM mit nicht eindeutiger Zuordnung von Queryvektoren



Zunächst kann in diesem Fall aus der Gesamtmenge der Dokumentvektoren $DV(QVM_i^{\ +-t})$, die in einer Voronoi-Region liegen, in der einer der Queryvektoren liegt, die Dokumentvektoren ausgeschlossen werden, die in einer Voronoi-Region liegen, in der sich nur negative Queryvektoren befinden. Diese Teilmenge soll mit $DV(\neg QVM_i^{\ +t} \wedge QVM_i^{\ -t})$ bezeichnet werden:

$$DV(QVM_i^{\ +-t})' = DV(QVM_i^{\ +-t}) \setminus DV(\neg QVM_i^{\ +t} \wedge QVM_i^{\ -t}),$$

$$DV(QVM_i^{\ +-t}) = \{x_k^{\ t} \in DV(QVM_i^{\ +-t}) \mid \exists_{\geq 1}\, q_{ij}^{\ +t}: x_k^{\ t} \in R_D^{\sim}{}_{s(1|j+)}^{\ t} \vee \exists_{\geq 1}\, q_{ij}^{\ -t}: x_k^{\ t} \in R_D^{\sim}{}_{s(1|j-)}^{\ t}\},$$

$$DV(\neg QVM_i^{\ +t} \wedge QVM_i^{\ -t}) = \{x_k^{\ t} \in DV(QVM_i^{\ +-t}) \mid \forall q_{ij}^{\ +t}: x_k^{\ t} \notin R_D^{\sim}{}_{s(1|j+)}^{\ t} \wedge$$

$$\exists_{\geq 1}\, q_{ij}^{\ -t}: x_k^{\ t} \in R_D^{\sim}{}_{s(1|j-)}^{\ t}\}. \qquad (389)$$

Alle Dokumentvektoren aus $DV(QVM_i^{\ +-t})'$ werden nachgewiesen, wobei verschiedene Rankingstrategien angewendet werden können, welche die Anzahl der positiven und negativen Queryvektoren in einer Voronoi-Region, bzw. die Distanzen zu den positiven und negativen Queryvektoren verwenden. Jedem Dokumentvektor $x_k^{\ t}$ aus $DV(QVM_i^{\ +-t})$ wird ein zwei-komponentiger Vektor zugeordnet, der die Anzahl der positiven bzw. negativen Queryvektoren aufzählt, die in der Voronoi-Region liegen, in der auch $x_k^{\ t}$ liegt. Hierzu können direkt die beiden Mengen $M_{Q+,i,s(1|k)}^{\ t}$ und $M_{Q-,i,s(1|k)}^{\ t}$ verwendet werden:

$$\forall x_k^t \in DV(QVM_i^{+-t}): (\#M_{Q+,i,s(1|k)}^t, \#M_{Q-,i,s(1|k)}^t). \quad (390)$$

Die Verwendung eines solchen zwei-komponentigen Vektors in Verbindung mit einer Pareto-Hierarchie durch der Bildung sukzessiver Pareto-Mengen führt jedoch zu einem groben Ranking, da die Elemente des Rankings eigentlich keine Dokumentvektoren sind, sondern die lokalen Dokumentvektorenmengen $M_{D,p}^t$ der Neurone $n_{D,p}^t$ aus der Vereinigungsmenge $N_{D+}^t \cup N_{D+-}^t$. Dies ergibt sich daraus, dass allen Dokumentvektoren $x_{D,pk}^t$ aus $M_{D,p}^t$ der gleiche Vektor zugeordnet wird.

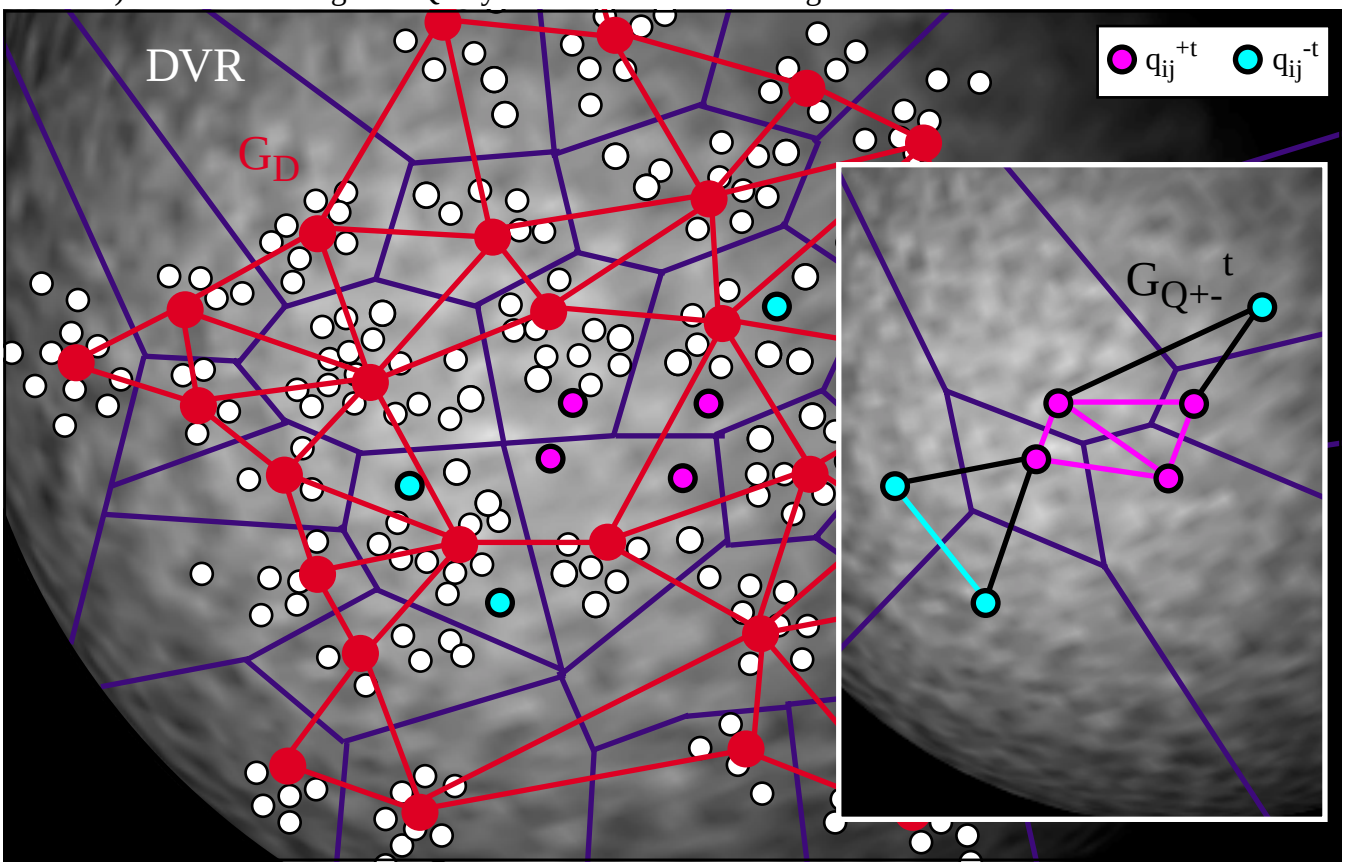
Ein feineres Ranking ergibt sich, wenn die Distanzen eines Dokumentvektors zu den einzelnen Queryvektoren bestimmt werden. Dabei können die Distanzen zu den positiven wie den negativen Queryvektoren zu d_k^{+t} und d_k^{-t} aggregiert werden, sodass jedem Dokumentvektor x_k^t ein zwei-dimensionaler Vektor (d_k^{+t}, d_k^{-t}) zugeordnet wird, deren erste Komponente minimiert und dessen zweite Komponente maximiert werden soll. Durch eine Sequenz aus sukzessiv deaktivierten Paretomengen kann ein feineres Ranking erzeugt werden, wobei dies jedoch nicht vertieft werden soll. Die Dokumentvektoren-Klassifikation N_D^t liefert nur die Dokumentvektoren, mit der das Ranking aufgebaut wird, sie liefert jedoch keinen eigenen Beitrag zur Erzeugung einer solchen Pareto-Hierachie.

3.8.4) Triangulation positiver und negativer Queryvektoren

Nachdem die Nutzung einer Dokumentvektoren-Clusterung beim Vorliegen positiver und negativer Queryvektoren betrachtet wurde, könnte eine Queryvektor-Clusterung betrachtet werden. Diese ist jedoch nur dann sinnvoll, wenn eine hinreichende Anzahl von Vektoren vorliegt, wovon im Fall einer einzelnen Retrieval-Interaktion eines Agenten kaum ausgegangen werden kann. Dies ändert sich bei der Akkumulation der Retrieval-Interaktionen eines einzelnen Agenten über einen längeren Zeitraum, oder bei der Verwendung von langfristigen Repräsentationen vieler Agenten im Rahmen eines Information-Filter-Systems, wenn Filtervektoren geclustert werden.

Trotzdem kann die Verwendung einer GNG-SOM (oder eines anderen Triangulations-Verfahrens) im Kontext einer vergleichsweise kleinen Anzahl von Queryvektoren in QVM_i^{+t} und QVM_i^{-t} sinnvoll sein, wenn die Queryvektoren nicht geclustert werden, sondern indem eine Delaunay-Triangulation der Queryvektoren selbst durchgeführt wird. Ein wachsender Graph wie ein GNG-SOM kann hierfür verwendet werden, indem der Aufbau solange durchgeführt wird, bis die Anzahl der Gewichtsvektoren gleich der Anzahl der Queryvektoren ist, die als Stimuli verwendet werden. Eine Adaption mit Stimuli-Umverteilungen wird solange durchgeführt, bis jedes Neuron in seiner lokalen Stimulusmenge genau ein Element besitzt, ohne dass eine Umverteilung einen besseren Wert eines Qualitätsmaß wie dem Quantifizierungsfehler ergibt. Danach wird der Gewichtsvektor des Neurons ersetzt durch den Stimulusvektor, d.h. durch den Queryvektor, unter Beibehaltung der lokalen Verbindungsstruktur, sodass sich eine Triangulation der Stimulusvektoren ergibt. Im betrachteten Kontext sollen die Queryvektoren aus QVM_i^{+t} und QVM_i^{-t} im Dokumentvektorenraum DVR plaziert und trianguliert werden, wobei der Graph der Queryvektoren durch G_{Q+-}^t bezeichnet werden soll (siehe Abb. 65)). Auf Strategien der unabhängigen Triangulation der positiven und der negativen Queryvektoren mit den Graphen G_{Q+}^t und G_{Q-}^t soll im weiteren zugunsten eines integrierten Graphen verzichtet werden.

Abb. 65) Positive und negative Queryvektoren mit ihrer Triangulation



Mit Hilfe der Voronoi-Regionen, die sich aus dem Graphen G_{Q+-}^{t} ergeben, können Dokumentvektoren aus DV^t neu geclustert werden, wobei alle Elemente aus DV^t verwendet werden können (siehe Abb. 66)), oder indem zunächst geeignete Teilmengen gebildet werden, die dann geclustert werden. Aufgrund der großen Anzahl von Dokumentvektoren in DV^t erscheint jedoch die Auswahl einer geeigneten Teilmenge als effizientere Strategie.

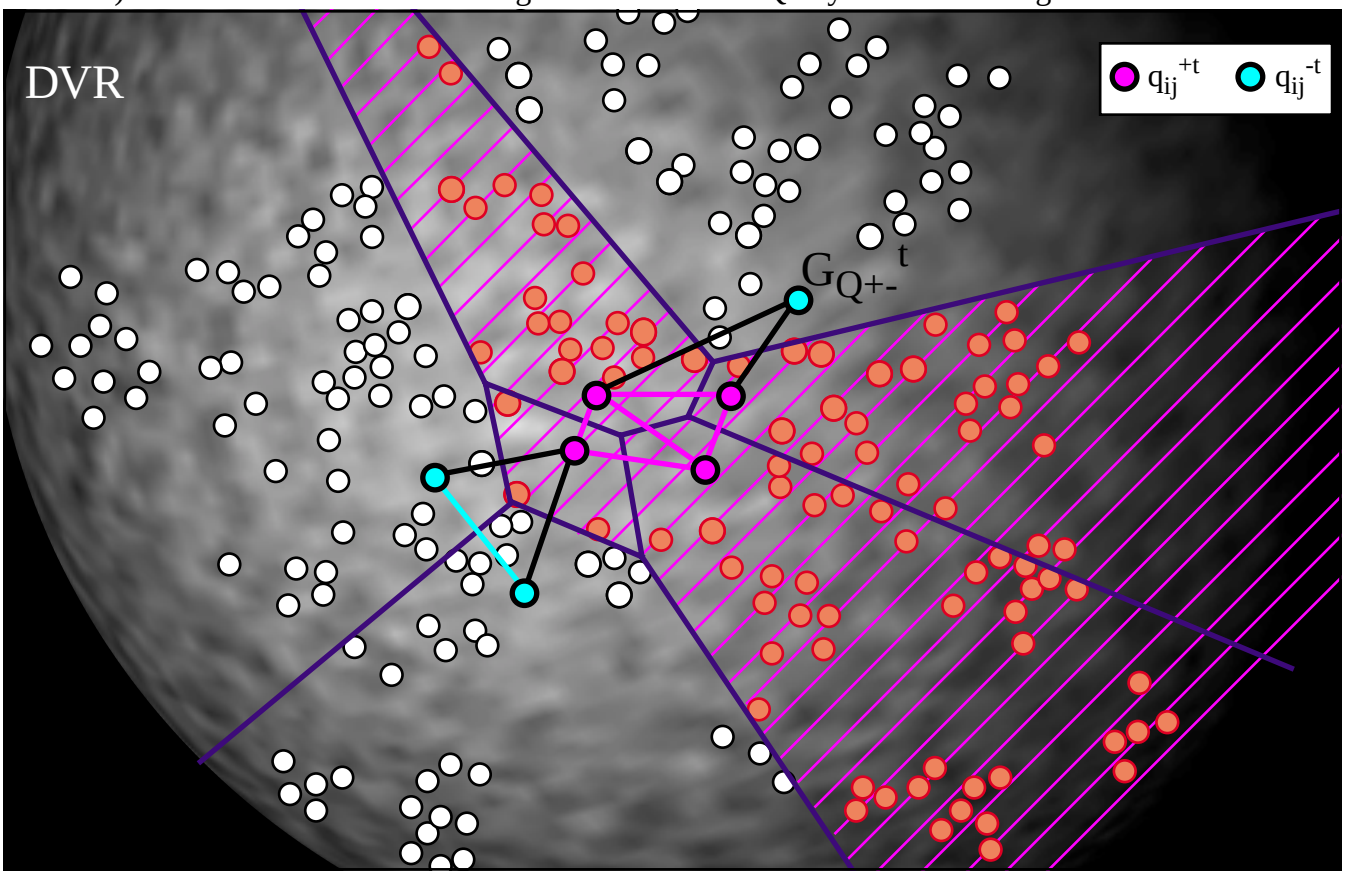
Als Grundlage für die Auswahl einer Teilmenge kann die Zuordnung der Queryvektoren zu den Dokumentvektor-Neuronen $n_{D,p}^{t}$ aus der GNG-SOM N_D^t verwendet werden. Hierzu wird die Vereinigung der Neurone aus den Mengen N_{D+}^{t}, N_{D-}^{t} und N_{D+-}^{t} gebildet, d.h. es werden alle Neurone ermittelt, in deren Voronoi-Region mindestens ein Queryvektor liegt, wobei die Vereinigungsmenge als Primärmenge $N_{D(QVM)prim}^{t}$ bezeichnet werden soll:

$$N_{D(QVM)prim}^{t} = \{n_{D,k}^{t} \in \{N_{D+}^{t} \cup N_{D-}^{t} \cup N_{D+-}^{t}\}\}. \quad (391)$$

Die Klassifikation auf der Basis von G_{Q+-}^{t} kann nun auf der Teilmenge der Dokumentvektoren durchgeführt werden, die in den lokalen Stimulusmengen der Neurone in der Primärmenge liegen. Diese Dokumentvektorenmenge kann ebenfalls als Primärmenge $DV(QVM_i^{+-t})_{prim}$ bezeichnet werden:

$$DV(QVM_i^{+-t})_{prim} = \bigcup_k M_{D,k}^{t}, \forall n_{D,k}^{t} \in N_{D(QVM)prim}^{t}. \quad (392)$$

Abb. 66) Dokumentvektoren-Clusterung auf der Basis der Queryvektoren-Triangulation



Zusätzlich kann eine Sekundärmenge $N_{D(QVM)sec}^{t}$ gebildet werden, die aus den unmittelbaren Nachbarneuronen der Elemente aus der Primärmenge besteht ohne die Elemente der Primärmenge selbst. D.h. es werden alle Neurone aus der Differenzmenge $N_D^t \setminus N_{D(QVM)prim}^{t}$ ermittelt, die in der Nachbarschaftsmenge $N(1 \mid G_D^t)_k$ eines Neurons $n_{D,k}^{t}$ aus der Primärmenge $N_{D(QVM)prim}^{t}$ liegen:

$$N_{D(QVM)sec}^{t} = \{n_{D,p}^{t} \in N_D^t \setminus N_{D(QVM)prim}^{t} \mid n_{D,p}^{t} \in N(1 \mid G_D^t)_k, \forall n_{D,k}^{t} \in N_{D(QVM)prim}^{t}\}. \quad (393)$$

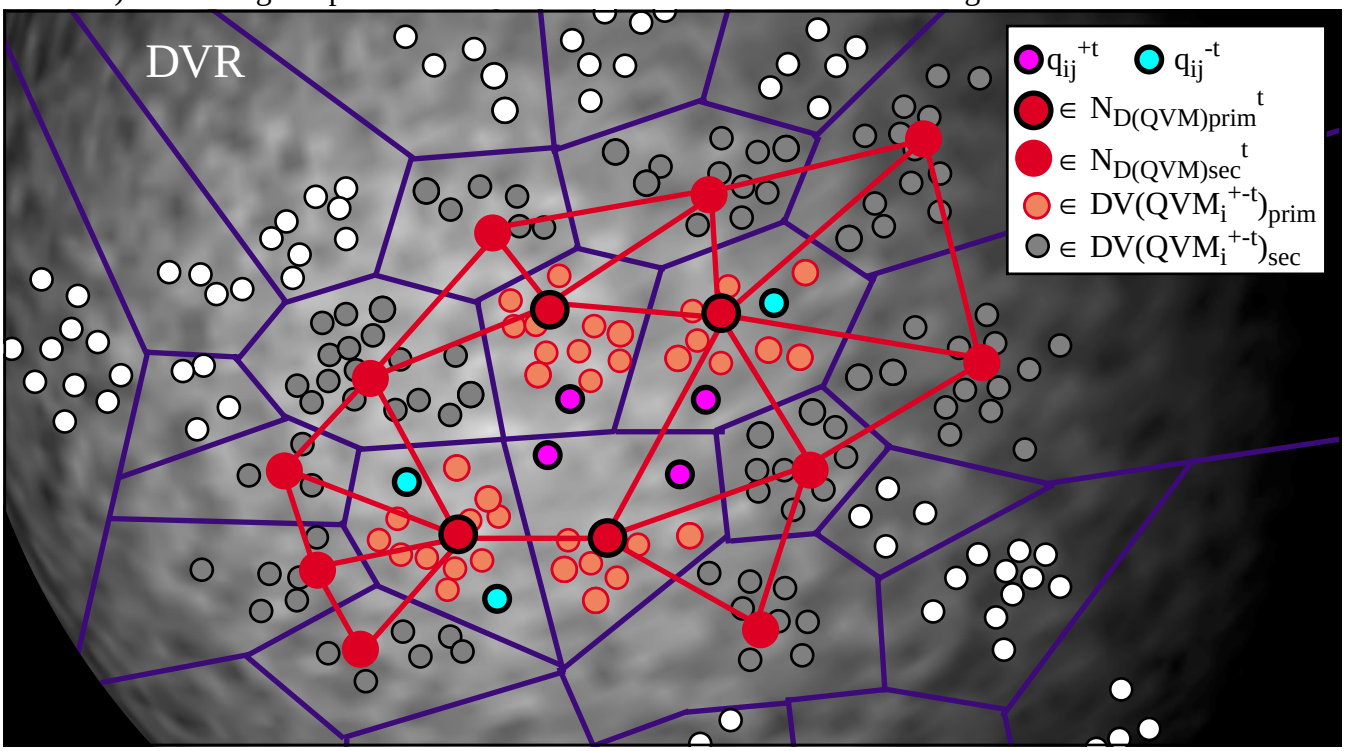
Die Gesamtklassifikation kann nun auf der Basis der Dokumentvektoren aus der oben definiert Primärmenge $DV(QVM_i^{+-t})_{prim}$ und der aus $N_{D(QVM)sec}^{t}$ abgeleiteten Sekundärmenge $DV(QVM_i^{+-t})_{sec}$ erfolgen (siehe Abb. 67) und Abb. 68)):

$$DV(QVM_i^{+-t})_{ges} = DV(QVM_i^{+-t})_{prim} \cup DV(QVM_i^{+-t})_{sec}\text{ , mit}$$
$$DV(QVM_i^{+-t})_{sec} = \bigcup_k M_{D,p}^{t}, \forall n_{D,p}^{t} \in N_{D(QVM)sec}^{t}. \quad (394)$$

In Abb. 68) wird die Klassifikation auf der Basis von $DV(QVM_i^{+-t})_{ges}$ durchgeführt, indem alle Dokumentvektoren aus dieser Grundmenge, die in einer Voronoi-Region eines positiven Queryvektors liegen, akzeptiert werden, und alle Dokumentvektoren, die in der Voronoi-Region eines negativen Queryvektors liegen, abgelehnt werden. Die Menge der akzeptierten Dokumentvektoren, die auf der Basis der Voronoi-Regionen des Queryvektorgraphen G_{Q+-}^{t} erzeugt wird, soll mit $DV(QVM_i^{+-t} \mid G_{Q+-}^{t})$ bezeichnet werden, d.h. darin sind alle x_k^t aus $DV(QVM_i^{+-t})_{ges}$ enthalten, für die ein positiver Queryvektor q_{ij}^{+t} aus der Menge QVM_i^{+-t} existiert, wobei x_k^t in der Voronoi-Region $R_{\tilde{Q}_{ij+}}^{t}$ liegt:

$$DV(QVM_i^{+-t} \mid G_{Q+-}^{t}) = \{x_k^t \in DV(QVM_i^{+-t})_{ges} \mid \exists q_{ij}^{+t} \in QVM_i^{+-t}: x_k^t \in R_{\tilde{Q}_{ij+}}^{t}\}. \quad (395)$$

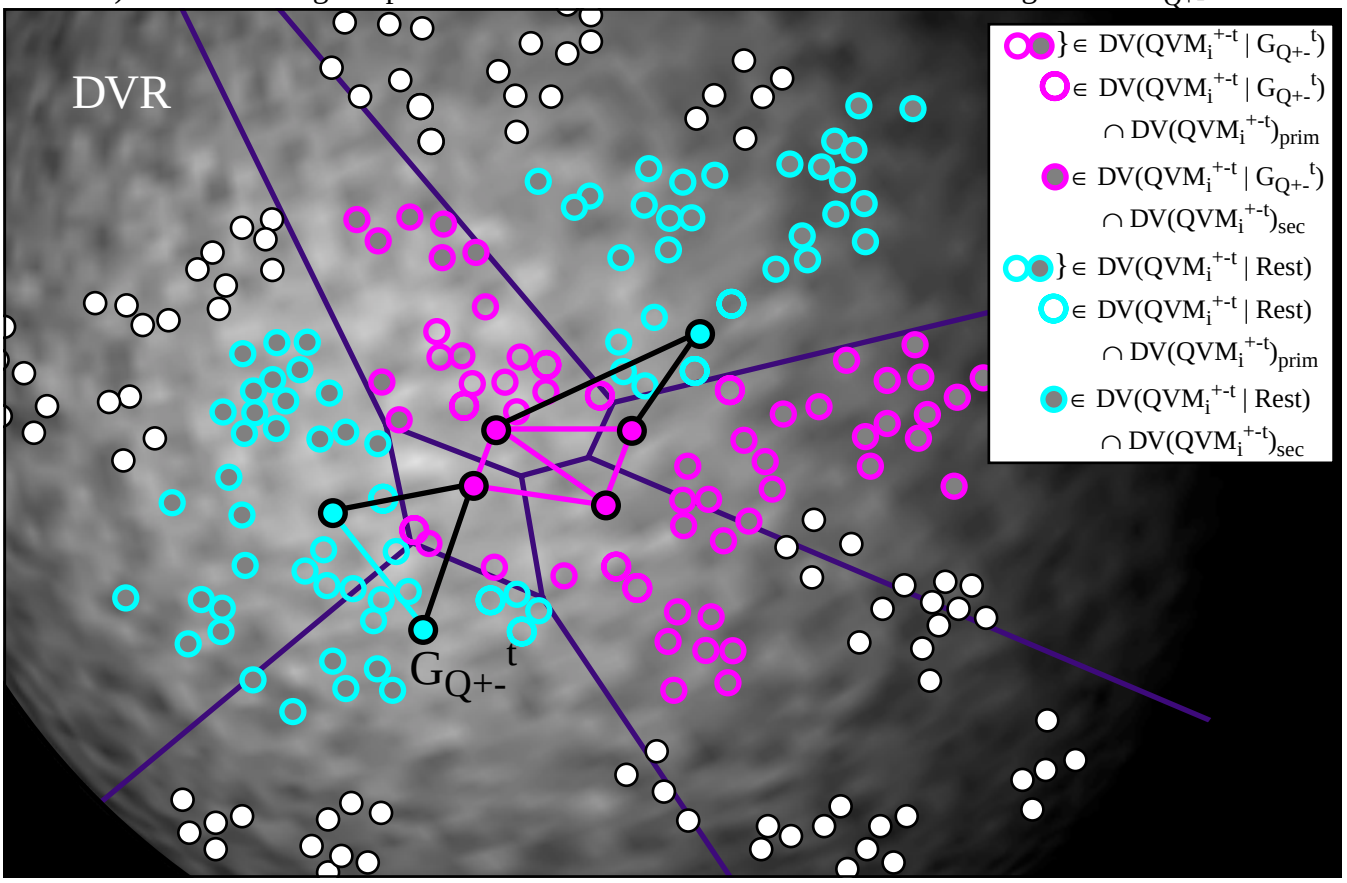
Abb. 67) Ermittlung der primären und sekundären Dokumentvektorenmenge



Die nicht akzeptierten Dokumentvektoren in der Restmenge $DV(QVM_i^{+-t} \mid Rest)$ zusammengefasst:

$$DV(QVM_i^{+-t} \mid Rest) = DV(QVM_i^{+-t})_{ges} \setminus DV(QVM_i^{+-t} \mid G_{Q+-}^{t}). \quad (396)$$

Abb. 68) Klassifizierung der primären und sekundären Dokumentvektorenmenge durch G_{Q+-}^{t}



3.8.5) Retrieval-Strategien mit Query-Modifikation

Komplexere Retrieval-Strategien lassen sich durchführen, wenn eine Voronoi-Zerlegung im Merkmals- und im Dokumentvektorenraum durchgeführt wurde, wodurch N_F^t und N_D^t erzeugt wurden. Entsprechende Strategien basieren darauf, dass der Queryvektor q_k^t modifiziert wird, indem Operationen im Merkmalsvektorenraum durchgeführt werden. Besitzen einzelne Komponenten von q_k^t den Wert Null, weil sie in der Original-Query Q_k^t nicht auftreten, so kann die Modifikation als Query- oder Queryvektor-Expansion (Efthimiadis (1993[106]), Harman (1992[157]), Kwok (1991[193], 1995[194]), Mitra et al. (1998[222]), Spink (1994[320])) bezeichnet werden, wenn der modifizierte Vektor an den entsprechenden Komponenten Werte ungleich Null aufweist, und sonst keine Komponenten auf Null gesetzt werden.

Im gegebenen Kontext sollen Query-Modifikationen durch die Strukturen N_F^t im Merkmalsraum durchgeführt werden, bei der die semantischen Strukturen genutzt werden, die durch Selbstorganisationsprozesse des unüberwachten Lernens erzeugt wurden, und die über die Strukturen in der Dokument-Merkmals-Matrix hinausgehen. Es soll die Fähigkeit der lateralen Aktivitätsausbreitung eingesetzt werden (siehe Abschnitt 2.1.8)), woraus zunächst eine Erweiterung der Neuronenstruktur in N_F^t folgt, indem der Inputwert $net_{F,i}^t$, der unveränderliche aber individuelle Schwellenwert $T_{F,i}$, der Aktivietätswert $z_{F,i}^t$ und der Outputwert $S(z_{F,i}^t)$ hinzugefügt werden, und indem die Iterationsabhängigkeit des Gewichtsvektors $w(x_F)_i$ und des Verbindungsvektors $C_{F,i}$ für die Phase der Aktivitätsausbreitung und die Retrievalphase ausgesetzt wird. Es ergibt sich eine Neuronenstruktur:

$$N_{F,AA}^t = \{n_{F,i}^t = (w(x_F)_i, M_{F,i}^t, C_{F,i}, net_{F,i}^t, T_{F,i}, z_{F,i}^t, S(z_{F,i}^t)) \mid i = 1, ..., \mu_{N,F}^t\}. \quad (397)$$

Durch die Modifikation soll der Graph G_F^t zu dem Graphen $G_{F,AA}^t$ werden, der sich von dem ersten dadurch unterscheidet, dass die Verbindungskanten nun die laterale Aktivitätsausbreitung ermöglicht, indem jede vorher ungerichtete Kante nun zwei-seitig gerichtet ist (siehe $G_{F,AA}^t$ in Abb. 69)).

Die Stimulusmenge $M_{F,i}^t$ soll trotz der konstanten Elemente während der Aktivitätsausbreitungs- und Retrievalphase eine Iterationsabhängigkeit behalten, da sich die Struktur einzelner Stimuluselemente durch die Aktivitätsausbreitung verändert. Dies ergibt sich daher, dass ein Inputwert $net(x)_{F,ij}^t$, ein Aktivitätswert $z(x)_{F,ij}^t$ und ein Outputwert $S(z(x)_{F,ij}^t)$ hinzugefügt werden, die sich am Anfang und am Ende einer Aktivitätsausbreitung verändern. Der externe Gewichtungsfaktor $c(x)_{F,ext}$, der für die Berechnung des Inputwertes $net(x)_{F,ij}^t$ bei der Initialisierung notwendig ist, sowie der Schwellenwert $T_{F,x}$, der für die Berechnung der Aktivität $z(x)_{F,ij}^t$ notwendig ist, sollen konstant und überindividuell sein, d.h. sie sind nicht auf der Ebene von einzelnen Stimuli, sondern auf der Ebene der Stimulusmengen gegeben:

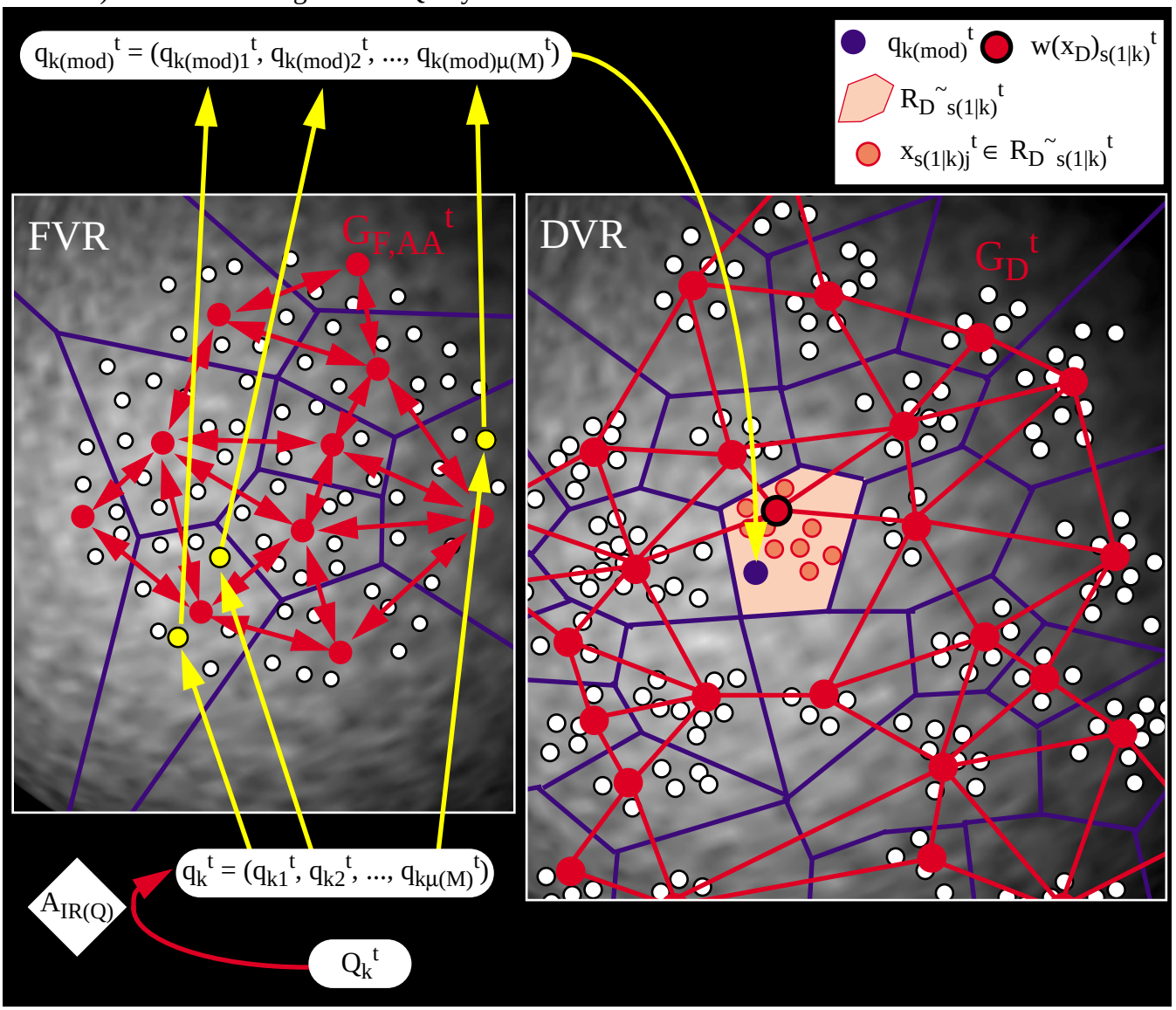
$$M_{F,i}^t = \{m_{F,ij}^t = (x_{F,ij}, n_{F,i}, c(x)_{F,ij}, net(x)_{F,ij}^t, z(x)_{F,ij}^t, S(z(x)_{F,ij}^t)) \mid c(x)_{F,ext}, T_{F,x}, j = 1, ..., \mu_{M(F,i)}^t\}. \quad (398)$$

Beim Preprocessing wird der Queryvektor $q_k = (q_{kj} \mid j = 1, ..., n = \mu_{M(F)})$, der sich nach der Indexierung $A_{IR(Q)}$ der Original-Query Q_k ergibt, als externer Inputvektor in die Stimulusvektorenverteilung M_F^t eingespeist, sodass jeder Stimulus $m_{F,ij}$ einen Initialisierungs-Inputwert $net(x)_{F,ij}^{t=0}$ als Funktion von der korrespondierenden Komponente q_{kj} und dem Initialisierungs-Gewichtungsfaktor $c(x)_{F,ext}$ erhält. Danach wird die Initialisierungs-Aktivität $z(x)_{F,ij}^{t=0}$ und der Initialisierungs-Outputwert $S(z(x)_{F,ij}^{t=0})$ des

Stimulus berechnet. Alle Outputwerte der Stimuli in einer lokalen Menge $M_{F,i}^t$ werden im nächsten Schritt dem Neuron $n_{F,i}$ zugeführt, das seinen Input-, Aktivitäts- und Outputwert berechnet.

Bei der eigentlichen lateralen Aktivitätsausbreitung wird der Neuron-Output den verbundenen Neuronen in $N(d_G=1 \mid G_F^t)_{F,i}$ zugeführt, und $n_{F,i}$ erhält von den Neuronen dieser Nachbarmenge ihren Outputwert, mit dem es seinen Input-, Aktivitäts- und Outputwert der nächten Iteration t+1 berechnet. Nachdem t_{max} oder $t_{F,max}$ Iterationen der Aktivitätsausbreitung über die Verbindungskanten des GNG-Graphen G_F^t durchgeführt wurden, wird das Verfahren abgebrochen. Das alternative Abbruchkriterium, wie das Erreichen eines Punkt-Attraktors als Aktivierungsgleichgewicht, soll unberücksichtigt bleiben, da a priori nicht sicher ist, bzw. nicht einfach ermittelbar ist, ob genau ein solcher Attraktor existiert, ob mehrere Punkt-Attraktoren existieren, ob Mehr-Punkt-Attraktoren oder Strange-Attractors (z.B. Peitgen et al. (1992[255], 1994[256])) existieren.

Abb. 69) Retrieval-Strategie durch Queryvektor-Modifikation



Beim Postprocessing wird die Aktivierung eines Neurons $n_{F,i}$ an die Stimuli $m_{F,ij}^{t(max)}$ in $M_{F,i}^{t(max)}$ weitergeleitet, indem die Gewichtungsfaktoren $c(x)_{F,ij}$ genutzt werden. Die sich ergebenden Aktivierungen

der Stimuli-Neurone werden in einem Vektor gesammelt, der als modifizierter Queryvektor $q_{k(mod)}$ interpretiert und weiter verwendet wird:

$$q_{k(mod)} = (q_{k(mod)j} = z(x)_{F,j}^{t(max)} \mid j = 1, ..., n = \mu_{M(F)}). \quad (399)$$

Dieser modifizierte Queryvektor $q_{k(mod)}$ ist in dem Dokumentvektorraum DVR als Ein-Punkt-Queryvektor interpretierbar, der in N_D^t genau ein Gewinner-Neuron $n_{D,s(1|k)}^t$ und eine Voronoi-Region $R_{D\tilde{}s(1|k)}^t$ spezifiziert (siehe rechte Teilabbildung in Abb. 69)). Im weiteren können die einfachen Retrieval-Strategien angewendet werden, indem z.B. die Dokumentvektoren als Retrieval-Dokumentvektoren-Menge $DVM(q_k^t)$ selektiert werden, die zu den Stimuli $m_{D,s(1|k)j}^t$ gehören, die dem Gewinner-Neuron zugeordnet sind, und die in der Stimulusmenge $M_{D,s(1|k)}^t$ liegen.

3.8.6) Queryvektor-Polyrepräsentation in Merkmalsgraphen

Ein Queryvektor kann durch einen Merkmalsgraphen remodelliert, modifiziert bzw. expandiert werden, indem zunächst Merkmalsvektoren mit vorliegenden Merkmalswerten aus dem Vektor initialisiert werden, die ihre Aktivität auf Merkmalsknoten des Graphen weiterleiten, gefolgt von einer Durchführung einer Aktivitätsausbreitung innerhalb des Merkmalsgraphen. Bei einem deterministischen Modell der Aktivitätsausbreitung erzeugt auf diese Weise ein Queryvektor $q_k = (q_{kj} \mid j = 1, ..., n = \mu_{M(F)})$ genau einen modifizierten Queryvektor $q_{k(mod)} = (q_{k(mod)j} \mid j = 1, ..., n = \mu_{M(F)})$, wenn man von numerischen Instabilitäten der Simmulation absieht, die durch Verfahren des Reliable-Scientific-Computing wie der Verwendung von Intervall-Arithmetiken potentiell beseitigt werden können (Adams & Kulisch (1993[3]), Bauch et al. (1987[30]), Kulisch (1993[192]), Klatte et al. (1993[182])).

Es könnten demgegenüber andere Klassen von Verfahren zur Aktivitätsausbreitung definiert werden, die bei mehrmaligem Input des gleichen Queryvektors unterschiedliche Aktivitätszustände der Merkmalsknoten nach dem Abbruch der Aktivitätsausbreitung liefern. Auf diese Weise könnte damit eine Queryvektor-Polyrepräsentation aus einer Query-Monorepräsentation erzeugt werden. D.h. aus einer Original-Query wird im ersten Schritt durch die Monorepräsentation der Indexierungsfunktion ein Original-Queryvektor q_k, aus dem soll eine Menge von modifizierten Queryvektoren $q_{k,p}$, $p = 1, ..., \delta$ erzeugt werden.

Unterschiedliche Verfahren lassen sich durch unterschiedliche Spezifizierungen der folgenden Verfahrenskomponenten definieren:
1) Preprocessing.
2) Aktivitätsausbreitung.
3) Postprocessing.

Preprocessing ist die Erzeugung von Aktivitäten bzw. Outputwerten in den Merkmals-Stimuli $m_{F,ij}^t = (x_{F,ij}, n_{F,i}, c(x)_{F,ij}, net(x)_{F,ij}^t, z(x)_{F,ij}^t, S(z(x)_{F,ij}^t))$ aus dem zugehörigen skalaren Inputwerten q_{kj} des Queryvektors für alle der $j = 1, ..., \mu_{M(F,i)}^t$ Merkmals-Stimuli, die dem Merkmals-Neuron $n_{F,i}$ zugeordnet sind, und für alle der $i = 1, ..., \mu_{N,F}^t$ Merkmals-Neurone. Der Wert q_{kj} soll als externer Input in ein Stimuli $m_{F,ij}^t$ verwendet werden, das zunächst mit dem für alle Stimuli konstanten externen Gewichtungsfaktor $c(x)_{F,ext}$ den Inputwert $net(x)_{F,ij}^t$ mit Hilfe der Inputfunktion $net(x)_F$ berechnet. Es folgt die

Berechnung der Aktivität $z(x)_{F,ij}^{t}$ mit Hilfe des konstanten Schwellenwertes $T_{F,x}$ und der Aktivitätsfunktion $z(x)_{Fn}$. Im dritten Schritt wird der Outputwert $S(z(x)_{F,ij}^{t})$ des Stimulus mit Hilfe der Outputfunktion $S(z(x)_F)$ gebildet. Der Outputwert aller Stimuli $m_{F,ij}^{t}$, die dem Neuron $n_{F,i}$ zugeordnet sind, wird als skalarer Input in dieses Neuron verwendet, das daraufhin seinen Inputwert, seine Aktivität und seinen Outputwert berechnet. Damit ist die Initialisierung der Phase der Aktivitätsausbreitung beendet, und die ermittelten Outputwerte werden im weiteren an andere Neurone anstatt Stimuli abgegeben, bis ein Kriterium zum Verfahrensabbruch der Aktivitätsausbreitung greift. Das Preprocessing ist somit definiert durch die Funktionen $net(x)_F$, $z(x)_F$ und $S(z(x)_F)$. Eine Queryvektor-Polyrepräsentation kann durch eine Preprocessing-Polyrepräsentation bei konstantem Aktivitätsausbreitungsverfahren durchgeführt werden, indem

1) der Funktionstyp beibehalten wird, und Funktionsparameter wie der Gewichtungsfaktor $c(x)_F$ und der Schwellenwert $T_{F,x}$ bei jeder neuen Präsentation von q_k verändert werden.
2) der Funktionstyp bei jeder neuen Präsentation von q_k verändert wird.

Eine Herangehensweise ist die Definition einer Grundmenge von Funktionsparametern im ersten Fall, bzw. einer Grundmenge von Funktionstypen im zweiten Fall. Soll eine Polyrepräsentation mit δ Elementen erzeugt werden, so können δ Funktions-Tripel $(net(x)_{F,j}, z(x)_{F,j}, S(z(x)_{F,j})$, $j = 1, ..., \delta$, definiert werden. Das Preprocessing wird somit um einen Initialisierungsteil erweitert, bei dem eines dieser Tripel allen $\mu_{M(F)}$ Merkmals-Stimuli $m_{F,ij}$ zugeordnet wird, gefolgt von der Präsentation des betrachteten Queryvektors q_k, bei der jeder Merkmals-Stimulus den ihm zugeordneten Wert q_{kj} als externen Input verarbeitet, und einen Inputwert $net(x)_{F,ij}^{t}$, einen Aktivitätswert $z(x)_{F,ij}^{t}$ und einen Outputwert $S(z(x)_{F,ij}^{t})$ berechnet. Die Outputwerte aller Merkmals-Stimuli, die dem Neuron $n_{F,i}$ zugeordnet sind, werden an dieses Neuron weiter geleitet, und es beginnt die eigentliche Aktivitätsausbreitung bis zu einem Abbruchkriterium, wobei die Input-, Aktivitäts und Outputfunktion der Neurone bei allen δ Durchgängen gleich bleibt. Nach dem Abbruch der Aktivitätsausbreitung wird das Postprocessing durchgeführt, bei dem der Output $S(z_{F,i}^{t(max)})$ des Neurons $n_{F,i}$ den zugehörigen Stimuli $m_{F,ij}^{t}$ zugeführt wird, die ihren Input $net(x)_{F,ij}^{t(max)}$ und ihre Aktivität $z(x)_{F,ij}^{t(max)}$ berechnen. Die Aktivität $z(x)_{F,ij}^{t(max)}$ oder der Output $S(z(x)_{F,ij}^{t(max)})$ eines Stimulus $m_{F,ij}$ kann als modifizierter Wert $q_{k,pj}$ des ursprünglichen Inputwertes q_{kj} in dem modifizierten Queryvektor $q_{k,p}$ verwendet werden. Diese Phasen werden δ mal mit jeweils einem anderen Funktions-Tripel $(net(x)_{F,j}, z(x)_{F,j}, S(z(x)_{F,j})$ für die Merkmals-Stimuli und mit gleichen Funktionen für die Neurone durchgeführt, wobei die sich ergebenden modifizierten Queryvektoren als Queryvektor-Polyrepräsentation im Dokumentvektorenraum verwendet werden.

Die Anzahl der Funktionen bzw. Funktions-Tripel kann auch kleiner als δ gewählt werden, wenn ein Zuordnungsverfahren verwendet wird, das Mehrfachziehungen erlaubt. Z.B. können drei Funktionsmengen $Net(x)_F = \{net(x)_{F,j} \mid j = 1, ...\}$, $Z(x)_F = \{z(x)_{F,j} \mid j = 1, ...\}$ und $S(z(x)_F) = \{S(z(x)_{F,j}) \mid j = 1, ...\}$ gegeben sein, die jeweils eine gleiche Anzahl von Elementen jedoch weniger als δ Elemente besitzen. Bei jedem Durchgang kann durch Ziehen je eines Elementes aus den drei Mengen ein Tripel erzeugt werden, der die Funktionen aller Merkmals-Stimuli bestimmt. Wird ein Ziehen mit Zurücklegen verwendet, so besteht die Möglichkeit, dass ein Tripel mehrmals erzeugt wird, was durch eine Ausschlussprüfung verhindert werden kann, die jedoch einen Prüfaufwand erfordert. Soll auf eine Prüfung verzichtet werden, so kann die Wahrscheinlichkeit des Auftretens eines gleichen Tripels durch eine größere Anzahl von Funktionen in den drei Funktionsmengen verringert werden.

Die bislang vorgestellten Strategien verwenden ein Funktions-Tripel für alle Merkmals-Stimuli. Als Alternative könnte jedem Stimulus $m_{F,ij}$ ein individueller Funktions-Tripel $(net(x)_{F,ij}, z(x)_{F,ij}, S(z(x)_{F,ij}))$ zugeordnet werden, indem eine Ziehung mit Zurücklegen aus den Mengen $Net(x)_F$, $Z(x)_F$ und $S(z(x)_F)$ erfolgt. Auf diese Weise lässt sich die Wahrscheinlichkeit, dass alle Merkmals-Stimuli bei zwei unabhängigen Präsentationen von q_k die gleichen Funktions-Tripel besitzen noch einmal stark verkleinern.

Bei einem symmetrischen Pre- und Postprocessing werden die gleichen Input-, Aktivierungs- und Output-Funktionen verwendet. Diese Voraussetzung muss nicht erfüllt werden, da die externe Zuführung eines Wertes q_{kj} für $m_{F,ij}$ etwas anderes ist, als die interne Zuführung des Outputs $S(z_{F,i}^{t(max)})$ von einem Neuron. Dies gilt insbesondere für die Inputfunktion, die bei der externen Zuführung einen Gewichtsfaktor $c(x)_{F,ij}$ verwendet, der nicht als Funktion einer Distanz zwischen dem Stimulusvektor $x_{F,ij}$ und einem anderen Punkt im Merkmalsvektorenraum, wie dem Gewichtsvektor $w(x_F)_i$, interpretiert werden kann. Wird ein asymmetrisches Pre- und Postprocessing verwendet, so wird ein Funktions-Tripel $(net(x)_{F,extern}, z(x)_{F,extern}, S(z(x)_{F,extern}))$ für das Preprocessing und ein davon unabhängiger Tripel $(net(x)_{F,intern}, z(x)_{F,intern}, S(z(x)_{F,intern}))$ für das Postprocessing definiert, wenn alle Merkmals-Stimuli die gleichen Funktionen verwenden sollen. Werden individuelle Funktions-Tripel zugeordnet, so wird einem individuellen Stimulus $m_{F,ij}$ die beiden Tripel $(net(x)_{F,extern,ij}, z(x)_{F,extern,ij}, S(z(x)_{F,extern,ij}))$ und $(net(x)_{F,intern,ij}, z(x)_{F,intern,ij}, S(z(x)_{F,intern,ij}))$ zugeordnet. Die jeweils korrespondierenden Funktionenstypen können aus der gleichen Grundmenge gezogen werden, d.h. $net(x)_{F,extern,ij}$ und $net(x)_{F,intern,ij}$ können aus $Net(x)_F$ gezogen werden, wobei im Fall der Ziehung mit Zurücklegen ein symmetrisches Pre- und Postprocessing als Spezialfall eingeschlossen ist.

Analog zu der Polyrepräsentation im Rahmen des Pre- und Post-Processing, kann eine Zuordnung von Funktionen für die eigentliche laterale Aktivitätsausbreitung durchgeführt werden, wobei nun die Funktionen für das Pre- und Post-Processing konstant gehalten werden. Hierzu muss eine Menge mit δ Funktions-Listen $(net(w)_{F,j}, z(w)_{F,j}, S(z(w)_{F,j}))$ für die Neurone aus $N_{F,AA}^t$ vorliegen. Bei jeder der δ Präsentationen von q_k wird ein Tripel für alle $\mu_{N,F}$ Neurone aus $N_{F,AA}^t$ verwendet. Alternativ können wiederum drei Mengen $Net(w)_F$, $Z(w)_F$ und $S(z(w)_F)$ vorliegen, wobei vor jeder der δ Präsentationen ein Tripel durch Ziehen je eines der Elemente aus den drei Mengen erzeugt wird. Es können auch individuelle Tripel für jedes Neuron $n_{F,i}$ erstellt werden, indem vor jeder Präsentation für jedes Neuron durch Ziehen aus der Menge der δ Tripel oder durch Ziehen aus den drei Einzelmengen ein Tripel $(net(w)_{F,i}, z(w)_{F,i}, S(z(w)_{F,i}))$ gebildet wird.

Alle bislang dargestellten Verfahren zur Polyrepräsentation zeichnen sich dadurch aus, dass δ Präsentationen und Aktivitätsausbreitungen durchgeführt werden, was zu einem entsprechenden Mehraufwand führt. Der Aufwand einer lateralen Aktivitätsausbreitung ist bestimmt von der Anzahl der Iterationen des Austausches von Outputwerten zwischen verbundenen, benachbarten Neuronen, sodass die Reduktion der Anzahl der Iterationen der wesentliche Faktor zur Verringerung des Aufwandes ist. Im Extremfall könnte genau eine Iteration der Aktivitätsausbreitung durchgeführt werden. Bei einem Szenario wird der Output für alle Merkmals-Stimuli genau einmal berechnet, gefolgt von der Festlegung einer gemeinsamen Input-, Aktivierungs- und Outputfunktion für alle Neurone bzw. individueller Funktionen für jedes Neuron. Die Outputwerte aller Stimuli, die zu einem Neuron gehören, werden an dieses Neuron weiter geleitet, das seinen Inputwert, seine Aktivierung und seinen Outputwert berechnet. Alle Neurone führen

ihren Outputwert den verbundenen Neuronen als Inputwert zu, die ihre Aktivierung und ihren Output neu berechnen, womit genau eine Iteration der Aktivitätsausbereitung beendet ist. Es folgt direkt das Postprocessing, bei dem der Output eines Neurons an die zugehörigen Merkmals-Stimuli weiter geleitet werden, die ihre Aktivierung bzw. ihren Outputwert berechnen.

Einen ähnlichen Ansatz kann man verfolgen, wenn man q_k genau einmal präsentiert und eine Aktivitätsausbreitung über δ Iterationen durchführt. Nach jeder Iteration werden die Outputwerte der Neurone an die zugehörigen Merkmals-Stimuli weitergegeben, und die sich ergebenden Aktivierungen oder Outputwerte der Stimuli werden ausgelesen und als modifizierter Queryvektor $q_{k,p}$ zusammengefasst. Diese Vorgehensweise ist quasi die Nutzung der Historie genau einer Aktivitätsausbreitung, wobei hervorzuheben ist, dass hierbei keine Polyrepräsentation der Input-, Aktivierungs- und Outputfunktion der Merkmale und der Neurone notwendig ist.

Diesen Vorteil besitzt auch eine stochastische Formulierung der Input-, Aktivierungs- und Outputfunktion, d.h. Komponenten dieser Funktionen sind als Zufallsvariablen definiert, die bei jedem Aufruf der Funktion durch ein neues, unabhängiges Zufallsexperiment entsprechend einer gegebenen Zufallsverteilung neu besetzt werden. Dies entspricht der prinzipiellen Sichtweise, dass Wahrscheinlichkeitsfunktionen und Zugehörigkeitsfunktionen eigene Formen der Polyrepräsentation sind.
Eine Stochastisierung der Aktivitätsausbreitung kann auch bei deterministischen Input-, Aktivierungs- und Outputfunktion durchgeführt werden, wenn die Entscheidung einer Ausbreitung stochastisch geregelt wird. In den bislang dargestellten Verfahren zur Aktivitätsausbreitung wird deterministisch davon ausgegangen, dass ein symmetrischer Austausch von Outputwerten zwischen verbundenen Neuronen stattfindet. Demgegenüber wird bei einer stochastischen Ausbreitung jeweils durch einen Zufallsprozess entschieden, ob eine Ausbreitung über eine Verbindungskante zwischen zwei Neuronen durchgeführt werden soll. Die binäre Entscheidung, ob eine Weiterleitung eines Outputwertes durchgeführt werden soll oder nicht, kann insbesondere als Funktion der Distanz der beiden Gewichtsvektoren der verbundenen Neurone modelliert werden, wobei mit zunehmender Distanz die Wahrscheinlichkeit der Weiterleitung kleiner werden soll. Wird die Hin- und die Herleitung von Outputwerten zweier verbundener Neurone unabhängig modelliert, so wird damit eine potentiell asymmetrische Aktivitätsausbreitung beschrieben, da z.B. bei einer bestimmten Iteration eine Übertragung von $n_{F,i}$ zu $n_{F,r}$ durchgeführt wird, nicht jedoch von $n_{F,i}$ zu $n_{F,r}$, was sich jedoch bei der nächsten Iteration wieder umkehren kann.

3.9) Relevanz-Feedback in IRS

Bei einem Relevanz-Feedback handelt es sich um eine Erweiterung im Rahmen des Retrievals, welche die Interaktion des Agenten mit seinen externen Informationsbedürfnissen erfordert. Der normale Retrievalvorgang, d.h. Formulierung einer Frage Q_i als Zeichensequenz, Query-Indexierung $A_{IR(Q)}$ mit dem Ergebnis eines Queryvektors q_i, Ermittlung einer Teilmenge von Dokumentvektoren und Präsentation von Referenzen zu den zugehörigen Dokumenten bzw. direkte Präsentation der Dokumente, wird bei einem Relevanz-Feedback im Prinzip mehrmals durchlaufen. Zwischen jedem dieser Durchgänge steht die Relevanz-Beurteilung der nachgewiesenen Dokumente durch den Agenten in Abhängigkeit von seinen Zielen und Informationsbedürfnissen.

Grundlegende Annahme aller Relevanz-Feedback-Verfahren ist, dass ein IS fehlerhafte und mit Unsicherheiten belegte Repräsentationen besitzt, die durch Relevanz-Bewertungen im Prinzip so verändert werden können, dass Fehler- und Unsicherheiten in der betreffenden Retrievalsituation gesenkt werden können. In der Literatur zum Relevanz-Feedback bei IRS findet man nur Repräsentationsmodifikationen, welche den Queryvektor oder die Dokumentvektoren einzeln bzw. zusammen betreffen (vergl. z.B. Panyr (1987b[252]), Harman (1992[157]), Lundquist et al. (1997[203])). IRS besitzen jedoch weitere Repräsentationsarten, d.h. für eine Modifikation von Repräsentationen stehen prinzipiell folgende Möglichkeiten zur Verfügung, wobei jeweils eine Monorepräsentation unterstellt wird:
1) Query-Indexierungsfunktion $A_{IR(Q)}$.
2) Queryvektor.
3) Dokument-Indexierungsfunktion $A_{IR(D)}$.
4) Dokumentvektoren.
5) Metrik im Dokumentvektorenraum DVR.
6) Metrik im Merkmalsvektorenraum FVR.
7) Retrievalregion.

Bei Polyrepräsentationen kommen weitere Faktoren hinzu, je nach dem Verfahren, durch das die Polyrepräsentation erzeugt wird. Zu erwähnen ist z.B.:
8) (Externe) Polyrepräsentationsfunktion der Query und der Dokumente (Erzeugungsfunktion künstlicher Zeichensequenzen).

Als ein limitierender Faktor bei einem menschlichen Agenten wird seine mentale Bereitschaft gesehen, die Relevanz-Beurteilung der nachgewiesenen Dokumente durchzuführen, da dies das Durcharbeiten der Dokumente bzw. von Dokumentteilen wie Abstract und Zusammenfassung erfordert. Es ist daher wichtig, die Anzahl der präsentierten Dokumente möglichst klein zu halten, was z.B. durch eine Clusterung der ermittelten Dokumentvektoren und der Präsentation der Zentroid-Dokumente erreicht werden kann (siehe Abschnitt 3.9.2.4)). Ein anderer Ansatz sind inkrementelle Verfahren, bei denen pro Feedback-Iteration wenige Dokumente präsentiert werden, demgegenüber aber mehr Iterationen durchgeführt werden, als im Referenzfall, wobei jedoch die Gesamtzahl der zu bewertenden Dokumente kleiner werden soll, als beim Referenzfall,

Allgemein soll jedoch davon ausgegangen werden, dass der Agent eine kognitive Bereitschaft zum Relevanz-Feedback mitbringt, die durch den möglichen Gewinn der Informationsbeschaffung innerhalb seines Problemlösungsprozesses motiviert ist. Die Bereitschaft zum Feedback wird als rationale Entscheidung im Bezug auf das allgemeine „p*G-C“-Kriterium betrachtet (siehe Anderson (1990[8], 1991[9], 1993[10]), siehe auch Abschnitt 1.5)), d.h. wenn der erwartete Gewinn G, der durch die Erstellung der Relevanz-Bewertungen entstehen kann, gewichtet mit der Chance, dass der Gewinn eintritt, größer als die Kosten C geschätzt werden, welche durch die Bewertungen dem Agenten entstehen würden.

Im Rahmen dieses Kapitels wird von einer binären Relevanz-Feedback-Funktion ausgegangen, die jedem Element D_{ij} der Menge $D_i \equiv D(q_i) \equiv D(Q_i)$ der nachgewiesenen Dokumente ein Relevanzwert $rel(D_{ij})$ aus der Menge $\{0, 1\}$ zuordnet. Dieser Wert für ein Dokument wird von dem IRS als Wert für ein

Dokumentvektor übernommen, d.h. das IRS ordnet den Relevanzwert $rel(D_{ij})$ dem korrespondierenden Dokumentvektor als $rel(x_{ij})$ zu. Eine solche Relevanz-Feedback-Funktion wird definiert durch:

$$RF_{IR}: D(\Theta) \rightarrow \{0, 1\}: D_{ij} \mid\rightarrow rel(D_{ij}) \equiv rel(x_{ij}),$$
$$D_{ij} \in D_i \equiv D(q_i) \equiv D(Q_i). \quad (400)$$

3.9.1) Relevanzbegriff und Relevanzproblematik

Der Relevanzbegriff steht unzweifelhaft im Zentrum des Information Retrievals, des Information Filterings und insbesondere des Relevanz-Feedbacks. Es existieren eine Vielzahl von Definitionen und Beschreibungen, die teilweise inkompatibel sind. Es können drei Ansätze bzw. Klassen von Ansätzen unterschieden werden (siehe hierzu Swanson (1987: 19ff[328])):

1) Ähnlichkeits-Relevanz.
2) Problemlösungs-Relevanz.
3) Reformulierungs-Relevanz.

3.9.1.1) Ähnlichkeits-Relevanz

Die Ähnlichkeits-Relevanz beschreibt Relevanz im Sinne von „zum gleichen Thema gehörend“ (Swanson (1987: 23f[328])). Dieser Relevanzbegriff lässt sich durch die Position von Informationsobjekten im DVR modellieren, d.h. Dokumente, die derart indexiert wurden, dass sie in räumlicher Nähe zueinander liegen werden als zum gleichen Thema gehörend betrachtet, worauf alle Cluster-Verfahren im DVR basieren. Die Einführung einer eigenen Variable Ähnlichkeits-Relevanz in den IR-Prozess wird dadurch jedoch fraglich, da sie als monotone Funktion der Distanzen im DVR modelliert werden könnte.

3.9.1.2) Problemlösungs-Relevanz

Wesentlich komplexer ist die Situation bei der Problemlösungs-Relevanz, die versucht den Nutzen zu quantifizieren, die Inhalte von Dokumenten zur Lösung des Problemes besitzen, die den Agenten bewogen hatten, ein externes Informationsbedürfnis zu formulieren. Dieser Ansatz besitzt eine Reihe von Schwierigkeiten, wenn die Relevanz bewertet werden soll, die sich aus der kognitiven bzw. neuro-kognitiven Sichtweise des IR-Prozesses ableiten lassen. Zunächst wird in diesem Rahmen zwischen Relevanz, als einer „möglichst objektiven“ Bewertung eines Inhaltes bezüglich seiner Rolle in einem Problemlösungsprozess, und Pertinenz, als einer subjektiven Bewertung, unterschieden (vergl. auch Panyr (1994: 20f[253])). Als Argument für eine solche Unterscheidung wird angeführt, dass ein Agent Inhalte nicht versteht, nicht integrieren kann, oder schon kennt. Eine solche Unterscheidung könnte gemacht werden, wenn eine Unterscheidung zwischen einer objektiven mentalen Repräsentation und einer subjektiven Repräsentation möglich wäre. Das Problematische ist jedoch die Annahme der Existenz einer objektiven Repräsentation von Inhalten, sowie eines Referenz-Lösungsweges in Form einer Sequenz mentaler Operationen, die eine Problemformulierung als Initialisierung besitzen und zu einer Problemlösung führt. Die Existenz von beidem muss im gegebenen Kontext jedoch abgelehnt werden. Mit objektiver Repräsentation kann höchstens ein Konsens bezüglich einer Fragestellung in einer Gruppe von Individuen bzw.

Experten gemeint sein. Dieser ist jedoch nicht-stationär, da in einem Wissensgebiet ständig neues Wissen hinzukommt, und bekanntes Wissen uminterpretiert und umbewertet wird. Weiterhin ist das Ergebnis abhängig von der ausgewählten Gruppe von Experten, sowie dem Verfahren, mit dem ein Konsens ermittelt wird. Diese durch den Konsens bedingten Einschränkungen gelten in gleicher Weise für die Problemstellung, den Lösungsweg und den Lösungsvorschlag, der als Lösung akzeptiert wird.

Die Fähigkeit eines Agenten Relevanz-Bewertungen abzugeben, bedeutet somit die Fähigkeit eine Gruppe von Experten zu emulieren, was selbst dann schwer vorzustellen ist, wenn er selbst ein Mitglied dieser Gruppe wäre. Die Schätzung des Konsens einer Gruppe durch die Gruppenmitglieder ist zwar Teil des Delphie-Verfahrens (siehe z.B. Fraunhofer-Institut für Systemtechnik und Innovationsforschung (1998[120])), doch wird diese Schätzung durch ein Feedback korrigiert, bei der aus den Einzelschätzungen ein Konsens bzw. Mittelwert gebildet wird, der den einzelnen Gruppenmitgliedern als Entscheidungsgrundlage für ihre nächste Schätzung zur Verfügung gestellt wird. Einem einzelnen Agenten steht diese Feedback-Option jedoch nicht zur Verfügung, sodass er eine entsprechende mentale Schätzoperation zwar durchführen kann, ohne dass aber die Richtigkeit bewertet und korrigiert werden könnte.

Das größte Problem der Bewertung eines Inhaltes, das aus einem Dokument extrahiert wurde, bezüglich eines Lösungsprozesses besteht jedoch darin, dass zum Zeitpunkt der Bewertung nicht genügend Informationen vorliegen, um diese Bewertung zu treffen, da hierzu eine Analyse des Lösungsweges notwendig ist, d.h. der Kenntnis des Ausgangspunktes, aller Zwischenschritte und des Endpunktes, der als Lösung akzeptiert wird. Diese Kenntnis ist jedoch nicht verfügbar, da sich der Agent zu diesem Zeitpunkt erst in einer der Zwischenphasen befindet. Es muss auch der Fall berücksichtigt werden, dass verschiedenen Inhalten zunächst eine positive Relevanz zugeordnet wird, dass sich der Problemlösungsprozess jedoch dadurch in eine Region eines lokalen Optimums bewegt, das für eine endgültige Lösung nicht gut genug ist. Zu dem Zeitpunkt, in dem erkannt wird, dass es sich um ein nicht ausreichend gutes Optimum handelt, müssten die zurückliegenden Positionen und Inhalte neu bewertet werden, wobei sich die Frage stellt, wie dies geschehen soll. Lokal betrachtet haben die Inhalte eine positive Relevanz, mittelfristig haben sie sich jedoch als irreführend herausgestellt, und wie sie am Ende des Lösungsprozesses neu bewertet werden, kann während des Lösungsprozesses nicht entschieden werden. Bei einem menschlichen Agenten kommt hinzu, dass die Analyse eines ganzen oder eines Teils eines Problemlösungsprozesses ausschließlich mit dem Instrument der Introspektion durchgeführt werden muss. Dieses Instrument besitzt jedoch enge Grenzen, d.h. viele mentale End- und Zwischenzustände sowie Operationen sind mit der Introspektion nicht erfassbar (siehe z.B. Metzinger (1996[218])).

Die Relevanzbeurteilung von externen Beiträgen für den internen Problemlösungsprozess kann innerhalb zwei Situationen beschrieben werden:
1) Situation ohne Zeit-Constraints.
2) Situation mit Zeit-Constraints.

In der Situation ohne Zeit-Constraints kann der Agent die nachgewiesenen Dokumente genau und ohne zeitliche Restriktionen analysieren. Insbesondere kann er die Texte selbst und nicht nur Teile wie Titel und Abstract analysieren. Er kann verschiedene Versuche zum Textverstehen und zur Integration der

gewonnenen Informationen in die mentalen Strukturen seines oder seiner Problemlösungsvorschläge unternehmen. Bei mehreren Dokumenten können die einzelnen Dokumente mit einer unterschiedlichen Intensität analysiert werden, was auch eine mehrmalige Analyse und unterschiedliche Reihenfolgen der Analyse umfassen kann. Bei dieser Vorgehensweise werden Informationen aus dem betreffenden Dokument faktisch extrahiert, im Gegensatz zu der Situation beim Vorliegen von Zeit-Constraints. Hier können nur Textteile wie Titel, Abstract oder Zusammenfassung analysiert werden, d.h. der Text wird nur patiziell analysiert. Weiterhin kann die Analyse selbst nur grob durchgeführt werden, d.h. es werden nur wenige semantische Strukturen aus den ausgewählten Textteilen erzeugt. Durch die patizielle und grobe Analyse wird faktisch eine Schätzung des Inhaltes des Textes erzeugt, bzw. eine Schätzung des Wissens, das dem Agenten zugänglich ist, was als Wissenschätzung bezeichnet werden kann.

Nach der Extraktion von Wissen aus einem Dokument bei der Situation ohne Zeit-Constraints kann dieses Wissen in die mentalen Zustände eingebaut werden, die einen Problemlösungsvorschlag repräsentieren, d.h. das faktisch vorliegende Wissen führt zu einer faktischen Veränderung der mentalen Strukturen. Bei der Situation mit Zeit-Constraints liegt nur ein patielles und grobes Wissen vor, das möglicherweise nicht, nur teilweise oder falsch in die bestehenden mentalen Strukturen eingefügt werden kann. Die Integration erfordert selbst einen Zeitaufwand, der beim Vorliegen von Zeit-Constraints überschritten werden kann, sodass dem Agenten nur die Möglichkeit verbleibt, zu schätzen, wie sich seine mentalen Strukturen verändern würden, wenn er das ihm zugängliche Wissen aus dem betrachteten Dokument in seine mentalen Strukturen integrieren würde. Faktisch handelt es sich um einen iterativen, zweistufigenSchätzprozess, da bei Zeit-Constraints das zugängliche Wissen und die Wirkung auf die mentalen Strukturen geschätzt werden muss. Diese Schätzoperationen besitzen direkte Analogien zu den Operationen, die bei dem Spezialfall „Optimal-Experiment-Design“ des Aktiven Lernens auftreten (Fedorov (1972[113]), Cohn (1995[73]), Cohn et al. (1995[74]), siehe Abschnitt 5.2.2)). An dieser Stelle sei erwähnt, dass zum einen der Outputwert f(x)^ für einen Stimuluskandidaten (x, _) geschätzt werden muss, zum anderen wird das Approximationsmodell simmuliert, das sich ergeben würde, wenn der Stimuluskandidat aufgenommen und das Modell aktualisiert würde.

Erzeugt ein System bezüglich eines oder mehrerer seiner Subsysteme ein Modell, so wird dies allgemein als Selbstmodell bezeichnet (siehe z.B. Metzinger (1993[217])). Schätzt ein Agent das ihm wahrscheinlich zugängliche Wissen eines Dokumentes z.B. anhand des Abstraktes, so erfordert dies eine Form von Selbstmodell bezogen auf die mentalen Strukturen betrachtet werden, die dem Agenten in seinem Problemlösungsareal momentan zur Verfügung stehen. Dieses Selbstmodell wird in einem nachfolgenden Schritt verwendet, um die veränderungen der mentalen Strukturen im Problemlösungsareal durch die neuen, geschätzten Wissensstrukturen zu simmulieren.

Unabhängig ob Veränderungen der mentalen Strukturen faktisch durchgeführt oder nur geschätzt werden, im weiteren muss die Bewertung der neuen Struktur erfolgen, so wie beim Optimal-Experiment-Design die Qualität eines neuen Approximationsmodells geschätzt werden muss, um zu entscheiden, ob ein Stimuluskandidat aufgenommen werden soll. Der Agent muss die Frage beantworten, welchen potentiellen Nutzen sein neues, simmuliertes oder faktisches mentales Modell im Hinblick auf das zu lösende Problem besitzen kann.

Zusammengenommen müssen bei einer Relevanzbewertung maximal die folgenden mentalen Schätzungsoperationen durchgeführt werden:
1) Schätzung des zugänglichen Wissens.
2) Schätzung des neuen mentalen Modells durch Integration des zugänglichen Wissens.
3) Schätzung des Nutzens des neuen Modells bezüglich des zu lösenden Problems.

Fasst man die Argumente bezüglich der Problemlösungs-Relevanz zusammen, so gelangt man zu dem Schluss, dass zum einen jegliche Relevanz-Beurteilung nur sinnvoll im Bezug zu einem individuellen Agenten und seinen momentanen subjektiven mentalen Zuständen ist (siehe auch Swanson (1987: 23[328])), sodass eine Unterscheidung zwischen Relevanz und Pertinenz nicht aufrecht erhalten werden kann. Zum anderen, dass eine Problemlösungs-Relevanz weder durch den Agenten durch Introspektion noch durch eine externe Instanz wie das IRS während des Problemlösungsprozesses valide operationalisiert werden kann.

3.9.1.3) Modellierung des Problemlösungsprozesses durch einen Zustandsraum

Wird Information Retrieval im Kontext eines Problemlösungsprozesses durch einen Agenten gesehen, so muss vorausgesetzt werden, dass eine hinreichende Beschreibung von Kriterien vorliegt, mit denen die Qualität eines Problemlösungsvorschlages bestimmt werden kann (Bewertungs- bzw. Fitnessfunktion; s.a. Bachelier (1999d[22])), sowie ein Abbruchkriterium, mit dem der Agent entscheiden kann, ob einer der Lösungsvorschläge hinreichend gut ist, um den Suchprozess zu beenden. Das Abbruchkriterium kann auch extern durch eine Instanz in der (sozialen) Umgebung des Agenten angesiedelt sein, wenn es sich um eine externe Problemstellung handelt.

Der Suchprozess nach einer hinreichenden Lösung lässt sich modellieren durch einen mentalen, n-dimensionalen Zustandraumes, in dem ein mentaler Zustand wie z.B. ein Problemlösungsvorschlag ein Punkt einnimmt. Der Zustandraum wird als eine Metapher bezeichnet, da keine hinreichenden Informationen vorliegen, um die einzelnen Dimensionen eines mentalen Zustandraum zu spezifizieren. Denkbar wäre beispielsweise die Modellierung einer mentalen Struktur durch eine Symbolstruktur in Form eines Graphen wie bei verschiedenen Formen des Genetic-Programming (Passing-Tree; siehe Koza (1992[188], 1994[189]), Koza et al. (1999[191]), Banzhaf et al. (1998[26]), Nordin (1997[238])). In einem solchen Fall genügt es die symbolischen Basisstrukturen zu definieren, und die Dimension des Zustandraumes ergibt sich aus der maximal zulässigen Größe der betrachteten Graphen.

Durch die Bewertungskriterien kann jedem Problemlösungsvorschlag ein Qualitätsmaß zugeordnet werden, wobei vereinfachend angenommen werden soll, dass es sich um genau einen Wert, im Gegensatz zu einem Bewertungsvektor bei einer Mehr-Ziel-Bewertung, handeln soll. Auf diese Weise entsteht ein (n+1)-dimensionaler bewerteter Zustandsraum, in dem potentielle Lösungen als lokale Maxima vorliegen. Je nach der Zielsetzung soll eine hinreichend gute Lösung gefunden werden, oder eine Menge alternativer Lösungen, mit einer Bewertung größer-gleich einem Schwellenwert.

Bei einer Monorepräsentation der Lösungsvorschläge liegt zu einem Zeitpunkt genau ein Vorschlag vor, auf den Operationen einwirken, die bewirken, dass ein anderer Punkt im Zustandsraum eingenommen

wird, was gleichbedeutend mit einem anderen Lösungsvorschlag ist. Verursacht wird diese Transformation durch Veränderungen der Struktur des Lösungsvorschlages, wie Deduktion oder die Integration von externem Wissen, das z.B. durch eine Analyse eines Textes oder einer Graphik in das mentale Gesamtsystem aufgenommen wurde.

Der Problemlösungsvorgang wird in dieser Sichtweise als diskrete Trajektorie von Punkten in dem mentalen Zustandsraum betrachtet, ausgehend von einem Initialisierungszustand, über eine Menge von Zwischenzuständen bis ein Punkt erreicht wird, dem eine hinreichend gute Bewertung zugeordnet wird, damit er als akzeptierte Lösung verwendet wird.

In einem solchen Modell können verschiedene Ansätze zur Spezifizierung des Relevanzbegriffes erprobt werden, die von der Bewertungsfunktion abgeleitet werden. Werden zwei nachfolgende Punkte in der Trajektorie betrachtet, so kann vereinfachend angenommen werden, dass extern aufgenommene Informationen zu einer Restrukturierung der mentalen Strukturen geführt hat, d.h. durch die Integration der Informationen ist der erste in den zweiten Zustand übergegangen. Dies ist in mehrfacher Hinsicht eine Vereinfachung, da nicht spezifiziert ist, ob alle extern aufgenommenen Informationen oder nur ein Teil integriert wurden. Weiterhin könnten auch Informationen, die schon vor einiger Zeit aufgenommen wurden, erst jetzt integriert worden sein. Wird jedoch diese Vereinfachung unterstellt, so kann die Relevanz einer Information durch ihre Wirkung festgelegt werden als Funktion der Veränderung der Bewertung, die durch den Übergang eines mentalen Zustandes in den nachfolgenden Zustand ergibt. D.h. ist die Bewertung des zweiten Zustandes größer als des ersten, so ist deren Differenz positiv, und die Relevanz der verursachenden Information ist ebenfalls positiv. Ist die Bewertung jedoch kleiner, so ist die Differenz negativ, und die Relevanz kann ebenfalls negativ werden, wenn sie z.B. auf [-1, 1] definiert ist.

Fasst man die Betrachtungen einer solchen Zustandsraum-Sichtweise zusammen, so wird die Problemlösungs-Relevanz von externen Beiträgen auf die Bewertung von Problemlösungsvorschlägen zurück geführt. Existieren jedoch konzeptionelle Probleme bei der Bewertung von Problemlösungsvorschlägen, so besitzt diese Vorgehensweise keinen Vorteil gegenüber einer direkten Relevanzbeurteilung. Vorteile besitzt diese Sichtweise bei einer algorithmischen Modellierung eines Problemlösungsprozesses als einem Optimierungsprozess z.B. durch ein Verfahren der Evolutionären Programm-Induktion wie dem Genetic-Programming, wenn eine konkrete Modifikation bewertet werden soll, da die Struktur vor der Modifikation sowie nach der Modifikation bekannt ist, und für beide ein Fitnesswert bestimmt werden kann, was der Bewertung des Problemlösungsvorschlages entspricht.

3.9.1.4) Reformulierungs-Relevanz

Der dritte Ansatz, in dem die Relevanz definiert wird, und der zu der Reformulierungs-Relevanz führt, ist die Betrachtung des IR-Prozesses als Versuchs-Irrtums-Prozess (Swanson (1987:20ff[328])). Der Ausgangspunkt ist der gleiche wie bei der Problemlösungs-Relevanz, d.h. IR wird im Rahmen eines Problemlösungsprozesses durch einen Agenten betrachtet, wobei der Unterschied darin besteht, dass entsprechend den Ansichten von Karl Popper zur wissenschaftlichen Theoriebildung in einer offenen Welt (Popper (1994[262], 1995[263])) alle Problemlösungsprozesse als Versuchs-Irrtums-Prozesse betrachtet werden (siehe auch Generate & Test bei Newell (1990[233])). Die Erzeugung neuer Theorien

durch Zufallsprozesse aus vorangegangenen Konzepten und Theorien spezifiziert jedoch nicht das Verfahren, mit dem dies geschieht, wobei in Swanson (1987[328]) nicht deutlich darauf hingewiesen wird, dass zufällige Veränderungen nicht gleichbedeutend sind mit gleichverteilt zufälligen Veränderungen. Zufällige Veränderungen werden durch stationäre bzw. nicht-stationäre Verteilungsfunktionen festgelegt, die beliebige Formen besitzen können, während eine gleichverteilt zufällige Veränderung eine spezielle Verteilungsfunktion der Form y = b, b = konstant, darstellt. D.h. es darf eine Spezifizierung einer Funktionsklasse nicht mit der übergeordneten Menge an Funktionsklassen gleichgesetzt werden. Zur Beschreibung eines Phänomens existieren meist mehrere konkurrierende Theorien, sodass die Entwicklung allgemein als populationsbasiertes Zufallsverfahren modelliert werden kann. Dadurch wird neben einer ungeschlechtlichen mutativen Veränderung einer einzelnen Theorie auch die Möglichkeit der rekombinativen Erzeugung neuer Theorien aus mehreren älteren Theorien eröffnet.

Wird der IR-Prozess als Versuchs-Irrtums-Prozess betrachtet, so liegt der Wert eines Dokumentes nicht in einem direkten Nutzen, sondern ergibt sich indirekt aus der Reformulierung der mentalen Struktur, der Problembeschreibung, einer Theorie oder der extern formulierten Frage (Swanson (1987:21[328])). Bei den Reformulierungen können Dokumente, die bezüglich der Ähnlichkeits-Relevanz als nicht-relevant eingestuft werden, genauso wertvoll sein wie als relevant eingestufte Dokumente. Diese Überlegung wird beim Queryvektor-Relevanz-Feedback, unabhängig von der Interpretation als Versuchs-Irrtums-Prozess genutzt, wenn bei einer gemischten Relevanz-Feedback-Strategie die als relevant bewerteten Dokumentvektoren ein positives Feedback und die als nicht-relevant bewerteten Dokumentvektoren ein negatives Feedback erzeugen (siehe Abschnitt 3.9.2.1)). Eine Einschränkung ergibt sich jedoch, indem alle präsentierten Dokumentvektoren nach Ansicht des IRS entsprechend der Ähnlichkeits-Relevanz als potentiell relevant eingestuft werden, d.h. es werden keine Dokumente präsentiert, von denen das IRS erwartet, dass sie nicht-relevant sind.

Die Reformulierung mentaler Strukturen des Agenten kann im Rahmen dieser Arbeit nicht betrachtet werden, sondern nur externalisiertes Verhalten, wie die Reformulierung einer Problembeschreibung und insbesondere der Query, die durch das IRS neu indexiert wird. Eine Sequenz aus reformulierten Queries und den abgeleiteten Queryvektoren liefert die Grundlage zu Verfahren, welche die Reformulierungs-Relevanz operationalisieren (siehe Abschnitt 3.9.7)).

3.9.1.5) Irrelevant vs. irreführend

Ein Aspekt der Relevanzproblematik im Kontext des Problemlösens betrifft die Interpretation der Bewertung „nicht relevant“ eines Informationsobjektes. Diese Bewertung kann im Sinne von irrelevant als nicht zu dem Thema gehörend interpretiert werden (Ähnlichkeits-Relevanz, siehe Abschnitt 3.9.1.1)), oder im Sinne von irreführend. Irreführende Informationsobjekte gehören zu dem betrachteten Themenbereich, führen jedoch dazu, dass der Agent durch die Aufnahme und Integration des Wissens in dem Informationsobjekt dazu geführt wird, einen Problemlösungsversuch zu unternehmen, der von einer hinreichend guten Lösung wegführt, oder zu einer lokalen, nicht hinreichend guten Lösung führt, deren Überwindung problematisch wird.

Eine Bewertung „irrelevant“ im Sinne von nicht zu einem Thema gehörend, kann durch eine binäre Kodierung erfolgen, indem die Alternative „0“ aus der Menge {0, 1} gewählt wird. Bei einer reellen Relevanzbewertung würde aus dem Intervall [0, 1] ein Wert ausgewählt, wobei „0“ wiederum den Pool irrelevant darstellt. Eine Bewertung „irreführend“ könnte im Rahmen einer dreiwertigen Kodierung als „-1“ aus {-1, 0, 1} bzw. als linke Grenze aus dem Intervall [-1, 1] gewählt werden. In beiden Fällen könnte Null als irrelevant und 1 als relevant kodiert werden.

Während die Entscheidung, ob ein Informationsobjekt zu einem Themenbereich gehört, direkt nach der Analyse des Objektes durch den Agenten durchgeführt werden kann, ist eine Entscheidung bezüglich der Irreführung zu diesem Zeitpunkt nicht möglich. Im einfachsten Fall muss der Agent zunächst mit Hilfe des Informationsobjektes einen neuen Problemlösungsversuch erzeugen und diesen bewerten, sodass er eine Aussage über die Irreführung bzw. den Grad der Irreführung machen kann. Effektiv reicht eine solche lokale Betrachtung jedoch nicht aus, da letztendlich erst nach dem Erreichen des Ziels, d.h. des ersten hinreichend guten Lösungsvorschlages, und nach der Analyse der Wege und Irrwege, die zu diesem Lösungsvorschlag geführt haben, eine Bewertung bezüglich der Irreführung von Einzelbeiträgen abschließend durchgeführt werden kann.

Doch selbst die Zugehörigkeit zu Themen ist im Kontext des Problemlösens nicht unkritisch, da Lösungen innerhalb eines Problembereichs von Lösungen in einem anderen, mental weit entfernten Problembereich inspiriert werden können, sodass Informationsobjekte, die nicht zu einem Thema gehören, durchaus relevant bezüglich eines Problems in diesem Themenbereich sind.

Die Problematik der Überwindung von lokalen, nicht hinreichenden Lösungen hängt mit der Irreversibilität der Integration von Wissenselementen aus Informationsobjekten in mentalen Modelle zusammen. Es existiert keine Möglichkeit, einen Reset einer Integrationsoperation durchzuführen, d.h. eines bewussten Vergessens eines Elementes bei einem menschlichen Agenten. Dies ist auch nicht sinnvoll, da die Bildung einer mentalen Schleife verhindert werden muss, d.h. ein Prozess muss dabei rückgängig gemacht werden, und es muss repräsentiert werden, warum er rückgängig gemacht wurde, um den gleichen Fehler nicht nochmals zu machen. Es muss eine Form von mentalem Backtracking unterstellt werden, bei der zum einen ein Zustand erzeugt wird, der vor der Integration eines Wissensobjektes in das mentale Modell bestanden hat, und zum anderen muss für den nächsten Schritt brücksichtigt werden, dass die Integration des betreffenden Objektes zu einem unerwünschten Zustand geführt hat.

3.9.2) Queryvektor-Relevanz-Feedback

3.9.2.1) Queryvektor-Feedback bei unklassifizierten Dokumentvektoren

Die Mehrzahl aller Verfahren zum Relevanz-Feedback betrifft die Modifikation des Queryvektors, wobei unterstellt wird, dass die Umsetzung des Informationsbedürfnisses des Agenten in eine Query fehlerhaft und mit Unsicherheiten behaftet sei. Diese Annahme ist in Anbetracht der Sequenz „Informationsbedürfnis -> Query -> Queryvektor“ jedoch nicht zwingend, da ein Queryvektor, der als nicht adäquat betrachtet wird, ebenso durch eine nicht adäquate Indexierungsfunktion $A_{IR(Q)}$ gebildet werden kann. Konsequenter wäre unter dieser Annahme eine Modifikation der Query anstatt des Queryvektors, doch

ein solcher Ansatz wurde bislang nicht versucht. Im Rahmen dieses Abschnittes soll die Modifikation eines Objektvektos bei konstanter $A_{IR(Q)}$ betrachtet werden.

Der Original-Queryvektor, der durch die Indexierung der Original-Query Q_i erzeugt wurde, soll mit $q_i^{t=0}$ bezeichnet werden, wobei der Index t die Iterationstiefe des Feedback-Verfahrens angibt. Entsprechend der Standardvorgehensweise bei einer Query- und Dokumentvektor-Mono-Repräsentation wird mit dem Queryvektor $q_i^{t=0}$ eine Retrievalmenge $DVM(q_i^{t=0})$ von Dokumentvektoren spezifiziert, die hier mit $x_{k|q(i)}^{t=0}$ bezeichnet werden sollen, und die z.B. in einer ε-Umgebung um $q_i^{t=0}$ liegen. Im nächsten Schritt wird aus der Dokumentvektorenmenge eine geordnete Liste erzeugt, wobei als Ordnungskriterium in der Iteration t=0 der Abstand zwischen Query- und Dokumentvektor verwendet wird:

$$DV(q_i^{t=0}) = (x_{j|q(i)}^{t=0} \mid d_{DVR}(q_i^{t=0}, x_{j|q(i)}^{t=0}) < d_{DVR}(q_i^{t=0}, x_{j+1|q(i)}^{t=0}) < \varepsilon,\ j = 1, ..., f^{t=0}). \quad (401)$$

Die Dokumente, die zu den Dokumentvektoren $x_{j|q(i)}^{t=0}$ gehören, werden dem Agenten vorgelegt, der jedem Element aus $DV(q_i^{t=0})$ einen Relevanzwert $rel(x_{j|q(i)}^{t=0})$ zuordnen soll.

Im Rahmen der Darstellungen in diesem Abschnitt wird der Fall betrachtet, dass die Relevanzfunktion rel(x) binär ist d.h.

$$rel(x) \in \{0, 1\}. \quad (402)$$

Von dieser Annahme geht die Standardliteratur zur Suchfragemodifikation beim Relevanz-Feedback aus (vergl. z.B. Panyr (1987b[252])), während im Kapitel 4) Relevanz-Approximationsmodelle betrachtet werden, die von reellen Relevanzfunktionen mit $rel(x) \in [0, 1]$ oder alternativ $rel(x) \in [-1, 1]$ ausgehen.

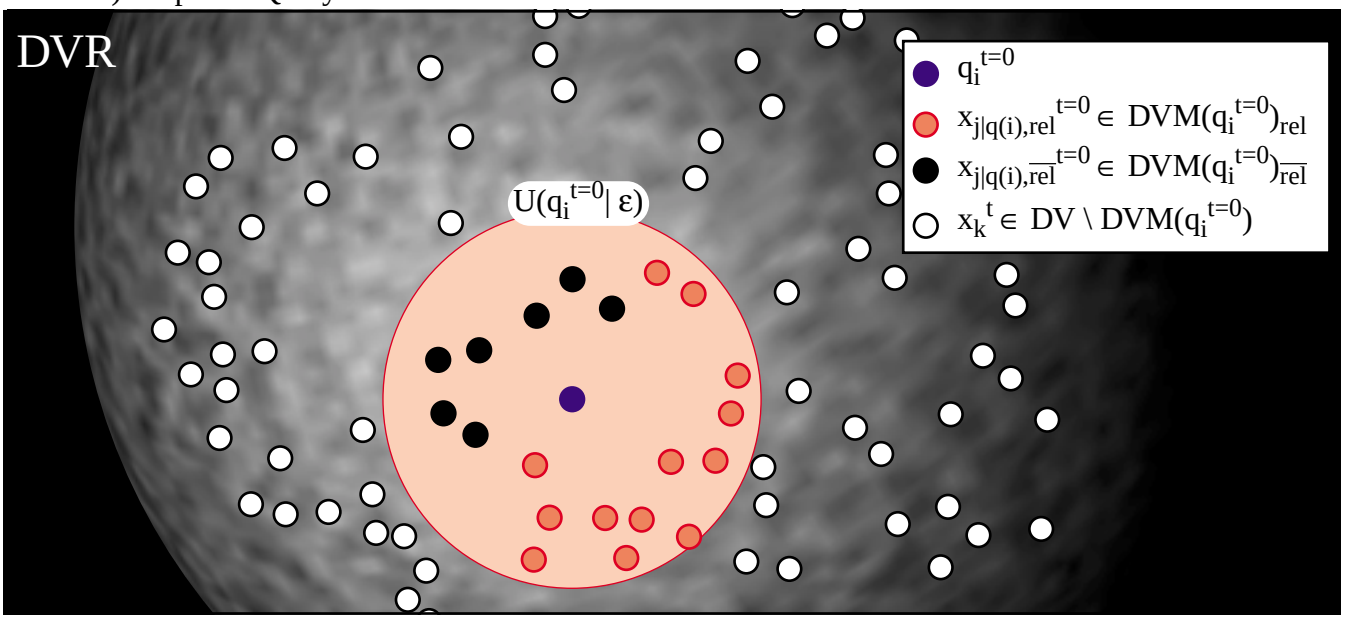
Durch die binäre Relevanzbewertung wird die Retrievalmenge $DVM(q_i^{t=0})$ zerlegt in eine Menge $DVM(q_i^{t=0})_{rel}$ von relevanten Dokumentvektoren und eine Menge $DVM(q_i^{t=0})_{\overline{rel}}$ von nicht-relevanten Dokumentvektoren (siehe Abb. 70)), wobei gelten soll, dass beide Mengen nicht-leer sind:

$$\begin{aligned} DVM(q_i^{t=0})_{rel} &= \{x_{j|q(i),rel}^{t=0} \mid rel(x_{j|q(i),rel}^{t=0}) = 1,\ j = 1, ..., f_{rel}^{t=0}\}, \\ DVM(q_i^{t=0})_{\overline{rel}} &= \{x_{j|q(i),\overline{rel}}^{t=0} \mid rel(x_{j|q(i),\overline{rel}}^{t=0}) = 0,\ j = 1, ..., f_{\overline{rel}}^{t=0}\}, \\ f_{rel}^{t=0} + f_{\overline{rel}}^{t=0} &= f^{t=0},\ f_{rel}^{t=0} \neq 0 \wedge f_{\overline{rel}}^{t=0} \neq 0. \end{aligned} \quad (403)$$

Mit Hilfe dieser beiden Mengen von Dokumentvektoren soll der Original-Queryvektor $q_i^{t=0}$ zu $q_i^{t=1}$ modifiziert werden. Wird $q_i^{t=0}$ als Ausgangspunkt verwendet, so ergibt sich ein besserer Queryvektor, wenn $q_i^{t=0}$ in Richtung der relevanten Dokumentvektoren verschoben wird, und wenn $q_i^{t=0}$ weg von den irrelevanten Dokumentvektoren verschoben wird. Mit Hilfe der arithmetischen Zentroid-Vektoren $\bar{s}_{DVM(i,rel)}^{t=0}$ und $\bar{s}_{DVM(i,\overline{rel})}^{t=0}$ der beiden Mengen $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ lässt sich dieser Vorgang in einem einzelnen, aggregierten Adaptionsprozess beschreiben als (Salton (1971[293]), Panyr (1987b[252])):

$$\begin{aligned} q_i^{t=1} &= \alpha * q_i^{t=0} + \beta * \bar{s}_{DVM(i,rel)}^{t=0} - \chi * \bar{s}_{DVM(i,\overline{rel})}^{t=0},\ \text{mit } \alpha, \beta, \chi \geq 0, \\ \bar{s}_{DVM(i,rel)}^{t=0} &= 1/f_{rel}^{t=0} * \sum_j x_{j|q(i),rel}^{t=0},\ \forall x_{j|q(i),rel}^{t=0} \in DVM(q_i^{t=0})_{rel}, \\ \bar{s}_{DVM(i,\overline{rel})}^{t=0} &= 1/f_{\overline{rel}}^{t=0} * \sum_j x_{j|q(i),\overline{rel}}^{t=0},\ \forall x_{j|q(i),\overline{rel}}^{t=0} \in DVM(q_i^{t=0})_{\overline{rel}}. \end{aligned} \quad (404)$$

Abb. 70) Einpunkt-Queryvektor mit binär bewerteten Retrieval-Dokumentvektoren

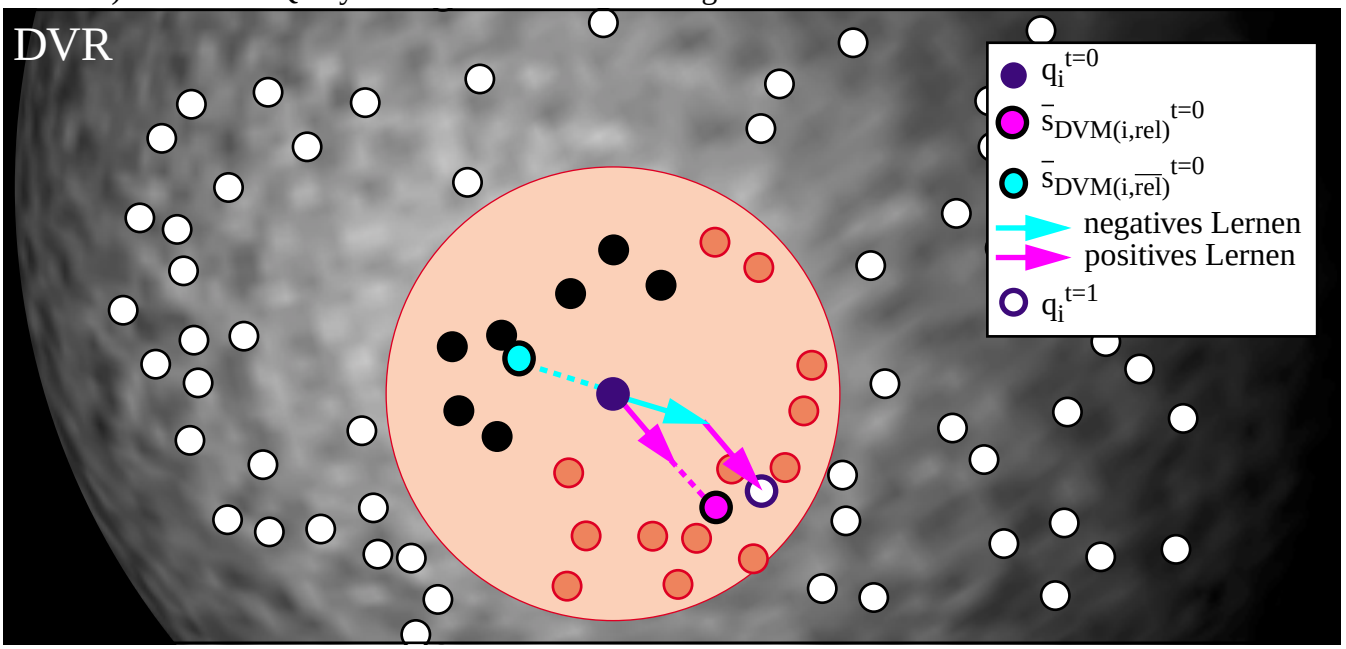


Allgemein lässt sich dies durch eine Relevanz-Feedback-Funktion rfb(.) beschreiben, die den vorangegangenen Queryvektor, die relevanten und nicht-relevanten Dokumentvektoren und die Parameter als Input entgegennimmt, und den neuen Queryvektor als Output liefert:

$$q_i^{t=1} = rfb(q_i^{t=0}, DVM(q_i^{t=0})_{rel}, DVM(q_i^{t=0})_{\overline{rel}}, \alpha, \beta, \chi), \text{ oder}$$
$$q_i^{t=1} = rfb(q_i^{t=0}, \bar{s}_{DVM(i,rel)}^{t=0}, \bar{s}_{DVM(i,\overline{rel})}^{t=0}, \alpha, \beta, \chi). \quad (405)$$

Eine solche Querymodifikation kann als Lernprozess in einer unüberwachten Umgebung interpretiert werden, analog dem Lernprozess beim Competitive-Learning (siehe z.B. Schöneburg et al. (1990: 101ff[302]), Bachelier (1995:25ff[14])) oder speziell bei den SOMs (Ritter et al. (1991[282]), Kohonen (1989[184], 1995[185]), Bachelier (1998a[15])).

Abb. 71) Gemischte Query-Relevanz-Feedback-Strategie



Die Verschiebung in Richtung eines Stimulusvektors wird als positive Lernoperation, und die Verschiebung weg von einem Stimulus wird als negativer Lernprozess bezeichnet (siehe Abschnitt 3.9.2.2)). D.h. die oben dargestellte allgemeine Form der iterativen Querymodifikation kann als eine Kombination bzw. Sequenz eines positiven und eines negativen Lernprozesses interpretiert werden, sodass sie als gemischte Relevanz-Feedback-Strategie bezeichnet werden kann (siehe Abb. 71)).

Demgegenüber kann eine positive Relevanz-Feedback-Strategie unterschieden werden, bei der $\chi := 0$ gesetzt wird, d.h. die als nicht-relevant bewerteten Dokumentvektoren werden nicht einbezogen, sodass ausschließlich eine positive Lernoperation durchgeführt wird.

Eine rein negative Relevanz-Feedback-Strategie, bei der $\beta := 0$ gesetzt würde, wird demgegenüber isoliert nicht eingesetzt, es sei denn, dass bei einer Feedback-Iteration nur negative Beispiele gefunden werden, d.h. wenn $DVM(q_i^{t=0})_{rel}$ leer ist. Dieser Fall ist jedoch keine bewusste Entscheidung für eine rein negative Feedback-Strategie, sondern ein Spezialfall der gemischten Strategie.

Nachdem eine neue Query $q_i^{t=1}$ erzeugt wurde, wird für diese, entsprechend dem verwendeten Retrievalverfahren, eine neue Retrieval-Dokumentvektorenmenge $DVM(q_i^{t=1})$ erzeugt. Die dazu korrespondierenden Dokumente werden dem Agenten vorgelegt, der sie in relevante und nicht relevante Dokumente klassifiziert, sodass die Mengen $DVM(q_i^{t=1})_{rel}$ und $DVM(q_i^{t=1})_{\overline{rel}}$ vorliegen, aus denen ein weiterer Queryvektor gebildet wird. Je nach der Wahl der Parameter α, β, χ bei der gemischen Strategie bzw. α, β bei der positiven Strategie, könnte der Queryvektor über die Grenze der ε-Umgebung $U(q_i^{t=0} \mid \varepsilon)$ hinaus verschoben werden. Diese Retrieval-Region ist jedoch die Region in DVR, in der die Relevanzbewertungen in der t'ten Iteration durchgeführt wurden, sodass dies als sinnvolles Kriterium interpretierbar ist, eine Gesamtverschiebeoperation auf die Retrievalregion einer Iteration zu begrenzen.

Die vorgestellte gemischte Relevanz-Feedback-Strategie, die standardmäßig in Queryvektor-Feedback-Verfahren verwendet wird, basiert auf den beiden künstlichen Fixpunkten der Adaption $\bar{s}_{DVM(i,rel)}^{t=0}$ und $\bar{s}_{DVM(i,\overline{rel})}^{t=0}$, die für eine positive und eine negative Einzel-Adaption verwendet werden. Aus Effizienzgründen ist es sicherlich günstig, einen einzigen Fixpunkt für die Adaptionsprozesse zu besitzen, doch es ist auch möglich keine aggregierten, künstlichen Fixpunkte, sondern die vorliegenden natürlichen Fixpunkte zu verwenden, d.h. die einzelnen Dokumentvektoren in den beiden Mengen $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$.

Dies führt zu einer Adaptionsgleichung, bei der $f_{rel}^{t=0}$ Summanden die positive Adaption und $f_{\overline{rel}}^{t=0}$ Summanden die negative Adaption beschreiben. Zu klären ist dabei die Rolle der Parameter, da bei der Verwendung konstanter Parameter β bzw. χ für alle Summanden eines Typs, diese ausklammerbar sind, sodass sich letztendlich wieder die obige gemischte Strategie ergibt. D.h. bei einer Adaptionsgleichung, die auf den $f_{rel}^{t=0} + f_{\overline{rel}}^{t=0}$ natürlichen Fixpunkten der Adaption basiert, müssen individuelle Parameter β_j bzw. χ_j als Gewichtungsfaktoren verwendet werden. Es muss ein Prinzip festgelegt werden, wie diese individuellen Parameter erzeugt werden sollen, wobei die folgenden beiden naheliegend sind:
1) Parameterwerte als Funktion der Distanz zu dem Queryvektor.
2) Parameterwerte als Funktion der Relevanzwerte bei rellem Relevanz-Feedback.

Da in diesem Kapitel ausschließlich binäre Relevanzbewertungen betrachtet werden sollen, wird eine monoton fallende Parameterfunktion unterstellt, denn mit zunehmender Distanz eines Dokumentvektors soll sein Einfluss auf die Adaption des Queryvektors kleiner werden. Normiert sind die Adaptionsparameter durch den Umgebungsparamter ε von $U(q_i^{t=0} \mid \varepsilon)$, da bei der betrachteten Retrieval-Strategie die Distanzen nur kleiner-gleich als ε sein können. D.h. eine solche Funktion besitzt eine Definitionsmenge von [0, ε], die z.B. auf eine Bildmenge [0, 1] abgebildet wird:

$$[0, \varepsilon] \rightarrow [0, 1]: d_{DVR}(q_i^{t=0}, x_j) \mid\rightarrow \beta_j \text{ bzw. } \chi_j. \tag{406}$$

Hierbei zeigt sich eine Beziehung zu den SOM-Adaptionsverfahren, da als eine einfache Parameterfunktion in Abhängigkeit von der Distanz die Delta-Funktion $\Delta q_i^{t=0} = \varepsilon * d_{DVR}(q_i^{t=0}, x_j)$ gewählt werden kann (siehe Abschnitt 3.9.2.2)). Für die Queryvektor-Adaption bedeutet dies die Verwendung von $f_{rel}^{t=0}$ Summanden für eine positive Adaption und $f_{\overline{rel}}^{t=0}$ Summanden für eine negative Adaption:

$$q_i^{t=1} = \alpha * q_i^{t=0} + \sum_j \beta_j * x_{j|q(i),rel}^{t=0} - \sum_j \chi_j * x_{j|q(i),\overline{rel}}^{t=0},$$
$$\forall x_{j|q(i),rel}^{t=0} \in DVM(q_i^{t=0})_{rel} \text{ und } \forall x_{j|q(i),\overline{rel}}^{t=0} \in DVM(q_i^{t=0})_{\overline{rel}}. \tag{407}$$

Die bislang vorgestellten Adaptionsformen bei einem Queryvektor-Feedback können als deterministische Einzel-Adaption oder als Aggregation von unabhängigen elementaren Adaptions-Operationen interpretiert werden, je nachdem ob im ersten Fall ein künstlicher Fixpunkt der Adaption oder im zweiten Fall eine Menge natürlicher Fixpunkte verwendet wird. Demgegenüber stehen Adaptionsverfahren, die wie im zweiten Fall die natürlichen Fixpunkte verwenden, jedoch abhängige Adaptions-Operationen durchführen, d.h. es wird wie bei den SOM-Lernverfahren eine Sequenz von elementaren Adaptions-Operationen durchgeführt, die abhängig sind, da das Ergebnis einer Operation als Ausgangspunkt der nachfolgenden Adaption betrachtet wird. Eine solche Adaptions-Operation besteht in der gleichverteilt zufälligen Ziehung mit Zurücklegen eines Elementes aus $DVM(q_i^{t=0})$, gefolgt von der Ermittlung, ob es sich um einen relavanten oder nicht relevanten Dokumentvektor handelt. Im ersten Fall wird eine positive Adaption mit einem individuellen Gewichtungsfaktor β_j durchgeführt

$$q_i^{t=0}(neu) = \alpha * q_i^{t=0}(alt) + \beta_j * x_{j|q(i),rel}^{t=0}, \tag{408}$$

während im zweiten Fall eine negative Adaption mit einem individuellen Gewichtungsfaktor χ_j durchgeführt wird.

$$q_i^{t=0}(neu) = \alpha * q_i^{t=0}(alt) - \chi_j * x_{j|q(i),\overline{rel}}^{t=0}. \tag{409}$$

Nach einer bestimmten Anzahl dieser Einzel-Adaptionen, die eine Funktion der Anzahl der nachgewiesenen Dokumentvektoren in $DVM(q_i^{t=0})$ ist, wird der sich ergebende Vektor als Queryvektor $q_i^{t=1}$ der nachfolgenden Iteration t=1 verwendet, mit dem eine neue Ergebnis-Dokumentvektorenmenge $DVM(q_i^{t=1})$ aus der Restmenge $DV \setminus DVM(q_i^{t=0})$ ermittelt wird. Ist diese zum ersten Mal leer, so ist dies ein Abbruchkriterium von Seiten des IRS, während der Agent nach jeder Dokumentpräsentation abbrechen kann.

3.9.2.2) SOM-Adaption beim Queryvektor-Feedback

Es ist auch eine SOM-Adaption anwendbar, um einen neuen Queryvektor auf der Basis der beiden Mengen $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ zu modifizieren. Ausgegangen wird von der SOM-Adaptionsgleichung $w_i^{neu} = w_i^{alt} + \Delta w_i^{alt}$, die einen positiven Adaptionsprozess beschreibt , bei der w_i^{alt} in Richtung eines Adaptionsfixpunktes wie x_k^t um den Betrag Δw_i^{alt} verschoben wird. Bei einer GNG-SOM-Adaptionsgleichung wird eine vereinfachte Form mit einem Lernparameter ε und der Distanz zwischen w_i^{alt} und dem Adaptionsfixpunkt verwendet, sodass sich ergibt

$$w_i^{neu} = w_i^{alt} + \Delta w_i^{alt}, \text{ mit } \Delta w_i^{alt} = \varepsilon * d(w_i^{alt}, x_k^t). \tag{410}$$

Eine negative Adaption unterscheidet sich dadurch, dass der zu adaptierende Vektor weg von einem Adaptionsfixpunkt verschoben wird, was sich durch das negative Vorzeichen des Deltavektors zeigt:

$$w_i^{neu} = w_i^{alt} - \Delta w_i^{alt}, \text{ mit } \Delta w_i^{alt} = \varepsilon * d(w_i^{alt}, x_k^t). \tag{411}$$

Im weitern sollen folgende Verfahren vorgestellt werden:
1) Einmalige Adaption durch Zentroid-Vektoren $\bar{s}_{DVM(i,rel)}^{t=0}$ und $\bar{s}_{DVM(i,\overline{rel})}^{t=0}$.
2) Mehrmalige Adaption durch Ziehung von Dokumentvektoren aus $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$.

Die einmalige Adaption verwendet im Prinzip den Ansatz einer gemischten Adaption, d.h. die beiden arithmetischen Zentroid-Vektoren $\bar{s}_{DVM(i,rel)}^{t=0}$ und $\bar{s}_{DVM(i,rel)}^{t=0}$ der Mengen $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ dienen als Adaptionsfixpunkte einer positiven und einer negativen Adaption. Bei einer sequentiellen Durchführung wird der Queryvektor $q_i^{t=0}$ zunächst in Richtung $\bar{s}_{DVM(i,rel)}^{t=0}$ um den Betrag $\Delta_{rel} q_i^{t=0}$ verschoben, wodurch $q_i^{t=1}$ erzeugt wird:

$$q_i^{t=1} = q_i^{t=0} + \Delta_{rel} q_i^{t=0}, \text{ mit } \Delta_{rel} q_i^{t=0} = \varepsilon_{rel} * d_{DVR}(q_i^{t=0}, \bar{s}_{DVM(i,rel)}^{t=0}). \tag{412}$$

Danach wird die negative Adaption durchgeführt, bei der $q_i^{t=1}$ von $\bar{s}_{DVM(i,\overline{rel})}^{t=0}$ um den Betrag $\Delta_{\overline{rel}} q_i^{t=0}$ weg verschoben wird, wobei $q_i^{t=1,}$ entsteht:

$$q_i^{t=1,} = q_i^{t=1} - \Delta_{\overline{rel}} q_i^{t=0}. \tag{413}$$

Dabei stellt sich die Frage, ob bei dem $\Delta_{\overline{rel}}q$-Wert die Distanz zwischen dem Queryvektor vor der ersten Modifikation und dem Fixpunkt oder nach der Modifikation verwendet werden soll, d.h.:

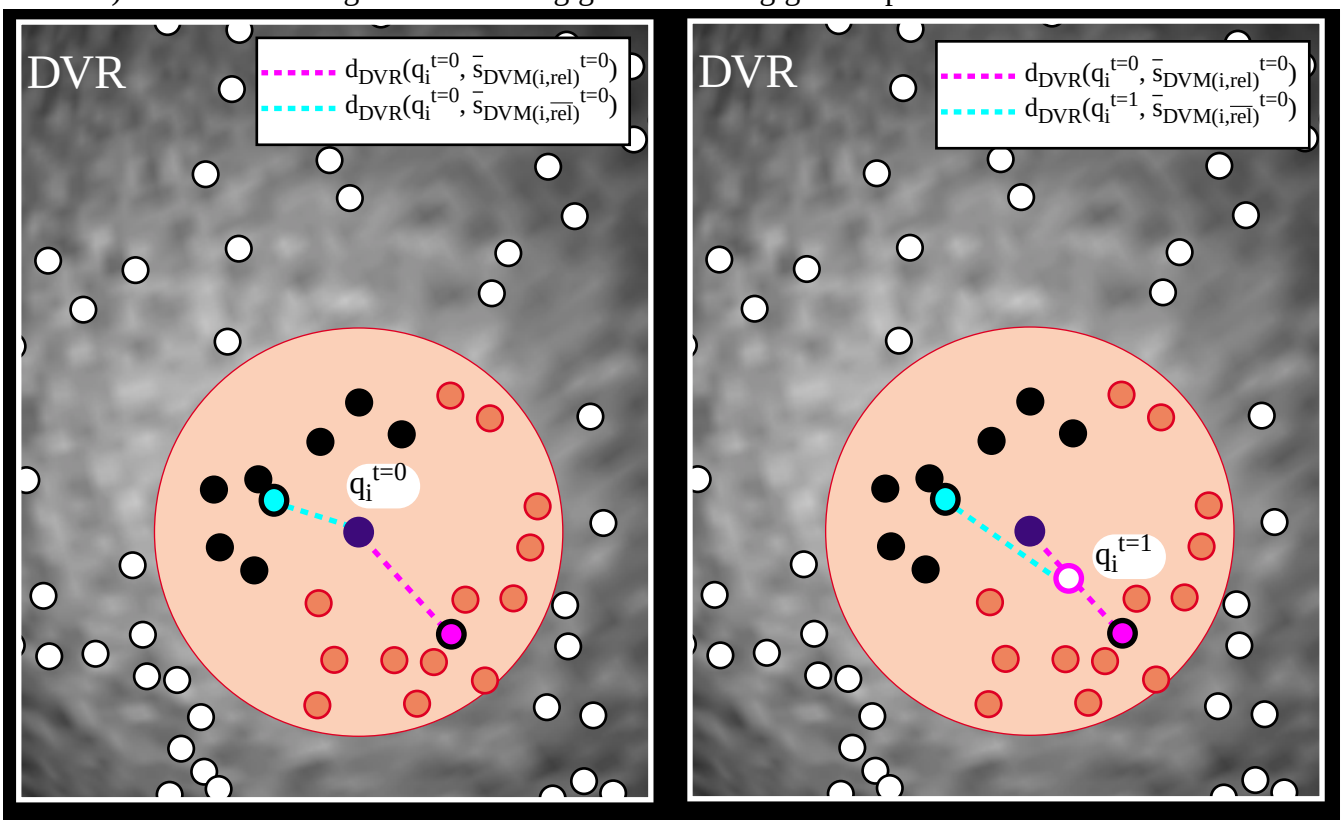
$$\Delta_{\overline{rel}} q_i^{t=0} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{t=0}, \bar{s}_{DVM(i,\overline{rel})}^{t=0}) \text{ oder}$$
$$\Delta_{\overline{rel}} q_i^{t=0} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{t=1}, \bar{s}_{DVM(i,\overline{rel})}^{t=0}). \tag{414}$$

Im ersten Fall wird eine parallele, unabhängige Adaption unterstellt, während der zweite Fall eine sequentielle, abhängige Adaption darstellt (siehe Abb. 72)). Bei einer abhängigen Adaption wird somit auch die Reihenfolge der Adaptionsprozesse relevant, d.h. eine positive und negative Adaption unterscheidet sich von einer negativen und positiven Adaption, da der Zwischenzustand in beiden Fällen unterschiedlich ist, sodass auch die Distanz zu dem anderen Fixpunkt und somit der zweite Δq-Wert unterschiedlich sind.

D.h. wird $q_i^{+-t=1}$‘ als das Ergebnis einer positiven Adaption $ad^+(q_i^{t=0})$ und einer darauf folgenden negativen Adaption $ad^-(q_i^{+t=1})$ betrachtet, so ist dieser Vektor ungleich dem Vektor $q_i^{-+t=1}$‘, als einer negativen Adaption $ad^-(q_i^{t=0})$ und einer darauf folgenden positiven Adaption $ad^+(q_i^{-t=1})$:

$$q_i^{+-t=1}‘ = ad^-(q_i^{+t=1}) = ad^-(ad^+(q_i^{t=0})) \neq q_i^{-+t=1}‘ = ad^+(q_i^{-t=1}) = ad^+(ad^-(q_i^{t=0})), \text{ mit}$$
$$q_i^{+t=1} = ad^+(q_i^{t=0}) = q_i^{t=0} + \Delta_{rel}q_i^{t=0}, \text{ mit } \Delta_{rel}q_i^{t=0} = \varepsilon_{rel} * d_{DVR}(q_i^{t=0}, \bar{s}_{DVM(i,rel)}^{t=0}),$$
$$ad^-(q_i^{+t=1}) = q_i^{+t=1} - \Delta_{\overline{rel}}q_i^{+t=1}, \text{ mit } \Delta_{\overline{rel}}q_i^{+t=1} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{+t=1}, \bar{s}_{Q+-(i,\overline{rel})}^{t=0}),$$
$$q_i^{-t=1} = ad^-(q_i^{t=0}) = q_i^{t=0} - \Delta_{\overline{rel}}q_i^{t=0}, \text{ mit } \Delta_{\overline{rel}}q_i^{t=0} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{t=0}, \bar{s}_{DVM(i,\overline{rel})}^{t=0}),$$
$$ad^+(q_i^{-t=1}) = q_i^{-t=1} + \Delta_{rel}q_i^{-t=1}, \text{ mit } \Delta_{rel}q_i^{-t=1} = \varepsilon_{rel} * d_{DVR}(q_i^{-t=1}, \bar{s}_{DVM(i,rel)}^{t=0}). \quad (415)$$

Abb. 72) Distanzbeziehungen bei unabhängiger und abhängiger Adaption



Da kein Kriterium ersichtlich ist, das die Reihenfolge einer positiven und negativen Adaption sinnvoll festlegen kann, soll eine parallele, unabhängige Adaption durchgeführt werden, wobei der modifizierte Queryvektor $q_i^{t=1}$ entsteht:

$$q_i^{t=1} = q_i^{t=0} + \Delta_{rel}q_i^{t=0} - \Delta_{\overline{rel}}q_i^{t=0}, \text{ mit}$$
$$\Delta_{rel}q_i^{t=0} = \varepsilon_{rel} * d_{DVR}(q_i^{t=0}, \bar{s}_{DVM(i,rel)}^{t=0}), \text{ und } \Delta_{\overline{rel}}q_i^{t=0} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{t=0}, \bar{s}_{Q+-(i,\overline{rel})}^{t=0}). \quad (416)$$

Bei der mehrmaligen Adaption werden Elemente aus $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ mit Zurücklegen gezogen, die als Fixpunkte für eine positive bzw. negative Adaption verwendet werden. Um die potentiell unterschiedliche Anzahl der Elemente in den beiden Mengen zu berücksichtigen, wird bei einer Adaptionsoperation aus der Gesamtmenge $DVM(q_i^{t=0})$ ein Element $x_{j,L}^{t=0}$ gezogen. Ist es Element von $DVM(q_i^{t=0})_{rel}$, so wird es als $x_{j,L,rel}^{t=0}$ bezeichnet und dient als Fixpunkt einer positiven Adaption:

$$q_i^{t=0}(neu) = q_i^{t=0}(alt) + \Delta q_i^{t=0}(alt), \text{ mit}$$
$$\Delta q_i^{t=0}(alt) = \varepsilon_{adapt,rel} * d_{DVR}(q_i^{t=0}(alt), x_{j,L,rel}^{t=0}). \quad (417)$$

Stammt es aus $DVM(q_i^{t=0})_{\overline{rel}}$, dann wird es als $x_{j,L,\overline{rel}}^{t=0}$ bezeichnet und ist Fixpunkt einer negativen Adaption:

$$q_i^{t=0}(neu) = q_i^{t=0}(alt) - \Delta q_i^{t=0}(alt), \text{ mit}$$
$$\Delta q_i^{t=0}(alt) = \varepsilon_{adapt,\overline{rel}} * d_{DVR}(q_i^{t=0}(alt), x_{j,L,\overline{rel}}^{t=0}). \quad (418)$$

Die Ziehung und Adaption wird sequentiell und zufällig durchgeführt, wobei die Anzahl eine Funktion der Anzahl der Elemente in $DVM(q_i^{t=0})$ ist. Nach dem Abbruch der Adaptionsphase wird der sich ergebende Queryvektor als Grundlage für die nächste Iteration t=1 des Queryvektor-Relevanzfeedbacks verwendet, d.h. $q_i^{t=0}(neu)$ der letzten Adaptionsoperation vor dem Abbruch wird als $q_i^{t=1}$ verwendet. Mit der Umgebung $U(q_i^{t=1} \mid \varepsilon)$ liefert dieser Queryvektor die nächste Dokumentvektoren-Ergebnismenge $DVM(q_i^{t=1})$ aus der Restmenge $DV \setminus DVM(q_i^{t=0})$, deren korrespondierende Dokumente dem Agenten zur Relevanzbewertung präsentiert werden.

Durch die Zufallsauswahl von Elementen aus einer Dokumentvektoren-Ergebnismenge innerhalb einer Iteration des Queryvektor-Feedbacks, ist die mehrmalige Adaption eine Form von stochastischer Adaption, da die sich letztendlich ergebende Position des adaptierten Queryvektors nur durch Simulation der Ziehungs- und Adaptionsprozesse ermittelt werden kann, wobei verschiedene Durchläufe jeweils abweichende Endpositionen liefern. Dies gilt auch, wenn die eigentliche Adaptions-Operation vollständig deterministisch ist. Im Gegensatz zu der vorgestellten positiven bzw. negativen deterministischen GNG-SOM-basierten Adaptions-Operation existieren stochastische Adaptionsprozesse, von denen Beispiele, die ebenfalls auf der GNG-SOM-Adaption basieren, im nachfolgenden Abschnitt beschrieben werden.

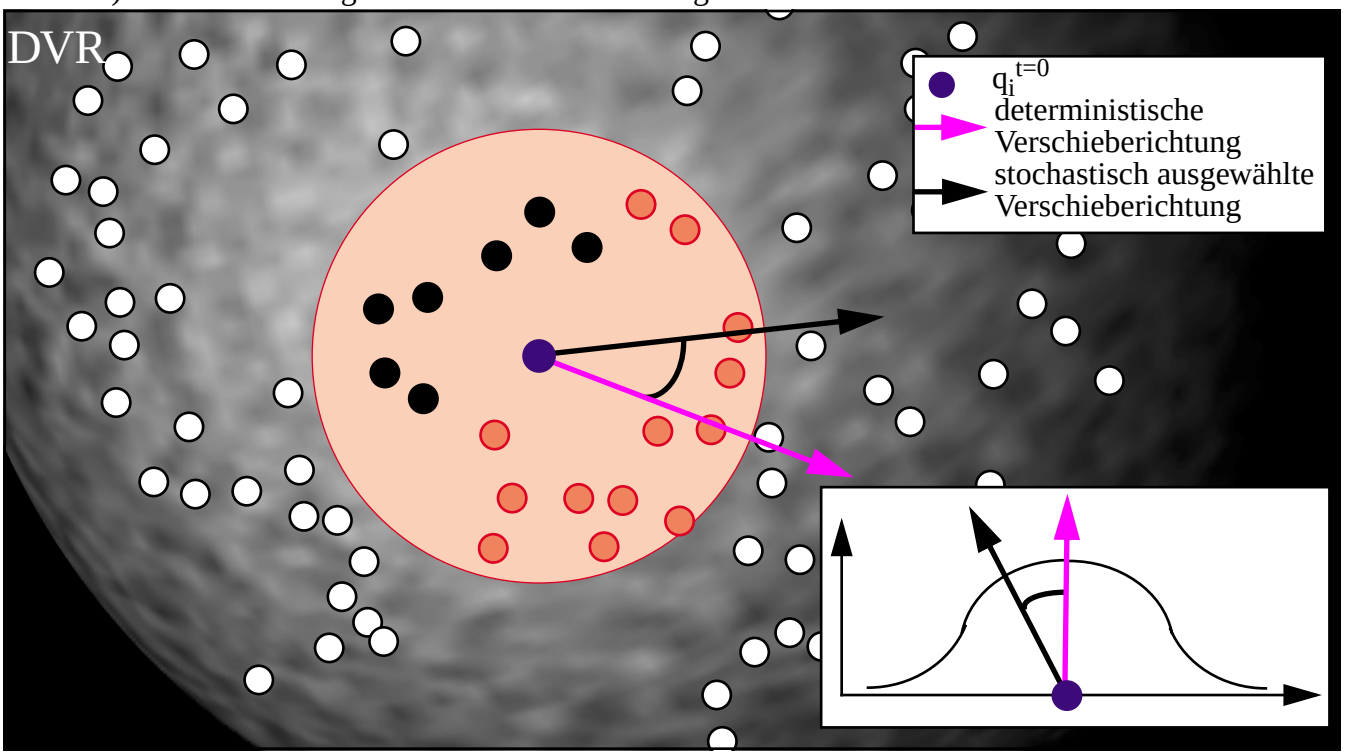
3.9.2.3) Stochastische Adaptions-Operationen beim Queryvektor-Feedback

Das dargestellte Verfahren des kombinierten positiven und negativen Lernens ist ein rein deterministisches Verfahren, zu dem nicht-deterministische Varianten existieren. Im Rahmen der Darstellungen des exploitativen und explorativen Wachstums eines GNG-Graphen (Bachelier (1999d:66ff[22])), wurden stochastische Verschiebeoperationen beschrieben, die auch in diesem Kontext eingesetzt werden können. Grundlage ist die Überlegung, dass eine Verschiebung durch eine Verschieberichtung und eine Verschiebestärke oder -distanz beschrieben wird. Ein stochastisches Verfahren liegt somit vor bei einer:
1) stochastischen Verschieberichtung und einer deterministischen Verschiebedistanz.
2) deterministischen Verschieberichtung und einer stochastischen Verschiebedistanz.
3) stochastischen Verschieberichtung und einer stochastischen Verschiebedistanz.

Die stochastischen Operationen sind jeweils durch eine Verteilungsfunktion spezifiziert. Eine stochastische Verschieberichtung kann aus der deterministischen Verschieberichtung gebildet werden, indem die deterministische Richtung das Maximum der Häufigkeit der Verteilung beschreibt, um das beispielsweise radialsymmetrisch eine Verteilungsfunktion mit monoton fallenden Werten gebildet wird (siehe Abb. 73)).

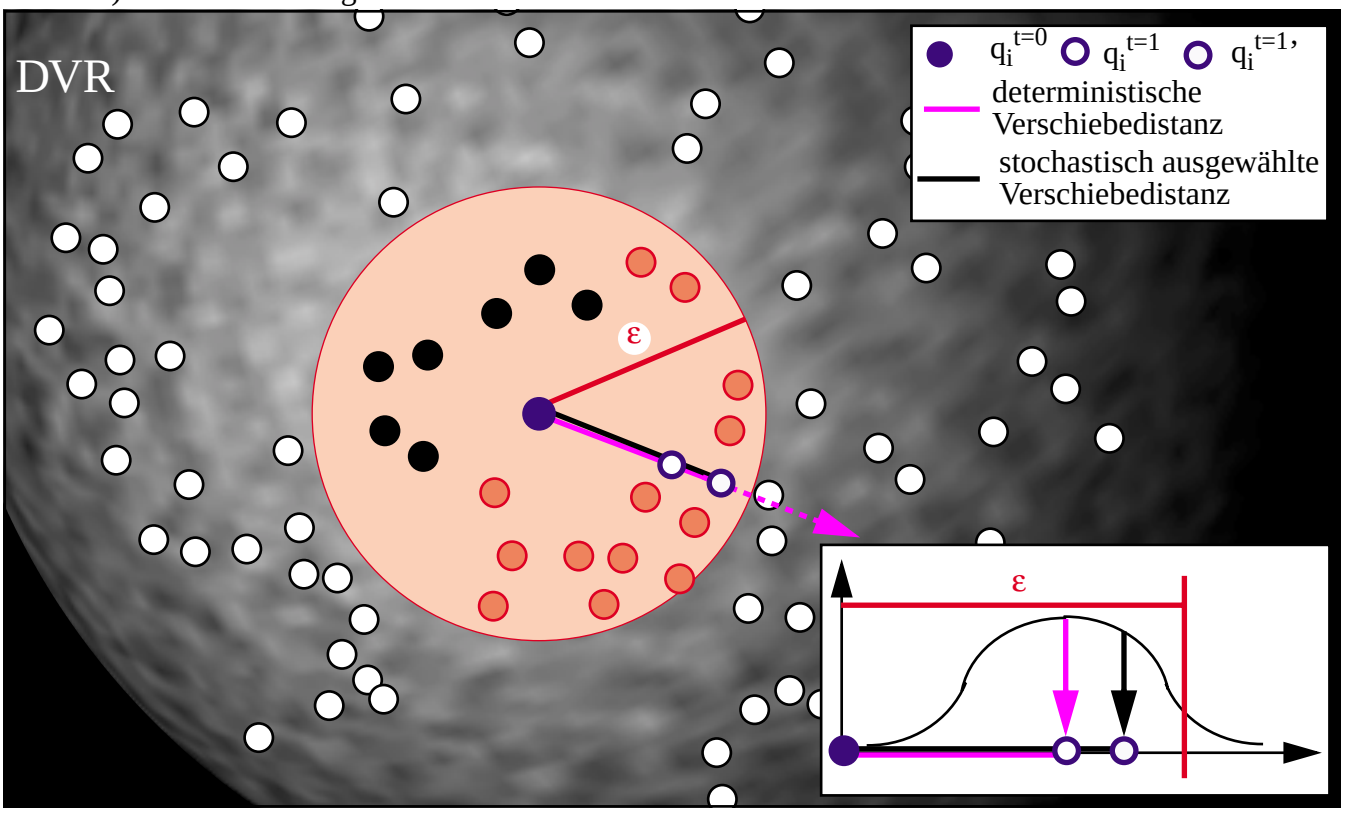
Die Richtung in einer Dimension kann durch einen positiven oder negativen Winkel bezüglich der deterministischen Verschieberichtung angegeben werden, sodass eine stochastische Richtung in einem n-dimensionalen Dokumentvektorenraum DVR durch einen Vektor mit n Winkeln vollständig beschrieben werden kann.

Abb. 73) Stochastisch ausgewählte Verschieberichtung



Eine stochastische Verschiebedistanz kann aus der deterministischen Verschiebedistanz gebildet werden, indem die deterministische Distanz das Maximum der Häufigkeit der Verteilung beschreibt, um das beispielsweise radialsymmetrisch eine Verteilungsfunktion mit monoton fallenden Werten gebildet wird (siehe Abb. 74)). D.h. die Position $q_i^{t=1}$, die durch die deterministische Vorgehensweise erzeugt wird, entspricht dem Häufigkeitsmaximum der Verteilungsfunktion. Durch ein Zufallsexperiment wird in einem konkreten Fall in Abhängigkeit von der Verteilungsfunktion eine Distanz ausgewählt, die größer oder kleiner als die deterministische Distanz sein kann, wobei als Maximum der Radius ε der ε-Umgebung um $q_i^{t=0}$ entsprechend der oben dargestellten Argumentation verwendet werden soll. Im Gegensatz zu einer stochastischen Verschiebungsrichtung, die in DVR n Parameter durch jeweils ein Zufallsexperiment festlegen muss, kann eine stochastische Distanz durch genau ein Zufallsexperiment festgelegt werden.

Abb. 74) Stochastisch ausgewählte Verschiebedistanz



3.9.2.4) Post-Retrieval-Operationen beim Queryvektor-Feedback

Post-Retrieval-Operationen sind Operationen, die nach der Ermittlung der Retrieval-Dokumentvektorenmenge durchgeführt werden, jedoch vor der Präsentation der selektierten Dokumente und der Bewertung durch den Agenten.

Eine offensichtlich notwendige Post-Retrieval-Operation ist die Bereinigung der Retrieval-Dokumentvektorenmengen im Verlauf der Feedback-Iterationen. Liegen in $DVM(q_i^{t=1})$ Dokumentvektoren, die bereits in $DVM(q_i^{t=0})$ vorhanden waren, so ist es sinnvoll, die korrespondierenden Dokumente nicht noch einmal zur Relevanzbewertung vorzulegen, da sie bereits bewertet wurden. Aus verschiedenen Gründen könnte dies von Nutzen sein, indem z.B. ermittelt werden soll, ob sich die Bewertungskriterien des Agenten aufgrund der Integration von neuem Wissen bzw. der Umstrukturierung von bekanntem Wissen während des Interaktionszeitraumes verändert haben, worauf hier jedoch nicht näher eingegangen werden soll.

Anstatt jede Iteration auf der Gesamtmenge DV der Dokumentvektoren ablaufen zu lassen, kann $DVM(q_i^{t=k})$ auf DV ohne die Dokumentvektoren gebildet werden, die in den vorangegangenen Iterationen nachgewiesen und bewertet wurden:

$$DVM(q_i^{t=k}) = \{x_{j|q(i)}^{t=k} \mid x_{j|q(i)}^{t=k} \in U(q_i^{t=k}|\,\varepsilon) \wedge x_{j|q(i)}^{t=k} \in DV \setminus \bigcup_{l=1->k-1} DVM(q_i^{l})\}. \qquad (419)$$

Diese Überlegung geht von der Konstanz der Bewertungsfunktion des Agenten während des Relevanz-Feedback-Verfahrens aus, d.h. die Funktion rel(x) wird als stationär betrachtet, sodass die Reihenfolge der Präsentation von Dokumenten innerhalb der Sequenz der Feedback-Iterationen irrelevant ist. Dies ist

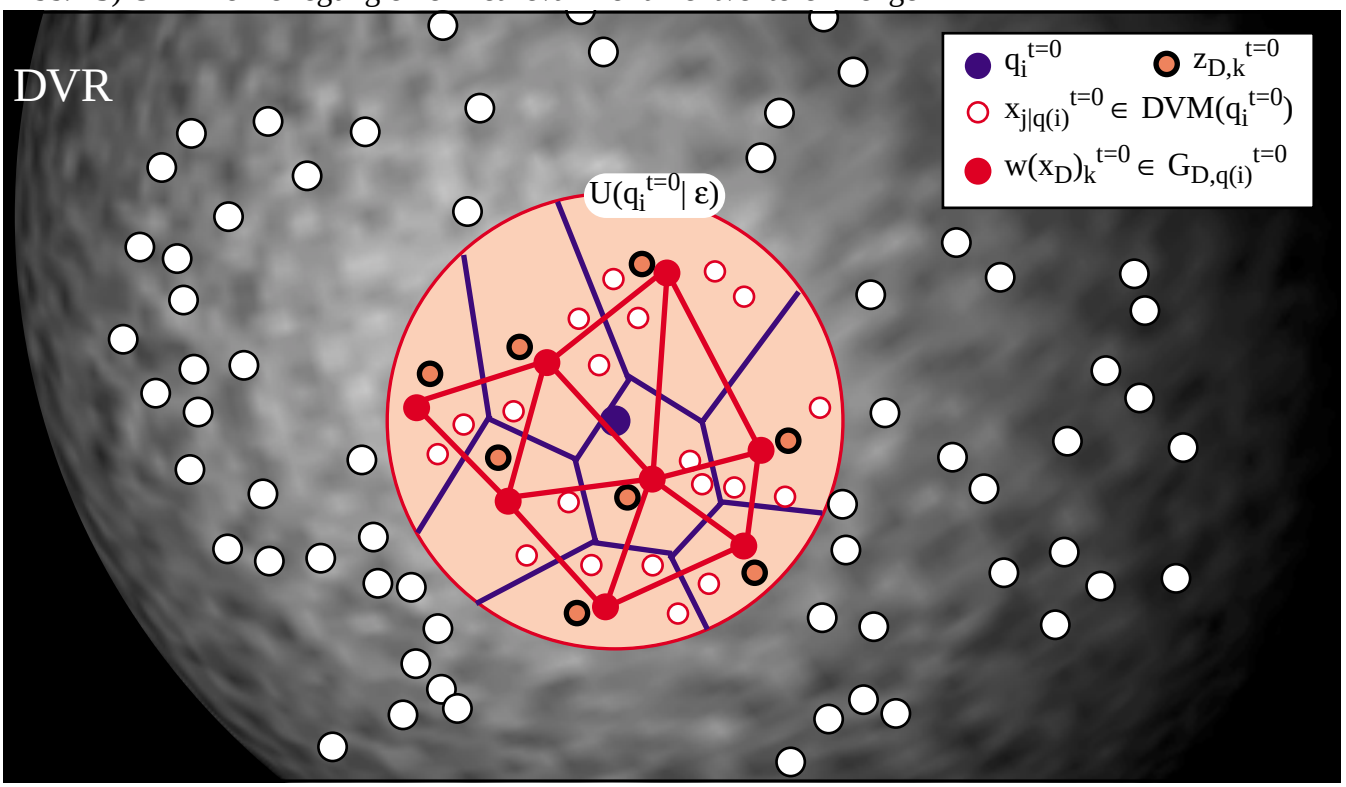
jedoch eine Vereinfachung, da die Integration jedes Wissensobjektes, das aus einem Dokument stammt, welches durch den Agenten bewertet wurde, in das Gesamtmodell des Agenten zu einer radikalen Veränderung der Ziele und der Relevanzfunktion führen kann.

Eine weitere Post-Retrieval-Operation ist die On-line-Clusterung jeder Retrieval-Dokumentvektorenmenge mit dem Ziel, die Anzahl der Objekte zu verringern, die durch den Agenten bewertet werden. Es soll der Fall betrachtet werden, dass $DVM(q_i^{t=0})$ als Stimuli für den Aufbau einer SC-GNG-SOM verwendet wird (siehe Abb. 75)). Die Retrievalregion $U(q_i^{t=0}| \varepsilon)$ wird durch die Voronoi-Regionen der sich ergebenden Gewichtsvektoren $w(x_D)_k^{t=0}$ des GNG-Graphen $G_{D,q(i)}^{t=0}$ zerlegt, und die Dokumentvektorenmenge $DVM(q_i^{t=0})$ wird in lokale Stimulusmengen $M_{D,k}^{t=0}$ zerlegt, die den einzelnen Gewichtsvektoren $w(x_D)_k^{t=0}$ zugeordnet sind.

Um die Zielsetzung der On-line-Clusterung zu erreichen, müssen mit Hilfe des GNG-Graphen $G_{D,q(i)}^{t=0}$ aus $DVM(q_i^{t=0})$ einzelne Elemente ausgewählt werden, deren korrespondierende Dokumente zur Bewertung vorgelegt werden. Da nur Dokumente bewertet werden können, nicht jedoch Vektoren in DVR, kann einem Gewichtsvektor $w(x_D)_k^{t=0}$ keine Relevanzbewertung zugeordnet werden, sodass der am nächsten liegende Dokumentvektor aus $M_{D,k}^{t=0}$ ausgewählt wird, der als Median-Dokumentvektor $z_{D,k}^{t=0}$ verwendet wird:

$$z_{D,k}^{t=0}: d_{DVR}(z_{D,k}^{t=0}, w(x_D)_k^{t=0}) = \min\{d_{DVR}(x_{kj|q(i)}^{t=0}, w(x_D)_k^{t=0}) \mid \forall x_{kj|q(i)}^{t=0} \in M_{D,k}^{t=0}\}. \quad (420)$$

Abb. 75) On-Line-Zerlegung einer Retrieval-Dokumentvektorenmenge

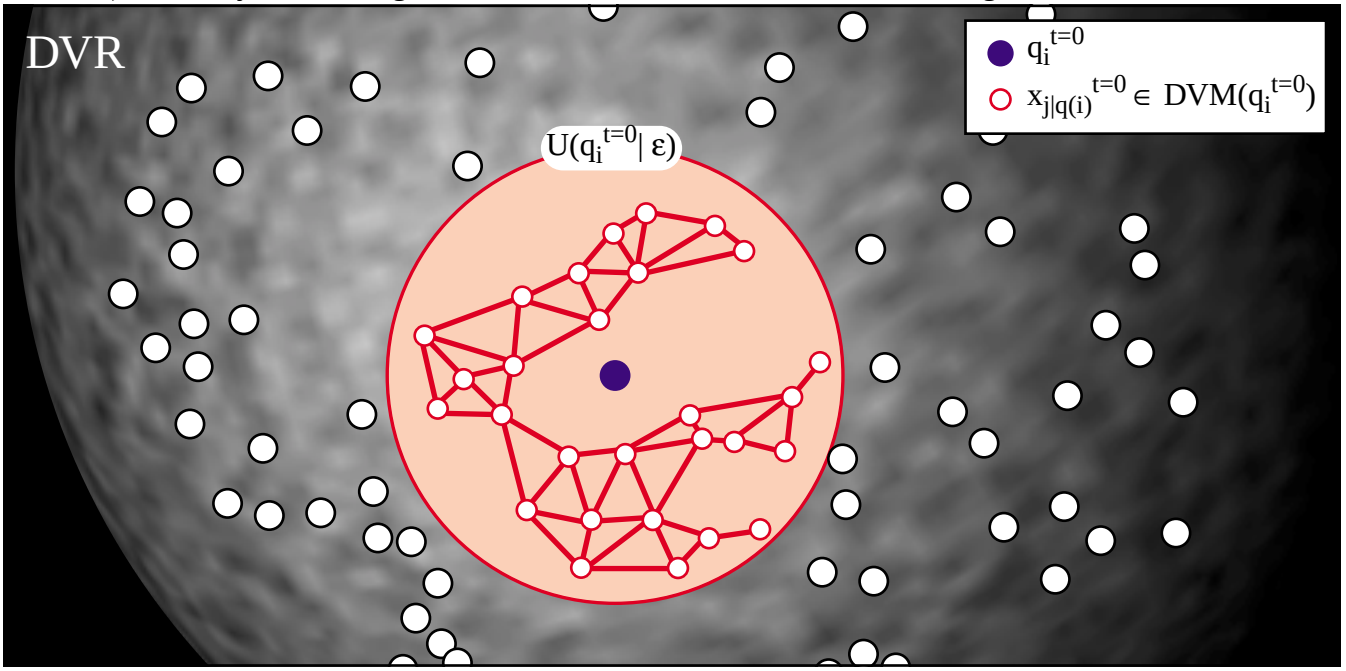


Das Dokument $D_k^{t=0}$, das zu dem Median-Dokumentvektor $z_{D,k}^{t=0}$ gehört, wird dem Agenten präsentiert und erhält eine Relevanzbewertung, mit der eine weitergehende Strategie durchgeführt werden kann.

Eine einfache Strategie ordnet jedem Dokumentvektor $x_{kj|q(i)}^{t=0}$ aus einer lokalen Stimulusmenge $M_{D,k}^{t=0}$ den gleichen Relevanzwert zu. Alternative Strategien nutzen z.B. eine Kernel-Regression, um den Elementen der Stimulusmengen Relevanzwerte zuzuordnen, wobei diese wieder verwendet werden können, um ein Ranking zu erzeugen. Dies erfordert jedoch eine reelle Relevanzbewertung, was im Kapitel 4) beschrieben wird.

Wird der Aufbau eines GNG-Graphen so lange weiter verfolgt, bis in jeder lokalen Stimulusmenge genau ein Element vorhanden ist, und wird der Gewichtsvektor jedes Neurons durch den Stimulusvektor ersetzt, so erhält man eine Delaunay-Teil-Triangulation der Retrieval-Dokumentvektormenge (siehe Abb. 76)). Mit einem solchen Graphen lassen sich interaktive Präsentations- und Bewertungsstrategien erstellen, bei denen der Agent zu einem Iterationszeitpunkt jeweils die unmittelbare Nachbarschaft $N(d_G=1)$ eines Bezugspunktes präsentiert bekommt. Je nach der Bewertung dieser Elemente wird ein neuer Bezugspunkt ausgewählt, dessen noch nicht bewertete Nachbarschaft im nächsten Iterationsschritt präsentiert wird. Durch die Auswahl geeigneter Graphensuchverfahren wird die Anzahl der Bewertungsoperationen in diesem Zusammenhang kleiner als die Anzahl der Knoten in dem Graph.

Abb. 76) Delaunay-Teil-Triangulation der Retrieval-Dokumentvektorenmenge



3.9.2.5) Queryvektor-Splitting

Das Query-Splitting (Panyr (1987b:148[252]), Borodin et al. (1971[56])) gehört zu den Operationen, die zu einer Effektivitätssteigerung beim Queryvektor-Relevanz-Feedback beitragen sollen. Faktisch handelt es sich dabei jedoch um Queryvektoren-Splitting, da die Operationen im Dokumentvektorenraum DVR durchgeführt werden und nicht im diskreten Raum $D(\Theta)$ von Zeichenketten, in dem Operationen definiert sind, die eine Query transformieren. Ein Queryvektor-Splitting wird in diesem Zusammenhang nicht als Post-Retrieval-Operation betrachtet, da das Splitting nach der Relevanzbewertung durch den Agenten durchgeführt wird, während die oben dargestellten Post-Retrieval-Operationen nach dem Retrieval und vor der Bewertung angesiedelt sind.

Mit genau einem Queryvektor ergeben sich Probleme, wenn die relevanten Dokumentvektoren z.B. in einer langgestreckten Region liegen, oder wenn mehrere Regionen mit relevanten Dokumentvektoren existieren, da in diesen Fällen die Annäherung an eine Teilmenge dieser Dokumentvektoren gleichzeitig zu einer Entfernung von einer anderen Teilmenge führt. Insbesondere bei mehreren lokalen Clustern von relevanten Dokumentvektoren ist das Aufteilen eines Queryvektors eine sinnvolle Strategie, wobei dadurch eine elegante Form von Polyrepräsentation durchgeführt wird, da die einzelnen Queryvektoren sich auf einzelne Eigenschaften (Regionen) spezifizieren können, ohne dass sie ihre Gemeinsamkeiten vollständig verlieren, da sie in dem gleichen Dokumentvektorenraum repräsentiert sind.

Beim Verfahren von Borodin et al. (1971[56]) werden die Dokumentvektoren in der Retrievalmenge untersucht, ob sie zu unterschiedlichen Clustern gehören. Ist die Korrelation bzw. Distanz größer als ein Schwellenwert, so gehören zwei Dokumente zu der gleichen Menge, ansonsten werden sie unterschiedlichen Mengen zugeordnet, was zu einem Splittung führt. Faktisch bedeutet diese Vorgehensweise eine Clusteranalyse der nachgewiesenen Dokumentvektoren, wie dies im Rahmen der Post-Processing-Operationen bereits beschrieben wurde. Jedem Cluster wird ein eigener Queryvektor zugeordnet, der durch eine gemischte Strategie entwickelt wird, wobei der Extremfall, dass keine positiven Elemente enthalten sind, zu der rein negativen Feedbackstrategie führt.

In Panyr (1987b:148[252]) wird erwähnt, dass die Begrenzung des Splittings durch geeignete Kriterien bzw. die Erkennung und Eliminierung unproduktiver Queryvektoren (Borodin et al. (1971:401[56])) wichtige Punkte zur Effizienzsteigerung bei diesen Verfahren darstellt. Dies wirkt sich auch auf die prinzipielle Anwendbarkeit aus, da die Anzahl der ermittelten Dokumente bei vielen Queryvektoren steigt, sodass die Ermittlung von Relevanzwerten durch die begrenzte Bereitschaft des Agenten zunehmend schwieriger wird.

Die Anzahl der Dokumente, für die eine Relevanzbewertung erwartet wird, kann begrenzt werden durch Verfahren, die im Rahmen der Polyrepräsentation des Queryvektors vorgestellt wurden, und die allgemein auf einem Combining-Verfahren der einzelnen Retrieval-Dokumentvektorenmengen basieren. D.h. die einzelnen Retrievalmengen werden nicht einfach in einer Vereinigungsmenge zusammengefasst, sondern es werden Rangfolgen gebildet, auf deren Plätze sich jeweils mehrere Dokumentvektoren befinden können. Die Zuordnung geschieht durch das mehrfache Auftreten von Dokumentvektoren in den einzelnen Retrievalmengen, d.h. je häufiger ein Dokumentvektor in den unabhängigen Retrievalmengen der gesplitteten Queryvektoren auftritt, desto wichtiger wird er bewertet, und desto höher befindet er sich in der Rangliste.

Einen alternativen Ansatz zum Query-Splitting kann im Zusammenhang mit einer mehrmaligen GNG-SOM-Adaption-Operation in natürlicher Weise hergeleitet werden, die bei der Queryvektor-Adaption bereits beschrieben wurde (siehe Abschnitt 3.9.2.2)). Dort wurde $q_i^{t=0}$ durch eine Sequenz von positiven bzw. negativen Einzel-Adaptionen verschoben, bis der sich ergebende Vektor nach einer bestimmten Anzahl von Adaptionen als Queryvektor $q_i^{t=1}$ verwendet wird. Wird diese GNG-SOM-Adaption mit einer GNG-SOM-Wachstums-Operation kombiniert, so ergibt sich daraus eine wachsende Queryvektor-GNG-SOM $N_q^{t=0}$. Unterschiedlich zu der Standardvorgehensweise ist, dass bei der Initialisierung genau ein Neuron $n(q)_{i1}^{t=0}$ mit dem Gewichtsvektor $w(q)_{i1}^{t=0} := q_i^{t=0}$ verwendet wird, anstatt zwei Neurone.

Damit die Standardvorgehensweise weiter verwendet werden kann, muss ein zweites Neuron $n(q)_{i2}^{t=0}$ mit einem Queryvektor $w(q)_{i2}^{t=0}$ eingeführt werden, wobei die Initialisierung nicht besonders wichtig ist, da der Vektor einer Vielzahl von Adaptionsoperationen ausgesetzt sein wird. Z.B. kann der Zentroid $\bar{s}_{DVM(i)}^{t=0}$ der nachgewiesenen Dokumentvektoren aus $DVM(q_i^{t=0})$ als Initialisierung des Gewichtsvektors $w(q)_{i2}^{t=0}$ verwendet werden.

Durchgeführt wird diese Form des Query-Splittings somit, indem nach der Bewertung durch den Agenten ein Netz $N_q^{t=0} = \{n(q)_{i1}^{t=0}, n(q)_{i2}^{t=0}\}$ initialisiert wird, mit $w(q)_{i1}^{t=0} := q_i^{t=0}$ und $w(q)_{i2}^{t=0} := \bar{s}_{DVM(i)}^{t=0}$. Es folgt eine Phase der positiven und negativen Adaptionen, indem aus $DVM(q_i^{t=0})$ jeweils gleichverteilt zufällig mit Zurücklegen ein Element $x_j^{t=0}$ gezogen wird, das als Fixpunkt der Adaption verwendet wird. Stammt $x_j^{t=0}$ aus $DVM(q_i^{t=0})_{rel}$, so wird eine positive Adaption durchgeführt, ansonsten eine negative Adaption. Das Gewinner-Neuron $n_{s(1|j)}^{t=0}$ wird einer Adaption mit $\varepsilon_{adapt,s}$ unterzogen, während alle seine Nachbar-Neurone einer Adaption mit $\varepsilon_{adapt,n} < \varepsilon_{adapt,s}$ unterzogen werden, was bei dem Initialisierungsfall das andere Neuron aus $N_q^{t=0}$ betrifft. Nachdem eine bestimmte konstante Anzahl von Ziehungen und Adaptionen durchgeführt wurde, wird die Adaptionsphase beendet und eine Wachstumsphase wird begonnen, bei der standardmäßig genau ein neues Neuron in das Netz eingefügt wird. Hierzu werden allen vorhandenen Neuronen ein lokales Fehlermaß zugeordnet und es wird das primäre und sekundäre Einfügezentrum als die beiden verbundenen Neurone mit dem größten Wert dieses Fehlermaßes ermittelt. Auf der Verbindungskante der Gewichtsvektoren dieser beiden Neurone wird ein neuer Gewichtsvektor initialisiert, gefolgt von den bekannten Aktualisierungsoperationen, welche die Verbindungsstruktur des Netzes betreffen. Nach einer solchen Wachstumsphase folgt wiederum eine Adaptionsphase, bis eine bestimmte Anzahl von Neuronen erreicht wurde, oder bis alle Neurone nach einer Adaptionsphase einen lokalen Fehlerwert kleiner einem Schwellenwert besitzen. Es ergibt sich eine Graphenstruktur ähnlich der On-Line-Zerlegung einer Retrieval-Dokumentvektorenmenge (siehe Abb. 75)), wobei die Gewichtsvektoren den Queryvektoren nach dem Querysplitting entsprechen.

3.9.2.6) Queryvektor-Polyrepräsentation beim Queryvektor-Feedback

Eine Polyrepräsentation eines Queryvektors wurde ohne den Kontext des Relevanz-Feedbacks zugunsten der Erzeugung der Polyrepräsentation einer Query vernachlässigt, die in Verbindung mit genau einer Indexierungsfunktion eine Polyrepräsentation eines Queryvektors liefert bzw. es wurde eine Query in Verbindung mit einer Polyrepräsentation der Indexierungsfunktion betrachtet, was ebenfalls eine Polyrepräsentation eines Queryvektors liefert. Die bislang dargestellten Möglichkeiten bezogen sich auf eine Wahrscheinlichkeitsverteilung um den Original-Queryvektor $q_i^{t=0}$, mit der eine Menge von anderen Queryvektoren durch Zufallsprozesse entsprechend der verwendeten Verteilung erzeugt werden.

Im Kontext des Relevanz-Feedbacks besteht nun die Möglichkeit, mit den Elementen aus $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ eine Menge von Queryvektoren zu erzeugen, die als Polyrepräsentation von $q_i^{t=0}$ betrachtet werden. Durch die Relevanzbewertungen handelt es sich um eine modifizierte bzw. angereicherte Repräsentation, da die neuen Queryvektoren Relevanz-Informationen nutzen, die bei der Erzeugung von $q_i^{t=0}$ durch die Indexierungsfunkion nicht verfügbar waren. Bei der Queryvektor-Polyrepräsentation können folgende Strategien unterschieden werden:

1) Combining-Strategie

   Bei dieser Strategie wird in jeder Iteration t aus q_i^t δ Vektoren q_{ik}^t erzeugt, die eine Queryvektormenge QVM_i^t bilden, wobei jeder Queryvektor eine eigene Ergebnis-Dokumentvektorenmenge $DVM(q_{ik}^t)$ liefert. Diese wird im einfachsten Fall zu einer gemeinsamen Menge $DVM(QVM_i^t)$ aggregiert, deren korrespondierende Dokumente dem Agenten präsentiert werden. Die Relevanzbewertungen werden verwendet, um einen einzelnen Queryvektor q_i^{t+1} zu generieren, aus dem wieder δ Vektoren q_{ik}^{t+1} erzeugt werden.

2) Einmalige Aufteilung

   Bei dieser Strategie wird bis zu einer bestimmten Iteration t genau ein Queryvektor verwendet, der genau eine Retrieval-Dokumentvektorenmenge erzeugt. In t wird genau einmal in der gesamten Feedback-Sequenz eine Polyrepräsentation von q_i^t durch δ Vektoren q_{ik}^t erzeugt, die einzelne Mengen $DVM(q_{ik}^t)$ liefern. Diese Einzel-Ergebnismengen werden jedoch nicht aggregiert, sondern verwendet, um δ Nachfolgevektoren q_{ik}^{t+1} zu erzeugen, die in allen weiteren Iterationen verwendet werden, d.h. jeder Vektor q_{ik}^{t+1} liefert eine Retrievalmenge $DVM(q_{ik}^{t+1})$, die nach der Bewertung zur Erzeugung von q_{ik}^{t+2} herangezogen wird.

3) Progressive Aufteilung

   Bei dieser Strategie wird in jeder Iteration aus einem vorhandenen Queryvektor eine Polyrepräsentation erzeugt, wobei in der nachfolgenden Iteration diese Vektoren wieder gesplittet werden.

Eine Polyrepräsentation mit Hilfe von Elementen aus $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ soll wiederum als statistische Polyrepräsentation durchgeführt werden, indem Resamplingverfahren angewendet werden, die aus den beiden Teilmengen unterschiedliche Elementlisten durch Ziehen mit Zurücklegen spezifizieren. Ein Beispiel ist die Durchführung einer gemischten Relevanz-Feedback-Strategie mit genau einem positiven und einem negativen Element, d.h. aus $DVM(q_i^{t=0})_{rel}$ wird genau ein positives Element und aus $DVM(q_i^{t=0})_{\overline{rel}}$ genau ein negatives Element gezogen, um einen Queryvektor $q_{ik}^{t=1}$ der Nachfolgeiteration zu erzeugen:

$$q_{ik}^{t=1} = \alpha * q_i^{t=0} + \beta * x_{j|q(i),rel}^{t=0} - \chi * x_{j|q(i),\overline{rel}}^{t=0}. \tag{421}$$

Ein weiteres Beispiel ist die Erzeugung von δ Bootstraplisten $DVL(q_i^{t=0})_{rel,k}$ und $DVL(q_i^{t=0})_{\overline{rel},k}$ aus den beiden Mengen $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ durch $f_{rel}^{t=0}$,faches Ziehen mit Zurücklegen im ersten Fall, und $f_{\overline{rel}}^{t=0}$,faches Ziehen mit Zurücklegen im zweiten Fall. Es ergeben sich die beiden Bootstrapmengen:

$$BDVL(q_i^{t=0})_{rel} = \{DVL(q_i^{t=0})_{rel,k} \mid k = 1, ..., \delta\},$$
$$BDVL(q_i^{t=0})_{\overline{rel}} = \{DVL(q_i^{t=0})_{\overline{rel},k} \mid k = 1, ..., \delta\}. \tag{422}$$

Für eine konkrete Polyrepräsentation wird aus diesen beiden Mengen jeweils ein Element $DVL(q_i^{t=0})_{rel,k}$ und $DVL(q_i^{t=0})_{\overline{rel},k}$ ohne Zurücklegen gezogen, gefolgt von der Bildung des arithmetischen Mittelwertvektors $\bar{s}_{DVM(i,rel,k)}^{t=0}$ bzw. $\bar{s}_{DVM(i,\overline{rel},k)}^{t=0}$:

$$\bar{s}_{DVM(i,rel,k)}^{t=0} = 1/f_{rel}^{t=0} * \sum_j x_{j|q(i),rel}^{t=0}, \forall x_{j|q(i),rel}^{t=0} \in DVL(q_i^{t=0})_{rel,k},$$
$$\bar{s}_{DVM(i,\overline{rel},k)}^{t=0} = 1/f_{\overline{rel}}^{t=0} * \sum_j x_{j|q(i),\overline{rel}}^{t=0}, \forall x_{j|q(i),\overline{rel}}^{t=0} \in DVL(q_i^{t=0})_{\overline{rel},k}. \tag{423}$$

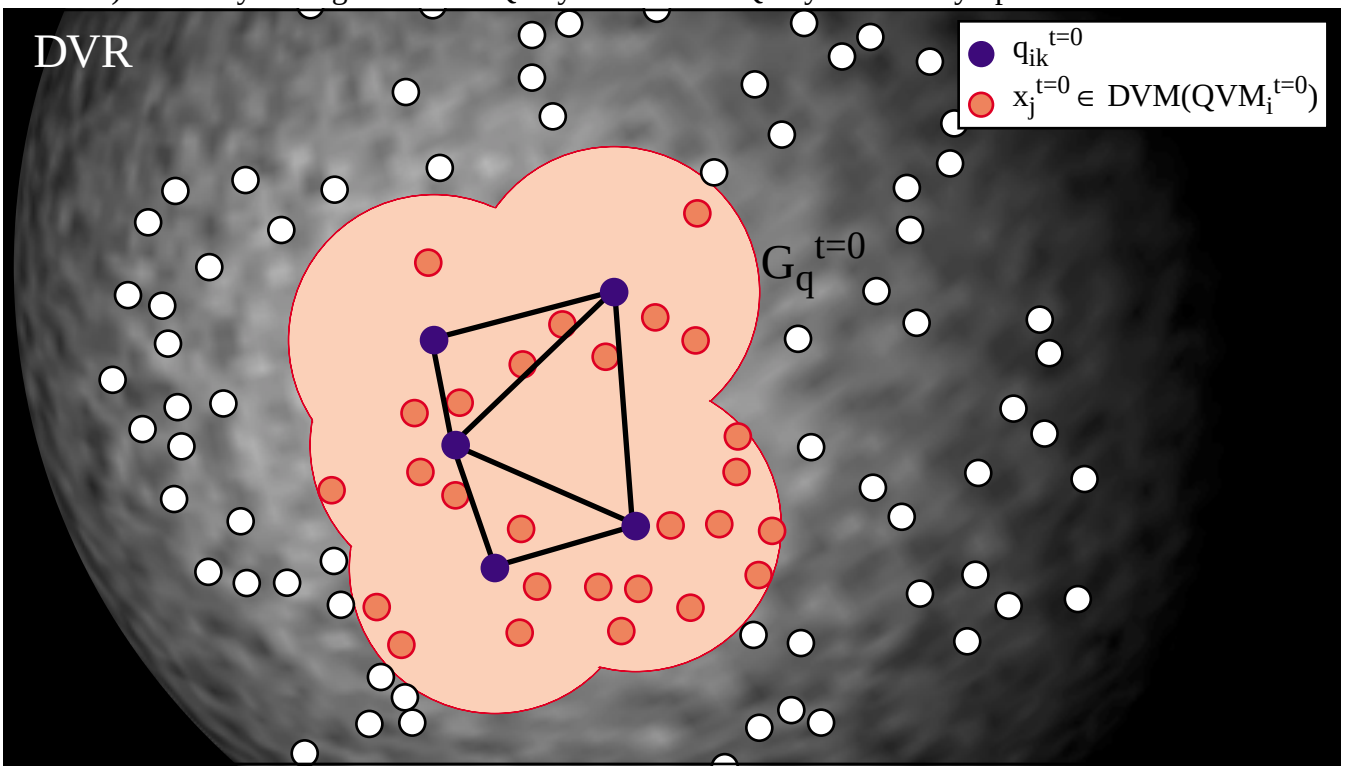
Ein poly-repräsentierter Nachfolge-Queryvektor $q_{ik}^{t=1}$ von $q_i^{t=0}$ im Rahmen des Relevanz-Feedbacks unter Verwendung der gemischten Strategie ergibt sich durch:

$$q_{ik}^{t=1} = \alpha * q_i^{t=0} + \beta * \bar{s}_{DVM(i,rel,k)}^{t=0} - \chi * \bar{s}_{DVM(i,\overline{rel},k)}^{t=0}, \text{ mit } \alpha,\beta,\chi \geq 0. \quad (424)$$

Wird anstatt der oben dargestellten gemischten Strategie ein Queryvektor-Feedback auf der Basis von GNG-SOM-Adaptionen unterstellt, so lassen sich damit andere Vorgehensweisen gestalten, von denen hier eine vorgestellt werden soll. Es soll angenommen werden, dass eine Queryvektormenge $QVM_i^t = \{q_{ik}^t \in DVR \mid k = 1, ..., \delta\}$ existiert, deren Elemente adaptiert werden, d.h. im Gegensatz zu einer stochastischen Erzeugung und Löschung von Queryvektoren, ist diese Form der Adaption quasi durch eine Erhaltung der Identität von Queryvektoren gekennzeichnet, da ein Vektor q_{ik}^t durch einen oder eine Sequenz von Adaptionsoperationen in q_{ik}^{t+1} überführt wird.

Es soll der Fall dargestellt werden, dass die Queryvektoren durch ein GNG-SOM-Verfahren strukturiert werden, wobei der Spezialfall einer Delaunay-Triangulation verwendet werden soll, d.h. ein SC-GNG-SOM-Verfahren wird solange durchgeführt, bis sich in der lokalen Lernmenge jedes Neurons genau ein Queryvektor befindet (siehe Abb. 77)).

Abb. 77) Delaunay-Triangulation der Queryvektoren bei Queryvektor-Polyrepräsentation



Parallel zu der Retrieval-Operation in t=0 und der Bewertung durch den Agenten wird ein Queryvektorennetz aufgebaut, das vereinfachend $N_q^{t=0}$ bezeichnet wird, d.h. $N_q^{t=0} := N_{iq}^{t=0}$. Dessen Neurone $n_{q,k}^{t=0}$ bestehen aus einem Gewichtsvektor $w(x_q)_k^{t=0} \in DVR$, der Verbindungsstruktur in Form des Verbindungsvektors $C_{q,k}^{t=0}$, sowie einer lokalen Stimulusmenge $M_{q,k}^{t=0}$, in der die Queryvektoren aus QVM_i^t eingetragen werden, die sich in der Voronoi-Region des betreffenden Gewichtsvektors befinden. Begonnen wird das Wachstum der SC-GNG-SOM wie üblich mit 2 Neuronen, und der Wachstumsprozess wird solange fortgesetzt, bis δ Neurone erzeugt wurden:

$$N_q^{t=0} = \{n_{q,k}^{t=0} = (w(x_q)_k^{t=0}, M_{q,k}^{t=0}, C_{q,k}^{t=0}) \mid k= 1, ..., \delta\}. \quad (425)$$

Der Adaptionsprozess wird solange fortgesetzt, bis in jeder der lokalen Stimulusmengen genau ein Queryvektor liegt. Danach wird der Gewichtsvektor eines Neurons ersetzt durch den Queryvektor, der sich in seiner Lernmenge befindet. Der zugehörige Graph im DVR wird als $G_q^{t=0}$ bezeichnet, und ist die Delaunay-Triangulation bzw. eine ein Teilgraph dieser Triangulation (siehe Abb. 77)):

$$N_q^{t=0} := N_{q,i}^{t=0} = \{n_{q,k}^{t=0} = (q_{ik}^{t=0}, M_{q,k}^{t=0}, C_{q,k}^{t=0}) \mid k= 1, ..., \delta\}. \quad (426)$$

Nach der Bewertung durch den Agenten ergeben sich die beiden Dokumentvektorenmengen $DVM(QVM_i^{t=0})_{rel}$ und $DVM(QVM_i^{t=0})_{\overline{rel}}$. Es folgt die eigentliche Queryvektor-Adaptionsphase von $N_q^{t=0}$, bei der eine mehrmalige Adaption mit positiver und negativer Stimulusmenge unterstellt wird. Bei der Adaption wird aus $DVM(QVM_i^{t=0})$ gleichverteilt zufällig ein Element $x_{j,L}^{t=0}$ mit Zurücklegen gezogen, und entsprechend seiner Zugehörigkeit zu $DVM(QVM_i^{t=0})_{rel}$ oder $DVM(QVM_i^{t=0})_{\overline{rel}}$ als positiver Fixpunkt $x_{j,L,rel}^{t=0}$ oder negativer Fixpunkt $x_{j,L,\overline{rel}}^{t=0}$ der Adaption verwendet. Unabhängig von der Art der Adaption wird zunächst das Gewinner-Neuron aus $N_q^{t=0}$ ermittelt, als das Neuron $n_{s(1|j)}^{t=0}$, dessen Gewichtsvektor $w(x_q)_{s(1|j)}^{t=0}$ die geringste Distanz von $x_{j,L}^{t=0}$ besitzt. Da die Gewichtsvektoren gleich einem der Queryvektoren gesetzt wurden, ergibt dies:

$$n_{s(1|j)}^{t=0}: d_{DVR}(q_{s(1|j)}^{t=0}, x_{j,L}^{t=0}) = \min\{d_{DVR}(q_k^{t=0}, x_{j,L}^{t=0}) \mid \forall n_{q,k}^{t=0} \in N_q^{t=0}\}. \quad (427)$$

Der Queryvektor dieses Gewinner-Neurons wird der Gewinner-Neuron-Adaption unterzogen, und seine unmittelbaren Nachbarn, die in der Menge $N(d_G=1 \mid G_q^{t=0})_{s(1|j)}$ gesammelt werden, werden einer Nachbar-Neuron-Adaption unterzogen. Alle anderen Neurone werden nicht adaptiert, d.h. diese Queryvektoren bleiben bei dieser Adaptionsoperation unberücksichtigt. Unabhängig ob eine positive oder negative Adaption vorliegt, die Verschiebedistanz in Form des Deltavektors $\Delta q_{s(1|j)}^{t=0}(alt)$ bzw. $\Delta q_{s(1|j)k}^{t=0}(alt)$ ergibt sich durch:

$$\Delta q_{s(1|j)}^{t=0}(alt) = \varepsilon_s * d_{DVR}(q_{s(1|j)}^{t=0}(alt), x_{j,L}^{t=0}),$$
$$\Delta q_{s(1|j)k}^{t=0}(alt) = \varepsilon_n * d_{DVR}(q_{s(1|j)k}^{t=0}(alt), x_{j,L}^{t=0}), \forall n_{s(1|j)k}^{t=0} \in N(d_G=1 \mid G_q^{t=0})_{s(1|j)}. \quad (428)$$

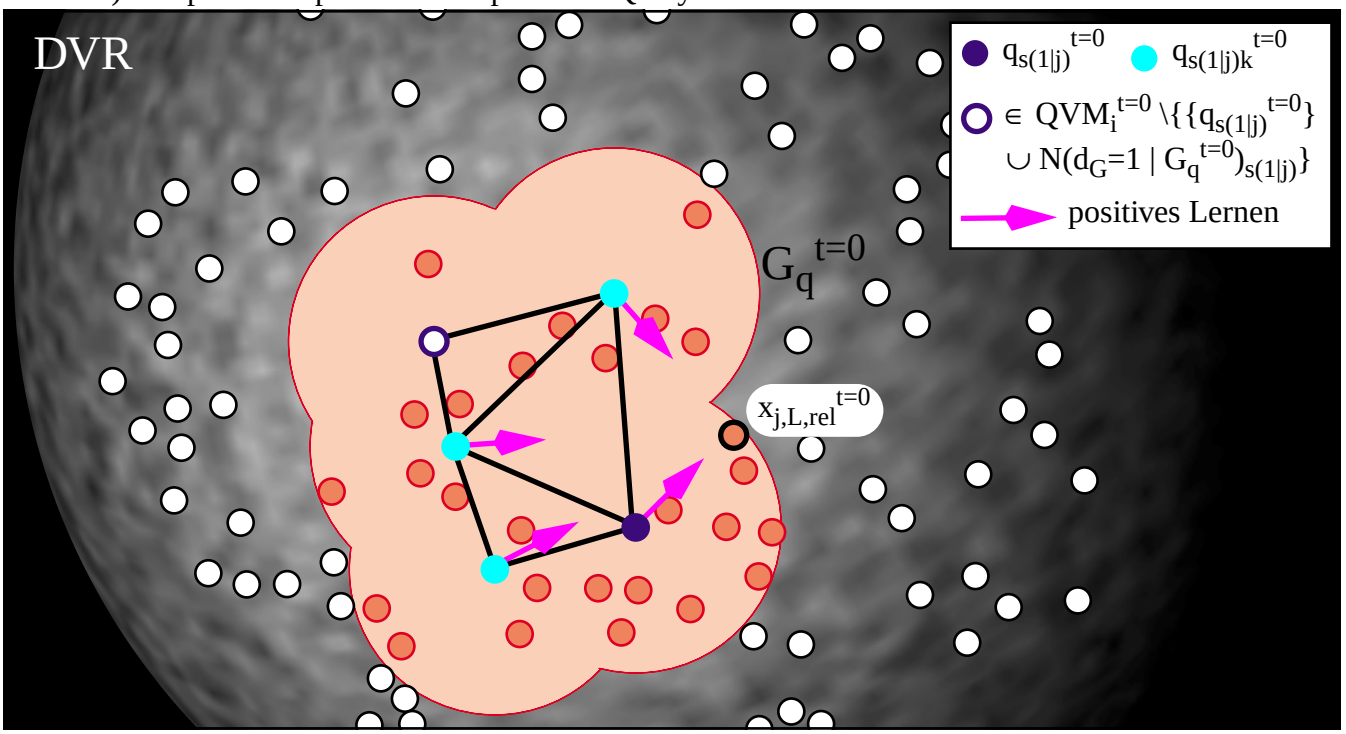
Stammt $x_{j,L}^{t=0}$ aus $DVM(QVM_i^{t=0})_{rel}$, so wird eine positive Adaption durchgeführt, bei der die Queryvektoren in Richtung $x_{j,L}^{t=0}$ verschoben werden (siehe Abb. 78)):

$$q_{s(1|j)}^{t=0}(neu) = q_{s(1|j)}^{t=0}(alt) + \Delta q_{s(1|j)}^{t=0}(alt),$$
$$q_{s(1|j)k}^{t=0}(alt) = q_{s(1|j)k}^{t=0}(alt) + \Delta q_{s(1|j)k}^{t=0}(alt). \quad (429)$$

Stammt $x_{j,L}^{t=0}$ aus $DVM(QVM_i^{t=0})_{\overline{rel}}$, so wird eine negative Adaption durchgeführt, bei der die Queryvektoren von $x_{j,L}^{t=0}$ weg verschoben werden:

$$q_{s(1|j)}^{t=0}(neu) = q_{s(1|j)}^{t=0}(alt) - \Delta q_{s(1|j)}^{t=0}(alt),$$
$$q_{s(1|j)k}^{t=0}(neu) = q_{s(1|j)k}^{t=0}(alt) - \Delta q_{s(1|j)k}^{t=0}(alt). \quad (430)$$

Abb. 78) Beispiel einer positiven Adaption der Queryvektoren



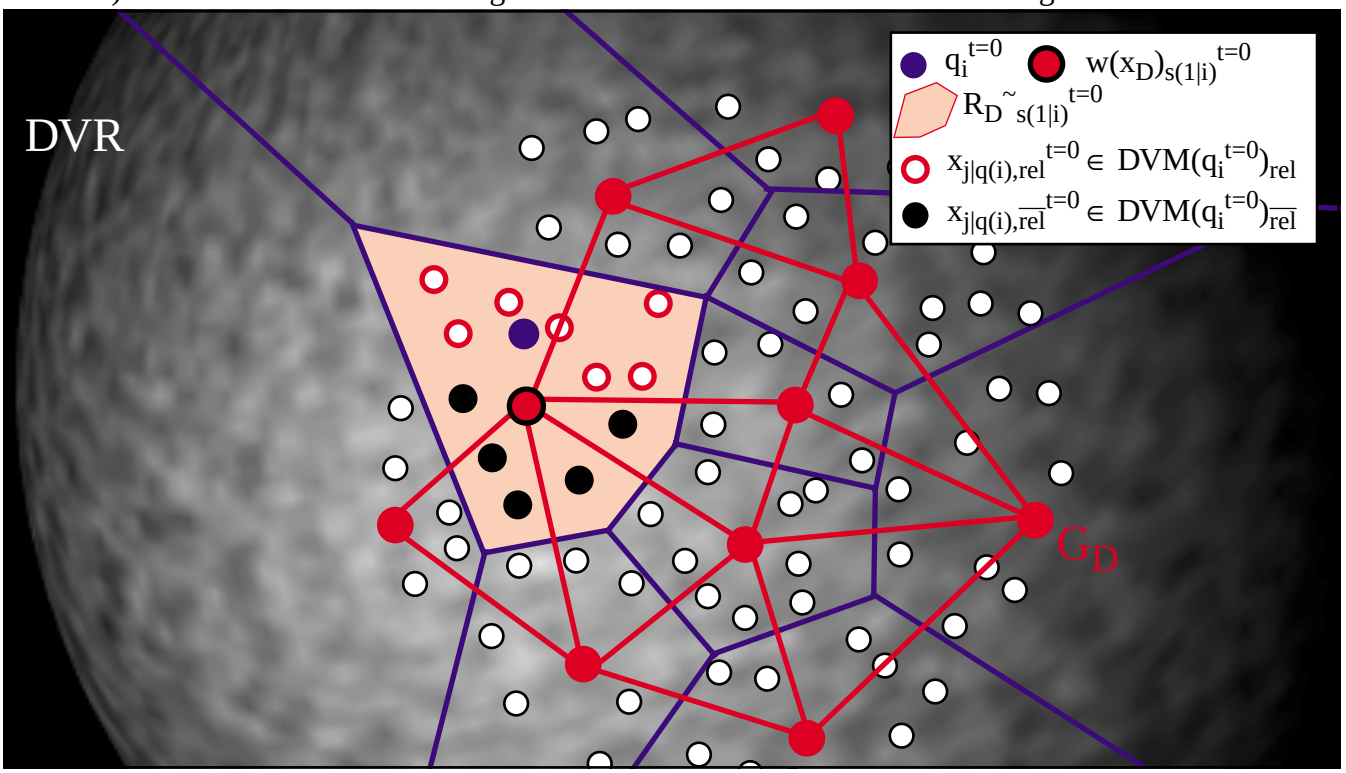
Nachdem eine bestimmte Anzahl von Ziehungen und Adaptionen durchgeführt wurde, die als Funktion der Anzahl der Dokumentvektoren in $DVM(QVM_i^{t=0})$ bestimmt wird, wird die sich ergebende Queryvektorverteilung als Ausgangspunkt für die nächste Iteration t=1 verwendet, wobei die Menge der Queryvektoren als $QVM_i^{t=1}$, ihr Graph im DVR als $G_{q,i}^{t=1}$ und das Netz als $N_{q,i}^{t=1}$ bezeichnet wird. Mit Hilfe der neuen Umgebungen $U(q_{ik}^{t=1} \mid \varepsilon)$ wird in t=1 die Gesamt-Ergebnismenge $DVM(QVM_i^{t=1})$ aus der Restmenge $DV^t \setminus DVM(QVM_i^{t=0})$ ermittelt, wobei ein Abbruch von Seiten des IRS erfolgt, wenn diese Menge zum ersten Mal leer wird.

3.9.2.7) Queryvektor-Feedback bei Dokumentvektoren-GNG-SOMs

Entgegen der On-Line-Clusterung bei einer ansonsten unklassifizierten Dokumentvektorenmenge DV, können Relevanz-Feedback-Strategien eine vorhandene Dokumentvektoren-Klassifikation DVK nutzen, wobei von einer Zerlegung durch eine SC-GNG-SOM ausgegangen werden soll. Prinzipiell gilt, dass bei jeder Iteration des Relevanz-Feedback-Verfahrens die Retrieval-Strategien verwendet werden, die bei einer SC-GNG-SOM der Dokumentvektorenmenge ohne Feedback anwendbar sind (siehe Abschnitt 3.8.2)). Dadurch können die weiteren Darstellungen verkürzt werden, wobei zunächst das Retrieval durch die Voronoi-Region des Gewinner-Neurons betrachtet werden soll (siehe Abb. 79)).

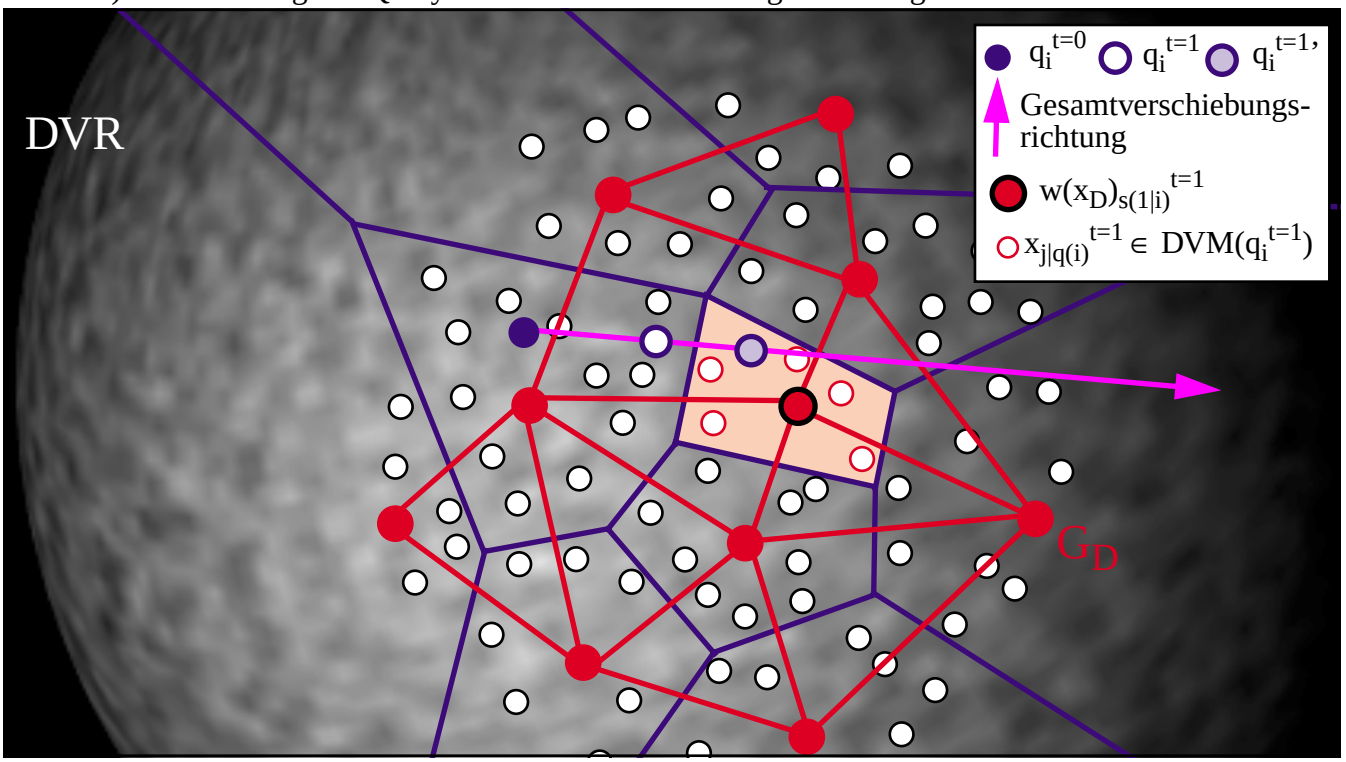
Die Retrieval-Dokumentvektorenmenge $DVM(q_i^{t=0})$ ist in diesem Fall gleich der lokalen Stimulusmenge $M_{D,s(1|i)}^{t=0}$ des Neurons $n_{D,s(1|i)}^{t=0}$, in dessen Voronoi-Region $R_{D\tilde{}s(1|i)}^{t=0}$ der Queryvektor $q_i^{t=0}$ liegt. Die Dokumentvektoren der Stimuli aus $M_{D,s(1|i)}^{t=0}$ werden durch den Agenten bewertet, wodurch die beiden Mengen $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ hervorgehen. Für jeden der beiden Mengen wird der Mittelwertsvektor gebildet, und diese beiden Vektoren bilden das positive und das negative Adaptionszentrum für den positiven und den negativen Lernprozess im Rahmen einer gemischten Strategie, aus der $q_i^{t=1}$ hervorgeht.

Abb. 79) Retrieval durch Voronoi-Region des Gewinner-Neurons und Bewertung der lokalen Stimuli



Das Problem dieser Vorgehensweise liegt darin, dass es zu einem Verfahrensabbruch kommt, wenn $q_i^{t=1}$ wieder in der Voronoi-Region $R_D\tilde{}_{s(1|i)}^{t=0}$ liegt, da somit keine neuen Dokumentvektoren ermittelt werden. Entweder muss die entsprechende Retrieval-Strategie erweitert werden, oder es muss eine der anderen SC-GNG-SOM-Retrieval-Strategien verwendet werden. Eine Modifikation könnte z.B. die Verschiebelänge bei konstanter Verschieberichtung verwenden (siehe Abb. 80)).

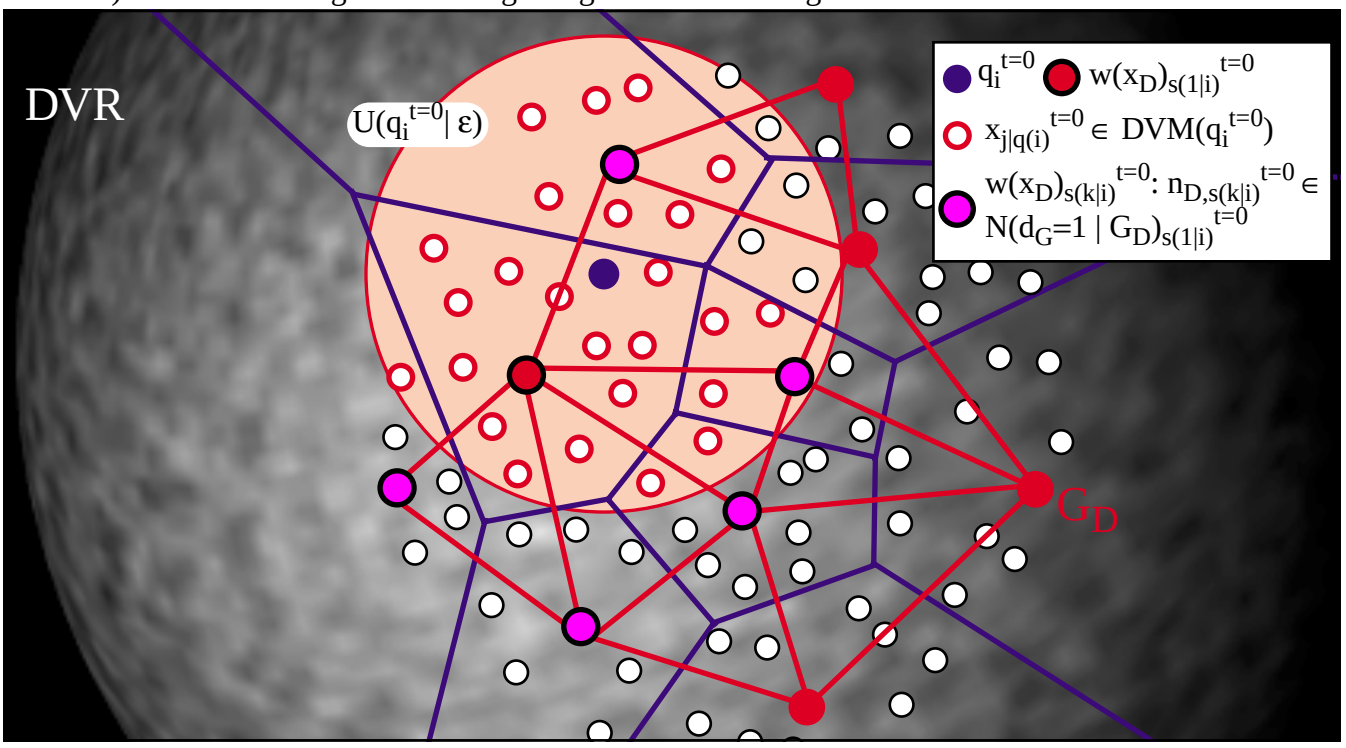
Abb. 80) Verschiebung des Queryvektors über Voronoi-Region hinweg



Wird $q_i^{t=1}$ bei konstanten Parametern α, β, χ ermittelt, und liegt der neue Queryvektor noch in $R_D\tilde{}_{s(1|i)}^{t=0}$, so kann aus der Kenntnis von $q_i^{t=0}$ und $q_i^{t=1}$ ein Strahl berechnet werden, der in $q_i^{t=0}$ beginnt, und über $q_i^{t=1}$ hinausführt. Entsprechend einer zu spezifizierenden Verschiebestrategie werden auf diesem Strahl versuchsweise Punkte festgelegt, die eine größere Distanz zu $q_i^{t=0}$ besitzen als $q_i^{t=1}$, und für die geprüft wird, ob sie in $R_D\tilde{}_{s(1|i)}^{t=0}$ liegen. Der erste Versuchspunkt, der nicht mehr in $R_D\tilde{}_{s(1|i)}^{t=0}$ liegt, wird als modifizierter Queryvektor $q_i^{t=1}$, für die Iteration t=1 des Relevanz-Feedbacks verwendet, wobei durch die Kenntnis des Gewinner-Neurons die neue Retrieval-Dokumentvektorenmengen gleich bekannt ist. Eine ähnliche Strategie kann durch eine Vergrößerung des Gewichtungsfaktors β des positiven Lernens und χ des negativen Lernens durchgeführt werden. Bei anderen Modifikationsarten kann die Nachbarschaftsmenge $N(d_G=1 \mid G_D)_{s(1|i)}^{t=0}$ verwendet werden, indem das Neuron aus dieser Menge ausgewählt wird, zu dessen Gewichtsvektor der verschobene Queryvektor $q_i^{t=1}$ die größte Annäherung gegenüber $q_i^{t=0}$ erzielt. Diese Alternative benötigt keine weiteren Spezifizierungen, wie die Festlegung der probeweise erzeugten Punkte auf der Verschiebegeraden, sodass diese Alternative vorzuziehen wäre.

Eine Retrieval-Strategie, die eine ε-Umgebung und die Nachbarschaftsmenge $N(d_G=1 \mid G_D)_{s(1|i)}^{t=0}$ verwendet, würde demgegenüber mit einer höheren Wahrscheinlichkeit bei jeder Feedback-Iteration neue Dokumentvektoren liefern, da eine größere Granularität besteht, im Vergleich zu der Auswahl einer gesamten lokalen Stimulusmenge. Wie bei der Gewinner-Neuron-Strategie wird das Neuron $n_{D,s(1|i)}^{t=0}$ mit dem Gewichtsvektor $w(x_D)_{s(1|i)}^{t=0}$ ermittelt, während die Retrieval-Dokumentvektorenmenge aus der Grundmenge der Vereinigung der lokalen Stimulusmengen der Neurone aus $N(d_G \leq 1 \mid G_D)_{s(1|i)}^{t=0}$ ermittelt wird. Dabei ist $DVM(q_i^{t=0})$ die Teilmenge aus dieser Grundmenge, deren Dokumentvektor weniger als ε von $q_i^{t=0}$ entfernt liegt (siehe Abb. 81)). Dokumentvektoren, die in der ε–Umgebung liegen, jedoch nicht in der Grundmenge, werden bei dieser Strategie aus $DVM(q_i^{t=0})$ ausgeschlossen.

Abb. 81) Retrieval-Strategie mit ε-Umgebung und Voronoi-Regionen



3.9.2.8) Queryvektor-Feedback mit positiven und negativen Queryvektoren

Bezogen auf den Retrieval-Prozess ist es unerheblich, ob eine bzw. mehrere positive Queryvektoren vorliegen, oder ob positive und negative Queryvektoren verwendet werden, da in jedem Fall eine Dokumentvektorenmenge $DV(q_i)$ bzw. $DV(q_i^+, q_i^-)$ oder $DV(QVM_i^{+-t})$' ermittelt wird, deren korrespondierende Dokumentmenge dem Agenten präsentiert wird. Der Unterschied im Rahmen eines Queryvektor-Relevanz-Feedbacks besteht jedoch darin, dass positive und negative Queryvektoren unterschiedlich adaptiert werden könnten, was im weiteren untersucht werden soll.

Ausgegangen werden soll von dem allgemeinen Fall des Vorliegens einer Menge QVM_i^{+t} von positiven Queryvektoren q_{ij}^{+t} und einer Menge QVM_i^{-t} von negativen Queryvektoren q_{ij}^{-t}, wobei eine Polyrepräsentation vorliegt, wenn in mindestens einer der beiden Mengen mehr als ein Element vorliegt. Die Queryvektoren definieren zusammen mit einer Retrievalstrategie, d.h. der Beschreibung von Retrievalregionen, eine Gesamtmenge $DV(QVM_i^{+-t})$ von Dokumentvektoren, aus denen diejenigen entfernt werden, die ausschließlich in negativen Regionen liegen, sodass $DV(QVM_i^{+-t})$' erzeugt wird. Die daraus gebildete geordnete Liste durch ein Rankingverfahren spielt in diesem Kontext keine Rolle. Es wird weiterhin eine binäre Relevanzbewertung durch den Agenten unterstellt, d.h. $DV(QVM_i^{+-t})$' wird in eine Menge der relevanten und in eine Menge der nicht relevanten Dokumentvektoren zerlegt:

$$DV(QVM_i^{+-t=0})_{rel} := \{x_{k,rel}^{t=0} \in DV(QVM_i^{+-t})' \mid rel(x_{j,rel}^{t=0}) = 1;\ j = 1, ..., f_{rel}^{t=0}\},$$
$$DV(QVM_i^{+-t=0})_{\overline{rel}} := \{x_{k,\overline{rel}}^{t=0} \in DV(QVM_i^{+-t})' \mid rel(x_{j,\overline{rel}}^{t=0}) = 0;\ j = 1, ..., f_{\overline{rel}}^{t=0}\}. \quad (431)$$

Wie beim Queryvektor-Feedback mit genau einem positiven Queryvektor $q_i^{t=0}$ wird zunächst der Mittelwertsvektor für diese beiden Mengen gebildet:

$$\bar{s}_{Q+-(i,rel)}^{t=0} = 1/f_{rel}^{t=0} * \sum_k x_{k,rel}^{t=0},\ \forall x_{k,rel}^{t=0} \in DV(QVM_i^{+-t=0})_{rel},$$
$$\bar{s}_{Q+-(i,\overline{rel})}^{t=0} = 1/f_{\overline{rel}}^{t=0} * \sum_k x_{k,\overline{rel}}^{t=0},\ \forall x_{k,\overline{rel}}^{t=0} \in DV(QVM_i^{+-t=0})_{\overline{rel}}, \quad (432)$$

gefolgt von der Adaption des Queryvektors durch eine gemischte Strategie

$$q_i^{t=1} = \alpha * q_i^{t=0} + \beta * \bar{s}_{Q+-(i,rel)}^{t=0} - \chi * \bar{s}_{Q+-(i,\overline{rel})}^{t=0},\ \text{mit } \alpha, \beta, \chi \geq 0, \quad (433)$$

oder einer SOM-Adaption

$$q_i^{t=1} = q_i^{t=0} + \Delta_{rel}q_i^{t=0} - \Delta_{\overline{rel}}q_i^{t=0},\ \text{mit}$$
$$\Delta_{rel}q_i^{t=0} = \varepsilon_{rel} * d_{DVR}(q_i^{t=0}, \bar{s}_{Q+-(i,rel)}^{t=0}),\ \text{und}\ \Delta_{\overline{rel}}q_i^{t=0} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{t=0}, \bar{s}_{Q+-(i,\overline{rel})}^{t=0}). \quad (434)$$

Bei dieser Standardform ist der Queryvektor q_i^t ein positiver Queryvektor, d.h. $q_i^t \equiv q_i^{+t}$, sodass beim Vorliegen von positiven und negativen Queryvektoren q_{ij}^{+t} und q_{ij}^{-t} direkt diese Form der Adaption für alle positiven Queryvektoren übernommen werden kann. Es verbleibt nur noch die Frage, ob bzw. wie die negativen Queryvektoren adaptiert werden sollen.

Die einfachste Strategie besteht im Verzicht auf die Adaption der negativen Queryvektoren, da diese als Repräsentation von Negativ-Beispielen interpretiert werden können. D.h. alle positiven Queryvektoren aus der Grundmenge QVM_i^+ mit einer konstanten Elementanzahl sollen durch eine gemischte Strategie

adaptiert werden, während alle negativen Queryvektoren aus der Grundmenge QVM_i^- mit einer konstanten Elementanzahl konstant bleiben:

$$\forall q_{ij}^+ \in QVM_i^+: q_{ij}^{+t=1} = q_{ij}^{+t=0} + \Delta_{rel} q_{ij}^{+t=0} - \Delta_{\overline{rel}} q_{ij}^{+t=0} \text{ und}$$
$$\forall q_{ij}^- \in QVM_i^-: q_{ij}^{-t=1} = q_{ij}^{-t=0}, \text{ mit}$$
$$\Delta_{rel} q_{ij}^{+t=0} = \varepsilon_{rel} * d_{DVR}(q_i^{+t=0}, \bar{s}_{Q+-(i,rel)}^{t=0}), \Delta_{\overline{rel}} q_{ij}^{+t=0} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{+t=0}, \bar{s}_{Q+-(i,\overline{rel})}^{t=0}). \quad (435)$$

Werden die negativen Queryvektoren weniger streng als Repräsentationen nicht relevanter Beispiele interpretiert, so ist eine Adaption sinnvoll, da durch die Kenntniss der neuen Menge $DV(QVM_i^{+-t=0})_{\overline{rel}}$ weitere nicht relevante Dokumentvektoren zur Verfügung stehen. D.h. die nicht relevanten Dokumentvektoren tragen zur Adaption der negativen Queryvektoren bei, wobei verschiedene Möglichkeiten der Realisierung existieren. Beispielsweise können alle neuen nicht relevanten Dokumentvektoren als negative Queryvektoren verwendet werden, d.h. die Menge QVM_i^{-t} wird bei jeder Feedback-Iteration um die Elemente aus $DV(QVM_i^{+-t=0})_{\overline{rel}}$ erweitert, sodass QVM_i^{-t+1} entsteht, wobei eine Adaption der Elemente aus QVM_i^{-t+1} ebenfalls nicht durchgeführt wird:

$$\forall q_{ij}^+ \in QVM_i^+: q_{ij}^{+t=1} = q_{ij}^{+t=0} + \Delta_{rel} q_{ij}^{+t=0} - \Delta_{\overline{rel}} q_{ij}^{+t=0} \text{ und}$$
$$\forall q_{ij}^- \in QVM_i^{-t+1} = QVM_i^{-t=0} \cup DV(QVM_i^{+-t=0})_{\overline{rel}} : q_{ij}^{-t=1} = q_{ij}^{-t=0}. \quad (436)$$

Anstatt alle nicht relevanten Dokumentvektoren als negative Queryvektoren zu verwenden, könnte auch eine Form der Aggregation der Elemente aus $DV(QVM_i^{+-t=0})_{\overline{rel}}$ erzeugt und als neuer negativer Queryvektor verwendet werden, wobei sich der bereits gebildete Mittelwertsvektor $\bar{s}_{Q+-(i,\overline{rel})}^{t=0}$ anbietet:

$$\forall q_{ij}^+ \in QVM_i^+: q_{ij}^{+t=1} = q_{ij}^{+t=0} + \Delta_{rel} q_{ij}^{+t=0} - \Delta_{\overline{rel}} q_{ij}^{+t=0} \text{ und}$$
$$\forall q_{ij}^- \in QVM_i^{-t+1} = QVM_i^{-t=0} \cup \{\bar{s}_{Q+-(i,\overline{rel})}^{t=0}\}: q_{ij}^{-t=1} = q_{ij}^{-t=0}. \quad (437)$$

Wird analog zu der Adaption der positiven Dokumentvektoren vorgegangen, so wird eine gemischte Strategie angewendet, die positive wie negative Queryvektoren auf der Basis der relevanten wie nicht relevanten aktuellen Dokumentvektoren adaptieren, ohne dass die Anzahl der Queryvektoren verändert wird. Ein positiver Queryvektor wird einer positiven Adaption mit dem Fixpunkt $\bar{s}_{Q+-(i,rel)}^{t=0}$ und einer negativen Adaption mit dem Fixpunkt $\bar{s}_{Q+-(i,\overline{rel})}^{t=0}$ unterzogen. Demgegenüber wird ein negativer Queryvektor einer negativen Adaption mit dem Fixpunkt $\bar{s}_{Q+-(i,rel)}^{t=0}$ und einer positiven Adaption mit dem Fixpunkt $\bar{s}_{Q+-(i,\overline{rel})}^{t=0}$ unterzogen:

$$\forall q_{ij}^+ \in QVM_i^+: q_{ij}^{+t=1} = q_{ij}^{+t=0} + \Delta_{rel} q_{ij}^{+t=0} - \Delta_{\overline{rel}} q_{ij}^{+t=0} \text{ und}$$
$$\forall q_{ij}^- \in QVM_i^-: q_{ij}^{-t=1} = q_{ij}^{-t=0} - \Delta_{rel} q_{ij}^{-t=0} + \Delta_{\overline{rel}} q_{ij}^{-t=0}, \text{ mit}$$
$$\Delta_{rel} q_{ij}^{+t=0} = \varepsilon_{rel} * d_{DVR}(q_i^{t=0}, \bar{s}_{Q+-(i,rel)}^{t=0}), \Delta_{\overline{rel}} q_{ij}^{+t=0} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{+t=0}, \bar{s}_{Q+-(i,\overline{rel})}^{t=0}),$$
$$\Delta_{rel} q_{ij}^{-t=0} = \varepsilon_{rel} * d_{DVR}(q_i^{-t=0}, \bar{s}_{Q+-(i,rel)}^{t=0}), \Delta_{\overline{rel}} q_{ij}^{-t=0} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{-t=0}, \bar{s}_{Q+-(i,\overline{rel})}^{t=0}). \quad (438)$$

Weiterhin kann von einer gemischten Strategie abgesehen werden, und eine Retrievalstrategie etabliert werden, bei der nur die relevanten Dokumentvektoren zur Adaption der positiven Queryvektoren und nur die nicht relevanten Dokumentvektoren zur Adaption der negativen Queryvektoren beitragen, wobei in

beiden Fällen eine positive Adaption, d.h. eine Verschiebung in Richtung des Adaptions-Fixpunktes, durchgeführt wird:

$$\forall q_{ij}^{+} \in QVM_i^{+}: q_{ij}^{+t=1} = q_{ij}^{+t=0} + \Delta_{rel} q_{ij}^{+t=0} \text{ und}$$
$$\forall q_{ij}^{-} \in QVM_i^{-}: q_{ij}^{-t=1} = q_{ij}^{-t=0} + \Delta_{\overline{rel}} q_{ij}^{-t=0}, \text{ mit}$$
$$\Delta_{rel} q_{ij}^{+t=0} = \varepsilon_{rel} * d_{DVR}(q_i^{t=0}, \bar{s}_{Q+-(i,rel)}^{t=0}), \Delta_{\overline{rel}} q_{ij}^{-t=0} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{-t=0}, \bar{s}_{Q+-(i,\overline{rel})}^{t=0}). \quad (439)$$

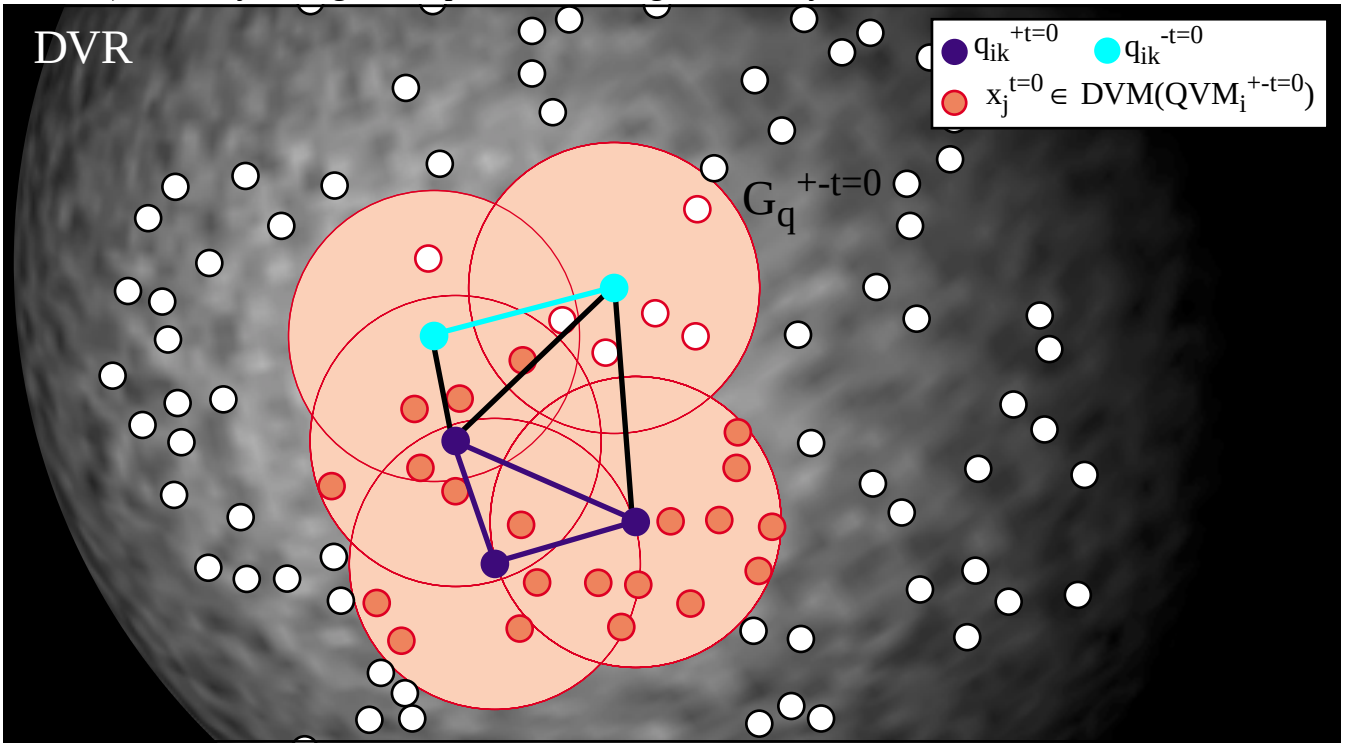
Wird die Menge der negativen Queryvektoren durch nicht relevante Dokumentvektoren erweitert, so könnte auch die Menge der positiven Queryvektoren durch die relevanten Dokumentvektoren erweitert werden. Zu klären ist dabei, ob in einer Iteration nur die alten Elemente der beiden Mengen QVM_i^{+} und QVM_i^{+} adaptiert werden sollen, oder auch die neuen Elemente. Wird eine gemischte Strategie für die positiven wie negativen Queryvektoren durchgeführt, und wird zuerst die Adaption und dann die Mengenerweiterung durchgeführt, so ergibt sich:

$$\forall q_{ij}^{+} \in QVM_i^{+t=0}: q_{ij}^{+t=1} = q_{ij}^{+t=0} + \Delta_{rel} q_{ij}^{+t=0} - \Delta_{\overline{rel}} q_{ij}^{+t=0} \text{ und}$$
$$\forall q_{ij}^{-} \in QVM_i^{-t=0}: q_{ij}^{-t=1} = q_{ij}^{-t=0} - \Delta_{rel} q_{ij}^{-t=0} + \Delta_{\overline{rel}} q_{ij}^{-t=0},$$
$$QVM_i^{+t+1} = QVM_i^{+t=0} \cup DV(QVM_i^{+-t=0})_{rel}, QVM_i^{-t+1} = QVM_i^{-t=0} \cup DV(QVM_i^{+-t=0})_{\overline{rel}}. \quad (440)$$

Werden zunächst die Mengen erweitert und dann die Adaption durchgeführt, so ergibt sich:

$$\forall q_{ij}^{+} \in QVM_i^{+t=0} = QVM_i^{+t=0} \cup DV(QVM_i^{+-t=0})_{rel}: q_{ij}^{+t=1} = q_{ij}^{+t=0} + \Delta_{rel} q_{ij}^{+t=0} - \Delta_{\overline{rel}} q_{ij}^{+t=0} \text{ und}$$
$$\forall q_{ij}^{-} \in QVM_i^{-t=0} = QVM_i^{-t=0} \cup DV(QVM_i^{+-t=0})_{\overline{rel}}: q_{ij}^{-t=1} = q_{ij}^{-t=0} + \Delta_{rel} q_{ij}^{-t=0} - \Delta_{\overline{rel}} q_{ij}^{-t=0}. \quad (441)$$

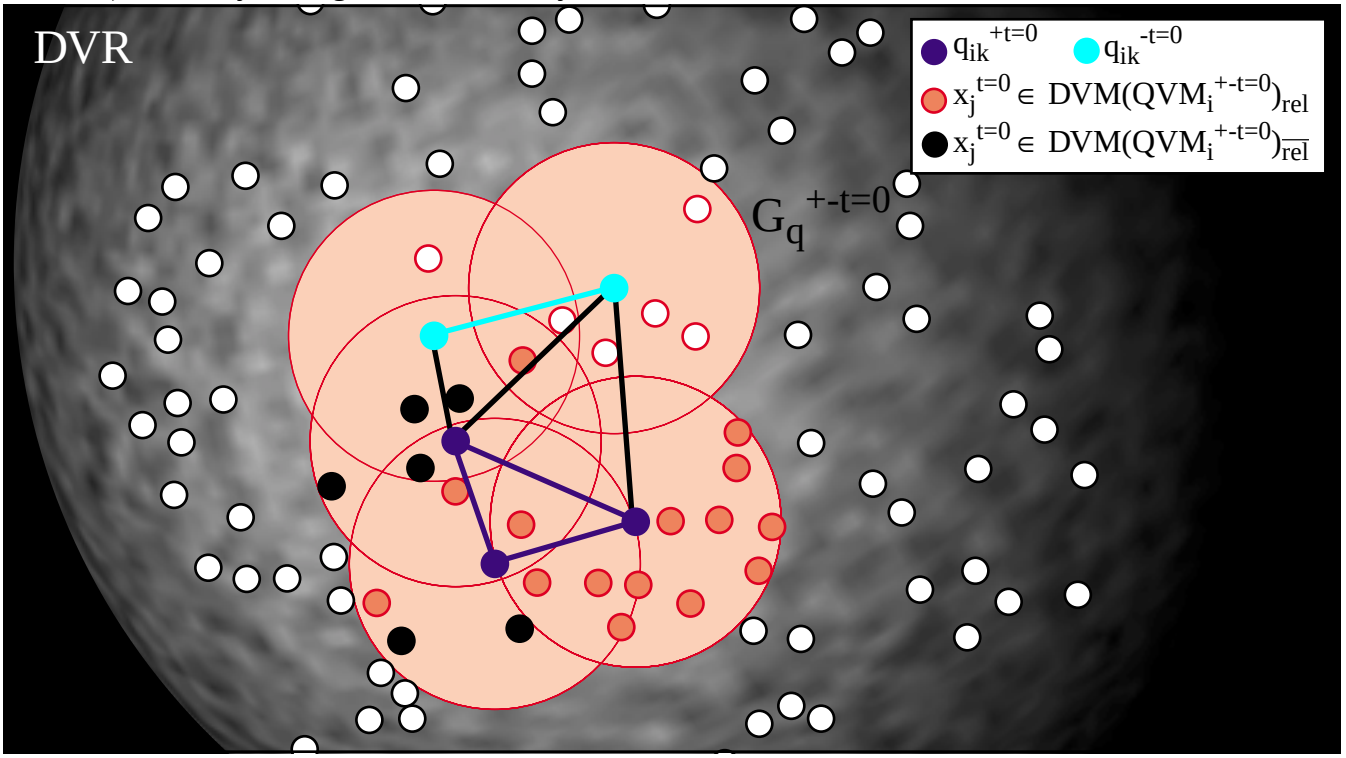
Abb. 82) Delaunay-Triangulation positiver und negativer Queryvektoren



Neben den vorgestellten Möglichkeiten der einmaligen Adaption, bei der die beiden Mittelwertsvektoren $\bar{s}_{Q+-(i,rel)}^{t=0}$ und $\bar{s}_{Q+-(i,\overline{rel})}^{t=0}$ als Fixpunkte in einer Adaptionsgleichung verwendet werden, bestehen Möglichkeiten der mehrmaligen Adaption, bei der richtige Dokumentvektoren aus den beiden Mengen $DV(QVM_i^{+-t=0})_{rel}$ und $DV(QVM_i^{+-t=0})_{\overline{rel}}$ als Fixpunkte der Adaption verwendet werden. Vorgestellt werden soll der allgemeine Fall, bei dem mehrere positive und negative Queryvektoren vorliegen, die in der Initialisierungs-Iteration durch ein SC-GNG-SOM zu einer Delaunay-Triangulation strukturiert wurden (siehe Abb. 82)). Es wird der Fall dargestellt, dass positive wie negative Queryvektoren in einer gemeinsamen Struktur $N_q^{+-t=0}$ integriert sind. Zu beachten ist, dass alle Dokumentvektoren, die in mindestens einer Umgebung eines positiven Queryvektors liegen, in die Gesamt-Ergebnismenge $DVM(QVM_i^{+-t=0})$ aufgenommen wurden, während Dokumentvektoren, die nur in einer oder mehreren Umgebungen von negativen Queryvektoren liegen, ausgeschlossen werden.

Nach der Bewertung der Dokumente, deren Dokumentvektor in $DVM(QVM_i^{+-t=0})$ enthalten sind, existiert eine direkte Zerlegung der Dokumentvektorenmenge in $DV(QVM_i^{+-t=0})_{rel}$ und $DV(QVM_i^{+-t=0})_{\overline{rel}}$ (siehe Abb. 83)).

Abb. 83) Delaunay-Triangulation der Queryvektoren mit relevanten und nicht relevanten Dok-Vektoren



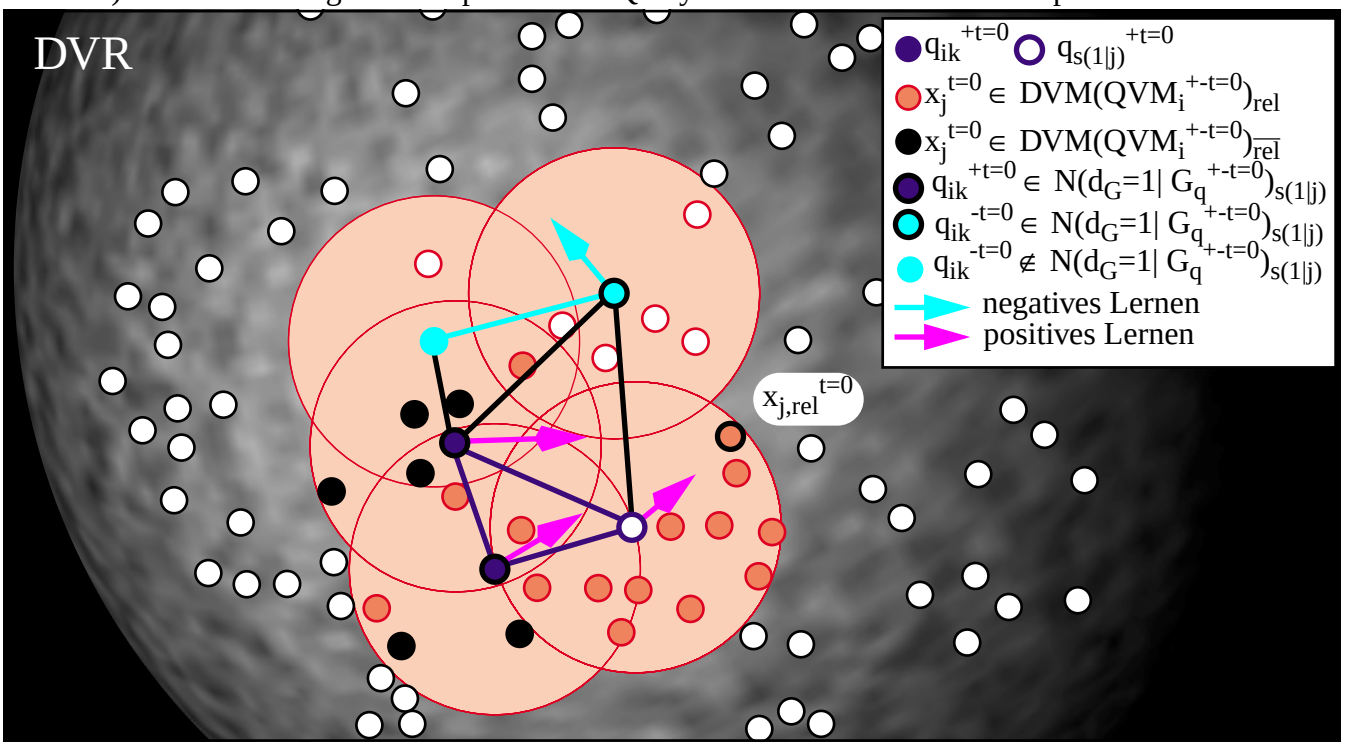
Die Adaptionsphase besteht aus einer Sequenz von Adaptions-Operationen, bei denen jeweils gleichverteilt zufällig mit Zurücklegen ein Element $x_j^{t=0}$ aus $DVM(QVM_i^{+-t=0})$ gezogen wird. Ist es Element von $DV(QVM_i^{+-t=0})_{rel}$, so wird es als $x_{j,rel}^{t=0}$ bezeichnet, während es als Element von $DV(QVM_i^{+-t=0})_{\overline{rel}}$ mit $x_{j,\overline{rel}}^{t=0}$ bezeichnet wird. Im nächsten Schritt wird aus dem Netz $N_q^{+-t=0}$ dasjenige Neuron $n_{s(1|j)}^{t=0}$ ausgewählt, dessen Gewichtsvektor, d.h. dessen Queryvektor den geringsten Abstand von $x_j^{t=0}$ besitzt. Dieses Gewinner-Neuron wird einer Adaption mit dem Parameter ε_s und seine verbundenen Nachbarn aus $N(d_G=1 \mid G_q^{+-t=0})_{s(1|j)}$ einer Adaption mit dem Parameter ε_n unterzogen, wobei die Art der Adaption von der Art des Fixpunktes und der Art der Queryvektoren abhängt.

Ist der Fixpunkt ein relevanter Dokumentvektor, d.h. $x_{j,rel}^{t=0}$, und ist der zu adaptierende Queryvektor positiv, so wird eine positive Adaption durchgeführt, während bei einem negativen Queryvektor eine negative Adaption durchgeführt wird. Ist der Fixpunkt nicht relevant, d.h. $x_{j,\overline{rel}}^{t=0}$, so wird bei einem positiven Queryvektor eine negative und bei einem negativen Queryvektor eine positive Adaption durchgeführt. Für den Queryvektor des Gewinner-Neurons ergeben sich somit die folgenen Alternativen:

$$x_{j,rel}^{t=0} \wedge q_{s(1|j)}^{+t=0}: q_{s(1|j)}^{+t=1} = q_{s(1|j)}^{+t=0} + \Delta q_{s(1|j)}^{+t=0}, \Delta q_{s(1|j)}^{+t=0} = \varepsilon_s * d_{DVR}(q_{s(1|j)}^{+t=0}, x_{j,rel}^{t=0})$$
$$x_{j,rel}^{t=0} \wedge q_{s(1|j)}^{-t=0}: q_{s(1|j)}^{-t=1} = q_{s(1|j)}^{-t=0} - \Delta q_{s(1|j)}^{-t=0}, \Delta q_{s(1|j)}^{-t=0} = \varepsilon_s * d_{DVR}(q_{s(1|j)}^{-t=0}, x_{j,rel}^{t=0})$$
$$x_{j,\overline{rel}}^{t=0} \wedge q_{s(1|j)}^{+t=0}: q_{s(1|j)}^{+t=1} = q_{s(1|j)}^{+t=0} - \Delta q_{s(1|j)}^{+t=0}, \Delta q_{s(1|j)}^{+t=0} = \varepsilon_s * d_{DVR}(q_{s(1|j)}^{+t=0}, x_{j,\overline{rel}}^{t=0})$$
$$x_{j,\overline{rel}}^{t=0} \wedge q_{s(1|j)}^{-t=0}: q_{s(1|j)}^{-t=1} = q_{s(1|j)}^{-t=0} + \Delta q_{s(1|j)}^{-t=0}, \Delta q_{s(1|j)}^{-t=0} = \varepsilon_s * d_{DVR}(q_{s(1|j)}^{-t=0}, x_{j,\overline{rel}}^{t=0}). \quad (442)$$

In Abb. 84) wird der erste Fall beschrieben, d.h. der Fixpunkt ist relevant und der Gewinner-Queryvektor ist positiv, sodass $q_{s(1|j)}^{+t=0}$ in Richtung $x_{j,rel}^{t=0}$ verschoben wird. Das gleiche gilt für die beiden positiven Queryvektoren in der Nachbarschaftsmenge $N(d_G=1|\ G_q^{+-t=0})_{s(1|j)}$. Der eine negative Queryvektor in der Nachbarschaftsmenge unterliegt einer negativen Adaption, d.h. $q_{ik}^{-t=0}$ wird weg von $x_{j,rel}^{t=0}$ verschoben. Es liegt noch ein negativer Queryvektor außerhalb der Nachbarschaft vor, der nicht adaptiert wird.

Abb. 84) Positive und negative Adaptionen der Queryvektoren bei relevantem Fixpunkt

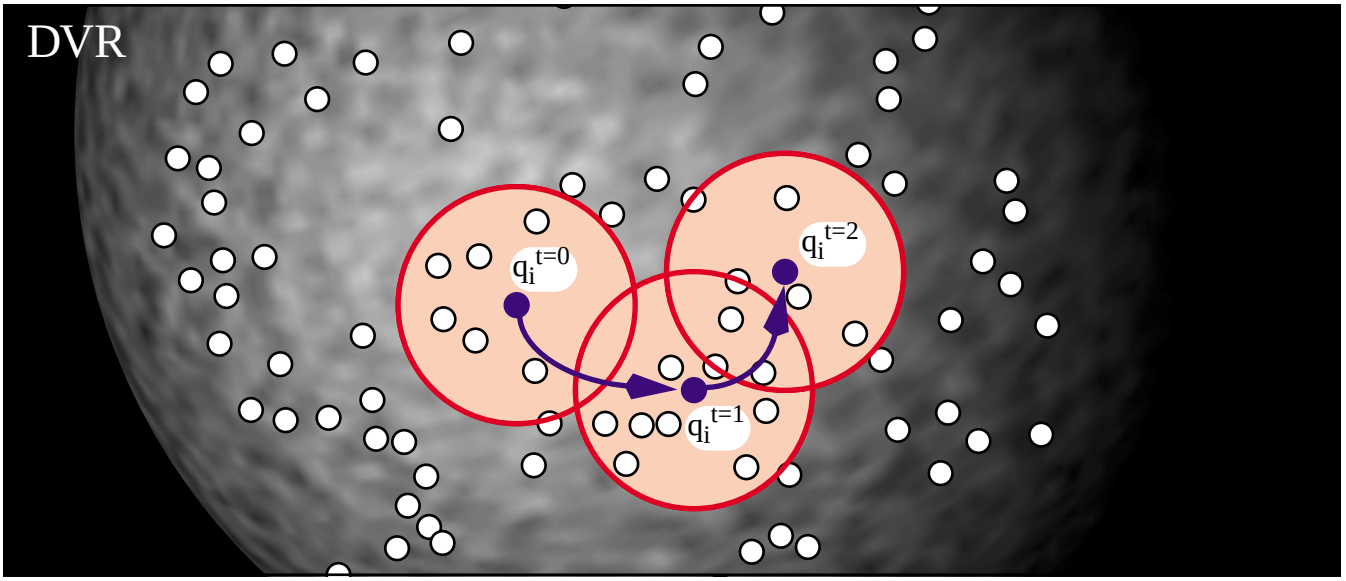


3.9.2.9) Queryvektoren-Trajektorie

Wird eine Queryvektoren-Monorepräsentation über eine Anzahl von Feedback-Iterationen betrachtet, so kann man eine Sequenz oder Trajektorie $q_i^{t=0}$ -> $q_i^{t=1}$ -> ... dieser Punkte im DVR einzeichnen (siehe Abb. 85)), die für Prognosezwecke genutzt werden kann. Denkbar wäre die Schätzung des Endpunktes der Trajektorie, bei dem der Agent die Feedback-Operation abbrechen wird, d.h. es wird zu einem Zeit-

punkt t < t(end) ein Punkt $q_i^{t(end)\wedge}$ gesucht, der den richtigen jedoch noch unbekannten Punkt $q_i^{t(end)}$ approximiert. Wird die Interaktion zwischen IRS und Agent im Rahmen des Query-Relevanz-Feedbacks als Suche nach einem Queryvektor interpretiert, der das Informationsbedürfnis des Agenten optimal beschreibt (siehe Panyr (1987b:146[252])), so ist die Approximation $q_i^{t(end)\wedge}$ ein Versuch, dieses Optimum zu extrapolieren.

Abb. 85) Trajektorie dreier nachfolgender Queryvektoren



Die Approximation von $q_i^{t(end)\wedge}$ beinhaltet zwei Komponenten:
1) Approximation einer Kurve tra_i im n-dimensionalen Raum DVR.
2) Approximation eines Punktes $q_i^{t(end)\wedge}$ auf dieser Kurve tra_i.

Es stellt sich die Frage, wie viele Queryvektoren für die Approximation der Kurve eingesetzt werden sollen. Es muss davon ausgegangen werden, dass die Anzahl der Stützpunkte, d.h. die Anzahl der Queryvektoren, im Vergleich zu der Anzahl n der Dimension des DVR sehr klein ist, sodass eine Unterdeterminiertheit der Approximation angenommen werden muss. Das Spektrum reicht von der Verwendung der letzten zwei Queryvektoren als Stützpunkte bis zur Verwendung aller bislang verfügbaren Queryvektoren. Werden die beiden letzten Stützpunkte verwendet, so bietet sich dies in Verbindung mit einer linearen Interpolation an.

Sollen alle bisherigen Queryvektoren auf dieser Kurve liegen, so muss ein Interpolationsverfahren angewendet werden (siehe Abb. 86)a), ansonsten kann ein Regressionsverfahren verwendet werden (siehe Abb. 86)b); siehe auch Abschnitt 2.1.2)).

Wird angenommen, dass eine Funktion tra_i durch ein symbolisches Regressionsverfahren spezifiziert wurde, so soll im nächsten Schritt ein Punkt auf der Kurve bestimmt werden, der jenseits des letzten Stützpunktes liegt, und der bestimmte weitere Eigenschaften besitzt. Die Approximation des nächsten Queryvektors erscheint dabei als leichtere Aufgabe als die Approximation des Endpunktes, sodass dieser Fall zunächst betrachtet werden soll. Aus den vorliegenden Queryvektoren sollen die Distanzen der jeweils aufeinander folgenden Vektoren bestimmt werden, d.h. $d_{DVR}(q_i^{t=0},q_i^{t=1})$ und $d_{DVR}(q_i^{t=1},q_i^{t=2})$ in

dem betrachteten Beispiel. Aus den Distanzwerten soll eine Distanzwertefunktion gebildet werden, aus welcher der nächste Wert zu $d_{DVR}(q_i^{t=2},q_i^{t=3})\wedge$ geschätzt wird (siehe Abb. 87)).

Abb. 86) Interpolation und Regression der Trajektorie dreier Queryvektoren

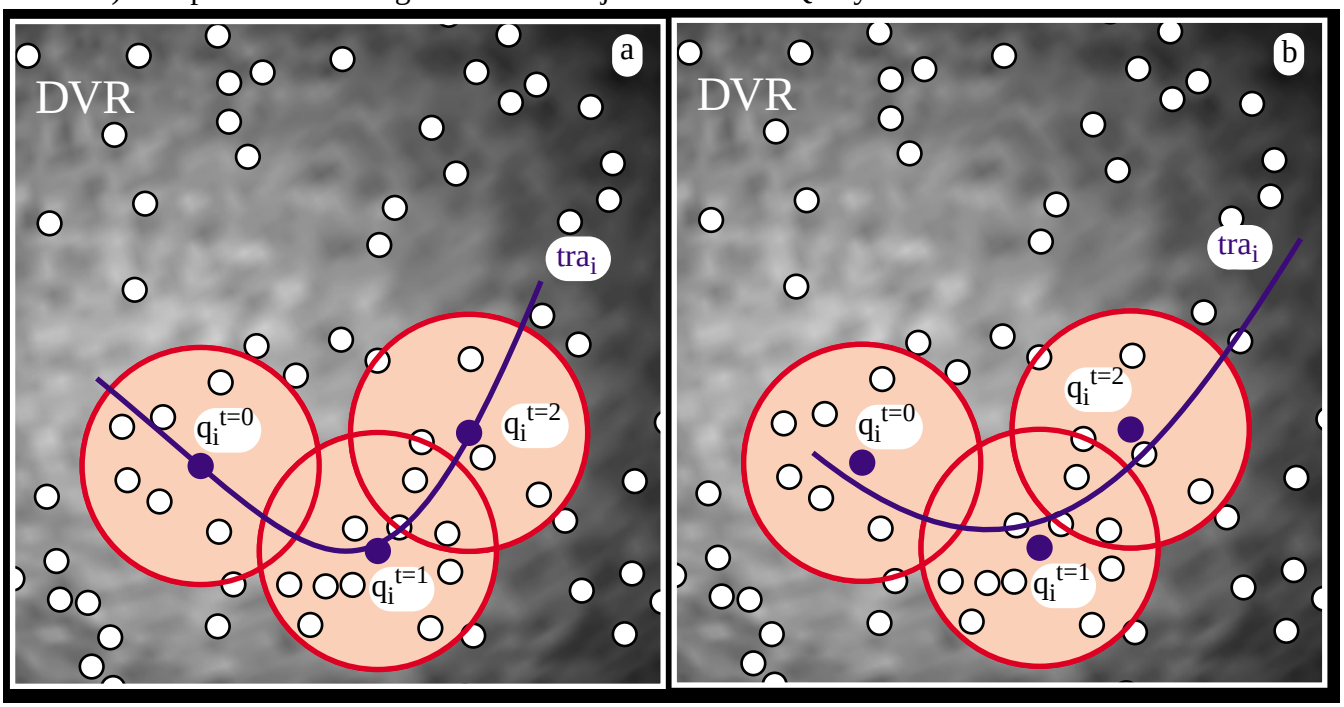
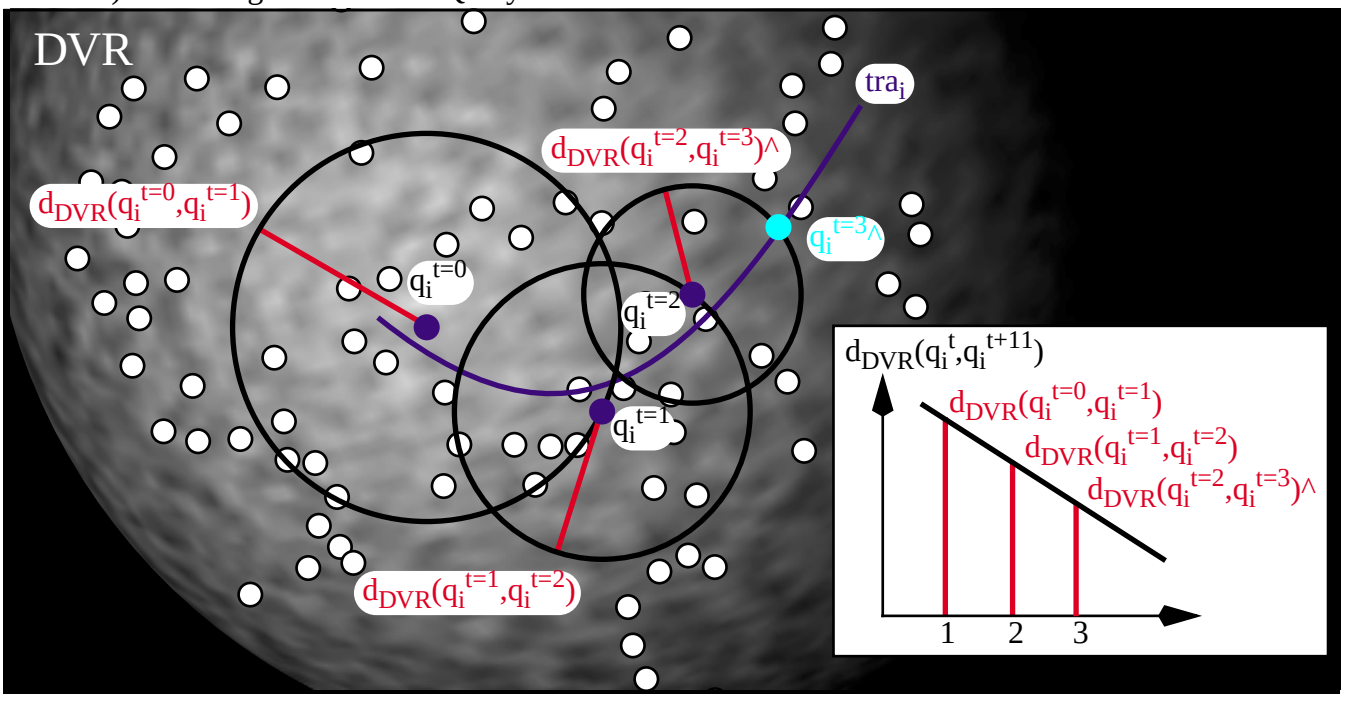
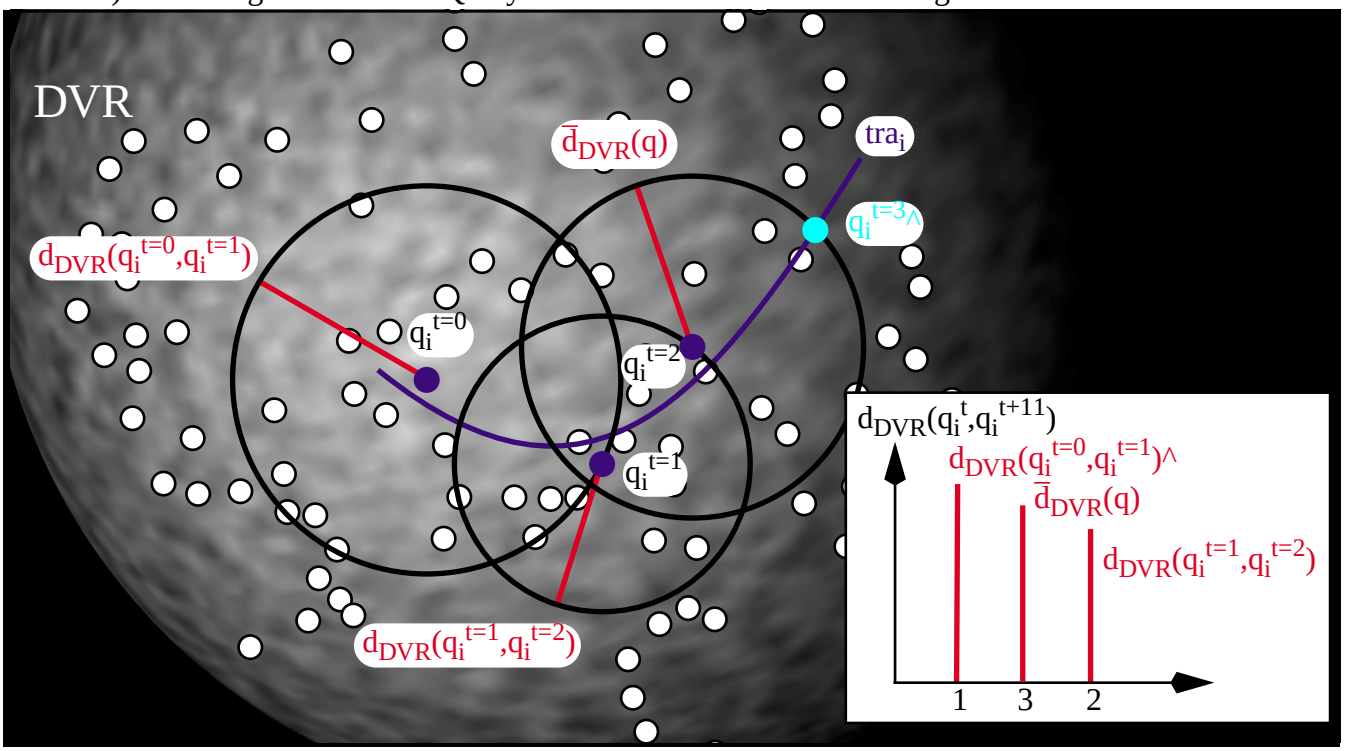


Abb. 87) Schätzung des nächsten Queryvektors durch monoton fallende Distanzwertfunktion



Alternativ könnte die durchschnittliche Distanz $\overline{d}_{DVR}(q)$ zweier nachfolgender Queryvektoren verwendet werden, um den nächsten Queryvektor zu schätzen. Hierzu wird um den letzten vorliegenden Queryvektor $q_i^{t=2}$ eine Hyperkugel mit dem Radius $\overline{d}_{DVR}(q)$ erzeugt. Die Queryvektorenkurve schneidet die Oberfläche dieser Hyperkugel in einem Punkt, der als Schätzung $q_i^{t=3\wedge}$ verwendet wird (siehe Abb. 88)).

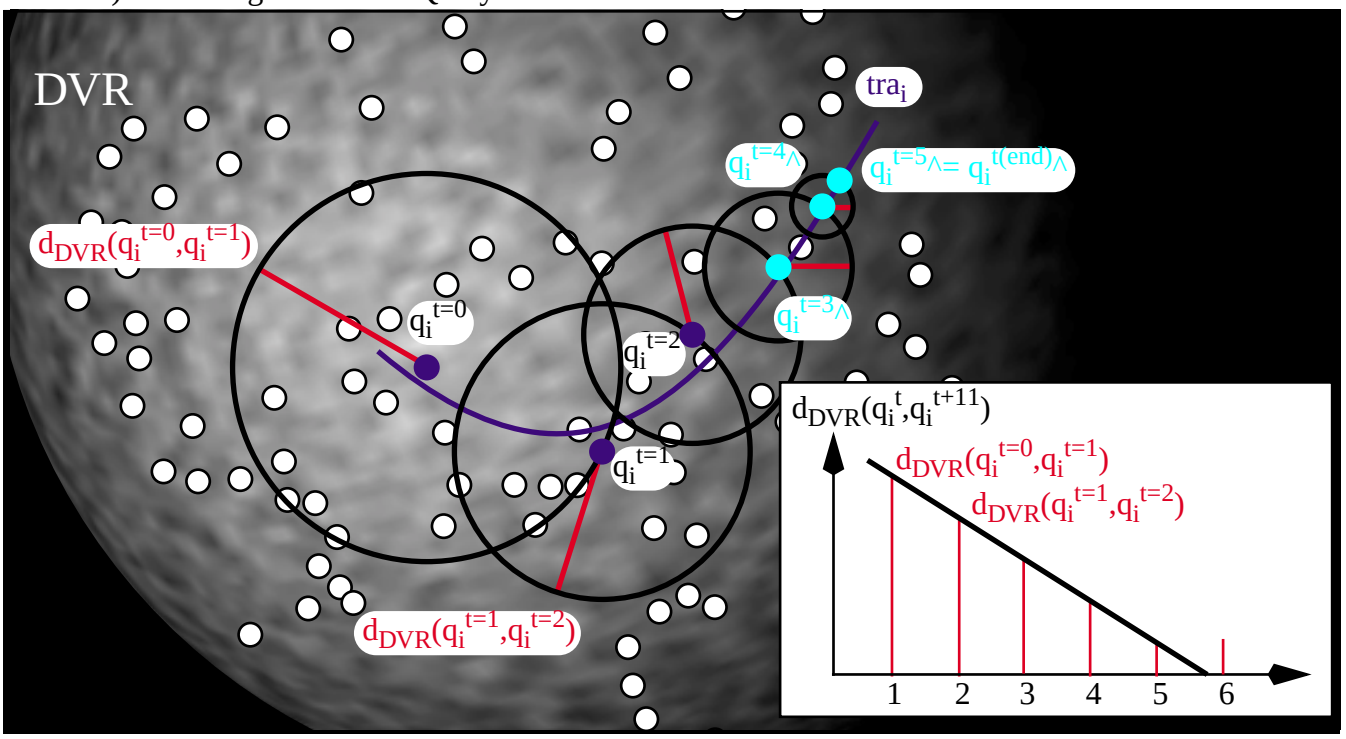
Abb. 88) Schätzung des nächsten Queryvektors durch Mittelwertsbildung



Von besonderem Interesse ist die bereits in Abb. 87) verwendete monoton fallende Distanzfunktion, die als lineare Funktion modelliert ist, da auf diese Weise eine Schnittstelle mit der Abszissenachse garantiert werden kann. Problematisch ist die Verwendung einer nicht-linearen, monoton fallenden Distanzfunktion, da dadurch der Schnittpunkt beliebig weit nach rechtsverlagert werden kann, was einem nicht realistischen Modell des Feedback-Prozesses entspricht. In diesem Fall werden sehr viele Iterationen vorhergesagt, die den Queryvektor weit von der bisherigen Region entfernen können, was im Gegensatz zu den wenigen Iterationen steht, die ein Agent bereit ist durchzuführen. Im Extremfall wäre auch eine Distanzfunktion modellierbar, die sich asymthotisch der Abszisse nähert, was einem Feedback-Prozess mit abzählbar unendlich vielen Iterationen entspricht.

Aus dieser Schnittstelle der unterstellten linearen Distanzfunktion kann auf den endgültigen Endpunkt der Queryvektoren-Trajektorie sowie auf die geschätzte Anzahl der Feedback-Iterationen geschlossen werden (siehe Abb. 89)). Ausgehend von der Schätzung $q_i^{t=3\wedge}$ werden auf der Kurve tra_i weitere Punkte spezifiziert, die den Schätzungen für $q_i^{t=4\wedge}$, $q_i^{t=5\wedge}$, ... entsprechen. Um aus $q_i^{t=3\wedge}$ die Schätzung $q_i^{t=4\wedge}$ zu erhalten, wird auf der monoton fallenden Distanzwertefunktion auf der Abszissenachse der Wert 4 mit dem zugehörigen Ordinatenwert $d_{DVR}(q_i^{t=3},q_i^{t=4})$ ermittelt. Um $q_i^{t=3\wedge}$ wird eine Hyperkugel mit dem Radius $d_{DVR}(q_i^{t=3},q_i^{t=4})$ erzeugt, die tra_i in $q_i^{t=4\wedge}$ schneidet. Diese Iterationen werden so lange durchgeführt, bis der nächste Distanzwert negativ wäre, was in dem betrachteten Beispiel bei einem Abszissenwert von 6 der Fall wäre. Aus diesem Wert folgt, dass die letzte Iteration den Wert von 5 besitzt, sodass $q_i^{t=5\wedge}$ als Schätzung des Endpunktes der Trajektorie verwendet werden kann, d.h. $q_i^{t=5\wedge} = q_i^{t(end)\wedge}$.

Abb. 89) Schätzung des letzten Queryvektors durch monoton fallende Distanzwertfunktion



3.9.3) Dokumentvektor-Relevanz-Feedback

3.9.3.1) Dokumentvektor-Feedback bei unklassifizierten Dokumentvektoren

War die Grundannahme beim Queryvektor-Relevanz-Feedback die mangelnde Fähigkeit der Formulierung einer Query, welche die Informationsbedürfnisse des Agenten adäquat beschreibt, so ist die Grundannahme beim Dokumentvektor-Relevanz-Feedback die potentiell vorliegende mangelnde Repräsentationsfähigkeit des IRS. Wird beim Queryvektor-Feedback die Relevanzbewertung dazu verwendet, um die Repräsentation „Queryvektor“ bei konstanten Dokumentvektoren in der Dokumentvektorenmenge DV zu adaptieren, so wird beim Dokumentvektor-Feedback die Relevanzbewertung dazu verwendet, um die Repräsentation „Dokumentvektor“ bei konstantem Queryvektor zu verändern.

Neben der Festlegung des Dokumentvektor-Feedback-Verfahrens ist von entscheidender Bedeutung, ob die Veränderungen der Dokumentvektorenverteilung temporär oder dauerhaft ist. Eine temporäre Veränderung wird nach dem Abbruch des Suchprozesses rückgängig gemacht, d.h. sie ist nur innerhalb einer Retrievalsitzung mit einem Agenten und seinem Informationsbedürfnis relevant. Eine dauerhafte Veränderung kann sich auf zwei Szenarien beziehen:
1) Veränderung für alle zukünftigen Sitzungen des gleichen Agenten.
2) Veränderung für alle zukünftigen Sitzungen beliebiger Agenten.

Die Veränderung in Abhängigkeit von einem Agenten führt dazu, dass die Dokumentvektorenstruktur, die sich nach einer oder mehreren Sitzungen ergibt, als ein Modell des Agenten interpretierbar wird. Bei einer gegebenen Menge von Dokumenten und einer gegebenen Menge dazu korrespondierender Dokumentvektoren ist dies als eine implizite, individuelle Indexierungsfunktion $A_{IR(D|Ag)}$ interpretierbar, d.h. als Modell wie der Agent Ag die Dokumente indexieren würde, bei einer gegebenen Menge von Merk-

malen, welche das Koordinatensystem von DVR bilden, und bei einer gegebenen Metrik in DVR. Implizit ist diese Indexierungsfunktion dadurch, dass bei einem neuen Dokument nicht direkt eine Position im DVR berechnet werden kann, sondern dass mit Hilfe der nicht-individuellen Funktion $A_{IR(D)}$ sowie lokaler, individueller Adaptionsoperationen im DVR eine Position erst erzeugbar wird. Wie aus einer impliziten Indexierungsfunktion eine explizite Funktion erzeugt werden kann, ist Gegenstand des Abschnittes über Indexierungsfunktions-Relevanz-Feedback (siehe Abschnitt 3.9.6)).

Gegen eine Veränderung für beliebige Agenten sprechen gewichtige Faktoren, da insbesondere die Reihenfolge der Retrievalsitzungen die Ergebnisse zukünftiger Sitzungen maßgeblich beeinflussen würde. Es existiert zudem im Vektorraum-Modell keine Repräsentationsform für Agentenziele, sodass Agenten mit unterschiedlichen und konträren Zielen Retrievalsitzungen durchführen können, ohne dass entsprechende Unterscheidungen möglich sind. Eine stetige Veränderung der Dokumentvektorenverteilung führt in diesem Szenario global betrachtet unweigerlich zu einer erhöhten Anzahl von Relevanz-Feedback-Operationen, d.h. die Anzahl der zu bewertenden Dokumentvektoren steigt, damit ein konkreter Agent eine vorgefundene, durch seine Vorgänger modifizierte Dokumentvektorenverteilung an seine individuellen Strukturen und Bedürfnisse anpasst. Eine effiziente Strategie wäre hier die Erzeugung einer Klassifikation von Modellen, ausgehend von einer gemeinsamen Dokumentvektorenverteilung.

In Salton (1971: 445[293]) (siehe Friedman et al. (1971[127]), Brauen (1971[57])) wird das Dokumentvektor-Relevanz-Feedback als Dokumentraum-Transformation bezeichnet, was aus mehreren Gründen eine falsche Bezeichnung ist. Zum einen finden dabei keine Veränderungen im Dokumentraum als dem Raum der Zeichensequenzen statt, sondern im Dokumentvektorenraum. Zum anderen handelt es sich bei den Operationen um keine Transformation eines Vektorraumes. Ein Vektorraum ist durch die Anzahl der Dimensionen und durch die verwendete Metrik spezifiziert, und keine der Operationen im Rahmen des Dokumentvektor-Relevanz-Feedback verändert einen dieser Faktoren. Die Operationen Dokumentvektor-Feedback verändern die Verteilung der Dokumentvektoren, und dies nicht gleichförmig, sodass es sich um lokale und nicht um globale Operationen handelt.

Nachfolgend soll das Verfahren von Friedman et al. (1971:449ff[127]) dargestellt werden, das von einer unklassifizierten Dokumentvektorenmenge DV ausgeht. Nachdem eine Query Q_i in den Queryvektor q_i indexiert wurde, wird durch eine Retrievalstrategie eine Dokumentvektorenmenge $DVM(q_i)^{t=0}$ aus der Grundmenge $DV(q_i)_{ges}^{t}$ ermittelt, die in der Initialisierungs-Iteration t=0 gleich der gesamten Dokumentvektorenmenge ist, d.h. $DV(q_i)_{ges}^{t=0} = DV$:

$$DVM(q_i)^{t=0} = \{x_{k|q(i)}^{t=0} \mid rel(x_{k|q(i)}^{t=0}) = 1, j = 1, ..., f^{t=0}\}. \quad (443)$$

Die nachgewiesenen Dokumente werden durch den Agenten bewertet, sodass sich die beiden Mengen $DVM(q_i)^{t=0}_{rel}$ und $DVM(q_i)^{t=0}_{\overline{rel}}$ ergeben:

$$\begin{aligned} DVM(q_i)^{t=0}_{rel} &= \{x_{l|q(i),rel}^{t=0} \mid rel(x_{k|q(i),rel}^{t=0}) = 1, l = 1, ..., f_{rel}^{t=0}\}, \\ DVM(q_i)^{t=0}_{\overline{rel}} &= \{x_{l|q(i),\overline{rel}}^{t=0} \mid rel(x_{k|q(i),\overline{rel}}^{t=0}) = 0, l = 1, ..., f_{\overline{rel}}^{t=0}\}, \\ f_{rel}^{t=0} + f_{\overline{rel}}^{t=0} &= f^{t=0}, f_{rel}^{t=0} \neq 0 \wedge f_{\overline{rel}}^{t=0} \neq 0. \end{aligned} \quad (444)$$

Es folgt die Berechnung der folgenden Werte für jedes Merkmal F_j:

1) Mittlere Summe der Gewichte des Merkmals F_j über den relevanten Dokumentvektoren $x_{l|q(i),rel}^{t=0}$:

$$r_{j|q(i),rel}^{t=0} = 1/f_{rel}^{t=0} * \sum_{l=1->f(rel,t=0)} x_{lj|q(i),rel}^{t=0}. \quad (445)$$

2) Mittlere Summe der Gewichte des Merkmals F_j über den irrelevanten Dokumentvektoren $x_{l|q(i),\overline{rel}}^{t=0}$:

$$n_{j|q(i),\overline{rel}}^{t=0} = 1/f_{\overline{rel}}^{t=0} * \sum_{l=1->f(\overline{rel},t=0)} x_{lj|q(i),\overline{rel}}^{t=0}. \quad (446)$$

3) Diskriminationswert des Merkmals F_j in der Iteration t=0:

$$d_{j|q(i),rel}^{t=0} = r_{j|q(i),rel}^{t=0} - n_{j|q(i),\overline{rel}}^{t=0}. \quad (447)$$

Im nächsten Schritt werden alle wichtigen Merkmale in der Iteration t=0 mit Hilfe des Diskriminationswertes ermittelt und in die Merkmalsliste $F_{q(i),disk}^{t=0}$ eingetragen. Es handelt sich dabei um die Merkmale $F_{j,disk}^{t=0}$ mit einem Diskriminationswert größer als ein konstanter Schwellenwert S_{disk}. Weiterhin sollen alle Merkmale enthalten sein, die in der Query explizit vorkommen:

$$F_{q(i),disk}^{t=0} = (F_{j|q(i),disk}^{t=0} \in F \mid d_{j|q(i),rel}^{t=0} > S_{disk}). \quad (448)$$

Für jeden der Merkmale in $F_{q(i),disk}^{t=0}$ werden die folgenden drei Parameter berechnet, welche die Wichtigkeit eines Merkmals im Queryvektor, im relevanten Dokumentvektor und im irrelevanten Dokumentvektor beschreiben:

1) Relative Gewicht des Merkmals $F_{j,disk}^{t=0}$ im Queryvektor q_i als Gewicht $q_{ij,disk}^{t=0}$ durch die Summe aller n Gewichte in q_i:

$$p(1)_{j|q(i),disk}^{t=0} = q_{ij,disk}^{t=0}/[\sum_{k=1->n} q_{ik,disk}^{t=0}]. \quad (449)$$

2) Relative, mittlere Gewicht des Merkmals $F_{j,disk}^{t=0}$ in den relevanten Dokumentvektoren als Summe der Gewichte des Merkmals F_j über den relevanten Dokumentvektoren $x_{l|q(i),rel}^{t=0}$ dividiert durch die Summe aller Gewichte in allen relevanten Dokumentvektoren:

$$p(2)_{j|q(i),disk}^{t=0} = [\sum_{l=1->f(rel,t=0)} x_{lj|q(i),rel}^{t=0}]/[\sum_{l=1->f(rel,t=0)} \sum_{k=1->n} x_{lk|q(i),rel}^{t=0}]. \quad (450)$$

3) Relative, mittlere Gewicht des Merkmals $F_{j,disk}^{t=0}$ in den irrelevanten Dokumentvektoren als Summe der Gewichte des Merkmals F_j über den irrelevanten Dokumentvektoren $x_{l|q(i),\overline{rel}}^{t=0}$ dividiert durch die Summe aller Gewichte in allen irrelevanten Dokumentvektoren:

$$p(3)_{j|q(i),disk}^{t=0} = [\sum_{l=1->f(\overline{rel},t=0)} x_{lj|q(i),\overline{rel}}^{t=0}]/[\sum_{l=1->f(\overline{rel},t=0)} \sum_{k=1->n} x_{lk|q(i),\overline{rel}}^{t=0}]. \quad (451)$$

Für alle Merkmale $F_{j|q(i),disk}^{t=0}$ aus der Liste $F_{q(i),disk}^{t=0}$ wird im nächsten Schritt der folgende Wert $T_{j|q(i),disk}^{t=0}$ in Abhängigkeit von den drei zuvor berechneten Parametern bestimmt:

$$T_{j|q(i),disk}^{t=0} = \begin{cases} \alpha_1 * p(1)_{j|q(i),disk}^{t=0} + \alpha_2 * p(2)_{j|q(i),disk}^{t=0}, & \text{wenn } d_{j|q(i),disk}^{t=0} > S_{disk}, \\ -\alpha_1 * p(3)_{j|q(i),disk}^{t=0}, & \text{sonst.} \end{cases} \quad (452)$$

Im letzten Schritt werden die Dokumentvektoren modifiziert, wobei zwei Teilmengen unterschieden werden:

1) Alle Dokumentvektoren $x_j^{t=0}$, außer den als irrelevant bewerteten, verändern ihre Position:

$$\forall x_j^{t=0} \in DV \setminus DVM(q_i)^{t=0}_{\overline{rel}}: x_j^{t=1} = (x_{jk}^{t=1} = x_{jk}^{t=0} + \Delta x_{jk}^{t=0} \mid k = 1, ..., n), \text{ mit} \quad (453)$$

$$\Delta x_{jk}^{t=0} = \begin{cases} x_{jk}^{t=0} * T_k^{t=0}, & \text{wenn } F_k^{t=0} \in F_{q(i),disk}^{t=0}, \\ 0, & \text{sonst.} \end{cases} \quad (454)$$

2) Alle irrelevanten Dokumentvektoren werden gleich dem Nullvektor gesetzt, wobei angenommen wird, dass alle Gewichte x_{jk}^t im Intervall [0, 1] definiert sind:

$$\forall x_j^{t=0} \in DVM(q_i)^{t=0}_{\overline{rel}}: x_j^{t=1} = (x_{jk}^{t=1} = 0 \mid k = 1, ..., n). \quad (455)$$

Die Setzung der irrelevanten Dokumentvektoren gleich dem Nullvektor wird interpretiert als Verschiebeoperation der Dokumentvektoren weg vom Queryvektor, wobei die Verschiebung maximal bezogen auf die Region $[0, 1]^n$ ist. Dies kann auch als Entfernung der irrelevanten Dokumentvektoren aus der Grundmenge interpretiert werden, sodass die Grundmenge $DV(q_i)_{ges}^{t=1}$ der nächsten Iteration t=1 sich ergibt durch:

$$DV(q_i)_{ges}^{t=1} = DV(q_i)_{ges}^{t=0} \setminus DVM(q_i)^{t=0}_{\overline{rel}}, \text{ bzw. allgemein}$$
$$DV(q_i)_{ges}^{t} = DV(q_i)_{ges}^{t-1} \setminus DVM(q_i)^{t-1}_{\overline{rel}}, \text{ mit } DV(q_i)_{ges}^{t=0} = DV. \quad (456)$$

Auf der Grundlage der neuen Grundmenge $DV(q_i)_{ges}^t$ wird die t'te Iteration durchgeführt, indem ausgehend von q_i und der gleichen Retrieval-Strategie eine neue Retrievalmenge $DVM(q_i)^t$ gebildet wird. Die neuen Dokumentvektoren, d.h. diejenigen, die bislang noch nicht in einer der Retrievalmengen $DVM(q_i)^k$, k = 0, 1, ..., t-1, enthalten waren, werden als Grundlage des Relevanz-Feedbacks verwendet, indem die korrespondierenden Dokumente dem Agenten zur Bewertung vorgelegt werden. Das Gesamtverfahren kann von Seiten des IRS abgebrochen werden, wenn bei einer Iteration keine neuen Dokumentvektoren mehr nachgewiesen werden können.

Die Modifikation aller nicht irrelevanten Dokumentvektoren kann als positive Verschieboperation mit q_i als Adaptionszentrum interpretiert werden, wobei jede der n Dimensionen im Dokumentvektorenraum DVR einen eigenen Betrag Δx_{jk}^t der Verschiebung zugeordnet bekommt, der davon abhängig ist, ob das zu dieser Dimension gehörende Merkmal in der Liste $F_{q(i),disk}^t$ enthalten ist. Ist es Element der Liste der wichtigen Merkmale, so wird eine Verschiebung parallel zu der betrachteten Koordinatenachse j im DVR in Abhängigkeit von der Wichtigkeit des Merkmals, operationalisiert durch $T_{j|q(i),disk}^t$, durchgeführt. Für alle anderen Dimensionen in DVR wird keine Verschiebung durchgeführt, operationalisiert durch $\Delta x_{jk}^t = 0$.

Prinzipiell stellt sich die Frage, warum Dokumentvektoren, die bereits als relevant oder irrelevant bewertet wurden, im Rahmen eines einzelnen Retrievalprozesses modifiziert werden sollten. Von Bedeutung sind zu einem Iterationszeitpunkt t nur diejenigen Dokumentvektoren, die noch nicht nachgewiesen und bewertet wurden. Unter diesem Gesichtspunkt sollten Dokumentvektor-Verschiebungen auf die jeweilige Restmenge beschränkt werden, mit:

$$DV(q_i)_{ges}^t = DV(q_i)_{ges}^{t-1} \setminus DVM(q_i)^{t-1}, \text{ mit } DV(q_i)_{ges}^{t=0} = DV. \quad (457)$$

3.9.3.2) Dokumentvektor-Feedback durch SOM-Adaption

Alternative Adaptionsstrategien könnten aus den SOM-Modellen abgeleitet werden, indem bewertete Dokumentvektoren als Stimuli für positive und negative Adaptionsprozesse verwendet werden. Wird eine nicht klassifizierte Dokumentvektorenmenge verwendet, so übernehmen z.B. die unbewerteten Dokumentvektoren, d.h. $DV(q_i)_{ges}^{t}$ die Rolle der Gewichtsvektoren, die ihre Positionen adaptieren. Anstatt alle bewertete Dokumentvektoren als Fixpunkte von Adaptionen zu verwenden, soll im weiteren jedoch der Fall betrachtet werden, dass Mittelwertsvektoren der jeweils relevanten bzw. nicht relevanten Dokumentvektorenmengen als Fixpunkte der Adaption betrachtet werden.

Da die Dokumentvektoren in diesem Szenario keinen Graphen bilden, kann auch keine GNG-SOM-Adaption durchgeführt werden, welche die Nachbarschaftsmenge $N(d_G=1)$ eines Gewinner-Dokumentvektors auswählt, und die zugehörigen Vektoren in Richtung des Adaptionszentrums verschiebt. Die Verwendung einer GNG-SOM-Adaptionsgleichung $x_k^{neu} = x_k^{alt} + \Delta x_k^{alt}$ ist jedoch möglich. Die Verwendung einer SOM-Adaptionsstrategie benötigt keine Analyse der Wichtigkeit von einzelnen Merkmalen, entsprechend der die Verschiebedistanz parallel zu der korrespondierenden Koordinatenachse in DVR bestimmt wird. Dies macht die Vorgehensweise zwar einfach, andererseits erfordert es die Einführung einer letztlich sehr aufwendigen Adaptionsphase, verursacht durch die große Anzahl von Dokumentvektoren ohne eine Strukturierung, wie sie eine SC-GNG-SOM bietet. Im Rahmen einer Klassifikation durch eine SC-GNG-SOM soll dies näher beschrieben werden (siehe Abschnitt 3.9.3.3)).

Um eine einheitliche Darstellung zu erreichen, soll der Fall dargestellt werden, dass bei einer Feedback-Iteration die beiden Mittelwertsvektoren als positives und negatives Adaptionszentrum zusammen mit jeweils einer Umgebung verwendet werden (siehe Abb. 90)). D.h. ausgehend von einem Queryvektor q_i, der über alle Dokumentvektoren-Feedback-Operationen konstant bleibt, wird in der ersten Iteration t=0 eine Ergebnismenge $DV(q_i)^{t=0}$ ermittelt, indem die ε-Umgebung $U(q_i \mid \varepsilon_i)$ gebildet und alle Dokumentvektoren innerhalb dieser Umgebung identifiziert werden:

$$DV(q_i)^{t=0} = \{x_{ik}^{t=0} \in DV \mid d_{DVR}(q_i, x_{ik}^{t=0}) < \varepsilon_i\}. \tag{458}$$

Die Dokumente, die zu den Vektoren aus $DV(q_i)^{t=0}$ korrespondieren, werden dem Agenten präsentiert, der diese bewertet, sodass die beiden Mengen $DV(q_i)_{rel}^{t=0}$ und $DV(q_i)_{\overline{rel}}^{t=0}$ gebildet werden können, welche die beiden Mittelwerte $\bar{s}_{DV(i,rel)}^{t=0}$ und $\bar{s}_{DV(i,\overline{rel})}^{t=0}$ liefern. Um den positiven Adaptionsfixpunkt $\bar{s}_{DV(i,rel)}^{t=0}$ wird eine ε-Umgebung $U(\bar{s}_{DV(i,rel)}^{t=0} \mid \varepsilon^+)$ erzeugt, die eine Dokumentvektorenmenge liefert, deren Elemente einer positiven Adaption unterzogen werden:

$$DV(U(\bar{s}_{DV(i,rel)}^{t=0} \mid \varepsilon^+)) = \{x_{ik+}^{t=0} \in DV \mid d_{DVR}(\bar{s}_{DV(i,rel)}^{t=0}, x_{ik+}^{t=0}) < \varepsilon^+\}. \tag{459}$$

Analog wird $\bar{s}_{DV(i,\overline{rel})}^{t=0}$ als negativer Adaptionsfixpunkt betrachtet, um den eine ε-Umgebung $U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-)$ erzeugt wird, die $DV(U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-))$ liefert, wobei die beiden Umgebungsparameter vereinfachend gleich gesetzt werden sollen, d.h. $\varepsilon^+ := \varepsilon^-$:

$$DV(U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-)) = \{x_{ik-}^{t=0} \in DV \mid d_{DVR}(\bar{s}_{DV(i,\overline{rel})}^{t=0}, x_{ik-}^{t=0}) < \varepsilon^-\}. \tag{460}$$

Die SOM-Adaptionsgleichung basiert darauf, dass ein Gewichtsvektor w_i in Richtung eines Stimulusvektos x_k verschoben wird, der als Fixpunkt der Adaption verwendet wird. Die Distanz der Verschiebung

wird als Δw_i bezeichnet, und berechnet sich als Bruchteil der Distanz zwischen w_i und x_k, wobei der Bruchteil durch einen Adaptionsparameter $\varepsilon \in [0, 1[$ angegeben wird. Die Richtung der Verschiebung wird durch eine Addition beschrieben, sodass sich ergibt:

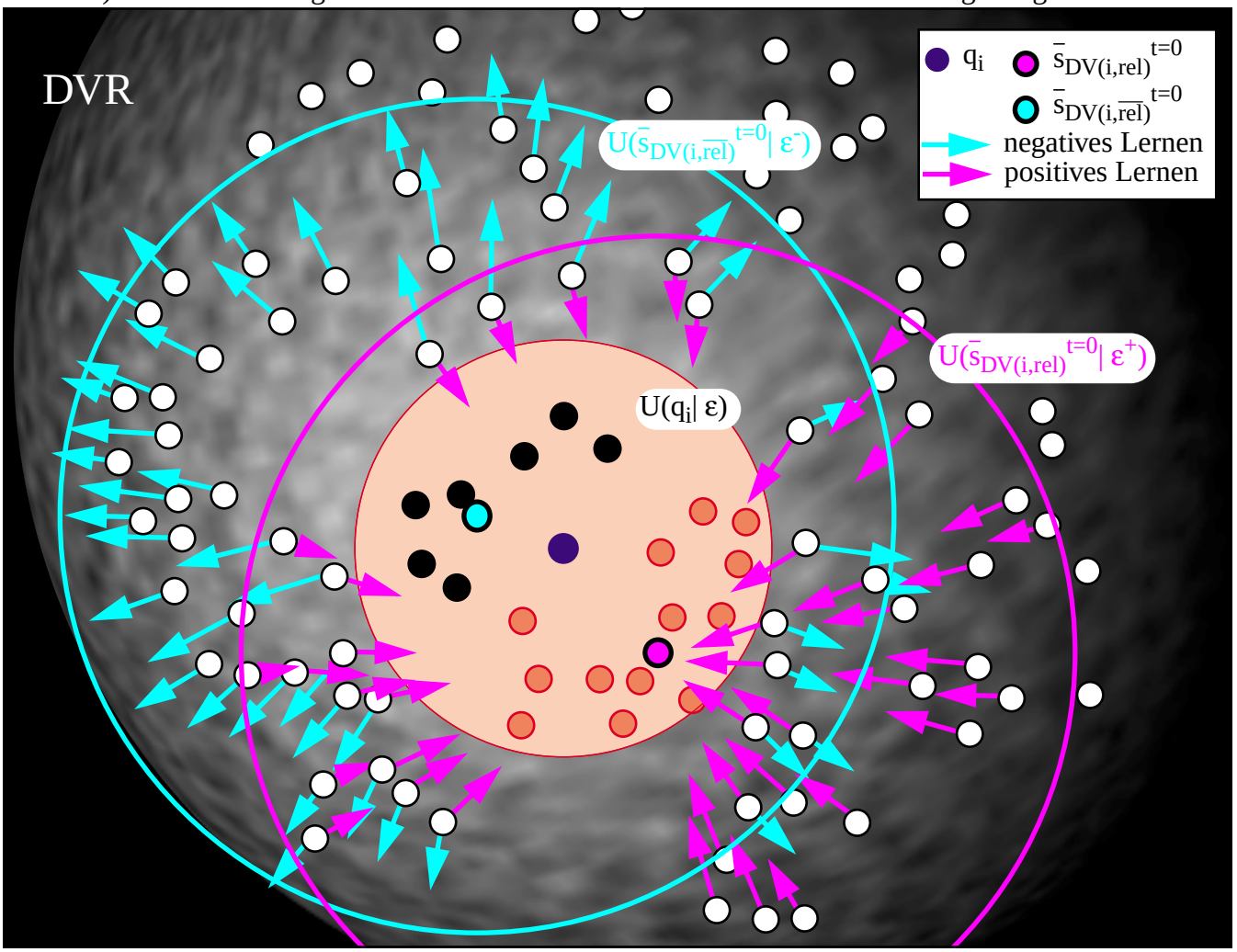
$$w_i^{neu} = w_i^{alt} + \Delta w_i^{alt}, \text{ mit}$$

$$\Delta w_i^{alt} = \varepsilon * d(w_i^{alt}, x_i) \text{ allgemein bzw.}$$

$$\Delta w_i^{alt} = \varepsilon * (w_i^{alt} - x_i) \text{ bei einer euklidischen Metrik.} \quad (461)$$

Die Verschiebung hin zu einem Fixpunkt kann als positive Adaption bezeichnet werden, während die Verschiebung weg von einem Fixpunkt als negative Adaption bezeichnet wird, die durch eine Subtraktion $w_i^{neu} = w_i^{alt} - \Delta w_i^{alt}$ operationalisiert wird.

Abb. 90) Positives und negatives Lernen von Dokumentvektoren innerhalb ε-Umgebungen 1



Werden diese Angaben über die SOM-Adaptionsgleichung auf den betrachteten Fall übertragen, so ist die positive Adaption problemlos zu beschreiben als:

$$x_{ik+}^{t=0,neu} = x_{ik+}^{t=0,alt} + \Delta x_{ik+}^{t=0,alt}, \text{ mit}$$

$$\Delta x_{ik+}^{t=0,alt} = \varepsilon_{adaption} * d_{DVR}(x_{ik+}^{t=0,alt}, \bar{s}_{DV(i,rel)}^{t=0}),$$

$$\forall x_{ik+}^{t=0} \in DV(U(\bar{s}_{DV(i,rel)}^{t=0} | \varepsilon^+)). \quad (462)$$

Wird der Mittelwertsvektor der nicht relevanten Dokumente der Iteration t=0 als Adaptionsfixpunkt einer negativen Adaption betrachtet, so werden alle Dokumentvektoren aus $DV(U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-))$ weg verschoben, wobei in beiden Fällen der gleiche Adaptionsparameter $\varepsilon_{adaption}$ unterstellt werden soll:

$$x_{ik-}^{t=0,neu} = x_{ik-}^{t=0,alt} - \Delta x_{ik-}^{t=0,alt}, \text{ mit}$$
$$\Delta x_{ik-}^{t=0,alt} = \varepsilon_{adaption} * d_{DVR}(x_{ik-}^{t=0,alt}, \bar{s}_{DV(i,\overline{rel})}^{t=0}),$$
$$\forall x_{ik-}^{t=0} \in DV(U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-)). \quad (463)$$

Wird die Retrievalstrategie in Form der Region $U(q_i \mid \varepsilon)$ im Verlauf des Dokumentvektoren-Relevanz-Feedbacks nicht verändert, d.h. bleibt der Queryvektor q_i sowie die Form und Größe der Region konstant, so bewirken ausschließlich negative Lernoperationen im Rahmen eines Feedback-Vorganges nichts, da die Dokumentvektoren, die ausschließlich von dem negativen Zentrum weg verschoben werden, nie in die Retrievalregion verschoben werden können, und somit nie nachgewiesen werden können. Relevant sind nur solche Operationen, die zu einer Verschiebung von Dokumentvektoren in die Region $U(q_i \mid \varepsilon)$ führen können. Es bietet sich daher an, diese Form der Dokumentvektoren-Adaption nur in Verbindung mit anderen Relevanz-Feedbackstrategien durchzuführen, bei denen sich die Position des Queryvektors verändert (Queryvektor-Relevanz-Feedback), und bei denen sich Form und Größe der Retrievalregion verändern (Retrievalregion-Relevanz-Feedback, siehe Abschnitt 3.9.5)).

Andererseits könnte bei der negativen Adaption argumentiert werden, dass $\bar{s}_{DV(i,\overline{rel})}^{t=0}$ als Repräsentantenvektor der nicht relevanten Dokumentvektoren ebenfalls eine positive Adaption verursachen soll, wenn unterstellt wird, dass die Dokumentvektoren aus $DV(U(\bar{s}_{DV(i,rel)}^{t=0} \mid \varepsilon^+))$ dem positiven Fixpunkt $\bar{s}_{DV(i,rel)}^{t=0}$ ähnlicher werden sollen, während die Dokumentvektoren aus $DV(U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-))$ dem negativen Fixpunkt $\bar{s}_{DV(i,\overline{rel})}^{t=0}$ ähnlicher werden sollen. Der oben dargestellte Fall macht die Elemente aus $DV(U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-))$ dem negativen Fixpunkt $\bar{s}_{DV(i,\overline{rel})}^{t=0}$ jedoch unähnlicher. In der Adaptionsgleichung würde dies die Verwendung der Addition bedeuten, bei Erhaltung der sonstigen Werte (siehe Abb. 91)):

$$x_{ik-}^{t=0,neu} = x_{ik-}^{t=0,alt} + \Delta x_{ik-}^{t=0,alt}. \quad (464)$$

Bei der Verwendung der positiven Adaptionsgleichung bezüglich eines positiven und eines negativen Fixpunktes ist der Fall sinnvoll anzuwenden, bei dem ausschließlich nicht relevante Dokumentvektoren in $DV(q_i)^{t=0}$ liegen, d.h. $DV(q_i)_{rel}^{t=0} = DV(q_i)^{t=0} \wedge DV(q_i)_{\overline{rel}}^{t=0} = \varnothing$ (siehe Abb. 92)). Hierbei können Dokumentvektoren, die außerhalb der Retrievalregion $U(q_i \mid \varepsilon)$ jedoch innerhalb der Region $U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-)$ liegen, in die konstant bleibende Retrievalregion verschoben werden, sodass diese Dokumentvektoren in der nachfolgenden Iteration Element der Ergebnismenge werden.

Abb. 91) Positives und negatives Lernen von Dokumentvektoren innerhalb ε-Umgebungen 2

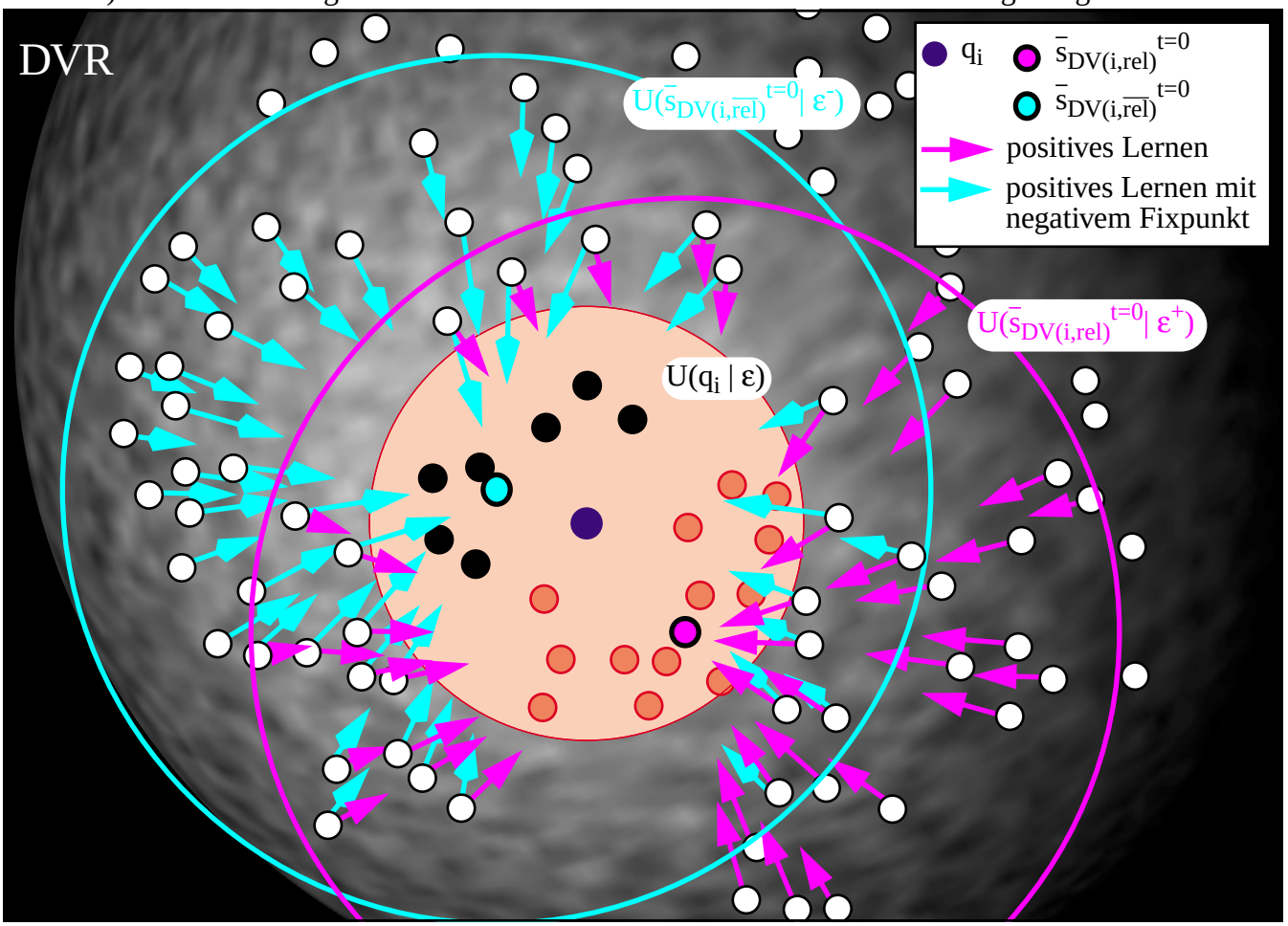
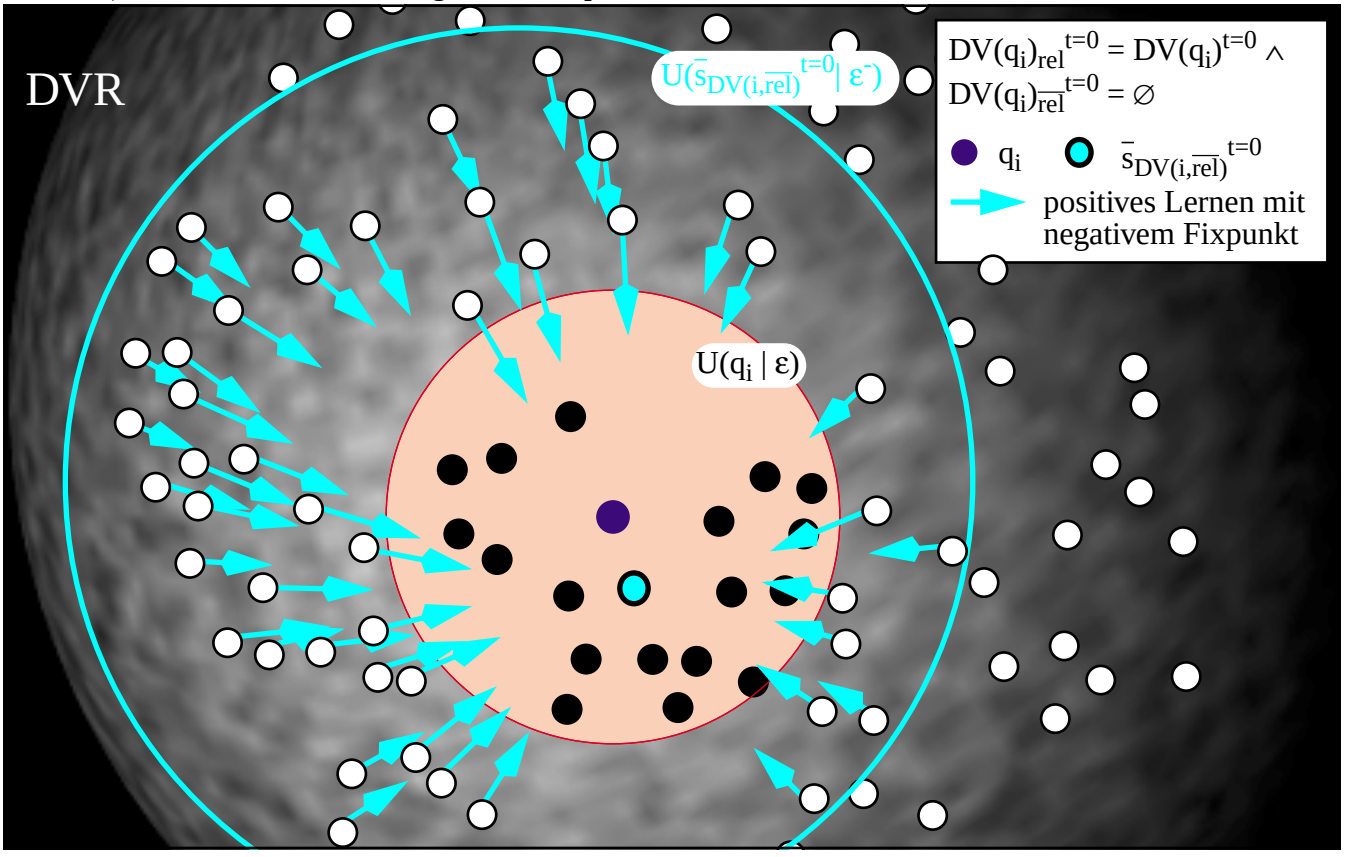


Abb. 92) Positives Lernen mit negativem Fixpunkt bei nicht relevanten Dokumentvektoren



Unzweideutig wäre die Adaption bei einem einzelnen Fixpunkt, wie z.B. die Verwendung des Mittelwertes $\bar{s}_{DV(i)}^{t=0}$ aller Dokumentvektoren aus $DV(q_i)^{t=0}$, doch ergibt sich in diesem Fall das Problem, dass eine positive Adaption eine negative Adaption aufheben würde, wenn die gleichen Dokumentvektoren und die gleichen Adaptionsparameter verwendet werden. D.h. es müssten entweder Umgebungen $U(\bar{s}_{DV(i,rel)}^{t=0} \mid \varepsilon^+)$ und $U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-)$ mit unterschiedlichen Umgebungsparametern $\varepsilon^+ \neq \varepsilon^-$ verwendet werden, oder es müssen unterschiedliche Adaptionsparameter $\varepsilon_{adaption}^{+} \neq \varepsilon_{adaption}^{-}$ verwendet werden, bzw. eine Kombination aus beiden Faktoren. Bezüglich der unterschiedlichen Parameter fehlen jedoch Kriterien um diese festzulegen, da begründet werden müsste, warum z.B. eine ε-Umgebung für die positive Adaption größer sein sollte als für die negative Adaption.

3.9.3.3) Dokumentvektor-Feedback bei SC-GNG-SOMs

Eine unklassifizierte Dokumentvektorenmenge besitzt, wie beschrieben, einen erheblichen Effizienznachteil im Zusammenhang mit einer SOM-Adaption bei einem Dokumentvektoren-Feedback, da sehr viele Dokumentvektoren adaptiert werden müssen, auch wenn eine Begrenzung durch Anwendungsregionen wie $U(\bar{s}_{DVM(i,\overline{rel})}^{t=0} \mid \varepsilon^-)$ und $U(\bar{s}_{DVM(i,rel)}^{t=0} \mid \varepsilon^+)$ eingeführt werden. Effizientere Strategien können im Zusammenhang mit einer klassifizierten Dokumentvektorenmenge gebildet werden, wobei die Verwendung einer SOM-Adaption für Feedbackprozesse die Verwendung einer SC-GNG-SOM für die Klassifizierung nahelegt.

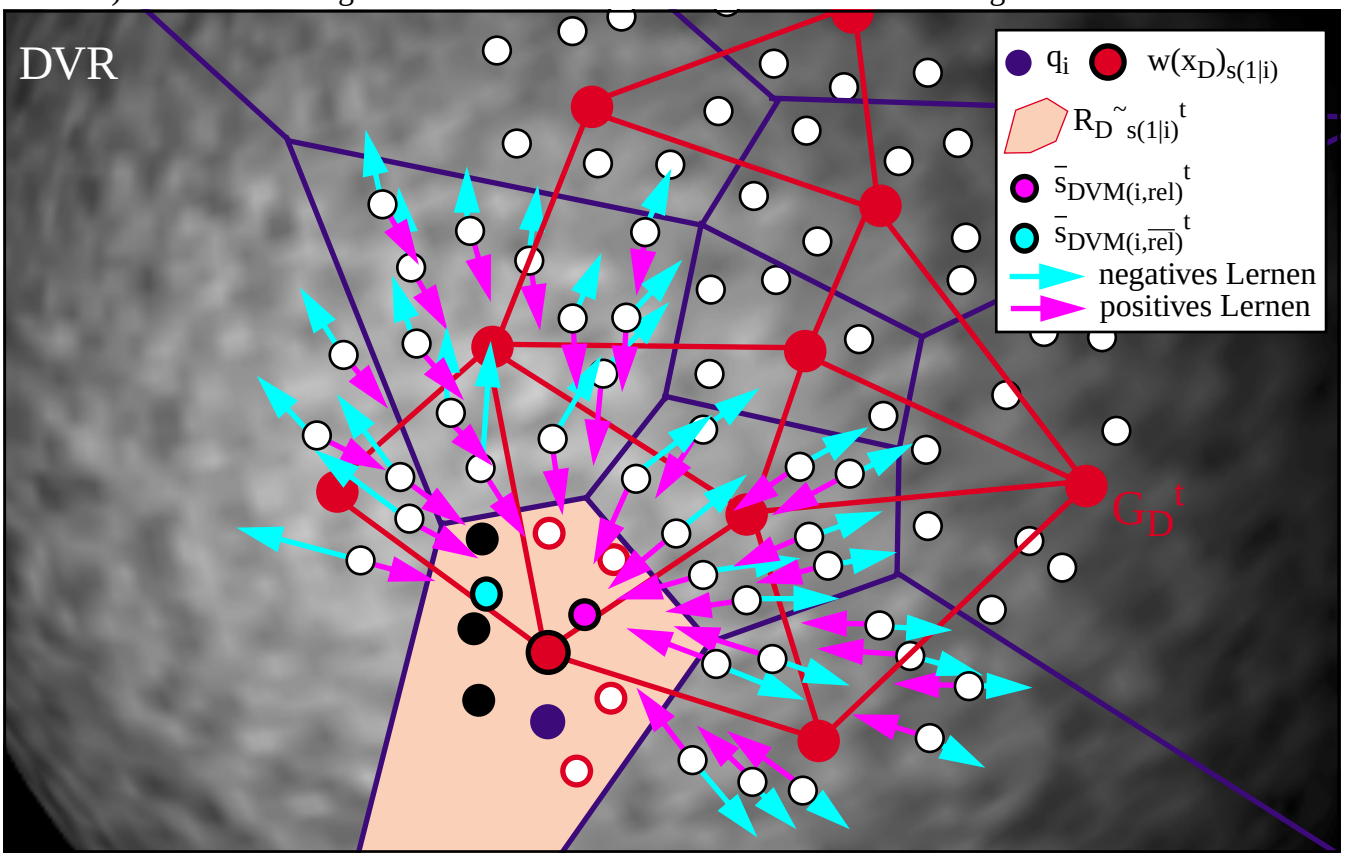
Zunächst soll der Fall betrachtet werden, dass q_i in der Voronoi-Region eines Neurons $n_{D,s(1|i)}$ liegt, wobei der fehlende Iterationsindex t bedeutet, dass kein Queryvektor-Feedback verwendet wird, sodass q_i immer in der Gewinner-Voronoi-Region $R_{D\tilde{}s(1|i)}^{t=0}$ liegen wird. Die Dokumentvektoren, die in der Gewinner-Voronoi-Region liegen, bilden die Retrievalmenge $DVM(q_i)^{t=0}$, deren korrespondierende Dokumente dem Agenten zur Bewertung vorgelegt werden. Entsprechend der Bewertung werden die beiden Mengen $DVM(q_i)^{t=0}_{rel}$ und $DVM(q_i)^{t=0}_{\overline{rel}}$ gebildet, aus denen die beiden Mittelwertsvektoren $\bar{s}_{DVM(i,rel)}^{t=0}$ und $\bar{s}_{DVM(i,\overline{rel})}^{t=0}$ ermittelt werden. Anstatt zwei Anwendungsregionen $U(\bar{s}_{DVM(i,\overline{rel})}^{t=0} \mid \varepsilon^-)$ und $U(\bar{s}_{DVM(i,rel)}^{t=0} \mid \varepsilon^+)$ um diese Mittelwertsvektoren zu definieren, sollen alle Dokumentvektoren einem positiven wie einem negativen Lernprozess unterzogen werden, die in den zu $R_{D\tilde{}s(1|i)}^{t=0}$ benachbarten Voronoi-Regionen liegen, d.h. es werden die Regionen betrachtet, deren Neurone in der Nachbarschaftsmenge $N(d_G=1 \mid G_D)_{s(1|i)}^{t=0}$ liegen.

Nach der Modifikation der betreffenden Dokumentvektoren können zwei Szenarien betrachtet werden:

1) Der Gewichtsvektorengraph G_D bleibt konstant.
2) Der Gewichtsvektorengraph wird entsprechend der neuen Dokumentvektorenverteilung adaptiert, sodass aus G_D^t ein neuer Graph G_D^{t+1} erzeugt wird.

Bleibt G_D konstant, so verändert sich auch die Voronoi-Region $R_{D\tilde{}s(1|i)}^{t=0}$ nicht, und dies unabhängig von der Anzahl der Iterationen des Relevanz-Feedbacks. Durch die Adaption der Dokumentvektoren, die nahe $\bar{s}_{DVM(i,rel)}^{t=0}$ und nahe der Voronoi-Grenze gelegen haben, können diese sich nun innerhalb von $R_{D\tilde{}s(1|i)}^{t=0}$ befinden, mit der Folge, dass die dazu korrespondierenden Dokumente in der nachfolgenden Iteration des Relevanz-Feedbacks dem Agenten zur Bewertung vorgelegt werden.

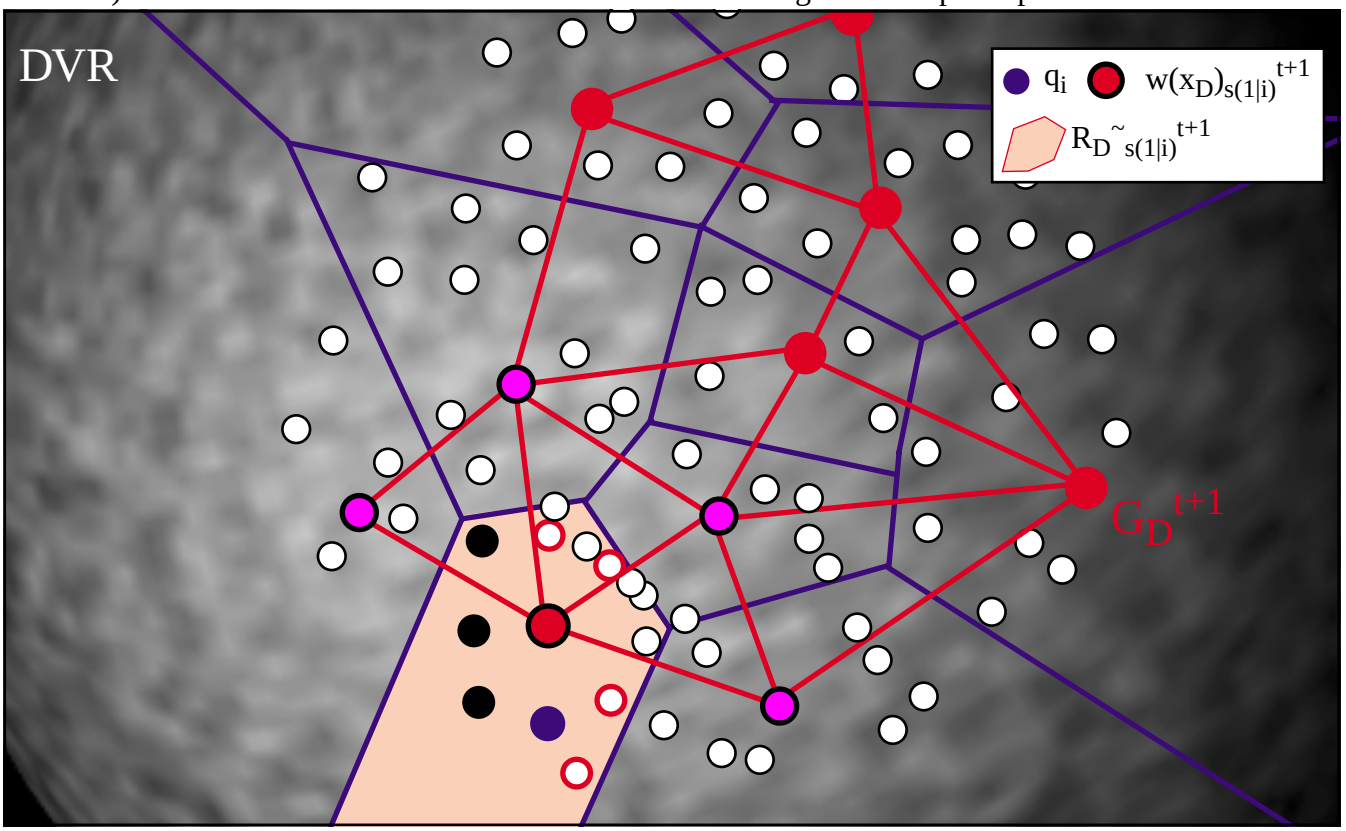
Abb. 93) Positives und negatives Lernen innerhalb benachbarter Voronoi-Regionen



Die Adaption der Dokumentvektoren kann aus der Perspektive der SC-GNG-SOM als Eliminierung alter Stimuli und der Erweiterung der Stimulusmenge um neue Elemente interpretiert werden. Die Gewichtsvektorenverteilung ist demgegenüber nicht mehr adäquat, sodass eine Nachadaptionsphase der Gewichtsvektoren durchgeführt werden muss (siehe Abb. 94)). Dabei werden zumindest alle Gewichtsvektoren der Neurone aus $N(d_G \leq 1 \mid G_D^t)_{s(1|i)}$ adaptiert, und die lokalen Stimulusmengen werden aktualisiert. Sollte ein Stimulusvektor eine starke negative Adaption durchgemacht haben, so könnte dies dazu führen, dass er sich in einer Voronoi-Region eines Neurons befindet, das nicht in der Nachbarschaftsmenge $N(d_G \leq 1 \mid G_D)_{s(1|i)}^t$ liegt. Auch in diesem Fall muss der Gewichtsvektor des betreffenden Neurons und seine lokale Stimulusmenge aktualisiert werden.

Durch die Adaption der Gewichtsvektoren verändern sich auch die zugehörigen Voronoi-Regionen, wobei die Veränderung von $\tilde{R}_{D\,s(1|i)}^{t}$ zur Folge hat, dass nicht mehr mit Sicherheit gesagt werden kann, dass der konstante Queryvektor in der Voronoi-Region wie in der vorangegangenen Iteration des Relevanz-Feedbacks liegt. Verändern sich die Voronoi-Regionen so stark, dass q_i in einer anderen Voronoi-Region liegt, so bedeutet dies bei einer Retrieval-Strategie, die auf den Voronoi-Regionen basiert, dass alle Dokumentvektoren der neuen Gewinner-Region die Retrievalmenge $DVM(q_i)^{t+1}$ bilden, und dass die korrespondierenden Dokumente dem Agenten präsentiert werden. Ein solcher Wechsel kann aber auch bedeuten, dass Dokumentvektoren, die in die alte Gewinner-Voronoi-Region verschoben wurden, nicht mehr in der neuen Retrievalmenge liegen. Bei einem Wechsel der Zugehörigkeit von q_i zu einer Voronoi-Region könnten die Stimuli, welche in die lokale Stimulusmenge des vorangegangenen Gewinner-Neurons neu aufgenommen wurden, durch eine Sonderregel in die aktuelle Retrievalmenge $DVM(q_i)^{t+1}$ aufgenommen werden.

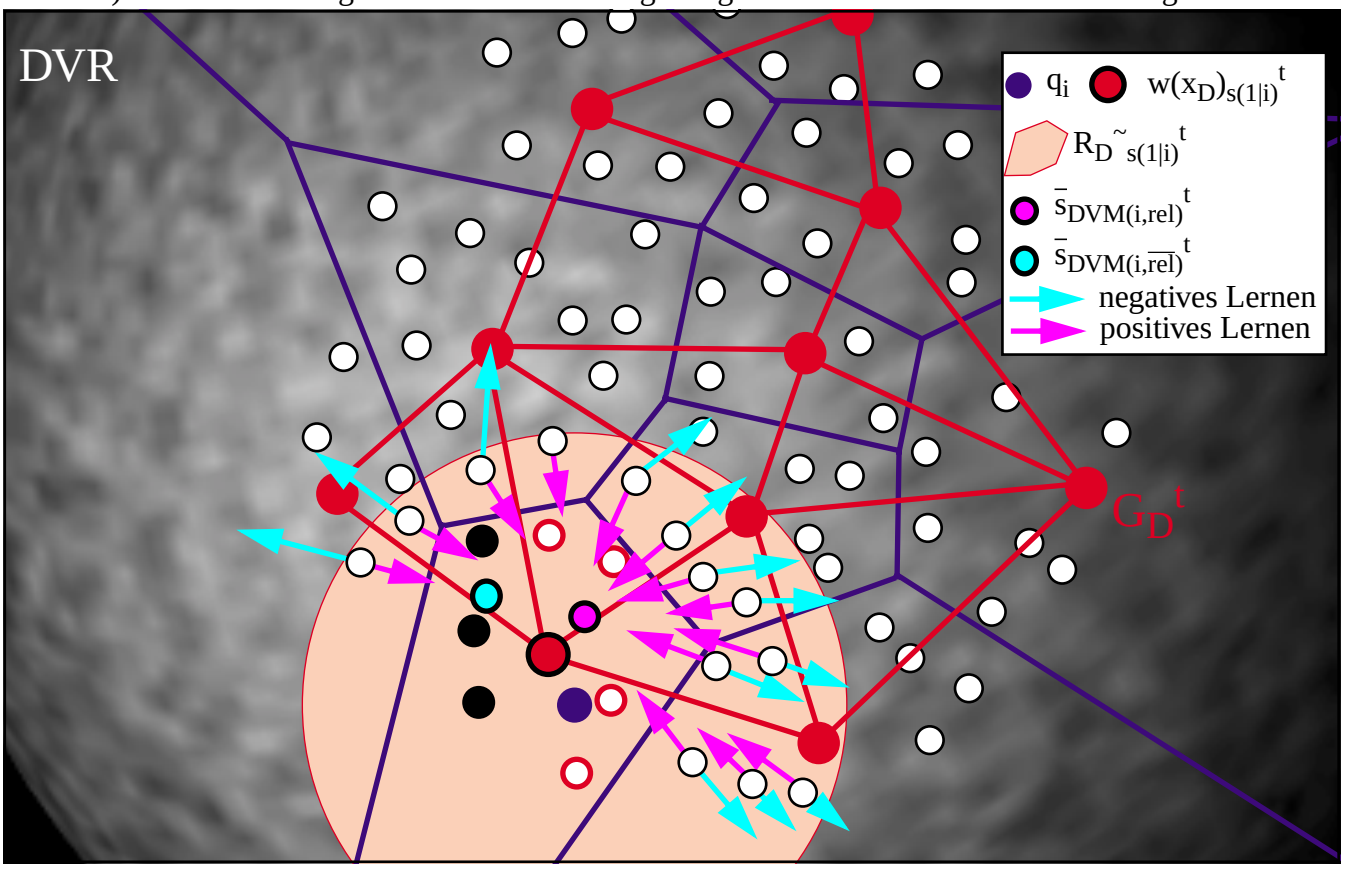
Abb. 94) Dokumentvektoren- und Gewichtsvektorenverteilung nach Adaptionsprozessen



Die Nachadaption zu $G_D{}^{t+1}$ kann unter dem Constraint durchgeführt werden, dass die Anzahl der Neurone konstant bleibt, da die Gesamtanzahl der Stimuli durch die Verschiebeoperation konstant bleibt. Andererseits kann dieser Constraint auch aufgehoben werden, sodass einzig ein Qualitätsmaß die Anzahl und Position der Gewichtsvektoren in $G_D{}^{t+1}$ bestimmt. In diesem Fall können ein oder mehrere Gewichtsvektoren hinzukommen oder entfernt werden, mit jeweils gravierenden Folgen für Voronoi-Regionen, die Zuordnung von q_i zu einer Voronoi-Region, und somit die Elemente der neuen Retrievalmenge in der Feedback-Iteration t+1. D.h. bei einer constraintlosen Nachadaption ergeben sich die meisten Veränderungspotentiale und Chancen, dass noch unbewertete Dokumentvektoren in die neue Retrievalmenge gelangen.

Wie beim Queryvektor-Relevanz-Feedback besteht auch bei Dokumentvektoren-Feedback die Möglichkeit eine Retrievalstrategie zu wählen, welche Nachbarschaften im Graphen $G_D{}^t$ mit einer ε-Umgebung kombiniert, d.h. die Elemente in der ε-Umgebung werden aus den Stimulusmengen der Neurone aus $N(d_G \leq 1 \mid G_D{}^t)_{s(1|i)}$ gewählt (siehe Abb. 95)). Alle Dokumentvektoren aus der Schnittmenge der ε-Umgebung und den Voronoi-Regionen der Neurone aus dieser Nachbarschaftsmenge werden einem positiven und einem negativen Adaptionsprozess unterzogen. Durch die ε-Umgebung werden die Adaptionsprozesse enger um q_i begrenzt, sodass kaum ein Dokumentvektor durch einen negativen Adaptionsprozess in eine weiter entfernte Voronoi-Region verschoben wird.

Abb. 95) Positives und negatives Lernen in ε-Umgebung und in benachbarten Voronoi-Regionen



Nach den Adaptionsprozessen der Dokumentvektoren kann wiederum eine Nachadaption der Gewichtsvektoren durch ein Feinadaptions-Verfahren der SC-GNG-SOM erfolgen, das auf lokalen Präsentationen, Stimulus-Umverteilungen und Summenadaptionen basiert. Es ergeben sich auch unter Beibehaltung der Anzahl der Neurone neue Positionen der Gewichtsvektoren, neue Voronoi-Regionen und neue lokale Stimulusmengen, sodass in der nächsten Iteration t+1 des Relevanz-Feedbacks wieder neue Dokumentvektoren in der ε-Umgebung um den Queryvektor liegen können, der selbst durch die veränderten Voronoi-Regionen einem anderen Gewinner-Neuron zugeordnet sein kann.

3.9.4) Gewichtsvektor-Relevanz-Feedback

Das allgemeine Problem der Verfahren zum Dokumentvektor-Relevanz-Feedback besteht in dem Speicheraufwand, wenn die sich ergebende Dokumentvektorenverteilung als Modell des Agenten gespeichert werden soll. Alle Dokumentvektoren, die ihre Position gegenüber dem nicht-individuellen Grundmodell verändert haben, müssen als n-dimensionaler Vektor gespeichert werden, während Dokumentvektoren, die ihre Position beibehalten haben, in dem individuellen Modell durch eine Referenz in die Ursprungs-Dokumentvektorenliste DV repräsentiert werden können. Im weiteren soll die Möglichkeit eines Gewichtsvektor-Relevanz-Feedback diskutiert werden, d.h. die Dokumentvektoren werden durch die Relevanzbewertungen nicht verändert, sondern die Gewichtsvektoren des Graphen G_D^t werden adaptiert. Dadurch werden die zugehörigen Voronoi-Regionen verändert, mit der Folge, dass sich die Zuordnung von Dokumentvektoren zu lokalen Stimulusmengen ebenfalls verändern. Je nach der verwendeten Retrievalstrategie kommen somit neue Dokumentvektoren in die Voronoi-Region des Gewinner-Neurons $n_{D,s(1|i)}^t$, deren korrespondierenden Dokumente in der nachfolgenden Iteration t+1 des Relevanz-Feed-

backs bewertet werden. Ein individuelles Modell eines Agenten besteht in diesem Kontext aus dem sich ergebenden individuellen Graphen $G_{D|Ag}^{t}$ bzw. der Neuronenstruktur $N_{D|Ag}^{t}$, die mit $\mu_{D|Ag}^{t}$ aus deutlich weniger Stützpunkten in DVR besteht, als eine Dokumentvektorenverteilung mit m Elementen. Die lokalen Stimulusmengen können vollständig durch Referenzen auf die Ursprungs-Dokumentvektorenliste DV repräsentiert werden, sodass die Speicherung eines individuellen Modells wesentlich effizienter durchgeführt werden kann.

Es soll zunächst der Fall betrachtet werden, dass als Retrieval-Region die Voronoi-Region $R_{D\tilde{}s(1|i)}^{t}$ des Gewinner-Neurons $n_{D,s(1|i)}^{t}$ betrachtet wird. Die Retrievalmenge $DVM(q_i)^t$ ergibt sich in diesem Szenario immer als die lokale Stimulusmenge $M_{D,s(1|i)}^{t}$ des Neurons $n_{D,s(1|i)}^{t}$. Nach der Relevanzbewertung können zwei Adaptionsstrategien angewendet werden:
1) Präsentation von Dokumentvektoren aus $R_{D\tilde{}s(1|i)}^{t}$.
2) Präsentation der beiden Mittelwertsvektoren $\bar{s}_{DVM(i,rel)}^{t}$ und $\bar{s}_{DVM(i,\overline{rel})}^{t}$.

Bei der Präsentation der Dokumentvektoren kann jedes Element aus $DVM(q_i)^t$ genau einmal zufällig präsentiert werden, d.h. es finden Zufallsziehungen ohne Zurücklegen statt, oder es findet eine Adaptionsphase auf der Basis der Gesamtmenge der Dokumentvektoren statt. Da jedoch sehr viele Dokumentvektoren vorliegen, müssten sehr viele Präsentationen und Adaptionen durchgeführt werden, sodass nur die Präsentation der Elemente aus $DVM(q_i)^t$ durchgeführt werden soll. Dies hat zur Folge, dass bei einer GNG-SOM-Adaption nur das Gewinner-Neuron und das zweite Element der Gewinnerliste adaptiert werden. Als alternative Adaptionsstrategie kann in diesem Kontext eine Adaption nach dem Neuronengas (NG-SOM) verwendet werden (Martinetz (1992[212]), Bachelier (1998a:23ff[15])), bei der prinzipiell alle Neurone an der Adaption beteiligt werden.

Bei der Verwendung der beiden Mittelwertsvektoren wird eine nicht-inkrementelle Strategie durch eine Summenadaption verfolgt, bei der für jeden Gewichtsvektor der positive und der negative Verschiebevektor berechnet wird (siehe Abb. 96)), die zu einem gemeinsamen Vektor aggregiert werden, der den Gewichtsvektor aus seiner Ursprungsposition innerhalb G_D^t verschiebt.

Durch die Verschiebung der Gewichtsvektoren verändern sich die Voronoi-Regionen, sodass manche Dokumentvektoren nun zu benachbarten Neuronen gehören (siehe Abb. 97)). In dem betrachteten Beispiel bleibt das Gewinner-Neuron bezüglich des Queryvektors q_i konstant.

Von Interesse sind die Dokumentvektoren, die neu in die Voronoi-Region des Gewinner-Neurons hinzukommen, da diese bei der Voronoi-Regionen-Strategie die Retrievalmenge $DVM(q_i)^{t+1}$ bilden, deren korrespondierende Dokumente dem Agenten in der Retrieval-Iteration t+1 zur Bewertung vorgelegt werden. Dieses Verfahren bricht ab, wenn nach einer Adaption der Gewichtsvektoren keine Wechsel von Dokumentvektoren in die aktuelle Voronoi-Region des Gewinner-Neurons erfolgt, bzw. wenn alle darin enthaltenen Dokumentvektoren bereits bewertet wurden. In einem solchen Fall kann anstelle des Abbruchs auf eine weniger restriktive Retrieval-Strategie umgeschaltet werden, die zusätzlich Elemente der benachbarten Voronoi-Regionen verwendet.

Abb. 96) Positives und negatives Lernen der Gewichtsvektoren

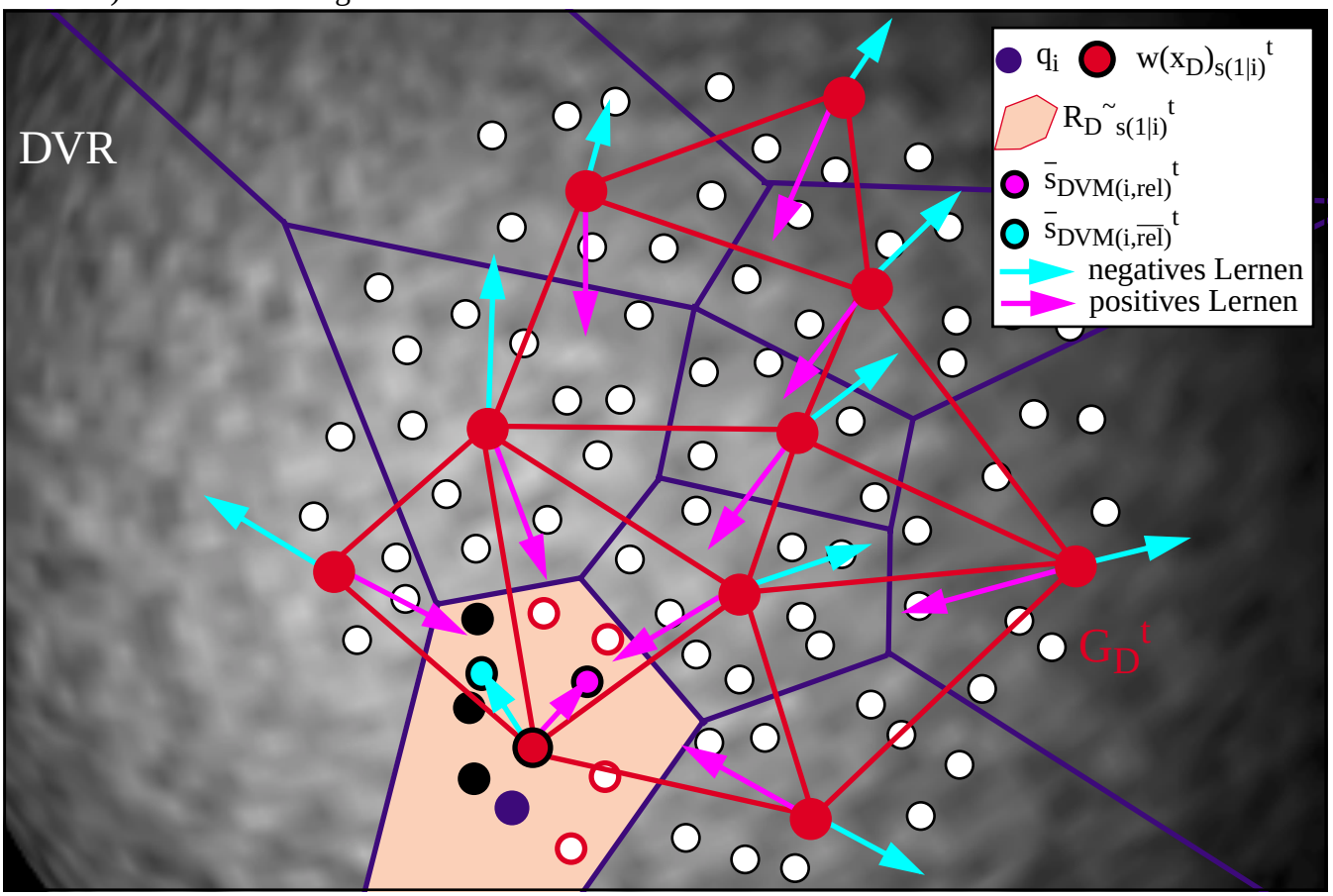
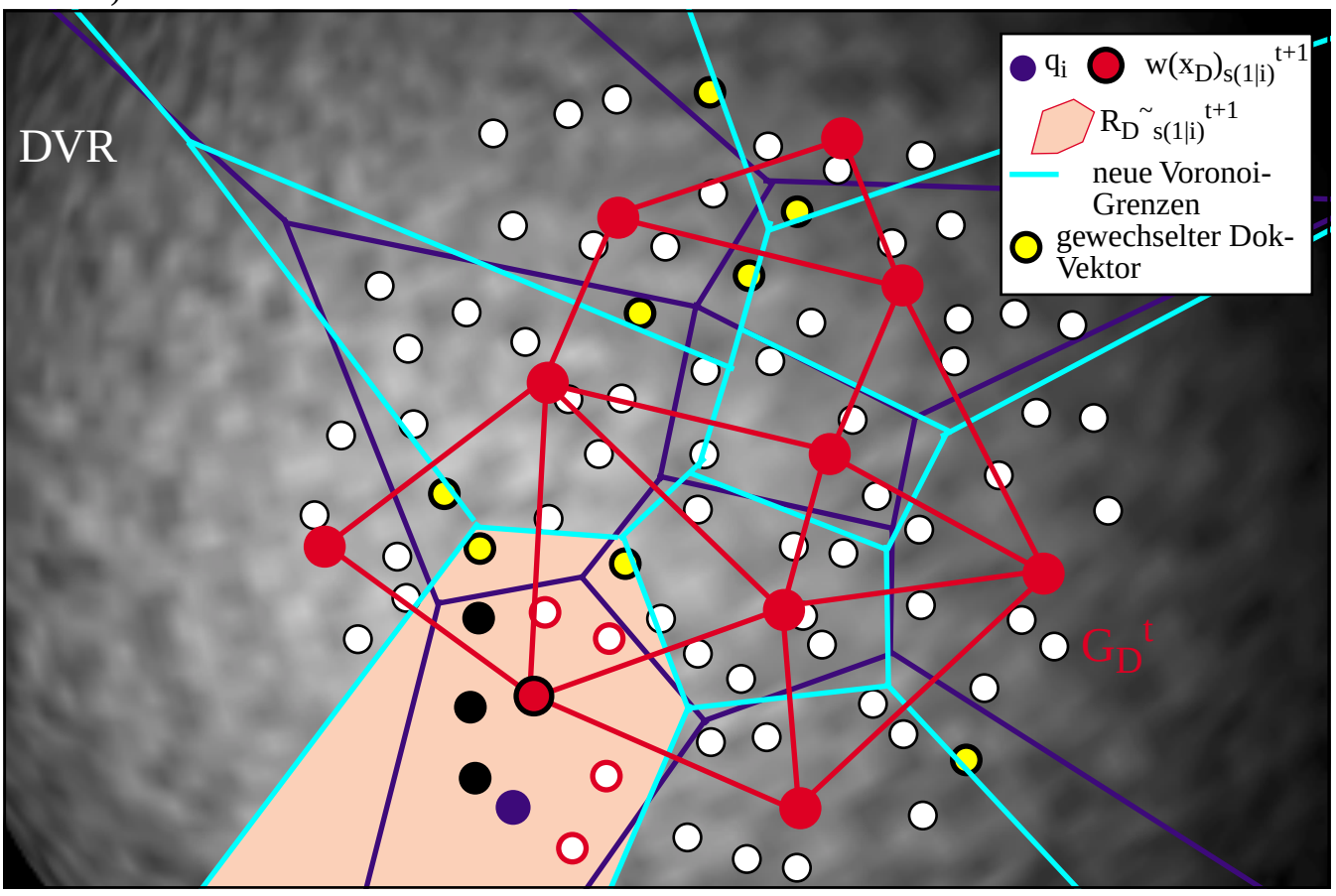


Abb. 97) Neue Voronoi-Grenzen und wechselnde Dokumentvektoren



3.9.5) Retrievalregion-Relevanz-Feedback

Betrachtet werden soll im weiteren Retrieval-Strategien im Dokumentvektorenraum, die auf der Basis von Mannigfaltigkeiten in DVR arbeiten, wie z.B. die einfache ε-Umgebung $U(q_i \mid \varepsilon)$ um ein Zentrum wie dem Queryvektor q_i. D.h. eine Retrieval-Strategie soll zum einen durch die Region aus DVR spezifiziert werden, in der nach Dokumentvektoren gesucht wird (Retrieval-Region), zum anderen durch die Art der Nachbereitung der gefundenen Dokumentvektoren, wie z.B. die Sortierung oder die Beschränkung auf eine maximale Anzahl von Dokumentvektoren. Bislang wurde in den Relevanz-Feedback-Strategien die binäre Bewertung der Elemente in der Retrieval-Dokumentvektorenmenge $DVM(q_i)^t$ verwendet, um Vektoren in DVR zu verschieben, wobei es sich um den Queryvektor, Dokumentvektoren bzw. Gewichtsvektoren der SC-GNG-SOM-Neurone handelte. In diesem Abschnitt sollen Möglichkeiten diskutiert werden, wie mit den Relevanz-Bewertungen Retrievalregionen modifiziert werden können.

Es ist zunächst festzustellen, dass bei einigen der Vektor-Relevanz-Feedback-Verfahren bereits eine Retrievalregion-Modifikation auftritt. Wird bei einem Queryvektor-Feedback eine ε-Umgebung $U(q_i^t \mid \varepsilon)$ mit $\varepsilon \in R^+$, als Retrievalregion verwendet, so ist mit der Veränderung des Queryvektors zu q_i^{t+1} auch eine Veränderung der Umgebung zu $U(q_i^{t+1} \mid \varepsilon)$ verbunden. Da diese Form der Retrievalregion durch die zwei Parameter q_i und ε definiert ist, folgt aus der Veränderung eines der Parameter eine Veränderung der Retrievalregion. Ein anderes Beispiel sind Mannigfaltigkeiten wie eine Voronoi-Region $R_{D\tilde{}_j}^t$, die durch die Knoten und Kanten des Graphen G_D^t definiert ist. Verändert sich die Position eines Gewichtsvektors, wird eine Kante gelöscht oder kommt eine hinzu, oder wird ein neuer Knoten in die Graphenstruktur eingefügt, so kann dies zu einer Veränderung jeder der vorhandenen Voronoi-Regionen führen, sodass durch die Gewichtsvektoren-Veränderung sich eine Veränderung der Retrieval-Voronoi-Region ergibt.

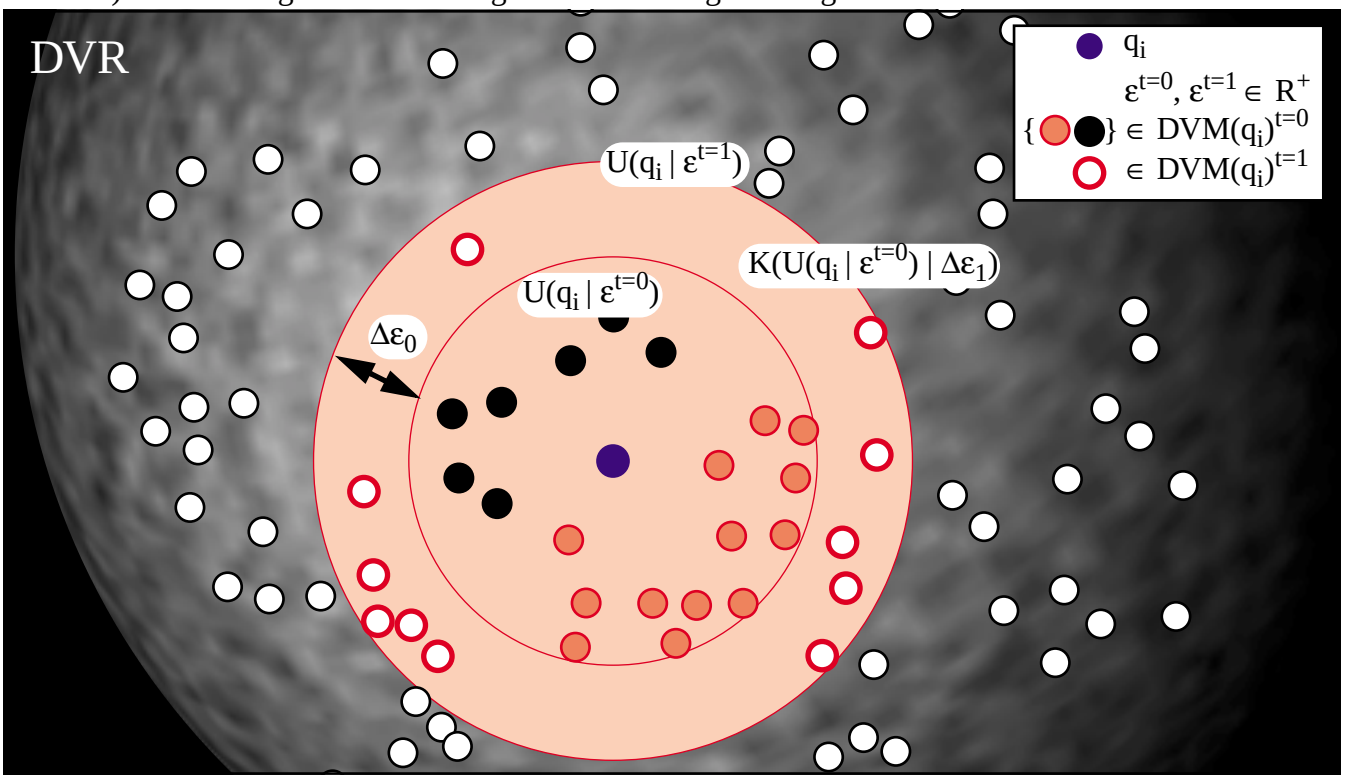
Die weiteren Darstellungen sollen sich auf ε-Umgebungen beschränken, wobei zunächst die Möglichkeiten der Veränderungen einer ε-Umgebung beschrieben werden, gefolgt von Methoden, wie diese Veränderungen infolge der Relevanz-Bewertungen durchgeführt werden.

Bei einem konstanten Queryvektor q_i kann die Retrievalregion nur durch die Veränderung des Umgebungsparameters ε erfolgen, der in den bislang betrachteten Umgebungen als Element aus R^+ betrachtet wurde. Wird die Initialisierungsregion $U(q_i \mid \varepsilon^{t=0})$ betrachtet, so können neue Dokumentvektoren ausschließlich durch eine ε-Vergrößerung ermittelt werden, d.h. $\varepsilon^{t=0} < \varepsilon^{t=1}$. Eine Vergrößerung erfolgt um $\Delta\varepsilon_1$, d.h. $\varepsilon^{t=1} = \varepsilon^{t=0} + \Delta\varepsilon_0$ (siehe Abb. 98)).

Für die Iteration t ist die Umgebung $U(q_i \mid \varepsilon^{t-1})$ nicht mehr relevant, da alle darin liegenden Dokumentvektoren nachgewiesen und bewertet wurden. Eine mehrfache Präsentation und Bewertung soll ausgeschlossen werden, da vereinfachend von einer stationären Relevanzfunktion des Agenten während einer Feedbacksitzung ausgegangen wird. Aus diesem Grunde interessiert nicht die ganze Region $U(q_i \mid \varepsilon^t)$ sondern nur die neue Region, die als äusserer $\Delta\varepsilon_1$-Kragen um $U(q_i \mid \varepsilon^{t-1})$ oder als Differenzregion von $U(q_i \mid \varepsilon^t)$ und $U(q_i \mid \varepsilon^{t-1})$ beschrieben werden kann:

$$\begin{aligned} K(U(q_i \mid \varepsilon^{t-1}) \mid \Delta\varepsilon_t) &= \{x \in DV\ R \mid \varepsilon^{t-1} < d_{DVR}(x, q_i) < \varepsilon^{t-1} + \Delta\varepsilon_0 = \varepsilon^t\} \\ &= U(q_i \mid \varepsilon^t) \setminus U(q_i \mid \varepsilon^{t-1}). \end{aligned} \quad (465)$$

Abb. 98) Veränderung der Retrievalregion durch ε-Vergrößerung



Die Verwendung eines Deltawertes führt dazu, dass bei einem konstanten Wert $\Delta\varepsilon_{t-1} = \Delta\varepsilon_t$ das Volumen der untersuchten Kragenregion stark zunimmt, da diese Operationen in einem n-dimensionalen Raum stattfindet. Auf diese Weise wird die Wahrscheinlichkeit, dass mehr Dokumentvektoren gefunden werden erhöht. Anstatt der Verwendung eines Deltawertes kann auch das Volumen einer Kragenregion $K(U(q_i \mid \varepsilon^{t-1}) \mid \Delta\varepsilon_t)$ als abhängige Variable von den Relevanzbewertungen der Iteration t-1 oder vorangegangener Iterationen geregelt werden. Aus dem Volumen wird dann der Deltawert $\Delta\varepsilon_t$ berechnet. Vereinfachend soll im weiteren von einer direkten Regelung des Deltawertes $\Delta\varepsilon_{t+1}$ von dem Relevanzniveau der beiden Vorgängeriterationen t und t-1 ausgegangen werden, sodass $\varepsilon^{t=0}$ und $\Delta\varepsilon_{t=0}$ gegeben sein müssen, und erst die dritte Region bzw. der zweite Kragen wird durch Relevanzbewertungen geregelt. Das Relevanzniveau kann im einfachsten Fall durch den Relevanzmittelwert oder komplexer durch eine Relevanz-Verteilung beschrieben werden, was jedoch erst bei mehr als zwei Relevanzwerten interessant wird. Unabhängig, wie die Dominanz von zwei Relevanz-Verteilungen definiert sein soll (s.a. Bachelier (1999b[20])), es ergeben sich drei Szenarien, aus denen $\Delta\varepsilon_{t+1}$ und somit ε^{t+2} hergeleitet werden muss:

1) Das Relevanzniveau hat sich in t gegenüber t-1 verbessert.
2) Das Relevanzniveau hat sich in t gegenüber t-1 verschlechtert.
3) Das Relevanzniveau ist konstant geblieben.

Bei einer Verbesserung kann argumentiert werden, dass eine weitere Vergrößerung der Retrievalregion gute Dokumentvektoren liefern könnte. Andererseits vergrößert sich der Abstand von Dokumentvektoren immer weiter zum Queryvektor q_i, sodass allgemein schlechter bewertete Dokumentvektoren erwartet werden müssen. Der Nachweis ausschließlich nicht-relevanter, neuer Dokumentvektoren in einer Kragenregion $K(U(q_i \mid \varepsilon^t) \mid \Delta\varepsilon_t)$ kann demgegenüber als Kriterium für den Abbruch des Relevanz-Feedbacks führen, d.h. $\Delta\varepsilon_{t+1}$ wird gleich Null gesetzt. Eine Verschlechterung des Relevanzniveaus kann zur Verkleinerung des nächsten Deltawertes verwendet werden.

Geregelt werden muss der Fall, dass bei einem berechneten Deltawert eine Kragenregion erzeugt wird, in der kein Dokumentvektor liegt, dessen korrespondierendes Dokument präsentiert werden könnte. Als Konsequenz würde der Feedbackabbruch stehen, was jedoch durch eine Sonderregel verhindert werden soll, indem eine größere Kragenregion erzeugt wird, d.h. als Mindestanforderung muss jeweils eine Kragenregion erzeugt werden, in der mindestens ein neuer Dokumentvektor liegt. Wird in einer gegebenen Kragenregion kein Dokumentvektor gefunden, so wird einfach der oder die zu q_i am nächsten liegenden Dokumentvektoren ausgewählt, die in keiner der bislang erzeugten Regionen liegen. D.h. ist eine Kragenregion $K(U(q_i \mid \varepsilon^t) \mid \Delta\varepsilon_t)$ leer, so wird die Dokumentvektorenrestmenge $DV \setminus \cup_{k=0->t} DVM(q_i)^k$ als Grundmenge für den am nächsten liegenden Dokumentvektor $x_{\Delta\varepsilon t}$ verwendet:

$$d_{DVR}(x_{\Delta\varepsilon t}, q_i) = \min\{d_{DVR}(x, q_i) \mid \forall x \in DV \setminus \cup_{k=0->t} DVM(q_i)^k\}. \quad (466)$$

Daraus folgt die Umgebung $U(q_i \mid \varepsilon^{t+1} = d_{DVR}(x_{\Delta\varepsilon t}, q_i)+\varepsilon)$, mit ε als einem kleinen positiven Wert, der durch die Definition einer Umgebung als offene Menge notwendig wird, um den Objektvektor $x_{\Delta\varepsilon t}$ als Element der Umgebung zu gewinnen. Die neue Kragenregion ergibt sich als Differenz aus dieser Umgebung mit der vorangegangenen Umgebung $U(q_i \mid \varepsilon^t)$:

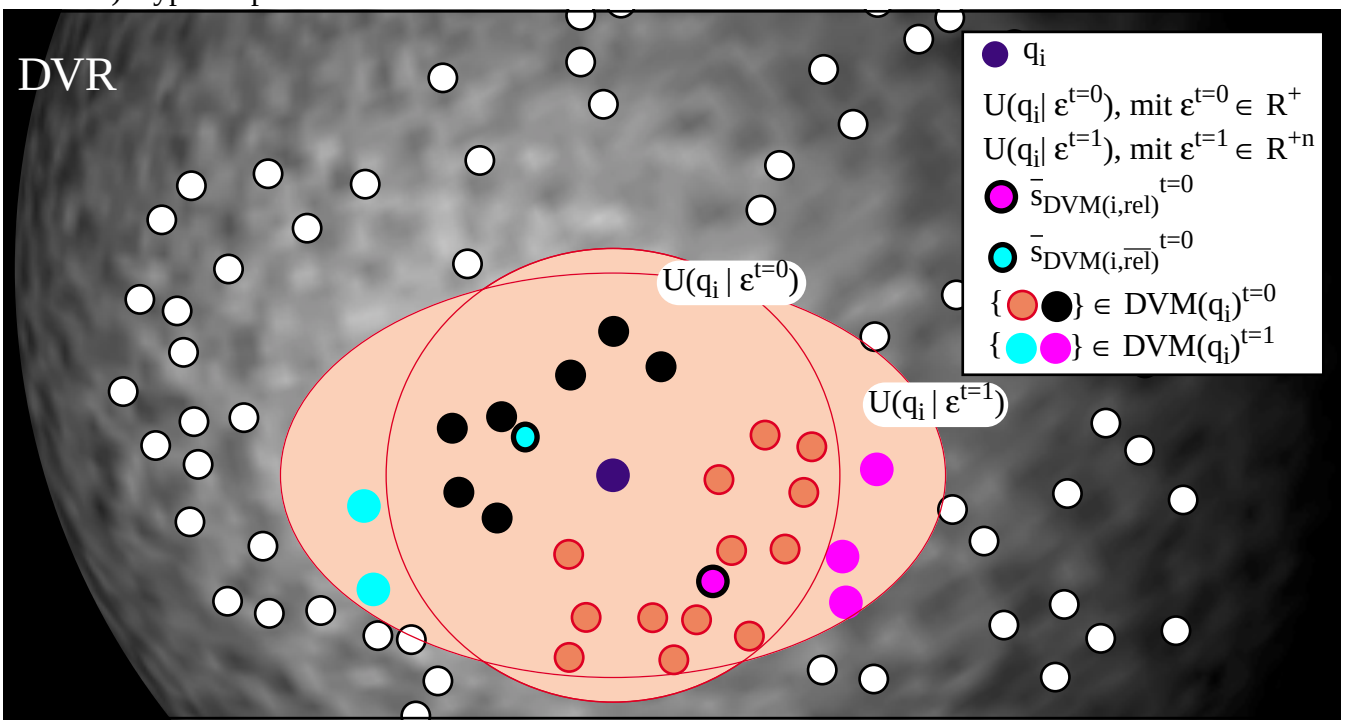
$$K(U(q_i \mid \varepsilon^t) \mid \Delta\varepsilon_t) := U(q_i \mid \varepsilon^{t+1} = d_{DVR}(x_{\Delta\varepsilon t}, q_i)+\varepsilon) \setminus U(q_i \mid \varepsilon^t). \quad (467)$$

Sollen mehrere neue Dokumentvektoren gewonnen werden, so wird die neue Umgebung durch die Distanz des am weitesten entfernten dieser Dokumentvektoren festgelegt.

Wird der Constraint $\varepsilon^t \in R^+$ aufgegeben, so kann ein Übergang von einer Hyperkugel im DVR zu einem Hyperellipsoid erfolgen, dessen n Achsen parallel zu den Achsen des Koordinatensystems verlaufen. Es kann z.B. in der Initialisierungs-Iteration ein Umgebungsparameter $\varepsilon^{t=0} \in R^+$ verwendet werden, wobei die Hyperkugel als ein Spezialfall eines Hyperellipsoides interpretiert wird, d.h. es wird ein Parametervektor $\varepsilon^{t=0} = (\varepsilon_k{}^{t=0} \in R^+ \mid \varepsilon_k{}^{t=0} = \varepsilon_{k+1}{}^{t=0}; k = 1, ..., n) \in R^{+n}$ betrachtet. In der Feedback-Iteration t=1 werden die einzelnen Komponenten des Umgebungsparametervektors ungleichförmig verändert, sodass die sich ergebende Umgebung $U(q_i \mid \varepsilon^{t=1})$ nur durch einen entsprechenden Parametervektor $\varepsilon^{t=1} = (\varepsilon_k{}^{t=1} \in R^+ \mid \varepsilon_k{}^{t=1} \neq \varepsilon_{k+1}{}^{t=1}; k = 1, ..., n) \in R^{+n}$ beschrieben werden kann (siehe Abb. 99)).

Die Möglichkeiten zur Adaption sind bei einem Hyperellipsoiden größer als bei einer Hyperkugel, da nun n Parameter zur Verfügung stehen, die vergrößert aber auch verkleinert werden können, d.h. in einer Umgebung $U(q_i \mid \varepsilon^t)$ können Teilregionen existieren, die in $U(q_i \mid \varepsilon^{t+1})$ nicht mehr enthalten sind, während der streng monotone Zuwachs Kennzeichen des vorangegangenen Verfahrens auf der Basis der Hyperkugeln war. Erhalten bleiben soll noch der Constraint, dass q_i unveränderlich der Mittelpunkt aller-Hyperellipsoiden bleibt. Diese erhöhte Flexibilität kann jedoch nicht verhindern, dass in einer entsprechenden neuen Retrievalregion $U(q_i \mid \varepsilon^{t=1})$ Dokumentvektoren liegen, die weit von dem Schwerpunkt der bekannten relevanten Dokumentvektoren entfernt liegen (siehe Abb. 99)).

Abb. 99) Hyperellipsoid mit wahrscheinlich relevanten und nicht relevanten Dokumentvektoren



Um die Anzahl der Dokumente zu verkleinern, die dem Agenten zur Relevanzbewertung vorgelegt werden, besteht z.B. die Möglichkeit diejenigen auszuselektieren, die näher an dem Schwerpunkt $\bar{s}_{DVM(i,\overline{rel})}^{t=0}$ der nicht relevanten bekannten Dokumentvektoren liegen, als an dem Schwerpunkt $\bar{s}_{DVM(i,rel)}^{t=0}$ der relevanten Dokumentvektoren. Es besteht weiterhin die Möglichkeit von distanzbasierten Rankingverfahren, indem zunächst die Dokumentvektoren entsprechend steigender Distanz zu $\bar{s}_{DVM(i,rel)}^{t=0}$ sortiert werden. Danach werden die Dokumentvektoren entsprechend sinkender Distanz zu $\bar{s}_{DVM(i,\overline{rel})}^{t=0}$ sortiert, wobei angenommen wird, dass für Dokumentvektoren, die weiter von $\bar{s}_{DVM(i,\overline{rel})}^{t=0}$ entfernt liegen, eine größere Chance existiert, dass sie relevant sind, als für Dokumentvektoren, nahe am Schwerpunkt der nicht relevanten Dokumentvektoren liegen.

Es verbleibt die Formulierung von Kriterien, mit denen der Parametervektor $\varepsilon^{t+1} \in R^{+n}$ aus $\varepsilon^{t} \in R^{+}$ in Abhängigkeit von den Relevanz-Bewertungen der Iteration t bzw. t und t-1 festgelegt werden kann. Die weiteren Darstellungen sollen sich auf den Übergang der Feedback-Iteration t=0 zu t=1 beziehen, sodass ausschließlich Dokumentvektoren aus $DVM(q_i^{t=0})$ mit ihren Relevanz-Bewertungen betrachtet werden können, und keine weiter zurückliegenden Feedback-Iterationen von Bedeutung sind. Als Anhaltspunkt können die Strategien betrachtet werden, die beim Queryvektor- bzw. beim Dokumentvektor-Relevanz-Retrieval formuliert wurden. Beispielsweise kann eine optimale Suchfrage q_{opt} als Punkt in DVR definiert werden, wenn die Differenz aus dem Mittelwert der Ähnlichkeiten zwischen q_{opt} und den relevanten Dokumentvektoren und dem Mittelwert der Ähnlichkeiten zwischen q_{opt} und den irrelevanten Dokumentvektoren maximal ist (Panyr (1987b: 146[252])). Ein denkbares Kriterium bezogen auf eine Retrievalregion könnte fordern, dass alle relevanten Dokumentvektoren in der Region liegen, aber keine irrelevanten bzw. so wenig irrelevante Dokumentvektoren wie möglich. Eine Retrievalregion soll somit unter dem Constraint festgelegt werden, dass in der Umgebung $U(q_i^{t=0} | \varepsilon^{t=1})$ die Elemente aus $DVM(q_i^{t=0})_{rel}$ liegen müssen, jedoch keine oder nur wenige Elemente aus $DVM(q_i^{t=0})_{\overline{rel}}$.

Eine Umgebung lässt sicht jedoch nicht oder nur schwer in Verbindung mit der Forderung formulieren, dass keine Elemente aus $DVM(q_i^{t=0})_{\overline{rel}}$ in der Umgebung liegen dürfen, wenn der Queryvektor stationär bleibt, d.h. wenn eine Region vom Typ $U(q_i \mid \varepsilon^t)$ verwendet werden soll. Ausgeschlossen ist dies, wenn Hyperkugeln verwendet werden, und problematisch, wenn ein Hyperellipsoid verwendet wird, wobei im zweiten Fall eine Suchaufgabe formuliert werden kann, indem ein $\varepsilon^{t=1} \in R^{+n}$ gesucht wird, das eine Umgebung $U(q_i \mid \varepsilon^{t=1})$ bildet, in der alle relevanten Dokumentvektoren, d.h. alle Elemente aus $DVM(q_i)^{t=0}{}_{rel}$ liegen, und das eine nicht-leere Retrieval-Dokumentvektorenmenge $DVM(q_i)^{t=1}$ liefert, d.h. es müssen Dokumentvektoren nachgewiesen werden, zu denen noch keine Relevanzbewertungen vorliegen:

$$\begin{gathered}\text{Suche ein } \varepsilon^{t=1} \in R^{+n}: \\ \forall x_{j|q(i),rel}{}^{t=0} \in DVM(q_i)^{t=0}{}_{rel}: x_{j|q(i,),rel}{}^{t=0} \in U(q_i \mid \varepsilon^{t=1}) \wedge \\ DVM(q_i)^{t=1} \neq \varnothing. \end{gathered} \tag{468}$$

Eine Kombination eines Queryvektor- und eines Retrievalregion-Feedbacks würde eine flexiblere Strategie ermöglichen, bei der die nicht relevanten Dokumentvektoren ganz oder größtenteils ausgeschlossen werden können. Der Mittelpunkt der neuen Region soll als adaptierter Queryvektor $q_i^{t=1}$ aus dem vorangegangenen Queryvektor $q_i^{t=0}$, den beiden Mittelwertsvektoren $\bar{s}_{DVM(i,rel)}{}^{t=0}$ und $\bar{s}_{DVM(i,\overline{rel})}{}^{t=0}$, sowie aus den Parametern $\alpha, \beta, \chi \geq 0$ durch das gemischte Queryvektor-Relevanz-Feedback-Verfahren berechnet werden, sodass die Suche nach einer neuen Retrievalregion die Festlegung des Parametervektors $\varepsilon^{t=1} \in R^{n+}$ erfordert. D.h. es wird nach einem Parametervektor $\varepsilon^{t=1}$ gesucht, bei dem die folgenden Constraints erfüllt sein müssen bzw. der Verstoss gegen diese Constraints soll minimiert werden:

$$\begin{gathered}\text{Suche ein } \varepsilon^{t=1} \in R^{+n}: \\ \forall x_{j|q(i,t=0),rel}{}^{t=0} \in DVM(q_i^{t=0})_{rel}: x_{j|q(i,t=0),rel}{}^{t=0} \in U(q_i^{t=1} \mid \varepsilon^{t=1}) \wedge \\ \forall x_{j|q(i,t=0),rel}{}^{t=0} \in DVM(q_i^{t=0})_{\overline{rel}}: x_{j|q(i,t=0),\overline{rel}}{}^{t=0} \notin U(q_i^{t=1} \mid \varepsilon^{t=1}) \wedge \\ DVM(q_i^{t=1}) \neq \varnothing. \end{gathered} \tag{469}$$

Unabhängig ob eine Region $U(q_i \mid \varepsilon^{t=1})$ oder $U(q_i^{t=1} \mid \varepsilon^{t=1})$ betrachtet wird, es muss festgelegt werden, wie die Suche im n-dimensionalen Raum der Parameter ausgehend von $\varepsilon^{t=0} = (\varepsilon_k{}^{t=0} \mid k = 1, ..., n) \in R^{n+}$ durchgeführt werden soll. Naheliegend ist ein stochastisches Local-Search-Verfahren, bei dem jeder der Parameter $\varepsilon_k{}^{t=0}$ durch eine positive oder negative Zufallszahl verändert wird, wobei der Constraint $\varepsilon_k{}^{t=1} \in R^+$ erfüllt sein muss. D.h. wird für einen Parameter eine große negative Zufallszahl erzeugt, die bei ihrer Addition zu einem negativen Parameterwert führt, so wird entweder allgemein der Betrag aller Operationen verwendet, oder die Operation wird nicht durchgeführt, und es wird eine neue, unabhängig erzeugte, reelle Zufallszahl erzeugt, bis das Ergebnis zum ersten Mal positiv wird.

Es lassen sich alternativ populationsbasierte Ansätze für das Problem formulieren, bei denen aus $\varepsilon^{t=0}$ eine Population von Individuen $a_i{}^{t=0}$ erzeugt wird, die aus einem Vektor $\varepsilon_i{}^{t=0}$ und einem Schrittweitenvektor $\sigma_i{}^{t=0}$ bestehen. Liegt eine Polyrepräsentation von Queryvektoren mit unterschiedlichen Retrievalregionen vor, so können diese als Initialisierungs-Population verwendet werden. Unabhängig wie eine Population von Individuen erzeugt wird, werden im weiteren Verlauf Reproduktions- und Selektionsoperationen durchgeführt, wie z.B. eine (μ,λ)-ES. Wird genau ein Initialisierungs-Individuum $a^{t=0} = (\varepsilon^{t=0},$

$\sigma^{t=0} := \varepsilon^{t=0}$) betrachtet, so werden zunächst durch ungeschlechtliche Reproduktion (Selbstmutation) μ - 1 Individuen erzeugt:

$$a_i^{t=0} = (\varepsilon_i^{t=0}, \sigma_i^{t=0}),\ \sigma_i^{t=0} = mut(\varepsilon^{t=0} \mid \varepsilon^{t=0}),\ \varepsilon_i^{t=0} = mut(\varepsilon_i^{t=0} \mid \sigma_i^{t=0}). \quad (470)$$

Diese Eltern-Population erzeugen durch geschlechtliche Reproduktionsoperationen in einer Generation λ Nachkommen, aus denen die besten μ für die Nachfolgegeneration selektiert werden.

Ausschlaggebend für das verwendete Suchverfahren ist die Definition der Fitnessfunktion in Abhängigkeit von den oben genannten Constraints, für deren Erfüllung bzw. Verletzung unterschiedliche Gewichtungsfaktoren festgelegt werden müssen. Am wichtigsten ist der Einschluss der relevanten Dokumentvektoren sowie der Ausschluss nicht relevanter Dokumentvektoren, sodass $x_{j|q(i,t=0),rel}^{t=0} \in U(q_i^{t=1} \mid \varepsilon^{t=1})$ und $x_{j|q(i,t=0),\overline{rel}}^{t=0} \notin U(q_i^{t=1} \mid \varepsilon^{t=1})$ mit einem hohen positiven Wert gewichtet wird. Das Fehlen eines relevanten Dokumentvektors, d.h. $x_{j|q(i,t=0),rel}^{t=0} \notin U(q_i^{t=1} \mid \varepsilon^{t=1})$, soll mit einem betragsmäßig großen negativen Wert bestraft werden, während der Einschluss eines nicht relevanten Dokumentvektors, d.h. $x_{j|q(i,t=0),\overline{rel}}^{t=0} \in U(q_i^{t=1} \mid \varepsilon^{t=1})$, mit einem betragsmäßig kleineren negativen Gewicht bestraft werden soll, da dies zwar nicht erwünscht ist, jedoch tolerierbarer als der Ausschluss eines relevanten Dokumentvektors ist. Die Gewichtungsfaktoren können im einzelnen von den Parametern $\alpha, \beta, \chi \geq 0$ des gemischten Relevanz-Feedback-Verfahrens abgeleitet bzw. übernommen werden, durch das der aktualisierte Queryvektor $q_i^{t=1}$ erzeugt wird.

Der Constraint, dass die Retrievalmenge unter keinen Umständen leer sein darf, ist ein Faktor, der erst gegen Ende des Suchprozesses Bedeutung gewinnt, wenn Individuen vorhanden sind, welche die Zerlegung der relevanten von den nicht relevanten Dokumentvektoren gut bzw. perfekt erfüllen, da eine leere Menge ein Ausschlusskriterium darstellt, und zu einem Weiterlaufen des Suchprozesses führt, bis das erste Individuum mit einer perfekten Zerlegung auftritt. Zu Beginn des Suchprozesses, wenn ungewünschte Einschlüsse und Ausschlüsse von bewerteten Dokumentvektoren vorliegen, soll keine Kandidatenmenge $DVM(q_i^{t=1})_i$ ermittelt werden, da dies mit aufwendigen Distanzbestimmungen im DVR verbunden ist, d.h. dieser Constraint soll zunächst aus der Fitnessfunktion ausgeschlossen werden. Zum Ende des Suchprozesses, soll er hinzutreten, d.h. es wird eine Form von progressiver Fitnessfunktion verwendet, bei der in einem späteren Stadium eine komplexere Fitnessfunktion durch die Hinzunahme eines weiteren Constraints erzeugt wird.

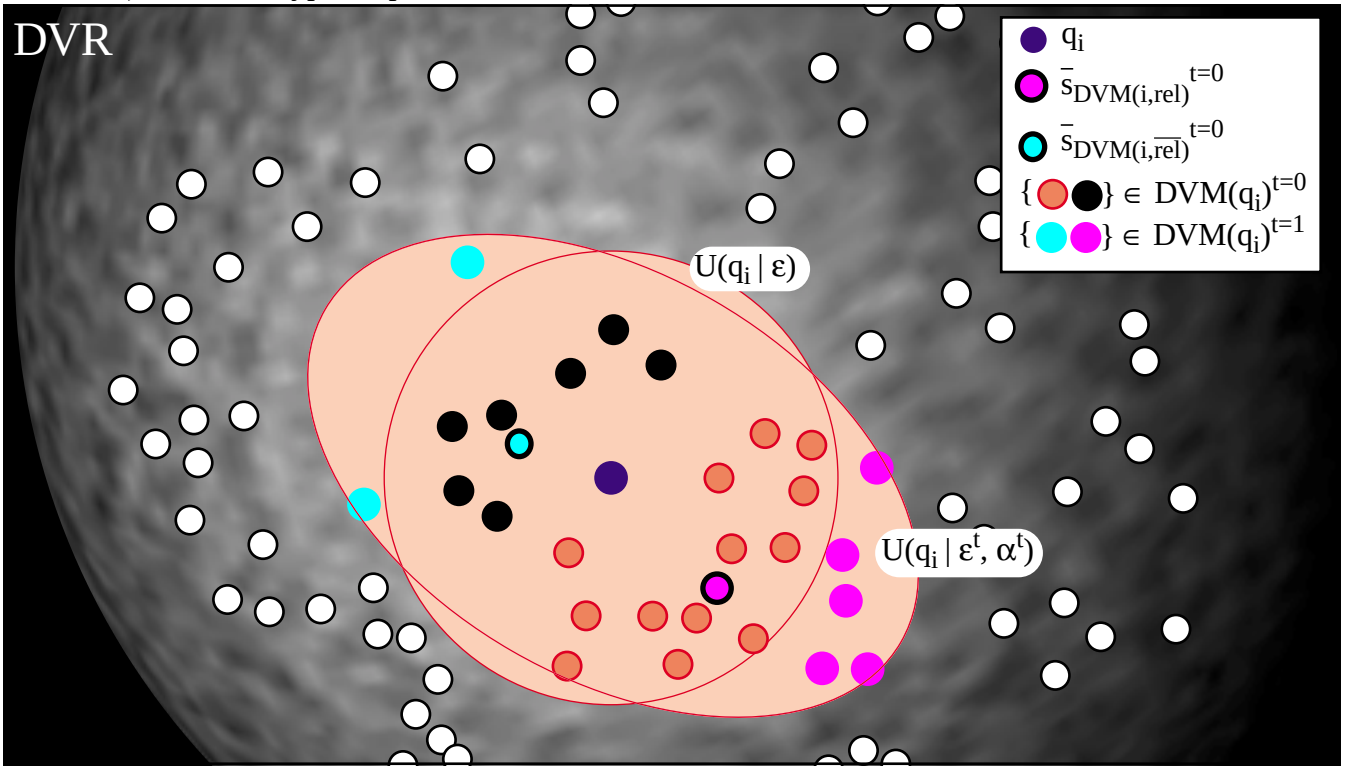
Die Anzahl der Elemente in einer Kandidatenmenge $DVM(q_i^{t=1})_i$ ist dabei selektionsneutral, solange die Menge nicht leer ist, d.h. es ist irrelevant ob zwei oder zehn Elemente enthalten sind. Denkbar wäre aber eine obere Grenze, da die Dokumente, die zu den Dokumentvektoren in der entsprechenden Menge korrespondieren, dem Agenten für eine Relevanzbewertung vorgelegt werden sollen. Dies kann durch die Ersetzung des Constraints „$DVM(q_i^{t=1}) \neq \varnothing$“ durch „$\#DVM(q_i^{t=1}) \in \{1, ..., f_{max}\}$“ erfolgen, oder indem eine Straffunktion eingeführt wird, die für Werte aus der Menge $\{1, ..., f_{max}\}$ ein Gewicht von Null verwendet, und für darüber liegende Werte negative Funktionswerte liefert, mit einem streng monoton steigenden Betrag in Abhängigkeit einer steigenden Anzahl von Dokumentvektoren in der Retrievalmenge.

Sollte sich eine längere Stagnationsphase ergeben, wobei Kandidaten mit einer nicht leeren Retrievalmenge vorliegen, so kann das Verfahren abgebrochen werden, bzw. es soll ein Restart erfolgen.

Eine analoge Vorgehensweise kann mit Retrievalregionen durchgeführt werden, die als rotierte Hyperellipsoide definiert sind, was durch den [n(n-1)/2]-dimensionalen Strategieparametervektor α^t von Rotationswinkel modelliert wird (korrelierte Mutationen, siehe Schwefel (1977[305], 1981[306]), Bäck & Schwefel (1993[23]), Bachelier (1998b: 77ff[16]), Nissen (1994: 155[237])). Es ergibt sich eine Umgebung $U(q_i \mid \varepsilon^t, \alpha^t)$ mit einem stationären bzw. $U(q_i^t \mid \varepsilon^t, \alpha^t)$ bei einem nicht stationären Queryvektor. Die beiden Parametervektoren ε^t und α^t werden in der gleichen Weise betrachtet wie oben der Strategievektor ε^t, indem ausgehend von q_i bzw. q_i^t eine Region gesucht wird, die eine Menge von Constraints bezüglich der relevanten, irrelevanten und neuen Dokumentvektoren erfüllen. Die Suche kann durch einen ES-Prozess durchgeführt werden, wobei nun Individuen mit n + n(n-1)/2 reellen Parametern betrachtet werden müssen, was den Aufwand der Reproduktion entsprechend aufwendiger werden lässt.

Wird ein stationärer Queryvektor verwendet, so ergeben sich bei einem rotierten Hyperellipsoid im Prinzip wieder die gleichen Einschränkungen wie bei einem nicht rotierten, d.h. die nicht relevanten Dokumentvektoren können nicht oder nur teilsweise ausgeschlossen werden, und es können in der neuen Retrieval-Dokumentvektorenmenge Elemente liegen, die näher zu dem Schwerpunkt $\bar{s}_{DVM(i,rel)}^{t=0}$ der bekannten nicht relevanten Dokumentvektoren liegen, als zu dem Schwerpunkt $\bar{s}_{DVM(i,rel)}^{t=0}$ der relevanten Dokumentvektoren (siehe Abb. 100)).

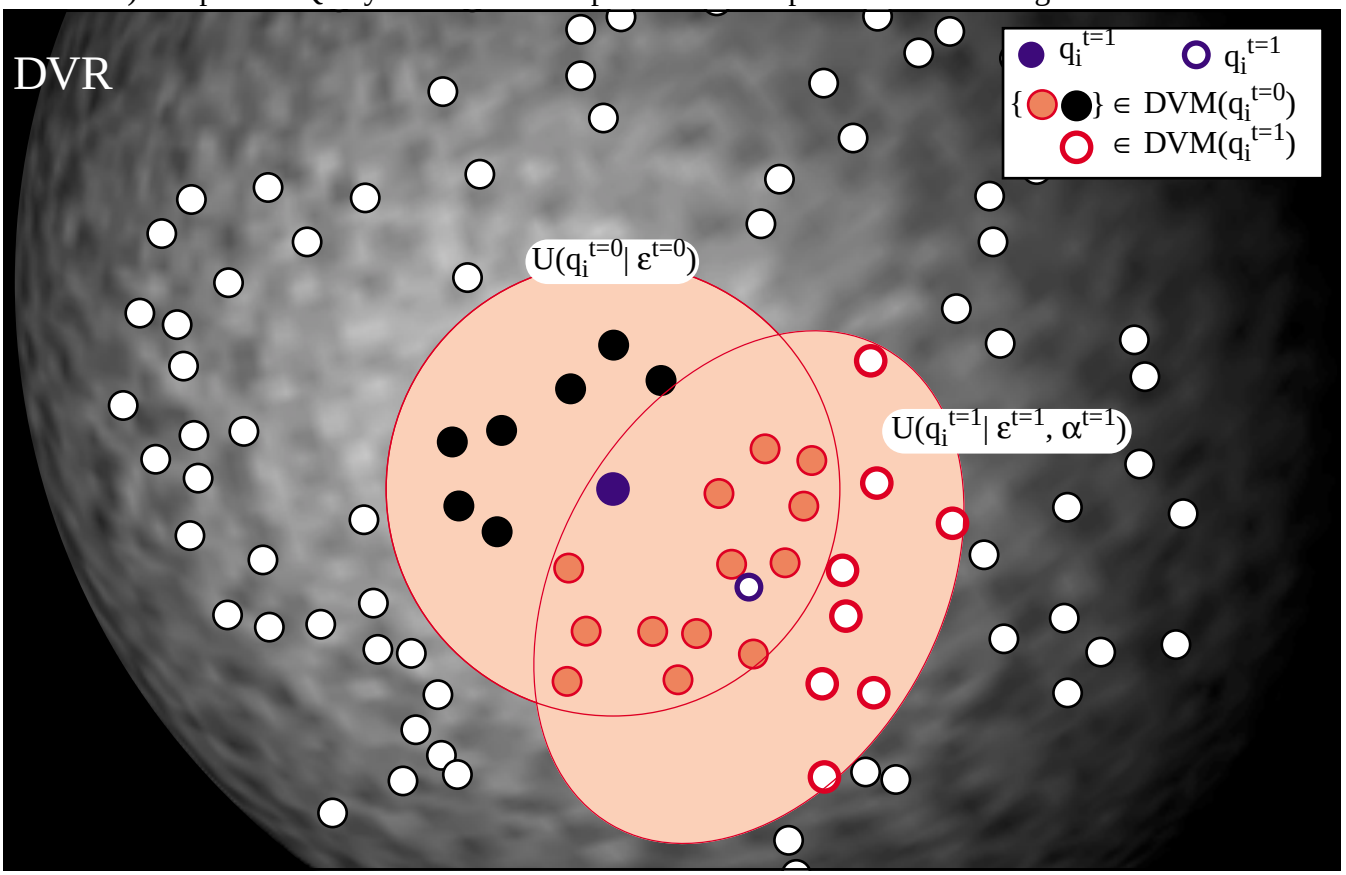
Abb. 100) Rotierter Hyperellipsoid mit wahrscheinlich relevanten und nicht relevanten Elementen



Mit Hilfe der Identifikation und dem Ausschluss dieser Dokumentvektoren, bzw. einer geeigneten Form von Ranking, die zwischen den wahrscheinlich relevanten und nicht relevanten in der neuen Retrievalmenge unterscheidet, kann der Aufwand der Bewertung durch den Agenten verringert werden.

Die Verwendung eines rotierten Hyperellipsoiden in Verbindung mit einem adaptierten Queryvektor, d.h. die Verwendung einer Region $U(q_i^{t=1} | \varepsilon^{t=1}, \alpha^{t=1})$, liefert für die constraint-kompatible Suche die größte Flexibilität mit entsprechend großem Aufwand.

Abb. 101) Adaptierter Queryvektor als Mittelpunkt einer adaptierten Retrievalregion



Im Laufe des Relevanz-Feedback-Verfahrens dürfte es zunehmend schwieriger werden, mit genau einer Retrievalregion eine Zerlegung aller relevanten und nicht relevanten Dokumentvektoren nachzuvollziehen, die in den Iterationen 0 bis t-1 ermittelt wurden. Ein Ansatz dieser Problematik ist die Aufgabe des Constraints, dass alle relevanten Dokumentvektoren, die in den Iterationen 1 bis t-1 ermittelt wurden, in der Retrievalregion der Iteration t liegen sollen, sondern es werden nur die relevanten Dokumentvektoren aus der jeweils vorangegangenen Iteration betrachtet. In den bisherigen Darstellungen wurde dieser Fall betrachtet, da die Iterationen t=0 und t=1 betrachtet wurden.

Ein anderer Ansatz besteht darin, mehrere Retrievalregionen einzusetzen, um beliebige Verteilungen von relevanten und nicht relevanten Dokumentvektoren zu modellieren. Diese Überlegung führt zu dem bereits dargestellten Querysplitting bzw. der Queryvektor-Polyrepräsentation im Rahmen des Relevanz-Feedbacks. Beide Ansätze nehmen bereits eine Form von Retrievalregion-Feedback vorweg, da um jeden der erzeugten Queryvektoren eine eigene Retrievalregion notwendig ist, um neue Dokumentvektoren zu ermitteln, deren zugehörige Dokumente dem Agenten präsentiert werden. Die Ansätze wurden jedoch nur mit einer konstanten Umgebung der Form $U(q_{ik}^{t} | \varepsilon \in R^{+})$ beschrieben, d.h. für alle parallel existierenden Queryvektoren q_{ik}^{t} unabhängig von der Iteration t, wird eine Hyperkugel mit dem gleichen Radius ε als Retrievalregion verwendet.

3.9.6) Indexierungsfunktion-Relevanz-Feedback

Eine Veränderung der Queryvektoren oder der Dokumentvektoren durch eine Relevanz-Feedback-Operation ist indirekt eine Form der Veränderung der Indexierungsfunktion $A_{IR(Q)}$ bzw. $A_{IR(D)}$, da eine entsprechende Query auf einen anderen Queryvektor abgebildet wird, bzw. Dokumente auf andere Dokumentvektoren abgebildet werden. Durch diese Interpretation der Feedback-Operation kann versucht werden, eine neue Indexierungsfunktion abzuleiten, die den geäusserten Relevanzurteilen des Agenten bezüglich des betrachteten Ziels angemessen ist.

Eine solche Vorgehensweise ist interessant, da eine individualisierte Indexierungsfunktion als eine explizite und kompakte Beschreibung des mentalen Modells des Agenten interpretiert werden kann, die sukzessive verfeinert und in anderen Kontexten wieder verwendet werden kann. Insbesondere eine explizite, individualisierte Dokument-Indexierungsfunktion besitzt gegenüber der Verwendung individualisierter Dokumentvektorenverteilungen den Vorteil, dass sie wesentlich speichereffizienter sein kann. Wie sich zeigen wird, ist der Aufwand zur Erzeugung individualisierter Indexierungsfunktionen jedoch sehr hoch, sodass diese Vorgehensweise vor allem als Off-Line-Verfahren einsetzbar ist, bei dem die Retrieval- und Relevanzbewertungsergebnisse eines Agenten ausgewertet werden, um ein Modell des Agenten bezüglich eines speziellen Informationsbedürfnisses zu generieren, das durch den Agenten in Verbindung mit einem IRS wiederverwendet werden kann, wenn er zukünftig ein ähnliches Informationsbedürfnis besitzt. Ein solches Modell kann auch im Rahmen eines Profils in einem Information-Filter-Systems eingesetzt werden, wenn das Informationsbedürfnis langfristig bestehen sollte.

Prinzipiell können zwei Strategieklassen zur Erzeugung individueller Indexierungsfunktionen unterschieden werden, wenn keine Performance-Constraints existieren:

1) Direkte Verwendung von Relevanzurteilen, die während den Feedback-Iterationen anfallen, um in jeder Iteration die Indexierungsfunktion $A_{IR(.)}^{t}$ zu modifizieren.
2) Durchführung eines Query- oder Dokumentvektoren-Feedbacks, und Verwendung der Zwischen- bzw. Endergebnissen, d.h. ein Zwischenergebnis q_i^t oder das Endergebnis $q_i^{t(end)}$ bei einem Queryvektor-Feedback bzw. eine Dokumentvektorenverteilung bei einem Dokumentvektor-Feedback wird verwendet, um daraus eine individualisierte Indexierungsfunktion zu erzeugen.

Im weiteren soll ausschließlich die zweite Möglichkeit betrachtet werden, da für den ersten Fall direkte Kriterien fehlen, durch die eine Adaption der Indexierungsfunktion durch Relevanzurteile durchgeführt werden kann. Dies ergibt sich einfach dadurch, da eine Indexierungsfunktion zwar als Punkt in einem Funktionsraum betrachtet werden kann, dass jedoch im DVR keine direkt abgeleiteten Fixpunkte existieren, die eine Adaption analog z.B. der gemischten Strategie beim Queryvektor-Feedback ermöglichen würde. Die Zwischen- und Endergebnisse im zweiten Fall sind jedoch diese Fixpunkte, mit deren Hilfe indirekt individuelle Indexierungsfunktionen bestimmbar werden. Die damit verbundene Fragestellung ist eine Form der Umkehrung der normalen Indexierung, bei der eine Query Q_i und eine Indexierungsfunktion $A_{IR(Q)}$ gegeben ist, und ein Queryvektor $q_i^{t=0}$ gesucht wird. Bei der vorliegenden Aufgabe ist eine Query Q_i und der Queryvektor $q_i^{t(end)}$ gegeben, und eine Indexierungsfunktion wird gesucht, die Q_i auf ein Zwischenergebnis q_i^t bzw. ein das Endergebnis $q_i^{t(end)}$ abbildet.

Verfahren, die Zwischenergebnisse verwenden, sind für eine On-Line-Suche nach individualisierten Indexierungsfunktionen geeignet, d.h. die Suche und die Relevanz-Feedback-Operationen können im gleichen Zeitintervall ablaufen, das durch die Interaktion von Agent und IRS bestimmt wird. Bei der Verwendung des oder der Endergebnisse kann nur eine Off-Line-Suche verwendet werden, da alle Relevanz-Feedback-Operationen durchgeführt wurden, um die Endergebnisse zu erlangen.

Im weiteren soll allgemein von einem populationsbasierten Suchverfahrens nach individuellen Indexierungsfunktionen ausgegangen werden, d.h. es wird eine Population $P(A_{IR})^t$ von Elternindividuen $A_{IR,j}{}^t$ bzw. eine Zwischenpopulation $ZP(A_{IR})^t$ von Nachkommen $A_{IR,N(j)}{}^t$ betrachtet. Die Population, die zum Zeitpunkt des Verfahrensabbruchs existiert bzw. die virtuelle Elite-Population , d.h. die Menge der besten Individuen, die während des Verfahrens erzeugt wurden, bildet eine natürliche Polyrepräsentation, des speziellen Informationsbedürfnisses des Agenten, die für weitere Anwendungen wie z.B. im Rahmen eines Polyrepräsentations-IFS verwendet werden können.

3.9.6.1) Detaillierte Beschreibung der Indexierungsfunktion

Es soll vereinfachend eine gemeinsame Indexierungsfunktion für die Queries und die Dokumente unterstellt werden, d.h. $A_{IR} := A_{IR(Q)} = A_{IR(D)}$. Allgemein wurde im Abschnitt 3.4) eine Indexierungsfunktion A_{IR} als Funktion definiert, die eine Zeichensequenz in einen metrischen Vektorraum abbildet:

$$A_{IR(D)}: D(\Theta) \rightarrow DVR: D_i \mid\rightarrow x_i. \text{ bzw.}$$
$$A_{IR(Q)}: D(\Theta) \rightarrow DVR: Q_i \mid\rightarrow q_i. \quad (471)$$

Diese Definition soll im weiteren detaillierter betrachtet werden, indem die Merkmalsgewichtungsmodelle (siehe Abschnitt 3.5)) hinzu gezogen werden. Hierzu soll die Vielzahl von Variablen, die im Rahmen der Indexierungsfunktion betrachtet werden, in zwei Klassen unterteilt werden:
1) Lokale Variablen
Diese können ausschließlich aus der Kenntnis der Zeichensequenz D_i bzw. Q_i hergeleitet werden. Hierzu gehören die absoluten Häufigkeiten der Merkmale F_j in D_i bzw. Q_i, die in dem Vektor $h(D_i)$ bzw. $h(Q_i)$ zusammengefasst werden:

$$h(D_i) = (h(F_j \mid D_i) \mid j = 1, ..., n),$$
$$h(Q_i) = (h(F_j \mid Q_i) \mid j = 1, ..., n). \quad (472)$$

Eine spezielle lokale Variable ist die Komponente mit dem größten Wert, als das Merkmal, das am häufigsten auftritt, und das als $F_{max|D(i)}$ bzw. $F_{max|Q(i)}$ mit den Häufigkeiten $h(F_{max|D(i)} \mid D_i)$ bzw. $h(F_{max|Q(i)} \mid Q_i)$ bezeichnet wird.

2) Globale Variablen
Diese Variablen können nur in Verbindung mit der Dokumentenliste D^t bzw. den Häufigkeitsvektoren $h(D_i)$, i = 1, ..., m, bestimmt werden. Hierzu gehören die absolute Häufigkeit des Auftretens der Merkmale F_j in D^t, die in dem Vektor $h(F \mid D^t)$ zusammengefasst werden:

$$h(F \mid D^t) =(h(F_j \mid D^t) \mid j = 1, ..., n). \quad (473)$$

Aus den Häufigkeiten $h(F_j \mid D^t)$ werden Gewichtungen $w(F_j \mid D^t)$ als sekundäre globale Variablen erzeugt, indem die Anzahl der Dokumente oder die Häufigkeit des am häufigsten auftretenden Merkmals in D^t hinzu gezogen wird.

Das Rauschen $\nu(F_j \mid D^t)$ und das Signal $sig(F_j \mid D^t)$ sind zwei besondere, sekundäre, globale Variablen, wobei für das Rauschen alle lokalen Variablen $h(F_j \mid D_i)$, i = 1, ..., m, benötigt werden. Ein Maß wie die Entropie $w(F_j \mid D^t)_{Entropie}$ ist eine tertiäre globale Variable, da sie aus dem Rauschen $\nu(F_j \mid D^t)$ berechnet wird. Ein anderes Beispiel für eine abgeleitete globale Variable ist der Variationsquotient V_j^t des Merkmals F_j, der sich als Quotient der Varianz var_j^t und der mittleren Häufigkeit $\overline{h}(F_j \mid D^t)$ des Auftretens von F_j in D^t berechnet.

Im allgemeinen Fall nimmt eine Indexierungsfunktion $A_{IR(Q)}$ den lokalen, absoluten Häufigkeitsvektor $h(Q_i) = (h(F_j \mid Q_i) \mid j = 1, ..., n)$ und den globalen, absoluten Häufigkeitsvektor $h(F \mid D^t) =(h(F_j \mid D^t) \mid j = 1, ..., n)$ als Input, und liefert einen Queryvektor q_i als Output. Alle Komponenten der absoluten Häufigkeitsvektoren stammen aus N_0, sodass eine detaillierte Beschreibung der Indexierungsfunktion $A_{IR(Q)}$ bestimmt werden kann zu:

$$A_{IR(Q)}: {N_0}^n \times {N_0}^n \rightarrow DVR: (h(Q_i), h(F \mid D^t)) \mid\rightarrow q_i. \tag{474}$$

D.h. die n reellen Komponenten des Vektors q_i werden als Funktion von 2 * n natürlichen Zahlen $h(F_j \mid Q_i)$, $h(F_j \mid D^t)$ berechnet:

$$q_i = (q_{ij} \mid j = 1, ..., n) = A_{IR(Q)}(h(F_j \mid Q_i), h(F_j \mid D^t) \mid j = 1, ..., n). \tag{475}$$

Werden abgeleitete Variablen wie das Rauschen, Signal, Mittelwert und Varianz mit berücksichtigt, so müssen die lokalen Häufigkeiten $h(F_j \mid D_k)$ für alle Merkmale und alle Dokumente berücksichtigt werden. Man gelangt somit zu einer allgemeinen Definition:

$$A_{IR(Q)}: {N_0}^n \times {N_0}^{n*m} \rightarrow DVR: (h(Q_i), (h(F_j \mid D_k) \mid j = 1, ..., n; k = 1, ..., m)) \mid\rightarrow q_i. \tag{476}$$

D.h. die n reellen Komponenten des Vektors q_i werden als Funktion von (n + n*m) natürlichen Zahlen berechnet:

$$q_i = (q_{ij} \mid j = 1, ..., n) = A_{IR(Q)}(h(F_j \mid Q_i), h(F_j \mid D_k) \mid j = 1, ..., n; k = 1, ..., m). \tag{477}$$

Eine solche Formulierung muss jedoch nicht bedeuten, dass für jede Komponente q_{ij} alle (n + n*m) Zahlen benötigt werden. Entsprechend den abgeleiteten Variablen, die bei den Merkmalsgewichtungen dargestellt wurden, wird q_{ij} aus den (1 + m) lokalen und globalen Variablen berechnet, die zu dem Merkmal F_j gehören:

$$q_{ij} = A_{IR(Q)}(h(F_j \mid Q_i), h(F_j \mid D_k) \mid k = 1, ..., m). \tag{478}$$

Diese Formulierung kann weiter spezifiziert werden, indem q_{ij} nur als Funktion von $h(F_j \mid Q_i)$ und $h(F_j \mid D)$ berechnet wird:

$$q_{ij} = A_{IR(q(i))}(h(F_j \mid Q_i), h(F_j \mid D)). \tag{479}$$

Wird die Darstellung aus Zobel & Moffat (1998[376]) zugrunde gelegt, so berechnet sich eine Queryvektorenkomponente q_{ij} aus der relativen Häufigkeit $r(F_j \mid Q_i)$ aus der Query und dem Gewichtungsfaktor $w(F_j \mid D)$ aus der gesamten Dokumentliste D. Wird z.B. eine logarithmische, relative Häufigkeit und eine normalisierte hyperbolische Gewichtung verwendet, so ergibt sich:

$$\begin{aligned} q_{ij} &= r(F_j \mid Q_i) * w(F_j \mid D), \text{ mit} \\ r(F_j \mid Q_i) &= 1 + \log_e(h(F_j \mid Q_i)), \text{ und} \\ w(F_j \mid D) &= \log_e[(m - h(F_j \mid D))/h(F_j \mid D)]. \end{aligned} \tag{480}$$

Wird eine Indexierungsfunktion als Programm verstanden, auf dem z.B. GP-Operationen durchgeführt werden (Koza (1992[188], 1994[189]), Koza et al. (1999[191]), Banzhaf et al. (1998[26]), Nordin (1997[238])) , so müssen neben den Variablen auch Operationen definiert werden, welche die Variablen verknüpfen. Aus den dargestellten Modellen zur Merkmalsgewichtung kann die Menge der Operationen auf die arithmetischen Basisoperationen beschränkt werden.

3.9.6.2) Indexierungsfunktionssuche nach dem Queryvektor-Feedback

3.9.6.2.1) Fitnessfunktion

Die Festlegung einer Indexierungsfunktion $A_{IR(q(i))}^{t(end)}$ nach der Durchführung einer Sequenz von Queryvektor-Feedback-Operationen lässt sich als symbolische Regression formulieren, wobei eine solche Formulierung unterspezifiziert ist, da unendlich viele Funktionen existieren, die dies leisten. Diese Unterspezifizierung lässt sich durch die Hinzunahme von Dokumenten mit ihren Häufigkeitsvektoren verringern, jedoch nie beseitigen. Bei der zusätzlichen Verwendung von Dokumenten wird nicht nur gefordert, dass eine Query Q_i direkt auf den Queryvektor $q_i^{t(end)}$ abgebildet wird, sondern dass Dokumente auf bestimmte Dokumentvektoren abgebildet werden, wenn von der Identität der Query- und der Dokument-Indexierungsfunktion ausgegangen wird. Die Query und Dokumente bilden somit Fitness-Cases z.B. im Rahmen eines Genetic-Programming-Systems, die jeweils aus einer Zeichensequenz Q_i bzw. D_k bestehen, aus dem Vektor der absoluten Häufigkeiten $h(Q_i)$ bzw. $h(D_k)$, und dem Zielvektor $q_i^{t(end)}$ bzw. x_k. Die Matrix der absoluten Häufigkeiten der Merkmale in den Dokumenten aus D ist potentiell bei allen Fitness-Cases beteiligt, und kann daher aus den einzelnen Elementen herausgezogen werden.

Unter der Annahme, dass ein Queryvektor-Relevanz-Feedback ohne Dokumentvektor-Feedback-Operationen durchgeführt wurde, können folgende Strategien zur Spezifizierung einer Menge von Fitness-Cases betrachtet werden:

1) Verwende beliebige Dokumente aus D und die korrespondierenden Dokumentvektoren. Werden alle Dokumente verwendet, so ergibt sich eine Menge FCM von Fitness-Cases:

$$FCM = \{(Q_i, h(Q_i), q_i^{t(end)}), (D_k, h(D_k), x_k) \mid k = 1, ..., m\}. \tag{481}$$

2) Verwende nur solche Dokumente und korrespondierende Dokumentvektoren, die während der Queryvektor-Feedback-Iterationen durch den Agenten als relevant bewertet wurden. Es wird die Vereinigung der Mengen $DVM(q_i^t)_{rel}$ der relevanten Dokumentvektoren der einzelnen Iterationen verwendet:

$$FCM = \{(Q_i, h(Q_i), q_i^{t(end)}), (D_k, h(D_k), x_k) \mid \forall x_k \in DVM(q_i)_{ges,rel}, \text{ mit}$$
$$DVM(q_i)_{ges,rel} = \cup_{t=0->t(end)} DVM(q_i^t)_{rel}. \quad (482)$$

Wird zusätzlich ein Dokumentvektoren-Relevanz-Feedback durchgeführt, so können die Dokumentvektoren aus der Ergebnisliste nach Abschluss der Verschiebe-Operationen in der Iteration t(end) als Constraints bzw. Fitness-Cases hinzukommen. In diesem Fall besitzen die Dokumentvektoren ebenfalls eine Abhängigkeit von der Feedback-Iteration t, sodass nach dem Abbruch des Relevanz-Feedback-Verfahren Dokumentvektoren der Art $x_k^{t(end)}$ vorliegen. Werden alle Dokumente als Fitness-Cases verwendet, so ergibt sich eine Menge von Fitness-Cases, mit der $A_{IR(q(i))}^{t(end)}$ bestimmt werden soll:

$$FCM = \{(Q_i, h(Q_i), q_i^{t(end)}), (D_k, h(D_k), x_k^{t(end)}) \mid k = 1, ..., m\}. \quad (483)$$

Im weiteren dieses Abschnittes soll der Index t als Iteration der populationsbasierten Suche nach individuellen Indexierungsfunktionen verwendet werden, während oben t die Iterationen des Queryvektor-Feedback-Verfahrens beschreibt.

Allgemein wird für einen Indexierungsfunktions-Kandidaten $A_{IR,j}^t$ in der t'ten Iteration der Suche die Indexierung der Fitness-Cases durchgeführt, d.h. ein Häufigkeitsvektor $h(Q_i)$ wird als Input in $A_{IR,j}^t$ verwendet, und die Funktion liefert den Outputvektor $A_{IR,j}^t(Q_i) \in DVR$. Bzw. es wird ein Vektor $h(D_k)$ als Input verwendet, und als Output ergibt sich der Vektor $A_{IR,j}^t(D_k) \in DVR$, die in die Menge der Fitness-Cases integriert werden, wobei der Fall eines zusätzlichen Dokumentvektoren-Relevanz-Feedbacks dargestellt werden soll:

$$FCM_{IR,j}^t = \{(Q_i, h(Q_i), q_i^{t(end)}, A_{IR,j}^t(Q_i)), (D_k, h(D_k), x_k^{t(end)}, A_{IR,j}^t(D_k)) \mid$$
$$k = 1, ..., m\}. \quad (484)$$

Um die Qualität, d.h. die Fitness, eines Kandidaten zu ermitteln, werden zunächst die Distanzen zwischen den Soll-Vektoren $q_i^{t(end)}$ bzw. $x_k^{t(end)}$ und den Ist-Vektoren $A_{IR,j}^t(Q_i)$ bzw. $A_{IR,j}^t(D_k)$ gebildet, die in die Struktur der Fitness-Cases eingetragen werden:

$$FCM_{IR,j}^t = \{(Q_i, h(Q_i), q_i^{t(end)}, A_{IR,j}^t(Q_i), d_{DVR}(q_i^{t(end)}, A_{IR,j}^t(Q_i))),$$
$$(D_k, h(D_k), x_k^{t(end)}, A_{IR,j}^t(D_k), d_{DVR}(x_k^{t(end)}, A_{IR,j}^t(D_k))) \mid k = 1, ..., m\}. \quad (485)$$

Die Fitnessfunktion f(.), die einen Kandidaten $A_{IR,j}^t$ bezüglich der Menge der verwendeten Fitness-Cases bewertet, wird aus den Distanzen der einzelnen Fitness-Cases gebildet. Z.B. kann die durchschnittliche Distanz als Fitnessmaß bestimmt werden:

$$f(A_{IR,j}^t) = 1/(m+1) * [d_{DVR}(q_i^{t(end)}, A_{IR,j}^t(Q_i)) + \sum_k d_{DVR}(x_k^{t(end)}, A_{IR,j}^t(D_k))]. \quad (486)$$

Dieser Wert soll minimiert werden, d.h. in einer Zwischenpopulation werden die Nachkommen zur Übernahme in die nächste Generation selektiert, welche die kleinsten f(.)-Werte besitzen. Eine alternativ definierte Fitnessfunktion verwendet unterschiedliche Gewichtungsfaktoren für die Indexierung einer Query und von Dokumenten, wobei die Query als wichtiger betrachtet wird, was durch einen Gewichtungsfaktor größer Eins vor der Queryvektordistanz operationalisiert wird.

Ein Spezialfall ist die Gleichgewichtung der Queryindexierung mit allen Dokumentindexierungen, indem bei m Dokumenten der Gewichtungsfaktor m bei der Queryvektordistanz verwendet wird:

$$f(A_{IR,j}^{t}) = 1/2m * [m * d_{DVR}(q_i^{t(end)}, A_{IR,j}^{t}(Q_i)) + \sum_k d_{DVR}(x_k^{t(end)}, A_{IR,j}^{t}(D_k))]. \quad (487)$$

Komplexere Fitnessfunktionen beziehen nicht nur den Mittelwert als Lageparameter, sondern auch Streunungsparameter wie die Varianz der Distanzen mit ein, wobei kleinere Varianzen höher bewertet werden als größere Varianzen. Im allgemeinen Fall führt eine solche Erweitung zu einer Mehr-Ziel-Optimierung mit Hilfe der verwendeten Lage- und Streuungsparameter. Anstatt der Verwendung von Parametern, die durch Aggregation der Distanzwerte erzeugt werden, kann eine Distanz-Verteilung pro Indexierungsfunktions-Kandidat gebildet werden. Verfahren, wie die Dominanz zweier diskreter oder stetiger Verteilungen bestimmbar ist, können aus den Darstellungen zur Bewertung des Populationsfortschrittes direkt übernommen werden, ohne dass hier näher darauf eingegangen werden soll (siehe hierzu Bachelier (1999b[20])).

Die Durchführung einer EA- oder spezieller einer GP-Suche nach einer geeigneten Indexierungsfunktion beginnt mit der Erzeugung einer Initialisierungs-Population von Funktionen, wobei von Bedeutung ist, ob eine Indexierungsfunktions-Mono- oder -Polyrepräsentation vorliegt. Bei einer Polyrepräsentation existieren Indexierungsfunktionen $A_{IR,k}$, k = 1, ..., δ, die direkt als Initialisierungs-Population verwendet werden können, im Gegensatz zu einer Mono-Repräsentation, bei der eine Initialisierungs-Population z.B. durch Mutations-Operationen auf A_{IR} erst erzeugt werden muss. Reicht die Anzahl δ nicht aus, so kann in der ersten Generation z.B. ein (δ, λ)-EA, und in den weiteren Generationen eine (μ, λ)-EA durchgeführt werden.

Der Abbruch des Suchverfahrens kann durch eine Stagnations- bzw. ein Qualitätskriterium definiert werden. Bei einem Stagnationskriterium erfolgt der Abbruch, wenn eine bestimmte Anzahl von Generationen keine Verbesserung ergeben hat, unabhängig ob eine Ein- oder Mehr-Ziel-Optimierung oder Verteilung verwendet wird. Bei einem Abbruch durch genau ein Qualitätskriterium, wie dem Distanzmittelwert, erfolgt der Abbruch, wenn zum ersten Mal ein Kandidat auftritt, dessen Fitnesswert einen Schwellenwert unterschreitet. Werden mehrere Parameter im Rahmen einer Mehr-Ziel-Minimierung verwendet, so kann ein Abbruch erfolgen, wenn zum ersten Mal jeder der Parameter unter einen parameterspezifischen Schwellenwert fällt.

3.9.6.2.2) Strategien zur Effizienzverbesserung

Das prinzipielle Problem der bislang dargestellten Vorgehensweise besteht in dem sehr großen Aufwand der Suche, was zum einen durch die geringe strukturelle Vorgabe verursacht ist, wenn ein GP-Verfahren verwendet wird. Zum anderen liegt dies in der Verwendung aller m Dokumentvektoren für die Evaluierung eines Kandidaten.

Werden wenige Vorgaben bezüglich der Struktur der Indexierungsfunktionen gemacht, so muss das Verfahren einen großen Raum möglicher Funktionen bzw. Programme durchsuchen. Wird eine parametrisierte Funktionsstruktur vorgegeben, so findet die Suche in einem Parameterraum anstatt in einem

kombinierten Struktur- und Parameterraum statt. Eine Parametersuche kann durch eine ES effizienter durchgeführt werden, als die Suche nach einer Programmstruktur mit der Festlegung von Programmparametern durch ein GP-System.

Wird der zweite Faktor betrachtet, so kann eine wesentliche Effizienzverbesserung durch eine Einschränkung der Anzahl der einbezogenen Fitness-Cases erreicht werden, indem beispielsweise nur eine Stichprobe aus den m Dokumentvektoren verwendet wird, die für alle Kandidaten über alle Generationen konstant ist. Interessanter sind jedoch progressive Evaluationsumgebungen, bei denen zunächst eine kleine Initialisierungs-Stichprobe verwendet wird, die mit jeder Generation erhöht werden kann, d.h. der Umfang der Fitness-Cases wird als eine monoton steigende Funktion modelliert, bis eine maximale Sollanzahl erreicht wird, die gleich der Gesamtzahl m der Fitness-Cases sein kann. Erst wenn die Qualität der Kandidaten besser wird und die Unterschiede der Fitnesswerte immer kleiner werden, ist die Erhöhung der Anzahl der Fitness-Cases notwendig, um eine ausreichende Differenzierung der Kandidaten zu gewährleisten, sodass die Selektion im Rahmen einer EA durchgeführt werden kann. Da es vorkommen kann, dass bei einer Teilmenge von Fitness-Cases das Abbruchkriterium auf der Basis eines Qualitätsmaßes frühzeitig erreicht wird, soll der Verfahrensabbruch bei einer solchen progressiven Evaluationsumgebung ausgesetzt werden, bis in einer späteren Generation die Sollanzahl der Fitness-Cases geprüft werden.

Neben der Stichprobenziehung aus der Gesamtmenge der Dokumente ist die Nutzung einer SC-GNG-Clusterung der Dokumente eine geeignete Strategie, um die Anzahl der Fitness-Cases zu reduzieren, indem der Median-Dokumentvektor pro Cluster ausgewählt wird, d.h. der Dokumentvektor, der dem Gewichtsvektor (oder dem Mittelwertsvektor) am nächsten liegt. Dabei muss beachtet werden, dass bei einem Dokumentvektoren-Feedback die Graphenstruktur einer SC-GNG-SOM an die veränderte Dokumentvektorenverteilung angepasst werden muss. Bei einem reinen Queryvektoren-Feedback bleibt demgegenüber die Dokumentvektorenverteilung und somit die Gewichtsvektorenverteilung erhalten.

3.9.6.3) Indexierungsfunktionssuche parallel zum Queryvektor-Feedback

Wird der sehr aufwendige Suchprozess parallel zum Queryvektor-Relevanz-Feedback durchgeführt, so kann zunächst vermutet werden, dass dies nur gelingt, wenn die Anforderungen an die Qualität der Lösung zu Beginn nicht zu streng sind, bzw. parallel zu dem Fortschreiten des Feedback-Verfahrens im Sinne einer progressiven Evaluationsumgebung gesteigert wird. Diese Einschränkungen müssen jedoch nicht notwendig gelten, wenn die unterschiedlichen Phasen des Relevanz-Feedbacks betrachtet werden, da dieser aus einer Phase der externen Bewertung durch den Agenten und einer Phase interner Vektormodifikationen besteht. Für das IRS fallen nur in der zweiten Phase Berechnungen an, während es in der ersten Phase sich anderem widmen könnte, wie der Suche nach individuellen Indexierungsfunktionen auf der Basis der zu diesem Zeitpunkt aktuellen Menge der Fitness-Cases. Diese Betrachtung kann zu einem reaktiven Verfahren führen, bei dem die Anzahl der Iterationen bei der A_{IR}-Suche nicht durch einen Maximalwert festgelegt wird, sondern durch die Zeit, die der Agent für seine Relevanzbeurteilung benötigt. Benötigt er länger, so können mehr Iterationen durchgeführt werden, ansonsten wird die A_{IR}-Suche früher unterbrochen, und nach den Feedback-Operationen auf der Basis der aktualisierten Menge der Fitness-Cases wieder aufgenommen.

Bei einer A_{IR}-Suche vor der Aktualisierung der Menge der Fitness-Cases können die Relevanzurteile des Agenten natürlich nicht einbezogen werden, da sie noch nicht vorliegen, sodass die A_{IR}-Suche in der Feedback-Iteration t auf der Basis von FCM^{t-1} durchgeführt werden muss.

Die nachfolgenden Darstellungen sollen sich wiederum auf ein populationsbasiertes Suchverfahren beziehen, bei dem eine Population $P(A_{IR})$ von μ Kandidaten über mehrere Generationen entwickelt wird, wobei neben t ein weiterer Iterationsindex τ notwendig wird. Die Bezeichnung $A_{IR,j}^{t,\tau}$ stellt somit eine Indexierungsfunktion innerhalb der τ'ten EA-Generation in der t'ten Relevanz-Feedback-Iteration dar. Es sollen zwei prinzipielle Fälle unterschieden werden:
1) Die A_{IR}-Suche hat keine Auswirkung auf die Queryvektor-Modifikation.
2) Zwischenergebnisse der A_{IR}-Suche werden verwendet, um den Queryvektor zu modifizieren.

Der erste Fall beschreibt die oben dargestellte Situation, dass in einer Feedback-Iteration t zuerst eine A_{IR}-Suche auf der Basis von FCM^{t-1} durchgeführt wird, während der Agent extern die Dokumente bewertet, die in der Iteration t-1 durch q_i^{t-1} ermittelt wurden. Die Queryvektor-Modifikation in t zu q_i^t geschieht nach einem der bekannten Adaptions-Schemas, ohne dass die, zu diesem Zeitpunkt vorliegenden, Indexierungsfunktionen in $P(A_{IR})^t$ daran beteiligt wären.

Im zweiten Fall wird ein Adaptions-Schema ersetzt durch eine Reindexierung durch ein ausgewähltes Individuum in $P(A_{IR})^t$, d.h. Q_i wird reindexiert, indem der Vektor $h(Q_i)$ als Input in die ausgewählte Indexierungsfunktion verwendet wird, und der sich ergebende Queryvektor wird als Fixpunkt des Retrievals in der nächsten Feedback-Iteration verwendet.

3.9.6.3.1) Queryvektor-Feedback ohne Reindexierung

In der (t = 0)'ten Feedback-Iteration liegt der Original-Queryvektor $q_i^{t=0}$ vor, welcher mit einer Indexierungsfunktion $A_{IR} := A_{IR(Q)} = A_{IR(D)}$ erzeugt wurde, die entweder allein vorliegt, d.h. wenn eine Indexierungsfunktions-Mono-Repräsentation verwendet wird, oder die aus einer Menge IM_{IR} von Indexierungsfunktionen ausgewählt wurde, wenn eine Polyrepräsentation verwendet wird. Der Fall, dass bei einer Polyrepräsentation Q_i durch mehrere Indexierungsfunktionen $A_{IR,k}$ indexiert wurde, und dass somit mehrere Queryvektoren $q_{i,k}^{t=0}$ vorliegen, soll hier nicht betrachtet werden.

Mit dem Original-Queryvektor $q_i^{t=0}$ wird eine Retrieval-Dokumentvektorenmenge $DVM(q_i^{t=0})$ nachgewiesen, und die zu dieser Menge korrespondierenden Dokumente werden dem Agenten zur Relevanzbewertung vorgelegt. Mit den Relevanzurteilen wird eine Zerlegung von $DVM(q_i^{t=0})$ in $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ möglich. Alle Elemente aus $DVM(q_i^{t=0})_{rel}$ definieren Fitness-Cases der Form $(D_k, h(D_k), x_k)$, die zusammen mit $(Q_i, h(Q_i), q_i^{t=0})$ die Menge $FCM^{t=0}$ bilden.

Nachdem $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ vorliegen, kann direkt $q_i^{t=1}$ und $DVM(q_i^{t=1})$ gebildet werden, und die korrespondierenden Dokumente werden dem Agenten zur externen Bewertung vorgelegt. Während dieser Zeit soll die A_{IR}-Suche durchgeführt werden, wobei das IRS sich bereits in der Feedback-Iteration t=1 befindet. D.h. in der Initialisierungs-Generation t=0 findet nur eine Feedback-Operation

jedoch keine A_{IR}-Suche statt, und die A_{IR}-Initialisierungs-Population wird mit $P(A_{IR})^{t=1,\tau=1}$ angegeben. Sie wird bei einer Mono-Repräsentation z.B. durch Mutationsoperationen aus A_{IR} erzeugt, oder sie wird mit Hilfe der Menge IM_{IR} erzeugt. Sind in IM_{IR} mit δ mehr als μ Elemente vorhanden, so wird eine Zufallsauswahl ohne Zurücklegen durchgeführt. Sind in IM_{IR} weniger als μ Elemente vorhanden, so wird in der Initialisierungs-Generation eine (δ,λ)-EA durchgeführt, d.h. es wird aus $P(A_{IR})^{t=1,\tau=1} := IM_{IR}$ direkt eine Zwischenpopulation $ZP(A_{IR})^{t=1,\tau=1}$ durch Reproduktions-Operationen erzeugt.

Unabhängig wie die λ'elementige Zwischenpopulation $ZP(A_{IR})^{t=1,\tau=1}$ erzeugt wird, es folgt die Fitnessbewertung jedes Nachkommen $A_{IR,N(j)}{}^{t=1,\tau=1}$, indem jedes der Elemente aus $FCM^{t=0}$ durch $A_{IR,N(j)}{}^{t=1,\tau=1}$ indexiert wird, wodurch sich die Vektoren $A_{IR,N(j)}{}^{t=1,\tau=1}(Q_i)$ und $A_{IR,N(j)}{}^{t=1,\tau=1}(D_k)$, $\forall(D_k, h(D_k), x_k) \in FCM^{t=0}$, ergeben. Mit diesen Bezugspunkten wird die Distanz $d_{DVR}(q_i{}^{t=0}, A_{IR,N(j)}{}^{t=1,\tau=1}(Q_i))$ und die Distanzen $d_{DVR}(x_k, A_{IR,N(j)}{}^{t=1,\tau=1}(D_k))$ gebildet, mit denen die Fitnessbewertung $f(A_{IR,N(j)}{}^{t=1,\tau=1})$ des Kandidaten berechnet wird, wobei von genau einem Wert wie dem Mittelwert ausgegangen werden soll, der zu minimieren ist. Ist dies für alle λ Kandidaten der Zwischenpopulation $ZP(A_{IR})^{t=1,\tau=1}$ durchgeführt worden, so werden die μ Nachkommen mit der besten Fitness, d.h. mit dem kleinsten Fitnesswert, selektiert, die zusammen die Nachfolgepopulation $P(A_{IR})^{t=1,\tau=2}$ bilden.

Nachdem eine maximale Anzahl von $\tau(max)$ Generationen durchgeführt wurde, oder nachdem die externen Bewertungen der Elemente aus $DVM(q_i{}^{t=1})$ vorliegen, wird die A_{IR}-Suche in der Feedback-Iteration t=1 unterbrochen. Die sich ergebende Population von Indexierungsfunktionen soll in jedem Fall mit $P(A_{IR})^{t=1,\tau=\tau(max)}$ bezeichnet werden, die als Initialisierung der EA-Operation in der Relevanz-Feedback-Iteration t = 2 verwendet wird, d.h. $P(A_{IR})^{t=2,\tau=1} := P(A_{IR})^{t=1,\tau=\tau(max)}$.

Nachdem die Bewertungen der Elemente aus $DVM(q_i{}^{t=1})$ vorliegen und somit $DVM(q_i{}^{t=1})_{rel}$ und $DVM(q_i{}^{t=1})_{\overline{rel}}$ gebildet werden können, wird direkt eine neue Feedback-Iteration t=2 begonnen, indem $q_i{}^{t=2}$ durch eine Adaptionsoperation und $DVM(q_i{}^{t=2})$ durch eine Retrievaloperation bestimmt werden. Mit der Kenntnis der Menge $DVM(q_i{}^{t=1})_{rel}$ wird die Menge der Fitness-Cases zu $FCM^{t=1}$ aktualisiert, wobei die folgenden Möglichkeiten betrachtet werden sollen:
1) Erweiterung: $FCM^{t=1} = \{(Q_i, h(Q_i), q_i{}^{t=1}), DVM(q_i{}^{t=0})_{rel} \cup DVM(q_i{}^{t=1})_{rel}\}$.
2) Ersetzung: $FCM^{t=1} = \{(Q_i, h(Q_i), q_i{}^{t=1}), DVM(q_i{}^{t=1})_{rel}\}$.

Die Erweiterung liefert einen Effektivitätsvorteil im Sinne einer progressiven Evaluationsumgebung, da die Unbestimmtheit der Indexierungsfunktionen durch die steigende Anzahl von Stützpunkten gesenkt wird. Dies wird jedoch mit einem Effizienznachteil erkauft, da der Aufwand der Evaluation eines Kandidaten ansteigt. Der Effekt einer Ersetzung ist abhängig von der Anzahl der Elemente in $DVM(q_i{}^{t=1})_{rel}$ gegenüber der Anzahl in $DVM(q_i{}^{t=0})_{rel}$, die unterschiedlich sein können, wenn Retrieval-Mannigfaltigkeiten verwendet werden. Hierdurch kann auch ein Effektivitätsnachteil entstehen, wenn die neue Anzahl kleiner ist. Der Extremfall, dass nur ein Element in $DVM(q_i{}^{t=1})_{rel}$ liegt, zeigt, dass die Ersetzung ohne zusätzliche Regelungen nicht sinnvoll ist. Eine kombinierte Strategie könnte demgegenüber neue Elemente aufnehmen, bis eine Sollanzahl von Fitness-Cases erreicht ist, und in weiteren Feedback-Iterationen Ersetzungen durchführen, wobei immer die Dokumentvektoren die Fitness-Cases bilden, die dem momentan aktuellen Queryvektor am nächsten liegen.

Nachdem die Menge der Fitness-Cases zu $FCM^{t=1}$ aktualisiert wurde, werden die EA-Operationen in der Feedback-Iteration t = 2 begonnen, indem aus der μ'elementigen Initialisierungs-Population $P(A_{IR})^{t=2,\tau=1} := P(A_{IR})^{t=1,\tau=\tau(max)}$ Eltern gezogen werden, die Nachkommen $A_{IR,N(j)}{}^{t=2,\tau=1}$ bilden, die in die Zwischenpopulation $ZP(A_{IR})^{t=2,\tau=1}$ aufgenommen werden, bis darin λ Elemente enthalten sind. Für jeden dieser Nachkommen werden alle Elemente aus $FCM^{t=1}$ indexiert, wodurch die Vektoren $A_{IR,N(j)}{}^{t=2,\tau=1}(Q_i)$, $A_{IR,N(j)}{}^{t=2,\tau=1}(D_k)$, $\forall(D_k, h(D_k), x_k) \in FCM^{t=1}$, gebildet werden. Für die Bestimmung der Fitness $f(A_{IR,N(j)}{}^{t=2,\tau=1})$ werden die Distanzen zwischen dem aktualisierten Queryvektor $q_i{}^{t=1}$ und dem Vektor $A_{IR,N(j)}{}^{t=2,\tau=1}(Q_i)$, sowie den relevanten Dokumentvektoren x_k und den Vektoren $A_{IR,N(j)}{}^{t=2,\tau=1}(D_k)$ gebildet und aggregiert. Es folgt die Selektion zur Übernahme in die Nachfolgepopulation, und eine neue Generation τ=2, bis das Abbruchkriterium mit τ(max) erfüllt wird.

Das Gesamtverfahren wird in der Feedback-Iteration abgebrochen, in der keine neuen Dokumentvektoren bzw. keine neuen relevanten Dokumentvektoren mehr nachgewiesen werden können, d.h. wenn:
1) $DVM(q_i{}^t) = \varnothing$.
2) $DVM(q_i{}^t) \neq \varnothing \wedge DVM(q_i{}^t)_{rel} = \varnothing$.

Der Unterschied dieser beiden Möglichkeiten besteht darin, dass der erste Fall ohne ein Mitwirken des Agenten entschieden werden kann, während für den zweiten Fall Relevanzentscheidungen notwendig werden. Wird eine Queryvektor-Adaption unterstellt, so ist der erste Fall ein absolutes Abbruchkriterium, da kein Adaptionsprozess durchgeführt werden kann und $q_i{}^{t+p} := q_i{}^t$, p = 1, ..., gelten würde. Die zweite Möglichkeit erfordert jedoch nicht unbedingt einen Abbruch, da mindestens ein nicht relevanter Dokumentvektor vorliegt ($DVM(q_i{}^t)_{\overline{rel}} \neq \varnothing$), sodass eine rein negative Relevanz-Feedback-Strategie durchgeführt werden kann, die einen abweichenden Vektor $q_i{}^{t+1}$ liefert, für den eine neue Retrievalmenge gesucht wird.

In der Feedback-Iteration t wird eine Indexierungsfunktions-Population $P(A_{IR})^{t,\tau}$ auf der Basis der Menge FCM^{t-1} der Fitness-Cases entwickelt. Eine leere Menge $DVM(q_i{}^t)_{rel}$ führt dazu, dass diese Menge unverändert bleibt, d.h. $FCM^t = FCM^{t-1}$. Wurden genügend Generationen in der vorangegangenen Feedback-Iteration durchgeführt, so besteht nun keine Notwendigkeit mit der gleichen Menge an Fitness-Cases eine weitere Entwicklung der Indexierungsfunktions-Population durchzuführen, sodass dies als Abbruchkriterium verwendet werden kann. Sollten das Qualitätsniveau der Population aufgrund eines zu frühen Abbruchs in der Feedback-Iteration t-1 einen Schwellenwert nicht überschreiten, so kann dies für eine Weiterentwicklung in der Feedback-Iteration t betrachtet werden, bis ein qualitätsabhängiges Abbruchkriterium greift.

3.9.6.3.2) Queryvektor-Feedback mit Reindexierung

Das dargestellte Verfahren ist charakterisierbar durch die fehlende Anwendung der neu erzeugten Indexierungsfunktionen auf die Original-Query. Die Veränderung des Queryvektors in der ersten Feedback-Iteration, d.h. von $q_i{}^{t=0}$ zu $q_i{}^{t=1}$, wird ausschließlich durch die Schwerpunkte $\bar{s}_{DVM(i,rel)}{}^{t=0}$ und $\bar{s}_{DVM(i,\overline{rel})}{}^{t=0}$ der relevanten und nicht relevanten Dokumentvektoren festgelegt, ohne dass eine der Indexierungsfunktionen $A_{IR,N(j)}{}^{t=0,\tau}$ beteiligt wäre. Dies äussert sich in der oben dargestellten Möglichkeit

der parallelen Durchführung der EA-Operationen und der Bestimmung von $q_i^{t=1}$ durch eine gemischte Feedback-Adaption, da beide Bereiche unabhängig durchgeführt werden können, weil jeweils keine Daten des anderen Bereichs benötigt werden.

Wird in jeder Feedback-Iteration t eine genügend große Anzahl von EA-Generationen τ durchgeführt, dass A_{IR}-Individuen mit einer hinreichend guten Indexierungsqualität bezogen auf die jeweilige Menge an Fitness-Cases vorliegen, so könnte ein Verfahrens-Szenario betrachtet werden, bei dem Q_i durch einen oder mehrere Indexierungsfunktionen reindexiert wird, und das der oder die daraus abgeleiteten Queryvektoren als Fixpunkte für die nachfolgende Feedback-Iteration verwendet werden.

Es soll der Fall betrachtet werden, dass das beste Individuum $A_{IR,best}^{t=0,\tau(max)}$ aus der End-Generationen $\tau(max)$ verwendet wird, um Q_i zu indexieren, d.h. $h(Q_i)$ wird als Inputvektor in die Indexierungsfunktion verwendet, die einen Outputvektor $A_{IR,best}^{t=0,\tau(max)}(Q_i) \in DVR$ liefert. Um festzulegen, welche der Individuen in der Population $P(A_{IR})^{t=0,\tau(max)}$ das beste ist, muss die Fitness jedes Individuums berechnet werden, wobei die Indexierung von Q_i Element der Fitness-Cases ist, d.h. $A_{IR,best}^{t=0,\tau(max)}(Q_i)$ kann nach seiner Erzeugung gespeichert und später wiederverwendet werden. Der nächste Schritt besteht darin, $A_{IR,best}^{t=0,\tau(max)}(Q_i)$ als Fixpunkt $q_i^{t=1}$ der nächsten Feedback-Iteration zu verwenden, auf dessen Basis die Retrievalmenge $DVM(q_i^{t=1}) = DVM(A_{IR,best}^{t=0,\tau(max)}(Q_i))$ erzeugt wird. Die Bewertung der korrespondierenden Dokumente durch den Agenten erzeugt $DVM(q_i^{t=1})_{rel}$ und $DVM(q_i^{t=1})_{\overline{rel}}$, mit denen die Menge der Fitness-Cases zu $FCM^{t=1}$ aktualisiert wird. Die μ Indexierungsfunktions-Individuen in $P(A_{IR})^{t=0,\tau(max)}$ werden als Initialisierungs-Population $P(A_{IR})^{t=1,\tau=1}$ verwendet, die eine λ'elementige Zwischenpopulation $ZP(A_{IR})^{t=1,\tau=1}$ erzeugt. Den darin enthaltenen Nachkommen wird ihr Fitnesswert auf der Basis der aktuellen Menge der Fitness-Cases zu $f(A_{IR,N(j)}^{t=1,\tau=1} \mid FCM^{t=1})$ bestimmt. Auf der Basis dieser Fitnesswerte wird die Selektion zur Übernahme in die Nachfolgegeneration $\tau = 2$ durchgeführt, bis wiederum die maximale Anzahl $\tau = \tau(max)$ an Generationen erreicht wird. Aus der sich ergebenden Population $P(A_{IR})^{t=1,\tau(max)}$ wird wiederum das beste Individuum $A_{IR,best}^{t=1,\tau(max)}$ ausgewählt, und die bereits berechnete Queryvektor $A_{IR,best}^{t=1,\tau(max)}(Q_i)$ wird als Fixpunkt $q_i^{t=2}$ der nächsten Feedback-Iteration verwendet. Der Verfahrensabbruch erfolgt dann, wenn die Retrievalmenge einer Feedback-Iteration t, d.h. $DVM(q_i^t) = DVM(A_{IR,best}^{t-1,\tau(max)}(Q_i))$, leer sein sollte.

3.9.7) Reformulierungs-Relevanz-Feedback

In den bislang vorgestellten Verfahren zum Relevanz-Feedback wurde nicht spezifiziert, um welche Art der Relevanz es sich dabei handeln soll. Implizit wurde jedoch eine Problemlösungs-Relevanz unterstellt, d.h. parallel zu der Präsentation der nachgewiesenen Dokumente wird der Agent aufgefordert zu bewerten, ob der Inhalt eines Dokumentes seiner Ansicht nach zu der Problemlösung beigetragen hat bzw. beitragen könnte. Der Relevanzwert wird auf diese Weise binär definiert, d.h. der Agent soll einem Dokument eine 0 zuordnen, wenn der Inhalt eines Dokumentes keinen oder einen minimalen Beitrag zur Problemlösung beiträgt, bzw. einem Dokument wird eine 1 zugeordnet, wenn der Inhalt ein nicht vernachlässigbarer Beitrag leistet. Auf die vielfältigen Probleme der Problemlösungs-Relevanz wurde bereits im Abschnitt 3.9.1.2) eingegangen, wie z.B. die prinzipiellen Beschränkungen durch die Introspektion des Agenten, oder die Beschränkungen während des Problemlösungsprozesses Beiträge zu

bewerten, da die Möglichkeit besteht, dass positive Beiträge in eine Sackgasse führen (lokales Optimum im Zustandsraummodell der Problemlösung; siehe auch Abschnitt 3.9.1.3)).

Wird die Reformulierungs-Relevanz als Grundlage für Feedback-Verfahren verwendet, so ergeben sich andere Vorgehensweisen, je nachdem, was reformuliert wird, und wie dies geschieht. Bezüglich der Frage was reformuliert wird, können die folgenden prinzipiellen Antworten gegeben werden:
1) Reformulierung der impliziten mentalen Strukturen.
2) Reformulierung der Query.
3) Reformulierung anderer Texttypen wie Beschreibung des Gesamtproblems bzw. von Teilproblemen, Problemlösungsvorschläge, Ideen, Assoziationen, Kommentare zu anderen Texten, ...

Die Reformulierung mentaler Strukturen eines Agenten ist für eine externe Instanz nicht erfassbar, und kann somit nur durch Introspektion des Agenten quantifiziert werden. Schwerpunktmäßig soll die Reformulierung der Query betrachtet werden, da dies einen Brückenschlag zwischen der konventionellen Sichtweise des Queryvektor-Feedbacks und der Reformulierungs-Sichtweise ermöglicht. Bei einer Reformulierung der Problembeschreibung wird davon ausgegangen, dass der Agent explizite Beschreibungen seines Problems, Teilprobleme, Problemlösungsvorschläge, Ideen, Assoziationen, Kommentare zu anderen Texten u.ä. besitzt, diese im Verlauf einer längeren Interaktion mit dem IRS ständig aktualisiert, und diese dem IRS zugänglich sind. D.h. das IRS kann Teile dieser Aufzeichnungen als Ergänzung zu den externen Informationsbedürfnissen in Form der Query verwenden, indem z.B. eine gegebene Query mit diesen Zeichensequenzen expandiert und dann indexiert wird.

Im weiteren soll davon ausgegangen werden, dass, wie in Kapitel 4) eine reelle Relevanzfunktion rel(.) verwendet wird, im Gegensatz zu den binären Relevanzfunktionen, die bei den bisherigen Relevanz-Feedback-Verfahren unterstellt wurden, d.h. es wird unterstellt:

$$\text{rel}(.): R^{+n} \rightarrow [0, 1]: x \mid\rightarrow \text{rel}(x). \tag{488}$$

3.9.7.1) Reformulierung der Query

Bei der konventionellen Sichtweise wird genau eine Query Q^t durch den Agenten formuliert, die durch das IRS zu dem Queryvektor $q_i^{t=0}$ indexiert wird, der mit Hilfe einer Retrieval-Strategie eine Retrieval-Dokumentvektorenmenge $DV(q_i^{t=0})$ spezifiziert. Zu den Dokumentvektoren $x_{j|q(i,t)}$ aus $DV(q_i^{t=0})$ werden Relevanzbewertungen $\text{rel}(x_{j|q(i,t)})$ durch den Agenten abgegeben, die zusammen mit q_i^t und Feedbackparametern wie α, β, χ als Input in die Relevanz-Feedback-Funktion rfb(.) verwendet werden, die als Output den adaptierten Queryvektor q_i^{t+1} liefert. Dies ergibt eine Verfahrenssequenz:

$$\begin{aligned} Q^{t=0} \rightarrow q_i^{t=0} \rightarrow DV(q_i^{t=0}) \rightarrow \text{rel}(x_{j|q(i,t)}),\ x_{q(i,t)j} \in DV(q_i^{t=0}) \\ \rightarrow q_i^{t+1} = \text{rfb}(q_i^t, DV(q_i^t)_{rel}, DV(q_i^t)_{\overline{rel}}). \end{aligned} \tag{489}$$

Im Gegensatz zu der automatischen Reformulierung des Queryvektors durch das IRS steht die individuelle Reformulierung der Query durch den Agenten zu Q_i^{t+1}, die durch das IRS regulär indexiert wird zu q_i^{t+1}:

$$Q^{t=0} \rightarrow q_i^{t=0} \rightarrow DV(q_i^{t=0}) \rightarrow Q_i^{t+1} \rightarrow q_i^{t+1} = A_{IR(Q)}(Q^{t+1}). \quad (490)$$

Die Sequenz von Formulierung der Query, Indexierung, Retrieval und Reformulierung der Query kann für sich genommen als Feedback-Strategie verwendet werden, ohne dass Relevanzwerte explizit beim Agenten oder dem IRS auftauchen.

Sollen demgegenüber explizite Reformulierungs-Relevanzwerte betrachtet werden so ergeben sich die folgenden Möglichkeiten:

1) Herleitung von Reformulierungs-Relevanzwerten aus den Queries bzw. den Queryvektoren.
   1.1) Verwendung der Ähnlichkeit bzw. Distanz der Queries Q_i^t und Q_i^{t+1}.
   1.2) Verwendung der Ähnlichkeit bzw. Distanz der Queryvektoren q_i^t und q_i^{t+1}.
2) Der Agent wird nach Reformulierungs-Relevanzwerten explizit gefragt, ohne dass eine Query-Reformulierung vom Agenten verlangt wird.
3) Mischstrategien mit Query-Reformulierung und Frage nach Reformulierungs-Relevanzwerten.

3.9.7.1.1) Relevanzwerte aus Queries bzw. Queryvektoren

Bei der Herleitung aus Queries besteht der Nachteil darin, dass eine Ähnlichkeits- bzw. Distanzfunktion von zwei Zeichenketten definiert werden muss, was z.B. durch eine Levenshtein-Funktion oder auf der Basis von Median-Strings durchgeführt werden kann (siehe Kohonen (1985[183])). Damit neben der Metrik $d_{DVR}(.)$ nicht noch ein Distanzmaß im Raum der Zeichensequenzen festgelegt werden muss, sollen im weiteren die Queryvektoren q_i^t und q_i^{t+1} verwendet werden, um Relevanzschätzungen durch das IRS durchzuführen. Bei binären Reformulierungs-Relevanzwerten bedeutet dies, dass aus der Kenntnis von q_i^t und q_i^{t+1} die Dokumentvektoren in $DV(q_i^t)$ in zwei bzw. drei Klassen zerlegt werden. In der einen Klasse befinden sich alle Dokumentvektoren, die näher an q_i^t als an q_i^{t+1} liegen, und in der anderen alle Dokumentvektoren, die näher an q_i^{t+1} als an q_i^t liegen, und in der dritten alle Dokumentvektoren, die den gleichen Abstand von beiden Queryvektoren besitzen.

Bei reellen Relevanzwerten soll aus der Kenntnis von q_i^t und q_i^{t+1} die Reformulierungs-Relevanzwerte für die Dokumentvektoren $x_{j|q(i,t)}$ aus $DV(q_i^t)$ berechnet werden, mit $rel_{ref}(x_{j|q(i,t)})^\wedge \in [0, 1]$. Wird zunächst der Fall betrachtet, dass in $DV(q_i^t)$ genau ein Element enthalten ist, d.h. $DV(q_i^t) := \{x_{q(i,t)}\}$, wobei es sich um den nächsten Dokumentvektor handelt, so soll die geschätzte Reformulierungs-Relevanz $rel_{ref}(.)^\wedge$ allgemein als Funktion der Distanz $d_{DVR}(q_i^t, q_i^{t+1})$ beschrieben werden:

$$rel_{ref}(x_{q(i,t)})^\wedge := rel_{ref}(x_{q(i,t)} \mid q_i^t, q_i^{t+1})^\wedge = f(d_{DVR}(q_i^t, q_i^{t+1})). \quad (491)$$

Entsprechend dieser Darstellung stellt sich die Frage, nach welchen Kriterien ein solcher Funktionszusammenhang weiter spezifiziert werden soll. Die naheliegendste Modellierung ist die Verwendung einer monoton steigenden Funktion, d.h. je umfassender die Reformulierung ist, gemessen durch die Distanz $d_{DVR}(q_i^t, q_i^{t+1})$, desto größer ist die Reformulierungs-Relevanz des verursachenden Wissens, d.h. des Dokumentes $D_{q(i,t)}$. Die Distanzen $d_{DVR}(x_{q(i,t)}, q_i^t)$ bzw. $d_{DVR}(x_{q(i,t}, q_i^{t+1})$ spielen bei diesem Schätzverfahren keine Rolle, obwohl $d_{DVR}(x_{q(i,t)}, q_i^t) = \min\{d_{DVR}(x_j, q_i^t) \mid \forall x_j \in D^t\}$ gilt.

Es kann z.B. ein Funktionsverlauf verwendet werden, der bei $d_{DVR}(.) = 0$ einen Wert $rel_{ref}(.)\hat{} = 0$ besitzt, und danach sich für asympthotisch der 1 nähert (siehe $y = 2/\pi * \arctan(x)$, für $x \geq 0$ in Abb. 102)). Es kann auch eine Schwellenwertfunktion verwendet werden, die bei $d_{DVR}(.) = 0$ einen Wert $rel_{ref}(.)\hat{} = 0$ besitzt, dann eine lineare Steigung von z.B. 1/2 besitzt (siehe Abb. 102)), bis die Gerade die zur x-Achse parallele Gerade $rel_{ref}(.)\hat{} = 1$ schneidet, und danach konstant auf der rechten Intervallgrenze bleibt ($y = 1$, wenn $x > 2$).

Abb. 102) Beispielsverläufe für Reformulierungs-Relevanzfunktionen 1

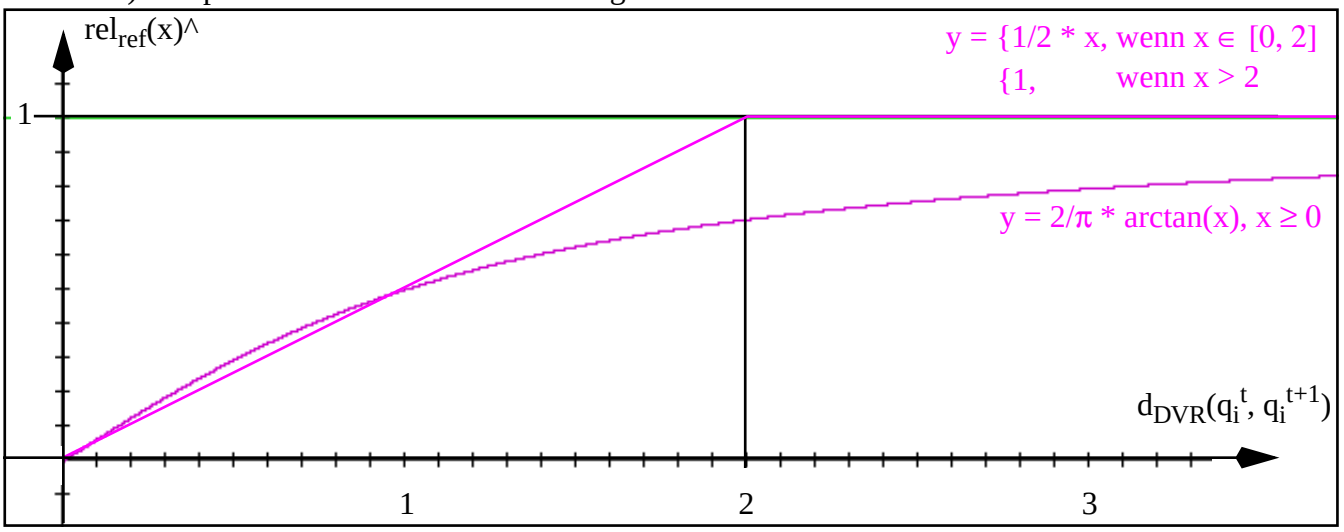
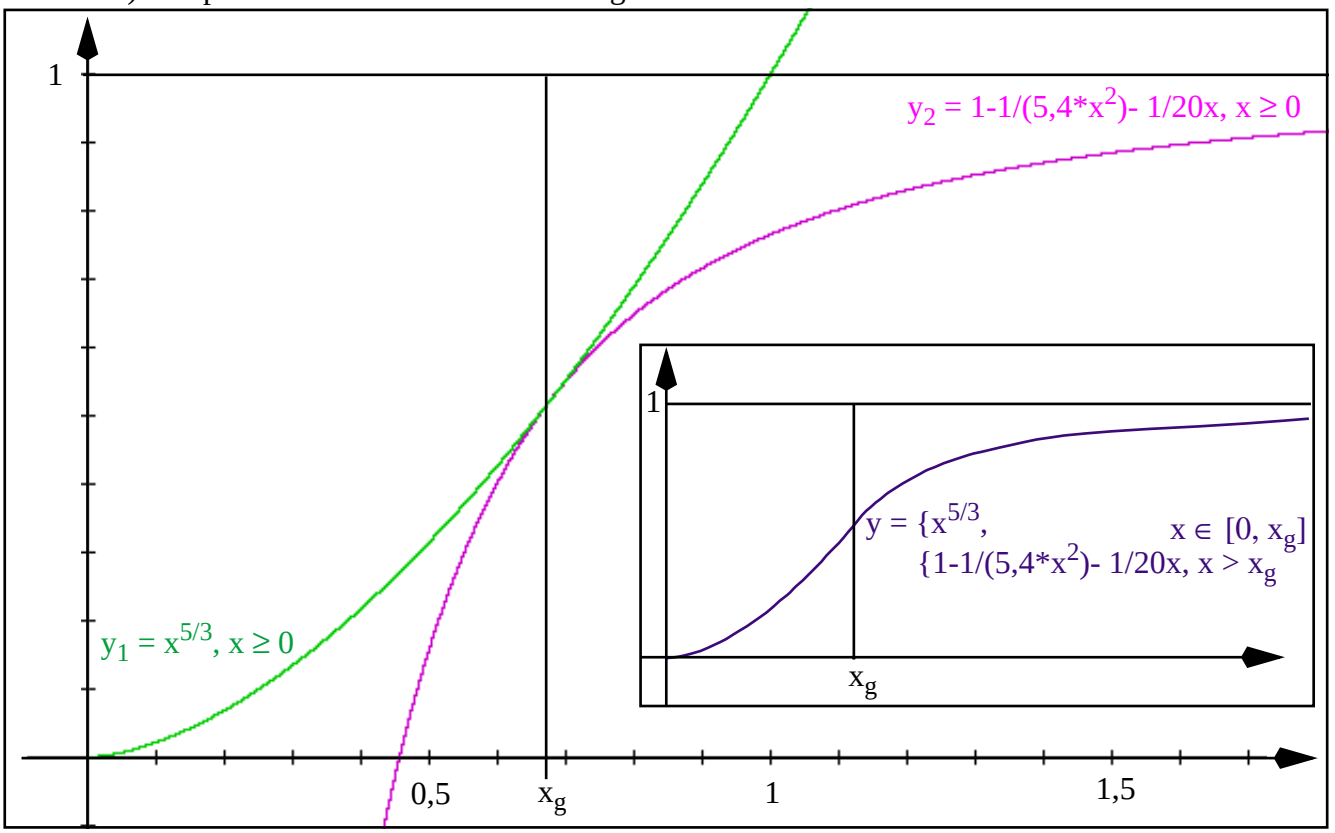


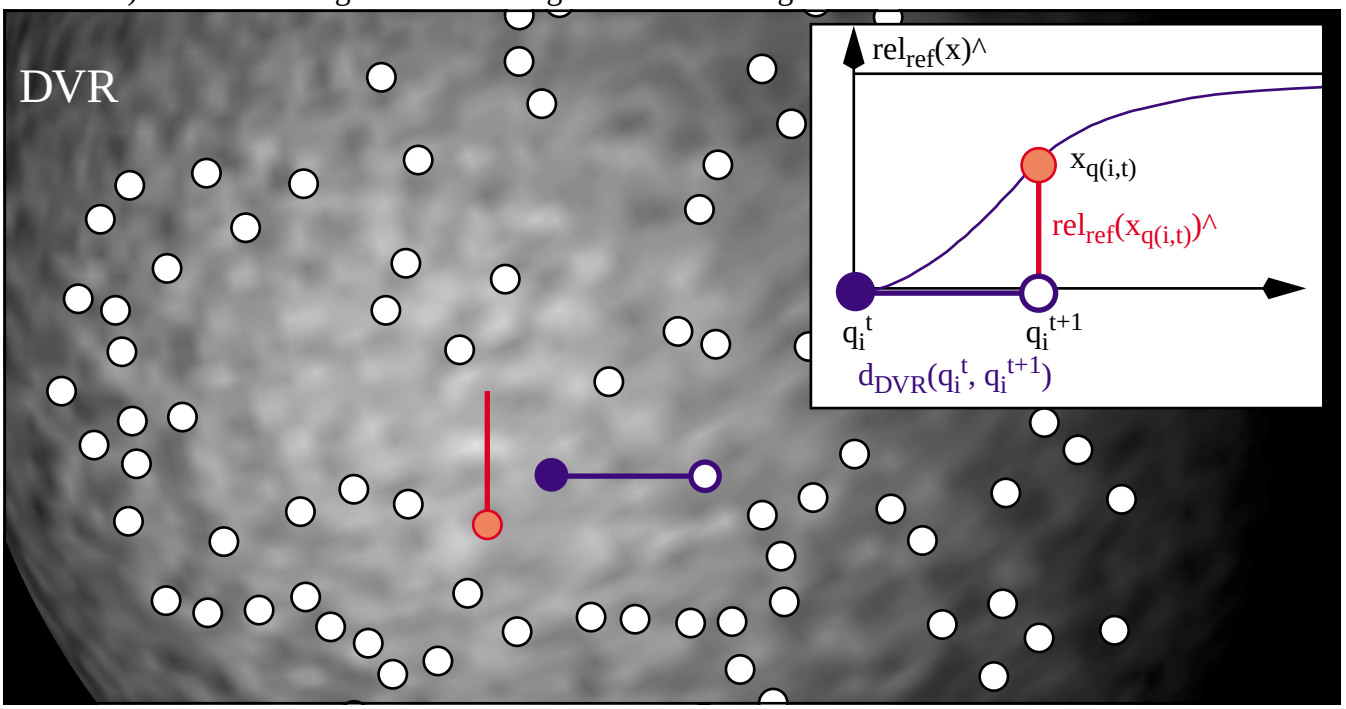
Abb. 103) Beispielsverläufe für Reformulierungs-Relevanzfunktionen 2



Denkbar ist auch die Komposition aus zwei sich schneidenden Funktionen, z.B. einer Potenzfunktion wie $y = x^{5/3}$ für ein Intervall $x \in [0, x_g]$ und einer gebrochen linearen Funktion $y = 1 - 1/(20x) - 1/(5{,}4x^2)$ für $x > x_g$, mit x_g als dem bzw. dem ersten Schnittpunkt der beiden Funktionen (siehe Abb. 103)).

Die Vorgehensweise der Relevanzschätzung ist in Abb. 104) dargestellt. Nachdem q_i^t und der nächste Dokumentvektor $x_{q(i,t)}$ vorliegen, und die reformulierte Query Q_i^{t+1} zu q_i^{t+1}indexiert wurde, kann die Distanz $d_{DVR}(q_i^t, q_i^{t+1})$ berechnet werden, die als Input in die Reformulierungs-Relevanzfunktion $rel_{ref}(x)^\wedge$ verwendet wird. Es ergibt sich ein Relevanzwert $rel_{ref}(x_{q(i,t)})^\wedge$, der dem Dokumentvektor $x_{q(i,t)}$ zugeordnet wird.

Abb. 104) Reformulierungs-Relevanz bei genau einem nachgewiesenen Dokumentvektor

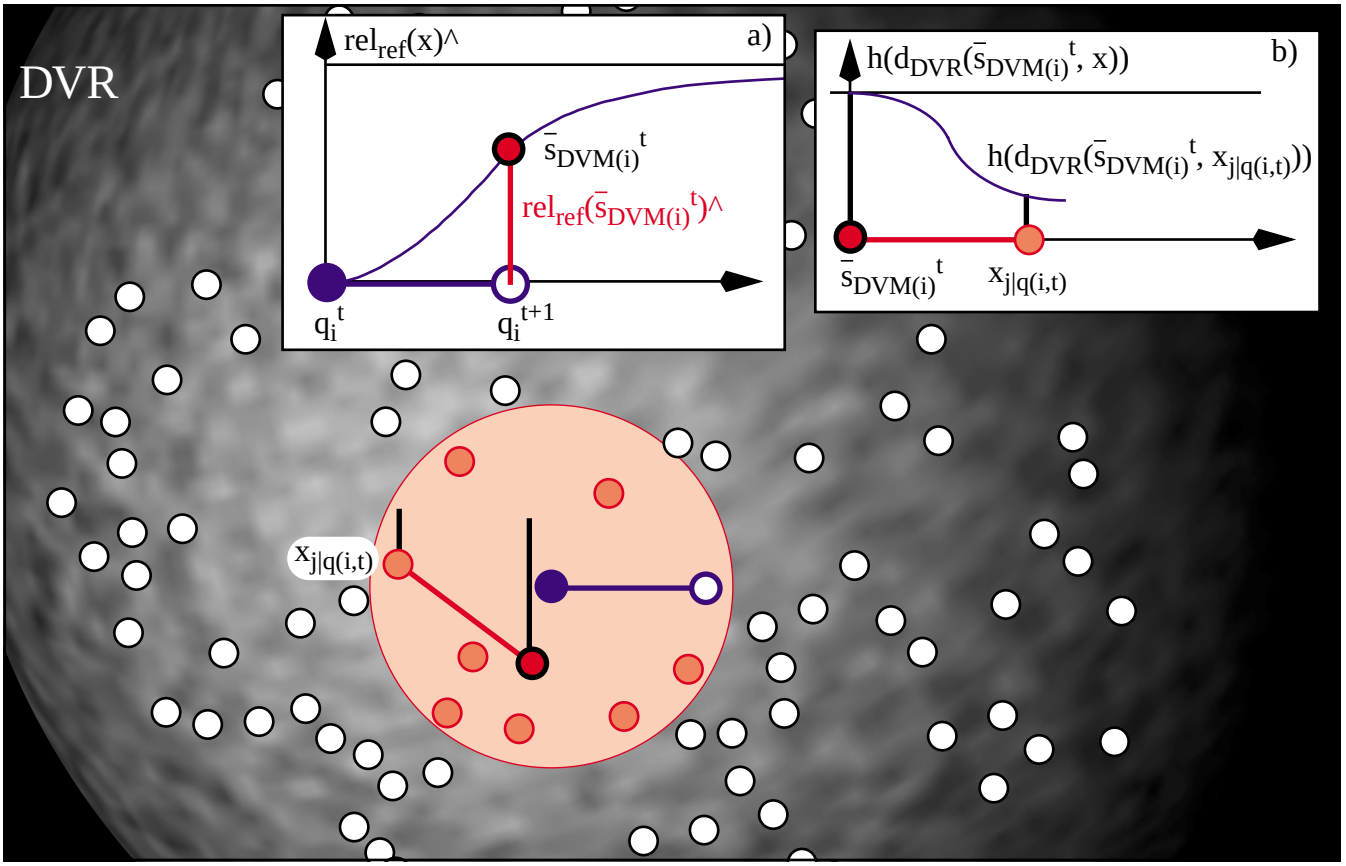


Problematischer wird die Situation, wenn in $DV(q_i^t)$ mehrere Dokumentvektoren enthalten sind, da die Wirkung der Inhalte nicht mehr direkt einem einzelnen Dokument zugeordnet werden kann. Wird angenommen, dass alle nachgewiesenen Dokumentvektoren den gleichen Einfluss auf die Reformulierung hatten, so wird ein Relevanzwert nach dem oben dargestellten Verfahren bestimmt, und allen Dokumentvektoren zugeordnet. Wird angenommen, dass die Dokumentvektoren unterschiedliche Relevanzwerte in Abhängigkeit von ihrem Inhalt und somit von ihrer Position im DVR besitzen, so muss ein Verfahren angewendet werden, das einen Reformulierungs-Relevanzwert auf die vorhandenen Dokumentvektoren umrechnet oder verteilt. Beispielsweise kann im ersten Schritt der arithmetische Mittelwertsvektor $\bar{s}_{DVM(i)}^t$ der Dokumentvektoren aus $DV(q_i^t)$ gebildet werden, der als Repräsentant von $DV(q_i^t)$ verwendet wird. Die Reformulierungs-Relevanz $rel_{ref}(\bar{s}_{DVM(i)}^t)^\wedge$ wird für diesen Vektor nach dem oben dargestellten Verfahren bestimmt, und alle Dokumentvektoren $x_{j|q(i,t)}$ aus $DV(q_i^t)$ erhalten einen Relevanzwert, der eine Funktion ihres Abstandes zu $\bar{s}_{DVM(i)}^t$ ist, z.B. berechnet durch eine LWR-Operation mit genau einem Gewichtungsterm:

$$rel_{ref}(x_{j|q(i,t)})^\wedge = h(d_{DVR}(\bar{s}_{DVM(i)}^t, x_{j|q(i,t)})) * rel_{ref}(\bar{s}_{DVM(i)}^t)^\wedge. \quad (492)$$

Diese Situation ist in Abb. 105) dargestellt. Nach der Festlegung von q_i^t, q_i^{t+1} und dem Mittelwertsvektor $\bar{s}_{DVM(i)}^t$ wird die Reformulierungs-Relevanz $rel_{ref}(\bar{s}_{DVM(i)}^t)^\wedge$ wie in dem Fall mit genau einem Dokumentvektor bestimmt (Teilabbildung a)). Für jeden Dokumentvektor $x_{j|q(i,t)}$ aus der Retrievalmenge $DV(q_i^t)$ wird der Gewichtungsfaktor $h(d_{DVR}(\bar{s}_{DVM(i)}^t, x_{j|q(i,t)}))$ bestimmt (Teilabbildung b)), der multipliziert mit $rel_{ref}(\bar{s}_{DVM(i)}^t)^\wedge$ als Schätzung der Reformulierungs-Relevanz $rel_{ref}(x_{j|q(i,t)})^\wedge$ verwendet wird.

Abb. 105) Reformulierungs-Relevanz bei mehreren nachgewiesenen Dokumentvektoren



Welchen Relevanzwert einer der Dokumentvektoren $x_{j|q(i,t)}$ aus $DV(q_i^t)$ zugeordnet bekommt ist somit zum einen von dem Vektor im DVR abhängig, der als Repräsentant ausgewählt wird, zum anderen von der verwendeten Distanz-Gewichtungsfunktion $h(d_{DVR}(., .))$. Anstatt den Mittelwertsvektor $\bar{s}_{DVM(i)}^t$ als Repräsentant zu wählen, könnte auch der Median-Dokumentvektor verwendet werden, als der Dokumentvektor aus $DV(q_i^t)$, der $\bar{s}_{DVM(i)}^t$ am nächsten liegt.

Der prinzipielle Nachteil dieser Vorgehensweise besteht wie auch bei allen Ansätzen zur Problemlösungs-Relevanz darin, dass nicht zwischen dem zugänglichen Wissen und dem gesamten Wissen in dem Dokument unterschieden werden kann, was nur bei einem Zugriff auf mentale Strukturen oder detaillierte Aufzeichnungen von Introspektions-Operationen gelingen könnte.

3.9.7.1.2) Direkte Frage nach Reformulierungs-Relevanzwerten

Bei dieser Vorgehensweise wird der Agent explizit nach Reformulierungs-Relevanzwerten gefragt. Es wird davon ausgegangen, dass nach der Präsentation der nachgewiesenen Dokumente der Agent Wissen aus den Dokumenten extrahiert, und dieses ihm zugängliche Wissen in seine mentalen Strukturen integriert. Auf diese Weise verändern sich seine mentalen Strukturen, die das zugrundeliegende Problem und einen bzw. mehrere Problemlösungsansätze repräsentieren. Da im Kontext der Query-Reformulierung nicht nach der Reformulierung der mentalen Strukturen gefragt wird, muss der Agent aus den reformulierten mentalen Strukturen des Problems und seiner Lösungsvorschläge ein externes Informationsbedürfnis formulieren, ohne dass dieses an das Sprachsynthesеareal weitergeleitet wird, d.h. das Informationsbedürfnis liegt als mentale Struktur im Problemlösungsareal vor. Dies geschieht unter der Voraussetzung, dass eine mentale Prüfung der neuen Lösungsvorschläge ergeben hat, dass diese den Kriterien einer akzeptablen Lösung noch nicht genügen, da ansonsten kein Bedarf an weiteren Operationen und speziell an der Formulierung eines externen Informationsbedürfnisses bestehen würde.

Bei der expliziten Frage nach Reformulierungs-Relevanzwerten kann zwischen einer Einzel- und einer Gesamtschätzung unterschieden werden:

1) Einzelschätzung: Schätzen Sie den Einfluss des Dokumentes $D_{j|q(i,t)}$ auf die Reformulierung ihres externen Informationsbedürfnisses.
2) Gesamtschätzung: Schätzen Sie den Einfluss der nachgewiesenen Dokumente auf die Reformulierung ihres externen Informationsbedürfnisses.

Bei einer Einzelschätzung wird ein einzelner Reformulierungs-Relevanzwert bezüglich eines Dokumentes $D_{j|q(i,t)}$, den der Agent quantifiziert, direkt ohne weitere Operationen als $rel_{ref}(x_{j|q(i,t)})$ interpretiert. Mit den reellen Relevanzwerten $rel_{ref}(x_{j|q(i,t)}) \in [0, 1]$ kann auch eine automatische Reformulierung des Queryvektors zu q_i^{t+1} mit Hilfe einer Relevanz-Feedbackfunktion durchgeführt werden, genauso wie mit Hilfe der binären Relevanzwerte, d.h. mit den Mengen $DV(q_i^t)_{rel}$ und $DV(q_i^t)_{\overline{rel}}$, bei den oben vorgestellten Ansätzen zum Queryvektor-Relevanz-Feedback. Diese Feedback-Verfahren sollen im Rahmen der Relevanz-Approximationsmodelle in Kapitel 4) dargestellt werden.

Bei einer Gesamtschätzung wird der Menge der nachgewiesenen Dokumente ein Relevanzwert zugeordnet, wobei wie im Fall der explizit reformulierten Query dieser Wert einem Repräsentanten der Dokumentmenge zugeordnet werden könnte. Wird der arithmetische Mittelwertsvektor $\bar{s}_{DVM(i)}^t$ hierfür verwendet, so wird der vom Agenten quantifizierte Reformulierungs-Relevanzwert als $rel_{ref}(\bar{s}_{DVM(i)}^t)^\wedge$ interpretiert. Den Dokumentvektoren $x_{j|q(i,t)}$ kann dann auf der Basis ihrer Distanzen zu $\bar{s}_{DVM(i)}^t$ und einer Distanz-Gewichtungsfunktion $h(d_{DVR}(\bar{s}_{DVM(i)}^t, x))$ ein eigener Reformulierungs-Relevanzwert $rel_{ref}(x_{j|q(i,t)})^\wedge$ zugeordnet werden.

Wird eine vorhandene mentale Struktur, die ein externes Informationsbedürfnis repräsentiert, an das Sprachsynthesеareal weitergeleitet, so liegt nach der Transformation eine Zeichensequenz vor, die als Query Q_i^{t+1} verwendet werden kann. Nur diese Transformation unterscheidet diese Variante von der nachfolgenden Variante, bei der die Query-Reformulierung Q_i^{t+1} und explizite, durch den Agenten angegebene Relevanzwerte vorliegen, die gemeinsam weiterverarbeitet werden.

3.9.7.1.3) Relevanzwerte aus Queryvektoren und Bewertung

Liegt eine explizite Query-Reformulierung Q_i^{t+1} vor, so liegen in Kombination mit einer Einzel- bzw. einer Gesamtschätzung der Reformulierungs-Relevanzwerte zunächst die folgenden Werte vor:

1) Q_i^{t+1}, $rel_{ref}(x_{j|q(i,t)})$, $\forall x_{j|q(i,t)} \in DV(q_i^t)$.
2) Q_i^{t+1}, $rel_{ref}(\bar{s}_{DVM(i)}^t)^\wedge$, bzw. nach der Verteilung der Relevanzwerte ergibt sich: Q_i^{t+1}, $rel_{ref}(\bar{s}_{DVM(i)}^t)^\wedge$, $rel_{ref}(x_{j|q(i,t)})^\wedge$, $\forall x_{j|q(i,t)} \in DV(q_i^t)$).

Eine Redundanz entsteht dadurch, dass durch die Indexierung von Q_i^{t+1} ein Queryvektor q_i^{t+1} als Referenz erzeugt wird, und dass mit den Relevanzwerten zusammen mit q_i^t und einem Adaptionsverfahren automatisch ein Queryvektor q_i^{t+1} erzeugt werden kann:

$$q_i^{t+1} = A_{IR(Q)}(Q^{t+1}) \text{ und } q_i^{t+1} = rfb(q_i^t, DV(q_i^t), rel_{ref}(x_{j|q(i,t)}), \forall x_{j|q(i,t)} \in DV(q_i^t)). \quad (493)$$

Ziel ist es, diese Redundanzen zu nutzen, um Adaptionsverfahren anzupassen, wobei das Relevanz-Feedbackverfahren rfb(.) angepasst werden kann, oder die Distanz-Gewichtungsfunktion $h(d_{DVR}(.))$.

Zunächst soll der Fall betrachtet werden, dass die Query durch den Agenten zu Q_i^{t+1} reformuliert wird, und dass allen nachgewiesenen Dokumenten Reformulierungs-Relevanzwerte durch den Agenten zugeordnet werden, sodass $DV(q_i^t)$ in $DV(q_i^t)_{rel}$ und $DV(q_i^t)_{\overline{rel}}$ zerlegt werden kann. Die Zerlegung kann durch binäre sowie durch reelle Relevanzwerten durchgeführt werden, wobei reelle Werte zusätzlich einen Schwellenwert erfordern, wenn ein Intervall [0, 1] verwendet wird.

Wird Q_i^{t+1} mit der regulären Indexierungsfunktion $A_{IR(Q)}$ indexiert, so soll der entstandene Queryvektor als Referenzvektor mit $q_{i,ref}^{t+1}$ bezeichnet werden. Gesucht wird nach einem Lernverfahren bzw. einer Relevanz-Feedbackstrategie, die z.B. q_i^t, $\bar{s}_{DVM(i,rel)}^t$ und $\bar{s}_{DVM(i,\overline{rel})}^t$ als Input verwendet, und $q_{i,ref}^{t+1}$ als Output liefert. Soll dies mit der gemischten Relevanz-Feedbackstrategie durchgeführt werden, so ergibt sich:

$$q_{i,ref}^{t+1} = \alpha * q_i^t + \beta * \bar{s}_{DVM(i,rel)}^t - \chi * \bar{s}_{DVM(i,\overline{rel})}^t, \text{ mit } \alpha, \beta, \chi \geq 0. \quad (494)$$

Die entsprechende Gleichung ist unterspezifiziert, da drei frei wählbare Parameter als Lernfaktoren vorliegen. Wird eine positive Relevanz-Feedbackstrategie durchgeführt, so ist die entsprechende Gleichung mit zwei Parametern immer noch unterspezifiziert mit:

$$q_{i,ref}^{t+1} = \alpha * q_i^t + \beta * \bar{s}_{DVM(i,rel)}^t, \text{ mit } \alpha, \beta \geq 0. \quad (495)$$

Wird z.B. $\alpha := 1$ gesetzt, so kann der Parameter β bestimmt werden, der den Agenten mit seiner Reformulierungsfähigkeit beschreibt.

Anstatt der Festlegung von zwei Lernparametern bei der gemischten Strategie bzw. eines Parameters bei der positiven Strategie kann auch ein Gleichungssystem herangezogen werden, wenn das Verhalten über mehr als eine Feedback-Iteration herangezogen wird. Bei einer gemischten Strategie müssen drei Iterationen mit den Queryvektoren $q_{i,ref}^t$ bis $q_{i,ref}^{t+3}$ betrachtet werden, wobei unter der Annahme, dass die drei Parameter gleich sind, diese bestimmt werden können:

$$q_{i,ref}^{t+1} = \alpha * q_{i,ref}^{t} + \beta * \bar{s}_{DVM(i,rel)}^{t} - \chi * \bar{s}_{DVM(i,\overline{rel})}^{t},$$
$$q_{i,ref}^{t+2} = \alpha * q_{i,ref}^{t+1} + \beta * \bar{s}_{DVM(i,rel)}^{t+1} - \chi * \bar{s}_{DVM(i,\overline{rel})}^{t+1},$$
$$q_{i,ref}^{t+3} = \alpha * q_{i,ref}^{t+2} + \beta * \bar{s}_{DVM(i,rel)}^{t+2} - \chi * \bar{s}_{DVM(i,\overline{rel})}^{t+2}, \text{ mit } \alpha, \beta, \chi \geq 0. \quad (496)$$

Wird mit einer Initialisierungsphase t=0 begonnen, so kann diese Vorgehensweise ab der zweiten Feedback-Iteration für jede weitere Iteration durchgeführt werden, d.h. für jede Iteration können drei eigene Parameter bestimmt werden, sodass sich eine Sequenz von Parameter-Tripeln (α^t, β^t, χ^t) ergibt:

$$((\alpha^{t=2}, \beta^{t=2}, \chi^{t=2}), (\alpha^{t=3}, \beta^{t=3}, \chi^{t=3}), ...). \quad (497)$$

Wird der Fall betrachtet, dass Q_i^{t+1} und eine Gesamt-Relevanzbewertung rel_{ref}^{t} durch den Agenten gegeben ist, so könnte die monoton steigende Distanz-Relevanzfunktion $rel_{ref}(d_{DVR}(.))$ bzw. deren Funktionsparameter remodelliert werden. Zunächst wird Q_i^{t+1} regulär indexiert, sodass q_i^{t+1} erzeugt wird. Die Distanz $d_{DVR}(q_i^t, q_i^{t+1})$ dient als Input in die Distanz-Relevanzfunktion, wobei der sich ergebende Relevanzwert einem Dokumentvektor, einer Dokumentvektorenmenge oder einem Repräsentanten zugeordnet wird. Der Gesamt-Relevanzwert wird der Dokumentvektorenmenge $DV(q_i^t)$ oder einem Repräsentanten wie $\bar{s}_{DVM(i)}^{t}$ zugeordnet. In dem vorliegenden Beispiel muss jedoch nicht entschieden werden, ob der Relevanzwert einem speziellen Vektor im DVR oder einer Menge von Vektoren zugeordnet wird. Gesucht wird somit eine Funktion $rel_{ref}(d_{DVR}(.))$, welche die folgende Bedingung erfüllt:

$$rel_{ref}^{t} = rel_{ref}(d_{DVR}(q_i^t, q_i^{t+1})). \quad (498)$$

Es muss eine Funktionsform vorgegeben werden, damit der oder die Parameter festgelegt werden können, sodass ein gegebener Input $d_{DVR}(q_i^t, q_i^{t+1})$ einen gegebenen Output erzeugt. Die einfachste Funktionsform ist eine lineare Funktion $y = \alpha * x$ mit genau einem Parameter α, sodass sich aus der allgemeinen Funktionsgleichung

$$rel_{ref}(d_{DVR}(.)) = \alpha * d_{DVR}(q_i^t, q_i^{t+1}) \quad (499)$$

die Form

$$rel_{ref}^{t} = \alpha * d_{DVR}(q_i^t, q_i^{t+1}) \quad (500)$$

ergibt, aus der α eindeutig bestimmt werden kann, da rel_{ref}^{t} und $d_{DVR}(q_i^t, q_i^{t+1})$ gegeben sind (siehe Abb. 102) mit $\alpha = 1/2$). Bei der allgemeinen linearen Funktion $y = \alpha * x + \beta$ ergibt sich durch die zwei freien Parameter eine Unterspezifizierung, die nur über den Verlauf der Interaktion aufgelöst werden kann, indem die Vektoren q_i^t, q_i^{t+1}, q_i^{t+2}, sowie die Werte rel_{ref}^{t} und rel_{ref}^{t+1} vorliegen. Bei einer Komposition aus zwei sich schneidenden Funktionen, wie z.B. einer Potenzfunktion und einer gebrochen linearen Funktion (siehe Abb. 103)), sind wesentlich mehr Parameter zu berücksichtigen:

$$y = \alpha * x^{\beta/\chi}; \beta, \chi \in N; \beta > \chi; \alpha = 2n-1, \beta = 2n;$$
$$y = \delta - \varepsilon/x - \phi/x^2; \varepsilon, \phi > 0. \quad (501)$$

Die Parameteranzahl kann reduziert werden durch $\alpha = \delta = 1$, sowie durch die Forderung, dass beide Funktionen einen Schnittpunkt besitzen müssen, doch es bleibt mehr als ein freier Parameter im Gegen-

satz zu der vereinfachten linearen Form, sodass diese Unterspezifizierung ebenfalls nur im Verlauf des Feedbackverfahrens beseitigt werden kann.

Wurde ein Gesamtrelevanzwert und Einzelrelevanzwerte $rel_{ref}(x_{j|q(i,t)})$ durch den Agenten spezifiziert, so ist es möglich, die Distanz-Gewichtungsfunktion $h(d_{DVR}(.))$ zu remodellieren, d.h. es kann nach einer Funktion gesucht werden, die gegebene Einzelwerte $rel_{ref}(x_{j|q(i,t)})$ bei einer gegebenen Gesamtrelevanz erzeugt. Ziel einer solchen Vorgehensweise ist eine Form von Kalibrierung zu Beginn der Interaktion mit einem Agenten, bei der eine Funktion festgelegt wird, die im weiteren verwendet wird. Dabei sollen die Einzelbewertungen durch die Gesamtbewertung ersetzt werden, d.h. der Agent soll nur noch eine Gesamtbewertung abgeben, und die Einzelbewertungen werden mit Hilfe der agentenspezifischen Funktion $h(d_{DVR}(., .))$ geschätzt. Dieses Ziel setzt voraus, dass die mentalen Bewertungen des Agenten, die durch die Funktion modelliert wird, im Verlauf des Feedback-Prozesses nicht wesentlich verändert. Kann davon nicht ausgegangen werden, so ist ein alternatives Ziel die Beobachtung der Veränderung der Funktion im Verlauf des Feedback-Prozesses. Mit den beobachteten Veränderungen könnten Rückschlüsse auf die mentalen Prozesse beim Problemlösen bzw. beim Reformulieren getroffen werden. Liegen entsprechende Sequenzen von Funktionen bzw. Funktionsparametern für mehrere Agenten vor, so können diese Daten analysiert werden, z.B. um durch eine Clusterung Gruppen von Agenten zu identifizieren, und um einen unbekannten Agenten später zu klassifizieren

Unabhängig von den Zielen ist die Festlegung eines Repräsentanten von $DV(q_i^t)$ wie $\bar{s}_{DVM(i)}^t$, dem die Gesamtrelevanz zugeordnet wird. Es ergibt sich somit eine Menge von Gleichungen, die näherungsweise erfüllt sein müssen, wobei die Kernelfunktion gesucht wird:

$$rel_{ref}(x_{j|q(i,t)}) = h(d_{DVR}(\bar{s}_{DVM(i)}^t, x_{j|q(i,t)})) * rel_{ref}(\bar{s}_{DVM(i)}^t), \forall x_{j|q(i,t)} \in DV(q_i^t). \quad (502)$$

Wird von einer konstanten Form der Kernelfunktion ausgegangen, so bedeutet dies die Suche nach geeigneten Funktionsparametern, wie z.B. die Verwendung einer Exp-Funktion bei den Kernelfunktionen bei Cohn (1995[73]) und Cohn et al. (1995[74])):

$$h(d_{DVR}(\bar{s}_{DVM(i)}^t, x)) = \exp(-k * (\bar{s}_{DVM(i)}^t - x)^2), \text{ bzw. allgemeiner}$$
$$h(d_{DVR}(\bar{s}_{DVM(i)}^t, x)) = \exp(-k * d_{DVR}(\bar{s}_{DVM(i)}^t, x)). \quad (503)$$

Ob es sich um eine Über- oder Unterspezifizierung handelt, ist abhängig von der Anzahl der Dokumentvektoren in $DV(q_i^t)$ sowie der Anzahl der Parameter der Kernelfunktion. In dem dargestellten Fall besitzt die Kernelfunktion genau einen Parameter, sodass für jeden Dokumentvektor $x_{j|q(i,t)}$ genau ein Parameter $k_{j|q(i,t)}$ bestimmt werden kann. Es kann nicht erwartet werden, dass die einzelnen Werte für unterschiedliche Dokumentvektoren übereinstimmen, sodass zumindest die Lage und Streuung der Parameterverteilung angegeben werden soll.

Eine solche Überspezifizierung kann Rückwirkungen auf das Verfahrensdesign besitzen, da es nicht notwendig ist, alle Dokumentvektoren aus $DV(q_i^t)$ durch den Agenten mit einer Relevanzbewertung zu versehen. Bei genau einem Kernelparameter würde genau eine Einzelbewertung genügen, wobei sich jedoch das Problem der Stimulusselektion ergibt. Denkbar wäre z.B. die Auswahl der beiden Dokumentvektoren mit der kleinsten und der größten Distanz zu dem Repräsentanten $\bar{s}_{DVM(i)}^t$. Es könnte auch eine inkre-

mentelle Form der Auswahl und Bewertung durchgeführt werden, die sich Prinzipien des Modellaufbaus durch aktives Lernen bedient, d.h. der Agent wird jeweils aufgefordert, ein Dokument aus der Restmenge zu bewerten, von dem das IRS erwartet, dass die Bewertung die Unsicherheit bzw. den Fehler des Modells, d.h. der gesuchten Funktion, minimieren wird.

Anstatt eine Kernel-Funktion festzulegen und die Parameter dieser Funktion zu spezifizieren, kann das Problem der Suche nach einer Distanz-Gewichtungsfunktion auch durch eine symbolische Regression mit Hilfe einer Evolutionären-Programm-Induktion wie dem Genetic-Programming gelöst werden (Koza (1992[188], 1994[189]), Koza et al. (1999[191]), Banzhaf et al. (1998[26]), Nordin (1997[238])). Dabei ist es nützlich, so viele der Einzelbewertungen wie möglich zu verwenden, wobei höchstens eine inkrementelle Form der Auswahl und Bewertung durch Aktives Lernen in Betracht gezogen werden kann, um die Anzahl der agentenbasierten Bewertungen zu verringern.

3.9.7.2) Reformulierung anderer Texttypen wie der Problembeschreibung

Der Reformulierungsansatz des IR bzw. allgemeiner des Problemlösungsprozesses legt zunächst nicht fest, was explizit betrachtet werden soll, wobei vereinbart wurde, dass nur explizit vorliegende, d.h. als Zeichensequenz vorliegende Strukturen betrachtet werden sollen. Die Reformulierung mentaler Strukturen in dem Problemlösungsareal bleibt somit unberücksichtigt, da sie höchstens durch Introspektion des Agenten und nicht von einer externen Instanz erfasst werden können.

Die Reformulierung der Query durch den Agenten, als explizit formuliertes, externes Informationsbedürfnis, ist ein Beispiel für die als Zeichensequenz vorliegenden Strukturen. Im Rahmen der Kritik query-basierter Retrievalansätze im Abschnitt 3.6) wurden andere Texttypen wie die Beschreibung des Gesamtproblems bzw. von Teilproblemen, Problemlösungsvorschläge, Assoziationslisten, Kommentare zu anderen Texten, annotierte Literaturlisten, Diskussionslisten und Brainstorming-Protokolle genannt, die vorliegen können, wenn der IR-Prozess als Einbettung in einen Problemlösungsprozess eventuell mit einer Gruppe von Agenten betrachtet wird. Für verschiedene Texttypen wie die Beschreibung des Problemes oder von Lösungsversuchen ist es sinnvoll zu fordern, dass sie im Verlauf einer längeren Interaktion mit dem IS ständig aktualisiert werden.

Liegen Texte dieser Art explizit als Zeichensequenzen vor, und sind sie dem IRS zugänglich, so sind sie indexierungsfähig. Es existieren zwei prinzipielle Szenarien der Nutzung alternativer Texttypen in Beziehung zu dem query-basierten Ansatz, wobei im weiteren der Texttyp „Problemlösungsvorschlag“ betrachtet werden soll:
1) Verwendung als Substitution der Query.
2) Verwendung als Ergänzung der Query.

Bei der Verwendung als Substitution wird die gesamte Interaktion zwischen dem Agenten und dem IS nicht auf Queries und deren Bewertung im Rahmen des Relevanz-Feedbacks aufgebaut, sondern auf der oder den anderen Texttypen. Insbesondere für die Problemlösungsvorschläge ergibt dies einen neuen semantischen Ansatz bei Beibehaltung der Verfahrenselemente, die bei den query-basierten Verfahren eingeführt wurden. D.h. der Agent präsentiert bei einer Monorepräsentation einen oder bei einer Polyre-

präsentation mehrere Problemlösungsvorschläge, die indexiert werden und durch Punkte im DVR repräsentiert werden, die als Problemlösungsvektoren im Gegensatz zu Queryvektoren bezeichnet werden können. Es folgt eine Retrievalphase, in der Dokumentvektoren ermittelt werden, die in der oder den Retrievalregionen der Problemlösungsvektoren liegen. Die zugehörigen Dokumente werden dem Agenten entsprechend einer bestimmten Rankingstrategie präsentiert, der diese analysiert, und somit seine mentalen Repräsentationen verändert. Je nach der betrachteten Gesamtstrategie reformuliert er seine Problemlösungsvorschläge oder bewertet die präsentierten Dokumente im Rahmen eines Relevanzfeedbacks entsprechend ihrem geschätzten Einfluss auf den Problemlösungsprozess oder den Reformulierungsprozess der Lösungsvorschläge.

Komplexere Veränderungen ergeben sich bei der Ergänzung der Query durch einen anderen Texttyp wie Problemlösungsvorschläge. Ergänzung kann dabei in zwei Kontexten gesehen werden:
1) Ergänzung im Sinne von unabhängiger Beigabe anderer Texte zu einer Query.
2) Ergänzung im Sinne der Komposition künstlicher Zeichensequenzen aus einer Original-Query und anderer Texte.

Liegt eine Query Q_i^t und ein Problemlösungsvorschlag L_i^t vor, so können diese beiden Zeichensequenzen mit der gemeinsamen Indexierungsfunktion A_{IR} zu q_i^t und l_i^t indexiert werden, was dem Fall der unabhängigen Beigabe entspricht. Die beiden Punkte im DVR bilden je das Zentrum einer Retrievalregion, in der Dokumentvektoren als Retrievalmenge ausgewählt werden, und die korrespondierenden Dokumente werden dem Agenten präsentiert.

Die Erzeugung künstlicher Zeichensequenzen z.B. durch ein stochastisches Verfahren wie dem Moving-Block-Bootstrap erzeugt zunächst eine gemeinsame Menge von Zeichenblocks aus den beiden Zeichensequenzen Q_i^t und L_i^t. Aus dieser Menge werden Elemente gleichverteilt zufällig mit Zurücklegen gezogen, und in der Reihenfolge ihrer Ziehung in einer Zeichensequenz angeordnet, bis eine Mindestlänge von Zeichen erreicht wurde. Jeder der δ auf diese Weise erzeugten Zeichensequenzen wird als Hybrid von Query und Lösungsvorschlag mit der gemeinsamen Indexierungsfunktion A_{IR} indexiert, und erzeugt somit einen Punkt im DVR, der das Zentrum einer Retrievalregion bildet, und mit denen Retrievalmengen von Dokumentvektoren spezifiziert werden.

3.9.8) Gleichzeitige Modifikation mehrerer Repräsentationen am Beispiel von Queryvektor- und Dokumentvektor-Feedback

Es können gemeinsame Modifikationen der unterschiedlichen Repräsentationen durchgeführt werden, für die oben ein einzelnes Relevanzfeedback dargestellt wurde (siehe Abschnitt 3.9)), d.h. Queryvektor., Dokumentvektoren, Indexierungsfunktion, Gewichtsvektor und Retrievalregion. Unberücksichtigt blieb dabei die Metrik im Dokumentvektorenraum und im Merkmalsvektorenraum. Um eine kombinatorische Explosion zu vermeiden, soll eine gemeinsame Modifikation von Queryvektor und Dokumentvektoren beschrieben werden.

Bei der Betrachtung eines gleichzeitigen Query- und Dokumentvektoren-Feedbacks sei eine Ursprungs-

Dokumentvektorenmenge $DV := DV^{t=0}$ und ein Ursprungs-Queryvektor $q_i^{t=0}$ gegeben, d.h. es wird von einer Monorepräsentation der Dokumentvektoren und des Queryvektors ausgegangen. Bei der ersten Iteration wird eine Ergebnis-Dokumentvektorenmenge $DV(q_i^{t=0}) := DV(q_i^{t=0} \mid DV^{t=0})$ spezifiziert, und die korrespondierenden Dokumente aus $D(q_i^{t=0})$ werden dem Agenten vorgelegt. Dieser erzeugt binäre Bewertungen, sodass die Ergebnismenge in $D(q_i^{t=0})_{rel}$ und $D(q_i^{t=0})_{\overline{rel}}$ zerlegt wird, was zu korrespondierenden Dokumentvektorenmengen $DV(q_i^{t=0})_{rel}$ und $DV(q_i^{t=0})_{\overline{rel}}$ führt. Für diese beiden Mengen werden Repräsentantenvektoren $\bar{s}_{DV(i,rel)}^{t=0}$ und $\bar{s}_{DV(i,\overline{rel})}^{t=0}$ gebildet. Diese beiden Vektoren werden als Fixpunkt der Adaption sowohl für die Adaption des Queryvektors als auch für die Adaption der Dokumentvektoren verwendet, wobei von zwei unabhängigen Operationen ausgegangen werden soll, d.h. beide Adaptionen beziehen sich auf $DV^{t=0}$, $q_i^{t=0}$, $\bar{s}_{DV(i,rel)}^{t=0}$ und $\bar{s}_{DV(i,\overline{rel})}^{t=0}$.

Die Queryvektor-Adaption wird durch die gemischte Feedbackstrategie durchgeführt zu:

$$q_i^{t=1} = \alpha * q_i^{t=0} + \beta * \bar{s}_{DV(i,rel)}^{t=0} - \chi * \bar{s}_{DV(i,\overline{rel})}^{t=0}, \text{ mit } \alpha, \beta, \chi \geq 0. \tag{504}$$

Bei der Dokumentvektoren-Adaption wird jeweils eine ε-Umgebung um die beiden Fixpunkte der Adaption erzeugt, d.h. es wird $U(\bar{s}_{DV(i,rel)}^{t=0} \mid \varepsilon^+)$ und $U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-)$ gebildet, vereinfachend mit $\varepsilon^+ := \varepsilon^-$. Für jede Umgebung werden die Dokumentvektoren aus der Gesamtmenge bestimmt, die innerhalb der Umgebung liegen:

$$\begin{aligned} DV(U(\bar{s}_{DV(i,rel)}^{t=0} \mid \varepsilon^+)) &= \{x_{ik+}^{t=0} \in DV \mid d_{DVR}(\bar{s}_{DV(i,rel)}^{t=0}, x_{ik+}^{t=0}) < \varepsilon^+\}, \\ DV(U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-)) &= \{x_{ik-}^{t=0} \in DV \mid d_{DVR}(\bar{s}_{DV(i,\overline{rel})}^{t=0}, x_{ik-}^{t=0}) < \varepsilon^-\}. \end{aligned} \tag{505}$$

Die Elemente der ersten Menge sollen einer positiven Adaption mit dem Fixpunkt $\bar{s}_{DV(i,rel)}^{t=0}$ unterzogen werden:

$$\begin{aligned} x_{ik+}^{t=0,neu} &= x_{ik+}^{t=0,alt} + \Delta x_{ik+}^{t=0,alt}, \text{ mit} \\ \Delta x_{ik+}^{t=0,alt} &= \varepsilon_{adaption} * d_{DVR}(x_{ik+}^{t=0,alt}, \bar{s}_{DV(i,rel)}^{t=0}), \\ &\forall x_{ik+}^{t=0} \in DV(U(\bar{s}_{DV(i,rel)}^{t=0} \mid \varepsilon^+)), \end{aligned} \tag{506}$$

und die Elemente der zweiten Menge werden einer negativen Adaption mit dem Fixpunkt $\bar{s}_{DV(i,\overline{rel})}^{t=0}$ unterzogen:

$$\begin{aligned} x_{ik-}^{t=0,neu} &= x_{ik-}^{t=0,alt} - \Delta x_{ik-}^{t=0,alt}, \text{ mit} \\ \Delta x_{ik-}^{t=0,alt} &= \varepsilon_{adaption} * d_{DVR}(x_{ik-}^{t=0,alt}, \bar{s}_{DV(i,\overline{rel})}^{t=0}), \\ &\forall x_{ik-}^{t=0} \in DV(U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^-)). \end{aligned} \tag{507}$$

Durch diese Adaptionsprozesse ergibt sich zum einen ein neuer Queryvektor $q_i^{t=1}$ und zum anderen eine neue Dokumentvektorenmenge $DV^{t=1}$, die als Ausgangspunkt für die nächste Feedback-Iteration dienen, d.h. um $q_i^{t=1}$ wird eine Retrievalregion mit dem gleichen Umgebungsparameter wie in t=0 erzeugt, und es wird aus $DV^{t=1}$ die Ergebnismenge $DV(q_i^{t=1}) := DV(q_i^{t=1} \mid DV^{t=1})$ spezifiziert. Korrekturen, die verhindern, dass Dokumente mehrmals präsentiert werden, können entweder unabhängig von diesen Prozessen duchgeführt werden, oder die zugrunde liegenden Dokumentvektorenmengen müssen bei jeder Iteration korrigiert werden. D.h. anstatt die Grundmenge $DV^{t=1}$ wieder als m'elementige Menge zu defi-

nieren, werden die nachgewiesenen Elemente der vorangegangenen Iteration bzw. der vorangegangenen Iterationen ausselektiert:

$$DV^{t=1} = DV^{t=0} \setminus DV(q_i^{t=0}) \text{ bzw.}$$
$$DV^{t} = DV^{t-1} \setminus DV(q_i^{t-1}),\ t > 0. \tag{508}$$

Aus Effizienzgründen wäre es auch sinnvoll, die Grundmenge für die Ermittlung der positiven und negativen Adaptionsmenge anzupassen:

$$DV(U(\bar{s}_{DV(i,rel)}^{t} \mid \varepsilon^{+})) = \{x_{ik+}^{t} \in DV^{t} = DV^{t-1} \setminus DV(q_i^{t-1}) \mid d_{DVR}(\bar{s}_{DV(i,rel)}^{t}, x_{ik+}^{t}) < \varepsilon^{+}\},$$
$$DV(U(\bar{s}_{DV(i,rel)}^{t} \mid \varepsilon^{-})) = \{x_{ik-}^{t} \in DV^{t} = DV^{t-1} \setminus DV(q_i^{t-1}) \mid d_{DVR}(\bar{s}_{DV(i,rel)}^{t}, x_{ik-}^{t}) < \varepsilon^{-}\}. \tag{509}$$

Wird eine solche Korrektur der Grundmenge der Dokumentvektoren durchgeführt, so ergibt sich zwanglos ein Abbruchkriterium des Feedback-Prozesses, wenn in einer Retrievalregion keine neuen Dokumentvektoren ermittelt werden können:

$$DV(q_i^{t}) := DV(q_i^{t} \mid DV^{t} = DV^{t-1} \setminus DV(q_i^{t-1})) = \varnothing. \tag{510}$$

4) Relevanz-Approximation in Mono- und Polyrepräsentations-IRS

4.1) Approximationsmodelle mit reellen Relevanzwerten

4.1.1) Binäre und reelle Relevanzwerte

Bei den Relevanz-Feedback-Verfahren aus dem Abschnitt 3.9) wurde der Standardfall einer binären Bewertung durch den Agenten Ag_i verwendet, d.h. es wird eine Relevanzfunktion vewendet, die jedem Dokumentvektor x_{ij}^t aus der Ergebnismenge $DVM(q_i^t)$ einen Wert aus {0, 1} zuordnet:

$$rel(x_{ij}^t) := rel(x_{ij}^t \mid Ag_i) \in \{0, 1\}, \forall x_{ij}^t \in DVM(q_i^t), \text{ mit z.B.}$$
$$DVM(q_i^t) = \{x_{ij}^t \in DV^t \mid d_{DVR}(q_i^t, x_{ij}^t) < \varepsilon, j = 1, ..., f^{t=0}\}. \quad (511)$$

Dies ist ein Spezialfall einer Klassifikation, bei der zwei Klassen C_0 und C_1 verwendet werden, die mit 0 bzw. 1 kodiert werden. Diese binäre Klassifikation kann als rudimentärste Form eines Rankings betrachtet werden, wenn anstatt einer Menge $\{C_0, C_1\}$ eine geordnete Liste wie (C_1, C_0) verwendet wird. Alle Dokumentvektoren mit einem Relevanzwert von 1 gehören der Klasse C_1 an und bilden den ersten Rang des Rankings, während Dokumentvektoren mit dem Relevanzwert 0 der Klasse C_0 angehören, welche den zweiten Platz des Rankings bildet.

Mit dem Konzept der geordneten Liste können nun nicht binäre Klassifikationen betrachtet werden, bei denen mehr als zwei Klassen C_k, $k = 1, ..., m_C > 2$, verwendet werden, die geordnet sind, und somit eine Form des Rankings bilden:

$$rel(x_{ij}^t) := rel(x_{ij}^t \mid Ag_i) \in \{1, 2, ..., m_C\}, \forall x_{ij}^t \in DVM(q_i^t), \text{ mit}$$
$$K_C = (C_k \mid k = 1, ..., m_C). \quad (512)$$

Das Ranking auf der Basis einer geordneten Liste von Klassen ergibt sich durch die Rangordnung, wobei ein kleinerer Relevanzwert als besser bewertet wird. Es ist zu berücksichtigen, dass ein Platz des Rankings mit einer leeren Menge belegt sein kann, d.h. kein Dokument aus $DVM(q_i^t)$ wird in diese Klasse eingeordnet. Weiterhin können in einer Klasse mehrere Dokumentvektoren aus $DVM(q_i^t)$ liegen, wodurch mehrere Dokumentvektoren auf einem Platz des Rankings liegen.

Der Wertebereich der Relevanzfunktion lässt sich von einer abzählbaren Liste weiter verallgemeinern, indem ein Intervall verwendet wird:

$$rel(x_{ij}^t) := rel(x_{ij}^t \mid Ag_i) \in [0, 1], \forall x_{ij}^t \in DVM(q_i^t). \quad (513)$$

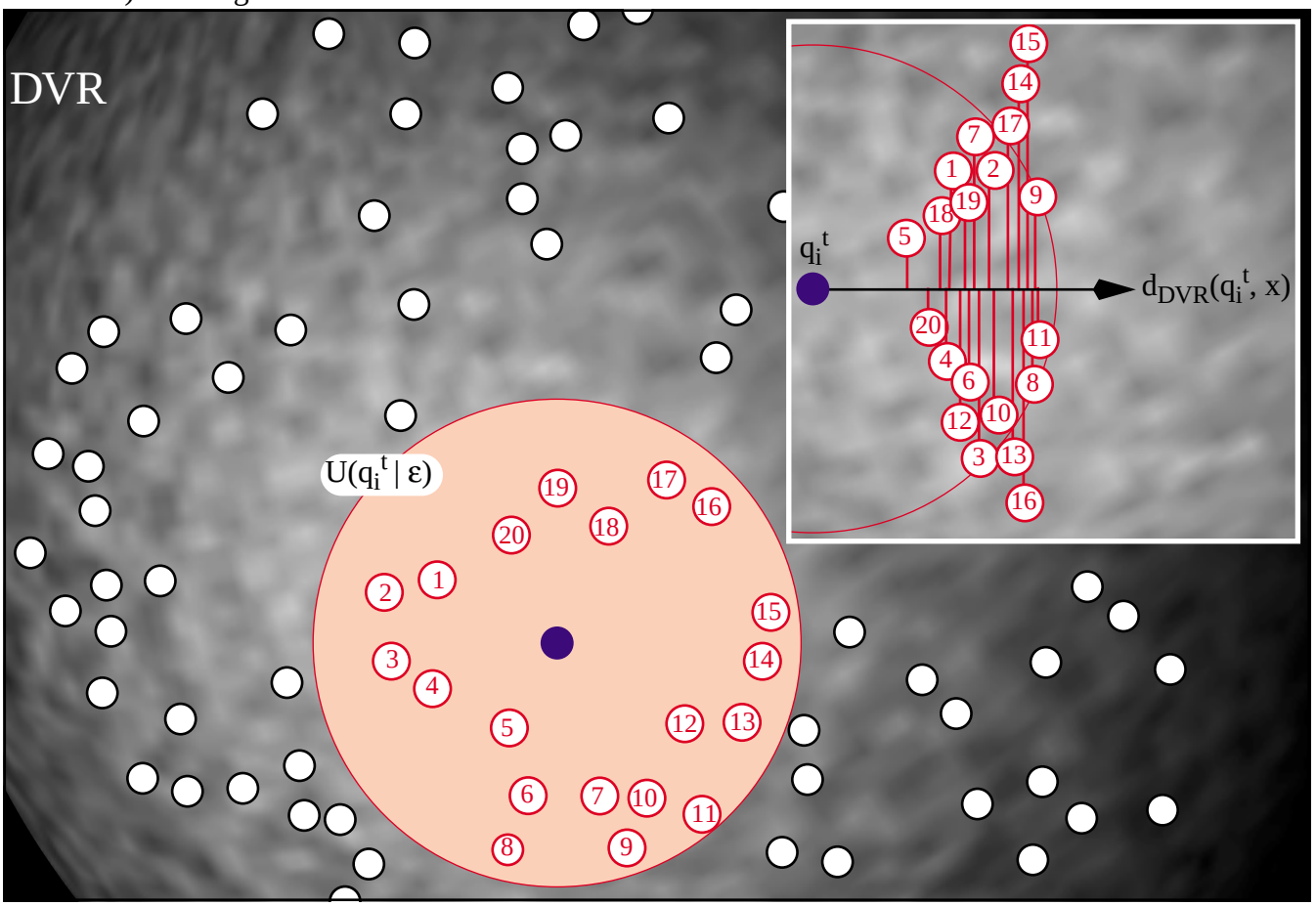
Die Verwendung des Intervalls [0, 1] kann im Kontext der Dichatomie „nicht relevant vs. relevant“ verwendet werden, während im Abschnitt 3.9.1.5) die Dichatomie „irreführend vs. relevant“ mit einem Mittel „nicht relevant“ diskutiert wurde. In diesem Fall wurde vorgeschlagen „irreführend“ mit -1, „nicht relevant“ mit 0 und relevant mit 1 zu kodieren. Soll dies mit einer diskontinuierlichen Variable geschehen, so folgt daraus die Verwendung der Menge {-1, 0, 1}, während bei einer kontinuierlichen Variablen die Verwendung des Intervalls [-1, 1] folgt:

$$rel(x_{ij}^t) := rel(x_{ij}^t \mid Ag_i) \in \{-1, 0, 1\}, \text{ bzw.}$$
$$rel(x_{ij}^t) := rel(x_{ij}^t \mid Ag_i) \in [-1, 1], \forall x_{ij}^t \in DVM(q_i^t). \quad (514)$$

4.1.2) Ranking und Distanz-Relevanzfunktion

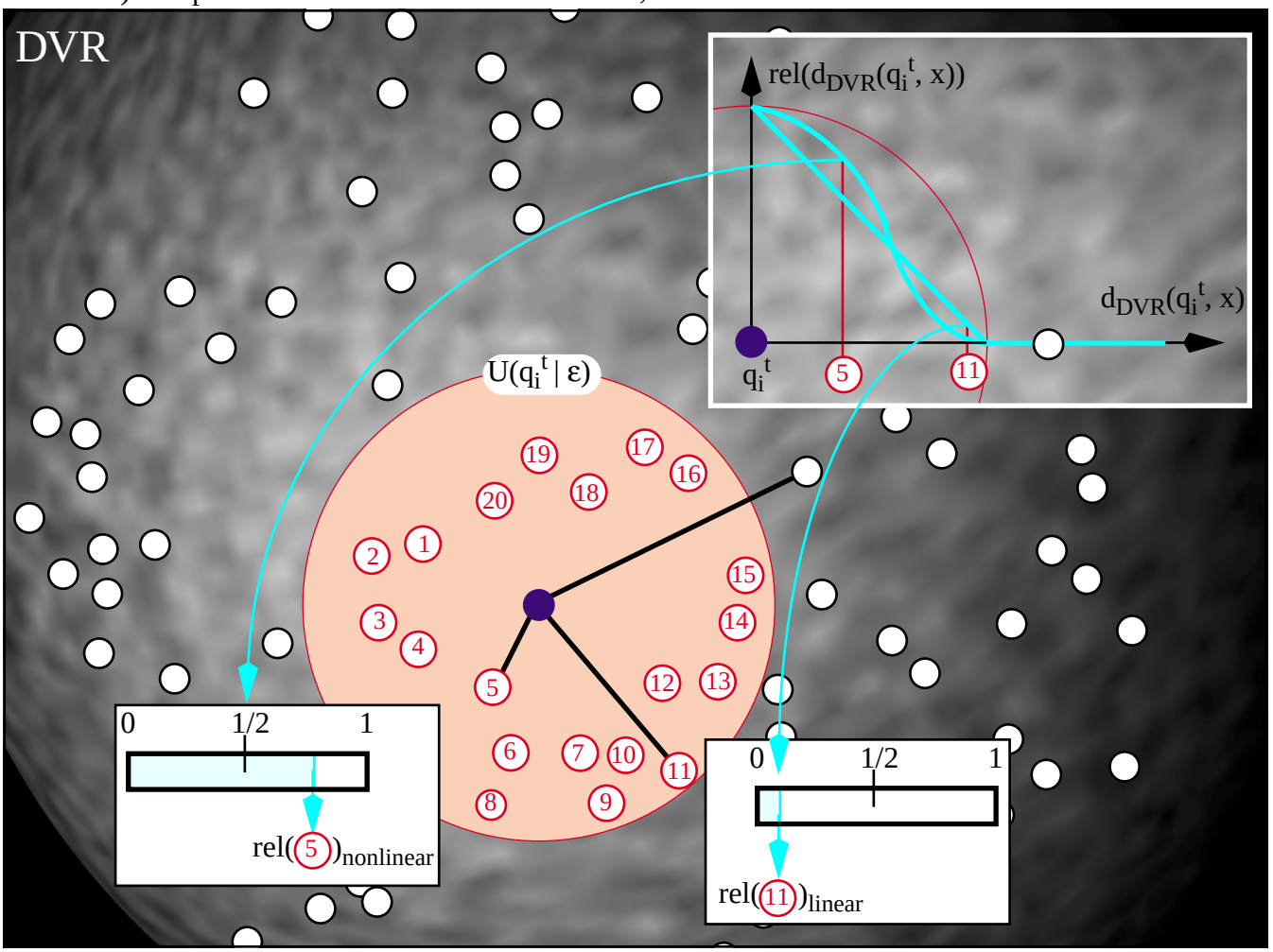
Im Rahmen des Rankings beim Retrieval in vektorraumbasierten IRS finden nur implizit Schätzungen der Relevanz in Abhängigkeit von der Distanz zwischen Dokumentvektor und Queryvektor q_i statt, wobei eine kleinere Distanz eine größere Relevanzschätzung bedeutet (siehe Abb. 106)). In dem dargestellten Beispiel wird eine Ergebnismenge $DVM(q_i^t) = \{x_j \mid j = 1, ..., 20\}$ auf eine geordnete Liste $DV(q_i^t) = (x_5, x_{20}, x_{18}, x_4, x_1, x_{12}, x_{19}, x_6, x_7, x_3, x_2, x_{10}, x_{17}, x_{13}, x_{14}, x_{16}, x_{15}, x_8, x_9, x_{11})$ abgebildet.

Abb. 106) Ranking durch Distanzmaß



Zunächst kann vermutet werden, dass durch das Ranking als Rangfolge eine Ordinal- oder Rangskala erzeugt wird, doch die Distanzen implizieren eine metrische Skala. Durch die Existenz eines natürlichen Nullpunktes $d_{DVR}(q_i^t, x) = 0$ wird damit zumindest eine Verhältnisskala impliziert. Für die Bildung eines solchen Rankings durch Distanzen werden ausschließlich Stützpunkte im Dokumentvektorraum DVR verwendet, da keine Distanz-Relevanz-Funktion festgelegt wurde. Mit einer Distanz-Relevanz-Funktion rel(d(q, x)) (siehe Abb. 107)) wird ein eindimensionaler Outputraum festgelegt, der zusammen mit dem DVR eine Input-Output-Kombination (x, rel(d(q, x))) erzeugt, mit der überwachtes Lernen möglich wird, anstatt unüberwachtem Lernen, wenn nur Stützpunkte aus DVR vorliegen. Sinnvoll sind monoton fallende Distanz-Relevanz-Funktionen, wobei der Fall dargestellt wird, dass ab einer Distanz größer dem Umgebungsparameter ε ein Relevanzwert von Null verwendet wird (siehe Abb. 107)).

Abb. 107) Beispiel einer linearen und nicht-linearen, unimodalen Distanz-Relevanz-Funktion



Eine solche Distanz-Relevanz-Funktion kann als Radial-Basis-Funktion (siehe z.B. Poggio & Girosi (1989[260])) verstanden werden, da sie in einem Punkt, dem Queryvektor, genau ein Maximum besitzt, und symmetrisch um dieses Maximum streng monoton fällt, bis ein Distanz-Schwellenwert erreicht ist, nach dem sie auf der Abszissenachse verläuft. Sie besitzt auch eine direkte Beziehung zu den Kernel-Funktionen, die im Rahmen der stützpunktbasierten LWR-Approximationsmodelle zur Relevanzschätzung eingesetzt werden sollen (siehe Abschnitt 2.1.2.1)). Eine LWR-Kernelfunktion wird als monoton fallende Gewichtungsfunktion modelliert, die als Input zwei Punkte x_k, x_j aus einem Inputraum bzw. die Distanz $d(x_k, x_j)$ dieser beiden Punkte aufnimmt, und einen Gewichtungswert $h(d(x_k, x_j))$ als skalaren Output liefert. Im Gegensatz zu den beiden Distanz-Relevanz-Funktionen, die in Abb. 107) dargestellt werden, besitzen die LWR-Kernel-Funktionen meist eine asympthotische Annäherung an die Distanzachse, wie z.B. eine Gauss-Verteilung.

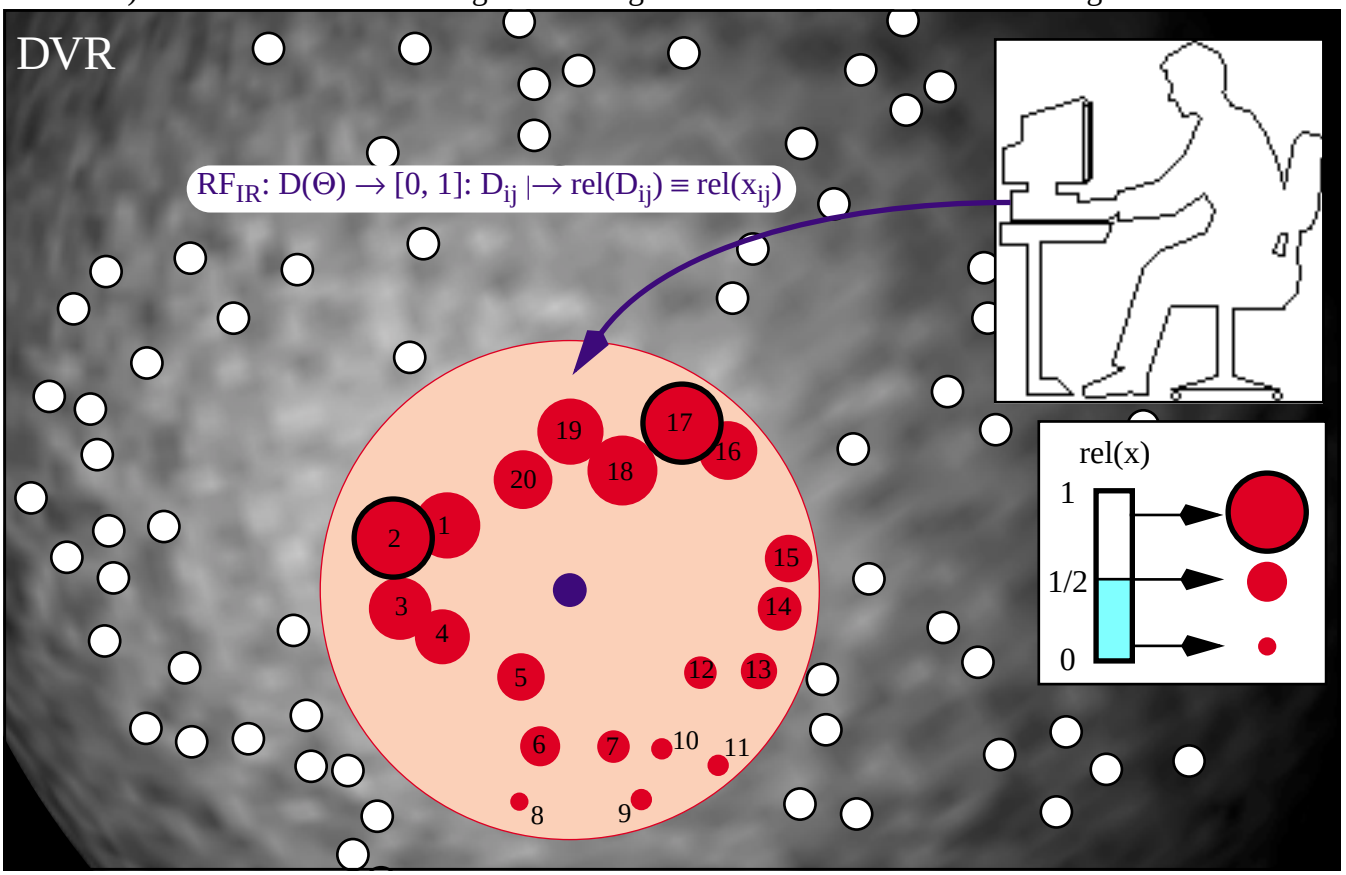
Wichtig für die Eigenschaft einer solche Distanz-Relevanz-Funktion ist, dass sie unimodal ist, d.h. dass sie genau ein Maximum besitzt. Damit werden die Einschränkungen bezüglich der Anwendbarkeit ersichtlich, da ein multimodaler Relevanzzusammenhang durch eine solchen Funktionstyp nicht adäquat modellierbar bzw. lernbar ist.

Ein multimodaler Relevanzzusammenhang ergibt sich immer dann, wenn ein Agent Dokumente als sehr relevant bewertet, deren Dokumentvektoren nicht in unmittelbarer Nachbarschaft im DVR liegen, sodass

mehrere Maxima einer Relevanzfunktion vorliegen. Diese Situation muss als Standard betrachtet werden, da eine Unimodalität interpretiert werden kann als Übereinstimmung der Repräsentationen des IRS mit den Repräsentationen des Agenten bezüglich eines Problems oder Ziels des Agenten. Selbst wenn dies in einem Einzelfall zutreffen würde, muss davon ausgegangen werden, dass die Diversität der Agenten (siehe Abschnitt 1.2.5)) manifestiert ist in einer Vielzahl unterschiedlicher, individueller Repräsentationen. Unterschiedliche Agenten mit unterschiedlichen Repräsentationen besitzen somit bezüglich einer konstanten Repräsentation in einem IRS multimodale Relevanzfunktionen. Die Einbettung des IR in einen komplexeren Problemlösungsprozess bedingt zudem die Zerlegung eines Problems in eine Struktur von Teilprobleme, wobei Dokumente bezüglich dieser Teilprobleme unterschiedlich bewertet werden können. Ein externes Informationsbedürfnis eines Agenten, welches sich auf eine Anzahl von Teilproblemen bezieht, impliziert somit bereits eine multimodale Relevanzstruktur.

Am Relevanz-Feedback und der Relevanzschätzung mit einem LWR-Approximationsmodell soll dies im Kontext der Multimodalität erläutert werden. Am Anfang steht die Bewertung von nachgewiesenen Dokumenten aus $D(q_i) = D_i$ durch reelle Relevanzwerte $rel(D_{ij})$ mit der Relevanz-Feedback-Funktion RF_{IR}, wobei die Relevanzwerte $rel(x_{ij})$ für die Dokumentvektoren direkt übernommen werden (siehe Abb. 108)).

Abb. 108) Reelle Relevanzbewertung einer nachgewiesenen Dokumentvektorenmenge

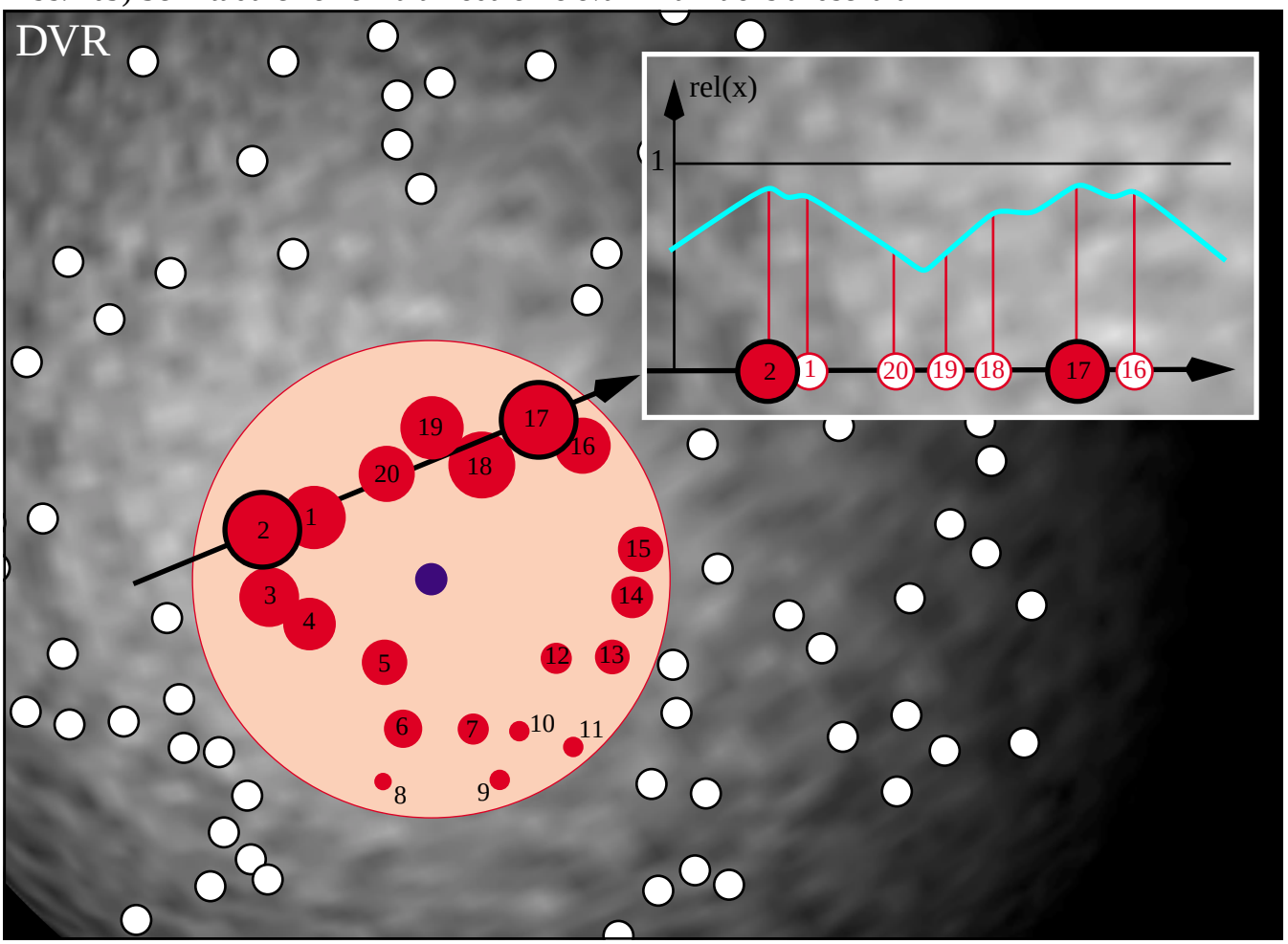


Es wird der Fall dargestellt, dass zwei Relevanz-Maxima in den Dokumentvektoren x_{i2} und x_{i17} vorliegen, sodass dadurch eine multimodale Relevanzfunktion erzeugt wird. D.h. das Relevanz-Approximationsmodell ist eine stetige Relevanzfunktion, die als Überlagerung von 20 unimodalen, stetigen Kernelfunktionen erzeugt wird, und zwei globale Maxima und mehrere lokale Maxima besitzt, je nach

der Art der verwendeten Kernelfunktionen. In Abb. 109) wird ein Schnitt durch eine multimodale Relevanz-Funktionslandschaft dargestellt, wobei auf der Schnittgeraden im Dokumentvektorenraum die beiden Maxima x_{i2} und x_{i17} liegen. Die lokalen Maxima werden wesentlich durch den Einfluss der Stützpunkte x_{i1}, x_{i17}, x_{i20}, x_{i19}, x_{i18}, x_{i16} erzeugt, die in der Nähe der Geraden liegen.

Jedem Punkt im DVR, dem kein Relevanzwert durch den Agenten zugeordnet wurde, kann jeweils eine Relevanzschätzung durch das verwendete stützpunktbasierte Approximationsmodell zugeordnet werden. Ein explizites, stützpunktbasiertes Approximationsmodell (siehe Abschnitt 2.1.2.1)) der Relevanzzusammenhänge verwendet die Elemente x_{ij} aus DVM_i als Stützpunkte im Dokumentvektorenraum und die Relevanzwertungen $rel(x_{ij})$ als Stützpunkte im eindimensionalen Relevanzvektorenraum, wodurch nichtlineare, nicht-monotone und insbesondere multimodale Zusammenhänge auf einem metrischen Skalenniveau modellierbar werden. Wird jedem Stützpunkt eine unimodale Kernelfunktion zugeordnet, so bedeutet dies, dass maximal so viele Maxima modelliert werden können, wie Stützpunkte vorliegen, während die Anzahl lokaler Maxima unbestimmt ist, und von der konkreten Überlagerung der Kernelfunktionen abhängt.

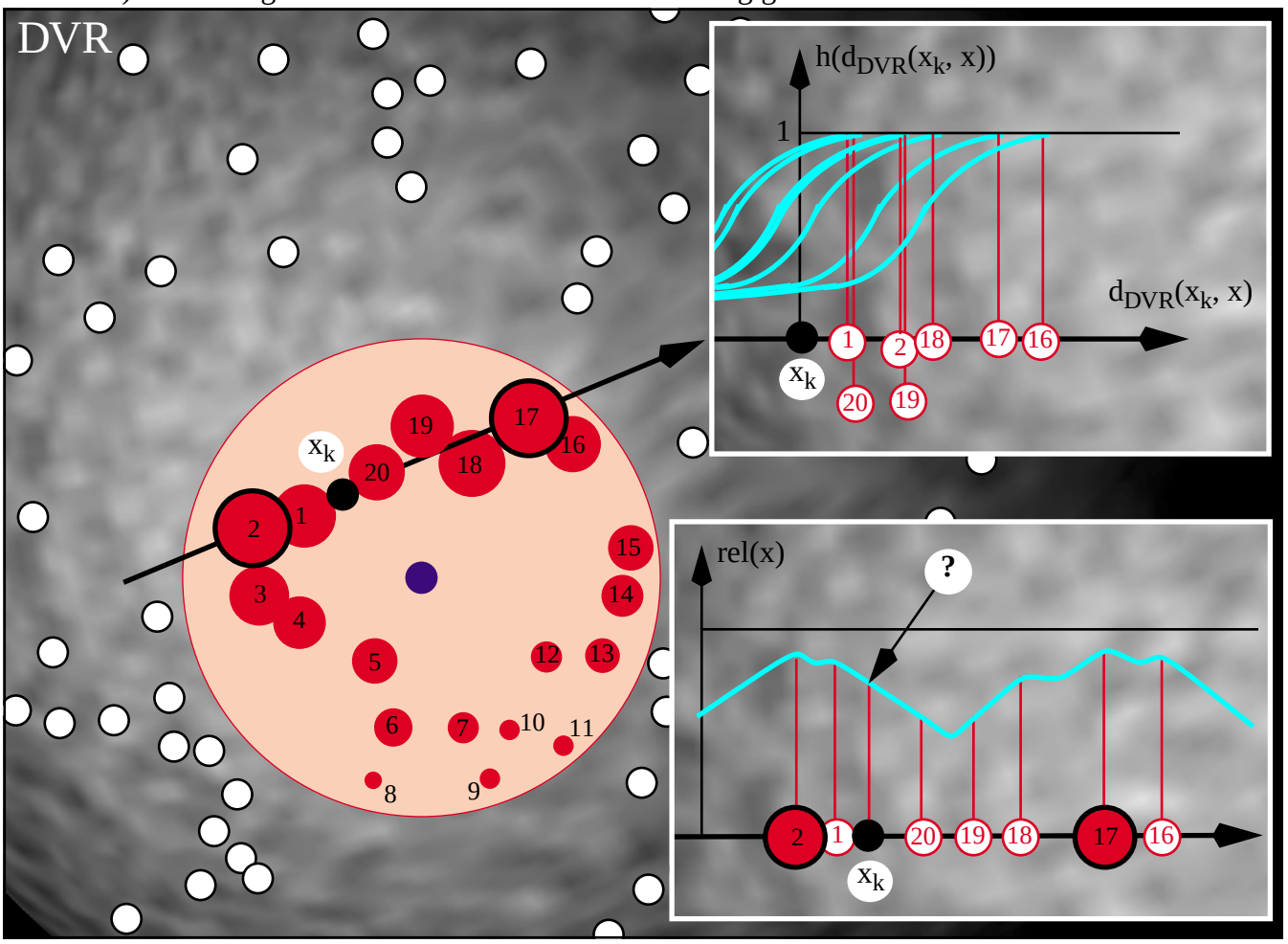
Abb. 109) Schnitt durch eine multimodale Relevanz-Funktionslandschaft



Eine Relevanzschätzung an einem beliebigen Punkt x_k im DVR, zu dem kein Dokument korrespondieren muss, soll für den Fall eines globalen, instanzbasierten Approximationsmodells AM(x) betrachtet werden, d.h. alle vorliegenden Stützpunkte tragen mit ihrem Relevanzwert und einer Gewichtung mit zur

Schätzung bei. Zur besseren Übersicht sollen jedoch nur eine Teilmenge von Stützpunkten betrachtet werden, wobei im ersten Schritt die Distanzen $d_{DVR}(x_k, x_{ij})$ zu x_k berechnet werden (siehe Abb. 110)). Danach wird für jeden Stützpunkt der Wert der Kernelfunktion $h(d_{DVR}(x_k, x_{ij}))$ als Funktion der Distanz $d_{DVR}(x_k, x_{ij})$ berechnet, z.B. indem die Euklidische Metrik in Verbindung mit einem gausschen Kernel verwendet wird, d.h. $h(d_{DVR}(x_k, x_{ij})) = \exp(-\alpha\ (x_k - x_{ij})^2)$ (siehe Cohn (1994[71]), Cohn et al. (1995[74])).

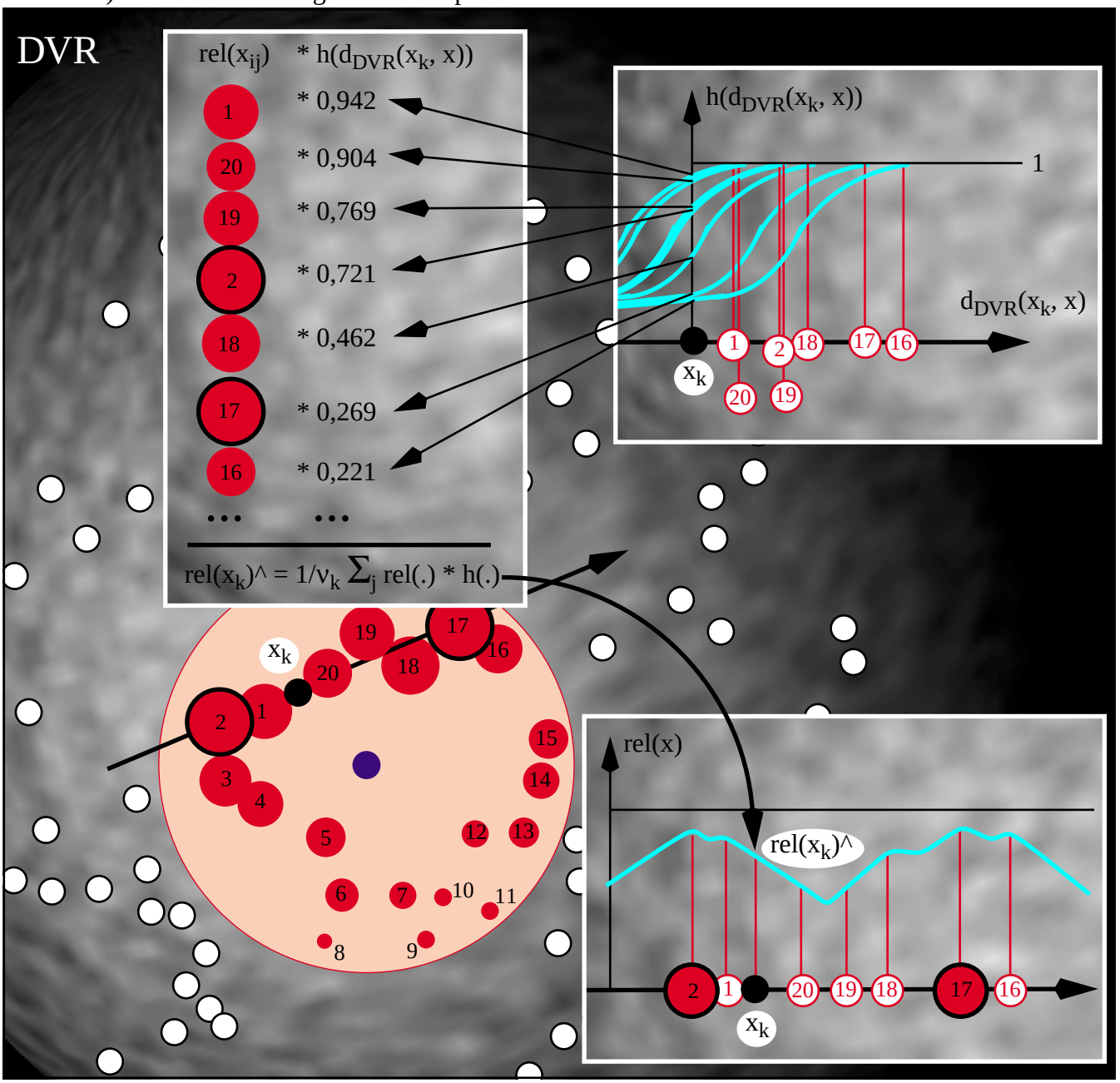
Abb. 110) Zuordnung von Kernelfunktionswerten in Abhängigkeit von Distanzen



Die nächsten Schritte bestehen in der Bildung der Gewichtung der Relevanzwerte $rel(x_{ij})$ mit den Kernelfunktionswerten $h(d_{DVR}(x_k, x_{ij}))$, die Summation der gewichteten Relevanzwerte, sowie die Normierung mit der Summe ν_k aller Gewichte (siehe Abb. 111)):

$$rel(x_k \mid AM(x \mid DVM_i)) = rel(x_k)^{\wedge} = 1/\nu_k \sum_j h(d_{DVR}(x_k, x_{ij})) * rel(x_{ij}), \text{ mit}$$
$$\nu_k = \sum_j h(d_{DVR}(x_k, x_{ij})),\ \forall x_{ij} \in DVM_i. \qquad (515)$$

Abb. 111) Relevanzschätzung durch Stützpunkt-Relevanzwerte und Kernelfunktionswerte



4.1.3) Relevanz-Klassifikations- und Approximationsmodelle

Im Rahmen von Relevanz-Klassifikationsmodellen wird versucht, aus einer endlichen Menge von Stützpunkten, die aus einem Tupel von $x \in$ DVR und einer Relevanzbewertung $rel(x) \in \{1, 2, ..., m_C\}$ bestehen, eine Schätzfunktion $rel(x)\hat{}$ abzuleiten, welche in der Lage ist, konsistente Schätzwerte, d.h. Klassenzugehörigkeiten, für beliebige Punkte aus dem DVR zu erzeugen. Die binäre Relevanzbewertung $rel(x) \in \{0, 1\}$ ist ein Spezialfall des Klassifikationsmodells mit genau zwei Klassen.

Im Kontext einer Relevanz-Feedback-Iteration lässt sich dies formulieren, dass ausgehend von einer Lernmenge M_L mit

$$M_L = \{m_{Lij} = (x_{ij}^t, rel(x_{ij}^t)) \mid rel(x_{ij}^t) \in \{1, 2, ..., m_C\}, \forall x_{ij}^t \in DVM(q_i^t)\} \quad (516)$$

ein Klassifikationsmodell gesucht wird, das allen vorliegenden Dokumentvektoren bzw. jedem Punkt aus dem DVR eine Relevanzschätzung zuordnen kann:

$$\begin{aligned} &rel(x)^{\wedge}: DVR \rightarrow \{1, ..., m_C\}: x_j^t \mid\rightarrow rel(x_j^t)^{\wedge}, \forall x_j^t \in DV^t, \text{ bzw.} \\ &rel(x)^{\wedge}: DVR \rightarrow \{1, ..., m_C\}: x \mid\rightarrow rel(x)^{\wedge}, \forall x \in DVR. \end{aligned} \tag{517}$$

Wie in Abschnitt 2.2) dargestellt wurde, kann prinzipiell zwischen einem Regressions- und einem Interpolationsmodell unterschieden werden, wobei diese Unterscheidung dort für kontinuierliche Variablen getroffen wurde. Dies lässt sich einfach für den diskontinuierlichen Fall eines Klassifikationsmodells reformulieren, indem angenommen wird, dass bei einem Regressionsmodell die Relevanzschätzung $rel(x_{ij}^t)^{\wedge}$ eines Dokumentvektors aus der Ergebnismenge $DVM(q_i^t)$) nicht notwendig exakt gleich dem Lernwert $rel(x_{ij}^t)$ sein muss:

$$rel(x)_{reg}^{\wedge}: DVR \rightarrow \{1, ..., m_C\}: x_{ij}^t \mid\rightarrow rel(x_{ij}^t)_{reg}^{\wedge} \neq rel(x_{ij}^t), \forall x_{ij}^t \in DVM(q_i^t)). \tag{518}$$

Bei einem Interpolationsmodell wird jedoch eine exakte Übereinstimmung für alle Lern-Stützpunkte gefordert:

$$rel(x)_{interpol}^{\wedge}: DVR \rightarrow \{1, ..., m_C\}: x_{ij}^t \mid\rightarrow rel(x_{ij}^t)_{interpol}^{\wedge} = rel(x_{ij}^t), \forall x_{ij}^t \in DVM(q_i^t)). \tag{519}$$

Analog zu den Relevanz-Klassifikationsmodellen, wird im Rahmen von Relevanz-Approximationsmodellen versucht, aus einer endlichen Menge von Stützpunkten, die aus einem Tupel von $x \in DVR$ und einer Relevanzbewertung $rel(x) \in [0, 1]$ bestehen, eine Schätzfunktion $rel(x)^{\wedge}$ abzuleiten, die konsistente Schätzwerte für beliebige Punkte aus dem DVR erzeugen kann. Allgemeiner kann dies durch die Einführung eines Relevanzraumes RelR formuliert werden, der als Outputraum im Gegensatz zum Inputraum DVR betrachtet wird. Innerhalb dieser Arbeit wird von einem ein-dimensionalen Relevanzraum ausgegangen, der sich auf ein Intervall wie [0, 1] oder [-1, 1] beschränkt. Ausnahmen könnten betrachtet werden, wenn z.B. verschiedene Typen von Relevanz verwendet werden, oder wenn ein Relevanztyp durch verschiedene Methoden ermittelt wird, sodass in diesen Fällen ein mehr-dimensionaler Relevanzraum aufgebaut wird. Dies gilt auch dann, wenn neben einem Relevanzwert ein oder mehrere Fehlermaße betrachtet werden, wobei in diesem Fall der mehr-dimensionale Outputraum nicht mehr als Relevanzraum bezeichnet werden kann, da die Relevanzbewertung bzw. -bewertungen nur einen Teil der Dimensionen des Outputraumes ausmachen.

Im Kontext einer Relevanz-Feedback-Iteration lässt sich ein Relevanz-Approximationsmodell formulieren, dass ausgehend von einer Lernmenge M_L mit

$$M_L = \{m_{Lij} = (x_{ij}^t, rel(x_{ij}^t)) \mid rel(x_{ij}^t) \in [0, 1], \forall x_{ij}^t \in DVM(q_i^t)\} \tag{520}$$

ein Approximationsmodell gesucht wird, das allen vorliegenden Dokumentvektoren bzw. jedem Punkt aus dem DVR eine Relevanzschätzung zuordnen kann:

$$\begin{aligned} &rel(x)^{\wedge}: DVR \rightarrow RVR := [0, 1]: x_j^t \mid\rightarrow rel(x_j^t)^{\wedge}, \forall x_j^t \in DV^t, \text{ bzw.} \\ &rel(x)^{\wedge}: DVR \rightarrow RVR := [0, 1]: x \mid\rightarrow rel(x)^{\wedge}, \forall x \in DVR. \end{aligned} \tag{521}$$

Bei einem Regressionsmodell muss die Relevanzschätzung $rel(x_{ij}^t)^\wedge$ eines Dokumentvektors aus der Ergebnismenge $DVM(q_i^t)$) nicht notwendig exakt gleich dem Lernwert $rel(x_{ij}^t)$ sein:

$$rel(x)_{reg}^\wedge: DVR \rightarrow RelR := [0, 1]: x_{ij}^t \mapsto rel(x_{ij}^t)_{reg}^\wedge \neq rel(x_{ij}^t), \forall x_{ij}^t \in DVM(q_i^t)). \quad (522)$$

Bei einem Interpolationsmodell wird jedoch eine exakte Übereinstimmung für alle Lern-Stützpunkte gefordert:

$$rel(x)_{interpol}^\wedge: DVR \rightarrow RVR := [0, 1]: x_{ij}^t \mapsto rel(x_{ij}^t)_{interpol}^\wedge = rel(x_{ij}^t), \forall x_{ij}^t \in DVM(q_i^t)). \quad (523)$$

Im Rahmen dieser Arbeit sollen ausschließlich Regressionsmodelle verwendet werden, da Interpolation besondere Anforderungen an die Qualität der Messung stellt, d.h. die Outputwerte müssen sehr exakt ermittelt werden, während Regressionsverfahren Outputwerte mit größerem Fehler bzw. größeren Unsicherheiten tolerieren können, da die Regressionsfunktion Fehler glättet. Diese Eigenschaft kommt der Grundsituation der Relevanzbewertung in IRS entgegen, da die Relevanzbewertungen mehreren Formen von Unsicherheit unterliegen. Es handelt sich bei den Relevanzwerten $rel(x_{ij}^t)$ faktisch schon um eine Relevanzschätzung durch den Agenten, da er zu dem Zeitpunkt, zu dem er die Relevanz bewerten muss, weder den Inhalt eines Dokumentes genau kennt, wenn ein Retrieval mit Zeit-Constraints unterstellt wird, noch die abschließende Bedeutung des Dokumentes für den gesamten Problemlösungsprozess überblicken kann (siehe auch Abschnitt 3.9.1)).

Standardmäßig werden im weiteren Local-Weighted-Regression-Verfahren betrachtet, die entweder auf den Lernstützpunkten selbst (instanzbasierte LWR; siehe Abschnitt 2.1.3)), oder auf Repräsentanten-Stützpunkten basieren, wobei diese als Gewichtsvektoren eines GNG-SOM-Verfahrens erzeugt werden (LWR-GNG-SOM; siehe Abschnitt 2.1.7)).

4.2) Feedback mit reellen Relevanzbewertungen bei unklassifizierten Dokumentvektoren

Alle in diesem Kapitel dargestellten Modelle basieren auf Relevanzbewertungen von Dokumentvektoren durch den Agenten oder durch Agentengruppen, sodass ein Relevanz-Feedback-Verfahren bzw. mindestens eine Iteration in einem Feedback-Verfahren vorausgesetzt wird. Die Relevanzbewertungen des Agenten werden als reell angenommen, wobei standardmäßig das Einheitsintervall [0, 1] verwendet werden soll.

4.2.1) Queryvektor-Adaption bei reellen Relevanzbewertungen

Werden reelle Relevanzbewertungen verwendet, so muss die Adaptions-Operation angepasst werden, da nicht mehr direkt eine Menge von relevanten und nicht-relevanten Dokumentvektoren vorliegt. Die Darstellungen sollen von einer unklassifizierten Dokumentvektorenmenge ausgehen. Beschrieben werden soll unter diesen Bedingungen die Queryvektor-Adaption bei Queryvektor-Mono- und -Polyrepräsentation sowie das Vorliegen positiver und negativer Queryvektoren.

4.2.1.1) Adaption bei Queryvektor-Monorepräsentation

Am Anfang steht wie üblich eine Query $Q_i^{t=0}$, die von einem Agenten formuliert wird, und durch die Indexierungsfunktion $A_{IR(Q)}$ auf den Queryvektor $q_i^{t=0}$ abgebildet wird. Die erste Retrieval-Iteration verläuft analog zu dem normalen Queryvektor-Feedback, indem bei einer unklassifizierten Dokumentvektorenmenge um $q_i^{t=0}$ eine ε-Umgebung $U(q_i^{t=0} \mid \varepsilon)$ gebildet wird, die eine Dokumentvektoren-Ergebnismenge $DVM(q_i^{t=0})$ liefert. Die zugehörige Dokumentmenge $DM(q_i^{t=0})$ wird dem Agenten präsentiert, bzw. es wird zunächst ein Ranking durchgeführt, indem die Menge $DVM(q_i^{t=0})$ in eine geordnete Liste $DV(q_i^{t=0})$ umgewandelt wird. Dazu werden die Dokumentvektoren entsprechend ihrer steigenden Distanz zu $q_i^{t=0}$ geordnet, da zu diesem Zeitpunkt noch kein anderes Ordnungskriterium vorliegt, das in den nachfolgenden Iterationen durch Relevanz-Approximationsmodelle geliefert wird.

Unabhängig ob eine Menge oder eine Liste verwendet werden, der Agent bewertet die präsentierten Dokumente durch einen Wert aus dem Intervall [0, 1], wobei standardmäßig angenommen werden soll, dass alle Dokumente aus einer Ergebnismenge $DVM(q_i^t)$ bewertet werden, d.h. es sollen keine „missing values“ betrachtet werden. Die Dokumentbewertung wird für den korrespondierenden Dokumentvektor übernommen, sodass Lernstimuli $m_j^{t=0}$ gebildet werden können, welche einen Dokumentvektor $x_j^{t=0}$ aus $DVM(q_i^{t=0})$ als Inputkomponente und den zugehörigen Relevanzwert $rel(x_j^{t=0})$ als Outputkomponente besitzen. Diese Stützpunkte werden in einer Stimulusmenge $M_{DV(m)}^{t=0}$ zusammengefasst:

$$M_{DV(m)}^{t=0} = \{m_j^{t=0} = (x_j^{t=0}, rel(x_j^{t=0})) \mid rel(x_j^{t=0}) \in [0, 1], \forall x_j^{t=0} \in DVM(q_i^{t=0})\}. \quad (524)$$

Der offensichtliche Unterschied zwischen einem Relevanz-Feedback mit einer binären Wertemenge und einem Verfahren mit einem Intervall als Wertemenge, besteht darin, dass Dokumentvektoren nicht mehr direkt als „nicht relevant“ betrachtet werden können, sodass aus $DVM(q_i^{t=0})$ nicht direkt eine Zerlegung in $DVM(q_i^{t=0})_{rel}$ und $DVM(q_i^{t=0})_{\overline{rel}}$ durchgeführt werden kann. Soll der Ablauf eines Queryvektor-Relevanz-Feedbacks soweit wie möglich erhalten bleiben, so muss eine der beiden folgenden Verfahrensklassen angewendet werden:

1) Verfahren mit $DVM(q_i^{t=0})$-Zerlegung durch einen Relevanz-Schwellenwert S_{rel}

Die Gesamtergebnismenge $DVM(q_i^{t=0})$ wird durch den Schwellenwert S_{rel} in eine Menge mit relevanten und nicht relevanten Dokumentvektoren binär zerlegt:

$$DVM(q_i^{t=0})_{rel} = \{x_{j,rel}^{t=0} \mid rel(x_{j,rel}^{t=0}) \geq S_{rel}, j = 1, ..., f_{rel}^{t=0}\},$$
$$DVM(q_i^{t=0})_{\overline{rel}} = \{x_{j,\overline{rel}}^{t=0} \mid rel(x_{j,\overline{rel}}^{t=0}) < S_{rel}, j = 1, ..., f_{\overline{rel}}^{t=0}\}, S_{rel} \in]0, 1[. \quad (525)$$

Im weiteren soll von einem iterationsunabhängigen Schwellenwert ausgegangen werden, im Gegensatz zu einem iterationsabhängigen Schwellenwert S_{rel}^t, der als Funktion der Relevanzverteilung der nachgewiesenen Dokumentvektoren einer Iteration gebildet wird.

1.1) Einmalige Adaption durch künstliche Adaptions-Fixpunkte

Wird eine einmalige Adaptions-Operation unterstellt, so wird für die Mengen je ein künstlicher Adaptions-Fixpunkt erzeugt, z.B. durch die arithmetischen Zentroidvektoren:

$$\bar{s}_{DVM(i,rel)}^{t=0} = 1/f_{rel}^{t=0} * \sum_j x_{j,rel}^{t=0},\ \forall x_{j,rel}^{t=0} \in DVM(q_i^{t=0})_{rel},$$
$$\bar{s}_{DVM(i,\overline{rel})}^{t=0} = 1/f_{\overline{rel}}^{t=0} * \sum_j x_{j,\overline{rel}}^{t=0},\ \forall x_{j,\overline{rel}}^{t=0} \in DVM(q_i^{t=0})_{\overline{rel}}. \quad (526)$$

In diesem Fall kann eine einmalige, gemischte Adaptionsstrategie oder eine einmalige GNG-SOM-Adaption als Kombination einer positiven und einer negativen Adaption durchgeführt werden:

$$q_i^{t=1} = \alpha * q_i^{t=0} + \beta * \bar{s}_{DVM(i,rel)}^{t=0} - \chi * \bar{s}_{DVM(i,\overline{rel})}^{t=0}, \text{ mit } \alpha, \beta, \chi \geq 0, \text{ oder}$$
$$q_i^{t=1} = q_i^{t=0} + \Delta_{rel} q_i^{t=0} - \Delta_{\overline{rel}} q_i^{t=0}, \text{ mit}$$
$$\Delta_{rel} q_i^{t=0} = \varepsilon_{rel} * d_{DVR}(q_i^{t=0}, \bar{s}_{DVM(i,rel)}^{t=0}), \text{ und } \Delta_{\overline{rel}} q_i^{t=0} = \varepsilon_{\overline{rel}} * d_{DVR}(q_i^{t=0}, \bar{s}_{DVM(i,\overline{rel})}^{t=0}). \quad (527)$$

1.2) Mehrmalige Adaption durch natürliche Adaptions-Fixpunkte

Bei der mehrmaligen Adaption wird wie bei einem SOM-Aufbau eine Sequenz von Einzel-Adaptionen durchgeführt, die jeweils ein Element aus $DVM(q_i^{t=0})_{rel}$ bzw. $DVM(q_i^{t=0})_{\overline{rel}}$ als natürlichen Adaptions-Fixpunkt verwendet. D.h. es wird aus der Gesamtmenge $DVM(q_i^{t=0})$ gleichverteilt zufällig mit Zurücklegen ein Element gezogen, dessen Zugehörigkeit zu $DVM(q_i^{t=0})_{rel}$ oder $DVM(q_i^{t=0})_{\overline{rel}}$ festlegt, ob eine positive bzw. negative Adaption durchgeführt wird. Bei $x_{j,rel}^{t=0} \in DVM(q_i^{t=0})_{rel}$ erfolgt die positive Adaption indem $q_i^{t=0}$ in Richtung $x_{j,rel}^{t=0}$ um $\Delta q_i^{t=0}$ verschoben wird:

$$q_i^{t=0}(neu) = q_i^{t=0}(alt) + \Delta q_i^{t=0}(alt), \text{ mit}$$
$$\Delta q_i^{t=0}(alt) = \varepsilon_{adapt,rel} * d_{DVR}(q_i^{t=0}(alt), x_{j,rel}^{t=0}). \quad (528)$$

Bei $x_{j,\overline{rel}}^{t=0} \in DVM(q_i^{t=0})_{\overline{rel}}$ erfolgt die negative Adaption indem $q_i^{t=0}$ weg von $x_{j,\overline{rel}}^{t=0}$ um $\Delta q_i^{t=0}$ verschoben wird, wobei vereinfachend angenommen wird, dass beide Adaptionsparameter gleich sind, d.h. $\varepsilon_{adapt,rel} = \varepsilon_{adapt,\overline{rel}}$:

$$q_i^{t=0}(neu) = q_i^{t=0}(alt) - \Delta q_i^{t=0}(alt), \text{ mit}$$
$$\Delta q_i^{t=0}(alt) = \varepsilon_{adapt,\overline{rel}} * d_{DVR}(q_i^{t=0}(alt), x_{j,\overline{rel}}^{t=0}). \quad (529)$$

2) Verfahren ohne $DVM(q_i^{t=0})$-Zerlegung

Wird auf die $DVM(q_i^{t=0})$-Zerlegung verzichtet, so müssen alle Elemente aus $DVM(q_i^{t=0})$ in Verbindung mit einer positiven Adaptionsoperation verwendet werden, was schon bei einem binären Relevanz-Feedback als ausschließlich positive Strategie durchgeführt werden kann, bei der nur die relevanten Dokumentvektoren zur Adaption des Queryvektors verwendet werden. In diesem Kontext können bei reellen Relevanzwerten alle nachgewiesenen Dokumentvektoren als relevant betrachtet werden, wobei Gewichtungsfaktoren o.ä. als Funktion des Relevanzwertes benutzt werden müssen.

2.1) Verwendung individueller Gewichtungsfaktoren

Die einzelnen Dokumentvektoren werden mit einem Gewichtungsfaktor β_j versehen, der die jeweilige Relevanz des Dokumentvektors wiederspiegelt, d.h. Vektoren mit einer großen Relevanz erhalten einen größeren Einfluss auf die Adaption als Vektoren mit einem geringen Relevanzwert.

2.1.1) Einmalige Adaption mit individuellen Gewichtungsfaktoren

Durch die Verwendung aller Dokumentvektoren muss eine positive Gesamtstrategie angewendet werden, sodass eine prinzipielle Adaptionsgleichung verwendet wird:

$$q_i^{t=1} = \alpha * q_i^{t=0} + \sum_j \beta_j^{t=0} * x_j^{t=0}, \text{ mit } \alpha, \beta_j^{t=0} \geq 0, \forall x_j^{t=0} \in DVM(q_i^{t=0}). \quad (530)$$

2.1.2) Mehrmalige Adaption mit individuellen Gewichtungsfaktoren

Bei der mehrmaligen Adaption werden aus $DVM(q_i^{t=0})$ Elemente $x_j^{t=0}$ gleichverteilt zufällig mit Zurücklegen gezogen, für die jeweils eine Einzel-Adaption durchgeführt wird:

$$q_i^{t=0}(neu) = \alpha * q_i^{t=0}(alt) + \beta_j^{t=0} * x_j^{t=0}. \quad (531)$$

Nachdem eine bestimmte Anzahl von Einzel-Adaptionen durchgeführt wurde, wird der sich ergebende Queryvektor als $q_i^{t=1}$ verwendet.

2.2) Verwendung individueller Adaptionsparameter

Bei dem Adaptionsparameter ε_{adapt} handelt es sich um eine Analogie im Rahmen SOM-Adaption zu dem Gewichtungsfaktor β im Rahmen einer einmaligen Gesamtadaption, sodass einmalige wie mehrmalige SOM-Adaptionen mit individuellen Adaptionsparametern $\varepsilon_{adapt,j}$ möglich sind. Die Parameter $\varepsilon_{adapt,j}$ können wiederum als monoton wachsende Funktion der Relevanzbewertung $rel(x_j^{t=0})$ bestimmt werden.

2.2.1) Einmalige Adaption mit individuellen Adaptionsparametern

Für jedes Element $x_j^{t=0}$ aus $DVM(q_i^{t=0})$ wird unabhängig ein Deltavektor $\Delta_j q_i^{t=0}$ gebildet, die zusammen in einer einzigen Adaptionsgleichung aggregiert werden:

$$q_i^{t=1} = q_i^{t=0} + \sum_j \Delta_j q_i^{t=0}, \text{ mit}$$
$$\Delta_j q_i^{t=0} = \varepsilon_{adapt,j} * d_{DVR}(q_i^{t=0}(alt), x_j^{t=0}). \quad (532)$$

2.2.2) Mehrmalige Adaption mit individuellen Adaptionsparametern

Aus $DVM(q_i^{t=0})$ werden Elemente $x_j^{t=0}$ gleichverteilt zufällig mit Zurücklegen gezogen, für die jeweils eine Einzel-Adaption durchgeführt wird:

$$q_i^{t=0}(neu) = q_i^{t=0}(alt) + \Delta_j q_i^{t=0}, \text{ mit}$$
$$\Delta_j q_i^{t=0} = \varepsilon_{adapt,j} * d_{DVR}(q_i^{t=0}(alt), x_j^{t=0}). \quad (533)$$

2.3) Verwendung individueller Ziehungswahrscheinlichkeiten

Bei den mehrmaligen Adaptionen wurde bislang von gleichverteilt zufälligen Ziehungen mit Zurücklegen ausgegangen. Ein individueller Gewichtungsfaktor β_j oder Adaptionsparameter $\varepsilon_{adapt,j}$ kann substituiert werden durch eine individuelle Ziehungswahrscheinlichkeit, die als monoton wachsende Funktion der Relevanzwerte definiert ist, d.h. Dokumentvektoren mit einer hohen Relevanzbewertung werden häufiger für eine Adaption gezogen als Dokumentvektoren mit einer kleineren Relevanz, sodass die relevanten Vektoren einen größeren Einfluss auf die Adaption besitzen. Nach der Ziehung findet eine positive SOM-Adaption mit einem nicht-individuellen Adaptionsparameter ε_{adapt} bzw. eine Adaption mit einem nicht individuellen Gewichtungsfaktor β statt.

4.2.1.2) Adaption bei Queryvektor-Polyrepräsentation

Bei einer Queryvektor-Polyrepräsentation sind im Prinzip alle Adaptionsverfahren analog anwendbar, die bei der Queryvektor-Monorepräsentation vorgestellt wurden, wobei die Adaptionsprozesse sich auf eine unstrukturierte oder eine strukturierte Queryvektorenmenge QVM_i^t beziehen kann. Im weiteren soll eine Strukturierung durch eine Delaunay-Triangulation in Verbindung mit einer Sequenz von GNG-SOM-Adaptionen dargestellt werden (siehe auch Abschnitt 3.9.2.2) und 3.9.2.6)). D.h. in der Initialisierungs-Iteration t=0 wird durch eine GNG-SOM eine Queryvektor-Delaunay-Triangulation bzw. -Teil-Triangulation $N_q^{t=0} := N_{iq}^{t=0}$ erzeugt, die im weiteren Verlauf des Feedbackverfahrens durch positive und eventuell negative Adaptionsprozesse verändert wird, bis nach einem Abbruch die vorliegenden Knoten des Queryvektorengraphes als $QVM_i^{t=1}$ verwendet werden.

Es soll der Fall analog zu der mehrmalige Adaption durch natürliche Adaptions-Fixpunkte im vorangegangenen Abschnitt im Kontext einer Queryvektor-Polyrepräsentation mit $N_q^{t=0}$ dargestellt werden, wobei GNG-SOM-Adaptionsgleichungen verwendet werden sollen. Hierzu ist die Festlegung eines Schwellenwertes S_{rel} notwendig, der $DVM(q_i^{t=0})$ in $DVM(q_i^{t=0})_{rel}$ bzw. $DVM(q_i^{t=0})_{\overline{rel}}$ zerlegt. Aus der Gesamt-Ergebnismenge wird gleichverteilt zufällig ein Element $x_j^{t=0}$ mit Zurücklegen gezogen, zu dem das Gewinner-Neuron $n_{s(1|j)}^{t=0}$ aus dem Queryvektorgraph $N_q^{t=0}$ sowie dessen unmittelbaren Nachbarn $N(d_G=1 \mid G_q^{t=0})_{s(1|j)}$ ermittelt wird. Das Neuron $n_{s(1|j)}^{t=0}$ wird einer GNG-SOM-Gewinner-Adaption mit dem Parameter $\varepsilon_{adapt,s}$ und die Neurone aus der Nachbarmenge einer GNG-SOM-Nachbar-Adaption mit dem Parameter $\varepsilon_{adapt,n}$ unterzogen. Da der Deltavektor unabhängig von einer positiven bzw. negativen Adaption festgelegt werden kann, ergibt sich für den Gewinner-Queryvektor $q_{s(1|j)}^{t=0}$ und die Nachbar-Queryvektoren $q_{s(1|j)k}^{t=0}$:

$$\Delta q_{s(1|j)}^{t=0}(alt) = \varepsilon_{adapt,s} * d_{DVR}(q_{s(1|j)}^{t=0}(alt), x_{j,rel}^{t=0}),$$
$$\Delta q_{s(1|j)k}^{t=0}(alt) = \varepsilon_{adapt,n} * d_{DVR}(q_{s(1|j)k}^{t=0}(alt), x_j^{t=0}), \forall n_{s(1|j)k}^{t=0} \in N(d_G=1 \mid G_q^{t=0})_{s(1|j)}. \quad (534)$$

Ob eine konkrete Adaption positiv oder negativ ist, ist davon abhängig, ob $x_j^{t=0}$ aus der Menge $DVM(q_i^{t=0})_{rel}$ oder aus der Menge $DVM(q_i^{t=0})_{\overline{rel}}$ stammt. Im ersten Fall wird eine positive und im zweiten Fall eine negative Adaption durchgeführt:

$$x_j^{t=0} \in DVM(q_i^{t=0})_{rel}:$$
$$q_{s(1|j)}^{t=0}(neu) = q_{s(1|j)}^{t=0}(alt) + \Delta q_{s(1|j)}^{t=0} \wedge q_{s(1|j)k}^{t=0}(neu) = q_{s(1|j)k}^{t=0}(alt) + \Delta q_{s(1|j)k}^{t=0}. \quad (535)$$

$$x_j^{t=0} \in DVM(q_i^{t=0})_{\overline{rel}}:$$
$$q_{s(1|j)}^{t=0}(neu) = q_{s(1|j)}^{t=0}(alt) - \Delta q_{s(1|j)}^{t=0} \wedge q_{s(1|j)k}^{t=0}(neu) = q_{s(1|j)k}^{t=0}(alt) - \Delta q_{s(1|j)k}^{t=0}. \quad (536)$$

Nachdem eine bestimmte Anzahl von Ziehungen und Adaptionen durchgeführt wurde, die als Funktion der Anzahl der Dokumentvektoren in $DVM(QVM_i^{t=0})$ bestimmt wird, wird die sich ergebende Queryvektorverteilung als Ausgangspunkt für die nächste Iteration t=1 verwendet, wobei die Menge der Queryvektoren als $QVM_i^{t=1}$ bezeichnet wird. Es folgt eine neue Retrieval-Operation, bis der Agent oder das IRS einen Abbruch erzeugt, was im zweiten Fall erfolgt, wenn die Ergebnis-Dokumentvektorenmenge in einer Iteration zum ersten Mal leer wird.

4.2.1.3) Adaption bei positiven und negativen Queryvektoren

Beim Vorliegen positiver und negativer Queryvektoren in QVM_i^{+t} und QVM_i^{-t} soll die Darstellung bei der Queryvektor-Polyrepräsentation im vorangegangenen Abschnitt weitergeführt werden, indem gefordert wird, dass in der Initialisierungs-Iteration eine Queryvektor-Delaunay-Triangulation aller Queryvektoren erzeugt wird, unabhängig ob es sich bei den einzelnen Knoten um positive oder negative Queryvektoren handelt (siehe auch Abschnitt 3.9.2.8)). Bei der mehrfachen GNG-SOM-Adaption ist die Art der Adaption nun vom Typ des gezogenen Dokumentvektors und vom Typ des Queryvektors abhängig. Bei einem relevanten Dokumentvektor werden positive Queryvektoren positiv adaptiert und negative Queryvektoren werden negativ adaptiert, während bei einem nicht relevanten Dokumentvektor positive Queryvektoren negativ und negative Queryvektoren positiv adaptiert werden. Zu unterscheiden ist zwischen dem jeweiligen Gewinner-Neuron und seinen Nachbarn, wobei der Gewinner-Queryvektor $q_{s(1|j)}^{+t=0}$ bzw. $q_{s(1|j)}^{-t=0}$ mit einem größeren Adaptionsparameter $\varepsilon_{adapt,s}$ adaptiert wird als die Nachbar-Queryvektoren $q_{s(1|j)k}^{+t=0}$ bzw. $q_{s(1|j)k}^{-t=0}$ mit $\varepsilon_{adapt,n}$, was jedoch keinen Einfluss darauf hat, ob eine positive oder negative Adaptions-Operation durchgeführt wird.

$$x_j^{t=0} \in DVM(q_i^{t=0})_{rel}:$$

$$[q_{s(1|j)}^{+t=0}(neu) = q_{s(1|j)}^{+t=0}(alt) + \Delta q_{s(1|j)}^{+t=0} \vee q_{s(1|j)}^{-t=0}(neu) = q_{s(1|j)}^{-t=0}(alt) - \Delta q_{s(1|j)}^{-t=0}]$$

$$\wedge [q_{s(1|j)k}^{+t=0}(neu) = q_{s(1|j)k}^{+t=0}(alt) + \Delta q_{s(1|j)k}^{+t=0} \vee q_{s(1|j)k}^{-t=0}(neu) = q_{s(1|j)k}^{-t=0}(alt) - \Delta q_{s(1|j)k}^{-t=0}]. \quad (537)$$

$$x_j^{t=0} \in DVM(q_i^{t=0})_{\overline{rel}}:$$

$$[q_{s(1|j)}^{+t=0}(neu) = q_{s(1|j)}^{+t=0}(alt) - \Delta q_{s(1|j)}^{+t=0} \vee q_{s(1|j)}^{-t=0}(neu) = q_{s(1|j)}^{-t=0}(alt) + \Delta q_{s(1|j)}^{-t=0}]$$

$$\wedge [q_{s(1|j)k}^{+t=0}(neu) = q_{s(1|j)k}^{+t=0}(alt) - \Delta q_{s(1|j)k}^{+t=0} \vee q_{s(1|j)k}^{-t=0}(neu) = q_{s(1|j)k}^{-t=0}(alt) + \Delta q_{s(1|j)k}^{-t=0}]. \quad (538)$$

4.2.2) Feedback mit Relevanz-Approximationsmodell ohne Veränderung des Queryvektors

Das Ziel eines Relevanz-Feedbacks ist es, in jeder neuen Iteration neue Dokumentvektoren nachzuweisen, von denen das IRS annimmt, dass deren Relevanzbewertung durch den Agenten positiv ausfällt. D.h. das IRS soll eine Relevanzschätzung durchführen, und die Dokumentvektoren mit den höchsten Schätzungen, die bislang noch nicht nachgewiesen wurden, werden ausgewählt. Dieser prinzipiellen Zielvorstellung soll mit dem Queryvektor-Relevanz-Feedback entsprochen werden, obwohl bei diesem Verfahren keinerlei explizite Relevanzschätzungen durch das IRS durchgeführt werden. Das Plausibilitätsargument beruht darauf, dass ein sukzessiv besserer Queryvektor aus den Relevanzbewertungen des Agenten konstruiert wird, und dass die Distanzen im DVR als implizite Relevanzschätzungen verwendet werden.

Mit einem expliziten Relevanz-Approximationsmodell muss dieses Plausibilitätsargument als Hilfskonstruktion nicht mehr verwendet werden, sondern man kann direkt Relevanschätzungen an den Stellen im DVR durchführen, an denen sich die Dokumentvektoren befinden. Bei einer solchen Vorgehensweise eines Relevanz-Feedbacks muss der Queryvektor nicht mehr adaptiert werden, d.h. er spielt nach der ersten Iteration keine bzw. nicht mehr die entscheidende Rolle mehr, da alle nachfolgenden Retrieval-

Operationen nicht mehr bzw. nicht mehr ausschließlich auf der Basis von Mannigfaltigkeiten im DVR durchgeführt werden müssen.

Im weiteren soll die Situation dargestellt werden, dass in der Initialisierungs-Iteration t=0 eines Feedback-Verfahrens mit Hilfe von $q_i^{t=0}$ eine Ergebnis-Dokumentvektorenmenge $DVM(q_i^{t=0})$ erzeugt wurde, zu der Relevanzbewertungen durch den Agenten erfragt wurde, sodass sich die Stimulusmenge $M_{DV(m)}^{t=0}$ ergibt, die für ein instanzbasiertes LWR-Approximationsmodell verwendet wird. D.h. die Stimuli aus $M_{DV(m)}^{t=0}$ bilden die Stützpunkte im n-dimensionalen Inputraum DVR und im ein-dimensionalen Relevanzraum RelR := [0, 1]. Zusammen mit einem Regressionsverfahren wie der Kernel-Regression, bilden diese Stützpunkte am Ende der Initialisierungs-Iteration t=0 ein Relevanz-Approximationsmodell, das vereinfachend mit $AM(rel(x))^{t=0} := AM(rel(x) \mid M_{DV(m)}^{t=0})$ bezeichnet werden soll.

In der nachfolgenden Iteration t=1 werden Dokumentvektoren x_k aus der Restmenge der Gesamt-Dokumentvektorenmenge DV und der Ergebnismenge $DVM(q_i^{t=0})$ durch das instanzbasierte Approximationsmodell bewertet, d.h. ihnen wird eine Relevanzschätzung zugeordnet:

$$rel(x_k)^\wedge := rel(x_k \mid AM(rel(x))^{t=0}) = 1/\nu_k \sum_j h(d_{DVR}(x_k, x_j^{t=0})) * rel(x_j^{t=0}), \text{ mit}$$
$$\nu_k = \sum_j h(d_{DVR}(x_k, x_j^{t=0})), \forall m_j^{t=0} \in M_{DV(m)}^{t=0}, x_k \in DV \setminus DVM(q_i^{t=0}). \quad (539)$$

Mit der Fähigkeit, nach der Initialisierungs-Iteration Relevanzschätzungen dieser Art durchzuführen, können neue Retrievalstrategien für die weiteren Iterationen erzeugt werden. Die einfachste, aber auch aufwendigste, Strategie besteht darin, alle restlichen Dokumentvektoren zu bewerten, die Vektoren mit der besten Relevanzschätzung auszuwählen und nach fallender Relevanzschätzung geordnet als Ergebnisliste der nächsten Iteration zu verwenden.

Bei den Retrievalstrategien, die auf Distanzen im DVR basieren, spielt die Position des Queryvektors die entscheidende Rolle, sodass Ergebnismengen in Abhängigkeit von dem Queryvektor angegeben werden, wie z.B. $DVM(q_i^{t=0})$. Liegt nach der Initialisierungs-Iteration ein Relevanz-Approximationsmodell vor, so ergeben sich die Ergebnismengen in Abhängigkeit von dem jeweils verwendeten Approximationsmodell, sodass eine Bezeichnung wie $DVM(AM(rel(x))^{t=0})^{t=1}$ verwendet werden kann. Diese Bezeichnung gibt an, dass eine Ergebnismenge bzw. - liste in der Iteration t=1 durch das Approximationsmodell aus der Iteration t=0 erzeugt wurde. Die Angabe zweier Iterationsbezeichungen ist dann notwendig, wenn das Approximationsmodell nach jeder Iteration aktualisiert wird, indem die Relevanzbewertungen durch den Agenten in Form von Stützpunkten bzw. Lernstimuli einbezogen werden. Strategien, die ein oder mehrere ältere Approximationsmodelle verwenden, d.h. dass die Differenz zwischen dem ersten und dem zweiten Index größer als Eins ist, sollen hier nicht dargestellt werden.

Wird eine konstante Anzahl von Dokumentvektoren festgelegt, z.B. indem in den weiteren Iterationen die gleiche Anzahl an Dokumentvektoren nachgewiesen wird, wie in der Initialisierungs-Iteration, so ergibt sich für die Ergebnisliste in t=1:

$$DV(AM(rel(x))^{t=0})^{t=1} = (x_j^{t=1} \mid rel(x_j^{t=1}) > rel(x_{j+1}^{t=1}), j = 1, ..., f^{t=0}), \text{ mit}$$
$$rel(x_1^{t=1}) = \max\{rel(x_p^{t=1}) \mid \forall x_p^{t=1} \in DV \setminus DVM(q_i^{t=0})\},$$

$$\text{rel}(x_2^{t=1}) = \max\{\text{rel}(x_p^{t=1}) \mid \forall x_p^{t=1} \in DV \setminus \{DVM(q_i^{t=0}) \cup \{x_1^{t=1}\}\}\},$$
$$...,$$
$$\text{rel}(x_{f(t=0)}^{t=1}) = \max\{\text{rel}(x_p^{t=1}) \mid \forall x_p^{t=1} \in DV \setminus \{DVM(q_i^{t=0}) \cup \{x_1^{t=1}, ..., x_{f(t=0)-1}^{t=1}\}\}\}. \quad (540)$$

Diese Strategie ist somit unabhängig von dem Original-Queryvektor $q_i^{t=0}$ und benötigt auch keine Adaptionsgleichung für die Anpassung des Queryvektors in den nachfolgenden Iterationen. Weiterhin ist sie unabhängig von der Festlegung bzw. Adaption von Retrieval-Mannigfaltigkeiten um Queryvektoren nach der Initialisierungs-Iteration.

Durch die weiteren Relevanzbewertungen des Agenten ist es möglich, das Approximationsmodell nach jeder Iteration zu aktualisieren, indem die Stimulusmenge aktualisiert wird. Nach der Iteration t=1 liegt zu jedem Element $x_j^{t=1}$ aus $DV(AM(\text{rel}(x))^{t=0})^{t=1}$ eine Relevanzbewertung $\text{rel}(x_j^{t=1})$ vor, sodass daraus ein Stimulus $m_j^{t=1}$ gebildet werden kann, der in die iterations-spezifische Stimulusmenge $M_{DV(m)}^{t=1}$ aufgenommen wird:

$$M_{DV(m)}^{t=1} = \{m_j^{t=1} = (x_j^{t=1}, \text{rel}(x_j^{t=1})) \mid j = 1, ..., f^{t=0}\}. \quad (541)$$

Die Gesamt-Stimulusmenge ergibt sich aus der Vereinigungsmenge der bis zu diesem Zeitpunkt vorliegenden iterations-spezifischen Stimulusmenge, d.h. $M_{DV(m)}^{t\leq 1}$ ergibt sich als $M_{DV(m)}^{t=0} \cup M_{DV(m)}^{t=1}$. Diese aktualisierte Stimulusmenge und ein Regressionsverfahren wie die instanzbasierte Kernel-Regression, ergeben ein aktualisiertes Approximationsmodell $AM(\text{rel}(x))^{t=1} := AM(\text{rel}(x) \mid M_{DV(m)}^{t\leq 1})$, mit dem die Elemente der Restmenge $DV \setminus \{DVM(q_i^{t=0}) \cup DV(q_i)^{t=1}\}$ bezüglich ihrer Relevanz bewertet werden können.

Im Verfahrensverlauf wird somit eine Sequenz von Approximationsmodellen $AM(\text{rel}(x))^{t=0}$, $AM(\text{rel}(x))^{t=1}$, ..., erzeugt, die jeweils auf einer streng monoton wachsenden Stimulusmenge basieren. Durch die zunehmende Kenntnis des Relevanzbewertungsverhaltens des Agenten kann davon gesprochen werden, dass die Approximationsmodelle zunehmend mehr Wissen integrieren und somit bei jeder Iteration besser werden. Andererseits kann die Qualität der einzelnen Modelle durch die unterschiedlichen Methoden quantifiziert werden, die im Abschnitt 2.3) beschrieben wurden, wobei es im konkreten Fall instanzbasierter Approximationsverfahren passieren kann, dass sich die Modellqualität kurzfristig verschlechtern kann, je nach dem verwendeten Qualitätsmaß. Dies kann als kurzfristiger Effekt eintreten oder als eine Form der Übergeneralisierung, die anzeigt, dass die bislang besuchte Region im DVR auf Grund der Stützpunkte so gut approximiert werden kann, dass weitere Stützpunkte keine Verbesserung, jedoch möglicherweise eine Verschlechterung mit sich führen werden.

Wird das bislang dargestellte Verfahren betrachtet, so muss das Abbruchkriterium vom Agenten ausgehen, da ohne weitere Einschränkungen das IRS die Iterationen solange weiter führt, bis die Restmenge leer ist, d.h. bis alle Dokumentvektoren einmal nachgewiesen und die korrespondierenden Dokumente dem Agenten präsentiert wurden. Durch Zusatzannahmen lassen sich Abbruchkriterien von Seiten des IRS konstruieren, indem ein Mindestwert der Relevanzschätzung festgelegt wird. Existieren nur noch Dokumentvektoren, die gemäß des momentanen Approximationsmodells eine Relevanzschätzung besitzen, die kleiner als der Relevanz-Mindestwert sind, so wird dies als Abbruchkriterium verwendet.

4.2.3) Feedback mit Approximationsmodell und nachträglicher Adaption des Queryvektors

Nach dem Abbruch existiert eine Stimulusmenge $M_{DV(m)}^{t(end)}$, die für bestimmte Zwecke verwendet werden kann, wie z.B. die Erzeugung eines besseren bzw. optimalen Queryvektors $q_i^{t(end)}$, indem $q_i^{t=0}$ durch eines der bekannten Verfahren adaptiert wird (siehe Abschnitt 4.2.1.1)). Z.B. kann $M_{DV(m)}^{t(end)}$ mit Hilfe eines Schwellenwertes S_{rel} in $M_{DV(m),rel}^{t(end)}$ und $M_{DV(m),\overline{rel}}^{t(end)}$ zerlegt werden, wobei die Stimuli für eine positive bzw. negative Adaption genutzt werden.

Der sich ergebende Queryvektor $q_i^{t(end)}$ kann als bessere Indexierung der ursprünglichen Query Q_i betrachtet werden, was Einfluss auf die Indexierungsfunktion $A_{IR(Q)}$ haben kann. Es kann ein Indexierungsfunktions-Relevanz-Feedback durchgeführt werden (siehe Abschnitt 3.9.6)), bei der nach einer Indexierungsfunktion $A_{IR(Q),i}$ gesucht wird, die nach dem Input der Query Q_i den Queryvektor $q_i^{t(end)}$ anstatt $q_i^{t=0}$ liefert:

$$A_{IR(Q),i}: D(\Theta) \rightarrow DVR: A_{IR(Q),i}(Q_i) = q_i^{t(end)}. \quad (542)$$

Wird eine komponentenweise Berechnung unterstellt, so berechnet sich eine Komponente $q_{ij}^{t(end)}$ aus der relativen Häufigkeit $r(F_j \mid Q_i)$ aus der Query und dem Gewichtungsfaktor $w(F_j \mid D)$ aus der gesamten Dokumentliste D (siehe Zobel & Moffat (1998[376])):

$$q_{ij}^{t(end)} = r(F_j \mid Q_i) * w(F_j \mid D). \quad (543)$$

Die relative Häufigkeit $r(F_j \mid Q_i)$ wird als Funktion der absoluten Häufigkeit $h(F_j \mid Q_i)$ in Q_i, und der Gewichtungsfaktor als Funktion der absoluten Häufigkeit $h(F_j \mid D)$ in D. Jede dieser beiden Funktionen kann durch eine Reihe von Parametern spezifiziert werden, wobei eine Berechnung der Parameter eindeutig möglich ist, wenn die Gesamtanzahl der Parameter gleich der Anzahl der Komponenten j = 1, ..., m ist, d.h. gleich der Anzahl der Merkmale, mit der Queries (und Dokumente) indexiert werden. In der Regel werden für die beiden Funktionen wesentlich weniger Parameter benötigt, sodass das Problem unterspezifiziert ist, d.h. ein Teil der Parameter kann frei festgelegt werden, während die anderen als Funktion dieser Parameter bestimmt werden.

4.2.4) Effizienzsteigerung des Modells ohne Queryvektor-Veränderung

4.2.4.1) Effizienzsteigerung durch Distanz- bzw. Kernel-Matrix

Die vorgestellte Retrievalstrategie besitzt jedoch ein erhebliches Effizienzproblem, da bei jeder Iteration alle verbleibenden Dokumentvektoren bewertet werden müssen, was bei der typisch großen Anzahl m^t von Elementen in der Gesamt-Dokumentvektorenliste DV^t einen hohen Aufwand bedeutet. Bei der Verwendung eines LWR-Verfahrens fällt der größte Anteil des Aufwandes durch die Distanzbestimmungen im Rahmen der Kernel-Funktion $h(d_{DVR}(x_k, x_j^{t=0}))$ an. Wird für alle m^t vorliegenden Dokumentvektoren eine Distanzmatrix erzeugt, die $m^t * (m^t/2 - 1)$ belegte Zellen $d_{DVR}(x_k, x_j)$ umfasst, so lässt sich das Problem entschärfen, indem daraus eine Kernel-Matrix mit den Zellen $h(d_{DVR}(x_k, x_j))$ berechnet wird, die für alle Retrievaloperationen beliebiger Agenten verwendet werden kann. Dies gelingt solange, bis die Dokumentvektorenmenge aktualisiert wird, d.h. bis neue Dokumentvektoren aufgenommen werden,

sodass die Distanz- und die Kernel-Matrix erweitert werden müssen. Bei einer nicht monoton wachsenden Dokumentvektorenmenge, d.h. wenn Dokumentvektorenlöschungen möglich sind, muss die Distanz- und die Kernel-Matrix ebenfalls entsprechend angepasst werden. Liegt eine Kernel-Matrix vor, so wird eine Relevanzschätzung als gewichtete Summe berechnet, was einen vergleichbaren Aufwand wie eine einzelne Distanzbestimmung ausmacht.

4.2.4.2) Effizienzsteigerung durch Einschränkung der Grundmenge

4.2.4.2.1) Einschränkung durch ε-Umgebung

Andere Möglichkeiten der Effizienzsteigerungen ergeben sich durch eine Beschränkung der Anzahl der Dokumentvektoren, die bei jeder Iteration bewertet werden. Dabei sind zwei prinzipielle Szenarien zu unterscheiden:
1) unklassifizierte Dokumentvektorenmenge.
2) klassifizierte Dokumentvektorenmenge (durch eine GNG-SOM).

Das Retrieval mit Relevanz-Approximationsmodellen bei einer GNG-SOM-klassifizierten Dokumentvektorenmenge wird in einem eigenen Abschnitt dargestellt (siehe Abschnitt 4.2.5)), sodass hier Verfahren im Rahmen einer unklassifizierten Dokumentvektorenmenge beschrieben werden sollen.

Eine einfache Möglichkeit ist die Erzeugung einer ε-Umgebung mit einem größeren Umgebungsparameter als in der Initialisierungs-Iteration, d.h. es wird $U(q_i^{t=0} \mid \varepsilon^{t=1})$ mit $\varepsilon^{t=1} > \varepsilon^{t=0}$ gebildet, und alle darin enthaltenen Dokumentvektoren werden als Gesamtergebnismenge nachgewiesen:

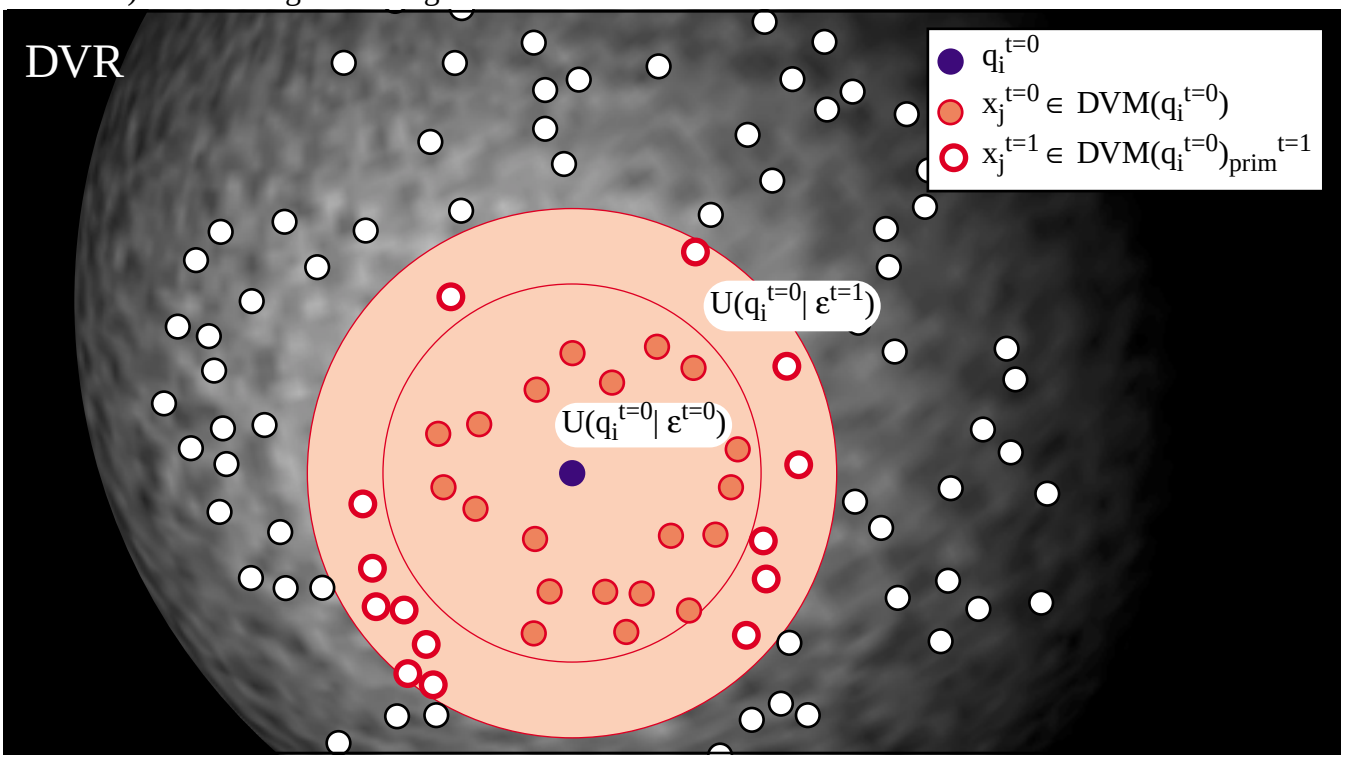
$$DVM(q_i^{t=0})^{t=1} = \{x_j^{t=1} \in DV \mid x_j^{t=1} \in U(q_i^{t=0} \mid \varepsilon^{t=1})\}. \tag{544}$$

Korrigiert wird diese Menge um die Dokumentvektoren, für die bereits ein Relevanzwert durch den Agenten ermittelt wurde, d.h. in der Iteration t=1 sind dies alle Dokumentvektoren aus der Gesamt-Ergebnismenge der Iteration t=0, die als $DVM(q_i^{t=0})^{t=0} := DVM(q_i^{t=0})$ bezeichnet werden soll (siehe Abb. 112)):

$$\begin{aligned} DVM(q_i^{t=0})_{prim}^{t=1} &= \{x_j^{t=1} \mid x_j^{t=1} \in DVM(q_i^{t=0})^{t=1} \wedge x_j^{t=1} \notin DVM(q_i^{t=0})^{t=0}\} \\ &= \{x_j^{t=1} \mid x_j^{t=1} \in \{DVM(q_i^{t=0})^{t=1} \setminus DVM(q_i^{t=0})^{t=0}\}\} \\ &= \{x_j^{t=1} \in DV \mid x_j^{t=1} \in U(q_i^{t=0} \mid \varepsilon^{t=1}) \wedge x_j^{t=1} \notin U(q_i^{t=0} \mid \varepsilon^{t=0})\}. \end{aligned} \tag{545}$$

Für die Elemente dieser Primärmenge $DVM(q_i^{t=0})_{prim}^{t=1}$ werden Relevanzschätzungen $rel(x_j^{t=1} \mid AM(rel(x))^{t=0})$ erzeugt, und die Dokumentvektoren werden nach fallender Relevanzschätzung geordnet. Sind in $DVM(q_i^{t=0})_{prim}^{t=1}$ mehr als $f^{t=0}$ Elemente enthalten, so werden die ersten $f^{t=0}$ Elemente der geordneten Liste als sekundäre Ergebnisliste $DV(q_i^{t=0})_{sek}^{t=1}$ verwendet, deren korrespondierende Dokumente dem Agenten vorgelegt werden. Sind in $DVM(q_i^{t=0})_{prim}^{t=1}$ gleich viele oder weniger als $f^{t=0}$ Elemente enthalten, so werden alle Elemente in einer geordneten Liste nachgewiesen.

Abb. 112) Primäre Ergebnismenge der Iteration t=1

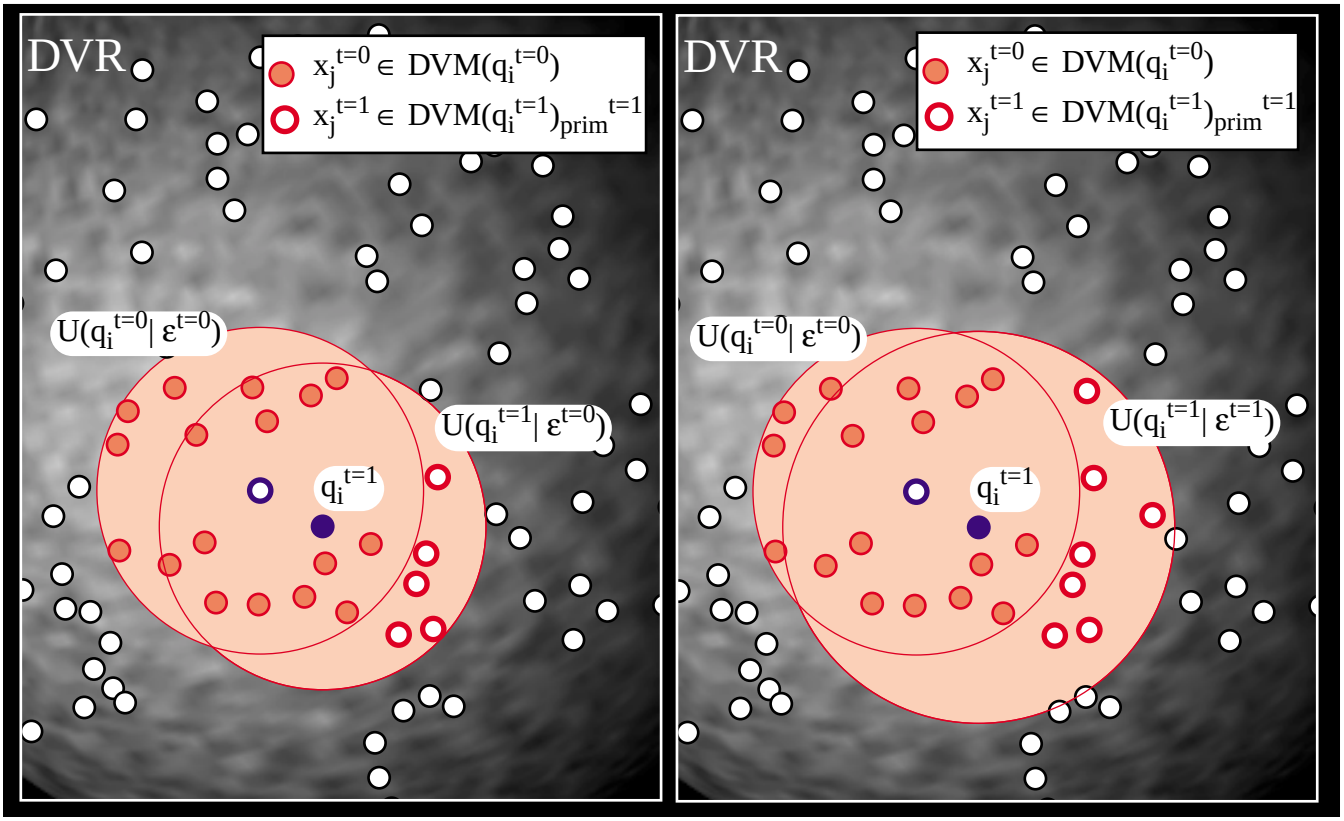


Im weiteren Verlauf des Verfahrens werden immer größere Umgebungen $U(q_i^{t=0} | \varepsilon^{t=2})$, $U(q_i^{t=0} | \varepsilon^{t=3})$, ..., erzeugt. Für alle Dokumentvektoren, die dem Agenten nicht präsentiert wurden, und für die somit keine richtige Relevanzwerte vorliegen, wird bei jeder Iteration eine Relevanzschätzung mit dem jeweils aktuellen Approximationsmodell erzeugt. D.h. in t=2 wird die Gesamtergebnismenge $DVM(q_i^{t=0})^{t=2}$ gebildet, die um $DVM(q_i^{t=0})^{t=0} \cup DV(q_i^{t=0})_{sek}^{t=1}$ korrigiert wird:

$$DVM(q_i^{t=0})_{prim}^{t=2} = \{x_j^{t=2} \mid x_j^{t=2} \in \{DVM(q_i^{t=0})^{t=2} \setminus \{DVM(q_i^{t=0})^{t=0} \cup DV(q_i^{t=0})_{sek}^{t=1}\}\},$$
$$DVM(q_i^{t=0})^{t=2} = \{x_j^{t=2} \mid x_j^{t=2} \in U(q_i^{t=0} \mid \varepsilon^{t=2})\}. \quad (546)$$

Eine Variante des vorgestellten Verfahrens ergibt sich, wenn bei jeder Iteration zusätzlich eine Queryvektor-Adaption durchgeführt wird. Zu unterscheiden ist dann, ob der gleiche Umgebungsparameter $\varepsilon^{t=0}$ wie in der Initialisierungs-Iteration verwendet wird, oder ob jeweils ein eigener, größerer Parameter verwendet wird. Im ersten Fall ergibt sich in der Iteration t=1 eine Umgebung $U(q_i^{t=1} | \varepsilon^{t=0})$ (siehe Abb. 113)a)) und im zweiten Fall $U(q_i^{t=1} | \varepsilon^{t=1})$ (siehe Abb. 113)b)).

Abb. 113) Queryvektor-Feedback mit gleicher bzw. größerer neuer Umgebung



4.2.4.2.2) Einschränkung durch GNG-SOM-Repräsentation

Weitere Möglichkeiten der Einschränkung von iterationsspezifischen Grundmengen ergeben sich durch die On-line-Clusterung der Ergebnismengen bzw. von geeigneten Teilmengen, was als eine Post-Retrieval-Operation im Abschnitt 3.9.2.4) eingeführt wurde. Hierbei wird in der Initialisierungs-Iteration eine SC-GNG-SOM $N_{D,q(i)}^{t=0}$ erzeugt, indem die Dokumentvektoren $x_j^{t=0}$ der Ergebnismenge $DVM(q_i^{t=0})$ als Stimuli verwendet werden. Die Neuronenstruktur $n_{D,k}^{t=0}$ besteht aus einem Gewichtsvektor $w(x_D)_k^{t=0}$ als Stützpunkt im DVR, einer lokalen Stimulusmenge $M_{DV(m),k}^{t=0}$, sowie der lokalen Verbindungsstruktur in Form des Verbindungsvektors $C_{D,k}^{t=0}$ (siehe Abb. 75)):

$$N_{D,q(i)}^{t=0} = \{n_{D,k}^{t=0} = (w(x_D)_k^{t=0}, M_{DV(m),k}^{t=0}, C_{D,k}^{t=0}) \mid k = 1, ..., \}. \quad (547)$$

Die Zielsetzung einer solchen temporären GNG-SOM ist es, mit Hilfe der Voronoi-Regionen, die sich aus dem Delaunay-Graphen $G_{D,q(i)}^{t=0}$ ergibt, eine Begrenzung der Grundmenge zu erzeugen, aus der die Primärergebnismenge der nächsten Iteration gebildet wird. Im Gegensatz zu der GNG-SOM-Repräsentation der Ergebnismenge im Kontext einer Post-Processing-Operation besitzt hier jeder der Dokumentvektoren $x_j^{t=0}$ eine Bewertung $rel(x_j^{t=0})$ durch den Agenten, sodass den Gewichtsvektoren $w(x_D)_k^{t=0}$ bzw. den lokalen Stimulusmengen $M_{DV(m),k}^{t=0}$ Relevanzschätzungen zugeordnet werden können. Die Zuordnung von Relevanzschätzungen zu Gewichtsvektoren wird im nächsten Abschnitt als Übergang von einem instanzbasierten zu einem prototypbasierten Approximationsmodell behandelt, sodass sich hier auf die lokalen Stimulusmengen fokussiert werden soll.

Abb. 114) GNG-SOM-Repräsentation einer Ergebnismenge

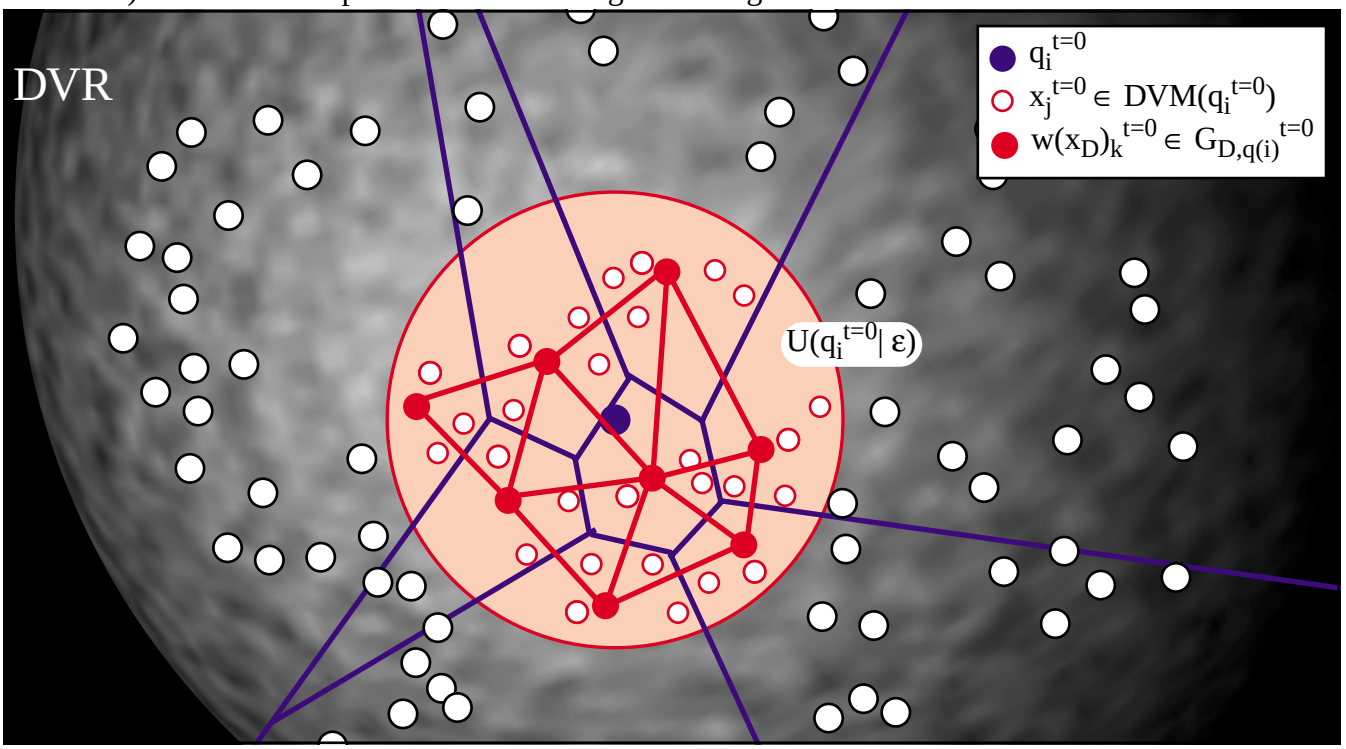
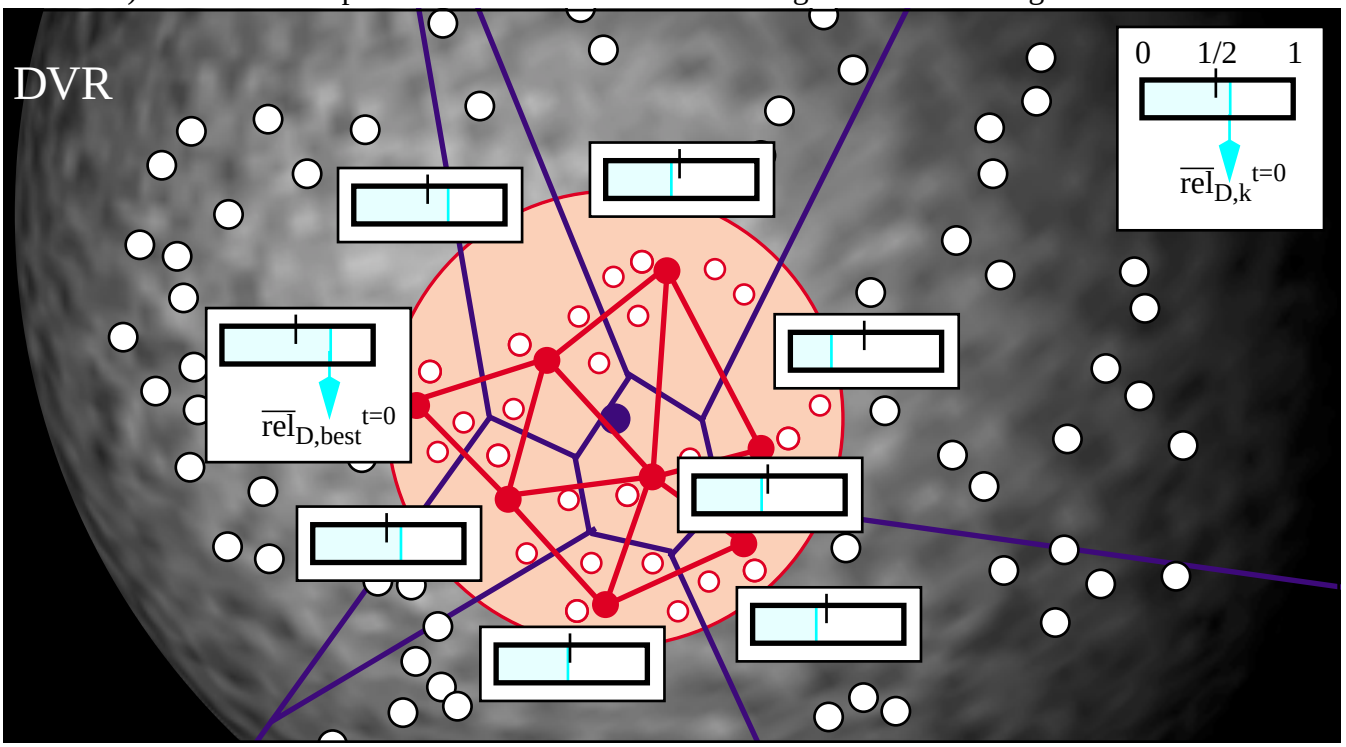


Abb. 115) GNG-SOM-Repräsentation mit Relevanzbewertung aller Voronoi-Regionen

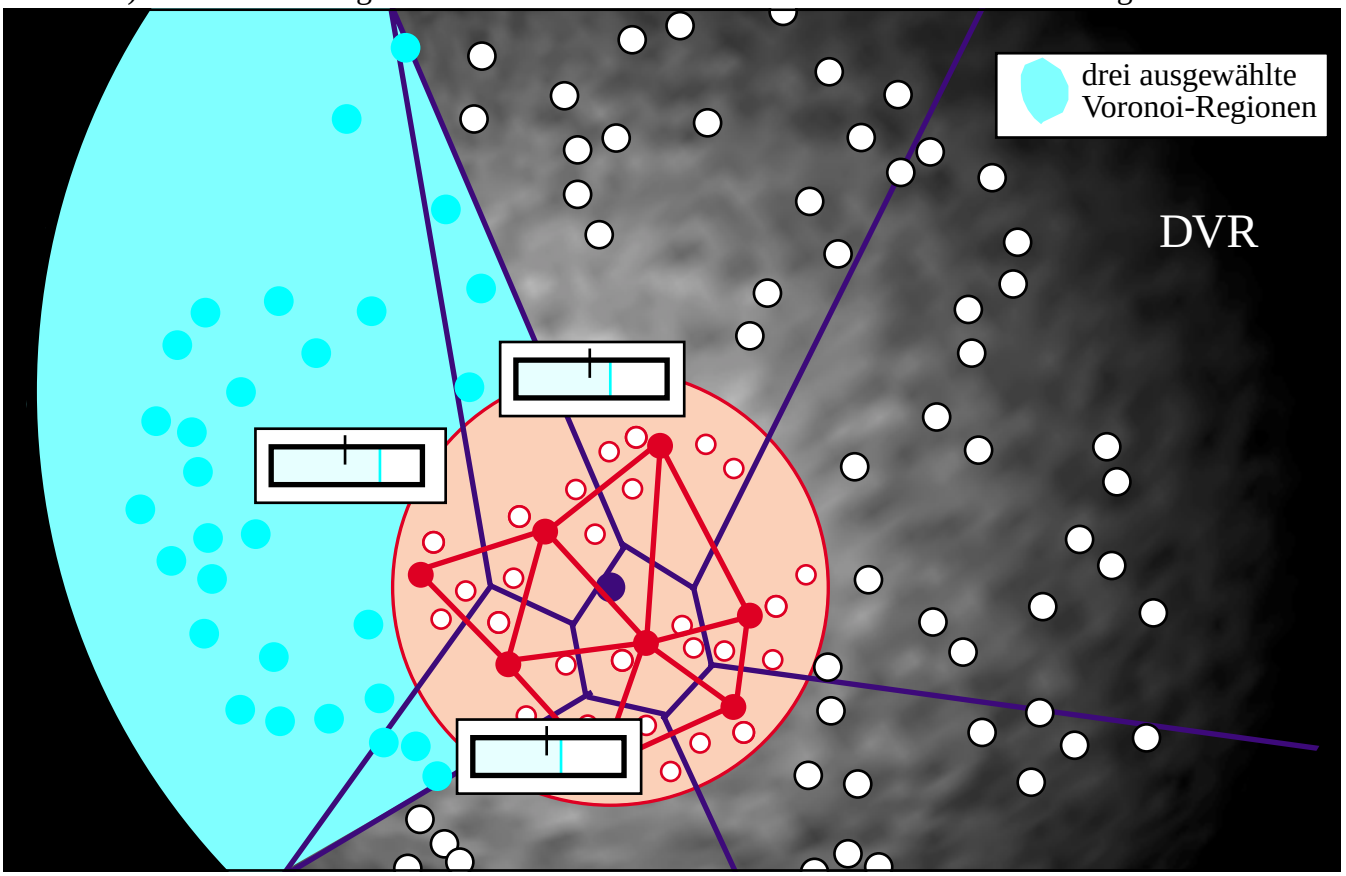


Ziel der Zuordnung von relevanzbasierten Bewertungen zu diesen Neuronen ist es, Voronoi-Regionen zu identifizieren, von denen erwartet wird, dass die darin liegenden, noch nicht durch den Agenten bewerteten Dokumentvektoren, hohe Relevanzwerte besitzen. Im einfachsten Fall kann einem Neuron $n_{D,k}^{t=0}$ der Relevanzmittelwert $\overline{rel}_{D,k}^{t=0}$ zugeordnet werden, (siehe Abb. 115)):

$$\overline{rel}_{D,k}^{t=0} = 1/\#M_{DV(m),k}^{t=0} * \sum_j rel(x_{jk}^{t=0}), \forall m_{jk}^{t=0} \in M_{DV(m),k}^{t=0}. \quad (548)$$

Beispielsweise können die besten drei Neurone mit dem höchsten Wert ausgewählt werden, wobei die noch nicht nachgewiesenen Dokumentvektoren aus den entsprechenden Voronoi-Regionen als Grundmenge für die Bestimmung der Ergebnismenge in der nachfolgenden Iteration t=1 verwendet werden (siehe Abb. 116)). D.h. für alle Elemente der Grundmenge wird eine Relevanzschätzung auf der Basis des momentan aktuellen Modells AM(rel(x))$^{t=0}$ gebildet, gefolgt von der Sortierung nach fallenden Relevanzschätzungen und der Auswahl der Dokumentvektoren mit den besten Schätzungen.

Abb. 116) Noch nicht nachgewiesene Dokumentvektoren aus den 3 besten Voronoi-Regionen



Alternativ zu einer einfachen Mittelwertsbildung kann ein Relevanz-Intervall bzw. eine Relevanz-Verteilung verwendet werden, um die besten Voronoi-Regionen zu identifizieren. Bei einem Relevanz-Intervall $KI_{rel,D,k}$ wird der Relevanzmittelwert und die -varianz gebildet, wobei ein Gewichtungsfaktor α als externer Parameter verwendet wird:

$$KI_{rel,D,k} = [\overline{rel}_{D,k}^{t=0} - \alpha * var_{rel,D,k}^{t=0}, \overline{rel}_{D,k}^{t=0} + \alpha * var_{rel,D,k}^{t=0}], \text{ mit}$$

$$var_{rel,D,k}^{t=0} = 1/\#M_{D,k}^{t=0} * \sum_j (\overline{rel}_{D,k}^{t=0} - rel(x_{jk}^{t=0}))^2, \forall m_{jk}^{t=0} \in M_{D,k}^{t=0}. \quad (549)$$

Ausgewählt werden dann die Neurone mit den besten Relevanz-Intervallen, wobei unterschiedliche Möglichkeiten existieren, ein Ranking auf der Basis von Intervallen mit Hilfe von Intervall-Arithmetiken durchzuführen (Bauch et al. (1987[30]); Klatte et al. (1993[182]); Kulisch (1993[192]); Bachelier (1998b:133ff[16]), siehe Abschnitt 2.6)). Eine einfache Strategie ist die Auswahl des Intervalls mit der größten rechten Intervallgrenze, sowie aller anderen Intervalle, die sich mit diesem Intervall überschneiden.

Zu beachten ist der Fall einer Voronoi-Region, in der außer den bereits nachgewiesenen Dokumentvektoren keine anderen Dokumentvektoren liegen. Diese werden ausgeschlossen, was jedoch eine Klassifikation aller noch nicht nachgewiesenen Dokumentvektoren bezogen auf das GNG-SOM erfordert. Da der Hauptaufwand hier durch Distanzbestimmungen zwischen Gewichtsvektoren und Dokumentvektoren anfällt, ist die GNG-SOM-basierte Vorgehensweise deutlich aufwendiger als ein Verfahren mit einer Distanz- bzw. Kernel-Matrix. Dies ergibt sich daraus, dass bei einer Distanzmatrix Distanzen zwischen Dokumentvektoren bestimmt werden, die solange konstant bleiben, bis neue Dokumentvektoren aufgenommen werden. Demgegenüber müssen die Distanzen zwischen Gewichts- und Dokumentvektoren bei jeder neuen GNG-SOM gebildet werden. Dies bedeutet, dass bei jeder Feedback-Iteration einer Retrieval-Operation eines Agenten diese Distanzen berechnet werden müssten.

4.2.5) Clusterung der Gesamtergebnismenge durch GNG-SOM

Ein instanzbasiertes Approximationsmodell $AM(rel(x))^t$ verwendet direkt eine Stimulusmenge $M_{DV(m)}^{\leq t}$ in Verbindung mit einem geeigneten Regressionsverfahren, um Relevanzschätzungen für Dokumentvektoren in einer Retrieval-Ergebnismenge zu erzeugen. Dabei muss beachtet werden, dass die Stimulusmenge monoton mit der Iterationsanzahl wächst, sodass die Güte der Schätzung, aber auch der Aufwand mit der Iterationsanzahl t zunimmt. Der Übergang von einer instanzbasierten zu einem prototypbasierten Verfahren bietet hier einen wesentlichen Ansatz zur Effizienzverbesserung, wobei standardmäßig GNG-SOMs zur Generierung von Prototypen verwendet werden sollen, was im nachfolgenden Abschnitt beschrieben werden soll. In diesem Abschnitt soll die Strukturierung einer primären Ergebnismenge betrachtet werden, um Dokumentvektoren-Prototypen zu erhalten, mit dem Ziel, dass der Agent weniger Dokumente bewerten muss. Diese Vorgehensweise korrespondiert zu einer der Post-Retrieval-Operationen, die in Abschnitt 3.9.2.4) dargestellt wurden.

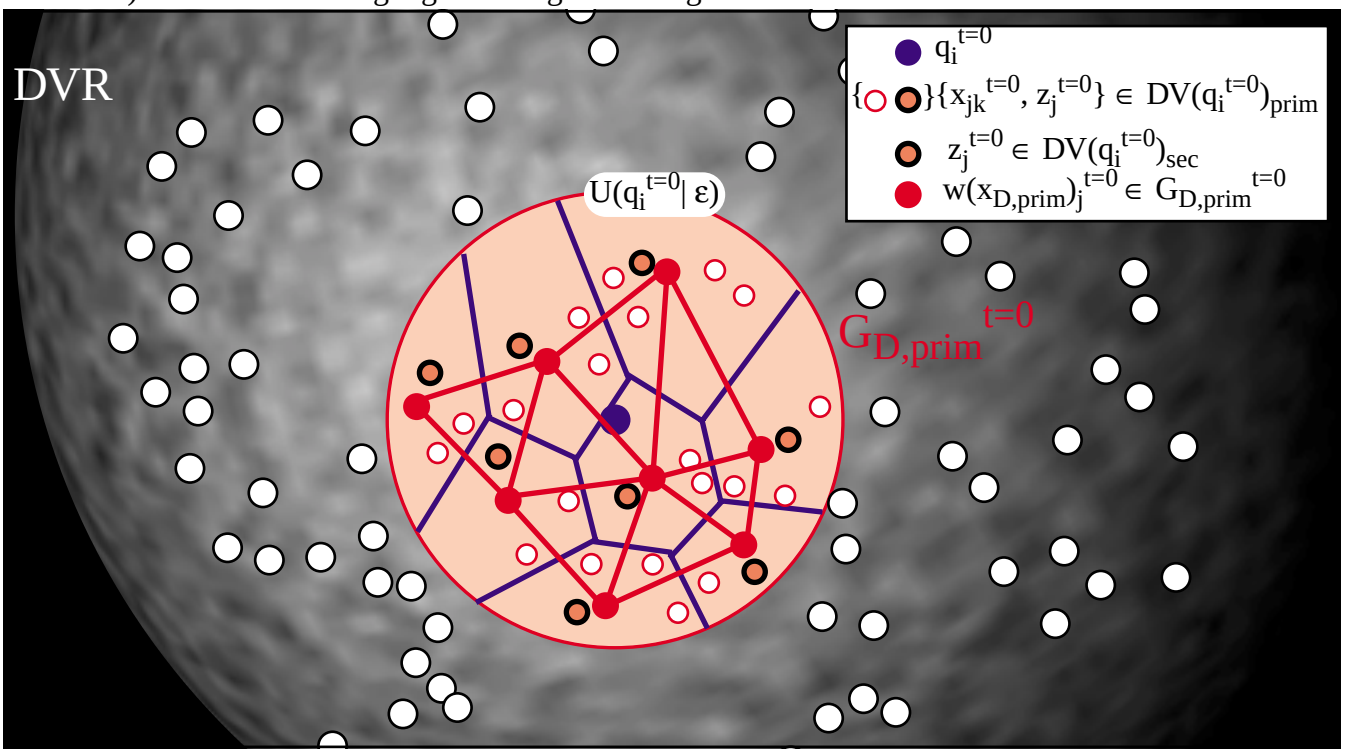
Bei der hier im Zusammenhang mit Ergebnismengen geschilderte Vorgehensweise soll zusätzlich eine Queryvektor-Adaption integriert werden, d.h. es werden Mengen $DV(q_i^{t=0})_{prim}$, $DV(q_i^{t=1})_{prim}$, ... betrachtet. Für jede dieser Mengen wird unabhängig in jeder betreffenden Iteration ein SC-GNG-SOM-Modell gebildet, das mit $N_{D,prim}^{t=0}$, $N_{D,prim}^{t=1}$, ... bezeichnet werden soll. Bei der Initialisierungs-Iteration wird $DV(q_i^{t=0})_{prim}$ durch die Umgebung $U(q_i^{t=0} \mid \varepsilon)$ erzeugt. Die Neuronenstruktur besteht aus einem Gewichtsvektor $w(x_{D,prim})_j^{t=0}$ als Stützpunkt im DVR, dem Verbindungsvektor $C_{D,prim,j}^{t=0}$ eines Neurons, sowie der lokalen Lernmenge, die für ein Neuron $n_{D,prim,j}^{t=0}$ mit $DV(q_i^{t=0})_{prim,j}$ bezeichnet werden soll:

$$N_{D,prim}^{t=0} = \{n_{D,prim,j}^{t=0} = (w(x_{D,prim})_j^{t=0}, DV(q_i^{t=0})_{prim,j}, C_{D,prim,j}^{t=0}) \mid j = 1, ... \},$$
$$\bigcup_j DV(q_i^{t=0})_{prim,j} = DV(q_i^{t=0})_{prim};$$
$$\forall n_{D,prim,j}^{t=0}: DV(q_i^{t=0})_{prim,j}: \neq \emptyset. \quad (550)$$

Für jedes Neuron $n_{D,prim,j}^{t=0}$ wird der Dokumentvektor $z_j^{t=0}$ aus $DV(q_i^{t=0})_{prim,j}$ als Medianvektor ermittelt, der die kleinste Distanz zu dem Gewichtsvektor $w(x_{D,prim})_j^{t=0}$ besitzt (siehe Abb. 117)):

$$d_{DVR}(z_j^{t=0}, w(x_{D,prim})_j^{t=0}) = \min\{d_{DVR}(x_{jk}^{t=0}, w(x_{D,prim})_j^{t=0}) \mid \forall x_{jk}^{t=0} \in DV(q_i^{t=0})_{prim,j}\}. \quad (551)$$

Abb. 117) GNG-SOM-Zerlegung einer Ergebnismenge mit Medianvektoren



Die sekundäre Dokumentvektorenmenge $DV(q_i^{t=0})_{sec}$ wird bei diesem Verfahren aus den Medianvektoren $z_j^{t=0}$ gebildet, die als Prototyp-Dokumentvektoren für jeweils eine lokale Stimulusmenge verwendet werden. D.h. die zugehörigen Dokumente zu diesen Prototyp-Dokumentvektoren werden dem Agenten ungeordnet oder sortiert nach steigenden Distanzen zu $q_i^{t=0}$ als sekundäre Ergebnismenge präsentiert.

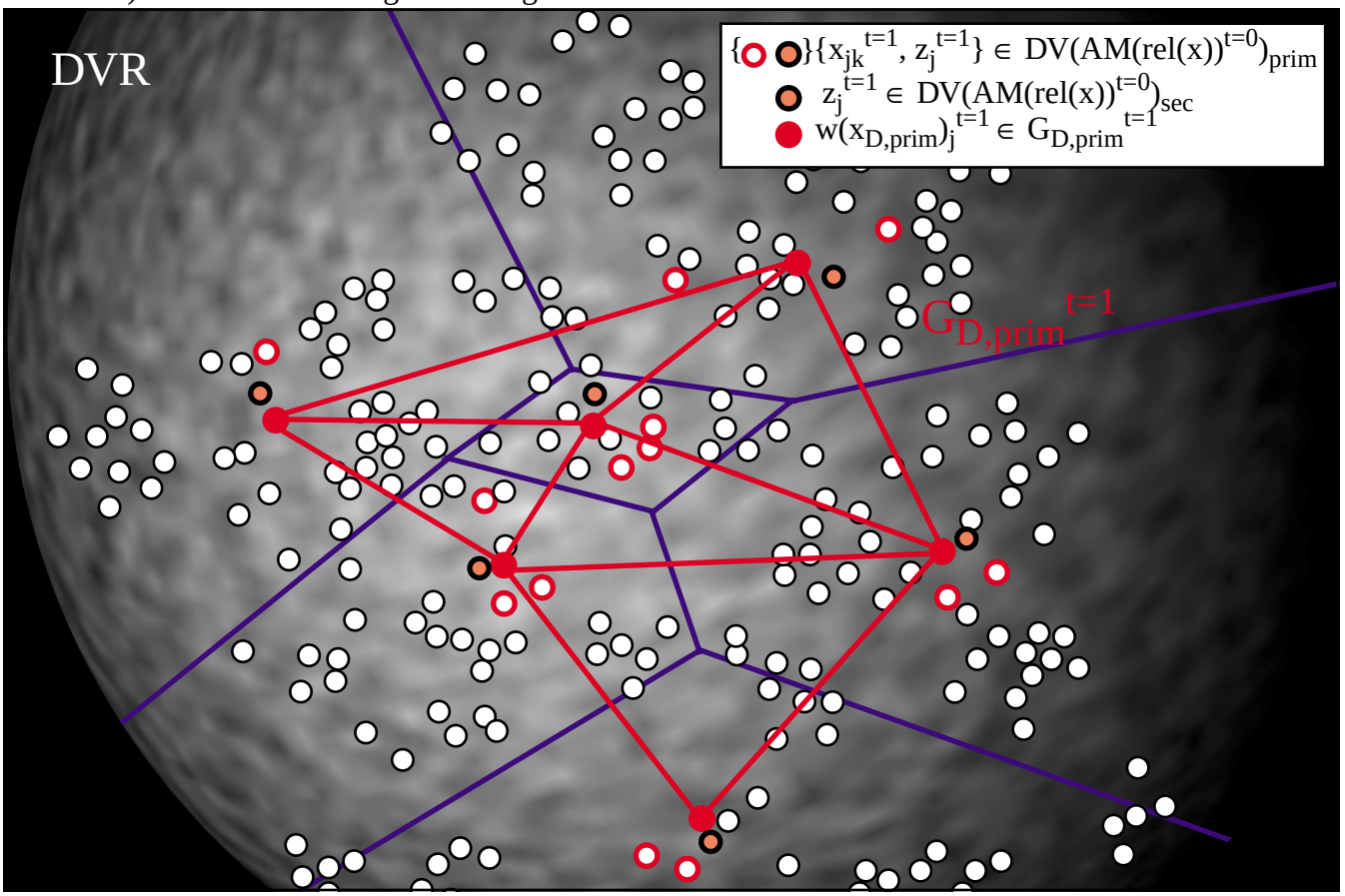
$$DVM(q_i^{t=0})_{sec} = \{z_j^{t=0} \mid \forall n_{D,prim,j}^{t=0} \in N_{D,prim}^{t=0}\} \text{ bzw.}$$
$$DV(q_i^{t=0})_{sec} = (z_j^{t=0} \mid d_{DVR}(q_i^{t=0}, z_j^{t=0}) < d_{DVR}(q_i^{t=0}, z_{j+1}^{t=0}); \forall n_{D,prim,j}^{t=0} \in N_{D,prim}^{t=0}). \quad (552)$$

Nach der Bewertung durch den Agenten wird die Stimulusmenge $M_{DV(m),sec}^{t=0}$ erzeugt, die zusammen mit einem LWR-Verfahren das Approximationmodell $AM(rel(x))^{t=0}$ bildet.

Wie die Restmenge $DVM(q_i^{t=0})_{prim} \setminus DVM(q_i^{t=0})_{sec}$ behandelt wird, kann durch verschiedene Vorgehensweisen festgelegt werden. Im einfachsten Fall werden die Elemente der Restmenge für die Iteration t=0 ignoriert, und da sie keine Bewertung besitzen, können sie in der nachfolgenden Iteration wieder Teil der Gesamtmenge werden, aus der die Primärmenge ermittelt wird. Wird dies unterstellt, so bleibt als letzte Operation innerhalb t=0 die Adaption des Queryvektors zu $q_i^{t=1}$ mit den Elementen aus $M_{DV(m),sec}^{t=0}$, womit die Initialisierungs-Iteration beendet ist und t=1 begonnen wird. In den weiteren Iterationen können die Primärmengen durch Umgebungen um die jeweiligen Queryvektoren aus der Restmenge wie $DV^t \setminus DVM(q_i^{t=0})_{sec}$ gebildet werden, oder es werden Retrievalverfahren mit der ausschlisslichen Hilfe des jeweiligen Approximationsmodells wie $AM(rel(x))^{t=0}$ durchgeführt, wobei in diesem Fall die Queryvektor-Adaption keine Rolle spielt. Im ersten Fall ist die Primär-Ergebnismenge immer lokal im DVR begrenzt (siehe Abb. 117)), während im zweiten Fall die Primärmenge über den gesamten DVR verteilt sein kann (siehe Abb. 118)). Bei der global verteilten Ergebnismenge durch Relevanzschätzungen mit $AM(rel(x))^{t=0}$ ist auch der GNG-SOM-Graph sowie die Medianvektoren global im DVR verteilt. Diese Medianvektoren bilden die Sekundär-Ergebnismenge, wobei der Queryvektor keine

weitere Rolle mehr spielt und durch die Abhängigkeit von dem momentan verwendeten Approximationsmodell $AM(rel(x))^{t=0}$ ersetzt wird, sodass die Bezeichnungen $DVM(AM(rel(x))^{t=0})_{prim}$ und $DVM(AM(rel(x))^{t=0})_{sec}$ verwendet werden können. Mit Hilfe der Relevanzbewertungen steht ab dieser Iteration die Möglichkeit offen, eine Liste $DV(AM(rel(x))^{t=0})_{sec}$ durch Sortieren nach fallender Relevanzschätzung zu erzeugen, anstatt eine Liste nach steigender Distanz zu dem aktuellen Queryvektor $q_i^{t=1}$.

Abb. 118) Global verteilte Ergebnismenge mit Medianvektoren



Wird der Queryvektor adaptiert und werden die Ergebnismengen lokal bestimmt, so kann die Restmenge $DVM(q_i^{t=0})_{prim} \setminus DVM(q_i^{t=0})_{sec}$ auch innerhalb der Iteration t=0 weiter verwendet werden, indem Schätzoperationen, die in dem obigen Beispiel erst in der nachfolgenden Iteration eingesetzt werden, vorgezogen werden. D.h. mit Hilfe von $AM(rel(x))^{t=0}$ werden Relevanzschätzungen für alle Dokumentvektoren der Restmenge erzeugt:

$$\forall x_k^{t=0} \in DVM(q_i^{t=0})_{prim} \setminus DVM(q_i^{t=0})_{sec}:$$

$$rel(x_k^{t=0})\wedge = rel(x_k^{t=0} \mid AM(rel(x))^{t=0}) = 1/\nu_k \sum_j h(d_{DVR}(z_j^{t=0}, x_k^{t=0})) * rel(z_j^{t=0}), \text{ mit}$$

$$\nu_k = \sum_j h(d_{DVR}(z_j^{t=0}, x_k^{t=0})), \forall z_j^{t=0} \in DVM(q_i^{t=0})_{sec}. \quad (553)$$

Diejenigen Dokumentvektoren, die bezüglich ihrer Relevanzschätzung besondere Eigenschaften besitzen, können dem Agenten direkt präsentiert werden, indem eine tertiäre Ergebnismenge $DVM(q_i^{t=0})_{ter}$ gebildet wird. Eine mögliche Eigenschaft wären Relevanschätzungen, die größer sind als der mittlere Relevanzwert $\overline{rel}(z^{t=0})_{sec}$, der durch die Bewertungen des Agenten ermittelt wurde:

$$DVM(q_i^{t=0})_{ter} = \{x_j^{t=0} \in DVM(q_i^{t=0})_{prim} \setminus DVM(q_i^{t=0})_{sec} \mid rel(x_j^{t=0}) \geq \overline{rel}(z^{t=0})_{sec}\}, \text{ mit}$$
$$\overline{rel}(z^{t=0})_{sec} = 1/\#DVM(q_i^{t=0})_{sec} * \sum_j rel(z_j^{t=0}), \forall z_j^{t=0} \in DVM(q_i^{t=0})_{sec}. \quad (554)$$

Die zu den Dokumentvektoren aus $DVM(q_i^{t=0})_{ter}$ korrespondierenden Dokumente werden dem Agenten präsentiert, der diese bewertet, sodass sich die Stimulusmenge $M_{DV(m),ter}^{t=0}$ ergibt, die zusammen mit $M_{DV(m),sec}^{t=0}$ die Menge $M_{DV(m)}^{t=0}$ bildet. Durch die neuen Stimuli ergibt sich ein aktualisiertes Modell $AM(rel(x))^{t=0}$, gefolgt von der Adaption zu $q_i^{t=1}$ und dem Beginn der nächten Iteration.

Eine alternative iterative Strategie ermittelt aus $DVM(q_i^{t=0})_{sec}$ genau das Element mit der größten Relevanzschätzung, dessen korrespondierendes Dokument durch den Agent bewertet wird. Mit dem daraus abgeleiteten Stimulus wird $M_{DV(m)}^{t=0}$ und somit $AM(rel(x))^{t=0}$ aktualisiert, gefolgt von einer Neuschätzung der Elemente aus der Restmenge, die alle durch den Agent noch nicht bewerteten Dokumente aus $DVM(q_i^{t=0})_{sec}$ oder $DVM(q_i^{t=0})_{prim}$ enthält. Da die Auswahl sich immer auf die Elemente aus $DVM(q_i^{t=0})_{sec}$ bezieht, muss ein zusätzliches Abbruchkriterium formuliert werden, da ansonsten iterativ alle Dokumentvektoren aus $DVM(q_i^{t=0})_{sec}$ ausgewählt werden, ohne dass die nächste, übergeordnete Iteration t=1 begonnen werden kann. Naheliegend ist dabei ein Abbruch, wenn alle Relevanzschätzungen auf der Basis des aktuellen Approximationsmodells der Elemente aus der momentanen Restmenge kleiner als ein Schwellenwert sind. In einem solchen Fall erscheint es sinnvoller, außerhalb von $DVM(q_i^{t=0})_{prim}$ liegende Dokumentvektoren zu prüfen, was durch eine Adaption zu $q_i^{t=1}$ und den Übergang zur nächten Iteration t=1 erfolgen kann.

Diese iterative Strategie besitzt ein Effizienzproblem, da viel mehr Relevanzschätzungen innerhalb einer Iteration t durchgeführt werden. Bei der ersten Sub-Iteration werden $\#DVM(q_i^{t=0})_{sec}$ Elemente bewertet, bei der zweiten Sub-Iteration $\#DVM(q_i^{t=0})_{sec}$ - 1, usw. Weiterhin wird durch die Erweiterung der Stimulusmenge um jeweils ein Element das daraus abgeleitete Approximationsmodell effektiver, doch gleichzeitig wird dessen Anwendung aufwendiger, da bei dem LWR-Verfahren in jeder Iteration ein zusätzlicher Summand hinzu kommt.

4.2.6) Prototypbasiertes GNG-SOM-Approximationsmodell aus Gesamtergebnismenge

Bislang wurde ein instanzbasiertes Approximationsmodell betrachtet, das durch eine Menge von Stützpunkten aus Dokumentvektor und Relevanzbewertung in Verbindung mit einem LWR-Verfahren definiert ist. In weiteren soll demgegenüber ein prototypbasiertes SC-GNG-SOM-Approximationsmodell betrachtet werden, das aus der jeweiligen Gesamtstimulusmenge gebildet und nach jeder Feedback-Iteration aktualisiert wird. Diese Vorgehensweise besitzt das Ziel, die Anzahl der Stützpunkte bei der nächsten Approximationsphase im Gegensatz zu einem instanzbasierten LWR-Approximationsmodell konstant zu halten bzw. kontrolliert und moderat steigen zu lassen.

Die Verwendung eines prototypbasierten SC-GNG-SOM-Approximationsmodells entspricht der Clusterung der bewerteten Dokumentvektoren im Gegensatz zu der Clusterung einer Gesamtergebnismenge (siehe Abschnitt 4.2.5)), da dort die Möglichkeit besteht, eine sekundäre Ergebnismenge als Teilmenge der Gesamtergebnismenge zu bilden.

In der Initialisierungs-Iteration t=0 wird ausgehend von $q_i^{t=0}$ die Dokumentvektoren-Ergebnismenge $DVM(q_i^{t=0})$ durch Distanzen im DVR durch $U(q_i^{t=0} \mid \varepsilon)$ bestimmt. Die zugehörigen Dokumente werden vom Agenten bewertet, sodass Stimuli $m_j^{t=0} = (x_j^{t=0}, rel(x_j^{t=0})$ in der Stimulusmenge $M_{DV(m)}^{t=0}$ gebildet werden können, die dazu verwendet werden, eine SC-GNG-SOM zu erzeugen, indem die Inputkomponenten der Stimuli Gewichtsvektoren im DVR präsentiert werden, die ihre Positionen entsprechend der positiven GNG-SOM-Adaptionsgleichung $w_i(neu) = w_i(alt) + \Delta w_i(alt)$ anpassen. Im Prinzip wird dabei eine Dokumentvektoren-GNG-SOM $N_D^{t=0}$ aus der Teilmenge der Dokumentvektoren gebildet, die bis zu diesem Zeitpunkt durch den Agenten bewertet wurde, sodass die Bezeichnung $N_{D,bew}^{t=0}$ verwendet werden soll. Die Neurone $n_{D,bew,i}^{t=0}$ des unüberwachten Modells bestehen wie üblich aus einem Gewichtsvektor $w(x_{D,bew})_i^{t=0} \in DVR$, der Verbindungsstruktur $C_{D,bew,i}^{t=0}$, sowie der lokalen Lernmenge $M_{DV(m),i}^{t=0}$, in der die Stimuli enthalten sind, deren Inputkomponente in der Voronoi-Region des Gewichtsvektors liegen:

$$N_{D,bew}^{t=0} = \{n_{D,bew,i}^{t=0} = (w(x_{D,bew})_i^{t=0}, M_{DV(m),i}^{t=0}, C_{D,bew,i}^{t=0}) \mid i = 1, ..., \mu_{N,D,bew}^{t=0}\},$$
$$\bigcup_i M_{DV(m),i}^{t=0} = M_{DV(m)}^{t=0}. \quad (555)$$

Da es sich bei dem Approximationsmodell $AM(rel(x))^t$ um ein überwachtes Modell handelt, muss jedes Neuron zusätzlich einen Stützpunkt im ein-dimensionalen Relevanzraum besitzen, der im gegebenen Fall gleich dem Intervall [0, 1] ist. Erzeugt werden kann dies durch ein unüberwachtes Modell, indem ein (n+1)-dimensionaler Vektor aus Input- und Output-Komponente der Stimuli für das Lernen verwendet wird, oder indem ein Batch-Lernverfahren in einer SC-GNG-SOM z.B. durch ein LWR-Verfahren angewendet wird (siehe Abschnitt 2.1.7)). Die zweite Alternative soll im weiteren verfolgt werden, wobei alle Stimuli, die in der lokalen Lernmenge $M_{DV(m),i}^{t=0}$ liegen, an der Schätzung der Relevanz an der Stelle des Gewichtsvektors $w(x_{D,bew})_i^{t=0}$ teilnehmen sollen:

$$rel(w(x_{D,bew})_i^{t=0})^\wedge = 1/\nu_i \sum_j h(d_{DVR}(w(x_{D,bew})_i^{t=0}, x_{ij}^{t=0})) * rel(x_{ij}^{t=0}), \text{ mit}$$
$$\nu_i = \sum_j h(d_{DVR}(w(x_{D,bew})_i^{t=0}, x_{ij}^{t=0})), \forall m_{ij}^{t=0} \in M_{DV(m),i}^{t=0}. \quad (556)$$

Auf diese Weise wird die Neuronenstruktur erweitert zu:

$$N_{D,bew}^{t=0} = \{n_{D,bew,i}^{t=0} = (w(x_{D,bew})_i^{t=0}, rel(w(x_{D,bew})_i^{t=0})^\wedge, M_{DV(m),i}^{t=0}, C_{D,bew,i}^{t=0})$$
$$\mid i = 1, ..., \mu_{N,D,bew}^{t=0}\}. \quad (557)$$

Das prototypbasierte GNG-SOM-Approximationsmodell $AM(rel(x))^{t=0}$ besteht somit aus den Stützpunkten, die durch die Neurone aus $N_{D,bew}^{t=0}$ gegeben sind, sowie aus einem Approximationsverfahren, das aus diesen Stützpunkten Relevanzschätzungen für beliebige Punkte aus dem DVR erzeugen kann.

Im weiteren Verlauf wird nach der Bildung von $AM(rel(x))^{t=0}$ die nächste Iteration t=1 eingeleitet, indem die gesamte Restmenge der noch nicht durch den Agenten bewerteten Dokumentvektoren durch $AM(rel(x))^{t=0}$ bewertet wird, bzw. es wird eine primäre Ergebnismenge $DVM(q_i^{t=0})_{prim}^{t=1}$ oder $DVM(q_i^{t=1})_{prim}$ ermittelt, wobei im ersten Fall der ursprüngliche Queryvektor und im zweiten Fall ein adaptierter Queryvektor als Zentrum der Retrievalregion verwendet wird. Unabhängig ob ein Dokumentvektor $x_j^{t=1}$ aus der Gesamt- oder einer Teilmenge stammt, für ihn wird eine Relevanzschätzung mit Hilfe des prototypbasierten $AM(rel(x))^{t=0}$ durchgeführt:

$$rel(x_j^{t=1} \mid AM(rel(x))^{t=0}) = rel(x_j^{t=1})^{\wedge} =$$
$$1/\nu_j \sum_i h(d_{DVR}(w(x_{D,bew})_i^{t=0}, x_j^{t=1})) * rel(w(x_{D,bew})_i^{t=0})^{\wedge}, \text{ mit}$$
$$\nu_j = \sum_i h(d_{DVR}(w(x_{D,bew})_i^{t=0}, x_j^{t=1})), \forall n_{D,bew,i}^{t=0} \in N_{D,bew}^{t=0}. \quad (558)$$

Es zeigt sich somit, dass die Relevanzschätzungen von Dokumentvektoren mit Hilfe von Relevanzschätzungen der Gewichtsvektoren bestimmt werden, die ihrerseits durch Relevanzwerte von Dokumentvektoren bestimmt werden, die vom Agenten stammen.

Im weiteren werden die bewerteten Dokumentvektoren nach fallenden Relevanzwerten geordnet, und die Dokumente, die zu den Dokumentvektoren mit den besten Relevanzschätzungen gehören, werden dem Agenten zur Bewertung vorgelegt. Wird als primäre Ergebnismenge die gesamte Restmenge $DV^t \setminus DVM(q_i^{t=0})$ verwendet, so wird durch die Sortierung nach fallenden Relevanzwerten die sekundäre Liste $DV(AM(rel(x))^{t=0})_{sec}$ erzeugt, aus der nach der Bewertung durch den Agenten die Stimulusmenge $M_{DV(m)}^{t=1}$ der Iteration t=1 gebildet wird. Bei einer Anzahl von $f^{t=1}$ Dokumenten, die von dem Agenten bewertet wurden, bzw. von $f^{t=1}$ Dokumentvektoren in der sekundären Liste ergibt sich:

$$M_{DV(m)}^{t=1} = \{m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1})) \mid j = 1, ..., f^{t=1}\}. \quad (559)$$

Die Gesamt-Stimulusmenge in t=1 soll mit $M_{DV(m)}^{t\leq 1}$ bezeichnet werden, die sich als Vereinigungsmenge von $M_{DV(m)}^{t=0}$ und $M_{DV(m)}^{t=1}$ ergibt. Das SC-GNG-SOM-Modell $N_{D,bew}^{t=0}$ wird im nächsten Schritt zu $N_{D,bew}^{t=1}$ aktualisiert, wobei sich die Vorteile bezüglich der Erweiterbarkeit von Stimulus-Cluster-GNG-SOMs besonders günstig auswirken (siehe Bachelier (1998c[17])). Es wird eine Clusterung der Stimuli aus $M_{DV(m)}^{t=1}$ erreicht, indem die Input-Komponenten $x_j^{t=1}$ der Stimuli $m_j^{t=1}$ dem Netz $N_{D,bew}^{t=0}$ präsentiert werden, wobei ein Stimulus dem Neuron $n_{D,bew,s(1|j)}^{t=1}$ zugeordnet wird, dessen Gewichtsvektor $w(x_{D,bew})_{s(1|j)}^{t=1}$ den geringsten Abstand zu der Input-Komponente $x_j^{t=1}$ besitzt, d.h. es wird eine SOM-basierte Form eines Single-Pass-Clusterverfahrens durchgeführt. Nachdem alle neuen Stimuli einem Neuron zugeordnet wurden, wird eine Nachadaptions- bzw. Feinadaptionsphase eingeleitet, bei der Stimuli entsprechend ihrer Input-Komponente umverteilt werden, bis alle Neurone einen Fehlerwert wie z.B. Quantifizierungsfehler (siehe Abschnitt 2.3.1)) kleiner einem Schwellenwert besitzen. Gelingt dies nicht, so werden neue Neurone in die Verbindungsstruktur eingefügt, gefolgt von erneuten Umverteilungen, bis der Schwellenwert unterschritten wird.

Nachdem das Netz $N_{D,bew}^{t=1}$ auf der Basis der Stützpunkte im DVR aktualisiert wurde, werden die Relevanzschätzungen der Gewichtsvektoren durch die Stimuli erneuert, die in der lokalen Stimulimenge des betreffenden Neurons liegen. Das Netz $N_{D,bew}^{t=1}$ bildet zusammen mit dem verwendeten LWR-Verfahren somit das Approximationsmodell $AM(rel(x))^{t=1}$, sodass die Nachfolge-Iteration t=2 begonnen werden kann, indem die restlichen noch nicht bewerteten Dokumentvektoren $x_j^{t=2}$ in $DV^t \setminus \{DVM(q_i^{t=0}) \cup DV(AM(rel(x))^{t=0})_{sec}\}$ einer neuen Relevanzschätzung $rel(x_j^{t=2} \mid AM(rel(x))^{t=1}) = rel(x_j^{t=2})^{\wedge}$ unterzogen werden.

4.2.7) Combining-Strategie bei Ergebnismengenbildung

Werden reelle Relevanzwerte verwendet, so können wie oben dargestellt, zwei unabhängige Verfahren zur Ergebnismengenbildung durchgeführt werden:

1) Verwendung von Umgebungen im DVR um einen Queryvektor bzw. eine Queryvektor-Polyrepräsentation und die Adaption des oder der vorliegenden Vektoren.
2) Verwendung von Relevanzschätzungen durch Approximationsmodelle.

Unabhängig sind die beiden Verfahren dadurch, dass im ersten Fall die Relevanzbewertungen zur Adaption des Queryvektors durch Adaptionsgleichungen verwendet werden, die eine Analogie zu den Verfahren beim Relevanz-Feedback mit binären Relevanzbewertungen besitzen, ohne dass in irgend einer Form Relevanz-Approximationsmodelle verwendet werden. Im zweiten Fall muss ein expliziter Queryvektor nach der Initialisierungs-Iteration keine Rolle mehr spielen, sondern die Ergebnismenge bzw. -liste wird jeweils durch die Relevanzschätzungen und die Sortierung der Dokumentvektoren durchgeführt.

In der Iteration $t > 0$ führt diese Strategie im ersten Fall zu einer Ergebnismenge $DVM(q_i^t)$ und im zweiten Fall zu einer Ergebnismenge $DVM(AM(rel(x))^t)$. Bei einer Combining-Strategie werden beide Verfahren unabhängig durchgeführt und die beiden Ergebnismengen werden zu einer gemeinsamen Dokumentvektorenmenge zusammengeführt. Dabei sind die Verfahren anwendbar, die im Zusammenhang mit der Polyrepräsentation eingeführt wurden, z.B. bei der Queryvektor-Polyrepräsentation q_{ik}^t, die die Einzel-Ergebnismengen $DVM(q_{ik}^t)$ führen, die zu einer gemeinsamen Menge $DVM(QVM_i^t)$ aggregiert werden (siehe Abschnitte 3.6.2.2) und 3.6.3.2)). Die einfachste Strategie ist dabei die Bildung der Vereinigungsmenge bzw. ein grobes Ranking mit zwei Plätzen, bei der die Dokumentvektoren, die in der Schnittmenge liegen, auf den ersten Platz gesetzt werden, während die restlichen Elemente der Vereinigungsmenge auf den zweiten Platz gesetzt werden.

4.3) Feedback mit reellen Relevanzbewertungen bei Dokumentvektoren-GNG-SOMs

Bei der Durchführung einer Retrieval-Strategie unter Verwendung von Relevanzschätzungen durch Approximationsmodelle wurde ein Effizienzproblem identifiziert, dessen Behebung im Rahmen einer unklassifizierten Dokumentvektorenmenge nur bedingt gelingt (siehe Abschnitt 4.2.4)). Die dort versuchten Strategien beziehen sich auf die Einschränkung der Anzahl der Dokumentvektoren aus einer jeweiligen Restmenge, denen eine Relevanzschätzung zugeordnet werden soll. Die gemachten Versuche laufen im Prinzip darauf hinaus, eine temporäre Strukturierung der ansonsten unstrukturierten Dokumentvektorenmenge zu erzeugen, d.h. eine Form von On-Line-Clusterung, gefolgt von Auswahloperationen, die eine Einschränkung der zu schätzenden Dokumentvektoren erzeugt. Die temporäre Strukturierung wird für eine Interaktion zwischen IRS und Agent erzeugt, bzw. im Extermfall für eine Feedback-Iteration innerhalb einer IRS-Agent-Interaktion. Verbunden ist damit jeweils ein großer Aufwand, der bei einer a priori durchgeführten Dokumentvektoren-Strukturierung unabhängig von individuellen IRS-Agent-Interaktion erbracht wurde, sodass die Einschränkung der Grundmenge nur durch die Auswahloperationen erzeugt werden muss.

Im weiteren wird eine Dokumentvektoren-Klassifikation durch eine SC-GNG-SOM unterstellt, d.h. es wird ein Netz N_{DV} unterstellt, das durch die momentan vorliegende Dokumentvektorenmenge DVM erzeugt wurde, und diese in disjunkte Teilmengen zerlegt. In der Grundform ist dieses Netz unabhängig von individuellen IRS-Agent-Interaktionen und den daraus bestehenden Feedback-Iterationen, sodass kein Iterationsindex t verwendet wird, was sich dann ändert, wenn das Netz im Kontext einer konkreten Sequenz von Feedback-Iterationen betrachtet wird. Ein Neuron $n_{DV,j}$ dieses Netzes besteht aus einem Gewichtsvektor $w(x_{DV})_j$ als einem Stützpunkt im DVR, der Repräsentation seiner lokalen Verbindungsstruktur durch den Verbindungsvektor $C_{DV,j}$, sowie der lokalen Dokumentvektorenmenge $M_{DV,j}$, in der die Dokumentvektoren aus DVM enthalten sind, die innerhalb der Voronoi-Region $\tilde{R}_{DV,j}$ liegen:

$$N_{DV} = \{n_{DV,j} = (w(x_{DV})_j, M_{DV,j}, C_{DV,j}) \mid j = 1, ..., \mu_{N,DV}\}. \quad (560)$$

Auf die Darstellung von Strategien, die eine Queryvektor-Adaption verwenden, soll im weiteren verzichtet werden, da eine solche Adaption den Queryvektor meist innerhalb einer Voronoi-Region verschiebt, insbesondere wenn als Ergebnismenge die lokale Stimulusmenge des Gewinner-Neurons in Verbindung mit nicht so großen Adaptionsgewichten (Verschiebegewichten) verwendet wird (siehe auch Abschnitt 3.9.2.7)). Wird eine Adaption verwendet, bei der gefordert wird, dass der Nachfolge-Queryvektor außerhalb der Voronoi-Region des Vorgänger-Queryvektors liegt, so besitzt dies faktisch keinen Einfluss auf die Dokumentvektoren-Ergebnismenge, wenn nicht nur das Gewinner-Neuron, d.h. das Neuron, in dessen Voronoi-Region der Queryvektor liegt, ausgewählt wird, sondern auch seine unmittelbar verbundenen Nachbarn, da eine Verschiebung über diese Voronoi-Regionen kaum sinnvoll ist. Aus diesen Gründen sollen im weiteren Strategien dargestellt werden, bei denen der Queryvektor $q_i^{t=0}$ nur in der Initialisierungs-Iteration t=0 eine Rolle spielt, und bei denen in den nachfolgenden Iterationen die Bestimmung der Ergebnismengen und -listen durch ein instanzbasiertes oder prototypbasiertes Relevanz-Approximationsmodell erfolgt.

4.3.1) Dokumentvektoren-GNG-SOM mit instanzbasiertem Modell

In der Initialisierungs-Iteration t=0 einer konkreten IRS-Agent-Interaktion wird für einen Queryvektor $q_i^{t=0}$ das Gewinner-Neuron $n_{DV,s(1|i)}^{t=0}$ bestimmt, in dessen Voronoi-Region $\tilde{R}_{DV,s(1|i)}$ der Queryvektor liegt:

$$d_{DVR}(q_i^{t=0}, w(x_{DV})_{s(1|i)}^{t=0}) = \min\{d_{DVR}(q_i^{t=0}, w(x_{DV})_j) \mid \forall n_{DV,j} \in N_{DV}\}. \quad (561)$$

Die einfachste Retrieval-Strategie besteht darin, alle Dokumentvektoren auszuwählen, die in der lokalen Menge $M_{DV,s(1|i)}^{t=0}$ des Gewinner-Neurons $n_{DV,s(1|i)}^{t=0}$ liegen, und als Ergebnismenge $DVM(q_i^{t=0})$ der Initialisierungs-Iteration zu verwenden:

$$DVM(q_i^{t=0}) := M_{DV,s(1|i)}^{t=0}. \quad (562)$$

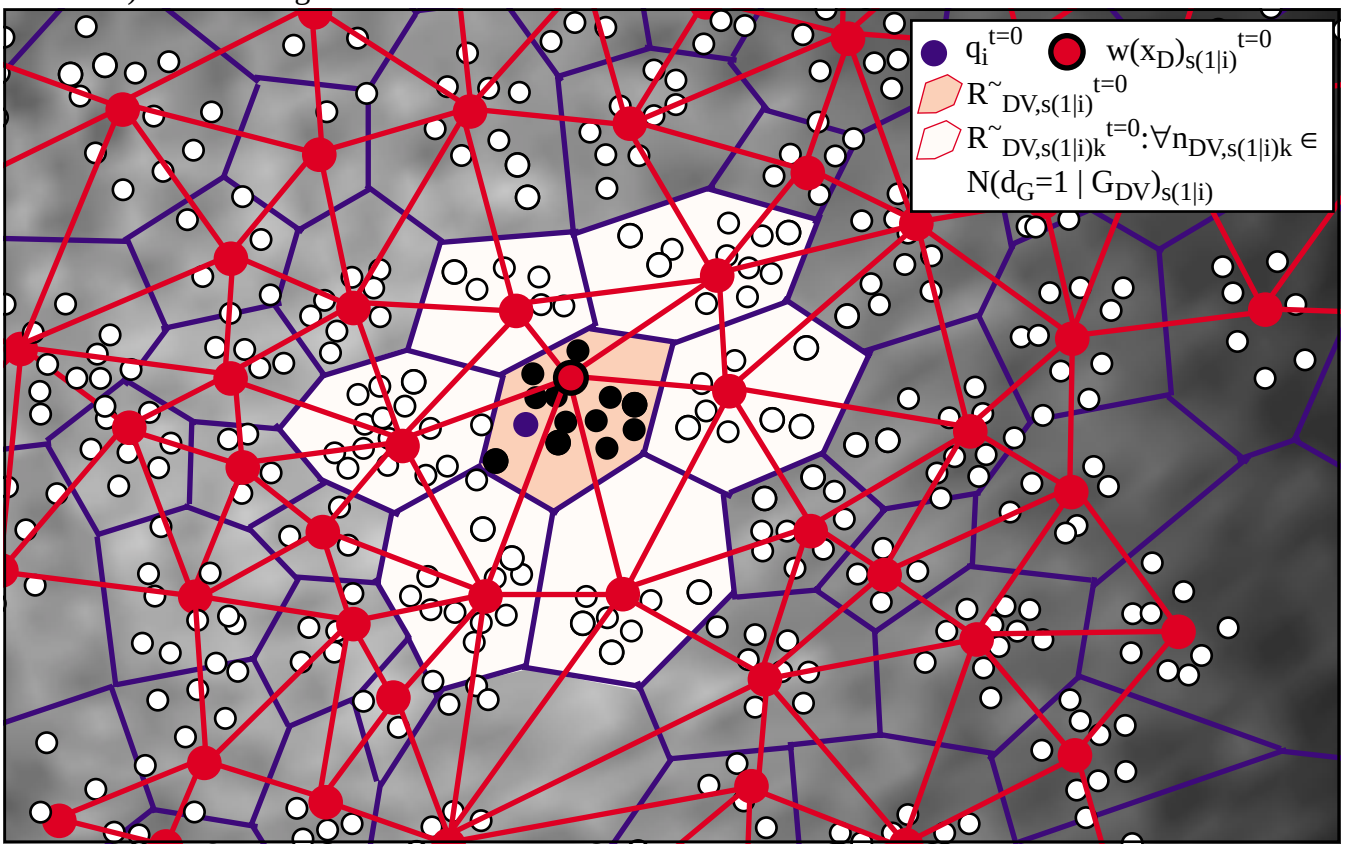
In der Initialisierungs-Iteration existiert noch kein Relevanz-Approximationsmodell durch das ein Ranking der Dokumentvektoren gebildet werden kann. Trotzdem kann die Menge $DVM(q_i^{t=0})$ in eine geordnete Liste $DV(q_i^{t=0})$ überführt werden, indem die Dokumentvektoren aus $M_{DV,s(1|i)}^{t=0}$ nach steigender Distanz zu $q_i^{t=0}$ geordnet werden. Die korrespondierenden Dokumente zu dieser Dokumentvektorenliste werden in der entsprechenden Reihenfolge dem Agenten präsentiert, der jeweils eine Relevanzbewertung

abgibt. Nach diesen Bewertungen durch den Agenten existiert die erste Stimulusmenge $M_{DV(m)}^{t=0}$ mit Elementen des Typs m = (x, rel(x)) und somit ein erstes Approximationsmodell $AM(rel(x))^{t=0}$ = $AM(rel(x) \mid M_{DV(m)}^{t=0})$, da eine instanzbasierte Approximation unterstellt wird. Mit diesem Modell können die noch nicht bewerteten Dokumentvektoren bezüglich ihrer Relevanz geschätzt werden, d.h. alle Dokumentvektoren außer den Elementen aus $M_{DV,s(1|i)}^{t=0}$ können bei einer Strategie ohne Einschränkung der Grundmenge geschätzt werden.

Ziel ist es jedoch, mit Hilfe der Strukturierung durch N_{DV} bzw. den Verbindungsgraphen G_{DV} eine Einschränkung zu erzeugen. Die einfachste Strategie besteht darin, zunächst die Neurone $n_{DV,s(1|i)k}$ zu ermitteln, die mit dem Gewinner-Neuron direkt verbunden sind, d.h. die im Verbindungsvektor $C_{s(1|i)k}$ eine entsprechende Komponente besitzen, die mit 1 belegt ist. Diese Neurone bilden die Nachbarschaftsmenge $N(d_G=1 \mid G_{DV})_{s(1|i)}$ (siehe Abb. 119)):

$$N(d_G=1 \mid G_{DV})_{s(1|i)} = \{n_{DV,s(1|i)k} \in N_{DV} \mid c_{s(1|i),s(1|i)k} = 1\}. \quad (563)$$

Abb. 119) Voronoi-Region des Gewinner-Neurons und seiner Nachbarn



Die primäre Ergebnismenge der Iteration t=1 soll als Vereinigungsmenge der lokalen Stimulusmengen der Neurone aus $N(d_G=1 \mid G_{DV})_{s(1|i)}$ gebildet werden, wobei die Abhängigkeit der Ergebnismenge von $q_i^{t=0}$ im Gegensatz zu t=0 nicht mehr im Zentrum steht, sodass allgemein $DVM_{prim}^{t=1}$ verwendet werden soll:

$$DVM_{prim}^{t=1} = \bigcup_k M_{DV,s(1|i)k}, \forall n_{DV,s(1|i)k} \in N(d_G=1 \mid G_{DV})_{s(1|i)}. \quad (564)$$

Allgemeiner lässt sich dies mit Hilfe der Neurone aus N_{DV} formulieren, in deren Voronoi-Regionen mindestens ein Dokumentvektor liegt, dem bislang eine Relevanzbewertung durch den Agenten zugeordnet wurde. Diese Menge der Neurone mit bewerteten Dokumentvektoren soll mit $N_{DV,bew}{}^{t} \subset N_{DV}{}^{t}$ bezeichnet werden, wobei sie vor der Bildung der Dokumentvektoren-Ergebnismenge in einer Iteration t gebildet wird. Dies bedeutet, dass $N_{DV,bew}{}^{t=0}$ leer ist, und dass $N_{DV,bew}{}^{t=1}$ gleich der ein-elementigen Menge mit dem Gewinner-Neuron $n_{DV,s(1|i)}{}^{t=0}$ ist. Wird unterschieden zwischen $M_{DV,k}$ als der Dokumentvektorenmenge eines Neurons $n_{DV,k}$ aus N_{DV}, und $M_{DV(m),k}{}^{\leq t}$ als der Stimulusmenge, deren Elemente die Struktur m = (x, rel(x)) besitzen, so handelt es sich bei $N_{DV,bew}{}^{t}$ um alle Neurone aus N_{DV}, für die $M_{DV(m),k}{}^{\leq t-1}$ nicht leer ist:

$$N_{DV,bew}{}^{t} = \{n_{DV,k} \in N_{DV} \mid M_{DV(m),k}{}^{\leq t-1} \neq \varnothing\}. \tag{565}$$

Die primäre Ergebnismenge $DVM_{prim}{}^{t}$ kann allgemein durch die Neurone aus $N_{DV,bew}{}^{t}$ und ihre Nachbarneurone definiert werden, korrigiert um die Gesamtmenge der Dokumentvektoren, die zu den Stimuli aus $M_{DV(m)}{}^{\leq t-1}$ gehören:

$$DVM_{prim}{}^{t} = \{\bigcup_k M_{DV,r}\} \setminus \{x_j \mid \forall m_j \in M_{DV(m)}{}^{\leq t-1}\},$$
$$\forall n_{DV,r} \in N(d_G \leq 1 \mid G_{DV})_k{}^{t}, \forall n_{DV,k} \in N_{DV,bew}{}^{t}. \tag{566}$$

Für die Iteration t=1 bedeutet dies:

$$DVM_{prim}{}^{t=1} = \{\bigcup_k M_{DV,r}\} \setminus \{x_j \mid \forall m_j \in M_{DV(m)}{}^{t=0}\},$$
$$\forall n_{DV,r} \in N(d_G \leq 1 \mid G_{DV})_k{}^{t}, \forall n_{DV,k} \in N_{DV,bew}{}^{t} = \{n_{DV,s(1|i)}{}^{t=0}\}. \tag{567}$$

Für die Dokumentvektoren aus $DVM_{prim}{}^{t=1}$ wird im weiteren eine Relevanzschätzung mit Hilfe $AM(rel(x))^{t=0}$ durchgeführt:

$$rel(x_j{}^{t=1} \mid AM(rel(x) \mid M_{DV(m)}{}^{t=0})) = rel(x_j{}^{t=1})\wedge, \forall x_j{}^{t=1} \in DVM_{prim}{}^{t=1}. \tag{568}$$

Die sekundäre Ergebnisliste ergibt sich, indem die Elemente aus $DVM_{prim}{}^{t=1}$ entsprechend der fallenden Relevanzschätzung geordnet werden:

$$DV_{sec}{}^{t=1} = (x_j{}^{t=1} \mid rel(x_j{}^{t=1})\wedge > rel(x_{j+1}{}^{t=1})\wedge; \forall x_j{}^{t=1} \in DVM_{prim}{}^{t=1}). \tag{569}$$

Die tertiäre Ergebnisliste $DV_{ter}{}^{t=1}$ ergibt sich, indem die besten, d.h. die Elemente auf den ersten Rängen der geordneten Liste $DV_{sec}{}^{t=1}$, ausgewählt werden, wobei die Anzahl konstant der ursprünglichen Anzahl der Elemente in $DVM(q_i{}^{t=0}) := M_{DV,s(1|i)}{}^{t=0}$ gewählt werden kann. Sollten in $DVM_{prim}{}^{t=1}$ weniger Elemente als in $DVM(q_i{}^{t=0})$ enthalten sein, so werden alle Elemente übernommen, doch für das Ranking werden die Relevanzschätzungen und das Sortieren trotzdem benötigt. Werden alle Elemente der tertiären Liste in allen Iterationen durch den Agenten bewertet, so lässt sich die primäre Ergebnismenge einer Iteration t einfach mit Hilfe der Vereinigung aller tertiären Listen der vergangenen Iterationen beschreiben:

$$DVM_{prim}{}^{t} = \{\bigcup_k M_{DV,r}\} \setminus \{\bigcup_k DV_{ter}{}^{\leq t-1}\},$$
$$\forall n_{DV,r} \in N(d_G \leq 1 \mid G_{DV})_k{}^{t}, \forall n_{DV,k} \in N_{DV,bew}{}^{t}. \tag{570}$$

Kann nicht davon ausgegangen werden, dass ein Agent immer alle Dokumente bewertet, die zu einer tertiären Dokumentvektorenliste korrespondieren, so wird allgemein die Menge der Neurone, denen bislang ein Stimulus m = (x, rel(x)) zugeordnet wurde, für die Ergebnismengenermittlung herangezogen.

Die Bewertungen der korrespondierenden Dokumente zu der Dokumentvektorenliste $DV_{ter}^{t=1}$ durch den Agenten werden den Dokumentvektoren in $DV_{ter}^{t=1}$ zugeordnet, wodurch neue Stimuli bzw. Stützpunkte im DVR und im Relevanzraum erzeugt werden, die in der Menge $M_{DV(m)}^{t=1}$ zusammengefasst werden, die vereinigt mit $M_{DV(m)}^{t=0}$ die Gesamtmenge $M_{DV(m)}^{t\leq 1}$ aller bisherigen Stimuli ergibt. Beim Vorliegen eines instanzbasierten Modells, liegt mit $M_{DV(m)}^{t\leq 1}$ auch das aktualisierte Modell zu $AM(rel(x))^{t=1}$ vor. Es soll kein Modell betrachtet werden, welches ausschließlich auf den Instanzen einer Iteration aufbaut, sodass in t=1 zwischen $AM(rel(x))^{t=1}$ und $AM(rel(x))^{t\leq 1}$ nicht unterschieden werden soll, d.h. es soll im weiteren die Bezeichnung $AM(rel(x))^{t=1}$ verwendet werden.

Erfolgt kein Abbruch von Seiten des Agenten, so wird nach der Spezifizierung von $AM(rel(x))^{t=1}$ die nächste Iteration t=2 eingeleitet, indem eine primäre Dokumentvektoren-Ergebnismenge $DVM_{prim}^{t=2}$ ermittelt werden muss. Hierzu wird wiederum zunächst die Menge aller Neurone $N_{DV,bew}^{t=2}$ ermittelt, denen Dokumentvektoren mit Relevanzschätzungen zugeordnet sind. Da in t=1 nicht notwendig jedem Neuron aus der Nachbarschaft $N(d_G=1 \mid G_{DV})_{s(1|i)}^{t=0}$ ein solcher Dokumentvektor zugeordnet wurde, muss $N_{DV,bew}^{t=2}$ nicht gleich $N(d_G\leq 1 \mid G_{DV})_{s(1|i)}^{t=0}$ sein. Die primäre Ergebnismenge $DVM_{prim}^{t=2}$ wird durch die Neurone aus $N_{DV,bew}^{t=2}$ und ihre Nachbarneurone korrigiert um die Dokumentvektoren, die zu $M_{DV(m)}^{t\leq 1}$ korrespondieren:

$$DVM_{prim}^{t=2} = \{\bigcup_k M_{DV,r}\} \setminus \{x_j \mid \forall m_j \in M_{DV(m)}^{t\leq 1}\},$$
$$\forall n_{DV,r} \in N(d_G\leq 1 \mid G_{DV})_k^t, \forall n_{DV,k} \in N_{DV,bew}^t. \quad (571)$$

Alle Dokumentvektoren aus der Primärmenge $DVM_{prim}^{t=2}$ werden durch $AM(rel(x))^{t=1}$ geschätzt. Es folgt die Bildung der sekundären und der tertiären Ergebnisliste, indem die Dokumentvektoren entsprechend ihrer fallenden Relevanzschätzung sortiert werden, und indem die Besten wieder ausgewählt werden.

4.3.2) Dokumentvektoren-GNG-SOM mit prototypbasiertem Modell

Im vorangegangenen Abschnitt wurde ein instanzbasiertes Approximationsmodell im Kontext einer GNG-SOM-klassifizierten Dokumentvektorenmenge verwendet. Die Verwendung eines prototypbasierten Approximationsmodells besitzt jedoch bei einer solchen Strukturierung entscheidende Vorteile und führt zu neuen Retrieval-Strategien, sodass diese Möglichkeiten im weiteren näher beschrieben werden sollen.

Ausgangspunkt ist das Netz N_{DV}, welches die Dokumentvektorenmenge DVM in disjunkte Teilmengen $M_{DV,j}$ zerlegt, sowie der Queryvektor $q_i^{t=0}$ der Initialisierungs-Iteration, der ein Gewinner-Neuron $n_{DV,s(1|i)}$ festlegt, in dessen Voronoi-Region $\tilde{R}_{DV,s(1|i)}$ er liegt. Damit wird die erste Dokumentvektoren-Ergebnismenge $DVM(q_i^{t=0}) := M_{DV,s(1|i)}^{t=0}$ festgelegt. Nach der Bewertung durch den Agenten wird die Stimulusmenge $M_{DV(m)}^{t=0}$ der bewerteten Dokumentvektoren aus der Initialisierungs-Iteration mit den

Stimuli $m_j^{t=0} = (x_j^{t=0}, rel(x_j^{t=0}))$, $\forall x_j^{t=0} \in DVM(q_i^{t=0})$, gebildet. Diese Stimulusmenge bildet zusammen z.B. mit einem LWR-Verfahren ein instanzbasiertes Approximationsmodell $AM(rel(x) \mid M_{DV(m)}^{t=0})$, mit den Relevanzschätzungen für Dokumentvektoren, aber auch für Prototypvektoren erzeugt werden können, worauf die Erzeugung eines prototypbasierten Approximationsmodells basiert.

Im Gegensatz zu den bislang dargestellen Verfahren mit prototypbasierten Approximationsmodellen in diesem Kapitel wird jedoch weder ein Netz auf der Basis einer Ergebnismenge oder einer bewerteten Ergebnismenge gebildet, sondern es wird das gegebene Netz N_{DV} verwendet, in dem ausgewählte Prototypen, d.h. Gewichtsvektoren, eine Relevanzschätzung erhalten. Modelle dieser Art sind von der Netzstruktur N_{DV} abhängig, und sollen mit $AM(rel(x) \mid N_{DV})$ bezeichnet werden, bzw. N_{DV} wird ersetzt durch eine Teilstruktur von Neuronen, denen eine Relevanzschätzung zugeordnet ist.

Die einfachste Strategie ist die Erzeugung einer Relevanzschätzung durch $AM(rel(x) \mid M_{DV(m)}^{t=0})$ an der Stelle des Gewichtsvektors des Gewinner-Neurons, sodass $AM(rel(x) \mid \{n_{DV,s(1|i)}^{t=0}\})$ erzeugt wird, doch dieses Modell hätte nur einen Stützpunkt, sodass die Einbeziehung seiner Nachbarn zu $AM(rel(x) \mid N(d_G=1 \mid G_{DV})_{s(1|i)}^{t=0})$ sinnvoller ist. Andererseits kann argumentiert werden, dass $n_{DV,s(1|i)}^{t=0}$ zu diesem Zeitpunkt das einzige Neuron ist, dem Dokumentvektoren mit Relevanzwerten des Agenten zugeordnet ist. Für spätere Iterationen $t > 0$ führt die Verwendung dieser Neuronenmenge zu einer Vereinfachung, da diese Menge mit $N_{DV,bew}^{t}$ bezeichnet wird, sodass ein prototypbasiertes Modell jeweils durch $AM(rel(x) \mid N_{DV,bew}^{t})$ beschrieben wird. Es soll vereinbart werden, dass nur die Neurone in das prototypbasierte Modell aufgenommen werden, die in $N_{DV,bew}^{t}$ enthalten sind, sodass $AM(rel(x))^{t=0} := AM(rel(x) \mid N_{DV,bew}^{t=0}) = AM(rel(x) \mid \{n_{DV,s(1|i)}^{t=0}\})$ gebildet wird, indem $n_{DV,s(1|i)}^{t=0}$ eine Relevanzschätzung durch $M_{DV}^{t=0}$ erhält:

$$rel(w(x_{DV})_{s(1|i)}^{t=0} \mid AM(rel(x) \mid M_{DV(m)}^{t=0})) = rel(w(x_{DV})_{s(1|i)}^{t=0})\wedge =$$
$$1/\nu_{s(1|i)} \sum_j h(d_{DVR}(x_j^{t=0}, w(x_{DV})_{s(1|i)}^{t=0})) * rel(x_j^{t=0}), \text{ mit}$$
$$\nu_{s(1|i)} = \sum_j h(d_{DVR}(x_j^{t=0}, w(x_{DV})_{s(1|i)}^{t=0})), \forall m_j^{t=0} \in M_{DV(m)}^{t=0}. \quad (572)$$

Das Gewinner-Neuron erhält somit eine erweiterte Neuronenstruktur:

$$n_{DV,s(1|i)}^{t=0} = (w(x_{DV})_{s(1|i)}^{t=0}, rel(w(x_{DV})_{s(1|i)}^{t=0})\wedge, M_{DV,s(1|i)}^{t=0}, C_{DV,s(1|i)}^{t=0}). \quad (573)$$

Nachdem das prototypbasierte Modell $AM(rel(x) \mid N_{DV,bew}^{t=0})$ erzeugt wurde, wird t=0 beendet und t=1 wird mit der Ermittlung der primären Ergebnismenge $DVM_{prim}^{t=1}$ eingeleitet. Hierzu soll wiederum die Menge $N_{DV,bew}^{t}$ der Neurone aus N_{DV} verwendet werden, denen ein Dokumentvektor mit einer Relevanzbewertung durch den Agenten zugeordnet wurde. Da diese Menge zu Beginn von t=1 ermittelt wird, so ergibt sich $N_{DV,bew}^{t=1} = \{n_{DV,s(1|i)}^{t=0}\}$. $DVM_{prim}^{t=1}$ wird bestimmt als Vereinigung der lokalen Dokumentvektorenmengen der Neurone aus $N_{DV,bew}^{t}$, sowie ihrer Nachbarn, korrigiert um die Dokumentvektoren, denen ein Relevanzwert zugeordnet wurde:

$$DVM_{prim}^{t=1} = \{\bigcup_k M_{DV,r}\} \setminus \{x_j \mid \forall m_j \in M_{DV(m)}^{t=0}\},$$
$$\forall n_{DV,r} \in N(d_G \leq 1 \mid G_{DV})_k^{t=1}, \forall n_{DV,k} \in N_{DV,bew}^{t=1} = \{n_{DV,s(1|i)}^{t=0}\}. \quad (574)$$

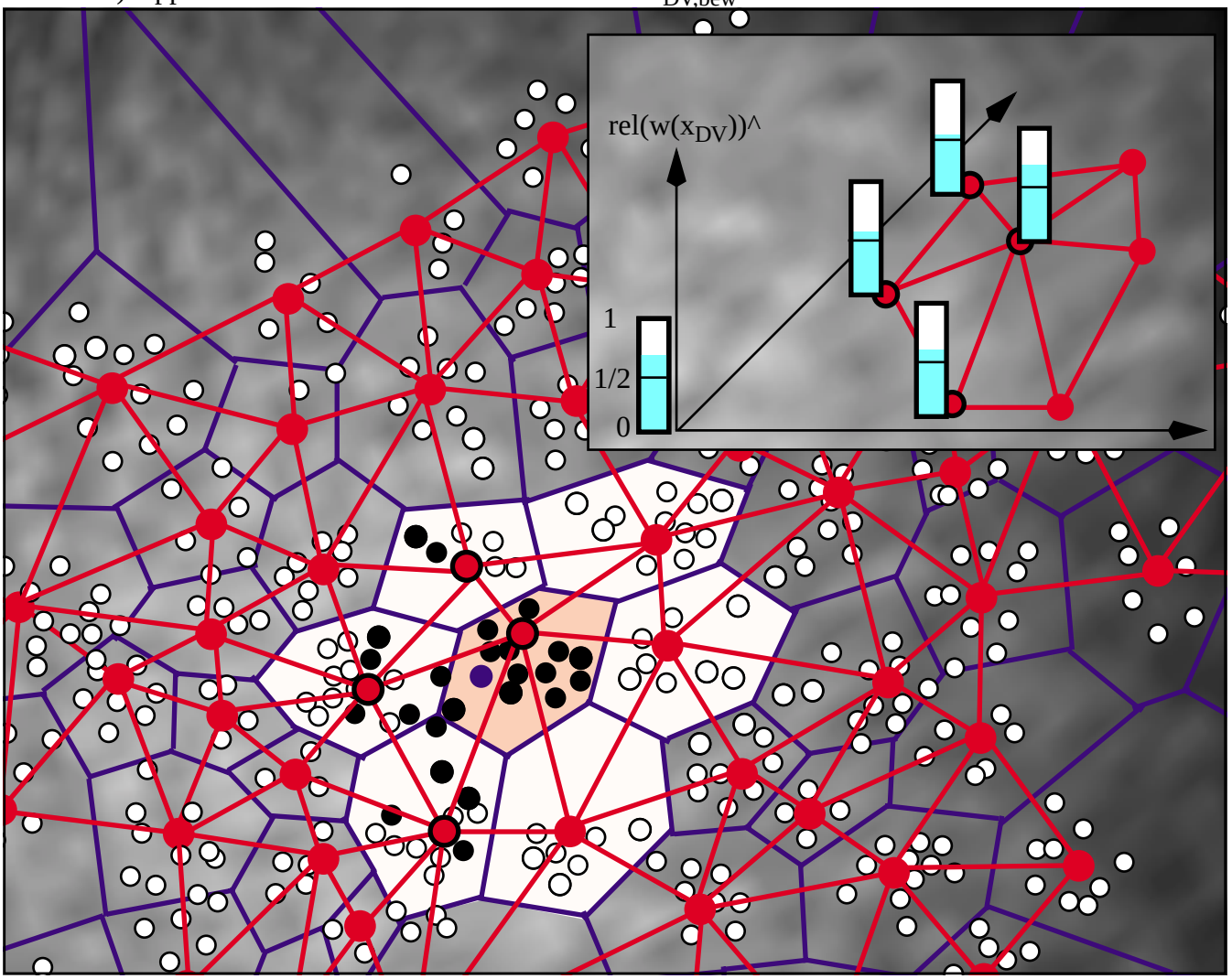
Für die Dokumentvektoren aus $DVM_{prim}^{t=1}$ wird eine Relevanzschätzung mit Hilfe AM(rel(x) | $N_{DV,bew}^{t=0}$) durchgeführt:

$$rel(x_j^{t=1} \mid AM(rel(x) \mid N_{DV,bew}^{t=0})) = rel(x_j^{t=1})\text{^}, \forall x_j^{t=1} \in DVM_{prim}^{t=1}. \quad (575)$$

Die sekundäre Ergebnisliste $DV_{sec}^{t=1}$ ergibt sich, indem die Elemente aus $DVM_{prim}^{t=1}$ entsprechend der fallenden Relevanzschätzung geordnet werden. Die tertiäre Ergebnisliste $DV_{ter}^{t=1}$ ergibt sich, indem die besten $\#M_{DV,s(1|i)}^{t=0} = \#DVM(q_i^{t=0})$ Dokumentvektoren, d.h. die Elemente auf den ersten Rängen der geordneten Liste $DV_{sec}^{t=1}$, ausgewählt werden. Die Dokumente, die zu den Elementen aus $DV_{ter}^{t=1}$ korrespondieren werden vom Agenten bewertet, sodass sich die Stimulusmenge $M_{DV(m)}^{t=1}$ ergibt, die vereinigt mit $M_{DV(m)}^{t=0}$ die Gesamtmenge $M_{DV(m)}^{t\leq 1}$ ergibt.

Es ist möglich, dass bestimmte Neurone aus der Nachbarschaft $N(d_G \leq 1 \mid G_{DV})_{s(1|i)}^{t=0}$ keine Dokumentvektoren aus der tertiären Liste $DV_{ter}^{t=1}$ besitzen, sodass ihnen auch keine Stimuli aus $M_{DV}^{t=1}$ zuordenbar sind. Abb. 120) zeigt den Fall, dass neben dem Gewinner-Neuron drei Neurone aus $N(d_G \leq 1 \mid G_{DV})_{s(1|i)}^{t=0}$ Dokumentvektoren besitzen, denen zunächst so gute Relevanzschätzungen zugeordnet wurden, dass sie in $DV_{ter}^{t=1}$ aufgenommen wurden, sodass sie nach der Bewertung der korrespondierenden Dokumente auch Relevanzwerte des Agenten besitzen.

Abb. 120) Approximationsmodell durch Neurone aus $N_{DV,bew}^{t}$ in t=1



Mit Hilfe der bewerteten Stimuli aus $M_{DV(m)}^{t\leq 1}$ liegt das instanzbasierte Approximationsmodell $AM(rel(x) \mid M_{DV(m)}^{t\leq 1})$ vor, mit dem ein aktualisiertes, prototypbasiertes Approximationsmodell erzeugt wird. Hierzu werden allen Neuronen, denen bis zu diesem Zeitpunkt ein Dokumentvektor mit einem Relevanzwert des Agenten zugeordnet wurde, eine Relevanzschätzung mit $AM(rel(x) \mid M_{DV(m)}^{t\leq 1})$ zugeordnet. Die Menge aller Neurone mit Elementen aus $M_{DV(m)}^{t\leq 1}$ ist jedoch gleich $N_{DV,bew}^{t=2}$, sodass sich als aktualisiertes Modell $AM(rel(x))^{t=1} := AM(rel(x) \mid N_{DV,bew}^{t=2})$ ergibt.

Erfolgte kein Abbruch von Seiten des Agenten, so wird nach der Spezifizierung von $AM(rel(x))^{t=1}$ die nächste Iteration t=2 eingeleitet, indem eine primäre Dokumentvektoren-Ergebnismenge $DVM_{prim}^{t=2}$ ermittelt wird. Hierzu wird wiederum die Menge $N_{DV,bew}^{t=2}$ verwendet, sodass sich die primäre Ergebnismenge $DVM_{prim}^{t=2}$ durch die Neurone aus $N_{DV,bew}^{t=2}$ und ihre Nachbarneurone korrigiert um die zu $M_{DV}^{t\leq 1}$ korrespondierenden Dokumentvektoren ergibt:

$$DVM_{prim}^{t=2} = \{\bigcup_k M_{DV,r}\} \setminus \{x_j \mid \forall m_j \in M_{DV(m)}^{t\leq 1}\},$$
$$\forall n_{DV,r} \in N(d_G \leq 1 \mid G_{DV})_k^t, \forall n_{DV,k} \in N_{DV,bew}^t. \quad (576)$$

Alle Dokumentvektoren aus der Primärmenge $DVM_{prim}^{t=2}$ werden durch $AM(rel(x))^{t=1} := AM(rel(x) \mid N_{DV,bew}^{t=2})$ geschätzt. Es folgt die Bildung der sekundären und der tertiären Ergebnisliste, deren korrespondierende Dokumente dem Agenten zur Bewertung präsentiert werden.

4.3.3) Dokumentvektoren-GNG-SOM mit Nachadaption

In diesem Abschnitt soll der Fall betrachtet werden, dass eine Dokumentvektoren-Klassifikation durch eine GNG-SOM N_{DV} vorliegt, die durch Stimuli lokal nachadaptiert wird, welche durch die Relevanzbewertungen des Agenten gebildet werden. D.h. es werden nicht mehr die Stützpunkte im DVR verwendet, die durch die Gewichtsvektoren der Neurone von N_{DV} vorgegeben sind, sondern es werden durch Nachadaptionsoperationen eigene Stützpunkte erzeugt, wobei die folgenden Vorgehensweisen zu unterscheiden sind:

1) Nachadaption ohne Wachstumsoperationen, d.h. mit Erhaltung der Neuronenanzahl.
2) Nachadaption mit Wachstumsoperationen, d.h. mit Erhöhung der Neuronenanzahl.

Ausgangspunkt für diese beiden Verfahren ist die globale, nicht individuelle GNG-SOM N_{DV}, die ausschließlich Dokumentvektoren als Stützpunkte im DVR besitzt:

$$N_{DV} = \{n_{DV,j} = (w(x_{DV})_j, M_{DV,j}, C_{DV,j}) \mid j = 1, ..., \mu_{N,DV}\}. \quad (577)$$

Durch den Queryvektor $q_i^{t=0}$ wird in der Initialisierungs-Iteration t=0 das Gewinner-Neuron $n_{DV,s(1|i)}^{t=0}$ aus N_{DV} bestimmt, in dessen Voronoi-Region $\tilde{R}_{DV,s(1|i)} q_i^{t=0}$ liegt. Standardmäßig wird bei einer GNG-SOM-klassifizierten Dokumentvektorenmenge die primäre Ergebnismenge $DVM_{prim}^{t=0} := DVM(q_i^{t=0})$ durch die Elemente aus $M_{DV,s(1|i)}^{t=0}$ gebildet, was durch die Bewertung durch den Agenten zu einer Stimulusmenge $M_{DV(m)}^{t=0}$ führt.

4.3.3.1) Nachadaption ohne Wachstumsoperationen

Bei der Nachadaption ohne Wachstumsoperationen werden die Dokumentvektor-Komponenten der Stimuli aus $M_{DV(m)}^{t=0}$ verwendet, um das individuelle GNG-SOM $N_{DV}^{t=0}$ aus der nicht individuellen GNG-SOM N_{DV} abzuleiten, indem Gewichtsvektoren adaptiert werden. Es wird ein zusätzlicher Index τ für die Adaption eingeführt, d.h. in τ =1 wird aus $M_{DV(m)}^{t=0}$ ein Element $m_j^{t=0} = (x_j^{t=0}, rel(x_j^{t=0}))$ zufällig mit Zurücklegen gezogen, wobei der Stimulus $m_j^{t=0}$ das Gewinner-Neuron $n_{DV,s(1|j)}^{t=0,\tau=1}$ identifiziert, dessen Gewichtsvektor $w(x_{DV})_{s(1|j)}^{t=0,\tau=1}$ einer Gewinner-Adaption mit dem Adaptionsparameter ε_s unterzogen wird. Die Gewichtsvektoren der Neurone aus der Nachbarschaftsmenge $N(d_G=1 \mid G_{DV}^{t=0,\tau=1})_{s(1|j)}$ werden nachfolgend einer Nachbarschafts-Adaption mit ε_n unterzogen, wobei $x_j^{t=0}$ jeweils Fixpunkt der positiven Adaption ist. Nach τ(end) zufälligen Ziehungen mit Zurücklegen und Adaptionsoperationen erfolgt ein Abbruch, wobei τ(end) als Funktion der Anzahl der Elemente aus $M_{DV(m)}^{t=0}$ ermittelt wird. Danach wird allen Neuronen, denen zu diesem Zeitpunkt mindestens ein Dokumentvektor mit einer Relevanzbewertung zugeordnet wurde, eine Relevanzschätzung auf der Basis aller verfügbaren Stimuli zugeordnet. Diese Neuronenmenge ist gleich $N_{DV,bew}^{t=1} = \{n_{DV,s(1|i)}^{t=0}\}$, wobei beachtet werden muss, dass noch die Iteration t=0 aktuell ist. Das Gewinner-Neuron erweitert somit seine Datenstruktur, indem eine Komponente für die Relevanzschätzung hinzu kommt.

Nach der Festlegung von $N_{DV}^{t=0}$ und somit dem korrespondierenden, prototypbasierten Approximationsmodell $AM(rel(x) \mid N_{DV}^{t=0})$ wird die Iteration t=0 beendet und t=1 wird begonnen, indem die primäre Ergebnismenge $DVM_{prim}^{t=1}$ ermittelt wird. Hierzu wird zunächst die Neuronenmenge $N_{DV,bew}^{t=1}$ mit $\{n_{DV,s(1|i)}^{t=0}\}$ verwendet, was gilt, wenn keine Wachstumsprozesse bei der Ableitung von $N_{DV}^{t=0}$ aus N_{DV} durchgeführt wurden. Die primäre Ergebnismenge $DVM_{prim}^{t=1}$ wird als Vereinigung der lokalen Dokumentvektorenmengen der Neurone aus $N_{DV,bew}^{t=1}$ und ihrer Nachbarneurone korrigiert um die Dokumentvektoren, die zu $M_{DV(m)}^{t=0}$ korrespondieren, festgelegt:

$$DVM_{prim}^{t=1} = \{\bigcup_k M_{DV,r}\} \setminus \{x_j \mid \forall m_j \in M_{DV(m)}^{t=0}\},$$
$$\forall n_{DV,r} \in N(d_G \leq 1 \mid G_{DV})_k, \forall n_{DV,k} \in N_{DV,bew}^{t=1},$$
$$\text{d.h. } \forall n_{DV,r} \in N(d_G \leq 1 \mid G_{DV})_{s(1|i)}. \quad (578)$$

Jedem Dokumentvektor $x_j^{t=1} \in DVM_{prim}^{t=1}$ wird eine Relevanzschätzung $rel(x_j^{t=1} \mid AM(rel(x) \mid N_{DV}^{t=0}))$ zugeordnet, und die sekundäre Ergebnisliste $DV_{sec}^{t=1}$ wird durch das Sortieren der Dokumentvektoren aus $DVM_{prim}^{t=1}$ entsprechend fallender Relevanzschätzungen erzeugt. Aus dieser Liste werden die ersten $\#M_{DV}^{t=0}$ Elemente ausgewählt, welche die tertiäre Liste $DV_{ter}^{t=1}$ bilden, und die korrespondierenden Dokumente werden dem Agenten in der entsprechenden Reihenfolge präsentiert. Die bewerteten Dokumentvektoren ergeben $M_{DV(m)}^{t=1}$, die vereinigt mit $M_{DV(m)}^{t=0}$ die Menge $M_{DV(m)}^{t\leq 1}$ ergibt.

Mit Hilfe von $M_{DV(m)}^{t=1}$ wird das Modell $N_{DV}^{t=0}$ zu $N_{DV}^{t=1}$ aktualisiert, indem Adaptionsoperationen wieder unter Verwendung des zusätzlichen Indexes τ durchgeführt werden, d.h. in τ =1 wird aus $M_{DV(m)}^{t=1}$ ein Element $m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1}))$ zufällig mit Zurücklegen gezogen, wobei der Stimulus $m_j^{t=1}$ das Gewinner-Neuron $n_{DV,s(1|j)}^{t=1,\tau=1}$ identifiziert, dessen Gewichtsvektor $w(x_{DV})_{s(1|j)}^{t=1,\tau=1}$ einer Gewinner-Adaption mit dem Adaptionsparameter ε_s unterzogen wird. Die Gewichtsvektoren der Neurone aus der Nachbarschaftsmenge $N(d_G=1 \mid G_{DV}^{t=1,\tau=1})_{s(1|j)}$ werden nachfolgend einer Nachbarschafts-

Adaption mit ε_n unterzogen, wobei $x_j^{t=1}$ jeweils Fixpunkt der positiven Adaption ist. Nach τ(end) zufälligen Ziehungen mit Zurücklegen und Adaptionsoperationen erfolgt ein Abbruch.

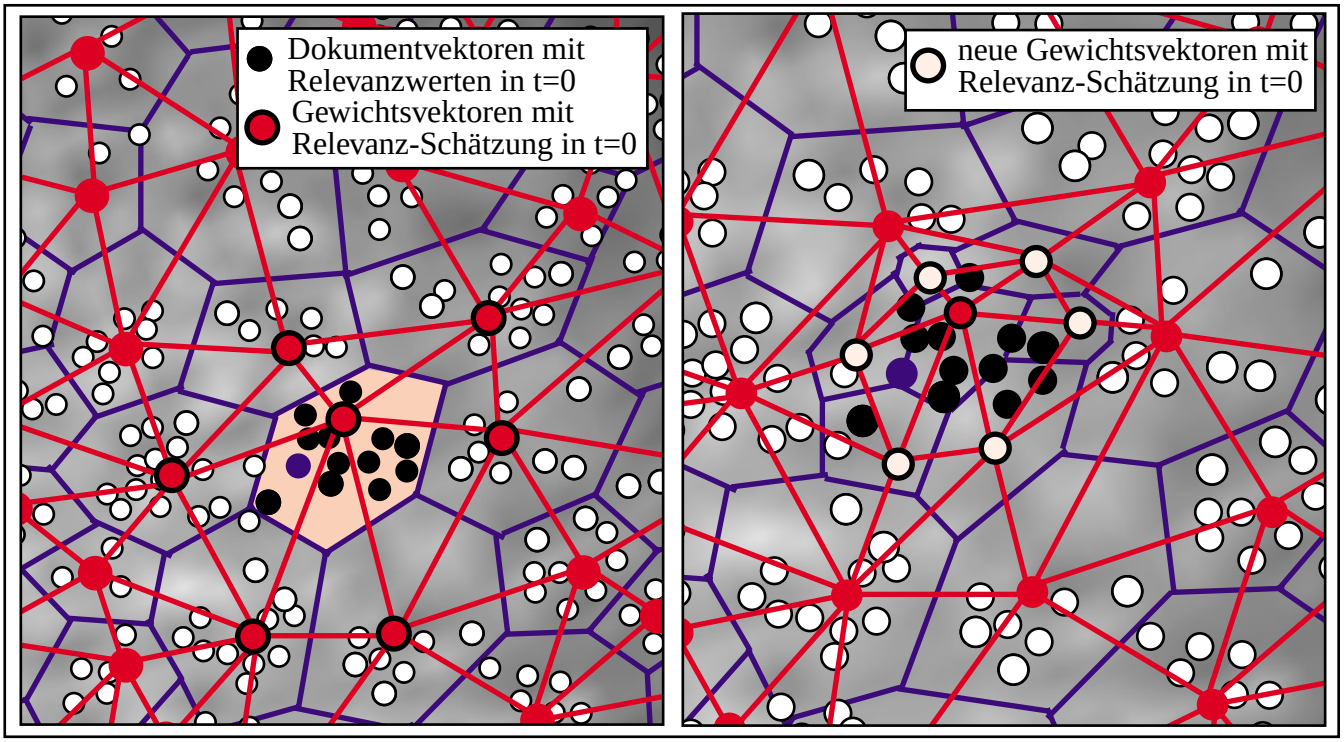
Allen Neuronen, denen mindestens ein Dokumentvektor mit Relevanzbewertung zugeordnet ist, wird daraufhin eine Relevanzschätzung auf der Basis ihres adaptierten Gewichtsvektors und allen Elementen aus $M_{DV(m)}^{t\leq1}$ zugeordnet, d.h. auf der Basis des instanzbasierten Modells AM(rel(x) | $M_{DV(m)}^{t\leq1}$). Die Menge der Neurone, welche diese Eigenschaft besitzt, ist gleich $N_{DV,bew}^{t=2}$. Das sich ergebende Modell $N_{DV}^{t=1,\tau=\tau(end)}$ wird als $N_{DV}^{t=1}$ übernommen, mit dem ein prototypbasiertes Approximationsmodell AM(rel(x) | $N_{DV}^{t=1}$) festgelegt ist. Die Iteration t=1 wird danach beendet, und t=2 begonnen, indem die primäre Ergebnismenge $DVM_{prim}^{t=2}$ ermittelt wird, deren Dokumentvektoren mit Hilfe von AM(rel(x) | $N_{DV}^{t=1}$) bewertet werden.

4.3.3.2) Nachadaption mit Wachstumsoperationen

Diese Vorgehensweise unterscheidet sich von der vorangegangenen dadurch, dass in allen Adaptionsphasen Wachstumsprozesse hinzu kommen, d.h. bei der Ableitung von $N_{DV}^{t=0}$ aus $N_{DV,}$ sowie bei allen Aktualisierungen von N_{DV}^{t} zu N_{DV}^{t+1}.

Es stellt sich die Frage, ob bereits bei der Ableitung von $N_{DV}^{t=0}$ aus N_{DV} Wachstumsoperationen mit Hilfe von $M_{DV(m)}^{t=0}$ durchgeführt werden sollen. Gegen die Anwendung des Wachstumsprozesses zu diesem frühen Zeitpunkt spricht, dass alle Dokumentvektoren, die dem Gewinner-Neuron zugeordnet sind, eine Bewertung erhalten haben. Wachstumsprozesse würden neue Gewichtsvektoren am Rand der Voronoi-Region $\tilde{R}_{DV,s(1|i)}^{t=0}$ erzeugen, die durch positive Adaptionen in Richtung $w(x_{DV})_{s(1|i)}^{t=0}$ wandern. Der Gewichtsvektor $w(x_{DV})_{s(1|i)}^{t=0}$ kann als eine Art Zentrum der Voronoi-Region $\tilde{R}_{DV,s(1|i)}^{t=0}$ interpretiert werden, bzw. als Zentrum der Region, welche durch die Dokumentvektoren in $M_{DV,s(1|i)}^{t=0}$ belegt wird. Entsprechend den Constraints, die beim Aufbau von N_{DV} verwendet werden, ist der Gewichtsvektor $w(x_{DV})_{s(1|i)}^{t=0}$ ein Repräsentant der Dokumentvektoren aus $M_{DV,s(1|i)}^{t=0}$, da ein lokales Fehlermaß, wie der Quantifizierungsfehler, auf der Basis der lokalen Stimulusmenge solange durch Wachstums- und Adaptionsoperationen verringert wird, bis er unter einen Schwellenwert fällt, was als Abbruchkriterium der Wachstums- und Adaptionsoperationen verwendet wird. Weitere Wachstumsoperationen erhöhen die lokale Gewichtsvektorendichte, und führen zu einer weiteren Verringerung des Fehlerwertes, sodass entweder ein zweiter, kleinerer Fehlerschwellenwert für den Abbruch angegeben werden muss, oder es muss eine Anzahl von Neuronen spezifiziert werden, die eingefügt werden sollen. Einfacher wäre die zweite Möglichkeit, indem auf der Mitte der Verbindungskante zwischen $w(x_{DV})_{s(1|i)}^{t=0}$ und jedem Nachbarn aus $N(d_G=1 \mid G_{DV+}^{t=0})_{s(1|i)}$ ein neuer Gewichtsvektor erzeugt wird, sodass die Anzahl der neuen Neurone immer gleich $\#N(d_G=1 \mid G_{DV+}^{t=0})_{s(1|i)}$ wäre (siehe Abb. 121)). Es wird die Bezeichnung $G_{DV+}^{t=0}$ verwendet, da sich die Verbindungsstruktur im Verlauf der Iterationen durch das Neuronenwachstum im Bezug zu G_{DV} verändert. Dargestellt wird die Situation direkt nach dem gleichzeitigen Einfügen aller neuen Gewichtsvektoren, ohne dass Adaptionsprozesse durchgeführt wurden.

Abb. 121) Einfügen von sechs neuen Gewichtsvektoren in t=0



Nach dem Einfügen kann sich der Fall ergeben, dass einzelne Neurone keinen Dokumentvektor in ihrer Voronoi-Region besitzen, oder dass der Queryvektor $q_i^{t=0}$ in einer anderen Voronoi-Region liegt, da sich die Voronoi-Region des ursprünglichen Gewinner-Neurons verkleinert, doch diese Spezialfälle sollen nicht weiter verfolgt werden.

Als Alternative soll der Fall betrachtet werden, dass in t=0 auf Wachstumsoperationen verzichtet wird, und nur Adaptionsoperationen durchgeführt werden, sodass aus N_{DV} das individuelle Modell $N_{DV+}^{t=0}$ mit der gleichen Anzahl von Neuronen, jedoch veränderten Gewichtsvektoren aller Neurone ausder Nachbarschaft $N(d_G \leq 1 \mid G_{DV+}^{t=0})_{s(1|i)}$ abgeleitet wird. Zu dieser GNG-SOM korrespondiert ein prototypbasiertes Approximationsmodell $AM(rel(x) \mid N_{DV+}^{t=0})$, womit mit t=0 beendet und t=1 begonnen werden kann.

In t=1 wird zunächst die primäre Ergebnismenge $DVM_{prim}^{t=1}$ ermittelt, indem die Neurone aus $N_{DV,bew}^{t=1} = \{n_{DV,s(1|i)}^{t=0}\}$ und ihre Nachbarn die Dokumentvektorenmengen vereinigen, korrigiert um die Dokumentvektoren, die zu $M_{DV(m)}^{t=0}$ korrespondieren. Jedem Dokumentvektor $x_j^{t=1} \in DVM_{prim}^{t=1}$ wird eine Relevanzschätzung $rel(x_j^{t=1} \mid AM(rel(x) \mid N_{DV+}^{t=0}))$ zugeordnet, und die sekundäre Ergebnisliste $DV_{sec}^{t=1}$ wird durch das Sortieren der Dokumentvektoren aus $DVM_{prim}^{t=1}$ entsprechend fallender Relevanzschätzungen erzeugt. Aus dieser Liste werden die ersten $\#M_{DV(m)}^{t=0}$ Elemente ausgewählt, welche die tertiäre Liste $DV_{ter}^{t=1}$ bilden. Die bewerteten Dokumentvektoren ergeben die Stimulusmenge $M_{DV(m)}^{t=1}$, die vereinigt mit $M_{DV(m)}^{t=0}$ die Menge $M_{DV(m)}^{t\leq 1}$ ergibt.

Mit Hilfe von $M_{DV(m)}^{t=1}$ wird das Modell $N_{DV+}^{t=0}$ zu $N_{DV+}^{t=1}$ aktualisiert, wobei hier Adaptions- und Wachstumsoperationen durchgeführt werden. Auf die Möglichkeit der Löschung von Neuronen, insbesondere solcher Neurone, in deren Voronoi-Region kein Dokumentvektor liegt, soll hier nicht explizit

eingegangen werden. Notwendig ist wiederum ein zusätzlicher Index τ, wobei der Standardfall einer GNG-SOM angenommen wird (siehe Abschnitt 2.1.5)), dass nach einer konstanten Anzahl von $\tau = \lambda(1)$ Adaptionsoperationen eine Wachstumsphase mit der Erzeugung genau eines neuen Neurons eingeschoben wird.

Bei der Aktualisierung zu $N_{DV+}^{t=1}$ wird in der Aktualisierungs-Iteration τ=1 somit ein Element $m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1}))$ aus der Grundmenge zufällig mit Zurücklegen gezogen, wobei hier im Gegensatz zu einer Aktualisierung ohne Wachstumsoperationen die Grundmenge $M_{DV(m)}^{t\leq1}$ anstatt $M_{DV(m)}^{t=1}$ verwendet werden soll. Der Stimulus $m_j^{t=1}$ identifiziert das Gewinner-Neuron $n_{DV,s(1|j)}^{t=1,\tau=1}$, dessen Gewichtsvektor $w(x_{DV})_{s(1|j)}^{t=1,\tau=1}$ einer Gewinner-Adaption mit dem Adaptionsparameter ε_s unterzogen wird. Die Gewichtsvektoren der Neurone aus der Nachbarschaftsmenge $N(d_G=1 \mid G_{DV+}^{t=1,\tau=1})_{s(1|j)}$ werden nachfolgend einer Nachbarschafts-Adaption mit ε_n unterzogen, wobei $x_j^{t=1}$ jeweils Fixpunkt der positiven Adaption ist.

Nach $\lambda(1)$ dieser Adaptionsoperationen wird der Zwischenzustand $N_{DV+}^{t=1,\tau=\lambda(1)}$ für eine Wachstums-Operation verwendet, indem zunächst das primäre Einfügezentrum bestimmt wird, als das Neuron mit einem maximalen, lokalen Fehlermaß. Als Grundmenge soll jedoch nicht die ganze Struktur $N_{DV+}^{t=1,\tau=\lambda(1)}$, sondern nur die Neurone ausgewählt werden, die mindestens einen Dokumentvektor mit einem Relevanzwert besitzen, d.h. es wird allgemein $N_{DV,bew}^{t}$ bzw. $N_{DV,bew}^{t,\tau}$ betrachtet. Die Aktualisierung zu $N_{DV+}^{t=1}$ wurde bislang als letzte Operation innerhalb einer Iteration betrachtet, sodass entsprechend dem Index t die Menge $N_{DV,bew}^{t=1,\tau}$ betrachtet werden müsste, doch dies wird durch die Einführung der Wachstumsoperationen innerhalb der Aktualisierung hinfällig, da jetzt nicht mehr $N_{DV,bew}^{t=1} = \{n_{DV,s(1|i)}^{t=0}\}$ sondern die Erweiterung nach Kenntnis der Stimulusmenge $M_{DV(m)}^{t=1}$ benötigt wird. Die Aktualisierung könnte in die nachfolgende Iteration t+1 verschoben werden, oder es wird nach der Erzeugung von $M_{DV(m)}^{t=1}$ und somit $M_{DV(m)}^{t\leq1}$ eine Aktualisierung der Menge der Neurone mit Relevanzwerten durchgeführt. Die zweite Alternative soll im weiteren verwendet werden, indem die Initialisierung der Adaptions- und Wachstumsoperationen mit τ=1, d.h. $N_{DV,bew}^{t=1,\tau=1}$, auf der Basis von $M_{DV(m)}^{t\leq1}$ erstellt wird.

Für die Wachstumsoperation in $\tau = \lambda(1)$ bedeutet dies, dass aus $N_{DV,bew}^{t=1,\tau=\lambda(1)}$ das Neuron $n_{DV,f(1)}^{t=1,\tau=\lambda(1)}$ mit dem größten lokalen Fehlerwert ausgewählt, und als primäres Einfügezentrum verwendet wird. Als Fehlermaß könnte der Quantifizierungsfehler der Dokumentvektoren verwendet werden, denen ein Relevanzwert zugeordnet wurde. Dies bedeutet jedoch, dass nur den Neuronen aus $N_{DV,bew}^{t=1,\tau}$ ein Fehlermaß zugeordnet werden kann, was sich auf die Auswahl des sekundären Einfügezentrums $n_{DV,f(2)}^{t=1,\tau=\lambda(1)}$ auswirken wird, wenn dieses ebenfalls durch einen maximalen, lokalen Fehlerwert aus der Nachbarschaft $N(d_G=1 \mid G_{DV+}^{t=1,\tau=\lambda(1)})_{f(1)}$ des primären Einfügezentrums spezifiziert werden soll. Aus $N(d_G=1 \mid G_{DV+}^{t=1,\tau=\lambda(1)})_{f(1)}$ kommen dann nur solche Neurone in Frage, die ebenfalls Element von $N_{DV,bew}^{t=1,\tau=\lambda(1)}$ sind. Sollen alle Elemente aus $N(d_G=1 \mid G_{DV+}^{t=1,\tau=\lambda(1)})_{f(1)}$ potentiell als sekundäres Einfügezentrum auswählbar sein, so muss das Selektionskriterium verändert werden, indem z.B. als Fehlermaß ein Quantifizierungsfehler auf der Basis aller Dokumentvektoren in den lokalen Mengen der Neurone zugelassen wird, oder indem einfach eine Zufallsauswahl aus $N(d_G=1 \mid G_{DV+}^{t=1,\tau=\lambda(1)})_{f(1)}$ getroffen wird.

Nachdem die Einfügezentren $n_{DV,f(1)}^{t=1,\tau=\lambda(1)}$ und $n_{DV,f(2)}^{t=1,\tau=\lambda(1)}$ festgelegt wurden, wird ein neues Neuron $n_{DV,r}^{t=1,\tau=\lambda(1)}$ initialisiert, indem in der Mitte der Verbindungskante zwischen den beiden Gewichtsvektoren der Einfügezentren der Gewichtsvektor des neuen Neurons erzeugt wird. Es folgt die Korrektur der Verbindungsstruktur zwischen dem neuen Neuron, den Einfügezentren und ihren verbundenen Nachbarn, sowie die lokale Umverteilung von Dokumentvektoren. Dabei werden alle Dokumentvektoren umverteilt, unabhängig ob ihnen ein Relevanzwert zugeordnet wurde oder nicht.

Mit dieser Umverteilung ist die Wachstumsphase abgeschlossen und es werden in der Folge weitere $\lambda(1)$ Adaptionsoperationen durchgeführt, wenn kein Abbruchkriterium greift. Dieses kann entweder performanceabhängig sein, was die Definition eines lokalen Fehlerschwellenwertes erfordert, bzw.. einer monoton fallenden Fehlerschwellenwert-Funktion, da bei jeder Iteration t bzw. ein Vielfaches von $\lambda(1)$ ein kleinerer Fehlerschwellenwert sinnvoll ist, da sich die Gewichtsvektorendichte lokal erhöht. Das Abbruchkriterium kann ebenso durch eine maximale Anzahl $\tau(end)$ von Adaptionen festgelegt werden, die analog wie bei der Aktualisierung ohne Wachstumsoperationen als Funktion der Anzahl der Elemente in $M_{DV(m)}^{t=1}$ oder $M_{DV(m)}^{t\leq 1}$ bestimmt wird.

Nach dem Abbruch der Adaptions- und Wachstumsprozesse in t=1 wird die sich ergebende Struktur $N_{DV+}^{t=1,\tau=\tau(end)}$ als $N_{DV+}^{t=1}$ übernommen. Es folgt die Relevanzschätzung für alle Neurone, denen ein Dokumentvektor mit Relevanzwert zugeordnet ist, d.h. für alle Elemente aus $N_{DV,bew}^{t=1,\tau=\tau(end)}$. Die Relevanzschätzungen werden jeweils instanzbasiert mit Hilfe aller Stimuli aus $M_{DV(m)}^{t\leq 1}$ durchgeführt, d.h. mit $AM(rel(x) \mid M_{DV(m)}^{t\leq 1})$. Danach ist die GNG-SOM $N_{DV+}^{t=1}$ und somit das prototypbasierte Approximationsmodell $AM(rel(x) \mid N_{DV+}^{t=1})$ komplett, und es wird die nachfolgende Iteration t= 2 begonnen, indem die primäre Ergebnismenge $DVM_{prim}^{t=2}$ ermittelt wird, deren Dokumentvektoren mit Hilfe von $AM(rel(x) \mid N_{DV+}^{t=1})$ bewertet werden.

4.4) Relevanz-Approximationsmodell-Polyrepräsentation

Die Auswirkungen reeller Relevanzbewertungen auf die Adaption der Queryvektoren bei einer Queryvektor-Polyrepräsentation wurde im Abschnitt 4.2.1.2) beschrieben. Wird eine Retrievalstrategie verwendet, die ausschließlich auf der Relevanzschätzung von noch nicht durch den Agenten bewerteter Dokumentvektoren beruht, so sind die Auswirkungen einer Queryvektor-Polyrepräsentation vernachlässigbar im Vergleich zu einer Queryvektor-Monorepräsentation, da die Polyrepräsentation nur in der Initialisierungs-Iteration t=0 eine Rolle bei der Ermittlung der Dokumentvektor-Ergebnismenge spielt. In den weiteren Iterationen t werden die Ergebnismengen durch Relevanzschätzungen durch das jeweilige Approximationsmodell $AM(rel(x))^{t-1}$ bestimmt, sodass des unerheblich wird, ob eine Queryvektor-Mono- oder -Polyrepräsentation vorliegt.

Demgegenüber eröffnet eine Approximationsmodell-Polyrepräsentation ein weites Feld für Verfahren von Combining Models (Wolpert (1990[365], 1993[367]), Freund (1995[124]), Freund & Schapire (1995[125], 1996[126]), Quinlan (1993[267], 1996[268]), Skalak (1996[313]), Merz (1998[216]), Gama (1999[141]), Witten & Ting (1999[364])), aus denen Effektivitätsverbesserungen erzeugt werden, die dazu führen, dass die Fehler der Relevanzschätzungen des IRS im Vergleich zu den Relevanzbewertun-

gen des Agenten kleiner werden. Damit erhöht sich die Wahrscheinlichkeit, die Dokumentvektoren in den Ergebnismengen zu sammeln, die von dem Agenten als hinreichend relevant bewertet werden, sodass erwartet werden kann, dass ein Relevanz-Feedback weniger Iterationen benötigten wird, um zu einem Abbruch von Seiten des Agenten zu gelangen. Unter diesem Aspekt könnte eine Approximationsmodell-Polyrepräsentation auch eine Effizienzverbesserung herbei führen, indem der höhere Aufwand innerhalb einer Iteration überkompensiert wird durch die geringere Anzahl der Iterationen d.h. allgemein durch den Gesamtaufwand über alle Iterationen.

Eine Approximationsmodell-Polyrepräsentation ist der Zwischenschritt zu der Anwendung aktiver Lernverfahren im Kontext von Relevanz-Feedback und Relevanz-Approximation (siehe Kapitel 5)). Es können damit fortgeschrittene Verfahren zum Modellaufbau angewendet werden, die nicht nur auf einfachen Relevanzschätzungen basieren, sondern ganz oder zusätzlich auf Fehlermaßen allgemein bzw. Momenten r‘ter Ordnung dieser Relevanzschätzungen (siehe Abschnitt 2.3.7)).

4.4.1) Polyrepräsentation bei instanzbasierten Approximationsmodellen

Instanzbasierte Approximationsmodelle bestehen aus einer Menge von Stimuli, d.h. aus Stützpunkten, von denen der Outputvektor bekannt ist, sowie aus einem Approximationsverfahren, das die Input- und Outputvektoren nutzt, um eine Outputschätzung an der Stelle eines Inputvektors zu erzeugen. Eine Polyrepräsentation kann somit durch die folgenden Faktoren erzeugt werden:

1) Stimulus-Polyrepräsentation und Approximationsverfahren-Monorepräsentation, d.h. es werden mehrere Stimulusmengen aufgebaut bei einem Approximationsverfahren.
2) Stimulus-Monorepräsentation und Approximationsverfahren-Polyrepräsentation, d.h. es wird eine Stimulusmenge in Verbindung mit mehreren Approximationsverfahren verwendet.
3) Stimulus-Polyrepräsentation und Approximationsverfahren-Polyrepräsentation, d.h. es werden mehrere Stimulusmengen erzeugt, auf die jeweils mehrere Approximationsverfahren zugreifen.

Diese Unterscheidung entspricht der Analogie bei der Queryvektor-Polyrepräsentation, die durch die Faktoren Query und Indexierungsfunktion erzeugt werden kann, wobei die Queries den Stimuli und das Approximationsverfahren der Indexierungsfunktion entspricht.

Im weiteren soll sich darauf beschränkt werden, die Polyrepräsentation durch eine Stimulus-Polyrepräsentation, d.h. durch multiple Stimulusmengen, in Verbindung mit einem Approximationsverfahren bei den instanzbasierten sowie bei den nachfolgenden prototypbasierten Modellen darzustellen.

4.4.1.1) Approximationsmodell-Polyrepräsentation durch Queryvektor-Polyrepräsentation

Multiple Ergebnismengen werden automatisch erzeugt, wenn eine Queryvektor-Polyrepräsentation mit $QVM^{t=0} = \{q_{ik}{}^{t=0} \mid k = 1, ..., \delta\}$ vorliegt, indem jeder Queryvektor eine Umgebung $U(q_{ik}{}^{t=0} \mid \varepsilon)$ festlegt, in der die darin enthaltenen Dokumentvektoren in eine Ergebnismenge $DVM(q_{ik}{}^{t=0})$ aufgenommen werden. Wird die Vereinigungsmenge $DVM(QVM^{t=0})$ gebildet und die korrespondierenden Dokumente dem

Agenten präsentiert, so ergeben sich Stimulusmengen $M_{DV(m),k}^{t=0}$, die zu den Ergebnismengen $DVM(q_{ik}^{t=0})$ korrespondieren. Zusammen mit einem Approximationsverfahren wie der Kernel-Regression, bildet jede dieser Stimulusmengen ein eigenes, instanzbasiertes Relevanz-Approximationsmodell $AM(rel(x) \mid M_{DV(m),k}^{t=0})$. Jedes dieser Modelle kann in der nachfolgenden Iteration t=1 verwendet werden, um Dokumentvektoren aus einer primären Ergebnismenge zu bewerten. Es soll der Fall dargestellt werden, dass die Elemente der jeweiligen Restmenge wie $DVM_{prim}^{t=1} = DV \setminus DVM(QVM^{t=0})$ bewertet werden, d.h. alle Dokumentvektoren, die bislang keine Bewertung durch den Agenten besitzen, ohne dass die Effizienzproblematik hierbei nochmals beschrieben werden soll (siehe Abschnitt 4.2.3)). D.h. für jeden Dokumentvektor $x_j^{t=1}$ aus $DV \setminus DVM(QVM^{t=0})$ können δ Relevanzschätzungen gebildet werden, die in die vorläufige Datenstruktur des Stimulus $m_j^{t=1}$ aufgenommen werden können:

$$m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1} \mid AM(rel(x) \mid M_{DV(m),k}^{t=0})) \mid k = 1, ..., \delta). \quad (579)$$

Damit aus der Primärmenge $DVM_{prim}^{t=1} = DV \setminus DVM(QVM^{t=0})$ eine geordnete Sekundärliste $DV_{sec}^{t=1}$ erzeugt werden kann, müssen die einzelnen Dokumentvektoren bezüglich ihrer Dominanz auf der Basis der Relevanzschätzungen geordnet werden, wobei zwei Strategien anwendbar sind:
1) Definition einer Aggregationsfunktion, wie dem arithmetischen Mittelwert:

$$m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1} \mid AM(rel(x) \mid M_{DV(m),k}^{t=0}), \overline{rel}(x_j^{t=1})) \mid k = 1, ..., \delta),$$
$$\overline{rel}(x_j^{t=1}) = 1/\delta * \sum_k rel(x_j^{t=1} \mid AM(rel(x) \mid M_{DV(m),k}^{t=0})). \quad (580)$$

2) Verwendung einer Mehr-Ziel-Strategie, wie z.B. die sukzessive Deaktivierung von Pareto-Mengen (siehe Abschnitt 2.4); Bachelier (1999d:32ff[22])), wobei auf den einzelnen Rängen in der Regel mehrere Dokumentvektoren liegen werden.

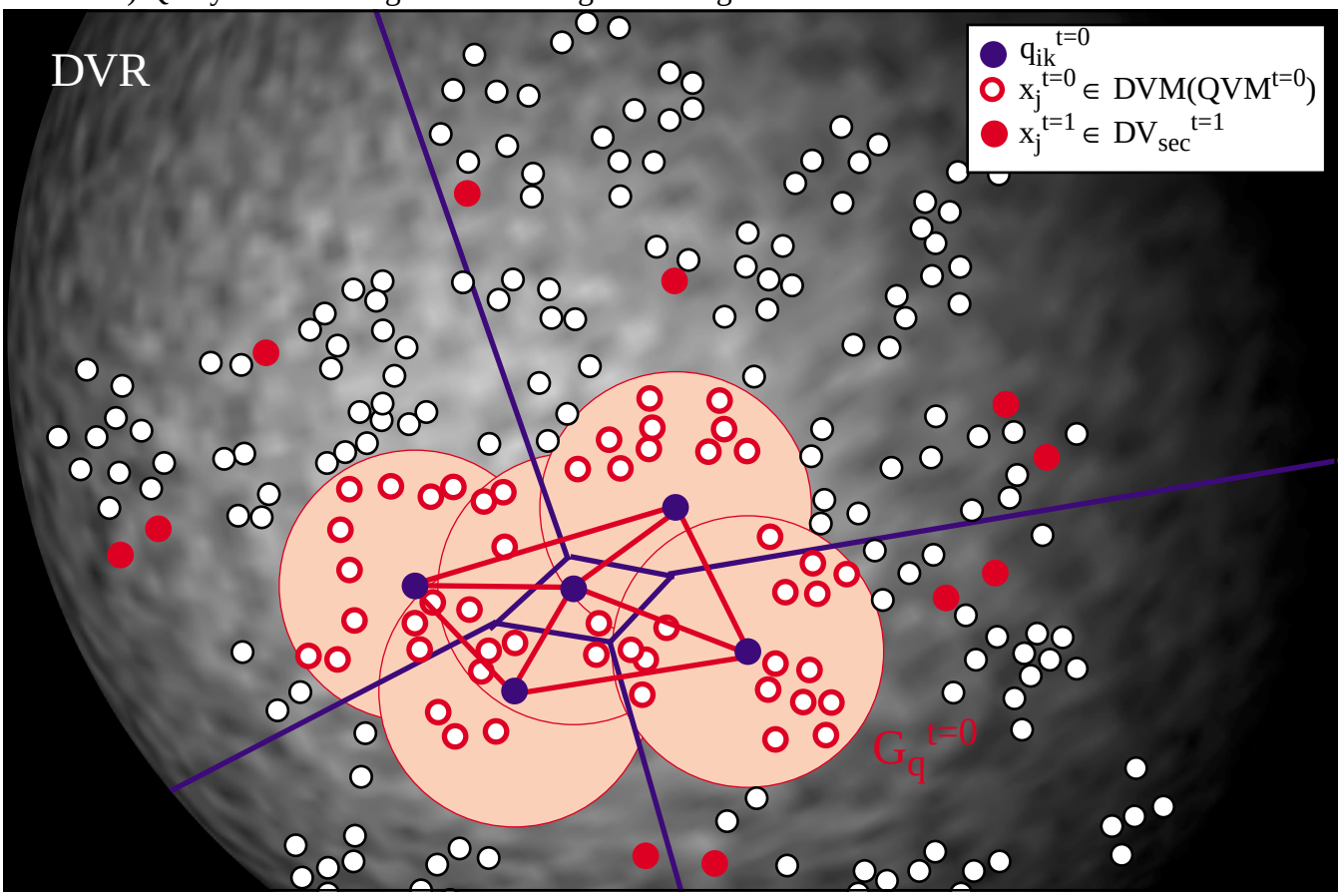
Unabhängig von dem konkreten Verfahren werden die ersten Elemente der Liste $DV_{sec}^{t=1}$ ausgewählt, welche die tertiäre Liste $DV_{ter}^{t=1}$ bilden und die korrespondierenden Dokumente werden dem Agenten präsentiert, der sie bewertet, sodass die Stimulusmenge $M_{DV(m)}^{t=1}$ entsteht. Vereinigt mit $M_{DV(m)}^{t=0}$ ergibt sich die Gesamtstimulusmenge $M_{DV(m)}^{t\leq1}$.

Geklärt werden muss, in welchem Zusammenhang die Stimuli zu den Queryvektoren stehen, bzw. ob ein Retrieval-Verfahren mit Queryvektor-Adaptionen durchgeführt wird. Wird keine Adaption durchgeführt, so bleiben die Elemente aus $QVM^{t=0}$ erhalten, doch durch die konstanten Umgebungen $U(q_{ik}^{t=0} \mid \varepsilon)$ bleiben die lokalen Ergebnismenge $DVM(q_{ik}^{t=0})$ sowie die daraus resultierenden Stimulusmengen $M_{DV,k}^{t=0}$ unverändert, unabhängig von $M_{DV(m)}^{t\leq1}$, sodass diese Option ausgeschlossen werden muss.

Bei einer konstanten Menge $QVM^{t=0}$ könnten die Umgebungen $U(q \mid \varepsilon)$ erweitert werden, indem bei jeder Iteration größere Umgebungs-Parameter verwendet werden, wobei dann geklärt werden muss, wie diese erzeugt werden. Gegen diese Option spricht, dass die Dokumentvektoren aus der sekundären Liste $DV_{sec}^{t=1}$, nicht lokal bezüglich der Positionen der Queryvektoren sind, sodass bei einer Erweiterung der Umgebungen Dokumentvektoren existieren können, denen durch den Agenten eine Relevanzbewertung zugeordnet wurde, die aber in keiner der Umgebungen liegen. Entsprechend dieser Argumentation soll die Erweiterung der Umgebungen ebenfalls ausgeschlossen werden.

Wird eine Zerlegung des DVR durch die Menge der Queryvektoren durchgeführt, indem eine Delaunay-Triangulation der Queryvektoren gebildet wird, deren korrespondierende Voronoi-Zerlegung eine Zerlegung des DVR darstellt (siehe Abschitt 3.9.2.6)), so ergibt sich jedoch eine andere Situation. Die Delaunay-Triangulation der Queryvektoren aus $QVM^{t=0}$ sei mit $G_q^{t=0}$ bezeichnet (siehe Abb. 122)).

Abb. 122) Queryvektor-Triangulation mit Ergebnismenge der Iteration t=0 und t=1



Unabhängig wo die Dokumentvektoren $x_j^{t=1}$ liegen, die in der sekundären Ergebnisliste $DV_{sec}^{t=1}$ enthalten sind, sie liegen immer in der Voronoi-Region genau einer der Queryvektoren aus $QVM^{t=0}$. Auf diese Weise werden die lokalen Stimulusmengen $M_{DV(m),k}^{t=0}$ der Queryvektoren in der Iteration t=1 nach der Auswahl von $DV_{sec}^{t=1}$ erweitert zu $M_{DV(m),k}^{t\leq1}$, indem die Dokumentvektoren aus $DV_{sec}^{t=1}$ dem Queryvektor $q_{ik}^{t=0}$ zugeordnet werden, wenn sie in dessen Voronoi-Region liegen. Durch diese aktualisierten Stimulusmengen liegen direkt die aktualisierten Approximationsmodelle $AM(rel(x) \mid M_{DV(m),k}^{t\leq1})$ vor. Es folgt der Beginn der nächsten Iteration t=2, indem alle durch den Agenten noch nicht bewertete Dokumentvektoren, d.h. die Primärmenge $DVM_{prim}^{t=2} = DV \setminus \{DVM(QVM^{t=0}) \cup DV_{sec}^{t=1}\}$, durch alle Modelle $AM(rel(x) \mid M_{DV(m),k}^{t\leq1})$, $k = 1, ..., \delta$, bewertet werden, gefolgt von der Bildung eines Rankings und der Sekundärliste $DV_{sec}^{t=2}$.

4.4.1.2) Approximationsmodell-Polyrepräsentation durch Bootstrap-Verfahren

Eine prinzipiell andere Vorgehensweise kann durch die Verwendung von Resampling-Verfahren wie dem Stimulus-Bootstrap, erreicht werden (siehe Abschnitt 2.2.1)). Hierbei wird aus einer wachsenden Gesamt-Stimulusmenge $M_{DV(m)}^{t=0}$, $M_{DV(m)}^{t\leq1}$, $M_{DV(m)}^{t\leq2}$, ..., in jeder Iteration durch zufälliges Ziehen mit Zurücklegen Bootstrap-Stimuluslisten erzeugt, die zusammen mit einem Approximationsverfahren jeweils ein Approximationsmodell repräsentieren. Diese Listen werden in einer Iteration unabhängig voneinander und unabhängig von den vorangegangenen Iterationen jeweils neu erzeugt, was den Unterschied zu Verfahren kennzeichnet, bei denen Stimulusmengen und die Modelle einen gewissen individuellen Charakter besitzen, der über Iterationsgrenzen erhalten bleibt, während sich die Stimulusmengen im gewissem Umfang verändern, wie dies bei der Zuordnung von Stimuli zu Queryvektoren im vorangegangenen Abschnitt der Fall ist.

Ausgegangen wird von einer Queryvektor-Monorepräsentation, wobei in der Iteration t=0 die Stimulusmenge $M_{DV(m)}^{t=0}$ durch die Bewertungen des Agenten aus der $f^{t=0}$'elementigen Menge $DVM(q_i^{t=0})$ gebildet wird. Aus dieser Gesamt-Stimulusmenge wird δ mal ein unabhängiges Ziehungesverfahren durchgeführt, bei denen jeweils $f^{t=0}$ Ziehungen mit Zurücklegen aus $M_{DV(m)}^{t=0}$ angewendet werden. Es ergeben sich Bootstrap-Stimuluslisten $BM_{DV(m),k}^{t=0}$, bei denen einzelne Stimuli $m_{k,j}^{t=0}$ aus $M_{DV(m)}^{t=0}$ mehrfach vorliegen können:

$$BM_{DV(m),k}^{t=0} = (m_{k,j}^{t=0} = (x_{k,j}^{t=0}, rel(x_{k,j}^{t=0})) \in M_{DV(m)}^{t=0} \mid j = 1, ..., f^{t=0}). \quad (581)$$

Die Stimuluslisten werden in einer Bootstrapmenge von Einzel-Stimuluslisten zusammengefasst:

$$BMM^{t=0} = \{BM_{DV(m),k}^{t=0} \mid k = 1, ..., \delta\}. \quad (582)$$

Zu jeder dieser Stimuluslisten $BM_{DV(m),k}^{t=0}$ korrespondiert ein instanzbasiertes Approximationsmodell $AM(rel(x) \mid BM_{DV(m),k}^{t=0})$, mit dem die Dokumentvektoren der primären Ergebnismenge der nächsten Iteration t=1 bewertet werden. Wird von der primären Dokumentvektorenmenge $DVM_{prim}^{t=1} = DV \setminus DVM(q_i^{t=0})$ ausgegangen, so wird für jeden Dokumentvektor $x_j^{t=1}$ aus dieser Primärmenge δ Relevanzschätzungen $rel(x_j^{t=1} \mid AM(rel(x) \mid BM_{DV(m),k}^{t=0}))$ durchgeführt, die in die vorläufige Datenstruktur des Stimulus $m_j^{t=1}$ aufgenommen werden können:

$$m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1} \mid AM(rel(x) \mid BM_{DV(m),k}^{t=0})) \mid k = 1, ..., \delta). \quad (583)$$

Weiterhin kann eine Relevanzschätzung mit dem Gesamtmodell durchgeführt werden, dass alle Stimuli aus $M_{DV(m)}^{t=0}$ verwendet, d.h. mit $AM(rel(x) \mid M_{DV(m)}^{t=0})$:

$$m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1} \mid AM(rel(x) \mid M_{DV(m)}^{t=0})),$$
$$rel(x_j^{t=1} \mid AM(rel(x) \mid BM_{DV(m),k}^{t=0})) \mid k = 1, ..., \delta). \quad (584)$$

Mit Hilfe dieser Relevanzschätzungen lassen sich nun einige Formen von Bootstrap-Schätzungen aggregieren. Im einfachsten Fall werden die δ Schätzungen der lokalen Bootstrap-Modelle mit Hilfe des arithmetischen Mittelwertes aggregiert:

$$rel(x_j^{t=1} \mid BMM^{t=0}) = 1/\delta \sum_j rel(x_j^{t=1} \mid AM(rel(x) \mid BM_{DV(m),k}^{t=0})), \forall BM_{DV(m),k}^{t=0} \in BMM^{t=0}. \quad (585)$$

Die δ einfachen Bootstrap-Einzelschätzungen können in der vorläufigen Struktur des Stimulus ersetzt werden durch die Bootstrap-Gesamtschätzung, wenn die Einzelschätzungen nicht mehr benötigt werden:

$$m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1} \mid AM(rel(x) \mid M_{DV(m)}^{t=0})), rel(x_j^{t=1} \mid BMM^{t=0})). \quad (586)$$

Es kann weiterhin eine Bias-Schätzung an der Stelle von $x_j^{t=1}$ erzeugt werden, indem diese Bootstrap-Schätzung verglichen wird mit der Schätzung durch das Gesamtmodell auf der Basis aller vorliegenden Stimuli, wobei eine 0,632-Korrektur durchgeführt werden sollte (Efron & Tibshirani (1993[105])):

$$bias(x_j^{t=1})_{0,632}\hat{} = (rel(x_j^{t=1} \mid BMM^{t=0}) - rel(x_j^{t=1} \mid AM(rel(x) \mid M_{DV(m)}^{t=0}))/0,632. \quad (587)$$

Diese Bias-Schätzung führt zu einer 0,632-bias-korrigierten Bootstrap-Gesamtschätzung, welche die Bootstrap-Gesamtschätzung und die Schätzung durch das Gesamtmodell in der Struktur eines Stimulu ersetzen kann:

$$m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1} \mid BMM^{t=0})_{0,632bias}), \text{ mit}$$
$$rel(x_j^{t=1} \mid BMM^{t=0})_{0,632bias} = rel(x_j^{t=1} \mid BMM^{t=0}) - bias(x_j^{t=1})_{0,632}\hat{}. \quad (588)$$

Nachdem allen Dokumentvektoren $x_j^{t=1}$ aus der primären Ergebnismenge $DVM_{prim}^{t=1}$ eine solche Bootstrap-Relevanzschätzung zugeordnet wurde, werden die Dokumentvektoren nach fallender Relevanzschätzung sortiert, wodurch die Sekundärliste $DV_{sec}^{t=1}$ entsteht. Die Dokumente, die den ersten Rängen, z.B. den ersten $f^{t=0}$ Rängen, entsprechen, werden dem Agenten präsentiert, der diese bewertet, sodass die Stimulusmenge $M_{DV(m)}^{t=1}$ ensteht, die mit $M_{DV(m)}^{t=0}$ zu $M_{DV(m)}^{t\leq1}$ vereingt wird. Mit dieser Gesamtstimulusmenge liegt das Approximationsmodell $AM(rel(x) \mid M_{DV}^{t\leq1})$ vor. Es folgt die Bildung von δ Bootstraplisten $BM_{DV(m),k}^{t=1}$ durch Ziehungen mit Zurücklegen aus $M_{DV(m)}^{t\leq1}$, die in der Menge der Stimuluslisten $BMM^{t=1}$ zusammengefasst werden, und die jeweils ein Modell $AM(rel(x) \mid BM_{DV(m),k}^{t=1})$ repräsentieren. Nachdem alle Modelle ermittelt wurden, wird die nächste Iteration t=2 eingeleitet, in der zunächst die primäre Ergebnismenge $DVM_{prim}^{t=2}$ ermittelt wird, deren Elemente durch die Bootstrap-Modelle bewertet werden, gefolgt von der Bildung der Sekundärliste $DV_{sec}^{t=2}$.

4.4.2) Polyrepräsentation bei prototypbasierten Approximationsmodellen

Bei instanzbasierten Modellen existiert die einzige Möglichkeit der Polyrepräsentation darin, unterschiedliche Stimulusmengen aufzubauen, wenn genau ein Approximationsverfahren verwendet werden soll, d.h. es muss eine Stimulus-Polyrepräsentation erzeugt werden, um eine Modell-Polyrepräsentation zu erzeugen. Prototypbasierte Modelle bestehen aus Stimuli, Prototypen und einem Approximationsverfahren, sodass die Erzeugung einer Polyrepräsentation diese drei Elemente verwenden kann, wobei im weiteren wieder von einer Approximationsverfahren-Monorepräsentation ausgegangen werden soll. Weiterhin soll berücksichtigt werden, dass die stützpunktbasierte Approximation auf der Basis der Prototypen erfolgt, sodass eine Prototyp-Polyrepräsentation in jedem Fall vorliegen muss. Es stellt sich somit die Frage, wie diese Prototyp-Polyrepräsentation erzeugt werden soll.

Es sind die folgenden Vorgehensweisen denkbar, von denen jedoch nur die beiden ersten Vorgehensweisen näher betrachtet werden sollen:

1) Erzeugung einer Stimulus-Polyrepräsentation z.B. durch Resampling-Verfahren, die ihrerseits zu je einem prototypbasierten Modell führt, sodass eine Prototyp-Polyrepräsentation gebildet wird.
2) Eine Prototyp-Monorepräsentation wird durch Resampling-Verfahren zu einer Prototyp-Polyrepräsentation, indem z.B. aus der Prototypmenge Elemente mit Zurücklegen gezogen werden.
3) Verwendung der Erzeugungshistorie eines GNG-SOM, indem bestimmte Zwischenzustände beim Wachstum gespeichert werden, d.h. die Zwischenzustände werden als Elemente einer Approximationsmodell-Polyrepräsentation direkt verwendet (siehe Bachelier (1999d:171ff[22])).

Wird berücksichtigt, dass die Prototypen aus einem Stützpunkt im n-dimensionalen Dokumentvektorenraum und einem Stützpunkt im ein-dimensionalen Relevanzraum bestehen, so ergibt sich eine spezielle Sicht auf die Approximationsmodell-Polyrepräsentation, bei der einem Prototypen genau ein Stützpunkt im DVR jedoch mehrere Stützpunkte im Relevanzraum zugeordnet werden. Diese Vorgehensweise kann ebenfalls als Prototyp-Polyrepräsentation interpretiert werden, die durch eine Stimulus-Polyrepräsentation erzeugt wird, d.h. sie kann als Spezialfall der ersten oben genannten Verfahrensklasse betrachtet werden. Sie ist im Gegensatz zu den anderen Verfahren dieser Klasse besonders speichereffizient, da alle GNG-SOMs in einer Repräsentationsform zusammengefasst werden können, bei welcher der n-dimensionale Stützpunkt (Gewichtsvektor) genau einmal auftritt, und die ein-dimensionalen Stützpunkte (Relevanzschätzungen) δ mal in einer Neuronenstruktur auftreten. Bei einer Polyrepräsentation mit δ Modellen und $\mu_{N,D}$ Prototypen pro Modell bedeutet die konventielle Speicherung $(n + 1) * \mu_{N,D} * \delta$ reelle Komponenten, während diese Form der Polyrepräsentation nur $(n + \delta) * \mu_{N,D}$ reelle Komponenten benötigt.

Weiterhin muss bei prototypbasierten Approximationsmodellen, die auf SC-GNG-SOMs aufbauen, unterschieden werden, ob eine unklassifizierte oder eine klassifizierte Dokumentvektorenmenge vorliegt. Im ersten Fall muss nach der ersten Bewertung von nachgewiesenen Dokumentvektoren in der Iteration t=0 ein SC-GNG-SOM aufgebaut werden, indem die bewerteten Stimuli verwendet werden (siehe Abschnitt 4.2.5)). Im zweiten Fall kann die existierende GNG-SOM verwendet werden, die auf der Basis aller Dokumentvektoren in DV aufgebaut und ständig aktualisiert wird, unabhängig ob einzelne Dokumentvektoren bewertet wurden. Diese beiden Fälle werden in den beiden nachfolgenden Abschnitten unter Berücksichtigung der unterschiedlichen Möglichkeiten dargestellt, mit denen eine Prototyp-Polyrepräsentation erzeugt werden kann.

4.4.2.1) Polyrepräsentierte Prototyp-Modelle bei unklassifizierten Dokumentvektoren

Bei einer im weiteren unterstellten Dokumentvektor-Monorepräsentation müssen die zu einer Iteration vorliegenden Stimuli, d.h. bewertete Dokumentvektoren, verwendet werden, um ein prototypbasiertes Approximationsmodell zu erzeugen. Hierzu wird zunächst unüberwacht eine SC-GNG-SOM aufgebaut, die im Abschnitt 4.2.5) als $N_{D,bew}$ bezeichnet wurde. Wird die Initialisierungs-Iteration t=0 nach der Bewertung der nachgewiesenen Dokumente betrachtet, die zu der Ergebnisliste $DV(q_i^{t=0})$ korrespondieren, so existiert die $f^{t=0}$ elementige Stimulusmenge $M_{DV(m)}{}^{t=0}$, mit deren Hilfe $N_{D,bew}{}^{t=0}$ on-scratch aufgebaut wird. Nach dem unüberwachten Aufbau mit Hilfe der Dokumentvektoren $x_j^{t=0}$, die zu den Stimuli

$m_j^{t=0} \in M_{DV(m)}^{t=0}$ gehören, wird jedem Neuron $n_{D,k}^{t=0} \in N_{D,bew}^{t=0}$ in Abhängigkeit von seinem Gewichtsvektor $w(x_D)_k^{t=0}$ eine Relevanzschätzung $rel(w(x_D)_k^{t=0})\hat{}$ zugeordnet. Diese kann z.B. streng lokal formuliert werden, indem alle Stimuli in der lokalen Stimulusmenge $M_{DV(m),k}^{t=0}$ des Neurons $n_{D,k}^{t=0}$ an einer stützpunktbasierten LWR-Approximation teilnehmen. Es ergibt sich die Struktur für eine solche SC-GNG-SOM mit $M_{DV,k}^{t=0}$ als der Menge der Dokumentvektoren, die in der Voronoi-Region des Neurons liegen:

$$N_{D,bew}^{t=0} = \{n_{D,k}^{t=0} = (w(x_D)_k^{t=0}, rel(w(x_D)_k^{t=0})\hat{}, M_{DV,k}^{t=0}, M_{DV(m),k}^{t=0}, C_{D,k}^{t=0}) \mid k = 1, ..., \mu_{N,D}^{t=0}\}. \quad (589)$$

4.4.2.1.1) Unabhängiger Aufbau von Prototyp-Modellen durch Stimulus-Bootstrap

Eine Möglichkeit eine Bootstrapmenge von Prototyp-Modellen aufzubauen, besteht in der unabhängigen Erzeugung dieser Modelle, indem zunächst δ Bootstrap-Stimuluslisten $M_{DV(m),p}^{t=0}$ durch $f^{t=0}$ faches zufälliges Ziehen mit Zurücklegen aus $M_{DV(m)}^{t=0}$ erzeugt werden. Mit diesen δ Bootstrap-Stimuluslisten kann jeweils ein unabhängiger Aufbau einer Bootstrap-GNG-SOM $N_{D,p,bew}^{t=0}$ erfolgen, wobei die Relevanzschätzung $rel(w(x_D)_k^{t=0})\hat{}$ an der Stelle der Gewichtsvektoren $w(x_D)_{p,k}^{t=0}$ durch die lokalen Stimuli in der Liste $M_{DV(m),p,k}^{t=0}$ mit einer stützpunktbasierten LWR-Approximation gebildet werden. Die Tatsache, dass einige Stimuli in einer lokalen Liste mehrfach vorliegen, ist für die Durchführung einer LWR-Approximation unerheblich, da die einzelnen Stimuli in der gewichteten Summe so behandelt werden, als wenn sie unterschiedlich wären. Es ergibt sich eine Menge $BMN_{D,bew}^{t=0}$ mit der Ursprungs-GNG-SOM $N_{D,bew}^{t=0}$ und δ Bootstrap-GNG-SOMs, die jeweils in Verbindung mit einem Approximationsverfahren ein prototypbasiertes Relevanz-Approximationsmodell $AM(rel(x) \mid N_{D,p,bew}^{t=0})$ ergeben:

$$BMN_{D,bew}^{t=0} = \{N_{D,bew}^{t=0}, N_{D,p,bew}^{t=0} \mid p = 1, ..., \delta\}, \text{ mit}$$
$$N_{D,p,bew}^{t=0} = \{n_{D,p,k}^{t=0} = (w(x_D)_{p,k}^{t=0}, rel(w(x_D)_{p,k}^{t=0})\hat{}, M_{DV,p,k}^{t=0}, M_{DV(m),p,k}^{t=0}, C_{D,p,k}^{t=0}) \mid k = 1, ..., \mu_{N,D}^{t=0}\}. \quad (590)$$

Ein solcher unabhängiger Aufbau lässt sich zwar in δ unabhängige Tasks parallelisieren, doch der Gesamtaufwand ist als sehr groß zu bewerten.

Im weiteren Verlauf eines unabhängigen Aufbaus von Modellen wird nach der Bildung der Bootstrap-Modellmenge die Iteration t=1 eingeleitet, gefolgt von der Erzeugung einer primären Ergebnismenge $DVM_{prim}^{t=1}$, ohne dass hier nochmals auf die Methoden eingegangen werden sollen, wie diese Menge aus der Restmenge $DV \setminus DV(q_i^{t=0})$ erzeugt werden kann. Die Elemente dieser Primärmenge, d.h. die Dokumentvektoren $x_j^{t=1}$ sollen durch die Approximationsmodelle Relevanzschätzungen erhalten, die zu den GNG-SOMs aus $BMN_{D,bew}^{t=0}$ korrespondieren. D.h. $AM(rel(x) \mid N_{D,bew}^{t=0})$ sowie die δ Modelle $AM(rel(x) \mid N_{D,p,bew}^{t=0})$ erzeugen jeweils eine Relevanzschätzung an der Stelle $x_j^{t=1}$. Bezogen auf einen Dokumentvektor $x_j^{t=1}$ aus der primären Ergebnismenge $DV_{prim}^{t=1}$ der Iteration t=1 bedeutet dies, dass alle δ Approximationsmodelle $AM(rel(x) \mid NL_{D,p,bew}^{t=0})$ eine Einzel-Relevanzschätzung abgeben, erweitert um die Relevanzschätzung auf der Basis der Ursprungs-GNG-SOM, d.h. durch das Modell $AM(rel(x) \mid N_{D,bew}^{t=0})$:

$$rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,bew}^{t=0})) = 1/\nu_j \sum_k h(d_{DVR}(x_j^{t=1}, w(x_D)_k^{t=0})) * rel(w(x_D)_k^{t=0})\text{^}, \text{ mit}$$
$$\nu_j = \sum_k h(d_{DVR}(x_j^{t=1}, w(x_D)_k^{t=0})), \forall n_{D,k}^{t=0} \in N_{D,bew}^{t=0},$$
$$rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,p,bew}^{t=0})) = 1/\nu_j \sum_k h(d_{DVR}(x_j^{t=1}, w(x_D)_{p,k}^{t=0})) * rel(w(x_D)_{p,k}^{t=0})\text{^}, \text{ mit}$$
$$\nu_j = \sum_k h(d_{DVR}(x_j^{t=1}, w(x_D)_{p,k}^{t=0})), \forall n_{D,p,k}^{t=0} \in N_{D,p,bew}^{t=0}, p = 1, ..., \delta. \quad (591)$$

Diese δ+1 Einzelschätzungen können zu unterschiedlichen Bootstrap-Gesamtschätzungen aggregiert werden. Die einfachste Möglichkeit ist die ungewichtete Mittelwertsbildung:

$$rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,p,bew}^{t=0}), p = 1, ..., \delta) =$$
$$1/\delta \sum_p rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,p,bew}^{t=0})). \quad (592)$$

Erweiterte Verfahren schätzen zunächst die Bias mit

$$bias(x_j^{t=1})_{0,632}\text{^} =$$
$$(rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,p,bew}^{t=0}), p = 1, ..., \delta) - rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,bew}^{t=0})))/0{,}632, \quad (593)$$

und verwenden diese Bias-Schätzung zu einer 0,632-bias-korrigierten Bootstrap-Gesamtschätzung:

$$rel(x_j^{t=1} \mid BMN_{D,bew}^{t=0})_{0,632bias} =$$
$$rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,p,bew}^{t=0}), p = 1, ..., \delta) - bias(x_j^{t=1})_{0,632}\text{^}. \quad (594)$$

Nachdem alle Dokumentvektoren $x_j^{t=1}$ in der primären Ergebnismenge $DVM_{prim}^{t=1}$ ihre aggregierte Bootstrap-Relevanzschätzung erhalten haben, werden sie entsprechend fallenden Schätzungen sortiert, sodass die sekundäre Ergebnisliste $DV_{sec}^{t=1}$ entsteht, aus der die ersten Elemente ausgewählt werden. Die korrespondierenden Dokumente werden dem Agenten präsentiert, der diese bewertet, sodass die Stimulusmenge $M_{DV(m)}^{t=1}$ entsteht, die vereinigt mit $M_{DV(m)}^{t=0}$ die Menge $M_{DV(m)}^{t\leq1}$ ergibt. Bei einem unabhängigen Aufbau von Modellen werden die herausragenden Eigenschaften einer SC-GNG-SOM bezüglich der Veränderung und insbesondere der Erweiterung der zugrunde liegenden Stimulusmenge nicht genutzt (siehe Bachelier (1998c[17])), sondern es folgt ein on-scratch-Aufbau des Gesamt-GNG-SOM $N_{D,bew}^{t=1}$ auf der Basis der Gesamt-Stimulusmenge $M_{DV(m)}^{t\leq1}$. Weiterhin werden die δ Bootstrap-Stimuluslisten $M_{DV(m),p}^{t\leq1}$ durch Ziehen mit Zurücklegen aus $M_{DV(m)}^{t\leq1}$ gebildet, gefolgt von dem on-scratch-Aufbau der Bootstrap-GNG-SOMs $N_{D,p,bew}^{t=1}$, wodurch die δ+1'elementige Menge $BMN_{D,bew}^{t=1}$ komplettiert wird. Es folgt der Übergang zur nachfolgenden Iteration t=2 mit der Ermittlung der Primärmenge $DV_{prim}^{t=2}$ sowie der Bewertung deren Dokumentvektoren durch die Modelle aus $BMN_{D,bew}^{t=1}$ sowie die Aggregation zu Bootstrap-Relevanzschätzungen.

Bei einer weniger rigiden Definition der Unabhängigkeit werden die Gesamt-GNG-SOMs $N_{D,bew}^{t}$ bei jeder Iteration on-scratch, d.h. unabhängig, auf der Basis der jeweils vorliegenden Gesamt-Stimulusmenge $M_{DV(m)}^{\leq t}$ aufgebaut, während die Bootstrap-GNG-SOMs $N_{D,p,bew}^{t}$ aus $N_{D,bew}^{t}$ unter Verwendung der unabhängig voneinander erzeugten Bootstrap-Stimuluslisten $M_{DV(m),p}^{\leq t}$ durch Nachadaptions-Operationen gebildet werden. Verfahren unter Verwendung eines Stimulus-Bootstraps und Nachadaptions-Operationen werden im Rahmen der abhängigen Methoden zum Aufbau von Prototyp-Modellen dargestellt werden, sodass an dieser Stelle darauf verzichtet werden soll.

4.4.2.1.2) Unabhängiger Aufbau von Prototyp-Modellen durch Neuronen-Bootstrap

Bei einem unabhängigen Aufbau durch Neuronen-Bootstrap wird in der Initialisierungs-Iteration t=0 zunächst das Gesamt-GNG-SOM $N_{D,bew}^{t=0}$ aus der Gesamt-Stimulusmenge $M_{DV(m)}^{t=0}$ on-scratch gebildet. Eine Bootstrap-GNG-SOM wird durch die zufällige Ziehung von $\mu_{N,D}^{t=0}$ Neuronen mit Zurücklegen aus $N_{D,bew}^{t=0}$ erzeugt, wobei es sich nicht mehr um eine Neuronenmenge, sondern um eine Neuronenliste $NL_{D,p,bew}^{t=0}$ handelt, da einzelne Neurone mehrfach vorliegen können, während andere Neurone im Vergleich zu $N_{D,bew}^{t=0}$ nicht mehr enthalten sind. Zusammen mit der Gesamt-GNG-SOM werden die δ Bootstrap-Neuronenlisten, zusammengefasst:

$$BNL_{D,bew}^{t=0} = \{N_{D,bew}^{t=0}, NL_{D,p,bew}^{t=0} \mid p = 1, ..., \delta\}. \quad (595)$$

Wie erwähnt, können in den Bootstraplisten einzelne Neurone bezogen auf $N_{D,bew}^{t=0}$ fehlen und einzelne Neurone können mehrfach vorliegen, sodass es sich nicht um konsistente GNG-SOMs handelt, da keine konsistente Verbindungsstruktur zwischen den Neuronen bzw. Gewichtsvektoren existiert. Das Fehlen einzelner Neurone könnte durch eine Korrektur der Verbindungsvektoren $C_{D,p,k}^{t=0}$ repräsentiert werden, indem die entsprechenden Komponenten entfernt werden. Dies kann zu einem nicht zusammenhängenden Graphen führen, wobei dies zwar eine konsistente GNG-SOM ergibt, in der jedoch die Suche nach einer Nachbarneuronenmenge nicht mehr durch die Verbindungsstruktur konsistent durchführbar ist. Demgegenüber bereiten mehrfach auftretende Neurone Probleme bezüglich ihrer Kodierung in den Verbindungsvektoren.

Wird jedoch berücksichtigt, dass die Neuronenlisten ausschließlich dazu verwendet werden sollen, um prototypbasierte Stützpunkte für ein Relevanz-Approximationsmodell zu liefern, so spielt bei bestimmten Formen einer LWR-basierten Approximation die Verbindungsstruktur zwischen den Prototypen keine Rolle. Soll die Relevanzschätzung an der Stelle eines Dokumentvektors $x_j^{t=1}$ aus der primären Ergebnismenge der nachfolgenden Iteration t=1 gebildet werden, so wurde bei prototypbasierten GNG-SOM-Modellen eine lokale Approximation verwendet, bei der zunächst das Gewinner-Neuron bestimmt wurde, das zusammen mit seinen verbundenen Nachbarneuronen die Relevanzschätzung durchführt. Bei einer solchen lokalen Formulierung wird eine konsistente Verbindungsstruktur, d.h. eine zusammenhängende Verbindungsstruktur erforderlich, wobei mehrfach auftretende Neurone kein großes Problem darstellen, da sie in der gewichteten Summe eines LWR-Verfahrens den anderen Neuronen gleich gestellt werden. Wird demgegenüber eine globale Formulierung einer LWR-basierten Approximation verwendet, bei der alle Neurone entsprechend der Position ihres Gewichtsvektors und ihrer Relevanzschätzung an der Bestimmung der Relevanzschätzung $rel(x_j^{t=1})$^ teilnehmen, so wird die Verbindungsstruktur zwischen den Neuronen und das mehrfache Vorliegen von Neuronen irrelevant, sodass diese globale Formulierung in Verbindung mit einem Neuronen-Resampling zur Prototyp-Polyrepräsentation verwendet werden soll. Für den weiteren Verlauf eines solchen Verfahrens bedeutet dies, dass nach der Bildung von $BNL_{D,bew}^{t=0}$ die nachfolgende Iteration t=1 eingeleitet wird, gefolgt von der Erzeugung einer primären Ergebnismenge $DVM_{prim}^{t=1}$, deren Elemente $x_j^{t=1}$ mit Hilfe der Modelle aus $BNL_{D,bew}^{t=0}$ bewertet werden. D.h. das Approxmationsmodell $AM(rel(x) \mid N_{D,bew}^{t=0})$ sowie die δ Modelle $AM(rel(x) \mid NL_{D,p,bew}^{t=0})$ erzeugen zunächst jeweils eine Einzel-Relevanzschätzung durch ein globales, prototypbasiertes LWR-Verfahren:

$$rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,bew}^{t=0})) = 1/\nu_j \sum_k h(d_{DVR}(x_j^{t=1}, w(x_D)_k^{t=0})) * rel(w(x_D)_k^{t=0})\text{^}, \text{ mit}$$
$$\nu_j = \sum_k h(d_{DVR}(x_j^{t=1}, w(x_D)_k^{t=0})), \forall n_{D,k}^{t=0} \in N_{D,bew}^{t=0},$$
$$rel(x_j^{t=1} \mid AM(rel(x) \mid NL_{D,p,bew}^{t=0})) = 1/\nu_j \sum_k h(d_{DVR}(x_j^{t=1}, w(x_D)_{p,k}^{t=0})) * rel(w(x_D)_{p,k}^{t=0})\text{^}, \text{ mit}$$
$$\nu_j = \sum_k h(d_{DVR}(x_j^{t=1}, w(x_D)_{p,k}^{t=0})), \forall n_{D,p,k}^{t=0} \in NL_{D,p,bew}^{t=0}, p = 1, ..., \delta. \quad (596)$$

Diese δ+1 Einzelschätzungen können wieder zu unterschiedlichen Bootstrap-Gesamtschätzungen aggregiert werden, wie $rel(x_j^{t=1} \mid AM(rel(x) \mid NL_{D,p,bew}^{t=0}), p = 1, ..., \delta)$ oder $rel(x_j^{t=1} \mid BNL_{D,bew}^{t=0})_{0,632bias}$. Es folgt die Sortierung der Dokumentvektoren aus $DVM_{prim}^{t=1}$ entsprechend fallender Relevanzschätzungen, wodurch die sekundäre Ergebnisliste $DV_{sec}^{t=1}$ entsteht, aus der die ersten Elemente ausgewählt und in $DV_{ter}^{t=1}$ übernommen werden. Die korrespondierenden Dokumente werden dem Agenten präsentiert, der diese bewertet, sodass die Stimulusmenge $M_{DV(m)}^{t=1}$ entsteht, die vereinigt mit $M_{DV(m)}^{t=0}$ die Menge $M_{DV(m)}^{t\leq1}$ ergibt. Mit diesen neuen Stimuli soll die Menge der Modelle $BNL_{D,bew}^{t=1}$ erzeugt werden, wobei im Fall einer rigide definierten Unabhängigkeit gefordert wird, dass $N_{D,bew}^{t=1}$ unabhängig von $N_{D,bew}^{t=0}$ on-scratch durch Präsentationen der Stimuli aus $M_{DV(m)}^{t\leq1}$ aufgebaut wird. Bei einer weniger rigiden Definition von Unabhängigkeit kann eine Nachadaption von $N_{D,bew}^{t=0}$ zu $N_{D,bew}^{t=1}$ erfolgen, wobei unterschiedliche Formen der SC-GNG-SOM-Nachadaption anwendbar sind (siehe Bachelier (1998c[17])). Beispielsweise kann eine Nachadaption erfolgen, bei der die Stimuli aus $M_{DV(m)}^{t=1}$ den Neuronen aus $N_{D,bew}^{t=0}$ zugeordnet werden, entsprechend der Distanz von Gewichtsvektor und Dokumentvektor. Danach wird für jedes Neuron ein lokales Fehlermaß wie der Quantifizierungsfehler bestimmt (siehe Abschnitt 2.3.1)), wobei im Fall, dass mindestens ein Neuron einen Fehlerschwellenwert überschreitet, lokale Präsentations-, Adaptions- und gegebenenfalls Wachstumsoperationen eingefügt werden.

Unabhängig, ob $N_{D,bew}^{t=1}$ on-scratch oder durch eine Form der Nachadaption erzeugt wird, es folgt die Bildung von Bootstrap-Neuronenlisten $NL_{D,p,bew}^{t=1}$, indem $\mu_{N,D}^{t=1}$ mal ein Neuron aus $N_{D,bew}^{t=1}$ gleichverteilt zufällig mit Zurücklegen gezogen wird. D.h. die Bootstrap-Neuronenlisten $NL_{D,p,bew}^{t=1}$ werden vollständig unabhängig von den Listen $NL_{D,p,bew}^{t=0}$ der vorangegangenen Iteration gebildet. Nachdem alle δ+1 Elemente von $BNL_{D,bew}^{t=1}$ erzeugt wurden, wird die nächste Iteration t=2 eingeleitet, und es wird die primäre Ergebnismenge ermittelt $DVM_{prim}^{t=2}$, deren Elemente durch die Modelle, die zu den Neuronenlisten aus $BNL_{D,bew}^{t=1}$ korrespondieren, bewertet. Die nach dem Sortieren entstehende sekundäre Ergebnisliste wird ganz oder im ersten Teil ausgewählt, und die korrespondierenden Dokumente werden dem Agenten präsentiert, der sie bewertet und über einen Verfahrensabbruch entscheidet.

4.4.2.1.3) Abhängiger Aufbau von Prototyp-Modellen durch Stimulus-Bootstrap

Verfahren zum abhängigen Aufbau von Prototyp-Modellen verwenden Ableitungen von Bootstrap-Modellen aus einem Gesamtmodell, das auf der Basis aller vorliegender Stimuli aufgebaut wird. Kombiniert wird dies mit einer konsequenten Nachadaption, sodass nur in der Initialisierungs-Iteration das Modell $N_{D,bew}^{t=0}$ on-scratch mit Hilfe der Gesamt-Stimulusmenge $M_{DV(m)}^{t=0}$ erzeugt wird. Dabei sind zwei prinzipielle Strategien anwendbar:

1) Einmalige Erzeugung der Bootstrap-GNG-SOMs aus der Gesamt-GNG-SOM, gefolgt von der Identitätserhaltung in den weiteren Iterationen.
2) Ableitung der Bootstrap-GNG-SOMs aus dem Gesamt-GNG-SOM in jeder Iteration, wobei nur die Gesamt-GNG-SOM ihre Identität beibehält.

4.4.2.1.3.1) Beibehaltung von Bootstrap-GNG-SOMs

Im ersten Fall wird aus dem Gesamtmodell $N_{D,bew}^{t=0}$ in der Initialisierungs-Iteration Bootstrap-Modelle $N_{D,p,bew}^{t=0}$ genau einmal abgeleitet werden, wobei in der nachfolgenden Iteration $N_{D,bew}^{t=0}$ und $N_{D,p,bew}^{t=0}$ zu $N_{D,bew}^{t=1}$ und $N_{D,p,bew}^{t=1}$ aktualisiert werden.

Die Ableitung aus $N_{D,bew}^{t=0}$ erfolgt durch ein Stimulus-Bootstrap-Verfahren, indem zunächst aus der Gesamt-Stimulusmenge $M_{DV(m)}^{t=0}$ durch Ziehen mit Zurücklegen Bootstraplisten $M_{DV(m),p}^{t=0}$ erzeugt werden. Zur Ableitung wird das Gesamtmodell kopiert und die Kopie wird mit $N_{D,p,bew}^{t=0}$ bezeichnet, gefolgt von der Umwandlung der neuronen-spezifischen Stimulusmengen $M_{DV(m),k}^{t=0}$ in Stimuluslisten $M_{DV(m),p,k}^{t=0}$, in denen mehrere gleiche Stimuli enthalten sein können. Ziel ist es im weiteren, mit Hilfe der Stimuli in $M_{DV(m),p}^{t=0}$ die Gewichtsvektoren und die Relevanzschätzungen an der Stelle der neuen Gewichtsvektoren zu verändern, sodass die sich ergebenden neuen Stützpunkte zusammen mit einem Approximationsverfahren das Bootstrap-Approximationsmodell $AM(rel(x) \mid N_{D,p,bew}^{t=0})$ ergibt.

Eine einfache Möglichkeit der Ableitung eines Gewichtsvektors von $w(x_D)_k^{t=0}$ zu $w(x_D)_{p,k}^{t=0}$ ist die Bildung des Mittelwertsvektors der Dokumentvektoren-Komponente der Elemente der lokalen Stimulusliste $M_{DV(m),p,k}^{t=0}$. Das Problem besteht jedoch darin, dass bei einem größeren Bootstrap-Parameter δ eine nicht vernachlässigbare Wahrscheinlichkeit besteht, dass mehrere Listen $M_{DV(m),p,k}^{t=0}$ für unterschiedliche p gleich sind, was in einem gleichen Gewichtsvektor resultiert. Gleiche Gewichtsvektoren müssen jedoch nicht gleiche Relevanzschätzungen an der Stelle der Gewichtsvektoren bedeuten. Eine Differenzierung der Stützpunkte kann durch eine Abweichung des Stützpunktes im DVR, im Relevanzraum oder in beiden Räumen erfolgen. D.h. eine Differenzierung bei gleichem Gewichtsvektor wird auch durch abweichende Relevanzschätzungen erreicht, die z.B. durch folgende Vorgehensweisen erzeugt werden können:

1) Verwendung einer nicht streng lokalen instanzbasierten Approximation, bei der z.B. $M_{DV(m),p,k}^{t=0}$ sowie die Stimuluslisten der benachbarten Neurone aus der Nachbarschaftsliste $N(d_G=1 \mid G_{D,bew}^{t=0})_k$ herangezogen werden.
2) Verwendung eines nicht-deterministischen instanzbasierten Approximationsverfahrens auf der Basis von $M_{DV(m),p,k}^{t=0}$.

Im Hinblick auf die Aufgabe der Bootstrap-GNG-SOMs bedeutet jedoch das Übereinstimmen der Gewichtsvektoren und der Relevanzschätzungen einzelner Neurone keine wirklich problematische Situation. Dies ergibt sich daraus, dass die einzelnen Bootstrapmodelle $AM(rel(x) \mid N_{D,p,bew}^{t=0})$ erzeugt werden, um differenzierende Relevanzschätzungen an der Stelle von noch nicht durch den Agenten bewerteten Dokumentvektoren zu erzeugen. Eine prototypbasierte LWR-Schätzung verwendet jedoch alle Neurone in den Modellen, sodass unterschiedliche Relevanzschätzungen immer erzeugt werden,

wenn mindestens ein unterschiedlicher Stützpunkt vorliegt. Die Übereinstimmung einzelner Stützpunkte im DVR, im Relevanzraum oder in beiden Räumen ist somit für den Approximationszweck unerheblich, doch aus Gründen der größeren Differenzierung sollte immer eine Unterscheidung aller Komponenten angestrebt werden, auf die ein Einfluss besteht, wenn keine anderen Constraints dagegen sprechen. Der wichtigste Constraint ist die Speichereffizienz, da in allen Komponenten differenzierte Modelle viel zusätzlichen Speicherplatz benötigen, weil keine Referenzen auf bekannte Komponenten gebildet werden können. Innerhalb dieses und des nächsten Abschnittes soll jedoch von differenzierten Bootstrap-GNG-SOMs ausgegangen werden, während im Abschnitt 4.4.2.1.3.3) eine speichereffiziente Abweichung von diesem Prinzip beschrieben wird.

Neben der Bildung des Mittelwertsvektors können auch bestimmte Formen der lokalen Stimuluspräsentation in Verbindung mit Adaptionsprozessen verwendet werden. Hierzu wird aus einer lokalen Stimulusliste $M_{DV(m),p,k}^{t=0}$ ein Element zufällig mit Zurücklegen gezogen, und der Gewichtsvektor $w(x_D)_{p,k}^{t=0}$ wird einer SOM-Adaption unterzogen. Nachdem eine bestimmte Anzahl dieser streng lokalen Adaptionen erfolgte, wird der sich ergebende Gewichtsvektor als Ergebnis der Ableitung bzw. Nachadaption verwendet und in die Neuronenstruktur von $n_{D,p,k}^{t=0}$ aufgenommen. Durch die Zufallsauswahl handelt es sich quasi um eine stochastische Variante der Mittelwertsbildung, sodass erwartet werden kann, dass der sich ergebende Gewichtsvektor nicht wesentlich von dem deterministischen Mittelwertsvektor abweicht.

Eine nicht so strenge lokale Form der Nachadaption adaptiert nicht nur $w(x_D)_{p,k}^{t=0}$, sondern auch die Gewichtsvektoren der verbundenen Neurone, d.h. der Elemente aus der Nachbarschaftsmenge $N(d_G=1 \mid G_{D,bew}^{t=0})_k$. Bei dieser Vorgehensweise kann jedoch keine unabhängige Durchführung der Nachadaption auf Neuronenebene mehr durchgeführt werden, da ein Neuron seine Nachbarneurone beeinflusst. Aus diesem Grunde muss die Nachadaption einer Bootstrap-GNG-SOMs durch eine andere Stimulusauswahl-Operation durchgeführt werden, indem zunächst gleichverteilt zufällig ein Neuron $n_{D,p,k}^{t=0}$ als Zentrum der Adaption ausgewählt wird, gefolgt von der gleichverteilt zufälligen Auswahl eines Stimulus aus der lokalen Liste $M_{DV(m),p,k}^{t=0}$. Es folgt eine Adaption von $w(x_D)_{p,k}^{t=0}$ und der Gewichtsvektoren der Neurone aus $N(d_G=1 \mid G_{D,bew}^{t=0})_k$ mit unterschiedlichen Adaptionsparametern ε_s und ε_n.

Da die Verbindungsvektoren der Neurone $n_{D,p,k}^{t=0}$ der Bootstrap-GNG-SOMs im Vergleich zu $N_{D,bew}^{t=0}$ mit $C_{D,k}^{t=0}$ konstant bleiben sollen, werden jedoch alle Formen der Nachadaption ausgeschlossen, die Verbindungen zu anderen Neuronen aufbauen bzw. diese löschen, oder die Neuronen-Wachstums- bzw. Neuronen-Löschoperationen beinhalten.

Nach den δ unabhängigen Ableitungen der Bootstrap-GNG-SOMs $N_{D,p,bew}^{t=0}$ ergibt sich die Menge $BMN_{D,bew}^{t=0}$:

$$BMN_{D,bew}^{t=0} = \{N_{D,bew}^{t=0}, N_{D,p,bew}^{t=0} \mid p = 1, ..., \delta\}, \text{ mit}$$

$$N_{D,bew}^{t=0} = \{n_{D,k}^{t=0} = (w(x_D)_k^{t=0}, rel(w(x_D)_k^{t=0})\wedge, M_{DV,k}^{t=0}, M_{DV(m),k}^{t=0}, C_{D,k}^{t=0}) \mid k = 1, ..., \mu_{N,D}^{t=0}\},$$

$$N_{D,p,bew}^{t=0} = \{n_{D,p,k}^{t=0} = (w(x_D)_{p,k}^{t=0}, rel(w(x_D)_{p,k}^{t=0})\wedge, M_{DV,p,k}^{t=0}, M_{DV(m),p,k}^{t=0}, C_{D,k}^{t=0}) \mid k = 1, ..., \mu_{N,D}^{t=0}\}. \quad (597)$$

Danach wird die nächste Iteration t=1 eingeleitet, gefolgt von der Bestimmung der Primärmenge $DVM_{prim}^{t=1}$, deren Elemente $x_j^{t=1}$ mit Hilfe der GNG-SOMs aus $BNL_{D,bew}^{t=0}$ bewertet werden. Die Einzel-Relevanzschätzungen $rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,bew}^{t=0}))$ und $rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,p,bew}^{t=0}))$ werden zu Bootstrap-Gesamtschätzungen $rel(x_j^{t=1} \mid AM(rel(x) \mid N_{D,p,bew}^{t=0}), p = 1, ..., \delta)$ bzw. $rel(x_j^{t=1} \mid BMN_{D,bew}^{t=0})_{0,632bias}$ aggregiert, mit deren Hilfe die Sortierung nach fallenden Relevanzschätzungen in Form der Sekundärliste $DV_{sec}^{t=1}$ und Tertiärliste $DV_{ter}^{t=1}$ erfolgt. Nach der Präsentation der korrespondierenden Dokumente und der Bewertung durch den Agenten ergibt sich die Stimulusmenge $M_{DV(m)}^{t=1}$, die vereinigt mit $M_{DV(m)}^{t=0}$ die Menge $M_{DV(m)}^{t\leq1}$ ergibt.

Betrachtet werden soll in diesem Abschnitt eine konsequente Anwendung von SC-GNG-SOM-Nachadaptionen aller einmal erzeugten Bootstrap-GNG-SOMs, d.h. $N_{D,bew}^{t=0}$ wird zu $N_{D,bew}^{t=1}$ und für alle p wird $N_{D,p,bew}^{t=0}$ zu $N_{D,p,bew}^{t=1}$ aktualisiert. Die erste Form der Nachadaption ordnet jedem neuen Stimulus aus $M_{DV(m)}^{t=1}$ ein Neuron aus $N_{D,bew}^{t=0}$ zu, in dessen lokale Stimulusmenge es aufgenommen wird, wodurch zunächst weder die existierenden Gewichtsvektoren, die Verbindungsstruktur oder die Anzahl der Neurone in $N_{D,bew}^{t=0}$ mit $\mu_{N,D}^{t=0}$ verändert wird. Jedem Neuron kann im weiteren ein lokaler Fehlerwert wie ein Quantifizierungsfehler zugeordnet werden, wobei die Überschreitung über einen Fehlerschwellenwert dazu führen soll, dass lokal ein neuer Gewichtsvektor und somit ein neues Neuron erzeugt wird, wodurch sich letztendlich eine Neuonenanzahl von $\mu_{N,D}^{t=1}$ und eine angepasste Verbindungsstruktur ergibt.

Alternativen, wie eine standardmäßige, globale Präsentation von Stimuli aus $M_{DV(m)}^{t=1}$ und $M_{DV(m),p}^{t=0}$, jedoch mit unterschiedlichen Ziehungswahrscheinlichkeiten, gefolgt von Gewinner- und Nachbarschafts-Neuron-Adaptionen und dazwischen geschobenen Wachstumsphasen ist ebenso möglich, doch die Verwendung von lokalen Präsentationen mit der Erzeugung neuer Gewichtsvektoren und der lokalen Stimulus-Umverteilung ist bei SC-GNG-SOM effizienter und soll daher weiter betrachtet werden (siehe Bachelier (1998c[17])).

Als erster Schritt der Nachadaption der Bootstrap-GNG-SOMs $N_{D,p,bew}^{t=1}$ steht die Erzeugung individueller Stimuluslisten $M_{DV(m),p}^{t\leq1}$. Zu diesem Zweck wird aus der Menge der neuen Stimuli $M_{DV(m)}^{t=1}$ eine Bootstrapliste $M_{DV(m),p}^{t=1}$ durch Ziehen mit Zurücklegen erzeugt, die mit $M_{DV(m),p}^{t=0}$ vereinigt wird. Die Stimuli aus $M_{DV(m),p}^{t=1}$ werden wieder den Neuronen $n_{D,p,k}^{t=1}$ zugeordnet, die ihre lokale Stimulusliste $M_{DV(m),p,k}^{t\leq1}$ aufbauen.

Die Erzeugung neuer Neurone beim Aufbau von $N_{D,bew}^{t=1}$ und die damit einhergehende Veränderung der Verbindungsstruktur hat die Konsequenz, dass $N_{D,bew}^{t=1}$ nicht mehr bezüglich dieser Eigenschaften mit den Bootstrap-GNG-SOMs $N_{D,p,bew}^{t=1}$ kompatibel sein kann, wenn bei der Aktualisierung dieser GNG-SOMs die entsprechenden Vorgaben von $N_{D,bew}^{t=1}$ nicht verwendet werden. D.h. entweder wird zunächst $N_{D,bew}^{t=1}$ gebildet, und die δ GNG-SOMs $N_{D,p,bew}^{t=1}$ übernehmen die Anzahl der Neurone und die Verbindungsstruktur ohne Rücksicht auf die lokalen Fehler, die durch die Stimuluslisten $M_{DV(m),p,k}^{t\leq1}$ erzeugt werden, oder die δ GNG-SOMs $N_{D,p,bew}^{t=1}$ entwickeln sich auf Grund der Stimuluslisten $M_{DV(m),p,k}^{t\leq1}$ und der lokalen Fehler ohne Rücksicht auf die Anzahl der Neurone und die Verbindungsstruktur von $N_{D,bew}^{t=1}$. Da in diesem Abschnitt die zweite Option betrachtet wird, bedeutet dies, dass ab der Iteration t=1 die Bootstrap-GNG-SOMs $N_{D,p,bew}^{t=1}$ individuelle Verbindungsstrukturen in

Form von Verbindungsvektoren $C_{D,k,p}^{t=1}$ besitzen, genauso wie eine individuelle Anzahl von Neuronen mit $\mu_{N,D,p}^{t=1}$.

Der Aufbau der Bootstrap-GNG-SOMs soll auch die Adaption der bestehenden Gewichtsvektoren und nicht nur die Generierung neuer Neurone beinhalten, indem z.B. bei Stimulus-Umverteilungen der Gewichtsvektor, dem ein Stimulus neu zugeteilt wird, adaptiert wird, eventuell zusätzlich die Gewichtsvektoren der Neurone, die mit dem Gewinner-Neuron verbunden sind, sodass alle Neurone in den GNG-SOMs $N_{D,p,bew}^{t=1}$ auch individuelle Gewichtsvektoren besitzen. Es folgt die Aktualisierung der Relevanzschätzungen an den Stellen der aktualisierten Gewichtsvektoren durch eine streng lokale, instanzbasierte LWR-Approximation oder durch eine weniger lokalere Definition, bei der die Stimuli in den Listen der verbundenen Neurone ebenfalls beteiligt werden. Nach diesen Operationen ergibt sich eine Modellmenge $BMN_{D,bew}^{t=1}$ mit:

$$BMN_{D,bew}^{t=1} = \{N_{D,bew}^{t=1}, N_{D,p,bew}^{t=1} \mid p = 1, ..., \delta\}, \text{ mit}$$

$$N_{D,bew}^{t=1} = \{n_{D,k}^{t=1} = (w(x_D)_k^{t=1}, rel(w(x_D)_k^{t=1})\wedge, M_{DV,k}^{t=1}, M_{DV(m),k}^{t\leq 1}, C_{D,k}^{t=1}) \mid k = 1, ..., \mu_{N,D}^{t=1}\},$$

$$N_{D,p,bew}^{t=1} = \{n_{D,p,k}^{t=1} = (w(x_D)_{p,k}^{t=1}, rel(w(x_D)_{p,k}^{t=1})\wedge, M_{DV,p,k}^{t=1}, M_{DV(m),p,k}^{t\leq 1}, C_{D,k,p}^{t=1}) \mid k = 1, ..., \mu_{N,D,p}^{t=1}\}. \quad (598)$$

Es folgt die Einleitung der nachfolgenden Iteration bis ein Abbruchkriterium von Seiten des IRS erfolgt, oder bis der Agent die Interaktion beendet.

4.4.2.1.3.2) Iterations-spezifische Neuableitung von Bootstrap-GNG-SOMs

Bei der einmaligen Ableitung von Bootstrap-GNG-SOMs ergab sich die Problematik, dass in den nachfolgenden Iterationen jede dieser GNG-SOMs eine eigene Anzahl von Neuronen und eine eigene Verbindungsstruktur aufbauen muss, wenn der Constraint erfüllt werden soll, dass alle lokale Fehlermaße kleiner einem Schwellenwert sein sollen. In diesem Abschnitt wird der Fall betrachtet, dass im Iterationsverlauf die Gesamt-GNG-SOM $N_{D,bew}^{t}$ aktualisiert wird, und dass in jeder Iteration aus der aktuellen GNG-SOM δ Bootstrap-Varianten abgeleitet werden, unabhängig von den Bootstrap-Varianten der vorangegangenen Iteration. D.h. in der Initialisierungs-Iteration wird $N_{D,bew}^{t=0}$ on-scratch durch die Stimuli $M_{DV(m)}^{t=0}$ aufgebaut, wonach die δ Bootstrap-GNG-SOMs $N_{D,p,bew}^{t=0}$ aus $N_{D,bew}^{t=0}$ mit Hilfe von Bootstrap-Stimuluslisten $M_{DV(m),p}^{t=0}$ abgeleitet werden. Beim Übergang zu t=1 wird $N_{D,bew}^{t=0}$ zu $N_{D,bew}^{t=1}$ aktualisiert, die δ Bootstrap-GNG-SOMs $N_{D,p,bew}^{t=0}$ werden gelöscht, und die δ Bootstrap-GNG-SOMs $N_{D,p,bew}^{t=1}$ werden wieder aus $N_{D,bew}^{t=1}$ abgeleitet.

Nachdem $N_{D,bew}^{t=0}$ on-scratch aus $M_{DV(m)}^{t=0}$ erzeugt wurde, kann ein Aufbau von δ Bootstrap-Stimuluslisten $M_{DV,p}^{t=0}$ erfolgen, die durch zufälliges Ziehen mit Zurücklegen aus $M_{DV(m)}^{t=0}$ eingeleitet werden. Nachdem der Aufbau von $N_{D,bew}^{t=0}$ beendet wurde, können daraus δ Bootstrap-GNG-SOMs $N_{D,p,bew}^{t=0}$ abgeleitet werden, indem zunächst $N_{D,bew}^{t=0}$ kopiert und die Kopie zu $N_{D,p,bew}^{t=0}$ umbenannt wird. Die Ableitung erfolgt durch die Bootstrap-Stimulusliste $M_{DV(m),p}^{t=0}$, indem die lokalen Stimulusmengen $M_{DV(m),k}^{t=0}$ der Neurone $n_{D,p,k}^{t=0}$ ersetzt werden durch die lokale Stimulusliste $M_{DV(m),p,k}^{t=0}$. Um dies zu erreichen, erfolgt eine Löschung der Stimuli, die in der Bootstrap-Gesamtsti-

mulusliste $M_{DV(m),p}^{t=0}$ nicht mehr enthalten sind. Danach wird eine Vervielfältigung der Stimuli durchgeführt, die in der Gesamtliste $M_{DV(m),p}^{t=0}$ mehrfach vorliegen, d.h. existiert ein Stimulus in $M_{DV(m),p}^{t=0}$ z.B. dreimal, und ist der Stimulus Element der lokalen Liste $M_{DV(m),p,k}^{t=0}$, so werden noch zwei identische Stimuli in diese lokale Liste aufgenommen bzw. durch Referenz erzeugt.

Nachdem alle lokalen Listen $M_{DV(m),p,k}^{t=0}$ bezüglich $M_{DV(m),p}^{t=0}$ in $N_{D,p,bew}^{t=0}$ aktualisiert wurden, werden die Gewichtsvektoren aktualisiert und die Relevanzschätzungen an den Stellen der aktualisierten Gewichtsvektoren werden neu festgelegt. Für die Aktualisierung der Gewichtsvektoren $w(x_D)_{p,k}^{t=0}$ kommen neben der Bildung des Mittelwertsvektors der Dokumentvektoren-Komponenten der Stimuli aus $M_{DV(m),p,k}^{t=0}$ bzw. eine LWR-Variante diejenigen SC-GNG-SOM-Nachadaptionen in Frage, welche die gegebene Verbindungsstruktur inklusive der Anzahl der Neurone konstant halten. Beispielsweise kann eine nicht lokale Nachadaption verwendet werden, die im ersten Schritt gleichverteilt zufällig ein Neuron $n_{D,p,k}^{t=0}$ aus $N_{D,p,bew}^{t=0}$ auswählt, im zweiten Schritt ein Element aus der Liste $M_{DV(m),p,k}^{t=0}$ gleichverteilt zufällig mit Zurücklegen zieht, und die Dokumentvektor-Komponente des gezogenen Stimulus als Fixpunkt der Adaption für den Gewichtsvektor $w(x_D)_{p,k}^{t=0}$ und die Gewichtsvektoren der verbundenen Neurone aus $N(d_G=1 \mid G_{D,p,bew}^{t=0})_k$ verwendet. Nachdem eine bestimmte Anzahl dieser Nachadaptions-Operationen durchgeführt wurde, die als Funktion der Anzahl der Stimuli in $M_{DV,p}^{t=0}$ bestimmt wird, wird die Gewichtsvektor-Nachadaption beendet, und die sich ergebenden Gewichtsvektor-Positionen werden konstant gehalten für die Anwendung des Modells $AM(rel(x) \mid N_{D,p,bew}^{t=0})$ in der nachfolgenden Iteration t=1. Es folgt noch die Aktualisierung der Relevanzschätzungen $rel(w(x_D)_{p,k}^{t=0})^\wedge$, indem z.B. streng lokal eine instanzbasierte LWR-Approximation auf der Basis der Stimulusliste $M_{DV(m),p,k}^{t=0}$ durchgeführt wird, oder weniger streng lokal auf der Vereinigung der Stimuluslisten der Neurone aus der Neuronen-Nachbarschaftsmenge $N(d_G \leq 1 \mid G_{D,p,bew}^{t=0})_k$. Abschließend folgt eine Prüfung der Zugehörigkeit der bewerteten Stimuli aus $M_{DV(m),p}^{t=0}$ und der unbewerteten Stimuli aus $M_{DV,p}^{t=0}$ auf der Basis ihrer Dokumentvektoren zu den Neuronen mit eventuell erforderlichen lokalen Umverteilungen, da durch die Adaptionsoperationen die Voronoi-Regionen verändert wurden. Dies ist nur bei den Verfahren notwendig, die eine SOM-Adaption verwenden, erübrigt sich jedoch bei den Verfahren, die einen ungewichteten oder (LWR-)gewichteten Mittelwert der Dokumentvektoren in einer lokalen Stimulusmenge $M_{DV(m),p,k}^{t=0}$ verwenden.

Nach diesen Operationen ergibt sich die Menge der GNG-SOMs:

$$BMN_{D,bew}^{t=0} = \{N_{D,bew}^{t=0}, N_{D,p,bew}^{t=0} \mid p = 1, ..., \delta\}, \text{ mit}$$

$$N_{D,bew}^{t=0} = \{n_{D,k}^{t=0} = (w(x_D)_k^{t=0}, rel(w(x_D)_k^{t=0})^\wedge, M_{DV,k}^{t=0}, M_{DV(m),k}^{t=0}, C_{D,k}^{t=0}) \mid k = 1, ..., \mu_{N,D}^{t=0}\},$$

$$N_{D,p,bew}^{t=0} = \{n_{D,p,k}^{t=0} = (w(x_D)_{p,k}^{t=0}, rel(w(x_D)_{p,k}^{t=0})^\wedge, M_{DV,p,k}^{t=0}, M_{DV(m),p,k}^{t=0}, C_{D,k}^{t=0}) \mid k = 1, ..., \mu_{N,D}^{t=0}\}. \quad (599)$$

Es folgt der Übergang zur Iteration t=1, bei der zunächst eine primäre Ergebnismenge $DVM_{prim}^{t=1}$ gebildet wird, deren Elemente $x_j^{t=1}$ mit Hilfe der GNG-SOMs aus $BNL_{D,bew}^{t=0}$ bewertet werden. Letztendlich ergibt sich nach der Bewertung durch den Agenten eine iterations-spezifische Stimulusmenge $M_{DV(m)}^{t=1}$, die vereinigt mit $M_{DV(m)}^{t=0}$ die Menge $M_{DV(m)}^{t\leq 1}$ ergibt.

Darauf folgt die Aktualisierung von $N_{D,bew}^{t=0}$ zu $N_{D,bew}^{t=1}$, indem eine SC-GNG-SOM-Nachadaption angewendet wird, welche die Anzahl der Neurone sowie die Verbindungsstruktur beliebig unter dem Constraint verändern kann, dass die Fehlermaße, die jedem Neuron in Abhängigkeit von seinem Gewichtsvektor und seiner lokalen Stimulusmenge, kleiner-gleich einem Fehlerschwellenwert ist. Denkbar ist zunächst eine Grobadapton, indem die Elemente aus $M_{DV(m)}^{t=1}$ nach einer zufälligen Ziehung ohne Zurücklegen global präsentiert werden, gefolgt von einer Gewinner- und Nachbarschafts-Neuron-Adaption der entsprechenden Gewichtsvektoren. Sind somit alle neuen Stimuli einmal präsentiert worden und einem Neuron zugeordnet worden (Single-Pass-Clusterung), so erfolgt eine Nachadaption, die zunächst prüft, ob lokale Fehlermaße existieren, die größer als der verwendete Schwellenwert sind. Ist dies der Fall, so werden neue Gewichtsvektoren und somit neue Neurone eingefügt, gefolgt von der lokalen Umverteilung von Stimuli, bis alle existierenden Neurone kompatibel mit dem Fehler-Constraint sind.

Nachdem $N_{D,bew}^{t=1}$ erzeugt wurde, werden die alten Bootstrap-GNG-SOMs $N_{D,p,bew}^{t=0}$ gelöscht, und neue GNG-SOMs $N_{D,p,bew}^{t=1}$ werden aus $N_{D,bew}^{t=1}$ abgeleitet, indem $N_{D,bew}^{t=1}$ δ mal kopiert wird, und indem jedem dieser GNG-SOMs $N_{D,p,bew}^{t=0}$ eine Bootstrapliste $M_{DV(m),p}^{t=1}$ zugeordnet wird, deren Elemente den Neuronen $n_{D,p,k}^{t=1}$ zugeordnet werden. Es folgt die gleiche Nachadaptionsart, wie in der Initialisierungs-Iteration, bei der die Gewichtsvektoren-Positionen der Neurone entsprechend den lokalen Stimuluslisten $M_{DV(m),p,k}^{t=1}$ zu $w(x_D)_{p,k}^{t=1}$ verändert werden, gefolgt von einer Relevanzschätzung an der Stelle der neuen Gewichtsvektoren. Auf diese Art entsteht die GNG-SOM-Menge $BMN_{D,bew}^{t=1} = \{N_{D,bew}^{t=1}, N_{D,p,bew}^{t=1} \mid p = 1, ..., \delta\}$, auf der die Modelle der nachfolgenden Iteration basieren.

4.4.2.1.3.3) Bootstrap-GNG-SOMs durch Aktualisierung von Relevanzschätzungen

Insbesondere das im vorangegangenen Abschnitt dargestellte Verfahren der iterations-spezifischen Ableitung von Bootstrapmodellen, lässt sich vereinfachen, indem auf eine Aktualisierung der Gewichtsvektoren der abgeleiteten Bootstrap-Modelle verzichtet wird, was eine ganz erhebliche Verbesserung der Speichereffizienz darstellt. Wie im Abschnitt 4.4.2) bereits beschrieben wurde, benötigt ein prototypbasiertes Modell mit $\mu_{N,D}$ Neuronen $(n + 1) * \mu_{N,D}$ reelle Komponenten in Form der Gewichtsvektoren und der Relevanzwerte, um seine Stützpunkte zu repräsentieren, was bei einem Bootstrap-basierten Verfahren $((n + 1) * \delta) * \mu_{N,D}$ Komponenten bedeutet. Besitzten alle δ Bootstrapmodelle jedoch den gleichen Gewichtsvektor, d.h. den gleichen Stützpunkt im DVR, und unterscheiden sie sich nur durch ihre Relevanzschätzungen, d.h. den Stützpunkt im Relevanzraum, so können δ Bootstrapmodelle mit $(n + \delta) * \mu_{N,D}$ reellen Komponenten repräsentiert werden.

In der Initialisierungs-Iteration wird $N_{D,bew}^{t=0}$ on-scratch aus $M_{DV(m)}^{t=0}$ erzeugt, gefolgt von der δ'fachen Kopie dieser GNG-SOM, wobei die Gewichtsvektoren der Kopien als Referenz auf die korrespondierenden Gewichtsvektoren der Neurone aus $N_{D,bew}^{t=0}$ repräsentiert werden. Es werden wieder δ Bootstrap-Stimuluslisten $M_{DV(m),p}^{t=0}$ erzeugt, die den Kopien $N_{D,p,bew}^{t=0}$ zugeordnet werden. Die Elemente der Listen werden auf die Neurone $n_{D,p,k}^{t=0}$ verteilt, wodurch die lokalen Listen $M_{DV(m),p,k}^{t=0}$ gebildet werden. Im weiteren werden die Relevanzschätzungen der Neurone $n_{D,p,k}^{t=0}$ auf der Basis von $w(x_D)_k^{t=0}$ und den lokalen Stimuluslisten bestimmt, wobei zwei Möglichkeiten existieren:

1) Streng lokale Formulierung, die nur $M_{DV(m),p,k}^{t=0}$ verwendet:

$$rel(w(x_D)_k^{t=0} \mid AM(rel(x) \mid M_{DV(m),p,k}^{t=0})) = 1/\nu_{p,k} \sum_k h(d_{DVR}(x_{p,k,j}^{t=0}, w(x_D)_k^{t=0})) * rel(x_{p,k,j}^{t=0}), \text{ mit}$$
$$\nu_{p,k} = \sum_k h(d_{DVR}(x_{p,k,j}^{t=0}, w(x_D)_k^{t=0})), \forall x_{p,k,j}^{t=0} \in M_{DV(m),p,k}^{t=0}. \quad (600)$$

2) Weniger strenge lokale Formulierung, welche die Stimuluslisten $M_{DV(m),p,q}^{t=0}$ der $n_{p,q}^{t=0}$ Neurone aus der Nachbarschaftsmenge $N(d_G \leq 1 \mid G_{D,p,bew}^{t=0})_k$ verwendet:

$$rel(w(x_D)_k^{t=0} \mid AM(rel(x) \mid N(d_G \leq 1 \mid G_{D,p,bew}^{t=0})_k)) =$$
$$1/\nu_{p,k} \sum_k h(d_{DVR}(x_{p,k,j}^{t=0}, w(x_D)_k^{t=0})) * rel(x_{p,k,j}^{t=0}), \text{ mit}$$
$$\nu_{p,k} = \sum_k h(d_{DVR}(x_{p,k,j}^{t=0}, w(x_D)_k^{t=0})),$$
$$\forall x_{p,q,j}^{t=0} \in M_{DV(m),p,q}^{t=0}, \forall n_{p,q}^{t=0} \in N(d_G \leq 1 \mid G_{D,p,bew}^{t=0})_k. \quad (601)$$

Eine GNG-SOM-Menge $BMN_{D,bew}^{t=0}$ lässt sich somit unter Berücksichtigung der gleichen Verbindungsstrukturen formulieren:

$$BMN_{D,bew}^{t=0} = \{N_{D,bew}^{t=0}, N_{D,p,bew}^{t=0} \mid p = 1, ..., \delta\}, \text{ mit}$$
$$N_{D,bew}^{t=0} = \{n_{D,k}^{t=0} = (w(x_D)_k^{t=0}, rel(w(x_D)_k^{t=0} \mid M_{DV(m),k}^{t=0})\wedge, M_{DV,k}^{t=0}, M_{DV(m),k}^{t=0}, C_{D,k}^{t=0}) \mid$$
$$k = 1, ..., \mu_{N,D}^{t=0}\},$$
$$N_{D,p,bew}^{t=0} = \{n_{D,p,k}^{t=0} = (w(x_D)_k^{t=0}, rel(w(x_D)_k^{t=0})_p\wedge, M_{DV,k}^{t=0}, M_{DV(m),p,k}^{t=0}, C_{D,k}^{t=0}) \mid$$
$$k = 1, ..., \mu_{N,D}^{t=0}\}, \text{ mit}$$
$$rel(w(x_D)_k^{t=0})_p\wedge \in \{rel(w(x_D)_k^{t=0} \mid AM(rel(x) \mid M_{DV,p,k}^{t=0})),$$
$$rel(w(x_D)_k^{t=0} \mid AM(rel(x) \mid N(d_G \leq 1 \mid G_{D,p,bew}^{t=0})_k))\}. \quad (602)$$

Durch die grundlegenden Mehrfachverwendungen der Gewichtsvektoren und der Verbindungsvektoren besteht die Möglichkeit, die Relevanzschäzungen in die Struktur der Neurone des Ursprungs-GNG-SOM zu integrieren. Werden die Bootstrap-Stimuluslisten $M_{DV(m),p,k}^{t=0}$ für weitere Operationen nicht mehr benötigt, so können sie nach der Relevanzschätzung gelöscht werden, wobei die Relevanzschätzungen dann vereinfacht $rel(w(x_D)_k^{t=0})_p\wedge$ bezeichnet werden können, unabhängig, ob die Relevanzschätzung streng lokal durch $M_{DV(m),p,k}^{t=0}$ oder durch die Bootstrap-Stimuluslisten der Neurone aus $N(d_G \leq 1 \mid G_D^{t=0})_k$ definiert wird. Ein Neuron $n_{D,k}^{t=0}$ besteht somit in dieser Formulierung aus einem Gewichtsvektor $w(x_D)_k^{t=0}$, einer lokalen Dokumentvektorenmenge $M_{DV,k}^{t=0}$, einer lokalen Stimulusmenge $M_{DV(m),k}^{t=0}$, einem Verbindungsvektor $C_{D,k}^{t=0}$, der Relevanzschätzung $rel(w(x_D)_k^{t=0})\wedge$, die auf der lokalen Stimulusmenge $M_{DV(m),k}^{t=0}$ basiert, sowie aus δ Relevanzschätzungen $rel(w(x_D)_k^{t=0})_p\wedge$, die auf den gelöschten, lokalen Stimulusmengen $M_{DV(m),p,k}^{t=0}$ basieren:

$$N_{D,bew}^{t=0} = \{n_{D,k}^{t=0} = (w(x_D)_k^{t=0}, rel(w(x_D)_k^{t=0})\wedge, rel(w(x_D)_k^{t=0})_p\wedge, M_{DV,k}^{t=0}, M_{DV(m),k}^{t=0}, C_{D,k}^{t=0} \mid$$
$$p = 1, ..., \delta) \mid k = 1, ..., \mu_{N,D}^{t=0}\}. \quad (603)$$

Es ergibt sich somit eine Approximationsmodell-Polyrepräsentation, die auf einer Relevanzschätzungs-Polyrepräsentation basiert, d.h. auf einer Stützpunkt-Polyrepräsentation im Relevanzraum anstatt auf einer Stützpunkt-Polyrepräsentation im DVR und im Relevanzraum. Da die unterschiedlichen Relevanz-

schätzungen durch Stimulus-Bootstraplisten erzeugt wurden, kann diese Vorgehensweise aber auch als Spezialfall einer Stimulus- und somit einer Stützpunkt-Polyrepräsentation verstanden werden.

In der nächsten Iteration t=1 wird zunächst die Primärmenge $DVM_{prim}^{t=1}$ ermittelt, deren Elemente $x_j^{t=1}$ durch die δ+1 Modelle AM(rel(x) | $N_{D,bew}^{t=0}$) und AM(rel(x) | $N_{D,p,bew}^{t=0}$), p = 1, ..., δ, bewertet werden. Diese Relevanzschätzung basiert auf der Distanz zwischen $x_j^{t=1}$ und den Gewichtsvektoren $w(x_D)_k^{t=0}$, die bei allen GNG-SOMs gleich sind. D.h. es wird genau einmal die $\mu_{N,D}^{t=0}$ Distanzen $d_{DVR}(x_j^{t=1}, w(x_D)_k^{t=0})$ berechnet, die in allen δ+1 Relevanzschätzungen wieder verwendet werden. Auf diese Weise ist die Aktualisierung der Relevanzschätzungen bei den GNG-SOMs nicht nur eine erhebliche Effizienzsteigerung bezüglich des Speicherplatzes, sondern auch eine erhebliche Effizienzsteigerung bezüglich der Rechenzeit, die benötigt wird, um die Dokumentvektoren der primären Ergebnismenge zu bewerten. Die δ+1 Einzelschätzungen werden daraufhin zu einer Bootstrap-Gesamtschätzung bzw. einer 0,632-Bias-korrigierten Gesamtschätzung aggregiert, auf der die Sortierung der Dokumentvektoren nach fallender Relevanzschätzung basiert. Es ergibt sich die Sekundärliste $DV_{sec}^{t=1}$ und die Tertiärliste $DV_{ter}^{t=1}$, und nach der Präsentation der korrespondierenden Dokumente und der Bewertung durch den Agenten die Stimulusmenge $M_{DV(m)}^{t=1}$, die vereinigt mit $M_{DV(m)}^{t=0}$ die Menge $M_{DV(m)}^{t\leq1}$ ergibt.

Wird die iterations-interne Ableitung der Bootstrap-GNG-SOMs aus einem Ursprungs-GNG-SOM unterstellt, so wird $N_{D,bew}^{t=0}$ zu $N_{D,bew}^{t=1}$ aktualisiert, indem zunächst die alten δ+1 Relevanzschätzungen $rel(w(x_D)_k^{t=0})^\wedge$, $rel(w(x_D)_k^{t=0})_p^\wedge$, p = 1, ..., δ, pro Neuron gelöscht werden. Es folgt die Verteilung der neuen Stimuli aus $M_{DV(m)}^{t=1}$ auf die Neurone, wodurch die Stimulusmengen $M_{DV(m),k}^{t\leq1}$ entstehen, mit denen die Aktualisierung der Gewichtsvektoren $w(x_D)_k^{t=0}$ zu $w(x_D)_k^{t=1}$ durchgeführt wird. Dabei werden Nachadaptionsverfahren verwendet, bei denen die Veränderung der Neuronenanzahl und die Verbindungsstruktur erlaubt wird, sodass nach dieser Nachadaption $\mu_{N,D}^{t=1}$ Neurone $n_{D,k}^{t=1}$ vorliegen. Es folgt die temporäre Erzeugung von globalen Bootstraplisten $M_{DV(m),p}^{t\leq1}$ durch Ziehungen aus $M_{DV(m)}^{t\leq1}$ bzw. von lokalen Bootstraplisten $M_{DV(m),p,k}^{t\leq1}$ durch Ziehungen aus $M_{DV(m),k}^{t\leq1}$, sodass sich bis zu diesem Zeitpunkt die Struktur ergibt:

$$N_{D,bew}^{t=1} = \{n_{D,k}^{t=1} = (w(x_D)_k^{t=1}, M_{DV,k}^{t=1}, M_{DV(m),k}^{t\leq1}, M_{DV(m),p,k}^{t\leq1}, C_{D,k}^{t=1} \mid p = 1, ..., \delta) \mid k = 1, ..., \mu_{N,D}^{t=1}\}. \quad (604)$$

Mit Hilfe der Stimulusmenge $M_{DV(m),k}^{t\leq1}$ wird die Relevanzschätzung $rel(w(x_D)_k^{t=1})^\wedge$ erzeugt, und bei einer streng lokalen Formulierung wird mit Hilfe einer der Stimuluslisten $M_{DV(m),p,k}^{t\leq1}$ die Schätzung $rel(w(x_D)_k^{t=1})_p^\wedge$ durch jeweils ein LWR-Approximationsverfahren erzeugt. Nach der Relevanzschätzung der Bootstrap-GNG-SOMs werden die Stimuluslisten nicht mehr benötigt, sodass sie gelöscht werden können und sich die folgende Struktur ergibt:

$$N_{D,bew}^{t=1} = \{n_{D,k}^{t=1} = (w(x_D)_k^{t=1}, rel(w(x_D)_k^{t=1})^\wedge, rel(w(x_D)_k^{t=1})_p^\wedge, M_{DV,k}^{t=1}, M_{DV(m),k}^{t\leq1}, C_{D,k}^{t=1} \mid p = 1, ..., \delta) \mid k = 1, ..., \mu_{N,D}^{t=1}\}. \quad (605)$$

4.4.2.1.3.4) Relevanzschätzungs-Dichtefunktion

In diesem Abschnitt wird das Konzept der Erzeugung einer Menge von Bootstrap-GNG-SOMs durch differenzierende Relevanzschätzungen an einer Menge von gleichen Stützpunkten im DVR, den Gewichtsvektoren $w(x_D)_k^t$, weiter abstrahiert, indem aus einer vorliegenden Menge von Bootstrap-Einzelschätzungen $rel(w(x_D)_k^t)_p\wedge$, p = 1, ..., δ, eine stufenförmige bzw. geglättete, stetige Relevanzschätzungs-Dichtefunktion erzeugt wird. Betrachtet wird das Neuron $n_{D,k}^t$, in dessen Datenstruktur die δ+1 Relevanzschätzungen $rel(w(x_D)_k^t)\wedge$, $rel(w(x_D)_k^t)_p\wedge$, p = 1, ..., δ, vorliegen, wobei die dargestellte Vorgehensweise für alle anderen $\mu_{N,D}^t$ Neurone von $N_{D,bew}^t$ durchgeführt werden muss.

Im ersten Schritt wird die kleinste und die größte Relevanzschätzung ermittelt, die mit $rel(w(x_D)_k^t)_{min}\wedge$ und $rel(w(x_D)_k^t)_{max}\wedge$ bezeichnet werden. Die kleinste Schätzung wird verwendet, um die linke Grenze und die größte Schätzung wird verwendet um die rechte Grenze eines Relevanz-Intervalls $IRel_k^t$ für das Neuron $n_{D,k}^t$ zu bilden:

$$IRel_k^t = [rel(w(x_D)_k^t)_{min}\wedge, rel(w(x_D)_k^t)_{max}\wedge], \text{ mit}$$
$$rel(w(x_D)_k^t)_{min}\wedge = \min\{rel(w(x_D)_k^t)\wedge, rel(w(x_D)_k^t)_p\wedge \mid p = 1, ..., \delta\},$$
$$rel(w(x_D)_k^t)_{max}\wedge = \max\{rel(w(x_D)_k^t)\wedge, rel(w(x_D)_k^t)_p\wedge \mid p = 1, ..., \delta\}. \quad (606)$$

Das Gesamtintervall wird in γ gleich breite Teilintervalle $IRel(l)_k^t$, l = 1, ..., γ, zerlegt, das den Durchmesser Δrel_k^t besitzt:

$$\Delta rel_k^t = (rel(w(x_D)_k^t)_{max}\wedge - rel(w(x_D)_k^t)_{min}\wedge)/\gamma. \quad (607)$$

Es ergibt sich die geordnete Liste $IRelL_k^t$ der Teilintervalle:

$$IRelL_k^t = (IRel(1)_k^t \mid l = 1, ..., \gamma), \text{ mit}$$
$$IRel(1)_k^t = [rel(w(x_D)_k^t)_{min}\wedge, rel(w(x_D)_k^t)_{min}\wedge + \Delta rel_k^t[,$$
$$IRel(2)_k^t = [rel(w(x_D)_k^t)_{min}\wedge + \Delta rel_k^t, rel(w(x_D)_k^t)_{min}\wedge + 2 * \Delta rel_k^t[,$$
$$...,$$
$$IRel(\gamma)_k^t = [rel(w(x_D)_k^t)_{min}\wedge + (\gamma\text{-}1) * \Delta rel_k^t, rel(w(x_D)_k^t)_{max}\wedge]. \quad (608)$$

Für jedes der Teilintervalle $IRel(1)_k^t$ wird die Menge $RelM(l)_k^t$ der Relevanzwerte ermittelt, das innerhalb des Teilintervalls liegt:

$$RelM(l)_k^t = \{rel(w(x_D)_k^t)_p\wedge \in IRel(1)_k^t\}. \quad (609)$$

Die relative Anzahl der Elemente in einer Teilmenge $RelM(l)_k^t$ soll mit s(.) bezeichnet werden, d.h. die absolute Anzahl der Relevanzschätzungen pro Menge $RelM(l)_k^t$ wird dabei durch die Gesamtanzahl δ+1 der Relevanzschätzungen dividiert:

$$s(RelM(l)_k^t) := \#RelM(l)_k^t/(\delta+1). \quad (610)$$

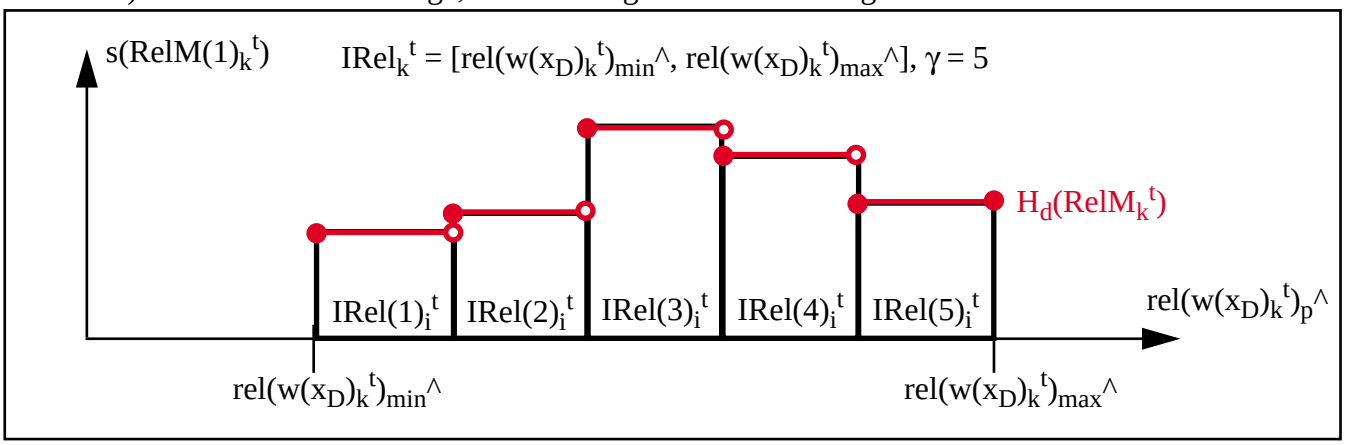
Die Häufigkeitsfunktion $H_d(RelM_k^t)$ ist eine Abbildung aus der Menge der reeelen Intervalle I auf N, indem jedem Teilintervall $IRel(1)_k^t$ die relative Anzahl $s(RelM(l)_k^t)$ der Relevanzwerte in der entspre-

chenden Teilmenge $RelM(l)_k^t$ zugeordnet wird:

$$H_d(RelM_k^t): I \rightarrow N: IRel(1)_k^t \mid\rightarrow s(RelM(1)_k^t). \quad (611)$$

Wird jedem Punkt eines Intervalls der s(.)-Wert zugeordnet, so gelangt man zu einer punktweise nicht stetigen Funktion, wobei die Grenzen so gewählt sind, wie die Teilintervalle $IRel(1)_k^t$ in der geordneten Liste $IRelL_k^t$ (siehe Abb. 123)).

Abb. 123) Punktweise nicht stetige, stufenförmige Relevanz-Häufigkeitsfunktion



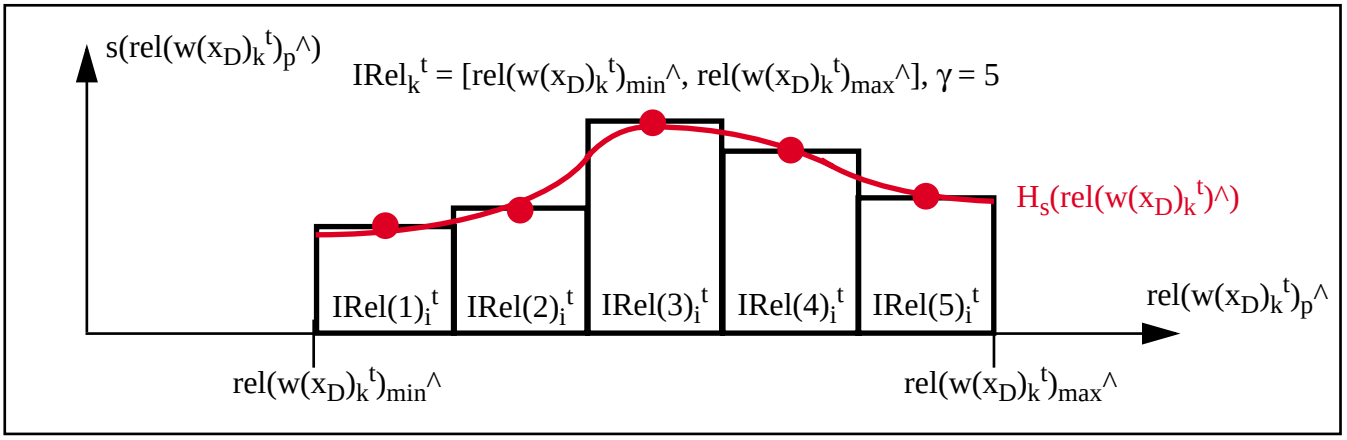
Durch die Verwendung der relativen Anzahl $s(RelM(1)_k^t)$ der Elemente, bezogen auf die Gesamtanzahl $\delta+1$ von Relevanzschätzungen, ist das Integral über $H_d(RelM_k^t)$ in den Grenzen $rel(w(x_D)_k^t)_{min}\hat{}$ und $rel(w(x_D)_k^t)_{max}\hat{}$ gleich Eins, sodass die Häufigkeitsfunktion als Wahrscheinlichkeitsdichte interpretiert werden kann.

Eine punktweise nicht stetige Dichtefunktion wird in eine geglättete, über der gesamten Definitionsmenge $[rel(w(x_D)_k^t)_{min}\hat{}, rel(w(x_D)_k^t)_{max}\hat{}]$ stetige Dichtefunktion umgewandelt, indem Stützpunkte ausgewählt werden, die als Input in ein Glättungs- oder ein Regressionsverfahren verwendet werden. D.h. zunächst wird eine Anzahl von Relevanzwerten aus dem Intervall $IRel_k^t$ spezifiziert, denen jeweils ihr s(.)-Wert zugeordnet wird. Sinnvoll ist es, aus jedem Teilintervall genau ein Relevanzwert zu spezifizieren, da alle Relevanzwerte innerhalb eines Teilintervalls den gleichen s(.)-Wert besitzen. Werden mehrere Relevanzwerte pro Teilintervall ausgewählt, so würde dies negative Auswirkungen auf das Regressionsverfahren besitzen. Denkbar wäre es, den Mittelpunkt der Teilintervalle auszuwählen und ihnen den s(.)-Wert des Teilintervalls zuzuordnen (siehe Abb. 124)). Eine solche stetige Relevanz-Dichtefunktion ist definiert als Abbildung von R^+ auf R^+, und ordnet jedem Relevanzwert $rel(w(x_D)_k)\hat{}$ aus dem Relevanz-Intervall $IRel_k^t$ einen reellen Häufigkeitswert $s(rel(w(x_D)_k)\hat{})$ zu:

$$H_s(rel(w(x_D)_k)\hat{}) : R^+ \rightarrow R^+: rel(w(x_D)_k)\hat{} \mid\rightarrow s(rel(w(x_D)_k)\hat{}), rel(w(x_D)_k)\hat{} \in IRel_k^t. \quad (612)$$

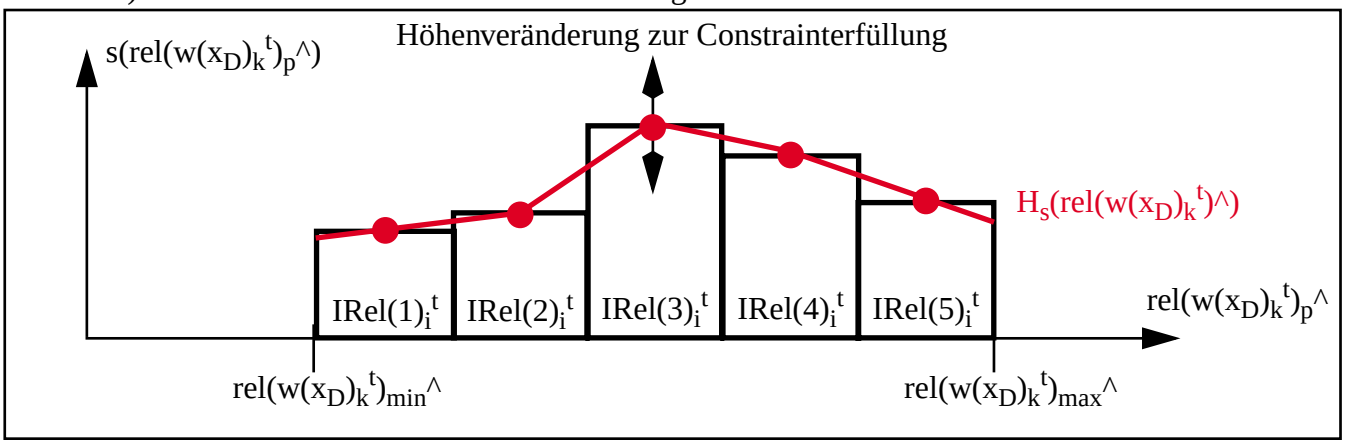
Der wichtigste Constraint, der bei der Regression erfüllt werden muss, ist der Dichte-Constraint, d.h. das Integral der Regressionsfunktion $H_s(rel(w(x_D)_k^t)\hat{})$ in den Grenzen $rel(w(x_D)_k^t)_{min}\hat{}$ und $rel(w(x_D)_k^t)_{max}\hat{}$ muss ebenfalls gleich bzw. näherungsweise gleich Eins sein. Auf diese Weise muss ein Optimierungs-Verfahren angewendet werden, um die Regressionsparameter diesem Constraint anzupassen, was diese Vorgehensweise aufwendig macht.

Abb. 124) Stützpunkte und Beispiel einer vollständig stetigen Relevanz-Dichtefunktion



Vereinfachend auf die Erfüllung des Constraint würde sich die Verwendung eines Linienzuges auswirken (siehe Abb. 125)), da zum einen das Integral einfach zu bestimmen ist, und zum anderen der Constraint durch einen einzigen Parameter optimierbar wäre, indem der gesamte Linienzug in seiner Höhe derart in s(.)-Richtung verschoben wird, dass das zugehörige Integral hinreichend durch Eins genähert wird.

Abb. 125) Relevanz-Dichtefunktion durch Linienzug



Mit Hilfe einer solchen Funktion $H_s(rel(w(x_D)_k^t)^)$ kann man durch Zufallsexperimente beliebig viele Relevanzschätzungen an der Stelle des Gewichtsvektors erzeugen, ohne dass die aufwendige Prozedur der Bildung einer Bootstrap-Stimulusliste und die Anwendung eines instanzbasierten Approximationsverfahrens durchgeführt werden muss. Da die Erzeugung einer Dichtefunktion jedoch selbst mit einem größeren Aufwand verbunden ist, ist dies nur dann sinnvoll, wenn geplant wird, eine Gesamt-Bootstrap-Relevanzschätzung mit sehr vielen Einzelschätzungen durchzuführen.

Werden zur Erzeugung der stetigen Relevanz-Dichtefunktion nur die Bootstrap-Relevanzschätzungen $rel(w(x_D)_k^t)_p^$, p = 1, ..., δ, verwendet, so kann mit $H_s(rel(w(x_D)_k^t)^)$ ein Spezialfall einer Bootstrap-Gesamtschätzung simuliert werden, bei dem $\delta \rightarrow \infty$ Bootstrap-Einzelschätzungen vorliegen. D.h. es kann mit dem Integral über $H_s(rel(w(x_D)_k^t)^)$ im Intervall $IRel_k^t$ der Fall beschrieben werden, dass $\delta \rightarrow \infty$ Bootstrap-Einzelschätzungen erzeugt und zu einer Bootstrap-Gesamtschätzung aggregiert werden:

$$rel(w(x_D)_k^t \mid H_s(rel(w(x_D)_k^t)\hat{}))) =$$
$$1/[rel(w(x_D)_k^t)_{max}\hat{} - rel(w(x_D)_k^t)_{min}\hat{}] * \int_{rel(k,min) \rightarrow rel(k,max)} H_s(rel(y)\hat{})\, dy, \text{ mit}$$
$$rel(k,min) := rel(w(x_D)_k^t)_{min}\hat{},\ rel(k,max) := rel(w(x_D)_k^t)_{max}\hat{}. \quad (613)$$

Dies lässt sich erweitern, indem mit der Relevanzschätzung auf der Basis aller Stimuli zunächst die Bias geschätzt wird mit

$$bias(w(x_D)_k^t)_{0,632}\hat{} =$$
$$(rel(w(x_D)_k^t \mid H_s(rel(w(x_D)_k^t)\hat{})) - rel(w(x_D)_k^t)\hat{})/0{,}632, \quad (614)$$

die in einer 0,632-bias-korrigierten Bootstrap-Gesamtschätzung verwendet wird:

$$rel(w(x_D)_k^t)_{0,632bias} =$$
$$rel(w(x_D)_k^t \mid H_s(rel(w(x_D)_k^t)\hat{})) - bias(w(x_D)_k^t)_{0,632}\hat{}. \quad (615)$$

4.4.2.2) Polyrepräsentierte Prototyp-Modelle bei klassifizierten Dokumentvektoren

Der Fall klassifizierter Dokumentvektoren wird dargestellt durch das Vorliegen einer Dokumentvektoren-Klassifikation in Form einer SC-GNG-SOM N_{DV}, welche die gesamte Dokumentvektorenmenge DV durch die Voronoi-Regionen $R\tilde{}_{DV,j}$ der einzelnen Neurone $n_{DV,j}$ zerlegt:

$$N_{DV} = \{n_{DV,j} = (w(x_{DV})_j, M_{DV,j}, C_{DV,j}) \mid j = 1, ..., \mu_{N,DV}\},$$
$$\bigcup_j M_{DV,j} = DV. \quad (616)$$

Aus dieser SC-GNG-SOM lässt sich ein prototypbasiertes Relevanz-Approximationsmodell direkt bilden, wenn einzelnen oder allen Neuronen ein Stützpunkt im Relevanzraum zugeordnet wird. Dies gelingt durch eine instanzbasierte Relevanzschätzung auf der Basis einer Dokumentvektorenmenge, die durch den Agenten bewertet wurde, und die nun Stimuli der Form m = (x, rel(x)) bilden. Eine Individualisierung, die sich auf einen speziellen Agenten bezieht, der eine Initialisierungs-Query $Q_i^{t=0}$ stellt, und die durch Einführung eines weiteren Indexes i erreicht werden kann, soll wie im Fall der Prototyp-Modellen bei unklassifizierten Dokumentvektoren entfallen. Eine individualisierte SC-GNG-SOM, bei der alle Neurone eine Relevanzschätzung zugeordnet bekommen, besitzt eine Struktur:

$$N_{DV}^t = \{n_{DV,j}^t = (w(x_{DV})_j^t, rel(w(x_{DV})_j^t)\hat{}, M_{DV,j}^t, M_{DV(m),j}^t, C_{DV,j}^t) \mid j = 1, ..., \mu_{N,DV}\}. \quad (617)$$

Eine Polyrepräsentation einer globalen GNG-SOM N_{DV} soll wie bei der Erzeugung einer GNG-SOM $N_{D,bew}^t$ aus einer wachsenden Stimulusmenge $M_{DV(m)}^{t=0}$, $M_{DV(m)}^{t\leq 1}$, ..., fallspezifisch erzeugt werden, d.h. aus den Bewertungen eines einzelnen Agenten werden Stimuli gebildet, die zur Bildung einer Menge von prototypbasierten Relevanz-Approximationsmodellen verwendet werden, deren Gewichtsvektoren und deren Verbindungsstruktur von N_{DV} übernommen wird, bzw. von N_{DV} abgeleitet wird. Im weiteren sollen nur Verfahren betrachtet werden, die weiter oben als abhängige Verfahren bezeichnet wurden (siehe Abschnitt 4.4.2.1.3)), die ausgehend von einem gegebenen Modell, dem Gesamt-GNG-SOM, Bootstrap-GNG-SOMs erzeugen. Die dort getroffene Verfahrensklassen-Unterscheidung zwischen ein-

maliger Erzeugung mit Identitätserhaltung und Ableitung in jeder Iteration kann auch im Kontext einer Dokumentvektor-Klassifikation getroffen werden.

Wie in den vorangegangenen Abschnitten dargestellt wurde, besitzen prototypbasierte Modelle mit ihren Komponenten Stimuli, Prototyp und Verbindungsstruktur, eine größere Variationsbreite bei der Erzeugung von Polyrepräsentationen, wobei die Verbindungsstruktur-Polyrepräsentation auch im weiteren ausgenommen werden soll. Grundsätzlich sollen Stimulus-Bootstraplisten $M_{DV(m),p}{}^{t}$ aus einer Stimulusmenge $M_{DV(m)}{}^{t}$ erzeugt und verwendet werden, d.h. es wird aus einer Stimulus-Monorepräsentation eine Stimulus-Polyrepräsentation erzeugt, mit der eine Prototyp-Monorepräsentation zu einer Prototyp-Polyrepräsentation erweitert wird. Der Fall einer Erzeugung einer Prototyp-Polyrepräsentation durch Resampling-Operationen auf der Menge der Prototypen bzw. Neurone ist hierzu eine Alternative, die im weiteren jedoch nicht mehr betrachtet werden soll, genauso wie die Verwendung der Erzeugungshistorie eines prototyp-basierten Modells. Weiterhin soll wie bislang eine Approximationsverfahren-Monorepräsentation verwendet werden, die auf einer Prototyp-Polyrepräsentation basierend eine Polyrepräsentation der Relevanzschätzungen von Dokumentvektoren aus einer Ergebnismenge bilden.

Da allgemein ein stützpunkt-basiertes Approximationsverfahren betrachtet wird, deren Stützpunkte eine Komponente im DVR und eine Komponente im Relevanzraum besitzen, ergibt sich kombinatorisch eine prototypbasierte Polyrepräsentation, wenn aus einem Prototypen
1) unterschiedliche Stützpunkte im DVR bei gleichem Stützpunkt im Relevanzraum
2) unterschiedliche Stützpunkte im Relevanzraum bei gleichem Stützpunkt im DVR
3) unterschiedliche Stützpunkte im DVR und im Relevanzraum erzeugt werden.

Der erste Fall wurde allgemein nicht betrachtet, während der zweite Fall in den Abschnitten 4.4.2.1.3.3) und 4.4.2.1.3.4) zugrunde gelegt wurde, bei denen nur die Relevanzschätzungen auf der Basis von Bootstrap-GNG-SOMs aktualisiert wurden unter Beibehaltung der Gewichtsvektoren eines Gesamt-GNG-SOM. Diese Vorgehensweise hat besondere Effizienzvorteile bezüglich der Speicherung der δ Bootstrap-GNG-SOMs und bezüglich der Anwendung der δ Bootstrap-Approximationsmodelle für die Relevanzschätzung von Dokumentvektoren aus Ergebnismengen. Der dritte Fall wird dagegen standardmäßig verwendet und soll vor dem zweiten Fall im weiteren dargestellt werden.

Ausgangspunkt ist in beiden Fällen ein Queryvektor $q_i{}^{t=0}$, der das Gewinner-Neuron $n_{DV,s(1|i)}{}^{t=0}$ aus N_{DV} bestimmt, in dessen Voronoi-Region $R^{\sim}_{DV,s(1|i)}$ er liegt. Durch die Zerlegung der Dokumentvektoren durch N_{DV} wird damit eine primäre Ergebnismenge bzw. ein Teil einer primären Ergebnismenge spezifiziert, indem im ersten Fall die Dokumentvektoren aus $M_{DV,s(1|i)}{}^{t=0}$ als Ergebnismenge $DVM(q_i{}^{t=0})$ verwendet werden, deren korrespondierende Dokumente dem Agenten zur Bewertung vorgelegt werden. Es ergibt sich somit die erste Stimulus-Gesamtmenge $M_{DV(m)}{}^{t=0}$, mit der eine streng lokal definierte Relevanzschätzung $rel(w(x_{DV})_{s(1|i)}{}^{t=0})^{\wedge}$ an der Stelle des Gewichtsvektors des Gewinner-Neurons erzeugt werden kann:

$$rel(w(x_{DV})_{s(1|i)}{}^{t=0})^{\wedge} = 1/\nu_{s(1|i)} \sum_j h(d_{DVR}(x_j{}^{t=0}, w(x_{DV})_{s(1|i)}{}^{t=0})) * rel(x_j{}^{t=0}),$$
$$\nu_{s(1|i)} = \sum_j h(d_{DVR}(x_j{}^{t=0}, w(x_{DV})_{s(1|i)}{}^{t=0})), \forall m_j{}^{t=0} \in M_{DV(m),s(1|i)}{}^{t=0} = M_{DV(m)}{}^{t=0}. \quad (618)$$

Da während des Retrievals keine Veränderung der Verbindungsstruktur von N_{DV} oder abgeleiteten GNG-SOMs erlaubt werden sollen, kann $C_{DV,s(1|i)}$ ohne einen Iterationsindex verwendet werden, sodass sich für das Gewinner-Neuron die Struktur ergibt:

$$n_{DV,s(1|i)}{}^{t=0} = (w(x_{DV})_{s(1|i)}{}^{t=0}, rel(w(x_{DV})_{s(1|i)}{}^{t=0})\wedge, M_{DV,s(1|i)}{}^{t=0}, M_{DV(m),s(1|i)}{}^{t=0}, C_{DV,s(1|i)}). \quad (619)$$

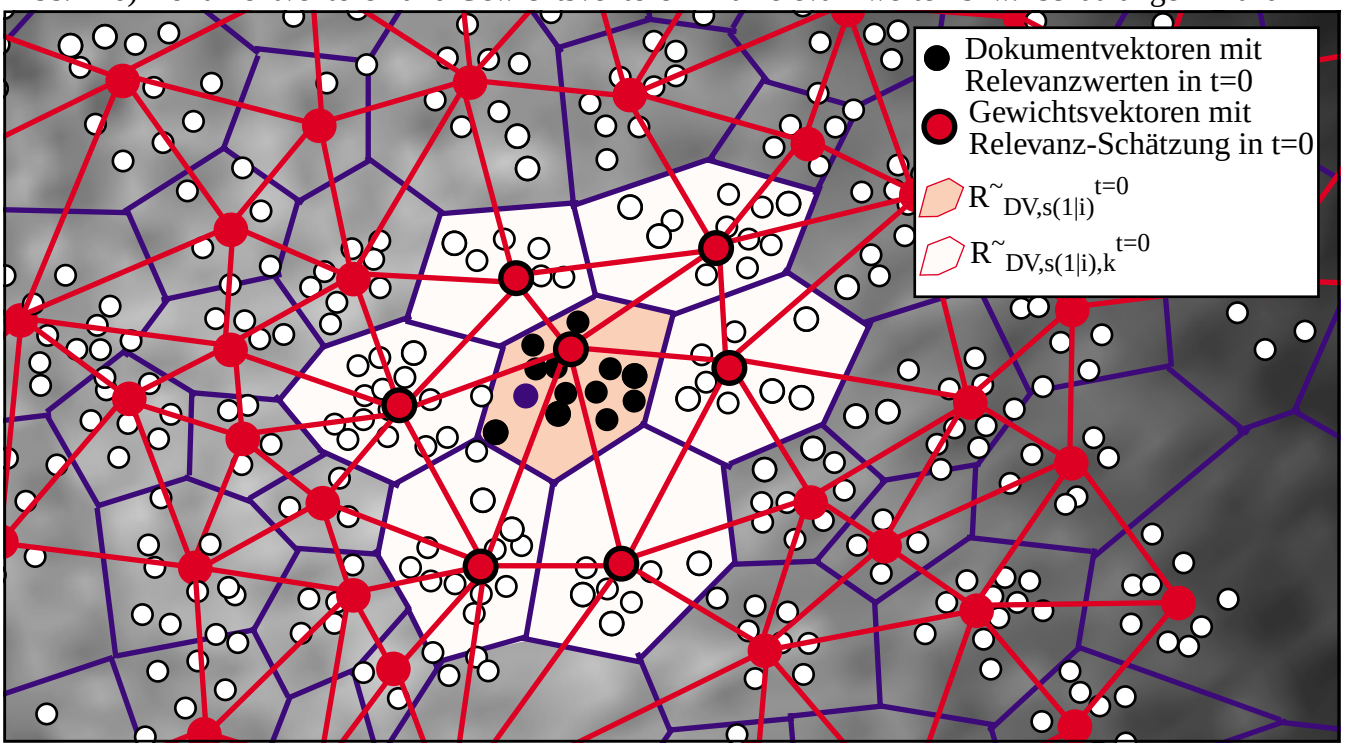
Es soll eine Neuronenmenge $\Delta N_{DV}{}^{t}$ eingeführt werden, die in jeder Iteration t die Neurone aufnimmt, deren Gewichtsvektoren innerhalb dieser Iteration adaptiert wurden. Für die Initialisierungs-Iteration bedeutet dies:

$$\Delta N_{DV}{}^{t=0} = \{n_{DV,s(1|i)}{}^{t=0}\}. \quad (620)$$

Neben der streng lokal definierten Relevanzschätzung können benachbarten Neuronen Relevanzschätzungen auch durch ein instanzbasiertes Approximationsverfahren auf der Basis von $M_{DV(m),s(1|i)}{}^{t=0}$ zugeordnet werden. Um eine lokale Vorgehensweise zu erhalten, sollen nur die unmittelbar verbundenen Neurone, d.h. die Elemente $n_{DV,s(1|i),k}$ aus der Nachbarschaftsmenge $N(d_G{=}1 \mid G_{DV})_{s(1|i)}{}^{t=0}$ eine Relevanzschätzung auf diese Weise erhalten (siehe Abb. 126)):

$$n_{DV,s(1|i),k}{}^{t=0} = (w(x_{DV})_{s(1|i),k}{}^{t=0}, rel(w(x_{DV})_{s(1|i),k}{}^{t=0})\wedge, M_{DV,s(1|i),k}{}^{t=0}, M_{DV(m),s(1|i),k}{}^{t=0}, C_{DV,s(1|i),k}),$$
$$\forall n_{DV,s(1|i),k} \in N(d_G{=}1 \mid G_{DV})_{s(1|i)}, \text{ mit}$$
$$rel(w(x_{DV})_{s(1|i),k}{}^{t=0})\wedge = 1/\nu_{s(1|i),k} \sum_j h(d_{DVR}(x_j{}^{t=0}, w(x_{DV})_{s(1|i),k}{}^{t=0})) * rel(x_j{}^{t=0}),$$
$$\nu_{s(1|i),k} = \sum_j h(d_{DVR}(x_j{}^{t=0}, w(x_{DV})_{s(1|i),k}{}^{t=0})), \forall m_j{}^{t=0} \in M_{DV(m),s(1|i)}{}^{t=0}. \quad (621)$$

Abb. 126) Dokumentvektoren und Gewichtsvektoren mit Relevanzwerten bzw. -schätzungen in t=0



In diesem Fall ergibt sich für die Initialisierungs-Iteration die Delta-Menge, die im weiteren Verlauf verwendet werden soll:

$$\Delta N_{DV}^{t=0} = N(d_G \leq 1 \mid G_{DV})_{s(1|i)}^{t=0}. \quad (622)$$

Es entsteht eine individuelle, globale GNG-SOM $N_{DV}^{t=0}$, die sich von der nicht-individuellen GNG-SOM N_{DV} dadurch unterscheidet, dass alle Neurone aus der Delta-Menge $\Delta N_{DV}^{t=0}$ einen abweichenden Gewichtsvektor und eine Relevanzschätzung besitzen. Diese Ersetzungs-Operation auf der Neuronen-Ebene eines GNG-SOMs soll in diesem Kontext durch die Bezeichung $\oplus$ beschrieben werden, d.h. die individuelle GNG-SOM $N_{DV}^{t=0}$ der Initialisierungs-Iteration ergibt sich durch die Verknüpfung der nicht-individuellen GNG-SOM N_{DV} und der Neuronenmenge, die in t=0 adaptiert und deren Datenstruktur um eine Relevanzschätzung erweitert wurde:

$$N_{DV}^{t=0} = N_{DV} \oplus \Delta N_{DV}^{t=0}. \quad (623)$$

Eine solche Darstellung definiert eine Prototyp-Monorepräsentation, die in eine Prototyp-Polyrepräsentation überführt werden soll. Wird die Ergebnismenge der Initialisierungs-Iteration bezeichnet mit $M_{DV(m)}^{t=0}$, so werden zunächst δ Bootstrap-Stimuluslisten $M_{DV(m),p}^{t=0}$ generiert, die Bootstrap-GNG-SOMs $N_{DV,p}^{t=0}$ festlegen, d.h. das Ursprungs-GNG-SOM $N_{DV}^{t=0}$ wird δ mal kopiert, die Kopien werden zu $N_{DV,p}^{t=0}$ umbenannt, und in allen Kopien wird die Stimulusmenge $M_{DV(m),s(1|i)}^{t=0}$ des Neurons $n_{DV,p,s(1|i)}^{t=0}$ durch die Bootstrap-Stimulusliste $M_{DV(m),p,s(1|i)}^{t=0}$ ersetzt. Ab diesem Zeitpunkt differenzieren sich die weiteren Vorgehensweisen, wobei zwischen der Modifikation der Stützpunkte im Relevanzraum unter Beibehaltung der Stützpunkte im Dokumentvektorenraum sowie der Modifikation im DVR und im Relevanzraum unterschieden wird.

4.4.2.2.1) Adaption der Stützpunkte im Relevanzraum und Erhaltung im DVR

Diese Form der Erzeugung einer Approximationsmodell-Polyrepräsentation kann als Vereinfachung der Modifikation im DVR und im Relevanzraum verstanden werden, die nicht nur recheneffizienter zu erzeugen ist, sondern auch wesentlich weniger Speicher benötigt und in der Anwendung wesentlich recheneffizienter ist, d.h. in der Zuordnung einer Relevanzschätzung zu einem Punkt im DVR, da Distanzen bzw. Kernel-Gewichte δ'fach wiederverwendet werden können.

Nachdem das Ursprungs-GNG-SOM $N_{DV}^{t=0}$ gebildet wurde, bei dem $n_{DV,s(1|i)}^{t=0}$ und seine Nachbarn Relevanzschätzungen erhalten haben, werden die δ Bootstrap-Stimuluslisten $M_{DV(m),p,s(1|i)}^{t=0}$ generiert, und $N_{DV}^{t=0}$ wird δ mal zu $N_{DV,p}^{t=0}$ kopiert und umbenannt. Für jede der Bootstrap-GNG-SOMs wird eine neue Relevanzschätzung $rel(w(x_{DV})_{s(1|i)}^{t=0})_p\wedge$ des Gewichtsvektors $w(x_{DV})_{s(1|i)}^{t=0}$ des Gewinner-Neurons $n_{DV,p,s(1|i)}^{t=0}$ erzeugt, indem eine instanzbasierte LWR-Approximation auf der Basis der Elemente von $M_{DV(m),p,s(1|i)}^{t=0}$ durchgeführt wird:

$$rel(w(x_{DV})_{s(1|i)}^{t=0})_p\wedge = rel(w(x_{DV})_{s(1|i)}^{t=0} \mid M_{DV(m),p,s(1|i)}^{t=0}) =$$
$$1/\nu_{p,s(1|i)} \sum_j h(d_{DVR}(x_j^{t=0}, w(x_{DV})_{s(1|i)}^{t=0})) * rel(x_j^{t=0}), \text{ mit}$$
$$\nu_{p,s(1|i)} = \sum_j h(d_{DVR}(x_j^{t=0}, w(x_{DV})_{s(1|i)}^{t=0})), \forall x_j^{t=0} \in M_{DV(m),p,s(1|i)}^{t=0}. \quad (624)$$

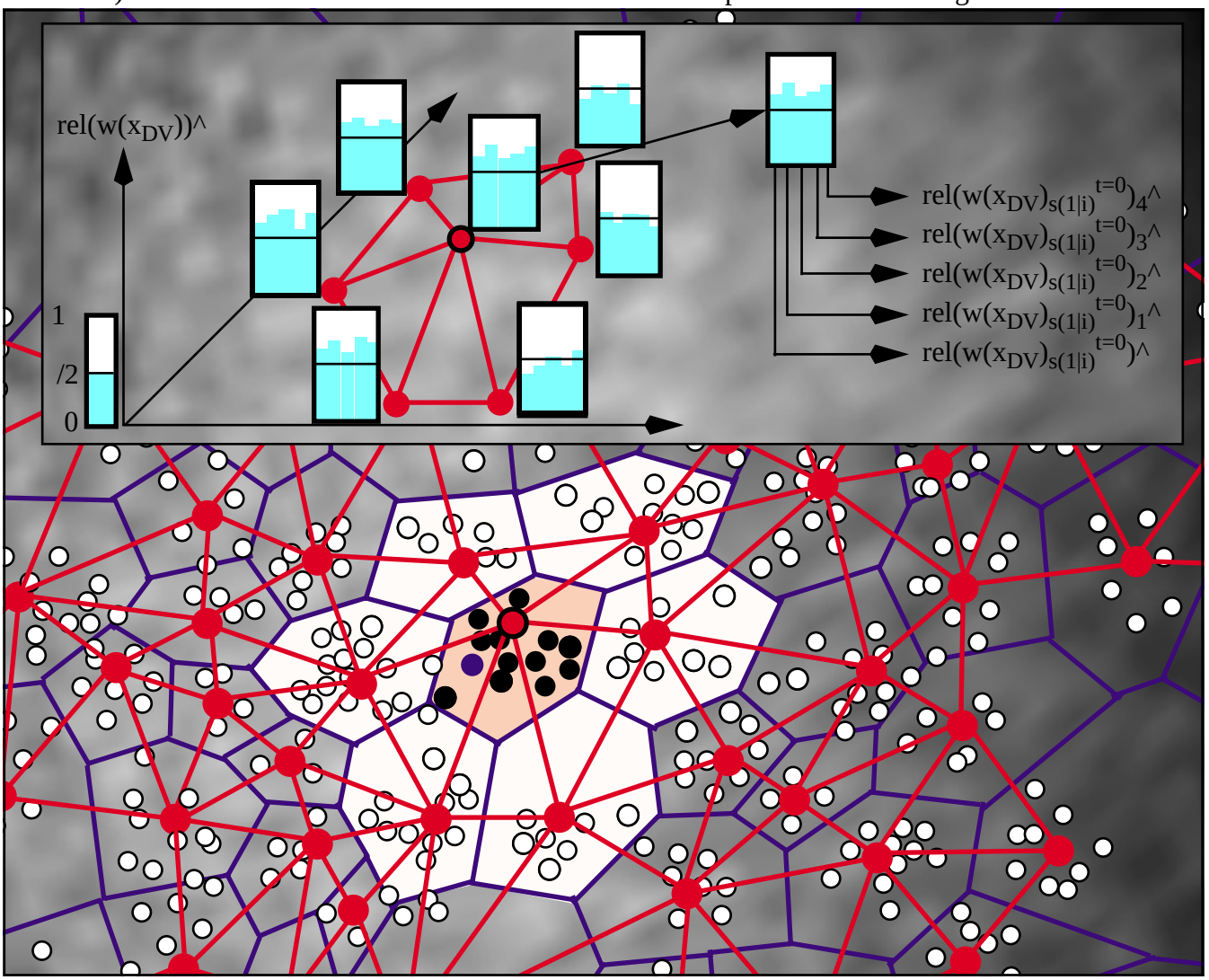
Für die Nachbarneurone aus $N(d_G=1 \mid G_{DV})_{s(1|i)}^{t=0}$ wird ebenfalls eine Relevanzschätzung an der Stelle der Gewichtsvektoren erfolgen, auch wenn in ihren Voronoi-Regionen keine Dokumentvektoren liegen, die bislang bewertet wurden. Auf diese Weise werden die korrespondierenden Elemente von $\Delta N_{DV}^{t=0}$ modifiziert, sodass die Bootstrap-Modelle komplett vorliegen.

Wie im Abschnitt 4.4.2.1.3.3) dargestellt wurde, können die Bootstrap-Stimuluslisten und die Relevanzschätzungen auch innerhalb der Datenstruktur der Neurone aus dem Ursprungsmodell $N_{DV}^{t=0}$ integriert werden, die modifiziert wurden. Beispielsweise ergibt sich dann das Gewinner-Neuron zu:

$$n_{DV,s(1|i)}^{t=0} = (w(x_{DV})_{s(1|i)}^{t=0}, rel(w(x_{DV})_{s(1|i)}^{t=0})\hat{}, rel(w(x_{DV})_{s(1|i)}^{t=0})_p\hat{}, M_{DV,s(1|i)}^{t=0}, M_{DV(m),s(1|i)}^{t=0}, M_{DV(m),p,s(1|i)}^{t=0}, C_{DV,s(1|i)} \mid p = 1, ..., \delta). \quad (625)$$

Diese Darstellung lässt sich visualisieren, indem den Gewichtsvektoren der Neurone in $N(d_G \leq 1 \mid G_{DV})_{s(1|i)}^{t=0}$ ihre unterschiedlichen Relevanzschätzungen zugeordnet werden (siehe Abb. 127)), wobei die erste Relevanzschätzung $rel(w(x_{DV})_{s(1|i)}^{t=0})\hat{}$ auf der Gesamt-Stimulusmenge $M_{DV(m),s(1|i)}^{t=0}$ basiert, und die anderen p = 4 Schätzungen $rel(w(x_{DV})_{s(1|i)}^{t=0})_p\hat{}$ auf den Bootstrap-Stimuluslisten $M_{DV(m),p,s(1|i)}^{t=0}$.

Abb. 127) Gewinner-Neuron und seine Nachbarn mit Bootstrap-Relevanzschätzungen

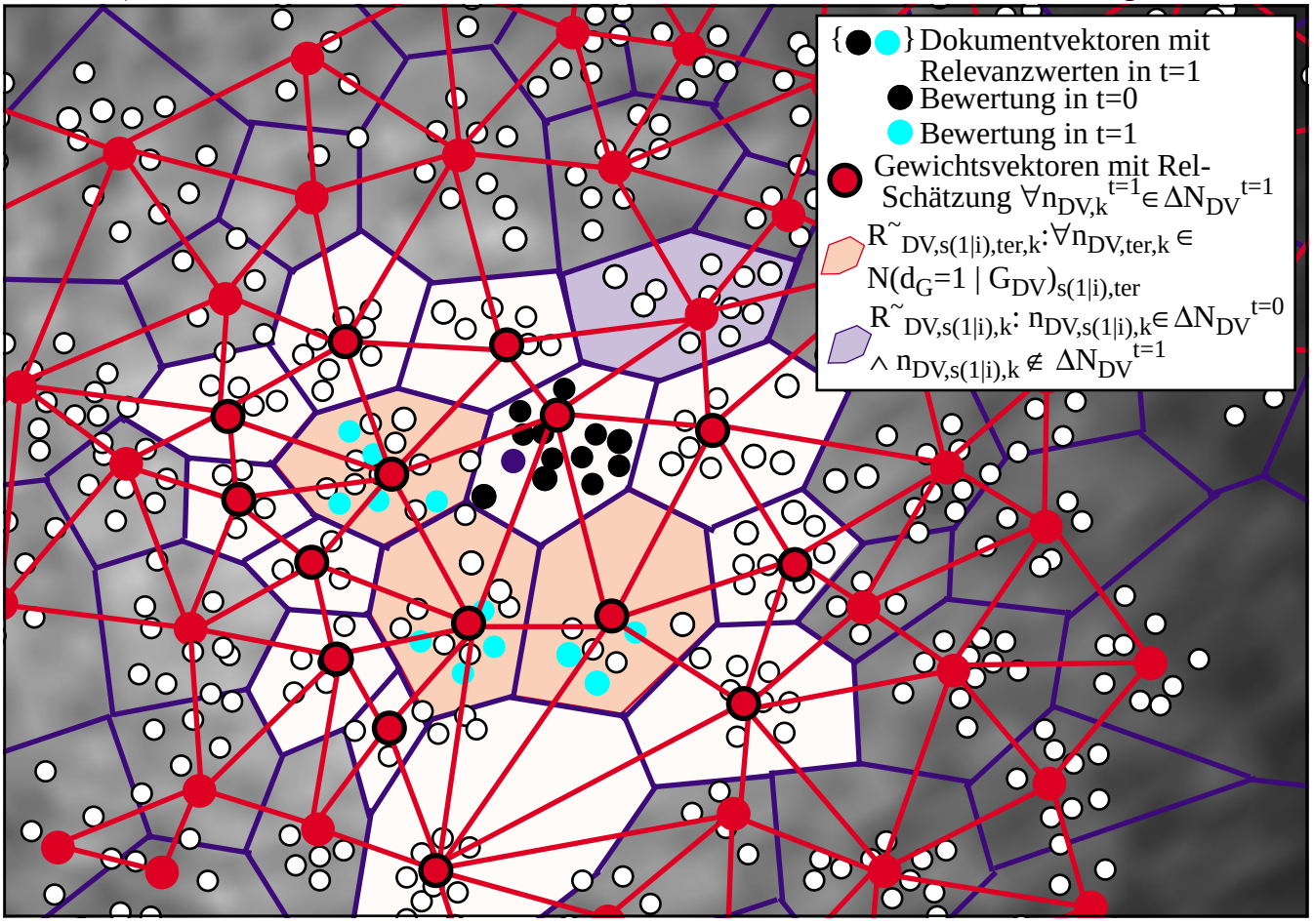


Nachdem die δ+1 Approximationsmodelle durch die Stützpunkte im DVR und im Relevanzraum spezifiziert wurden, wird die Iteration t=1 begonnen, indem die primäre Ergebnismenge $DVM_{prim}^{t=1}$ bestimmt wird, die als Vereinigungsmenge der Dokumentvektorenmengen der Neurone aus der Nachbarschaftsmenge $N(d_G \leq 1 \mid G_{DV})_{s(1|i)}^{t=0}$ festgelegt sein soll, korrigiert um die Dokumentvektoren, für die bereits ein Relevanzwert vorliegt, d.h. ohne die Elemente aus $M_{DV,s(1|i)}^{t=0}$ bzw. allgemeiner die Stimuli $M_{DV}^{t=0}$ aus der Iteration t=0:

$$DVM_{prim}^{t=1} = \{\bigcup_k M_{DV,k}\} \setminus \{x_j^{t=0} \mid \forall m_j^{t=0} \in M_{DV(m)}^{t=0}\},$$
$$\forall n_{DV,k} \in N(d_G \leq 1 \mid G_{DV})_{s(1|i)}^{t=0}. \quad (626)$$

Für ein Element $x_j^{t=1}$ aus $DVM_{prim}^{t=1}$ werden jeweils δ+1 Einzel-Relevanzschätzungen durch die vorliegenden Approximationsmodelle erzeugt, die zu einer Bootstrap-Gesamtschätzung $rel(x_j^{t=1} \mid rel(w(x_{DV})_{s(1|i)}^{t=0})_p\wedge, p = 1, ..., \delta)$ bzw. einer 0,632-Bias-korrigierten Gesamtschätzung $rel(x_j^{t=1} \mid rel(w(x_{DV})_{s(1|i)}^{t=0})\wedge, rel(w(x_{DV})_{s(1|i)}^{t=0})_p\wedge)_{0,632bias}$ aggregiert werden. Mit diesen aggregierten Schätzungen wird eine sekundäre Ergebnisliste $DV_{sec}^{t=1}$ erzeugt, indem die Elemente nach fallender Relevanzschätzung geordnet werden. Aus $DV_{sec}^{t=1}$ werden die ersten Listenelemente in die tertiäre Liste $DV_{ter}^{t=1}$ übernommen, und die korrespondierenden Dokumente dieser Dokumentvektoren werden durch den Agenten bewertet, sodass die Stimulusmenge $M_{DV(m)}^{t=1}$ gebildet wird, die vereinigt mit $M_{DV(m)}^{t=0}$ die Menge $M_{DV(m)}^{t\leq 1}$ ergibt.

Abb. 128) Dokumentvektoren und Gewichtsvektoren mit Relevanzwerten bzw. -schätzungen in t=1



Nach den Relevanzschätzungen werden die Stützpunkte $rel(w(x_{DV})_{s(1|i)}^{t=0})^\wedge$ und $rel(w(x_{DV})_{s(1|i)}^{t=0})_p^\wedge$ nicht mehr benötigt, und $N_{DV}^{t=0}$ wird zu $N_{DV}^{t=1}$ aktualisiert. Hierzu werden alle Neurone aus $N(d_G=1 \mid G_{DV})_{s(1|i)}^{t=0}$ identifiziert, deren lokale Dokumentvektorenmenge mindestens ein Element aus der tertiären Liste $DV_{ter}^{t=1}$ enthält, wobei diese Neurone in der nachfolgenden Nachadaption Gewinner-Adaptionen durchführen. In Abb. 128) wird der Fall dargestellt, dass unter den 6 Elementen aus $N(d_G=1 \mid G_{DV})_{s(1|i)}^{t=0}$ drei Neurone existieren, in deren Dokumentvektorenmengen Elemente aus $DV_{ter}^{t=1}$ liegen, d.h. diese Dokumentvektoren besitzen nun Relevanzwerte. Diese Teilmenge von Neuronen soll mit $N(d_G=1 \mid G_{DV})_{s(1|i),ter}$ und die Elemente mit $n_{DV,ter,k}^{t=1}$ bezeichnet werden. Die unmittelbare Nachbarmenge eines Elementes $n_{DV,ter,k}^{t=1}$ wird mit $N(d_G=1 \mid G_{DV})_{ter,k}^{t=1}$ bezeichnet, wobei die Gewichtsvektoren dieser Neurone Nachbar-Adaptionsoperationen durchführen, falls die Neurone nicht selbst Element von $N(d_G=1 \mid G_{DV})_{s(1|i),ter}$ sind.

Die Adaptionsoperationen sollen sich auf die neue Stimulusmenge $M_{DV(m)}^{t=1}$ beziehen, d.h. es wird bei einer Nachadaptions-Operation ein Stimulus $m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1}))$ gleichverteilt zufällig mit Zurücklegen aus $M_{DV(m)}^{t=1}$ gezogen, dessen Dokumentvektor $x_j^{t=1}$ ein Gewinner-Neuron $n_{DV,s(1|j)}^{t=0}$ und eine Nachbarschaftsmenge $N(d_G=1 \mid G_{DV})_{s(1|j)}^{t=1}$ spezifiziert. Der Gewichtsvektor $w(x_{DV})_{s(1|j)}^{t=1}$ wird einer Gewinner-Adaption mit ε_s unterzogen, und die Gewichtsvektoren $w(x_{DV})_{s(1|j),k}^{t=1}$ der Neurone $n_{DV,s(1|j),k}^{t=1}$ aus der Nachbarschaft werden einer Nachbar-Adaption mit ε_n unterzogen. Während der gesamten Nachadaption werden somit die Gewichtsvektoren der Elemente der Vereinigungsmenge der Nachbarschaften $N(d_G\leq 1 \mid G_{DV})_{s(1|j)}^{t=1}$ aller Gewinner-Neurone bezüglich aller Stimuli aus $M_{DV(m)}^{t=1}$ modifiziert, sodass sich eine Delta-Menge $\Delta N_{DV}^{t=1}$ ergibt:

$$\Delta N_{DV}^{t=1} = \bigcup_j N(d_G\leq 1 \mid G_{DV})_{s(1|j)}^{t=1}, \forall m_j^{t=1} \in M_{DV(m)}^{t=1}. \quad (627)$$

Für diese Neurone wird eine Relevanzschätzung auf der Basis aller verfügbaren Stimuli, d.h. aller Elemente aus $M_{DV(m)}^{t\leq 1}$ gebildet, sodass die aktualisierte GNG-SOM $N_{DV}^{t=1}$ komplett vorliegt, wenn die Neurone aus $\Delta N_{DV}^{t=1}$ die korrespondierenden Neurone aus $N_{DV}^{t=0}$ ersetzen.

Im nächsten Schritt werden aus $N_{DV}^{t=1}$ die Bootstrap-GNG-SOMs $N_{DV,p}^{t=1}$ abgeleitet, indem zunächst aus $M_{DV(m)}^{t\leq 1}$ Bootstrap-Stimuluslisten $M_{DV(m),p}^{t\leq 1}$ generiert werden, die für Relevanzschätzungen herangezogen werden. Wie in jeder Iteration werden die Neurone neu bewertet, die bei der letzten Aktualisierung modifiziert wurden, d.h. in t=1 werden den Elementen $n_{DV,k}^{t=1}$ aus $\Delta N_{DV}^{t=1}$ jeweils δ+1 Relevanzschätzungen auf der Basis der Menge $M_{DV(m)}^{t\leq 1}$ und den δ Listen $M_{DV(m),p}^{t\leq 1}$ zugeordnet.

Zu beachten sind die Neurone, die Element von $\Delta N_{DV}^{t=0}$ waren, jedoch kein Element von $\Delta N_{DV}^{t=1}$ mehr sind (siehe Abb. 128)), d.h. es sind Nachbarn von $n_{DV,s(1|i)}^{t=0}$, in deren Voronoi-Regionen kein Dokumentvektor in die tertiäre Liste $DV_{ter}^{t=0}$ aufgenommen wurde, und somit auch kein Dokumentvektor mit einer Relevanzbewertung durch den Agenten vorliegt. Diesen Neuronen wurde in der vorangegangenen Iteration eine Relevanzschätzung zugeordnet. Da entsprechend den verwendeten Relevanz-Approximationsmodellen die Voronoi-Region nicht als relevant bewertet wurde, soll in der aktuellen Iteration diesem Neuron keine Relevanzschätzung mehr zugeordnet werden. Es hat eine gewisse Verschiebung der Aufmerksamkeit des Verfahrens weg von dieser Voronoi-Region stattgefunden, was auch dadurch gerechtfertigt ist, dass keine Approximationsstützpunkte dem Neuron zugeordnet werden konnten. Weiterhin

werden den Dokumentvektoren in den lokalen Stimulusmengen der Neurone diesen Typs in t=1 keine neuen Relevanzschätzungen mehr zugeordnet, d.h. sie werden für den weiteren Ablauf in t = 1 nicht mehr berücksichtigt. Denkbar ist jedoch, dass in einer nachfolgenden Iteration die Aufmerksamkeit sich wieder einem solchen Neuron zuwendet, wenn in dessen Nachbarschaft ein Neuron liegen sollte, das ein Dokumentvektor besitzt, das für eine tertiäre Ergebnisliste ausgewählt wurde. Dies ergibt sich dadurch, dass dieses Neuron in der darauffolgenden GNG-SOM-Aktualisierung Gewinner-Adaptionen unterliegt, und seine Nachbarn werden Nachbar-Adaptionen unterzogen. Daraus folgt, dass ein Neuron vom betrachteten Typ Element der Deltamenge ΔN_{DV} wird, denen jeweils Relevanzschätzungen zugeordnet werden.

Nachdem $N_{DV}^{t=1}$ und $N_{DV,p}^{t=1}$ erzeugt wurden, wird t=2 begonnen, indem die primäre Ergebnismenge $DVM_{prim}^{t=2}$ gebildet wird, als Vereinigung der Dokumentvektoren der Neurone, deren Gewichtsvektoren in t=1 adaptiert wurden, korrigiert um die Dokumentvektoren, für die eine Relevanzbewertung bereits vorliegt:

$$DVM_{prim}^{t=2} = \{\bigcup_k M_{DV,k}\} \setminus \{x_j^t \mid \forall m_j^t \in M_{DV(m)}^{t\leq 1}\},$$
$$\forall n_{DV,k} \in N(d_G \leq 1 \mid G_{DV})_{ter,s(1|i)}^{t=1}. \quad (628)$$

Sinnvoll wäre jedoch eine allgemeinere Beschreibung von DVM_{prim}^{t} ohne Bezugnahme auf die Initialisierungs-Iteration mit $n_{DV,s(1|i)}$. Wird mit $N_{D,bew}^{t}$ eine GNG-SOM bezeichnet, die in einer unklassifizierten Dokumentmenge auf der Basis der Dokumentvektoren aufgebaut wird, die bis zum Zeitpunkt t durch den Agenten bewertet wurden, d.h. durch $M_{DV(m)}^{\leq t}$, so soll mit $N_{DV,bew}^{t}$ eine Neuronenmenge bzw. GNG-SOM als Teilmenge einer klassifizierten Dokumentvektorenmenge N_{DV} bezeichnet werden, in deren Dokumentvektorenmenge mindestens ein Element enthalten ist, zu dem eine Relevanzbewertung durch den Agenten vorliegt. $N_{DV,bew}^{t}$ wird zu Beginn der Iteration t definiert, sodass die Dokumentvektoren, die in der Iteration t in die tertiäre Dokumentvektorenliste übernommen werden, und die $M_{DV(m)}^{t}$ bilden, nicht berücksichtigt werden.

Es handelt es sich bei $N_{DV,bew}^{t}$ um alle Neurone aus N_{DV}, für die $M_{DV(m),k}^{\leq t-1}$ nicht leer ist:

$$N_{DV,bew}^{t} = \{n_{DV,k} \in N_{DV} \mid M_{DV(m),k}^{\leq t-1} \neq \emptyset\}. \quad (629)$$

Die primäre Ergebnismenge DVM_{prim}^{t} kann durch die Neurone aus $N_{DV,bew}^{t}$ und ihre Nachbarneurone korrigiert um die zu $M_{DV}^{\leq t-1}$ korrespondierenden Dokumentvektoren, definiert werden:

$$DVM_{prim}^{t} = \{\bigcup_k M_{DV,r}\} \setminus \{x_j^t \mid \forall m_j^t \in M_{DV(m)}^{t\leq 1}\},$$
$$\forall n_{DV,r} \in N(d_G \leq 1 \mid G_{DV})_k^t, \forall n_{DV,k} \in N_{DV,bew}^{t}. \quad (630)$$

Die Initialisierung in t=0 erfolgt mit $N_{DV,bew}^{t=0} = \emptyset$ und mit $DVM_{prim}^{t=0} = M_{DV}^{t=0} = M_{DV,s(1|i)}$. $N_{DV,bew}^{t=1}$ wird als $\{n_{DV,s(1|i)}^{t=0}\}$ festgelegt, sodass die primäre Ergebnismenge $DVM_{prim}^{t=1}$ mit Hilfe der Neurone aus $N(d_G \leq 1 \mid G_{DV})_{s(1|i)}^{t=1}$ gebildet wird. Die richige Unterscheidungsfähigkeit greift erst ab t=2, da nicht alle Neurone aus $N(d_G \leq 1 \mid G_{DV})_{s(1|i)}^{t=1}$ in die Menge $N_{DV,bew}^{t=2}$ übernommen werden müssen. Es werden nur die Neurone übernommen, die mindestens einen Dokumentvektor enthalten, der in $DV_{ter}^{t=1}$ aufgenommen wurde, und für den somit eine Relevanzbewertung vorliegt.

4.4.2.2.2) Adaption der Stützpunkte im DVR und im Relevanzraum

Die Grundlage dieser Vorgehensweise besteht darin, mit der Bootstrap-Stimulusliste $M_{DV(m),p,s(1|i)}^{t=0}$ für $N_{DV,p}^{t=0}$ eigene Gewichtsvektoren und daraus abgeleitet eigene Relevanzschätzungen zu generieren. Für die Modifikation des Gewichtsvektors kommt die Bildung des Mittelwertsvektors, sowie alle SC-GNG-SOM-Nachadaptionen in Frage, welche die gegebene Verbindungsstruktur inklusive der Anzahl der Neurone konstant halten. Wie im vorangegangenen Abschnitt sollen Adaptionsoperationen mit einer Gewinner- und einer Nachbar-Adaption durchgeführt werden, d.h. nach der Ziehung eines Stimulus $m_j^t = (x_j^t, rel(x_j^t))$ wird das Gewinner-Neuron $n_{DV,s(1|j)}^t$ ermittelt, dessen Gewichtsvektor eine Gewinner-Adaption mit ε_s durchführt, und dessen Nachbarn aus $N(d_G=1 \mid G_{DV})_{s(1|j)}^t$ eine Nachbar-Adaption mit ε_n.

Betrachtet wird die Ursprungs-GNG-SOM $N_{DV}^{t=0}$, die aus N_{DV} erzeugt wurde, indem $n_{DV,s(1|i)}^{t=0}$ und seine Nachbarn nachadaptiert wurden, d.h. indem Elemente aus $M_{DV(m)}^{t=0}$ zufällig mit Zurücklegen gezogen wurden, gefolgt von einer Gewinner- und Nachbar-Adaption. Nachdem eine bestimmte Anzahl dieser lokalen Präsentationen durchgeführt wurde, wird für $n_{DV,s(1|i)}^{t=0}$ und seine Nachbarn eine Relevanzschätzung mit einer instanzbasierten LWR-Approximation auf der Basis der Stimuli aus $M_{DV}^{t=0}$ durchgeführt.

Die Monorepräsentation $N_{DV}^{t=0}$ wird zu einer Polyrepräsentation erweitert, indem δ Bootstrap-Stimuluslisten $M_{DV,p}^{t=0} := M_{DV,p,s(1|i)}^{t=0}$ generiert werden, gefolgt von der Nachadaption zu $N_{DV,p}^{t=0}$. In der Iteration t=1 sind die folgenden Ableitungen möglich:
1) Ableitung von $N_{DV,p}^{t=0}$ aus $N_{DV}^{t=0}$ mit $M_{DV(m),p}^{t=0}$.
2) Ableitung von $N_{DV,p}^{t=0}$ aus N_{DV} mit $M_{DV(m),p}^{t=0}$.

Wird beachtet, dass $N_{DV}^{t=0}$ aus N_{DV} durch Nachadaptionen erzeugt wurde, so kann eine Rangfolge der Strukturen nach fallendem Grad der Nachadaption eingeführt werden, mit N_{DV} als Nullpunkt, d.h. $(N_{DV}^{t=0}, N_{DV})$ für t=0, $(N_{DV}^{t=1}, N_{DV}^{t=0}, N_{DV})$ für t=1, usw. Sinnvoll wäre in diesem Zusammenhang die Regel, dass in einer Iteration nur GNG-SOMs des gleichen Adaptionsgrades für Approximationsmodelle verwendet werden sollen, sodass die $N_{DV,p}^{t=0}$ aus N_{DV} abgeleitet werden, wie $N_{DV}^{t=0}$ aus N_{DV} abgeleitet wurde. Die Nachadaption in t=0 zu Erzeugung eines $N_{DV,p}^{t=0}$ erfolgt wie üblich, indem Elemente aus $M_{DV(m),p}^{t=0}$ zufällig mit Zurücklegen gezogen wurden, gefolgt von einer Gewinner- und Nachbar-Adaptionen, sowie der späteren Bildung von Relevanzschätzungen für $n_{DV,p,s(1|i)}^{t=0}$ und seine Nachbarn auf der Basis von $M_{DV,p}^{t=0} := M_{DV,p,s(1|i)}^{t=0}$. Da die Bootstrap-GNG-SOMs $N_{DV,p}^{t=0}$ aus N_{DV} gebildet werden und nicht aus $N_{DV}^{t=0}$, bedeutet dies, dass die Bildung von $N_{DV}^{t=0}$ und den δ Modellen $N_{DV,p}^{t=0}$ parallel und quasi unabhängig durchgeführt werden kann.

Die erzeugten Bootstrap-GNG-SOMs werden in die Menge $BMN_{DV}^{t=0}$ eingefügt, bis diese die Struktur besitzt:

$$BMN_{DV}^{t=0} = \{N_{DV}^{t=0}, N_{DV,p}^{t=0} \mid p = 1, ..., \delta\}. \quad (631)$$

Nachdem die Menge $BMN_{DV}^{t=0}$ vollständig erzeugt wurde, wird die Iteration t=1 begonnen, indem eine primäre Ergebnismenge $DVM_{prim}^{t=1}$ ermittelt wird. Hierzu wird zunächst die Neuronenmenge $N_{DV,bew}^{t=1}$ ermittelt, als die Menge von Neuronen aus N_{DV}, die Dokumentvektoren in ihrer Voronoi-

Region besitzt, denen ein Relevanzwert durch den Agenten zugeordnet wurde. Für $N_{DV,bew}^{t=1}$ wird die Menge $\{n_{DV,s(1|i)}^{t=0}\}$ verwendet. Die primäre Ergebnismenge $DVM_{prim}^{t=1}$ kann durch die Neurone aus $N_{DV,bew}^{t=1}$ und ihre Nachbarneurone korrigiert um die Dokumentvektoren, die zu $M_{DV(m)}^{t\leq 0} = M_{DV(m)}^{t=0}$ korrespondieren, definiert werden:

$$DVM_{prim}^{t=1} = \{\bigcup_k M_{DV,r}\} \setminus \{x_j^t \mid \forall m_j^t \in M_{DV(m)}^{t=0}\},$$
$$\forall n_{DV,r} \in N(d_G \leq 1 \mid G_{DV})_k, \forall n_{DV,k} \in N_{DV,bew}^{t=1}. \quad (632)$$

Da alle Dokumentvektoren des Neurons $n_{DV,s(1|i)}$ eine Relevanzbewertung besitzen, bedeutet dies, dass alle Dokumentvektorenmengen seiner Nachbarn $N(d_G=1 \mid G_{DV})_{s(1|i)}$ vereingt werden, um $DVM_{prim}^{t=1}$ zu bilden:

$$DVM_{prim}^{t=1} = \bigcup_k M_{DV,s(1|i),k}^{t=0}, \forall n_{DV,s(1|i),k}^{t=0} \in N(d_G=1 \mid G_{DV})_{s(1|i)}^{t=0}. \quad (633)$$

Jedem Dokumentvektor $x_j^{t=1} \in DVM_{prim}^{t=1}$ wird eine Menge von Relevanzschätzungen mit Hilfe der Approximationsmodelle zugeordnet, die zu den Elementen aus $BMN_{DV}^{t=0}$ korrespondieren, d.h. mit $AM(rel(x) \mid N_{DV}^{t=0})$ und $AM(rel(x) \mid N_{DV,p}^{t=0})$, p = 1, ..., δ, gefolgt von der Aggregation dieser Einzelschätzungen zu einer Bootstrap-Gesamtschätzung $rel(x_j^{t=1} \mid N_{DV,p}^{t=0}$, p = 1, ..., δ) bzw. einer 0,632-Bias-korrigierten Gesamtschätzung $rel(x_j^{t=1} \mid BMN_{DV}^{t=0})_{0,632bias}$. Mit diesen aggregierten Schätzungen wird eine sekundäre Ergebnisliste $DV_{sec}^{t=1}$ erzeugt, indem die Elemente aus $DVM_{prim}^{t=1}$ nach fallender Relevanzschätzung geordnet werden. Aus $DV_{sec}^{t=1}$ werden die ersten Listenelemente ausgewählt, welche die tertiäre Liste $DV_{ter}^{t=1}$ bilden. Nach der Bewertung ergibt sich $M_{DV(m)}^{t=1}$, die vereinigt mit $M_{DV(m)}^{t=0}$ die Menge $M_{DV(m)}^{t\leq 1}$ ergibt.

Nach den Relevanzschätzungen haben die Modelle $N_{DV,p}^{t=0}$, p = 1, ..., δ ihre Funktion erfüllt, und da der Fall der iterations-spezifischen Ableitung von Bootstrap-GNG-SOMs betrachtet wird, werden diese Modelle gelöscht.

Mit Hilfe von $M_{DV(m)}^{t=1}$ wird das Modell $N_{DV}^{t=0}$ zu $N_{DV}^{t=1}$ aktualisiert, indem Elemente aus $M_{DV(m)}^{t=1}$ zufällig mit Zurücklegen gezogen wurden, gefolgt von einer Gewinner- und Nachbar-Adaption. Nachdem eine bestimmte Anzahl dieser lokalen Präsentationen durchgeführt wurde, werden für alle Neurone, deren Gewichtsvektor während der Nachadaption verändert wurde, d.h. alle Neurone aus der Delta-Menge $\Delta N_{DV}^{t=1}$ mit

$$\Delta N_{DV}^{t=1} = \bigcup_j N(d_G \leq 1 \mid G_{DV})_{s(1|j)}, \forall m_j^{t=1} \in M_{DV(m)}^{t=1}. \quad (634)$$

eine Relevanzschätzung mit Hilfe eines instanzbasierten LWR-Approximationsverfahrens und aller Stimuli aus $M_{DV(m)}^{t\leq 1}$ erzeugt.

Im weiteren werden die Bootstrap-GNG-SOMs $N_{DV,p}^{t=1}$ durch Nachadaptionen erzeugt, wobei in t=1 die folgenden Ableitungen möglich sind:
1) Ableitung von $N_{DV,p}^{t=1}$ aus $N_{DV}^{t=1}$ mit $M_{DV(m),p}^{t=1}$.
2) Ableitung von $N_{DV,p}^{t=1}$ aus $N_{DV}^{t=0}$ mit $M_{DV(m),p}^{t=1}$.
3) Ableitung von $N_{DV,p}^{t=1}$ aus N_{DV} mit $M_{DV(m),p}^{t=1}$.

Es wird wieder die Regel angewendet, dass in einer Iteration nur GNG-SOMs des gleichen Adaptionsgrades für Approximationsmodelle verwendet werden sollen, sodass die $N_{DV,p}^{t=1}$ aus $N_{DV}^{t=0}$ parallel abgeleitet werden, so wie $N_{DV}^{t=1}$ aus $N_{DV}^{t=0}$ abgeleitet wurde. Im Rahmen der Ableitung wird zunächst $N_{DV}^{t=1}$ δ mal kopiert und die Kopien werden zu $N_{DV,p}^{t=1}$, $p = 1, ..., \delta$, umbenannt, gefolgt von der Bildung Bootstrap-Stimuluslisten $M_{DV(m),p}^{t\leq1}$ durch Ziehen mit Zurücklegen aus $M_{DV(m)}^{t\leq1}$. Die Neurone $n_{DV,p,k}^{t=1}$ aus $N_{DV,p}^{t=1}$, in deren lokalen Stimulusmengen Elemente aus $M_{DV(m)}^{t=1}$ liegen, d.h. Neurone aus $\Delta N_{DV,p}^{t=1} := \Delta N_{DV}^{t=1}$, erhalten entsprechend $M_{DV(m),p}^{t\leq1}$ Stimuluslisten $M_{DV(m),p,k}^{t\leq1}$, in denen einzelne Elemente nicht mehr vorkommen, und andere mehrmals enthalten sein können. Es folgt eine individuelle Nachadaption für alle Bootstrap-GNG-SOMs, indem bei einer Nachadaptions-Operation ein Stimulus $m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1}))$ aus $M_{DV(m),p}^{t\leq1}$ gleichverteilt zufällig mit Zurücklegen gezogen wird, und der Dokumentvektor lokal seinem Gewinner-Neuron und seinen Nachbarn präsentiert wird. Der Gewichtsvektor des Gewinner-Neurons führt eine Gewinner-Adaption und die Gewichtsvektoren der Nachbar-Neurone jeweils eine Nachbar-Adaption mit $x_j^{t=1}$ als Fixpunkt durch. Nachdem eine bestimmte Anzahl dieser lokalen Präsentationen durchgeführt wurde, werden die Gewichtsvektoren, die adaptiert wurden, beibehalten und auf ihrer Basis werden Aktualisierungen der Relevanzschätzungen gebildet, wobei alle Elemente aus der Liste $M_{DV(m),p}^{t\leq1}$ verwendet werden, d.h. mehrfaches Vorkommen wird in der LWR-Approximation auch mehrfach berücksichtigt.

Modifiziert werden bei einer Nachadaption auf der Basis der Bootstrap-Stimulusliste $M_{DV(m),p}^{t\leq1}$ die Gewinner-Neurone $n_{DV,p,s(1|j)}^{t=1}$ aller Stimuli $m_j^{t=1}$ aus $M_{DV(m),p}^{t\leq1}$, sowie alle Nachbar-Neurone dieser Gewinner-Neurone, sodass sich eine Deltamenge $\Delta N_{DV,p}^{t=1}$ ergibt:

$$\Delta N_{DV,p}^{t=1} = \bigcup_j N(d_G\leq1 \mid G_{DV,p})_{s(1|j)}^{t=1}, \forall m_j^{t=1} \in M_{DV(m),p}^{t=1}. \quad (635)$$

Alle Neurone aus dieser Deltamenge ersetzen korrespondierende Neurone aus der GNG-SOM $N_{DV}^{t=1}$, wodurch die Bootstrap-GNG-SOM $N_{DV,p}^{t=1}$ erzeugt wird:

$$N_{DV,p}^{t=1} = N_{DV}^{t=1} \oplus \Delta N_{DV}^{t=1}. \quad (636)$$

Nach diesen Operationen liegen neben der Ursprungs-GNG-SOM die δ Bootstrap-GNG-SOMs vor, die in der Menge $BMN_{DV}^{t=1} = \{N_{DV}^{t=1}, N_{DV,p}^{t=1} \mid p = 1, ..., \delta\}$ zusammengefasst werden. Somit wird die Iteration t=1 beendet, und die nächste Iteration wird mit der Ermittlung der primären Ergebnismenge $DVM_{prim}^{t=2}$ begonnen. Hierzu wird zunächst die Menge aller Neurone $n_{DV,k}$ aus N_{DV} gebildet, für welche die Stimulusmenge $M_{DV,k}^{t\leq1}$ nicht leer ist, d.h. denen Dokumentvektoren zugeordnet sind, zu denen eine Bewertung vorliegt:

$$N_{DV,bew}^{t=2} = \{n_{DV,k} \in N_{DV} \mid M_{DV(m),k}^{t\leq1} \neq \emptyset\}. \quad (637)$$

Die primäre Ergebnismenge $DVM_{prim}^{t=2}$ wird durch die Neurone aus $N_{DV,bew}^{t=2}$ und ihre Nachbarneurone korrigiert um $M_{DV}^{t\leq1}$ festgelegt als:

$$DVM_{prim}^{t=2} = \{\bigcup_k M_{DV,r}\} \setminus \{x_j^t \mid \forall m_j^t \in M_{DV(m)}^{t\leq1}\},$$
$$\forall n_{DV,r} \in N(d_G\leq1 \mid G_{DV})_k, \forall n_{DV,k} \in N_{DV,bew}^{t=2}. \quad (638)$$

Für alle Dokumentvektoren aus $DVM_{prim}^{t=2}$ werden jeweils δ+1 Relevanzschätzungen mit Hilfe der GNG-SOMs aus $BMN_{DV}^{t=1}$ gebildet, die zu einem der Bootstrap-Gesamtschätzungen aggregiert werden. Mit diesen Schätzungen wird die geordnete Liste $DV_{sec}^{t=2}$ nach fallenden Werten aufgebaut, aus der die ersten Elemente in die tertiäre Liste $DV_{ter}^{t=2}$ übernommen werden, deren korrespondierende Dokumente dem Agenten präsentiert werden, der diese bewertet. Wird vom Agenten kein Abbruch durchgeführt, so ergeben sich die Stimulusmengen $M_{DV(m)}^{t=2}$ und $M_{DV(m)}^{t\leq 2}$.

4.4.2.2.3) Adaption mit Wachstum der Stützpunkte im DVR und Relevanzraum

In diesem Abschnitt soll der Spezialfall der Adaption der Stützpunkte im DVR und im Relevanzraum dargestellt werden, bei dem die Nachadaption mit Wachstumsoperationen durchgeführt wird (siehe auch Abschnitt 4.3.3.2)).

Ausgangspunkt ist N_{DV}, das durch den Queryvektor $q_i^{t=0}$ das Gewinner-Neuron $n_{DV,s(1|i)}^{t=0}$ spezifiziert, wodurch nach der Bewertung des Agenten die Stimulusmenge $M_{DV(m)}^{t=0}$ hervorgeht. Mit Hilfe von $M_{DV(m)}^{t=0}$ wird aus der nicht individuellen GNG-SOM N_{DV} die individuelle GNG-SOM $N_{DV}^{t=0}$ durch Nachadaptionen abgeleitet, wobei wie im Abschnitt 4.3.3.2) in t=0 auf Wachstum aufgrund der Tatsache verzichtet werden soll, da alle bewerteten Stimuli genau einem Neuron zugeordnet sind. Im Kontext der Polyrepräsentation wird parallel zu der Ableitung von $N_{DV}^{t=0}$ aus N_{DV} δ mal ein Bootstrap-Verfahren angewendet, indem aus $M_{DV(m)}^{t=0}$ durch Ziehen mit Zurücklegen Bootstraplisten $M_{DV(m),p}^{t=0}$ erzeugt werden, mit denen $N_{DV,p}^{t=0}$ aus N_{DV} abgeleitet wird. Die Ableitungen erfolgen, indem Stimuli aus $M_{DV(m)}^{t=0}$ bzw. $M_{DV(m),p}^{t=0}$ gleichverteilt zufällig gezogen werden, gefolgt von einer Gewinner-Adaption und Nachbar-Adaptionen, und der Berechnung einer Relevanzschätzung mit Hilfe aller jeweils zur Verfügung stehender Stimuli für das Gewinner-Neuron. Da keine unabhängigen Wachstumsprozesse innerhalb dieser ersten Nachadaptionsphase erlaubt sind, besitzen alle Modelle $N_{DV}^{t=0}$ bzw. $N_{DV,p}^{t=0}$ die gleiche Anzahl von Neuronen sowie die gleiche Verbindungsstruktur.

Durch die Festlegung der Modelle $N_{DV}^{t=0}$ bzw. $N_{DV,p}^{t=0}$, die in der Menge $BMN_{DV}^{t=0} = \{N_{DV}^{t=0}, N_{DV,p}^{t=0} \mid p = 1, ..., \delta\}$ zusammengefasst werden, existieren die korrespondierenden, prototypbasierten Relevanz-Approximationsmodelle $AM(rel(x) \mid N_{DV}^{t=0})$ und $AM(rel(x) \mid N_{DV,p}^{t=0})$, sodass die Iteration t=0 beendet, und t=1 begonnen werden kann. Zunächst wird die primäre Ergebnismenge $DVM_{prim}^{t=1}$ mit Hilfe der Neurone ermittelt, denen mindestens ein Stimulus m = (x, rel(x)) zugeordnet ist, d.h. mit Hilfe von $N_{DV,bew}^{t=1} = \{n_{DV,s(1|i)}^{t=0}\}$. Die Ergebnismenge $DVM_{prim}^{t=1}$ ergibt sich, indem die lokalen Dokumentvektorenmengen der Elemente von $N_{DV,bew}^{t=1}$ und ihrer Nachbarn vereinigt und um $M_{DV(m)}^{t=0}$ korrigiert werden.

Die Elemente $x_j^{t=1}$ aus $DVM_{prim}^{t=1}$ werden durch die δ+1 Approximationsmodelle $AM(rel(x) \mid N_{DV}^{t=0})$, $AM(rel(x) \mid N_{DV,p}^{t=0})$, p = 1, ..., δ, bewertet, gefolgt von der jeweiligen Aggregation der Einzelbewertungen zu einer Bootstrap- bzw. bias-korrigierten Bootstrap-Gesamtschätzung. Mit dieser werden die Dokumentvektoren nach fallender Relevanzschätzung sortiert, sodass sich die sekundäre Ergebnisliste $DV_{sec}^{t=1}$ ergibt, aus der die ersten $\#M_{DV(m)}^{t=0}$ Elemente in die tertiäre Liste $DV_{ter}^{t=1}$ übernommen wer-

den. Die dazu korrespondierenden Dokumente werden durch den Agenten bewertet, sodass sich die Stimulusmengen $M_{DV(m)}^{t=1}$ und $M_{DV(m)}^{t\leq 1}$ ergeben.

Mit Hilfe von $M_{DV(m)}^{t\leq 1}$ wird das Gesamtmodell $N_{DV}^{t=0}$ zu $N_{DV}^{t=1}$ aktualisiert, indem nun Nachadaptionen mit Wachstumsoperationen verwendet werden, wobei ein zusätzlicher Index τ eingeführt wird. Bei $\tau = \lambda(1)$ bzw. einem Vielfachen von $\lambda(1)$ wird eine Wachstumsphase mit der Erzeugung genau eines neuen Neurons eingeschoben. Es werden Elemente $m_j^{t=1} = (x_j^{t=1}, rel(x_j^{t=1}))$ aus $M_{DV(m)}^{t\leq 1}$mit Zurücklegen gezogen, die jeweils ein Gewinner-Neuron $n_{DV,s(1|j)}^{t=1,\tau=1}$ spezifizieren, dessen Gewichtsvektor $w(x_{DV})_{s(1|j)}^{t=1,\tau=1}$ einer Gewinner-Adaption mit dem Adaptionsparameter ε_s unterzogen wird. Die Gewichtsvektoren der Nachbar-Neurone aus $N(d_G=1 \mid G_{DV}^{t=1,\tau=1})_{s(1|j)}$ werden nachfolgend einer Nachbarschafts-Adaption mit ε_n unterzogen, wobei $x_j^{t=1}$ jeweils Fixpunkt der positiven Adaption ist. Nach $\lambda(1)$ dieser Adaptionsoperationen wird der Zwischenzustand $N_{DV}^{t=1,\tau=\lambda(1)}$ für eine Wachstums-Operation verwendet, indem zunächst das primäre Einfügezentrum bestimmt wird, als das Neuron, mit einem maximalen, lokalen Fehlerwert. Als Grundmenge sollen nur die Neurone ausgewählt werden, die mindestens einen Dokumentvektor mit einem Relevanzwert besitzen, d.h. es wird allgemein $N_{DV,bew}^{t,\tau}$ und im speziellen $N_{DV,bew}^{t=1,\tau=\lambda(1)}$ betrachtet, das auf der Basis von $M_{DV(m)}^{t\leq 1}$ erstellt wird. Für $\tau = \lambda(1)$ wird aus $N_{DV,bew}^{t=1,\tau=\lambda(1)}$ das Neuron $n_{DV,f(1)}^{t=1,\tau=\lambda(1)}$ mit dem größten lokalen Fehlerwert ausgewählt, und als primäres Einfügezentrum verwendet. Als Fehlermaß wird der Quantifizierungsfehler der Dokumentvektoren verwendet, denen ein Relevanzwert zugeordnet wurde.

Dies bedeutet jedoch, dass nur den Neuronen aus $N_{DV,bew}^{t=1,\tau=\lambda(1)}$ ein solches Fehlermaß zugeordnet werden kann, nicht jedoch ihren Nachbarn, was sich auf die Auswahl des sekundären Einfügezentrums $n_{DV,f(2)}^{t=1,\tau=\lambda(1)}$ auswirkt. Das sekundäre Einfügezentrum soll somit durch ein anderes Selektionskriterium ausgewählt werden, wie z.B. den Quantifizierungsfehler bezogen auf alle vorliegenden Dokumentvektoren, die einem Neuron zugeordnet sind, oder einfach durch eine Zufallsauswahl aus der Menge der verbundenen Nachbarn des primären Zentrums.

Nachdem das primäre und sekundäre Einfügezentrum festgelegt wurde, wird ein neues Neuron initialisiert, indem in der Mitte der Verbindungskante zwischen den beiden Gewichtsvektoren der Einfügezentren der Gewichtsvektor des neuen Neurons erzeugt wird. Es folgt die Korrektur der Verbindungsstruktur zwischen dem neuen Neuron, den Einfügezentren und ihren verbundenen Nachbarn, sowie die lokale Umverteilung von Dokumentvektoren und Stimuli. Dabei werden alle Dokumentvektoren der beteiligten Neurone umverteilt, unabhängig ob ihnen ein Relevanzwert zugeordnet wurde oder nicht. Beteiligt sind die beiden Einfügezentren, ihre verbundenen Nachbarn, inklusive des neuen Neurons. Mit dieser Umverteilung ist die Wachstumsphase abgeschlossen und es werden in der Folge weitere $\lambda(1)$ Adaptionsoperationen durchgeführt, wenn kein Abbruchkriterium greift. Das Abbruchkriterium kann durch eine maximale Anzahl $\tau(end)$ von Adaptionen festgelegt werden, die als Funktion der Anzahl der Elemente in $M_{DV}^{t\leq 1}$ bestimmt wird.

Es folgt eine instanzbasierte Relevanzschätzung für alle Neurone, denen mindestens ein Dokumentvektor mit Relevanzwert zugeordnet ist, wobei alle verfügbaren Stimuli aus $M_{DV(m)}^{t\leq 1}$ beteilgt werden, d.h. die Schätzungen werden mit $AM(rel(x) \mid M_{DV(m)}^{t\leq 1})$ durchgeführt. Bei den Neuronen handelt es sich nicht mehr um $N_{DV,bew}^{t=1,\tau=\lambda(1)}$, da diese Bezeichnung für das GNG-SOM vor der Wachstumsoperation ver-

wendet wurde, sodass mit $N_{DV,bew}^{t=1,\tau=\lambda(1)}$ eine Zwischenstruktur eingeführt werden soll, die nach der nächsten Adaption zu $N_{DV,bew}^{t=1,\tau=\lambda(1)+1}$ wird.

Nachdem das Abbruchkriterium angewendet wurde, ist die GNG-SOM $N_{DV}^{t=1}$ und somit das prototypbasierte Approximationsmodell AM(rel(x) | $N_{DV}^{t=1}$) komplett, d.h. damit sind die Prozesse bei einer Approximationsmodell-Monorepräsentation abgeschlossen. Bei einer Approximationsmodell-Polyrepräsentation müssen die Bootstrap-GNG-SOMs $N_{DV,p}^{t=1}$ und somit die Approximationsmodelle AM(rel(x) | $N_{DV,p}^{t=1}$) generiert werden, wobei die prinzipiellen Möglichkeiten bestehen:
1) Beibehaltung der Bootstrap-GNG-SOMs, d.h. Ableitung von $N_{DV,p}^{t=1}$ aus $N_{DV,p}^{t=0}$.
2) Iterations-spezifische Neuableitung, d.h. Ableitung von $N_{DV,p}^{t=1}$ aus $N_{DV}^{t=1}$.

Im ersten Fall werden die Bootstrap-Modelle, die in t=0 erzeugt wurden, weiter verwendet und aktualisiert, indem zunächst iterations-spezifische Bootstraplisten $M_{DV(m),p}^{t=1}$ aus $M_{DV(m)}^{t=1}$ durch Ziehen mit Zurücklegen erzeugt werden, wobei $M_{DV(m),p}^{t=1}$ vereinigt mit $M_{DV(m),p}^{t=0}$ die Liste $M_{DV(m),p}^{t\leq1}$ ergibt. Mit Hilfe einer solchen Bootstrapliste wird $N_{DV,p}^{t=1}$ aus $N_{DV,p}^{t=0}$ durch Nachadaptionen erzeugt, wobei in diesem Abschnitt Nachadaptionen mit Wachstumsoperationen betrachtet werden. Diese Nachadaptionen sind zumindest durch ein Abbruchkriterium an die Bildung von $N_{DV}^{t=1}$ gekoppelt, da insgesamt τ(end) Adaptionen durchgeführt werden, sodass bei gleichem $\lambda(1)$ die gleiche Anzahl von Neuronen eingefügt werden, d.h. alle $N_{DV,p}^{t=1}$ und $N_{DV}^{t=1}$ besitzen die gleiche Anzahl von Neuronen. Unterschiedlich sind jedoch die Quantifizierungsfehler der einzelnen Neurone, da die Stimulusmenge $M_{DV(m)}^{t\leq1}$ sich von den Stimuluslisten $M_{DV(m),p}^{t\leq1}$ unterscheidet. Daraus folgt, dass an unterschiedlichen Stellen neue Gewichtsvektoren eingefügt werden, sodass sich nicht korrespondierende Neurone sowie nicht übereinstimmende Verbindungsstrukturen in $N_{DV,p}^{t=1}$ ergeben.

Völlig entkoppelt wären die Nachadaptionen der $N_{DV,p}^{t=1}$, wenn ein performance-abhängiges Abbruchkriterium verwendet wird, da somit eine unterschiedliche Anzahl von Neuronen erzeugt werden kann, die ihre Gewichtsvektoren an abweichenden Stellen in den jeweiligen Verbindungsstrukturen in der lokalen Umgebung des Gewinner-Gewichtsvektors $w(x_{DV})_{s(1|i)}^{t=0}$ besitzen. Auf die Darstellung dieser Vorgehensweise soll jedoch verzichtet werden.

Im zweiten Fall wird zunächst $N_{DV}^{t=1}$ aus $N_{DV}^{t=0}$ durch Nachadaptionen mit Wachstumsoperationen abgeleitet, gefolgt von der Ableitung der Bootstrapmodelle $N_{DV,p}^{t=1}$ aus $N_{DV}^{t=1}$ mit Hilfe von Bootstraplisten $M_{DV(m),p}^{t\leq1}$, die aus $M_{DV(m)}^{t\leq1}$ durch Ziehen mit Zurücklegen generiert wurden. D.h. die Bootstrapmodelle $N_{DV,p}^{t=0}$, die in der vorangegangenen Iteration erzeugt wurden, werden nach der Bewertung der Dokumentvektoren in $DVM_{prim}^{t=0}$ gelöscht, da sie nicht weiter verwendet oder aktualisiert werden. Die prinzipielle Frage ist dabei, wie die Ableitung der Bootstrapmodelle $N_{DV,p}^{t=1}$ aus $N_{DV}^{t=1}$ gestaltet wird, d.h. ob die Stützpunkte im DVR und im Relevanzraum verändert werden, was durch eine Nachadaption mit oder ohne Wachstum durchgeführt werden kann, oder ob die Stützpunkte im DVR übernommen werden und nur unterschiedliche Stützpunkte im Relevanzraum mit $rel(w(x_{DV})_k^{t=0})_p\hat{}$, p = 1, ..., δ, generiert werden. Der Aufwand dieser Alternativen ist fallend, d.h. die Nachadaption mit Wachstum ist aufwendiger als ohne Wachstum, die wiederum erheblich aufwendiger ist als die ausschließliche Generierung von Bootstrapstützpunkten im Relevanzraum.

Unabhängig auf welche der angedeuteten Arten die Elemente aus $BMN_{DV}^{t=1} = \{N_{DV}^{t=1}, N_{DV,p}^{t=1} \mid p = 1, ..., \delta\}$ erzeugt werden, es folgt das Ende der Iteration t=1 und der Beginn von t=2, indem $DVM_{prim}^{t=2}$ mit Hilfe der Neurone und ihrer Nachbarn ermittelt wird, denen momentan mindestens ein Dokumentvektor mit einer Relevanzbewertung zugeordnet ist, d.h. mit den Elementen aus $N_{DV,bew}^{t=2}$ und ihren Nachbarn. Es soll genau eine primäre Ergebnismenge auf der Basis von $N_{DV}^{t=1}$, $M_{DV(m)}^{t\leq1}$ und $N_{DV,bew}^{t=2}$ gebildet werden, ohne die Berücksichtigung der Bootstrapmodelle und -listen $N_{DV,p}^{t=1}$ und $M_{DV(m),p}^{t\leq1}$.

4.5) Nutzung von Ergebnissen vergangener Interaktionen

Die Relevanzurteile von Agenten bezüglich einer Query sowie daraus abgeleitete Repräsentationen wie Relevanz-Approximationsmodelle bzw. adaptierte Repräsentationen wie z.B. ein optimierter Queryvektor sind wertvolle Daten, da zum einen ein Agent nicht beliebig für Relevanzurteile zur Verfügung steht und zum anderen, da die Erzeugung der Repräsentationen teilweise große Rechenressourcen erfordert. Aus diesem Grunde ist es sinnvoll, die Ergebnisse der Interaktion eines Agenten mit dem IRS in einer geeigneten Form zu speichern, sodass diese Daten für aktuelle Interaktionen mit anderen Agenten oder dem gleichen Agenten nutzbringend angewendet werden könnten. Die Nutzung von Interaktionen anderer Agenten wird im Kontext des Information Filterings als soziales bzw. kollaboratives Filtering bezeichnet (Belkin & Croft (1992[36]), Goldberg et al. (1992[147])).

Grundlage der weiteren Betrachtungen soll das Datenobjekt Interaktion(.) sein, in dem mehrere Attribute als Ergebnis der Agent-IRS-Interaktion gesammelt werden. Es soll von einer sequentiellen Vorgehensweise ausgegangen werden, d.h. während der Existenzzeit des IRS werden Interaktionen durchgeführt, die durch ihren Beginn und ihr Ende bezeichnet werden können, wobei der Beginn einer Interaktion eine geordnete, zeitlich sequentielle Liste festlegt, da zu einem diskreten Zeitpunkt genau eine Interaktion begonnen werden soll. Diese sequentielle Ordnung der Interaktionen ist unabhängig davon, dass sich Interaktionen überlappen können, d.h. dass eine neue Interaktion begonnen werden kann, während eine oder mehrere andere Interaktionen noch laufen. Ebenso unabhängig ist die Sequenz davon, dass einzelne Interaktionen unterschiedliche Längen besitzen können, die in absoluten Zeiteinheiten oder in der Anzahl der Interaktionen im Rahmen eines Relevanz-Feedbacks beschrieben werden können.

Diese Unabhängigkeiten ermöglichen es anstatt absoluter Zeiteinheiten den einzelnen Interaktionen entsprechend ihrem Anfang natürliche Zahlen zuzuordnen, wobei angenommen werden soll, dass bislang T-1 Interaktionen durchgeführt wurden, die in der geordneten Interaktionsliste $I^{\leq T-1}$ zusammengefasst werden:

$$I^{\leq T-1} = (\text{Interaktion}(\tau) \mid \tau = 1, ..., T-1). \quad (639)$$

Im weiteren werden die Komponenten der Datenstruktur des Interaktions-Objektes beschrieben, wobei die sich ergebende Struktur gegebenenfalls erweitert wird.

Die erste Komponente, die in die Datenstruktur Interaktion(τ) aufgenommen werden soll, ist die Query Q_τ als Zeichensequenz, die von einem Agenten formuliert wurde. Optional kann der Agent bezeichnet und in die Datenstruktur aufgenommen werden, doch standardmäßig soll davon abgesehen werden, wäh-

rend bei der Clusterung von Agenten zu Nutzergruppen dies sinnvoll ist, wenn erwartet wird, dass ein Agent in Zukunft wieder eine Interaktion einleiten wird, bzw. wenn langfristige Informationsbedürfnisse wie beim Information-Filtering betrachtet werden.

Wird die Query Q_τ aufgenommen, so kann auch ein Queryvektor aufgenommen werden, wobei zunächst der Initialisierungs-Queryvektor $q_\tau^{t=0} = A_{IR(Q)}(Q_\tau)$ erwogen werden kann. Doch selbst wenn kein Queryvektor-Relevanz-Feedback durchgeführt wird, bei dem in jeder Iteration t ein adaptierter Queryvektor q_τ^t erzeugt wird, kann nach dem Ende der Interaktion mit Hilfe aller durch den Agenten bewerteter Dokumentvektoren ein optimierter Queryvektor erzeugt werden. Dieser soll vereinfachend mit q_τ bezeichnet und in die Interaktions-Objektstruktur aufgenommen werden. Notwendig für dessen Erzeugung ist in jedem Fall die Stimulusmenge, die aus allen bewerteten Dokumentvektoren besteht, die während der Interaktion gebildet wurden, d.h. die bis zur Abbruch-Iteration t(end,τ) gesammelt wurde, und die vereinfachend mit $M_{DV(m),\tau} := M_{DV(m)}^{\leq t(end,\tau)}$ bezeichnet werden soll:

$$M_{DV(m),\tau} = \{m_{j,\tau} = (x_j, rel(x_j)_\tau) \mid j = 1, ..., \mu_{DV(m),\tau}\}. \quad (640)$$

Nicht unbedingt notwendig ist die Aufnahme des Initialisierungs-Queryvektors $q_\tau^{t=0}$, da neben Nachadaptionsverfahren, die aus $q_\tau^{t=0}$ mit Hilfe der Elemente aus $M_{DV(m),\tau}$ den Queryvektor q_τ erzeugen, auch Verfahren existieren, die q_τ on-scratch aus $M_{DV(m),\tau}$ erzeugen können. Beispielsweise kann $M_{DV(m),\tau}$ durch einen Relevanz-Schwellenwert in eine relevante $M_{DV(m),rel,\tau}$ und eine nicht relevante Teilmenge $M_{DV(m),\overline{rel},\tau}$ zerlegt werden, gefolgt von einer GNG-SOM-Adaptionsphase mit positiven und negativen Adaptionen, die auf einen Vektor wirken, der nicht gleich dem Initialisierungs-Queryvektor $q_\tau^{t=0}$ zu sein braucht, wie z.B. dem Mittelwertsvektor der Dokumentvektoren der Stimuli in $M_{DV(m),\tau}$. Mit zunehmender Länge der Adaptionsphase wird die Abhängigkeit von dem Initialisierungsvektor verringert, sodass ein Mittelwertsvektor oder ein zufällig initialisierter Vektor innerhalb der Region, in der die betrachteten Dokumentvektoren liegen, ebenso geeignet ist wie $q_\tau^{t=0}$. Fasst man diese drei Komponenten zusammen, so ergibt sich folgende Datenstruktur der Interaktions-Objekte:

$$I^{\leq T-1} = (Interaktion(\tau) = (Q_\tau, q_\tau, M_{DV(m),\tau}) \mid \tau = 1, ..., T-1). \quad (641)$$

Bei den Relevanzfeedback-Verfahren, die im Zentrum dieses Kapitels stehen, wird während der Interaktionen ein instanzbasiertes bzw. prototypbasiertes Approximationsmodell aufgebaut. Durch die Speicherung der Stimuli $M_{DV,\tau}$ ist das instanzbasierte Approximationsmodell $AM(rel(x) \mid M_{DV(m),\tau})$ spezifiziert. Wird ein prototypbasiertes Modell gebildet und verwendet, so ist es sinnvoll, das GNG-SOM-Endmodell $N_{DV}^{t(end,\tau)}$ ebenfalls aufzunehmen, das vereinfachend durch $N_{DV,\tau}$ bezeichnet werden soll, sodass sich für die Struktur der Interaktionsobjekte ergibt:

$$I^{\leq T-1} = (Interaktion(\tau) = (Q_\tau, q_\tau, M_{DV(m),\tau}, N_{DV,\tau}) \mid \tau = 1, ..., T-1). \quad (642)$$

Durch die Bezeichnung $N_{DV,\tau}$ ist noch nichts über die Art der prototypbasierten GNG-SOMs gesagt, d.h. ob sie während der Interaktionen beginnend bei t=0 on-scratch aufgebaut werden, wenn z.B. keine globale Dokumentvektorenklassifikation N_{DV} vorliegt. Ein Modell $N_{DV,\tau}$ kann auch aus der Gesamt-GNG-SOM N_{DV} durch verschiedene Methoden abgeleitet werden, wobei prinzipiell die Adaption der Stützpunkte im DVR und im Relevanzraum bzw. die Erhaltung der Stützpunkte im DVR und die Adaption der Stützpunkte im Relevanzraum unterschieden werden kann. Im weiteren soll festgelegt werden, dass ein

Modell $N_{DV,\tau}$ die Gewichtsvektorenverteilung von N_{DV} übernimmt, und dass diejenigen Neurone aus $N_{DV,\tau}$ eine erweiterte Datenstruktur besitzen, denen mindestens ein Stimulus aus $M_{DV(m),\tau}$ zuordenbar ist, indem sie eine Relevanzschätzung am Stützpunkt dieses Gewichtsvektors besitzen. Es soll vereinbart werden, dass die Relevanzschätzung jeweils durch das gesamte instanzbasierte Approximationsmodell durchgeführt wird. D.h. ein Gewichtsvektor $w(x_{DV})_{k,\tau}$ wird durch $AM(rel(x) \mid M_{DV(m),\tau})$ bewertet, wodurch eine Relevanzschätzung $rel(w(x_{DV})_k)_\tau\text{^}$ erzeugt wird. Das Neuron $n_{DV,k,\tau}$ zeichnet sich dadurch aus, dass mindestens ein Element in seiner lokalen Dokumentvektorenmenge $M_{DV,k}$ eine Relevanzbewertung durch den Agenten besitzt, wobei kein Index τ verwendet wird, da durch die Übernahme der Stützpunkte im DVR alle Neurone $n_{DV,k,\tau}$ für unterschiedliche τ den gleichen Gewichtsvektor, die gleiche Verbindungsstruktur, und somit auch die gleiche Voronoi-Region und die gleichen darin enthaltenen Dokumentvektoren besitzen. Dies bedeutet, dass die Relevanzschätzung für alle Neurone durchgeführt wird, deren lokale Stimulusmenge $M_{DV(m),k,\tau}$ nicht leer ist:

$$rel(w(x_{DV})_k)_\tau\text{^} = rel(w(x_{DV})_k \mid AM(rel(x) \mid M_{DV,\tau})) =$$
$$1/\nu_{k,\tau} \sum_j h(d_{DVR}(x_{j,\tau}, w(x_{DV})_{k,\tau})) * rel(x_{j,\tau}),$$
$$\nu_{k,\tau} = \sum_j h(d_{DVR}(x_{j,\tau}, w(x_{DV})_{k,\tau})),\ \forall m_{j,\tau} \in M_{DV,\tau},$$
$$\forall n_{DV,k,\tau} \in N_{DV,\tau}: M_{DV(m),k,\tau} \neq \emptyset. \quad (643)$$

Alle anderen Neurone mit $M_{DV(m),k,\tau} = \emptyset$ besitzen keine Relevanzschätzung, doch bei Bedarf können einzelnen Neuronen dieser Art Relevanzschätzungen auf die gleiche Weise zugeordnet werden, wie den anderen Neuronen, indem das instanzbasierte Approximationsmodell angewendet wird. Das Modell $N_{DV,\tau}$ besitzt mit seinen zwei Typen von Neuronen somit die folgende Struktur:

$$N_{DV,\tau} = \{n_{DV,k} \mid k = 1, ..., \mu_{N,DV,\tau}\} \text{ mit} \quad (644)$$

$$n_{DV,k} = \begin{cases} \{(w(x_{DV})_k, rel(w(x_{DV})_k)_\tau\text{^}, M_{DV,k}, M_{DV(m),k,\tau}, C_{D,k}), & \text{wenn } M_{DV(m),k,\tau} \neq \emptyset, \\ \{(w(x_{DV})_k, M_{DV,k}, M_{DV(m),k,\tau}, C_{D,k}), & \text{wenn } M_{DV(m),k,\tau} = \emptyset. \end{cases} \quad (645)$$

Ausgangspunkt der weiteren Betrachtungen ist zum Zeitpunkt T eine Query Q_T, die durch die Indexierungsfunktion $A_{IR(Q)}$ auf einen Queryvektor $q_T^{t=0}$ abgebildet wird, sodass eine provisorische Beschreibung der Interaktion durch Interaktion(T) = (Q_T, $q_T^{t=0}$, _, _) gegeben werden kann. Da von einer globalen Dokumentvektorenklassifikation mit N_{DV} ausgegangen wird, folgt die Bestimmung des Gewinner-Neurons $n_{DV,s(1|T)}^{t=0}$ aus N_{DV} und die Festlegung der primären Dokumentvektoren-Ergebnismenge $DVM_{prim,T}^{t=0}$, die standardmäßig als die Menge der Dokumentvektoren bestimmt wird, die in der Voronoi-Region $R\tilde{}_{DV,s(1|T)}^{t=0}$ des Gewinner-Neurons liegen, d.h.:

$$DVM_{prim,T}^{t=0} := M_{DV,s(1|T)}. \quad (646)$$

4.5.1) Selektionsverfahren für Interaktionsobjekte bei Mono- und Polyrepräsentation

Die prinzipielle Frage besteht in einem Auswahlkriterium der Interaktionsobjekte Interaktion(τ) bei Kenntnis von Interaktion(T) = (Q_T, $q_T^{t=0}$, _, _). Bei einer Polyrepräsentation der Interaktionsnutzung, die alle Interaktionsobjekte nutzt, entfällt die Notwendigkeit der Festlegung eines Auswahlkriteriums, es soll

jedoch davon ausgegangen werden, dass immer eine Teilmenge bzw. -liste aus $I^{\leq T-1}$ spezifiziert wird, die mit $I_T^{\leq T-1}$ bezeichnet werden soll. Bei einer Monorepräsentation enthält diese Liste genau ein Element, während bei einer Polyrepräsentation der Interaktionsnutzung mehrere Elemente aus $I^{\leq T-1}$ spezifiziert werden.

Grundlage der Auswahl eines oder mehrerer Elemente muss eine Definition von Ähnlichkeit zwischen dem Interaktionsobjekt Interaktion(T) und den Interaktionsobjekten Interaktion(τ) sein, wobei berücksichtigt werden muss, dass Interaktion(T) zu diesem Zeitpunkt unvollständig ist. D.h. eine Ähnlichkeit muss auf der Ähnlichkeit der Queries Q_T und Q_τ bzw. der Queryvektoren $q_T^{t=0}$ und q_τ durchgeführt werden. Eine Ähnlichkeit auf der Basis von Stimulusmengen $M_{DV(m),T}$ und $M_{DV(m),\tau}$ bzw. Modellen $N_{DV,T}$ und $N_{DV,\tau}$ kann allein schon deswegen nicht durchgeführt werden, da zu dem betreffenden Zeitpunkt die Komponenten $M_{DV(m),T}$ und $N_{DV,T}$ nicht vorliegen, zumal weiterführende Überlegungen notwendig wären, wie Ähnlichkeiten von Stimulusmengen und von GNG-SOMs sinnvoll definierbar wären. Da es sich bei Stimulusmengen und Modellen um Objekte mit vielen Einzelkomponenten handelt, würde eine Ähnlichkeitsfunktion auf der Basis dieser Einzelkomponenten, d.h. Stützpunkten, basieren. Im günstigsten Fall könnte eine eindeutige Zuordnung von Komponenten des einen Objektes zu Komponenten des anderen Objektes durchgeführt werden, sodass die Anzahl der Berechnungen gleich der Anzahl der Komponenten eines Objektes ist. Im ungünstigsten Fall müsste jede Komponente des ersten Objektes mit allen Komponenten des zweiten Objektes verglichen werden, sodass die Gesamtanzahl sich aus dem Produkt der Anzahl der Komponenten der beiden Objekte ergibt. Im Gegensatz hierzu berechnet sich die Interaktionen-Ähnlichkeit auf der Basis der Queries bzw. der Queryvektoren aus genau einer Ähnlichkeitsfunktion, sodass diese Vorgehensweise auch wesentlich effizienter ist, als potentielle Ähnlichkeiten auf der Basis von Stimulusmengen und Modellen.
Es soll vereinbart werden, dass die Ähnlichkeit bzw. die Distanz zwischen den Queryvektoren $q_T^{t=0}$ und q_τ verwendet werden soll, um den oder die Elemente der Ergebnisliste $IL_T^{\leq T-1}$ aus $I^{\leq T-1}$ zu spezifizieren:

$$d_{DVR}(q_T^{t=0}, q_\tau), \forall \text{Interaktion}(\tau) \in I^{\leq T-1}. \tag{647}$$

Bei einer Monorepräsentation wird die Interaktion Interaktion($\tau_{T(min)}$) ausgewählt, deren Queryvektor $q_{\tau T(min)}$ den geringsten Abstand von $q_T^{t=0}$ besitzt:

$$IL_T^{\leq T-1} = (\text{Interaktion}(\tau_{T(min)})), \text{ mit}$$
$$d_{DVR}(q_T^{t=0}, q_{\tau T(min)}) = \min\{d_{DVR}(q_T^{t=0}, q_\tau), \forall \text{Interaktion}(\tau) \in I^{\leq T-1}\}. \tag{648}$$

Grundlage einer Polyrepräsentation kann die Bildung einer Liste von Interaktionen sein, geordnet nach steigender Distanz ihres nachadaptierten Queryvektors zu dem Initialisierungs-Queryvektor $q_T^{t=0}$:

$$IL_T^{\leq T-1} = (\text{Interaktion}(p) \in I^{\leq T-1} \mid d_{DVR}(q_T^{t=0}, q_p) < d_{DVR}(q_T^{t=0}, q_{p+1}); p = 1, ..., T-1). \tag{649}$$

Aus der Liste $IL_T^{\leq T-1}$ können die ersten δ Elemente ausgewählt werden, wodurch die Teilliste $IL_{T(\delta)}^{\leq T-1}$ erzeugt wird:

$$IL_{T(\delta)}^{\leq T-1} = (\text{Interaktion}(p) \in IL_T^{\leq T-1} \mid d_{DVR}(q_T^{t=0}, q_p) < d_{DVR}(q_T^{t=0}, q_{p+1}); p = 1, ..., \delta). \tag{650}$$

Alternativ zu der Festlegung des externen Parameters δ kann eine externe Distanzschwelle $S_{d(q,\tau)}$ festgelegt werden, sodass die ersten Interaktionen aus $IL_T^{\leq T-1}$ in die Liste $IL_{T(S(d(q,\tau)))}^{\leq T-1}$ übernommen werden, bis die Distanz $d_{DVR}(q_T^{t=0}, q_\tau)$ zum ersten Mal größer als $S_{d(q,\tau)}$ wird.

$$IL_{T(S(d(q,\tau)))}^{\leq T-1} = (\text{Interaktion}(p) \in IL_T^{\leq T-1} \mid d_{DVR}(q_T^{t=0}, q_p) < d_{DVR}(q_T^{t=0}, q_{p+1}); \; d_{DVR}(q_T^{t=0}, q_p) \leq S_{d(q,\tau)}). \quad (651)$$

Der Nachteil einer solchen Distanzschwelle besteht darin, dass sie so klein gewählt werden kann, dass keine Interaktion aus $IL_T^{\leq T-1}$ diese Bedingung erfüllt, und somit $IL_{T(S(d(q,\tau)))}^{\leq T-1}$ leer wird. In diesem Fall muss eine Strategie mit einem externen Parameter δ angewendet werden, oder es wird auf Relevanzschätzungen in t=0 verzichtet, wobei ein Ranking auf der Basis der Distanzen $d_{DVR}(q_T^{t=0}, x_j^{t=0})$ gebildet wird.

Liegt eine Dokumentvektorenklassifikation durch eine globale GNG-SOM N_{DV} vor, so ist mit Hilfe der Voronoi-Zerlegung ein parameterfreies Vorgehen möglich, bei dem weder δ noch eine Distanzschwelle $S_{d(q,\tau)}$ extern festgelegt werden muss. Hierzu wird für das Gewinner-Neuron $n_{DV,s(1|T)}^{t=0}$, das durch den Queryvektor $q_T^{t=0}$ festgelegt ist, die Menge der Interaktionen aus $I^{\leq T-1}$ ermittelt, deren nachadaptierte Queryvektoren in der Voronoi-Region $R^{\sim}_{DV,s(1|T)}$ liegen, d.h. es werden die Interaktionen ausgewählt, deren Queryvektor q_τ in der gleichen Voronoi-Region wie $q_T^{t=0}$ liegen. Wird ein solches Vorgehen erwogen, so ist es sinnvoll, die Beschreibung einer Interaktion um das Gewinner-Neuron zu erweitern, da auf diese Weise in T keine Distanzen berechnet werden müssen, sondern nur die Indentität bzw. Nicht-Identität der korrespondierenden Komponenten von Interaktion(τ) festgestellt werden muss:

$$I^{\leq T-1} = (\text{Interaktion}(\tau) = (Q_\tau, M_{DV(m),\tau}, q_\tau, n_{DV,s(1|\tau)}, N_{DV,\tau}) \mid \tau = 1, ..., T-1). \quad (652)$$

Eine geordnete Ergebnisliste $IL_{s(1|T)}^{\leq T-1}$ ergibt sich aus $I^{\leq T-1}$, indem für jede Interaktion Interaktion(τ) geprüft wird, ob $n_{DV,s(1|p)} = n_{DV,s(1|T)}^{t=0}$ erfüllt ist. Ist dem so, dann wird Interaktion(p) in $IL_{s(1|T)}^{\leq T-1}$ kopiert und es wird Interaktion(p+1) geprüft, ansonsten wird Interaktion(p) nicht in die Liste $IL_{s(1|T)}^{\leq T-1}$ kopiert, gefolgt von der Prüfung von Interaktion(p+1) bis Interaktion(T-1):

$$IL_{s(1|T)}^{\leq T-1} = (\text{Interaktion}(p) \in I^{\leq T-1} \mid n_{DV,s(1|p)} = n_{DV,s(1|T)}^{t=0}). \quad (653)$$

Alternativ kann die Prüfung über die Lokalisierung des Queryvektors q_p in der Voronoi-Region $R^{\sim}_{DV,s(1|T)}^{t=0}$ geprüft werden, falls das Gewinner-Neuron nicht in die Struktur der Interaktionsobjekte aufgenommen wurde:

$$IL_{s(1|T)}^{\leq T-1} = (\text{Interaktion}(p) \in I^{\leq T-1} \mid q_p \in R^{\sim}_{DV,s(1|T)}^{t=0}). \quad (654)$$

Das Problem dieser Vorgehensweise besteht wie der Verwendung einer Distanzschwelle darin, dass die jeweilige Ergebnisliste $IL_{s(1|T)}^{\leq T-1}$ leer sein kann. In einem solchen Fall kann die Interaktion Interaktion($\tau_{T(min)}$) ermittelt werden, deren Queryvektor $q_{\tau T(min)}$ den kleinsten Abstand von $q_T^{t=0}$ besitzt. Im Rahmen einer Polyrepräsentation sollen jedoch mehrere Interaktionsobjekte spezifiziert werden. Mit diesem Queryvektor $q_{\tau T(min)}$ ist das Gewinner-Neuron $n_{DV,s(1|\tau T(min))}$ spezifiziert, sodass alle Interaktionobjekte aus $I^{\leq T-1}$ ermittelt werden können, deren Queryvektor ebenfalls in der Voronoi-Region $R^{\sim}_{DV,s(1|\tau T(min))}^{t=0}$ liegen:

$$IL_{s(1|\tau T(min))}^{\leq T-1} = (\text{Interaktion}(p) \in IL^{\leq T-1} \mid q_p \in \tilde{R}_{DV,s(1|\tau T(min))}^{t=0}). \quad (655)$$

4.5.2) Nutzung von Stimulusmengen vergangener Interaktionen

Entsprechend der Struktur einer Interaktion soll zunächst die Nutzung der bewerteten Dokumentvektoren aus den vergangenen Iterationen dargestellt werden, d.h. die Stimuli aus allen $M_{DV(m),\tau}$ können im Prinzip für die Interaktion T genutzt werden.

4.5.2.1) Ergebnismengen durch lokale Operationen in T

Wird davon ausgegangen, dass $q_T^{t=0}$ lokal das Gewinner-Neuron $n_{DV,s(1|T)}^{t=0}$ spezifiziert, und dass $DVM_{prim,T}^{t=0}$ als Dokumentvektorenmenge $M_{DV,s(1|T)}$ verwendet wird, so kann versucht werden, jedem Element $x_j^{t=0}$ aus $DVM_{prim,T}^{t=0}$ bei einer Monorepräsentation ein Relevanzwert einer anderen Interaktion bzw. mehrere Relevanzwerte aus verschiedenen Interaktionen zuzuordnen.

4.5.2.1.1) Monorepräsentation von Relevanzwerten

Wird eine solche lokale Vorgehensweise in Kombination mit einer Monorepräsentation verwendet, so wird die Interaktion $\tau_{T(min)}$ verwendet, in deren Interaktionsobjekt Interaktion($\tau_{T(min)}$) der Queryvektor $q_{\tau T(min)}$ mit der kleinsten Distanz zu $q_T^{t=0}$ zu finden ist, wobei nicht alle Dokumentvektoren ein korrespondierendes Element in $M_{DV(m),\tau T(min)}$ besitzen müssen. Würde der Initialisierungs-Queryvektor $q_\tau^{t=0}$ anstatt der nachadaptierte Queryvektor $q_{\tau T(min)}$ verwendet, und lägen $q_T^{t=0}$ und $q_\tau^{t=0}$ in der gleichen Voronoi-Region, so wären allen Dokumentvektoren in dieser Voronoi-Region eine Relevanzschätzung in der Interaktion Interaktion($\tau_{T(min)}$) zugeordnet worden, da alle diese Dokumentvektoren in die primäre Ergebnismenge $DVM_{prim,\tau T(min)}^{t=0}$ aufgenommen wurden. Da jedoch der nachadaptierte Queryvektor q_τ verwendet wird, wurde nicht notwendig jedem Dokumentvektor in der Voronoi-Region $\tilde{R}_{DV,s(1|\tau)}$ während der Interaktion $\tau_{T(min)}$ ein Relevanzwert zugeordnet.

Es stellt sich dabei die Frage, was mit den Dokumentvektoren aus $DVM_{prim,T}^{t=0}$ geschehen soll, die kein korrespondierendes Element in $M_{DV(m),\tau T(min)}$ besitzen, d.h. für die kein Relevanzwert vorliegt. Eine Möglichkeit wäre diese Elemente zu ignorieren, und eine korrigierte, primäre Ergebnismenge ohne diese Elemente zu erzeugen, gefolgt von der Umwandlung in eine sekundäre Ergebnisliste entsprechend fallenden Relevanzwerten. Das Problem dieser Vorgehensweise besteht darin, dass zu wenige Elemente aus der ursprünglichen Ergebnismenge $DVM_{prim,T}^{t=0} = M_{DV,s(1|T)}$ einen Relevanzwert besitzen könnten bzw. im Extremfall kein einziges Element, sodass die korrigierte Dokumentvektorenmenge leer wäre. Aus diesem Grunde soll das Ignorieren dieser Teilmenge ausgeschlossen werden.

Alternativ hierzu können allen Dokumentvektoren aus $M_{DV,s(1|T)}$, denen kein Relevanzwert in der Interaktion $\tau_{T(min)}$ zugeordnet wurde, eine Relevanzschätzung zugeordnet werden, die gleichberechtigt neben den Relevanzwerten verwendet wird, um ein Ranking, d.h. die sekundäre Ergebnisliste $DV_{sec,T}^{t=0}$ aufzubauen. Bei dieser Vorgehensweise stellt sich die Frage, durch welches Relevanz-Approximationsmodell

die Schätzungen durchgeführt werden sollen. Naheliegend ist das Modell $N_{DV,\tau T(min)}$ innerhalb der Datenstruktur des Interaktionsobjektes Interaktion($\tau_{T(min)}$). Bei der Erzeugung von Relevanzschätzungen wird dann jedoch ein Approximationsmodell von vorangegangenen Interaktionen genutzt, sodass diese Vorgehensweise eine Mischung aus der Nutzung von Stimulusmengen und Approximationsmodellen darstellt (siehe Abschnitt 4.5.4)).

Nach der Bewertung der Dokumente durch den Agenten, die zu den Elementen aus $DV_{sec,T}^{t=0}$ korrespondieren, liegt die erste Stimulusmenge $M_{DV(m),T}^{t=0}$ vor, die ein instanzbasiertes Approximationsmodell AM(rel(x) | $M_{DV(m),T}^{t=0}$) spezifiziert, mit dem ausgewählte Gewichtsvektoren von N_{DV} bewertet werden können, wie der Gewichtsvektor $w(x_{DV})_{s(1|T)}^{t=0}$ des Gewinner-Neurons $n_{DV,s(1|T)}^{t=0}$ und seiner Nachbarn, wodurch das prototypbasierte Modell $N_{DV,T}^{t=0}$ erzeugt wird. Dieses korrespondiert zu dem ersten prototypbasierten Approximationsmodell AM(rel(x) | $N_{DV,T}^{t=0}$) in T, sodass damit die Initialisierungs-Iteration beendet wird, und t=1 begonnen wird.

Im weiteren ist zu klären, ob die Stimulusmengen der vorangegangenen Iterationen in den weiteren Iterationen t > 0 noch verwendet werden sollen. Im einfachsten Fall wird auf Interaktionen τ mit den Stimulusmengen $M_{DV(m),\tau}$ verzichtet, und die weiteren Iterationen werden ausschließlich mit Hilfe der Stimuli in $M_{DV(m),T}^{t}$ und den Modellen $N_{DV,T}^{t}$ durchgeführt. Soll die Stimulusmenge $M_{DV(m),\tau T(min)}$ für ein oder mehrere Iterationen weiter verwendet werden, so wird für t=1 wie üblich die primäre Dokumentvektoren-Ergebnismenge $DVM_{prim,T}^{t=1}$ ermittelt, deren Elemente durch AM(rel(x) | $N_{DV,T}^{t=0}$) eine Relevanzschätzung erhalten. Die Ergebnismenge $DVM_{prim,T}^{t=1}$ lässt sich im Rahmen der Monorepräsentation bezüglich des Iterationsobjektes Interaktion($\tau_{T(min)}$) in eine Teilmenge $DVM_{prim,T(min)+}^{t=1}$ zerlegen, deren Elemente Stimuli aus $M_{DV(m),\tau T(min)}$ zugeordnet werden können, d.h. die Dokumentvektorenkomponente eines solchen Stimulus entspricht einem dieser Dokumentvektoren aus $DVM_{prim,T(min)+}^{t=1}$. Die Restmenge $DVM_{prim,T(min)-}^{t=1}$ besteht aus den Dokumentvektoren, für die kein Relevanzwert aus der Interaktion $\tau_{T(min)}$ zur Verfügung steht. Für das weitere Vorgehen stehen Alternativen zur Verfügung, wie z.B.:

1) Eine Relevanzschätzung durch AM(rel(x) | $N_{DV,T}^{t=0}$) wird nur für die Elemente aus der Teilmenge $DVM_{prim,T(min)-}^{t=1}$ durchgeführt, und die Relevanzwerte aus der Interaktion $\tau_{T(min)}$ für die Elemente von $DVM_{prim,T(min)+}^{t=1}$ werden gleichberechtigt verwendet, um eine geordnete Liste nach fallenden Werten zu erzeugen, die als sekundäre Ergebnisliste $DV_{sec,T}^{t=1}$ verwendet wird.
2) Die Dokumentvektoren aus $DVM_{prim,T(min)+}^{t=1}$ berechnen einen Gesamt-Relevanzwert aus dem Wert, der ihnen in der Interaktion $\tau_{T(min)}$ zugeordnet wurde und der Relevanzschätzung durch AM(rel(x) | $N_{DV,T}^{t=0}$). Die Dokumentvektoren aus $DVM_{prim,T(min)-}^{t=1}$ behalten ihre Relevanzschätzung durch AM(rel(x) | $N_{DV,T}^{t=0}$) bei. Die Relevanzwerte der Elemente beider Teilmengen werden gleichberechtigt verwendet, um eine geordnete Liste $DV_{sec,T}^{t=1}$ nach fallenden Werten zu erzeugen.
3) Neben dem Approximationsmodell AM(rel(x) | $N_{DV,T}^{t=0}$) wird das Modell AM(rel(x) | $N_{DV,\tau T(min)}$) aus der Interaktion $\tau_{T(min)}$ verwendet, um die Dokumentvektoren aus der gesamten Menge $DVM_{prim,T}^{t=1}$ (bzw. aus der Teilmenge $DVM_{prim,T(min)-}^{t=1}$) zu bewerten. Die Schätzungen werden mit den Relevanzwerten aus der Iteration $\tau_{T(min)}$ aggregiert, und mit den Gesamtwerten wird die geordnete Liste $DV_{sec,T}^{t=1}$ erzeugt.

Nach der Bewertung durch den Agenten ergibt sich die Stimulusmenge $M_{DV(m),T}^{t=1}$, die vereinigt mit $M_{DV(m),T}^{t=0}$ die Gesamtmenge $M_{DV(m),T}^{t\leq1}$ ergibt, mit der $N_{DV,T}^{t=1}$ und somit das prototypbasierte Approximationsmodell AM(rel(x) | $N_{DV,T}^{t=1}$) generiert wird, sodass t=2 begonnen werden kann.

Ein Abbruchkriterium muss nicht durch die Erschöpfung der Stimuli aus $M_{DV(m),\tau T(min)}$ abgeleitet werden, d.h. wenn alle Dokumentvektoren dieser Stimuli in einer tertiären Ergebnisliste vorlagen, da in allen dargestellten Vorgehensweisen Approximationsmodelle verwendet werden, die andere Dokumentvektoren schätzen können. Somit kann ein Abbruchkriterium verwendet werden, das greift, wenn alle Schätzungen und alle Relevanzwerte der Elemente einer momentan vorliegenden Ergebnismenge kleiner als ein Relevanzschwellenwert sind.

4.5.2.1.2) Polyrepräsentation von Relevanzwerten

Im Rahmen der Polyrepräsentation wird standardmäßig die Interaktionsobjektliste $IL_T^{\leq T-1}$ gebildet, in der die Objekte Interaktion(p) nach steigender Distanz ihres nachadaptierten Queryvektors zu $q_T^{t=0}$ sortiert sind. Wird ein externer Parameter δ vorgegeben, so werden die ersten δ Elemente aus $IL_T^{\leq T-1}$ ausgewählt, welche die Liste $IL_{T(\delta)}^{\leq T-1}$ bilden. Mit Hilfe der Stimuli aus den Mengen $M_{DV(m),p}$ der Objekte Interaktion(p) aus $IL_{T(\delta)}^{\leq T-1}$ kann versucht werden, jedem Dokumentvektor $x_j^{t=0}$ aus $DVM_{prim,T}^{t=0}$ Relevanzwerte zuzuordnen. Da die Elemente der Mengen $M_{DV(m),p}$ unterschiedliche Dokumentvektorenkomponenten besitzen, bedeutet dies, dass wahrscheinlich nicht jeder Dokumentvektor $x_j^{t=0}$ δ Relevanzwerte zugeordnet bekommt, und dass unterschiedliche Elemente aus $DVM_{prim,T}^{t=0}$ auch eine unterschiedliche Anzahl von Relevanzwerten erhalten.

Nicht auszuschließen ist der Fall, dass bei einem gegebenen Parameter δ einzelnen Dokumentvektoren aus $DVM_{prim,T}^{t=0}$ kein Relevanzwert zugeordnet werden kann. Im Extremfall kann ein Dokumentvektor existieren, der in der gesamten Geschichte des IRS von keinem Agenten bewertet wurde, sodass einem solchen Element unabhängig von der Wahl von δ kein Relevanzwert aus einem Interaktionsobjekt Interaktion(τ) zugeordnet werden kann. Wird von diesem Extremfall abgesehen, so wird durch die Polyrepräsentation jedoch die Wahrscheinlichkeit erhöht, dass einem Element aus $DVM_{prim,T}^{t=0}$ mindestens ein Relevanzwert aus der Vergangenheit zugeordnet werden kann. Werden mehrere Relevanzwerte zugeordnet, so werden diese in einem Zwischenschritt aggregiert, sodass jeder Dokumentvektor ein Gesamtrelevanzwert besitzt, die nach fallenden Werten sortiert die sekundäre Ergebnisliste $DV_{sec,T}^{t=0}$ bilden, die in t=0 gleich $DV_{ter,T}^{t=0}$ gesetzt wird.

Dokumentvektoren, denen in t=0 kein Relevanzwert eines Interaktionsobjektes Interaktion(p) aus $IL_{T(\delta)}^{\leq T-1}$ zugeordnet werden kann, für die jedoch ein korrespondierender Stimulus in einer der Interaktionsobjekte aus $IL_T^{\leq T-1} \setminus IL_{T(\delta)}^{\leq T-1}$ existiert, könnten durch eine Ausnahmeregel einen solchen Wert erhalten. D.h. es wird für Interaktion(δ+1) geprüft, ob ein solcher Dokumentvektor einen korrespondierenden Stimulus in $M_{DV(m),\delta+1}$ besitzt. Ist dies der Fall, so wird der Relevanzwert des Stimulus verwendet, und die weitere Suche wird abgebrochen. Ist dies nicht der Fall, so wird in Interaktion(δ+2), Interaktion(δ+3), ... die Suche nach einem korrespondierenden Stimulus fortgesetzt, bis zum ersten Mal ein solcher Stimulus gefunden wird.

Alternativ zu dieser Ausnahmeregel könnte ein Dokumentvektor, dem kein Relevanzwert aus den Interaktionsobjekten aus $IL_{T(\delta)}^{\leq T-1}$ zugeordnet werden kann, einfach an das Ende des Rankings gesetzt werden, d.h. ihm wird der letzte Platz in der sekundären Ergebnisliste $DV_{sec,T}^{t=0}$ zugeordnet, sodass das korrespondierende Dokument dem Agenten als letztes präsentiert wird. Bei mehreren dieser Dokumentvektoren kann der letzte Rankingplatz mit mehreren Dokumentvektoren bzw. Dokumenten belegt werden.

Allgemein könnte das Prinzip verfolgt werden, dass bei einer Monorepräsentation die eine Stimulusmenge $M_{DV(m),\tau T(min)}$ ausschließlich für die Initialisierungs-Iteration t=0 verwendet wird, während bei einer Polyrepräsentation die Stimulusmengen $M_{DV(m),p}$, p = 1, ..., δ, für weitere Iterationen genutzt werden, da eine größere Chance besteht, dass die Dokumentvektoren, die in den primären Ergebnismengen $DVM_{prim,T}^{t}$, t > 0, nachgewiesen werden, mindestens einen korrespondierenden Stimulus in einer der Mengen $M_{DV(m),p}$ besitzen.

Für alle Iterationen t>0 werden die jeweils aktualisierten Approximationsmodelle $AM(rel(x) \mid N_{DV,T}^{t-1})$ verwendet, um eine Relevanzschätzung für alle Elemente aus $DVM_{prim,T}^{t}$ zu erzeugen. Daneben sollen alle Relevanzwerte aus den Stimulusmengen $M_{DV(m),p}$ einbezogen werden, wobei die Relevanzwerte unterschiedlicher Interaktionsobjekte zu einem Gesamt-Relevanzwert aggregiert werden sollen. Dieser Gesamtrelevanzwert wird zusammen mit der Relevanzschätzung durch $AM(rel(x) \mid N_{DV,T}^{t-1})$ verwendet, indem ein gewichteter Mittelwert gebildet wird, wobei das Gewicht der Relevanzschätzung mit steigender Iterationsanzahl t steigt.

4.5.2.2) Ergebnismengen durch Übernahme aus ausgewählten Interaktionssmengen

Ein anderer Ansatzpunkt wird bei einer nicht lokalen Vorgehensweise verwendet, indem $q_T^{t=0}$ nicht die Dokumentvektorenmenge $M_{DV,s(1|T)}$ als primäre Ergebnismenge $DVM_{prim,T}^{t=0}$ spezifiziert, sondern indem bei einer Monorepräsentation die Stimuli aus $M_{DV(m),\tau T(min)}$ verwendet werden, deren Dokumentvektorenkomponenten insgesamt oder teilweise die primäre Ergebnismenge $DVM_{prim,T}^{t=0}$ der Initialisierungs-Iteration t=0 in der Interaktion T bilden. Naheliegend wäre z.B. aus $M_{DV(m),\tau T(min)}$ die δ Stimuli mit den besten Relevanzwerten zu ermitteln, unabhängig wo die Dokumentvektoren der Stimuli liegen, d.h. unabhängig ob sie in $\tilde{R}_{DV,s(1|T)}^{t=0}$ liegen oder nicht. Wird eine solche Vorgehensweise erwogen, so ist es sinnvoll, die Stimuli aus allen Mengen $M_{DV(m),\tau}$ nach fallenden Relevanzwerten zu ordnen, damit die ersten Elemente entsprechend einem externen Parameter δ bzw. einem Relevanzschwellenwert direkt übernommen werden können. Wird eine solche, nach Relevanzwerten geordnete Liste allgemein mit $ML_{DV(m),\tau}$ bezeichnet, so ergibt sich für die Struktur der Interaktionsobjekte:

$$I^{\leq T-1} = (\text{Interaktion}(\tau) = (Q_\tau, ML_{DV(m),\tau}, q_\tau, N_{DV,\tau}) \mid \tau = 1, ..., T-1). \quad (656)$$

Diese allgemeine Liste wird in die für T spezifische Liste $IL_T^{\leq T-1}$ durch Sortieren nach steigenden Distanzen umgewandelt.

$$IL_T^{\leq T-1} = (\text{Interaktion}(p) \in I^{\leq T-1} \mid d_{DVR}(q_T^{t=0}, q_p) < d_{DVR}(q_T^{t=0}, q_{p+1}); p = 1, ..., T-1). \quad (657)$$

4.5.2.2.1) Monorepräsentation der Interaktionsmenge

Bei einer Monorepräsentation wird das erste Element Interaktion($\tau_{T(min)}$) aus $IL_T^{\leq T-1}$ ausgewählt, und die ersten δ Elemente aus der Stimulusliste $ML_{DV(m),\tau T(min)}$ werden als Liste $ML_{DV(m),\tau T(min)(\delta)}$ direkt als tertiäre Ergebnisliste $DV_{ter,T}^{t=0}$ verwendet, d.h. durch das Vorsortieren nach Relevanzwerten und der fehlenden Durchführung von Relevanzschätzungen und Aggregationen von Relevanzwerten kann in t=0 auf eine Erzeugung einer primären Ergebnismenge und einer sekundären Ergebnisliste verzichtet werden. Es ist sinnvoll, bereits in t=0 die geordnete Stimulusliste $ML_{DV(m),\tau T(min)}$ mit einem Index t zu versehen, da auf diese Weise iterationsspezifische Restmengen leichter beschrieben werden können, d.h. die Gesamtliste wird als $ML_{DV(m),\tau T(min)}^{t=0}$ bezeichnet, und die ersten δ Elemente aus dieser Liste bilden in der gleichen Reihenfolge die Liste $ML_{DV(m),\tau T(min)(\delta)}^{t=0}$:

$$ML_{DV(m),\tau T(min)(\delta)}^{t=0} = (m_{j,\tau T(min)}^{t=0} = (x_{\tau T(min),j}^{t=0}, rel(x_{\tau T(min),j}^{t=0})) \mid rel(x_{\tau T(min),j}^{t=0} > rel(x_{\tau T(min),j+1}^{t=0}, j = 1, ..., \delta). \quad (658)$$

Die tertiäre Dokumentvektoren-Ergebnisliste $DV_{ter,T}^{t=0}$ ergibt sich als Liste der Dokumentvektoren der Stimuli aus $ML_{DV(m),\tau T(min)(\delta)}^{t=0}$ in der gleichen Reihenfolge wie in dieser Stimulusliste:

$$DV_{ter,T}^{t=0} = (x_{\tau T(min),j}^{t=0} \mid rel(x_{\tau T(min),j}^{t=0} > rel(x_{\tau T(min),j+1}^{t=0}, j = 1, ..., \delta). \quad (659)$$

Die korrespondierenden Dokumente werden dem Agenten in der gleichen Reihenfolge präsentiert, und nach der Bewertung ergibt sich die Stimulusmenge $M_{DV(m),T}^{t=0}$, mit der $N_{DV,T}^{t=0}$ und somit das prototypbasierte Approximationsmodell AM(rel(x) | $N_{DV,T}^{t=0}$) spezifiziert wird. In der nachfolgenden Iteration t=1 wird die Restliste aus $ML_{DV(m),\tau T(min)}^{t=0}$ und $ML_{DV(m),\tau T(min)(\delta)}^{t=0}$ gebildet, die als Grundliste für die Iteration t=1 verwendet wird, und die mit $ML_{DV(m),\tau T(min)}^{t=1}$ bezeichnet werden soll. Aus dieser Liste $ML_{DV(m),\tau T(min)}^{t=1}$ könnten wiederum die ersten δ Elemente als tertiäre Liste $DV_{ter,T}^{t=1}$ übernommen werden, doch berücksichtigt diese Vorgehensweise nicht die nun vorliegende erste Version des Approximationsmodells in T, d.h. von AM(rel(x) | $N_{DV,T}^{t=0}$). Eine Möglichkeit ist die Bewertung aller Elemente in der Restliste $ML_{DV(m),\tau T(min)}^{t=1}$ durch AM(rel(x) | $N_{DV,T}^{t=0}$), sodass jedes Element ein Relevanzwert aus der Iteration $\tau_{T(min)}$ und eine Relevanzschätzung aus der Iteration T besitzt. Diese beiden Relevanzwerte können aggregiert werden, wobei eine iterationsabhängige Gewichtung verwendet werden kann, die im Verlauf der Iteration t dem jeweiligen Modell AM(rel(x) | $N_{DV,T}^{t}$) ein wachsendes Gewicht zuordnet. Mit diesem aggregierten Relevanzwert werden die Stimuli der Liste $ML_{DV(m),\tau T(min)}^{t=1}$ nach fallenden Werten neu sortiert, wobei die sich ergebende Liste vereinfachend mit $ML_{DV(m),\tau T(min)(N)}^{t=1}$ bezeichnet werden soll. Diese Liste besitzt den Charakter einer sekundären Ergebnisliste, sodass $DV_{sec,T}^{t=1}$ aus den Dokumentvektoren der Stimuli aus $ML_{DV(m),\tau T(min)(N)}^{t=1}$ gebildet wird, die in der gleichen Reihenfolge übernommen werden. Das gleiche gilt für die tertiäre Ergebnismenge $DV_{ter,T}^{t=1}$, indem aus $DV_{sec,T}^{t=1}$ die ersten δ Dokumentvektoren übernommen werden.

Nach der Bewertung durch den Agenten entsteht die Stimulusliste $M_{DV(m),T}^{t=1}$, die vereinigt mit $M_{DV(m),T}^{t=0}$ die Gesamtmenge $M_{DV(m),T}^{t\leq 1}$ ergibt, die $N_{DV,T}^{t=1}$ und AM(rel(x) | $N_{DV,T}^{t=1}$) spezifiziert. Indem als Grundmenge $ML_{DV(m),\tau T(min)}$ verwendet wird, wird $M_{DV(m),T}^{t}$ im Laufe der Iterationen an diese Grundmenge angeglichen, da keine anderen Dokumentvektoren verwendet werden, was die Charakteristik dieser Vorgehensweise ausmacht. Ein Abbruch erfolgt somit durch das IRS, wenn die Restmenge aus $ML_{DV(m),\tau T(min)}^{t}$ bei einer Iteration leer wird. Sollte in diesem Fall vom Agenten noch kein

Abbruch gewünscht werden, so könnte aus der T-spezifischen Interaktionsobjektliste $IL_T^{\leq T-1}$ das zweite Element nach Interaktion($\tau_{T(min)}$) gewählt werden, das als Interaktion($\tau_{T(min)}+1$) bezeichnet werden kann. Im weiteren wird die Schnittmenge zwischen der Grundmenge $ML_{DV(m),\tau T(min)}$ und der Stimulusliste $ML_{DV(m),\tau T(min)+1}$ der zweitbesten Interaktion gebildet. Sollte diese nicht leer sein, werden die Dokumentvektorenkomponenten dieser Stimuli durch das in t aktuelle Modell $AM(rel(x) \mid N_{DV,T}^{t-1})$ bewertet, gefolgt von der Aggregation dieser Schätzung mit dem Relevanzwert aus der Interaktion $\tau_{T(min)}+1$, und der Bildung einer neuen geordneten Liste nach fallenden Relevanzwerten.

4.5.2.2.2) Polyrepräsentation der Interaktionsmenge

Bei einer Polyrepräsentation werden die ersten δ_1 Elemente Interaktion(p) aus $IL_T^{\leq T-1}$ ausgewählt, wodurch die Teilliste $IL_{T(\delta 1)}^{\leq T-1}$ entsteht:

$$IL_{T(\delta 1)}^{\leq T-1} = (\text{Interaktion}(p) \in I^{\leq T-1} \mid d_{DVR}(q_T^{t=0}, q_p) < d_{DVR}(q_T^{t=0}, q_{p+1}); p = 1, ..., \delta_1). \quad (660)$$

Der nächste Schritt betrifft die Stimuluslisten $ML_{DV(m),p}$, p = 1, ..., δ_1, der Interaktionsobjekte Interaktion(p) aus $IL_{T(\delta 1)}^{\leq T-1}$. Z.B. kann die Vereinigungsmenge aller Listenelemente erfolgen, die mehrfach auftretende Stimuli übernimmt, d.h. es können in dieser Menge Stimuli mit gleichen Dokumentvektorenkomponenten und unterschiedlichen Relevanzwerten vorliegen, ohne dass diese aggregiert werden:

$$M_{DV(m),T(\delta 1)} = \bigcup_p ML_{DV(m),p}, \forall ML_{DV(m),p} \in IL_{T(\delta 1)}^{\leq T-1}. \quad (661)$$

Es folgt die Sortierung nach fallenden Relevanzwerten, was als sekundäre Stimulus-Ergebnisliste betrachtet werden kann, die jedoch nicht spezifisch für eine Iteration sondern spezifisch für T ist:

$$ML_{DV(m),T(\delta 1)} = (m_j \in M_{DV(m),T(\delta 1)} \mid rel(x_j) > rel(x_{j+1})). \quad (662)$$

Aus dieser Liste wird die tertiäre Liste abgeleitet, indem z.B. die ersten δ_2 Elemente ausgewählt werden, wobei dies als iterationsspezifische Liste mit t=0 formuliert wird:

$$ML_{DV(m),ter,T(\delta 1)}^{t=0} = (m_j \in ML_{DV(m),T(\delta 1)} \mid rel(x_j) > rel(x_{j+1}), j = 1, ..., \delta_2). \quad (663)$$

Der sekundären und tertiären Stimulus-Ergebnisliste entspricht die sekundäre und tertiäre Dokumentvektoren-Ergebnisliste $DV_{sec,T}^{t=0}$ und $DV_{ter,T}^{t=0}$, indem die Dokumentvektorenkomponenten der Stimuli in der gleichen Reihenfolge übernommen werden:

$$DV_{sec,T}^{t=0} = (x_j \mid \forall m_j \in ML_{DV(m),sec,T(\delta 1)}: rel(x_j) > rel(x_{j+1})),$$
$$DV_{ter,T}^{t=0} = (x_j \in DV_{sec,T}^{t=0} \mid rel(x_j) > rel(x_{j+1}), j = 1, ..., \delta_2). \quad (664)$$

Es ergibt sich nach der Bewertung der korrespondierenden Dokumente die Stimulusmenge $M_{DV(m),T}^{t=0}$, mit der $N_{DV,T}^{t=0}$ und das Approximationsmodell $AM(rel(x) \mid N_{DV,T}^{t=0})$ spezifiziert wird. In der nachfolgenden Iteration t=1 wird die Restliste aus $ML_{DV(m),T(\delta 1)}$ und $ML_{DV(m),sec,T(\delta 1)}^{t=0}$ gebildet, die als Grundliste für die Iteration t=1 verwendet wird, die mit $ML_{DV(m),T(\delta 1)}^{t=1}$ bezeichnet werden soll. Aus der Liste $ML_{DV(m),T(\delta 1)}^{t=1}$ könnten wiederum die ersten δ_2 Stimuli für $ML_{DV(m),ter,T(\delta 1)}^{t=1}$ gezogen werden, doch dies berücksichtigt nicht das nun vorliegende Approximationsmodell $AM(rel(x) \mid N_{DV,T}^{t=0})$.

Die Berücksichtigung erfolgt, indem alle Stimuli $m_j^{t=1}$ aus $ML_{DV(m),T(\delta 1)}^{t=1}$ durch $AM(rel(x) \mid N_{DV,T}^{t=0})$ bewertet werden, sodass sich Relevanzschätzungen $rel(x_j)^{\wedge t=1}$ ergeben:

$$rel(x_j)^{\wedge t=1} = rel(x_j \mid AM(rel(x) \mid N_{DV,T}^{t=0})), \forall m_j^{t=1} \in ML_{DV(m),T(\delta 1)}^{t=1}. \quad (665)$$

Die Relevanzschätzungen $rel(x_j)^{\wedge t=1}$ werden in die Datenstruktur der Stimuli temporär integriert,

$$m_j = (x_j, rel(x_j), rel(x_j)^{\wedge t=1}), \quad (666)$$

gefolgt von der Aggregation mit dem vorliegenden Relevanzwert aus einer der vorangegangenen Interaktionen, wodurch $rel(x_j)^{t=1}$ entsteht, ohne dass hier auf die Art der Gewichtung der beiden Relevanzwerte eingegangen werden soll. Da die Reihenfolge durch das Sortieren der aggregierten Relevanzwerte erfolgt, soll mit $M_{DV(m),T(\delta 1)}^{t=1}$ die Menge der Stimuli aus $ML_{DV(m),T(\delta 1)} \setminus ML_{DV(m),sec,T(\delta 1)}^{t=0}$ bezeichnet werden, während $ML_{DV(m),T(\delta 1)}^{t=1}$ die geordnete Liste der Elemente aus $M_{DV(m),T(\delta 1)}^{t=1}$ bezeichnet, die nach fallendem aggregierten Relevanzwert gebildet wird. Diese Liste entspricht einer sekundären Stimulus-Ergebnisliste der Iteration t=1, aus der eine tertiäre Liste $ML_{DV(m),sec,T(\delta 1)}^{t=1}$ durch Auswahl der ersten δ_2 Stimuli gebildet wird, für die eine Dokumentvektoren-Ergebnisliste $DV_{ter,T}^{t=0}$ durch Übernahme der Dokumentvektorenkomponenten in der gleichen Reihenfolge gebildet wird.

Ein Abbruch von Seiten des IRS kann erfolgen, wenn die Restmenge $M_{DV(m),T(\delta 1)}^{t}$ in einer Iteration leer wird, was jedoch kaum eintreten wird, da in der Gesamtmenge $M_{DV(m),T(\delta 1)}$ die Stimuli aus δ_1 Interaktionen vereinigt war. Ein alternativer Abbruch könnte erfolgen, wenn alle aggregierten Relevanzwerte aus $M_{DV(m),T(\delta 1)}^{t}$ kleiner einem Schwellenwert sind.

Eine Alternative zu der Bildung der Vereinigungsmenge der Stimuluslisten $ML_{DV(m),p}$ aller Interaktionsobjekte aus $IL_{T(\delta 1)}^{\leq T-1}$ ist die Auswahl des oder der ersten Elemente aus den Einzellisten ohne Zurücklegen, wodurch die korrigierten Listen $ML_{DV(m),p}^{t=0}$, p = 1, ..., δ_1, entstehen. Die Vereinigung der ausgewählten Stimuli besitzt den Charakter einer primären Stimulus-Ergebnismenge $M_{DV(m),prim,T(\delta 1)}^{t=0}$, die für die Iteration t=0 spezifisch ist. Dieser Menge entspricht die primäre Dokumentvektoren-Ergebnismenge $DVM_{prim,T}^{t=0}$, indem die Dokumentvektorenkomponenten der Stimuli aus $M_{DV(m),prim,T(\delta 1)}^{t=0}$ ausgewählt werden. Entsprechend fallenden Relevanzwerten werden die Stimuli aus $M_{DV(m),prim,T(\delta 1)}^{t=0}$ geordnet, wodurch eine sekundäre Liste $ML_{DV(m),sec,T(\delta 1)}^{t=0}$ entsteht, welche der sekundären Liste $DV_{sec,T}^{t=0}$ entspricht. Je nach der Anzahl der ausgewählten Stimuli pro Einzelliste kann die sekundäre Liste direkt als tertiäre Liste verwendet werden oder es wird eine Auswahl der ersten Elemente aus $ML_{DV(m),sec,T(\delta 1)}^{t=0}$ bzw. $DV_{sec,T}^{t=0}$ als tertiäre Liste übernommen. Wird z.B. angenommen, dass die beiden ersten Elemente aus den Einzellisten ausgewählt werden, so liegen in $M_{DV(m),prim,T(\delta 1)}^{t=0}$, $DVM_{prim,T}^{t=0}$ und $DV_{sec,T}^{t=0}$ $2 * \delta_1$ Elemente vor. Ist die Sollanzahl der tertiären Liste δ_1, so werden die ersten δ_1 Elemente aus $ML_{DV(m),sec,T(\delta 1)}^{t=0}$ übernommen, sodass die korrespondierenden Dokumentvektoren in der gleichen Reihenfolge die Liste $DV_{ter,T}^{t=0}$ bilden.

Nach der Bewertung der zu $DV_{ter,T}^{t=0}$ korrespondierenden Dokumentliste ergibt sich die Stimulusmenge $M_{DV(m),T}^{t=0}$, mit der $N_{DV,T}^{t=0}$ und $AM(rel(x) \mid N_{DV,T}^{t=0})$ festgelegt ist. In t=1 wird zum einen die Restmenge $M_{DV(m),prim,T(\delta 1)}^{t=0} \setminus ML_{DV(m),sec,T(\delta 1)}^{t=0}$ mit δ_1 Elementen als erster Teil der primären Stimu-

lus-Ergebnismenge $M_{DV(m),prim,T(\delta 1)}^{t=1}$ verwendet. Der andere Teil ergibt sich, indem aus den korrigierten Einzellisten $ML_{DV(m),p}^{t=0}$, p = 1, ..., δ_1, jeweils das erste Element ohne Zurücklegen ausgewählt wird, wodurch die korrigierten Listen $ML_{DV(m),p}^{t=1}$ erzeugt werden. Nachdem $M_{DV(m),prim,T(\delta 1)}^{t=1}$ auf diese Weise vollständig ist, wird mit Hilfe von $AM(rel(x) \mid N_{DV,T}^{t=0})$ die Rangfolge der Elemente festgelegt, indem jeder Stimulus $m_j^{t=1}$ bewertet wird, wodurch die Schätzung $rel(x_j)^{\wedge t=1}$ erzeugt wird, die zusammen mit dem Relevanzwert $rel(x_j)$ zu $rel(x_j)^{t=1}$ aggregiert wird. Die Elemente aus $M_{DV(m),prim,T(\delta 1)}^{t=1}$ werden entsprechend den aggregierten Werten fallend geordnet, sodass die sekundäre Liste $ML_{DV(m),sec,T(\delta 1)}^{t=1}$ entsteht. Aus dieser werden die ersten δ_1 Stimuli in die tertiäre Liste übernommen, zu welcher die tertiäre Dokumentvektorenliste $DV_{ter,T}^{t=1}$ korrespondiert.

Das Abbruchkriterium von Seiten des IRS kann auch hier mit Hilfe eines Relevanzschwellenwertes festgelegt werden, indem ein Abbruch erfolgt, wenn alle Elemente aus $M_{DV(m),prim,T(\delta 1)}^{t=1}$ bzw. $ML_{DV(m),sec,T(\delta 1)}^{t=1}$ einen aggregierten Relevanzwert besitzen, der kleiner diesem Schwellenwert ist.

4.5.3) Nutzung von nachadaptierten Queryvektoren vergangener Interaktionen

Durch die Indexierungsfunktion $A_{IR(Q)}$ wird zu Beginn der Interaktion T die Query Q_T auf den Queryvektor $q_T^{t=0}$ abgebildet, gefolgt von der Identifizierung des Gewinner-Neurons $n_{DV,s(1|T)}^{t=0}$ bei einer Dokumentvektorenklassifikation durch eine globale GNG-SOM N_{DV}. Bei der Nutzung von Interaktionsobjekten Interaktion(τ) aus $I^{\leq T-1}$ wird die Distanz zwischen $q_T^{t=0}$ und den nachadaptierten Queryvektoren q_τ als Selektionskriterium verwendet, wobei Interaktion($\tau_{T(min)}$) als das Interaktionsobjekt ausgewählt wird, das die geringste Distanz $d_{DVR}(q_T^{t=0}, q_{\tau T(min)})$ besitzt. Die Verwendung von $n_{DV,s(1|\tau T(min))}^{t=0}$ als dem Neuron, in dessen Voronoi-Region $q_{\tau T(min)}$ liegt, anstatt von $n_{DV,s(1|T)}^{t=0}$ ist keine sinnvolle Alternative, da durch das Minimal-Distanz-Kriterium die Gewinner-Neurone in den meisten Fällen übereinstimmen werden, d.h. es liegt $n_{DV,s(1|T)}^{t=0} = n_{DV,s(1|\tau T(min))}^{t=0}$ vor.

Würde demgegenüber der Initialisierungs-Queryvektor $q_\tau^{t=0}$ bei den einzelnen Interaktionen τ mit dem Initialisierungs-Queryvektor $q_T^{t=0}$ der Interaktion T verglichen, so könnte ein Interaktionsobjekt identifiziert werden, dessen bereits korrigierter Queryvektor in der Interaktion T verwendet wird. Der Initialisierungs-Queryvektor $q_\tau^{t=0}$ kann zu diesem Zweck aus Q_τ nochmals durch $A_{IR(Q)}$ gebildet werden, oder $q_\tau^{t=0}$ wird in der Iteration τ gespeichert, sodass eine erweiterte Datenstruktur aller Interaktionsobjekte notwendig wird. Mit $q_\tau^{t=0}$ soll die Initialisierung und mit q_τ soll der nachadaptierte Queryvektor auf der Basis der Kenntnis aller Stimuli aus $M_{DV(m)}$ bezeichnet werden:

$$I^{\leq T-1} = (\text{Interaktion}(\tau) = (Q_\tau, M_{DV(m),\tau}, q_\tau^{t=0}, q_\tau, N_{DV,\tau}) \mid \tau = 1, ..., T\text{-}1). \quad (667)$$

Unabhängig davon ist $q_\tau^{t(end)}$ der Queryvektor in der letzten Iteration bei einem Queryvektor-Feedback, bei dem in jeder Iteration t eine Nachadaption des vorliegenden Queryvektors q_τ^{t-1} auf der Basis der zuletzt ermittelten Stimuli in $M_{DV(m),\tau}^t$ durchgeführt wird.

4.5.3.1) Monorepräsentation der Nutzung nachadaptierter Queryvektoren

Aus $I^{\leq T-1}$ wird das Interaktionsobjekt Interaktion($\tau_{T(min)}$) mit der geringsten Distanz zu $q_T^{t=0}$ bestimmt, sodass sich bei einer Monorepräsentation die ein-elementige, interaktionsspezifische Ergebnisliste ergibt:

$$IL_T^{\leq T-1} = (\text{Interaktion}(\tau_{T(min)})), \text{ mit}$$
$$d_{DVR}(q_T^{t=0}, q_{\tau T(min)}^{t=0}) = \min\{d_{DVR}(q_T^{t=0}, q_\tau^{t=0}), \forall \text{Interaktion}(\tau) \in I^{\leq T-1}\}. \quad (668)$$

In der Interaktion T wird nun der nachadaptierte Queryvektor $q_{\tau T(min)}$ des Gewinner-Interaktionsobjektes Interaktion($\tau_{T(min)}$) anstatt dem Initialisierungs-Queryvektor $q_T^{t=0}$ verwendet, um das Gewinner-Neuron zu spezifizieren, das somit als $n_{DV,s(1|\tau T(min))}$ bezeichnet wird:

$$d_{DVR}(q_{\tau T(min)}, w(x_{DV})_{DV,s(1|\tau T(min))}) =$$
$$\min\{d_{DVR}(q_{\tau T(min)}, w(x_{DV})_{T(min),j}) \mid n_{DV,T(min),j} \in N_{DV,\tau T(min)}\}. \quad (669)$$

Ein Verfahren, das ohne Distanzberechnungen in der Interaktion T auskommt, speichert in den Interaktionen τ das Gewinner-Neuron $n_{DV,s(1|\tau)}$ bezüglich des nachadaptierten Queryvektors q_τ, sodass sich eine Datenstruktur für die Interaktionsobjekte ergibt:

$$I^{\leq T-1} = (\text{Interaktion}(\tau) = (Q_\tau, M_{DV(m),\tau}, q_\tau^{t=0}, q_\tau, n_{DV,s(1|\tau)}, N_{DV,\tau}) \mid \tau = 1, ..., T-1). \quad (670)$$

Durch den Queryvektor $q_{\tau T(min)}$ und das Gewinner-Neuron $n_{DV,s(1|\tau T(min))}$ kann die primäre Dokumentvektoren-Ergebnismenge $DVM_{prim,T}^{t=0}$ der Initialisierungs-Iteration t=0 in T durch die Stimuli in der lokalen Menge $M_{DV(m),s(1|\tau T(min))}$ festgelegt werden. Es werden die Dokumentvektoren betrachtet, die in der Voronoi-Region des Gewinner-Neurons liegen, und für die in der Interaktion $\tau_{T(min)}$ eine Bewertung durch den damaligen Agenten abgegeben wurde. Diese Dokumentvektoren werden nach fallenden Relevanzwerten der korrespondierenden Stimuli sortiert, wodurch sich die sekundäre Dokumentvektoren-Ergebnisliste $DV_{sec,T}^{t=0}$ ergibt, die in der Initialisierungs-Iteration gleich der tertiären Liste gesetzt wird, sodass die dazu korrespondierenden Dokumente in der gleichen Reihenfolge dem Agenten präsentiert werden.

Durch die Bewertung entsteht die Stimulusmenge $M_{DV(m),T}^{t=0}$ und somit das instanzbasierte Modell $AM(rel(x) \mid M_{DV(m),T}^{t=0})$, mit dem der Gewichtsvektor des Gewinner-Neurons $n_{DV,s(1|\tau T(min))}$ und eventuell die Gewichtsvektoren seiner Nachbarn aus $N(d_G=1 \mid G_{DV})_{s(1|\tau T(min))}$ bewertet werden. Die Struktur dieser Neuronen wird somit um eine Relevanzschätzung erweitert, sodass das prototypbasierte Approximationsmodell $AM(rel(x) \mid N_{DV,T}^{t=0})$ entsteht.

Damit könnte die direkte Rolle von Interaktion($\tau_{T(min)}$) in der Interaktion T beendet werden, sodass in t=1 die primäre Ergebnismenge $DVM_{prim,T}^{t=1}$ in der Region um den Gewichtsvektor des Gewinner-Neurons ermittelt wird. Indirekt bleibt der Einfluss jedoch erhalten, da auch weiterhin die Region um $q_{\tau T(min)}$ und nicht um $q_T^{t=0}$ untersucht wird. Konkret bedeutet dies in der Iteration t, dass zunächst alle Neurone ermittelt werden, denen ein Stimulus in T ermittelt wurde, gefolgt von der Vereinigung der lokalen Dokumentvektorenmengen dieser Neurone und ihrer Nachbarn und korrigiert um die Dokumentvektoren, denen der Agent in T eine Bewertung zugeordnet hat. Diese Dokumentvektoren werden durch das

Modell AM(rel(x) | $N_{DV,T}^{t-1}$) bewertet und nach fallenden Relevanzschätzungen sortiert, gefolgt von der Auswahl der ersten δ Dokumentvektoren in der tertiären Liste $DV_{ter,T}^{t}$. Nach der Bewertung der korrespondierenden Dokumente ergibt sich $M_{DV(m),T}^{t}$, die vereinigt mit $M_{DV(m),T}^{\leq t-1}$ die aktuelle Stimulusmenge $M_{DV(m),T}^{\leq t}$ ergibt, mit der das Modell $N_{DV,T}^{t-1}$ zu $N_{DV,T}^{t}$ aktualisiert wird.

Es muss kein neues Abbruchkriterium definiert werden, d.h. ein Abbruch von Seiten des IRS kann erfolgen, wenn alle Dokumentvektoren einer primären Ergebnismenge $DVM_{prim,T}^{t}$ in Verbindung mit dem aktuellen Approximationsmodell AM(rel(x) | $N_{DV,T}^{t-1}$) nur Relevanzschätzungen kleiner einem Schwellenwert besitzen.

Das geschilderte Verfahren kann kombiniert werden mit einem iterativen Queryvektor-Feedback, indem nach t=0 der Queryvektor $q_{\tau T(min)} := q_{\tau T(min)}^{t=0}$ auf der Basis der Stimuli in $M_{DV(m),T}^{t=0}$ zu $q_{\tau T(min)}^{t=1}$ adaptiert wird. Andererseits wird bei einem Verzicht auf ein iteratives Queryvektor-Feedback nach dem Abbruch der Interaktion T in t=t(end) ein nachadaptierter Queryvektor auf der Basis der Gesamtstimulusmenge $M_{DV(m),T}^{\leq t(end)}$ erzeugt, der als Nachadaption des Initialisierungs-Queryvektors $q_{\tau T(min)}$ gebildet werden kann, oder allein aus den Elementen von $M_{DV(m),T}^{\leq t(end)}$ in Verbindung mit einem Adaptionsverfahren oder einer Mittelwertsbildung. Die Gesamtstimulusmenge $M_{DV(m),T}^{\leq t(end)}$ wird als $M_{DV(m),T}$ in die Datenstruktur des Interaktionsobjektes Interaktion(T) eingetragen, der nachadaptierte Queryvektor wird als q_T eingetragen, und der Initialisierungs-Queryvektor $q_{\tau T(min)}$ übernimmt die Rolle von $q_T^{t=0}$. Zuletzt wird noch das Gewinner-Neuron $n_{DV,s(1|T)}$ bestimmt, dessen Gewichtsvektor den geringsten Abstand von dem nachadaptierten Queryvektor q_T besitzt, sodass das Interaktionsobjekt Interaktion(T) in zukünftigen Interaktion benutzt werden kann, um einen Queryvektor q_T und das Gewinner-Neuron zu liefern:

$$\text{Interaktion}(T) = (Q_T, M_{DV(m),T}, q_{\tau T(min)}, q_T, n_{DV,s(1|T)}, N_{DV,T}). \quad (671)$$

4.5.3.2) Polyrepräsentation der Nutzung nachadaptierter Queryvektoren

Bei einer Polyrepräsentation werden die δ_1 Interaktionsobjekte Interaktion(p) = (Q_p, $M_{DV(m),p}$, $q_p^{t=0}$, q_p, $n_{DV,s(1|p)}$, $N_{DV,p}$) ermittelt, deren Initialisierungs-Queryvektor $q_p^{t=0}$ die kleinsten Distanzen zu $q_T^{t=0}$ besitzen, sodass sich die geordneten Listen $IL_T^{\leq T-1}$ und $IL_{T(\delta)}^{\leq T-1}$ ergeben:

$$IL_T^{\leq T-1} = (\text{Interaktion}(p) \in I^{\leq T-1} \mid d_{DVR}(q_T^{t=0}, q_p^{t=0}) < d_{DVR}(q_T^{t=0}, q_{p+1}^{t=0}); p = 1, ..., T-1),$$
$$IL_{T(\delta 1)}^{\leq T-1} = (\text{Interaktion}(p) \in IL_T^{\leq T-1} \mid d_{DVR}(q_T^{t=0}, q_p^{t=0}) < d_{DVR}(q_T^{t=0}, q_{p+1}^{t=0}); p = 1, ..., \delta_1). \quad (672)$$

Anstelle des Queryvektors $q_T^{t=0}$ werden nun die Queryvektoren $q_{\tau p}$ der ausgewählten Interaktionsobjekte Interaktion(p) aus $IL_{T(\delta 1)}^{\leq T-1}$ verwendet, bzw. es werden die zugehörigen Gewinner-Neurone in einer Neuronenmenge $N_{T(\delta 1)}$ zusammengefasst, sodass mehrfaches Auftreten korrigiert wird. Liegen im Extremfall alle Queryvektoren $q_{\tau p}$ in der Voronoi-Region eines Neurons, so besitzt $N_{T(\delta 1)}$ genau ein Element, und die Polyrepräsentation reduziert sich zu einer Monorepräsentation.

Wird von einer Polyrepräsentation ausgegangen, d.h. von $N_{T(\delta 1)}$ mit mehreren Elementen $n_{DV,s(1|p)}$, so wird die Vereinigung der lokalen Stimulusmengen $M_{DV(m),p,s(1|p)}$ dieser Neurone gebildet. Die Doku-

mentvektorenkomponenten dieser Stimuli bilden die primäre Ergebnismenge $DVM_{prim,T}^{t=0}$, die entsprechend den Relevanzwerten fallend geordnet werden, sodass die sekundäre Liste $DV_{sec,T}^{t=0}$ entsteht. Dabei werden die Relevanzwerte unterschiedlicher Interaktionen und somit unterschiedlicher Agenten als gleichwertig behandelt. Aus der Liste $DV_{sec,T}^{t=0}$ werden die ersten δ_2 Elemente ausgewählt, welche bei einer Polyrepräsentation die tertiäre Ergebnisliste $DV_{ter,T}^{t=0}$ bilden, deren korrespondierende Dokumente in der gleichen Reihenfolge präsentiert werden.

Nachdem $M_{DV(m),T}^{t=0}$, $AM(rel(x) \mid M_{DV(m),T}^{t=0})$, $N_{DV,T}^{t=0}$ und $AM(rel(x) \mid N_{DV,T}^{t=0})$ gebildet wurden, wird die Iteration t=1 begonnen, wobei die primäre Ergebnismenge wiederum durch die Neurone und ihre Nachbarn definiert werden kann, denen mindestens ein Stimulus aus $M_{DV(m),T}^{t=0}$ zuordenbar ist. Eine explizite Unterscheidung zwischen Mono- und Polyrepräsentation entfällt bei dieser allgemeinen Definition der primären Ergebnismenge, deren Elemente für t>0 durch das jeweilige Approximationsmodell $AM(rel(x) \mid N_{DV,T}^{t-1})$ bewertet werden, d.h. bei der Nutzung von vergangenen, nachadaptierten Queryvektoren wird in jedem Fall eine Monorepräsentation der Approximationsmodellnutzung unterstellt (siehe auch Abschnitt 4.5.4)). Es ändert sich auch nicht die Struktur des sich ergebenden Interaktionsobjektes Interaktion(T) bei einer Polyrepräsentation, sodass die Darstellungen bei der Monorepräsentation analog gelten.

4.5.4) Nutzung von Relevanz-Approximationsmodellen vergangener Interaktionen

Als letzte Komponente eines Interaktionsobjektes $Interaktion(\tau) = (Q_\tau, M_{DV(m),\tau}, q_\tau, N_{DV,\tau})$ soll die Nutzung der Relevanz-Approximationsmodelle betrachtet werden, die sich aus der individualisierten GNG-SOM $N_{DV,\tau}$ als $AM(rel(x) \mid N_{DV,\tau})$ ableitet. Dieses prototypbasierte Modell steht neben dem instanzbasierten Modell $AM(rel(x) \mid M_{DV(m)})$, das sich aus der Kenntnis der Stimuli, d.h. der bewerteten Dokumentvektoren, ableitet. Der Schwerpunkt soll jedoch auf den prototypbasierten Modellen liegen, wobei $AM(rel(x) \mid M_{DV(m)})$ verwendet wird, um $N_{DV,\tau}$ zu erzeugen, da $N_{DV,\tau}$ aus der nicht individuellen GNG-SOM N_{DV} abgeleitet wird. Es soll der Spezialfall betrachtet werden, dass die Stützpunkte im DVR übernommen werden, d.h. die Gewichtsvektoren der Neurone werden unverändert übernommen, und ein Neuron $n_{DV,k,\tau}$ aus $N_{DV,\tau}$ wird aus $n_{DV,k} \in N_{DV}$ gebildet, indem zusätzlich eine Komponente als ein Stützpunkt im Relevanzraum gebildet wird. Dies geschieht mit dem instanzbasierten Approximationsmodell $AM(rel(x) \mid M_{DV(m)})$, z.B. indem die globale Variante verwendet wird, bei der alle vorliegenden Stimuli in Verbindung mit einem LWR-Verfahren genutzt werden. Es kann auch eine lokale Variante verwendet werden, welche die Stimuli aus $M_{DV(m),k,\tau}$ nutzt, sowie die Stimuli der Nachbarneurone von $n_{DV,k,\tau}$, wobei $M_{DV(m),k,\tau}$ die Teilmenge von $M_{DV(m),\tau}$ ist, deren Dokumentvektor-Komponente Element der Voronoi-Region des Gewichtsvektors $w(x_{DV})_k$ ist. Allgemein wird die Relevanzschätzung an der Stelle des Gewichtsvektors $w(x_{DV})_k$ spezifiziert durch:

$$rel(w(x_{DV})_k)_\tau\text{^} = rel(w(x_{DV})_k \mid AM(rel(x) \mid M_{DV(m),\tau})). \tag{673}$$

Das so erzeugte überwachte GNG-SOM $N_{DV,\tau}$ besitzt die nachfolgende Struktur mit $M_{DV(m),k,\tau}$ als Stimulusteilmenge von $M_{DV(m),\tau}$, deren Dokumentvektoren sich innerhalb der Voronoi-Region des Gewichtsvektors $w(x_{DV})_k$ befinden, und $M_{DV(m),k}$ als die Dokumentvektoren, die in der Voronoi-Region von $w(x_{DV})_k$ liegen:

$$N_{DV,\tau} = \{n_{DV,k} = (w(x_{DV})_k, rel(w(x_{DV})_k)_\tau\text{^}, M_{DV,k}, M_{DV(m),k,\tau}, C_{DV,k}) \mid k = 1, ..., \mu_{N,DV,\tau}\}. \quad (674)$$

Diese Vorgehensweise hat den Vorteil, dass alle GNG-SOMs $N_{DV,\tau}$, $\tau = 1, ..., T\text{-}1$, die gleichen Stützpunkte im DVR sowie die gleiche Verbindungsstruktur besitzen und sich nur durch die Stützpunkte im Relevanzraum unterscheiden.

4.5.4.1) Monorepräsentation der Approximationsmodell-Nutzung

Grundlage der Auswahl eines Interaktionsobjektes aus $I^{\leq T-1}$ bei der Monorepräsentation ist wiederum die kleinste Distanz zwischen $q_T^{t=0}$ und den nachadaptierten Queryvektoren, sodass sich als Ergebnisliste mit genau einem Element $I_T^{\leq T-1} = (\text{Interaktion}(\tau_{T(min)}))$ ergibt, mit $N_{DV,\tau T(min)}$ als dem GNG-SOM, das in dem Relevanz-Approximationsmodell $AM(rel(x) \mid N_{DV,\tau T(min)})$ verwendet wird.

Es soll die lokale Vorgehensweise betrachtet werden, bei der $q_T^{t=0}$ das Gewinner-Neuron $n_{DV,s(1|T)}^{t=0}$ spezifiziert, und bei der die primäre Ergebnismenge $DVM_{prim,T}^{t=0}$ als Dokumentvektorenmenge $M_{DV,s(1|T)}$ festgelegt wird. Ohne die Kenntnis eines Approximationsmodells werden die Dokumentvektoren aus $M_{DV,s(1|T)}$ entsprechend ihrer Distanz zu $q_T^{t=0}$ sortiert, und die korrespondierenden Dokumente werden dem Agenten der Interaktion T zur Bewertung vorgelegt. Mit der Kenntnis von $AM(rel(x) \mid N_{DV,\tau T(min)})$ werden alle Elemente $x_j^{t=0}$ aus $DVM_{prim,T}^{t=0} = M_{DV,s(1|T)}$ bewertet, indem ihnen eine Relevanzschätzung $rel(x_j^{t=0})\text{^} = rel(x_j^{t=0} \mid AM(rel(x) \mid N_{DV,\tau T(min)}))$ zugeordnet wird. Es folgt das Sortieren nach fallenden Relevanzwerten, sodass die geordnete, sekundäre Ergebnisliste $DV_{sec,T}^{t=0}$ erzeugt wird. Soll die Anzahl der Dokumentvektoren aus der Initialisierungs-Iteration als Referenz für die weiteren Iterationen verwendet werden, so bedeutet dies, dass in t=0 die sekundäre mit der tertiären Liste zusammenfällt, und dass die Dokumente, die zu den Dokumentvektoren korrespondieren, die in $DV_{sec,T}^{t=0}$ vorliegen, in der entsprechenden Reihenfolge dem Agenten präsentiert werden.

Durch die Bewertung entsteht die Stimulusmenge $M_{DV(m),T}^{t=0}$ und somit das instanzbasierte Modell $AM(rel(x) \mid M_{DV(m),T}^{t=0})$, mit dem der Gewichtsvektor des Gewinner-Neurons $n_{DV,s(1|T)}^{t=0}$ und eventuell die Gewichtsvektoren seiner Nachbarn aus $N(d_G=1 \mid G_{DV})_{s(1|T)}^{t=0}$ bewertet werden. Die Struktur dieser Neuronen wird somit um eine Relevanzschätzung erweitert, sodass aus dem nicht individuellen Modell N_{DV} die erste Version des überwachten Modells $N_{DV,T}^{t=0}$ entsteht, zu dem das prototypbasierte Approximationsmodell $AM(rel(x) \mid N_{DV,T}^{t=0})$ korrespondiert.

Der weitere Ablauf kann so gestaltet werden, wie der Ablauf ohne Kenntnis der Interaktionsliste $IL^{\leq T-1}$, indem $DVM_{prim,T}^{t=1}$ gebildet wird, deren Dokumentvektoren mit $AM(rel(x) \mid N_{DV,T}^{t=0})$ bewertet werden, d.h. bei diesem Vorgehen wird $IL^{\leq T-1}$ ausschließlich für die Initialisierungs-Iteration genutzt. Im Rahmen einer Monorepräsentation ist dies ein naheliegender Ansatz, da die Alternative in der weiteren Nutzung des konstanten Modells $AM(rel(x) \mid N_{DV,\tau T(min)})$ zusätzlich zu der Nutzung der sich entwickelnden Modelle $AM(rel(x) \mid N_{DV,T}^{t})$, $t = 0, 1, ...$, besteht, was als eine Form der Polyrepräsentation der Approximationsmodelle gesehen werden kann. Dabei wird für ein Element $x_j^{t=1}$ aus $DVM_{prim,T}^{t=1}$ eine Relevanzschätzung durch beide Modelle durchgeführt, die im weiteren aggregiert werden. Im einfachsten Fall wird der ungewichtete Mittelwert beider Schätzungen bestimmt, oder es wird eine Gewichtungsfunktion definiert, die im Verlauf der Iterationen t die Gewichtung verändert. Da angenommen

werden kann, dass $N_{DV,T}^{t}$ zu Beginn der Iterationen weniger Stützpunkte als $N_{DV,\tau T(min)}$ besitzt, könnte die Gewichtung von $AM(rel(x) \mid N_{DV,T}^{t})$ kleiner sein als die von $AM(rel(x) \mid N_{DV,\tau T(min)})$, wobei sich die Gewichtung von $AM(rel(x) \mid N_{DV,T}^{t})$ mit der Iteration t ansteigt. Eine solche Argumentation ist sinnvoll, wenn man annimmt, dass die Bedeutung der Relevanzschätzungen des anderen Agenten gleichwertig zu dem Agenten in der Interaktion T ist. Soll die Bedeutung des Agenten in T hervorgehoben werden, so könnte die Gewichtung in t=0 jeweils gleich 1/2 gewählt werden, und mit steigendem t wird die Gewichtung von $AM(rel(x) \mid N_{DV,T}^{t})$ erhöht.

Eine Alternative zu der Verwendung zweier Modelle ist die Integration beider Modelle zu einem einzigen Modell für eine Iteration t, was bei stützpunktbasierten Approximationsmodellen problemlos funktioniert, indem die Vereinigungsmenge bzw. Vereinigungsliste aller Stützpunkte gebildet wird. Wird die prototypbasierte Vorgehensweise unterstellt, so wird in t=1 die Vereinigung der Neuronenmengen $N_{DV,T}^{t=0}$ und $N_{DV,\tau T(min)}$ gebildet. Die Ergebnismenge bzw. -liste ist kein GNG-SOM mit einer konsistenten Verbindungsstruktur, doch für ein stützpunktbasiertes Approximationsmodell, das alle Stützpunkte verwendet, ist dies irrelevant. Das sich ergebende Modell $AM(rel(x) \mid N_{DV,T}^{t=0} \cup N_{DV,\tau T(min)})$, bewertet alle Elemente $x_j^{t=1}$ aus $DVM_{prim,T}^{t=1}$, gefolgt von der Bildung der sekundären Liste $DV_{sec,T}^{t=1}$ durch Sortierung der Dokumentvektoren nach fallenden Schätzungen. In der gleichen Weise könnte die instanzbasierte Vorgehensweise durchgeführt werden, indem die Mengen $M_{DV(m),T}^{t=0}$ und $M_{DV(m),\tau T(min)}$ vereinigt werden, sodass sich $AM(rel(x) \mid M_{DV(m),T}^{t=0} \cup M_{DV(m),\tau T(min)})$ ergibt.

4.5.4.2) Polyrepräsentation der Approximationsmodell-Nutzung

Im Rahmen der Polyrepräsentation der Modell-Nutzung werden aus $I^{\leq T-1}$ mehrere Interaktionsobjekte spezifiziert, die zumindest in der Initialisierungs-Iteration genutzt werden. Dies ist mit Hilfe der Distanzen $d_{DVR}(q_T^{t=0}, q_\tau)$ einfach zu realisieren, indem die nach τ geordnete Liste $I^{\leq T-1}$ zunächst nach steigenden Distanzen umsortiert wird, was die interaktionsspezifische, geordnete Liste $IL_T^{\leq T-1}$ ergibt. Aus der Liste $IL_T^{\leq T-1}$ können dann die ersten δ_1 Elemente ausgewählt werden, wodurch die Teilliste $IL_{T(\delta 1)}^{\leq T-1}$ erzeugt wird:

$$IL_{T(\delta 1)}^{\leq T-1} = (Interaktion(p) \in IL_T^{\leq T-1} \mid d_{DVR}(q_T^{t=0}, q_p) < d_{DVR}(q_T^{t=0}, q_{p+1}); p = 1, ..., \delta_1). \quad (675)$$

Nachdem eine Teilmenge $IL_{T(\delta 1)}^{\leq T-1}$ von Interaktionsobjekten spezifiziert wurde, werden die korrespondierenden prototypbasierten Approximationsmodelle $AM(rel(x) \mid N_{DV,p})$, $\forall Interaktion(p) \in IL_{T(\delta 1)}^{\leq T-1}$ genutzt, um die Elemente $x_j^{t=0}$ der primären Ergebnismenge $DVM_{prim,T}^{t=0}$ zu bewerten:

$$rel(x_j^{t=0})_p\wedge := rel(x_j^{t=0} \mid AM(rel(x) \mid N_{DV,p})) =$$
$$1/\nu_{p,j} \sum_k h(d_{DVR}(x_j^{t=0}, w(x_{DV})_{p,k}\wedge)) * rel(w(x_{DV})_{p,k}\wedge),$$
$$\nu_{p,j} = \sum_k h(d_{DVR}(x_j^{t=0}, w(x_{DV})_{p,k}\wedge)), \forall n_{DV,p,k}\wedge \in N_{DV,p}. \quad (676)$$

Im nächsten Schritt werden diese Einzelbewertungen aggregiert, indem im einfachsten Fall ein ungewichteter Mittelwert gebildet wird:

$$rel(x_j^{t=0})\wedge := 1/\delta_1 \sum_p rel(x_j^{t=0} \mid AM(rel(x) \mid N_{DV,p})), \forall Interaktion(p) \in IL_{T(\delta 1)}^{\leq T-1}. \quad (677)$$

Eine gewichtete Summe kann gebildet werden, indem jeder Relevanzschätzung ein Gewichtungsfaktor zugeordnet wird, der als Funktion der Distanz $d_{DVR}(q_T^{t=0}, q_p)$ gewählt wird. Eine spezielle Gewichtung ist die Verwendung einer Kernel-Regression, mit den Gewichten $h(d_{DVR}(q_T^{t=0}, q_p))$:

$$rel(x_j^{t=0})\wedge := 1/\nu_j \sum_p h(d_{DVR}(q_T^{t=0}, q_p)) * rel(x_j^{t=0} \mid AM(rel(x) \mid N_{DV,p})),$$
$$\nu_j = \sum_p h(d_{DVR}(q_T^{t=0}, q_p)), \forall Interaktion(p) \in IL_{T(\delta 1)}^{\leq T-1}. \quad (678)$$

Nachdem diese Gesamt-Relevanzschätzung für alle $x_j^{t=0}$ aus $DVM_{prim,T}^{t=0}$ durchgeführt wurde, werden die Elemente nach fallender Gesamtschätzung geordnet, sodass die sekundäre Liste $DV_{sec,T}^{t=0}$ erzeugt wird. In t=0 soll die sekundäre und die tertiäre Liste zusammenfallen, sodass die zu den Elementen aus $DV_{sec,T}^{t=0}$ korrespondierenden Dokumente präsentiert und bewertet werden. Durch die Bewertung entsteht die Stimulusmenge $M_{DV(m),T}^{t=0}$ und somit das instanzbasierte Modell $AM(rel(x) \mid M_{DV(m),T}^{t=0})$, mit welchem dem Gewichtsvektor des Gewinner-Neurons $n_{DV,s(1|T)}^{t=0}$ und seiner Nachbarn aus $N(d_G=1 \mid G_{DV})_{s(1|T)}^{t=0}$ eine Relevanzschätzung zugeordnet wird. Es entsteht somit die GNG-SOM $N_{DV,T}^{t=0}$ und das prototypbasierte Approximationsmodell $AM(rel(x) \mid N_{DV,T}^{t=0})$.

In der nachfolgenden Iteration t=1 kann $N_{DV,T}^{t=0}$ an die Seite der Modelle $N_{DV,p}$ treten, die Teil der ausgewählten Interaktionsobjekte Interaktion(p) aus $IL_{T(\delta 1)}^{\leq T-1}$ sind, da eine Modell-Polyrepräsentation unterstellt wird. D.h. die Elemente $x_j^{t=1}$ aus $DVM_{prim,T}^{t=1}$, die standardmäßig durch die Menge $N_{DV,bew,T}^{t=0}$ der Neurone festgelegt wird, denen mindestens ein Relevanzwert durch den momentanen Agenten zugeordnet wurde, werden durch die δ_1 Approximationsmodelle $AM(rel(x) \mid N_{DV,p})$ sowie durch $AM(rel(x) \mid N_{DV,T}^{t=0})$ bewertet, gefolgt von einer Aggregation der δ_1+1 Einzelschätzungen. Dabei soll $rel(x_j^t \mid AM(rel(x) \mid N_{DV,T}^t))$ mit zunehmender Iteration t an Bedeutung gewinnen, d.h. der Gewichtungsfaktor dieser Einzelschätzung soll mit t wachsen. Bei der oben dargestellten Kernel-Regression ist dies nicht möglich, da das Maximum $h(d_{DVR}(q_T^{t=0}, q_T^{t=0}))$ der Gewichtungsfunktion vorliegt, welches nicht weiter wachsen kann, sodass ab t=1 eine andere Gewichtungsfunktion verwendet werden muss. Festzulegen ist eine Folge, die gegen 1 konvergiert, und die das Gewicht der Relevanzschätzung des momentanen Agenten beschreibt, wobei der Wert 1 minus einem Wert der Folge das Gewicht ist, das sich die δ Einzelschätzungen teilen müssen, was im einfachsten Fall zu gleichen Teilen geschieht.

Die aggregierten Gesamtrelevanzschätzungen werden verwendet, um die Elemente aus $DVM_{prim,T}^{t=1}$ nach fallenden Schätzungen zu sortieren, wodurch die sekundäre Ergebnisliste $DV_{sec,T}^{t=1}$ entsteht. Da ab t>0 die Anzahl $\#DVM_{prim,T}^{t=0} := \delta_2$ als Referenz verwendet werden soll, und da in der Regel in $DVM_{prim,T}^{t=1}$ mehr Elemente vorliegen, wenn die Vereinigung aller Elemente der lokalen Dokumentvektorenmengen der Neurone aus $N_{DV,bew,T}^{t=0}$ gebildet werden, so werden aus $DV_{sec,T}^{t=1}$ die ersten δ_2 Dokumentvektoren in die Liste $DV_{ter,T}^{t=1}$ übernommen. Die dazu korrespondierenden Dokumente werden dem Agenten präsentiert, sodass nach der Bewertung die Stimulusmenge $M_{DV(m),T}^{t=1}$ vorliegt, die vereinigt mit $M_{DV(m),T}^{t=0}$ die Menge $M_{DV(m),T}^{t\leq 1}$ bildet. Mit dem instanzbasierten Modell $AM(rel(x) \mid M_{DV(m),T}^{t\leq 1})$ werden die Gewichtsvektoren der Neurone bewertet, denen bislang mindestens ein Stimulus zugeordnet wurde, sodass sich daraus das GNG-SOM $N_{DV,T}^{t=1}$ und das prototypbasierte Approximationsmodell $AM(rel(x) \mid N_{DV,T}^{t=1})$ ergibt. Damit wird die Iteration t=1 beendet und t=2 wird mit der Erzeugung von $DVM_{prim,T}^{t=2}$ begonnen.

4.5.5) Nutzung von Suchregionen vergangener Interaktionen

Neben der Nutzung von Komponenten der Interaktionsobjekte Interaktion(τ) = (Q_τ, $q_\tau^{t=0}$, q_τ, $n_{DV,s(1|\tau)}$, $M_{DV(m),\tau}$, $N_{DV,\tau}$) existieren weitere Möglichkeiten der Nutzung vorangegangener Interaktionen, indem eine Analyse der Komponenten durchgeführt wird. Z.B. kann aus den Tripeln (Q_τ, $q_\tau^{t=0}$, q_τ), τ = 1, ..., T-1, ein Indexierungsfunktion-Feedbackverfahren durchgeführt werden (siehe Abschnitt 3.9.6)), sodass die neue Query-Indexierungsfunktion $A_{IR(Q)}^{T-1}$ beim Input der Queries Q_τ, τ = 1, ..., T-1, jeweils den Queryvektor q_τ anstatt den Queryvektor $q_\tau^{t=0}$ liefert. Diese neue Indexierungsfunktion $A_{IR(Q)}^{T-1}$ würde in der nächsten Interaktion, d.h. in T, eingesetzt, was als Substitution oder Ergänzung der oben beschriebenen Verfahren zur Nutzung von Queryvektoren vergangener Interaktionen verwendet werden kann.

Mit der Kenntnis der Interaktionsobjekte Interaktion(τ) kann auch eine deskriptive Statistik über Ausdehnung der Region durchgeführt werden, in der ein Suchprozess während einer Interaktion erfolgte. Der Zweck der Kenntnis einer solchen Region besteht darin, die Dokumentvektoren aus dieser Region zu Beginn der Interaktion zu Kandidaten zu machen, und sie von einem Relevanz-Approximationsmodell bewerten zu lassen.

Ist keine globale Dokumentvektorenklassifikation vorhanden, so kann die Distanz zwischen dem Initialisierungs-Queryvektor $q_\tau^{t=0}$ und dem Queryvektor $q_\tau^{t=t(end)}$ bei einem Queryvektor-Relevanzfeedback bzw. dem nachadaptierten Queryvektor q_τ als Grundlage der Beschreibung der Suchregion verwendet werden. Z.B. kann angenommen werden, dass $q_\tau^{t=0}$ als Ausgangspunkt der Suche den Mittelpunkt einer Mannigfaltigkeit wie z.B. einer Hyperkugel im DVR bildet, und dass q_τ als Endpunkt der Suche auf der Oberfläche dieser Hyperkugel liegt, sodass $d_{DVR}(q_\tau^{t=0}, q_\tau)$ als Radius dieser Hyperkugel interpretiert werden kann.

Alternativ könnte der Mittelpunkt $1/2(q_\tau^{t=0} + q_\tau)$ als Mittelpunkt der Suchregion betrachtet werden, sodass der Radius auf $1/2*d_{DVR}(q_\tau^{t=0}, q_\tau)$ sinkt, was günstiger ist, da bei großen Regionen auch mehr Dokumentvektoren nachgewiesen werden, die bewertet werden müssen.

Am besten lässt sich die Suchregion jedoch durch die Dokumentvektoren beschreiben, die in den Ergebnismengen bzw. Ergebnislisten der einzelnen Interaktionen einer Interaktion τ liegen. Auf ihre Eignung geprüft wurden alle Dokumentvektoren aus der Vereinigung aller primärer Ergebnismengen, d.h. aus

$$DVM_{prim,\tau}^{\leq t(end,\tau)} = \bigcup_{t=1->t(end,\tau)} DVM_{prim,\tau}^{t}. \quad (679)$$

Alternativ kann die Menge aller Dokumentvektoren aus der Vereinigung aller tertiären Ergebnislisten betrachtet werden, was den Vorteil hat, dass alle diese Dokumentvektoren in den Stimuli aus $M_{DV(m),\tau}$ enthalten sind, und somit direkt zugänglich sind:

$$DV_{ter,\tau}^{\leq t(end,\tau)} = \bigcup_{t=1->t(end,\tau)} DV_{ter,\tau}^{t}. \quad (680)$$

Wird die Gesamtstimulusmenge $M_{DV(m),\tau}$ zur Beschreibung der Suchregion verwendet, so kann der arithmetische Mittelwertsvektor $\bar{x}_\tau$ der Dokumentvektorenkomponenten aller Stimuli berechnet werden, sowie der Stimulus $m_{\tau,max}$ mit der größten Distanz zu diesem Mittelwertsvektor, wobei diese Distanz als Radius einer Hyperkugel verwendet wird, welche die Suchregion beschreibt:

$$\bar{x}_\tau = 1/\#M_{DV(m),\tau} * \sum_j x_{\tau,j}, \forall m_{\tau,j} \in M_{DV(m),\tau},$$
$$d_{DVR}(\bar{x}_\tau, x_{\tau,max}) = \max\{d_{DVR}(\bar{x}_\tau, x_{\tau,j}) \mid m_{\tau,j} \in M_{DV(m),\tau}\}. \quad (681)$$

Es lassen sich auch andere Formen der Mannigfaltigkeit anstatt einer Hyperkugel verwenden, wie z.B. Hyperellipsoide, worauf jedoch nicht weiter eingegangen werden soll, da im weiteren von einer globalen Dokumentvektorenklassifikation durch eine GNG-SOM N_{DV} ausgegangen werden soll.

Wird eine solche GNG-SOM N_{DV} verwendet, so ergeben sich effektivere Strategien, um Suchräume zu spezifizieren, indem die Voronoi-Regionen verwendet werden. Wird wiederum $M_{DV(m),\tau}$ als Grundlage zur Beschreibung verwendet, so können die Neurone spezifiziert werden, denen mindestens ein Stimulus zugeordnet werden kann, d.h. die in ihren Voronoi-Regionen mindestens ein Dokumentvektor besitzen, der durch den Agenten in der Interaktion τ bewertet wurde. Innerhalb einer Iteration t wurde diese Neuronenmenge bezüglich der bis dahin bekannten Gesamtstimulusmenge $M_{DV(m)}^{\leq t}$ mit $N_{DV,bew}^{\ t}$ bezeichnet, sodass die Gesamtneuronenmenge bezüglich $M_{DV(m),\tau}$ mit $N_{DV,bew,\tau} := N_{DV,bew,\tau}^{t(end,\tau)}$ bezeichnet werden soll. Wird geplant, dass die Suchräume verwendet werden, so ist es sinnvoll, diese Neuronenmenge, die nach $t = t(end,\tau)$ direkt vorliegt, in die Datenstruktur der Interaktionsobjekte aufzunehmen:

$$\text{Interaktion}(\tau) = (Q_\tau, q_\tau^{t=0}, q_\tau, n_{DV,s(1|\tau)}, M_{DV(m),\tau}, N_{DV,\tau}, N_{DV,bew,\tau}). \quad (682)$$

Der Suchraum ergibt sich in diesem Zusammenhang als Vereinigung der Voronoi-Regionen $\tilde{R}_{DV,\tau,k}$ der Neurone $n_{DV,\tau,k}$ aus $N_{DV,bew,\tau}$:

$$SR_\tau = \bigcup_k \tilde{R}_{DV,\tau,k}, \forall n_{DV,\tau,k} \in N_{DV,bew,\tau}. \quad (683)$$

Notwendig ist jedoch eine Form von Distanzbeschreibung, wenn für $q_T^{t=0}$ eine Region innerhalb von N_{DV} als zu erwartende Suchregion spezifiziert werden soll. Statt der metrischen Distanz $d_{DVR}(.,.)$ kann innerhalb N_{DV} die Graphendistanz $d_{G(DV)}$ verwendet werden, als dem kürzesten Weg zwischen zwei Gewichtsvektoren innerhalb des Graphen G_{DV}, der durch die Verbindungsmatrix der Neurone bzw. Gewichtsvektoren gegeben ist. Für eine Interaktion τ ist der Beginn eines Suchpfades mit dem Gewinner-Neuron $n_{DV,s(1|\tau)}$ bezüglich des Initialisierungs-Queryvektors $q_\tau^{t=0}$ bereits gegeben. Es muss das Neuron $n_{DV,\tau,max}$ gesucht werden, das Element von $N_{DV,bew,\tau}$ ist, und gleichzeitig den größten Graphenabstand von $n_{DV,s(1|\tau)}$ besitzt:

$$d_{G(DV),\tau,max} = d_{G(DV)}(n_{DV,s(1|\tau)}, n_{DV,\tau,max}) =$$
$$\max\{d_{G(DV)}(n_{DV,s(1|\tau)}, n_{DV,\tau,k}) \mid n_{DV,\tau,k} \in N_{DV,bew,\tau}\}. \quad (684)$$

Die Datenstruktur der Interaktionsobjekte kann entsprechend erweitert werden:

$$\text{Interaktion}(\tau) = (Q_\tau, q_\tau^{t=0}, q_\tau, n_{DV,s(1|\tau)}, M_{DV(m),\tau}, N_{DV,\tau}, N_{DV,bew,\tau}, d_{G(DV),\tau,max}). \quad (685)$$

Die deskriptive Statistik über alle Interaktionsobjekte Interaktion(τ) aus $I^{\leq T-1}$ bezieht sich auf diese maximale Graphendistanz, indem ausgewählte Momente wie Mittelwert und Varianz bestimmt werden:

$$d_{G(DV),max}^{\leq T-1} = 1/(T-1) * \sum_\tau d_{G(DV),\tau,max},$$
$$var_{d(G(DV),max)}^{\leq T-1} = 1/(T-1) * \sum_\tau (d_{G(DV),max}^{\leq T-1} - d_{G(DV),\tau,max})^2, \forall \text{Interaktion}(\tau) \in I^{\leq T-1}. \quad (686)$$

Diese beiden reellen Beschreibungsparameter werden auf natürliche Zahlen abgebildet, da Graphendistanzen auf der Menge N_0 definiert sind. Z.B. kann in beiden Fällen abgerundet werden, wobei die beiden Bezeichnungen $d_{G(DV),max}^{\leq T-1}$ und $var_{d(G(DV),max)}^{\leq T-1}$ beibehalten werden sollen. Die sich ergebenden natürlichen Zahlen, die für die Interaktion T gültig sind, werden in die Datenstruktur der Interaktionsliste $I^{\leq T-1}$ aufgenommen:

$$I^{\leq T-1} = (\text{Interaktion}(\tau), d_{G(DV),max}^{\leq T-1}, var_{d(G(DV),max)}^{\leq T-1} \mid \tau = 1, ..., T-1). \quad (687)$$

In der nachfolgenden Interaktion T wird nach der Ermittlung des Initialisierungs-Queryvektors $q_T^{t=0}$ und des Gewinner-Neurons $n_{DV,s(1|T)}^{t=0}$ eine Menge von Neuronen um $n_{DV,s(1|T)}^{t=0}$ spezifiziert, die als Funktion der Graphendistanz-Parameter festgelegt werden.

Im einfachsten Fall kann der Mittelwertsparameter $d_{G(DV),max}^{\leq T-1}$ allein verwendet werden, indem alle Neurone aus N_{DV} ermittelt werden, die von $n_{DV,s(1|T)}^{t=0}$ eine Graphendistanz kleiner-gleich $d_{G(DV),max}^{\leq T-1}$ besitzen, was durch die Nachbarschaftsmenge $N(d_G \leq d_{G(DV),max}^{\leq T-1} \mid G_{DV})_{DV,s(1|T)}$ beschrieben wird. Die Dokumentvektoren-Ergebnismenge wird mit Hilfe dieser Neuronenmenge festgelegt, indem die Vereinigungsmenge aller lokalen Dokumentvektorenmengen $M_{DV,k}$ gebildet wird:

$$DVM_{prim,T}^{t=0} = \bigcup_k M_{DV,k}, \forall n_{DV,k} \in N(d_G \leq d_{G(DV),max}^{\leq T-1} \mid G_{DV})_{DV,s(1|T)}. \quad (688)$$

Sinnvoll ist diese Vorgehensweise in t=0, wenn ein Relevanz-Approximationsmodell $AM(rel(x) \mid N_{DV,\tau T(min)})$ von einem ausgewählten Interaktionsobjekt wie $Interaktion(\tau_{T(min)})$ verwendet wird, um die Dokumentvektoren aus $DVM_{prim,T}^{t=0}$ zu bewerten, und sie entsprechend fallenden Relevanzschätzungen zu ordnen, wodurch die sekundäre Ergebnisliste $DV_{sec,T}^{t=0}$ entsteht. Würde in der Initialisierungs-Iteration auf Relevanzschätzungen verzichtet, so kann $DVM_{prim,T}^{t=0}$ als lokale Dokumentvektorenmenge $M_{DV,s(1|T)}$ des Gewinner-Neurons verwendet werden. Für die Iteration t=1 werden dann die Vereinigungen der lokalen Dokumentvektorenmengen der Neurone aus der Nachbarschaft $N(d_G \leq d_{G(DV),max}^{\leq T-1} \mid G_{DV})_{DV,s(1|T)}$ ohne das Gewinner-Neuron gebildet, was durch die Nachbarschaftsmenge $N(1 \leq d_G \leq d_{G(DV),max}^{\leq T-1} \mid G_{DV})_{DV,s(1|T)}$ beschrieben werden kann:

$$DVM_{prim,T}^{t=0} = M_{DV,s(1|T)},$$

$$DVM_{prim,T}^{t=1} = \bigcup_k M_{DV,k}, \forall n_{DV,k} \in N(1 \leq d_G \leq d_{G(DV),max}^{\leq T-1} \mid G_{DV})_{DV,s(1|T)}. \quad (689)$$

Wird in t=0 ein Relevanz-Approximationsmodell angewendet, was sinnvoll ist anzunehmen, da die Graphendistanz-Parameter aus $I^{\leq T-1}$ ermittelt wurden, sodass auch andere Komponenten der Liste $I^{\leq T-1}$ verwendet werden können, so muss ein Parameter spezifiziert werden, der die Anzahl der Elemente in der tertiären Liste $DV_{ter,T}^{t=0}$ festlegt. Dieser Parameter muss nicht extern festgelegt werden, da die Anzahl wie üblich durch die Anzahl der Elemente in der lokalen Dokumentvektorenmenge des Gewinner-Neurons festgelegt werden kann. D.h. aus der geordneten Liste $DV_{sec,T}^{t=0}$ werden die ersten $\#M_{DV,s(1|T)}$ Dokumentvektoren ausgewählt und in $DV_{ter,T}^{t=0}$ übernommen, gefolgt von der Präsentation der korrespondierenden Dokumente. Die Ergebnismenge der nächsten Iteration t=1 kann rekursiv festgelegt werden durch die Ergebnismenge der vorangegangenen Iteration ohne die Elemente der tertiären Liste der vorangegangenen Iteration:

$$DVM_{prim,T}^{t=1} = DVM_{prim,T}^{t=0} \setminus DV_{ter,T}^{t=0}, \text{ oder allgemein}$$
$$DVM_{prim,T}^{t} = DVM_{prim,T}^{t-1} \setminus DV_{ter,T}^{t-1}. \quad (690)$$

Die sich ergebende Datenstruktur $DVM_{prim,T}^{t=1}$ ist eine Menge, deren Elemente nach aktualisierten Relevanzschätzungen geordnet werden müssen. Zunächst wird in t=0 nach der Bewertung durch den Agenten die Stimulusmenge $M_{DV(m),T}^{t=0}$ erzeugt, welche das instanzbasierte Approximationsmodell $AM(rel(x) \mid M_{DV(m),T}^{t=0})$ direkt und das prototypbasierte Approximationsmodell $AM(rel(x) \mid N_{DV,T}^{t=0})$ indirekt über die Relevanzschätzung ausgewählter Neurone aus $N_{DV,T}^{t=0}$ festlegt. Mit $AM(rel(x) \mid N_{DV,T}^{t=0})$ werden die Elemente aus $DVM_{prim,T}^{t=1}$ bewertet, und nach fallenden Relevanzschätzungen sortiert, sodass $DV_{sec,T}^{t=1}$ erzeugt wird. Aus dieser Liste werden wiederum die ersten $\#M_{DV,s(1|T)}$ Elemente ohne Zurücklegen entnommen, welche die Liste $DV_{ter,T}^{t=1}$ bilden, während die Restmenge als $DVM_{prim,T}^{t=2}$ in der nachfolgenden Iteration verwendet wird. Alternativ kann in den weiteren Iterationen das oder die Approximationsmodelle weiter verwendet werden, die in t=0 eingesetzt wurden, worauf jedoch nicht weiter eingegangen werden soll.

Durch die Bildung der Restmenge in einer Iteration liegt ein Abbruchkriterium von Seiten des IRS nahe. Liegen zum ersten Mal weniger als $\#M_{DV,s(1|T)}$ Elemente in einer Restmenge vor, so wird die gesamte Restmenge bei der nächsten Gelegenheit präsentiert, gefolgt von einem Abbruchsversuch von Seiten des IRS. Sollen demgegenüber weitere Dokumente nachgewiesen werden, d.h. wird der Abbruchversuch von Seiten des IRS durch den Agenten überschrieben, so kann wie üblich die Neuronenmenge $N_{DV,bew,T}^{t-1}$ verwendet werden, um weitere Dokumentvektoren als Kandidaten zu ermitteln. D.h. es wird die Vereinigungsmenge aller lokalen Dokumentvektorenmengen von Neuronen und ihren Nachbarn durchgeführt, denen mindestens ein Stimulus aus $M_{DV(m),T}^{\leq t-1}$ zuordenbar ist, korrigiert um die Dokumentvektoren, die zu den Stimuli aus $M_{DV(m),T}^{\leq t-1}$ korrespondieren. Im Kontext des Abbruchversuches durch das IRS bedeutet dies, dass die Dokumentvektoren-Vereinigungsmenge ohne $DVM_{prim,T}^{t=0}$ gebildet wird, die als Ergebnisliste $DVM_{prim,T}^{t}$ verwendet wird.

4.6) Korrektur der Relevanzschätzungen um Fehlerschätzungen

Alle bislang dargestellten Verfahren basieren ausschließlich auf der Verwendung von Relevanzschätzungen durch ein Relevanz-Approximationsmodell, um Kandidaten-Dokumentvektoren der primären Ergebnismenge DVM_{prim}^{t} zu bewerten und sie somit in eine Rangfolge nach fallenden Relevanzschätzungen bringen zu können, was dem Ranking der sekundären Ergebnisliste DV_{sec}^{t} entspricht. Unberücksichtigt wurde bislang die Tatsache, dass alle Relevanz-Approximationsmodelle ihre Schätzungen mit Fehlerwerten ausgeben, wobei zwischen dem systematischen Fehler (Bias) und der Output-Varianz unterschieden werden kann. Die Summe aus quadrierter Bias und Output-Varianz ergibt den mittleren quadratischen Fehler MSE (siehe Abschnitt 2.3.3); Geman et al. (1992[142]), Cohn (1995[73]), Cohn et al. (1995[74])). Wird angenommen, dass an jeder Stelle des Inputraumes DVR, die gleiche Fehlerstärke und die gleiche Fehlerrichtung, d.h. Über- bzw. Unterschätzung des richtigen Wertes, erzeugt wird, so spielt der Fehler für ein Ranking keine Rolle. Wird bei jeder Relevanzschätzung z.B. das gleiche, unbekannte Maß an Überschätzung durchgeführt, so bleibt die Rangfolge der Dokumentvektoren unverändert.

Die Situation ändert sich jedoch, wenn ein Approximationsmodell in unterschiedlichen Regionen des Inputraumes unterschiedliche Qualitätseigenschaften besitzt, was der Regelfall ist. Dies gilt bei stützpunktbasierten Modellen im Besonderen, da die Unsicherheit einer Schätzung mit zunehmender Distanz von Stützpunkten zunimmt (siehe die Darstellungen über Anwendungsregionen von Approximationsmodellen in Bachelier (1999d[22])). Es erscheint daher sinnvoll, die Relevanzschätzungen $rel(x_j^t)^\wedge$ eines Modells $AM(rel(x))^{t-1}$ um eine Fehlerschätzung zu korrigieren, die von der Lage des Dokumentvektors abhängig ist, um damit die Rangfolge der Kandidaten aus DVM_{prim}^t aufzubauen. Zu beachten ist, dass der Fehler des Relevanz-Approximationsmodells nur dann bestimmt werden kann, wenn der richtige Relevanzwert an der Stelle x_j^t bekannt wäre, was jedoch nicht der Fall ist, sodass ausschließlich Fehlerschätzungen betrachtet werden können. Dabei soll der gleiche Typ von Approximation verwendet werden wie bei der zugrunde liegenden Relevanzschätzung, d.h. es werden stützpunktbasierte LWR-Fehler-Approximationsmodelle erzeugt, wie dies in allgemeiner Form im Abschnitt über Output-, Bias- und Varianz-Approximationsmodelle beschrieben wurde (siehe Abschnitt 2.3.5)).

Im weiteren soll der Fall eines instanzbasierten bzw. eines prototypbasierten Modelltyps beschrieben werden, wobei der gleiche Typ für die Relevanz- und die Fehlerschätzung verwendet werden soll. In jedem Fall ist zum Zeitpunkt t die Gesamt-Stimulusmenge $M_{DV(m)}^{\leq t-1}$ Grundlage für alle weiteren Darstellungen.

4.6.1) Instanzbasierte Fehlermodelle

4.6.1.1) Monorepräsentation von instanzbasierten Fehlermodellen

Grundlage aller Verfahrensvarianten ist die Zuordnung einer Relevanzschätzung $rel(x_j)^\wedge$ zu jedem der Stimuli m_j aus $M_{DV(m)}^{\leq t-1}$ neben dem richtigen Relevanzwert $rel(x_j)$. Bei der Verwendung instanzbasierter Modelle und bei einer Modell-Monorepräsentation kann hierfür ein Element aus der Klasse der Gewinnerlisten-Verfahren verwendet werden (siehe Abschnitt 2.3.8)), da ein globales, instanzbasiertes LWR-Approximationsverfahren durch $d_{DVR}(x_j, x_j) = 0$ in Verbindung mit einer exponentiellen Kernelfunktion $h(d_{DVR}(.,.))$ völlig vom Gewinner-Stimulus dominieren würde, was eine viel zu geringe Bias an diesem Punkt impliziert. Vereinfachend soll das Leave-First-Out-Verfahren betrachtet werden, bei dem der erste Gewinner-Stimulus bei einer Präsentation von x_j, d.h. m_j selbst, nicht an der LWR-Approximation teilnehmen darf, sodass das instanzbasierte Modell $AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \setminus \{m_j\})$ die Relevanz-Approximation übernimmt:

$$rel(x_j)^\wedge = rel(x_j \mid AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \setminus \{m_j\})) =$$
$$1/\nu_j * \sum_k h(d_{DVR}(x_j, x_k)) * rel(x_k),$$
$$\nu_j = \Sigma_k h(d_{DVR}(x_j, x_k)), \forall m_k \in M_{DV(m)}^{\leq t-1} \setminus \{m_j\}. \quad (691)$$

Die Datenstruktur des Stimulus ergibt sich bei einer Approximationsmodell-Monorepräsentation zu:

$$m_j = (x_j, rel(x_j), rel(x_j)^\wedge, bias(x_j)), \text{ mit}$$
$$bias(x_j) = rel(x_j) - rel(x_j)^\wedge. \quad (692)$$

Zusätzlich kann ein Output-Varianz-, d.h. ein Relevanz-Varianz-Approximationsmodell, aufgebaut werden, sodass an der Stelle jedes Stimulus aus $M_{DV(m)}^{\leq t-1}$ eine Relevanz-Varianz $\sigma(rel(x_j))^2$ bestimmt werden muss (Cohn et al. (1995[74])):

$$\sigma(rel(x_j))^2 = 1/\nu_j \sum_k h(d_{DVR}(x_j, x_k)) * (rel(x_k) - rel(x_j)\hat{})^2,$$
$$\nu_j = \sum_k h(d_{DVR}(x_j, x_k)), \forall m_k \in M_{DV(m)}^{\leq t-1} \setminus \{m_j\}. \quad (693)$$

Die Datenstruktur von m_j erweitert sich zu:

$$m_j = (x_j, rel(x_j), rel(x_j)\hat{}, bias(x_j), \sigma(rel(x_j))^2). \quad (694)$$

Mit dieser Datenstruktur und einem LWR-Verfahren liegt ein instanzbasiertes Relevanz-Approximationsmodell $AM(rel(x) \mid M_{DV(m)}^{\leq t-1})$, ein Relevanz-Bias-Approximationsmodell $AM(bias(x) \mid M_{DV(m)}^{\leq t-1})$ und ein Relevanz-Varianz-Approximationsmodell $AM(\sigma(rel(x))^2 \mid M_{DV(m)}^{\leq t-1})$ vor.

Angemerkt werden muss, dass die Bias im Gegensatz zu der Varianz eine Richtung besitzt, da keine quadrierte Bias verwendet wird. D.h. die Bias kann positiv sein, wenn die Relevanzschätzung kleiner ist als der richtige Relevanzwert, was einer Unterschätzung der Relevanz an dieser Stelle bedeutet, oder die Bias kann negativ sein, wenn die Schätzung größer ist, was einer Überschätzung der Relevanz an dieser Stelle bedeutet.

Die oben genannten Modelle können ab der Iteration t=1 gebildet werden, nachdem in t=0 die Dokumentvektorenmenge $M_{DV}^{t=0}$ festgelegt, und durch den Agenten bewertet wurde, sodass die Stimulusmenge $M_{DV(m)}^{t=0}$ vorliegt. D.h. nach der Bewertung der Dokumentvektoren durch den Agenten in einer Iteration t-1 werden die drei Modelle spezifiziert, gefolgt von dem Übergang in die nächste Iteration, in der zunächst die primäre Ergebnismenge DVM_{prim}^t ermittelt wird. Für jeden Dokumentvektor x_j^t aus DVM_{prim}^t wird mit Hilfe der drei Modelle eine Relevanz-, eine Bias- und eine Varianz-Schätzung gebildet, wobei die Gewichte h(.) und somit der Normierungsfaktor ν_j^t nur einmal bestimmt werden müssen und noch zweimal wieder verwendet werden können:

$$rel(x_j^t)\hat{} = 1/\nu_j^t * \sum_k h(d_{DVR}(x_j^t, x_k)) * rel(x_k),$$
$$bias(x_j^t)\hat{} = 1/\nu_j^t * \sum_k h(d_{DVR}(x_j^t, x_k)) * bias(x_k),$$
$$\sigma(rel(x_j^t))^2\hat{} = 1/\nu_j^t * \sum_k h(d_{DVR}(x_j^t, x_k)) * \sigma(rel(x_k))^2,$$
$$\nu_j^t = \sum_k h(d_{DVR}(x_j^t, x_k)), \forall m_k \in M_{DV(m)}^{\leq t-1}. \quad (695)$$

Diese Schätzungen werden in die noch unvollständige Datenstruktur des Stimulus m_j^t eingetragen:

$$m_j^t = (x_j^t, \_, rel(x_j^t)\hat{}, bias(x_j^t)\hat{}, \sigma(rel(x_j^t))^2\hat{}), \forall m_j \in DVM_{prim}^t. \quad (696)$$

Im nächsten Schritt muss entschieden werden, mit welchem Ordnungskriterium die sekundäre Ergebnisliste DV_{sec}^t aus der Menge DVM_{prim}^t mit Hilfe der Schätzungen $rel(x_j^t)\hat{}$, $bias(x_j^t)\hat{}$ und $\sigma(rel(x_j^t))^2\hat{}$ gebildet werden soll.

Eine Möglichkeit korrigiert zunächst die Relevanzschätzung $rel(x_j)^\wedge$ um die geschätzte Bias $bias(x_j^t)^\wedge$, indem die Summe aus beiden Komponenten gebildet wird:

$$rel(x_j)^\wedge' = rel(x_j)^\wedge + bias(x_j). \quad (697)$$

Die korrigierte Relevanzschätzung $rel(x_j)^\wedge'$ wird als Mittelpunkt eines Intervalls betrachtet, dessen linke und rechte Grenze durch den Betrag der Relevanz-Standardabweichung und ein Skalierungsfaktor α gegeben ist. Im einfachsten Fall wird $\alpha := 1$ gesetzt, sodass die unvollständige Datenstruktur von m_j^t durch den Dokumentvektor und das sich ergebende Relevanz-Intervall umformuliert werden kann:

$$m_j^t = (x_j^t, \_, [rel(x_j)' - |\sigma(rel(x_j^t))^\wedge|, rel(x_j)' + |\sigma(rel(x_j^t))^\wedge|]). \quad (698)$$

Um eine Rangfolge der Dokumentvektoren aus DVM_{prim}^t zu erzeugen, muss ein Verfahren spezifiziert werden, das die Dominanz zwischen Intervallen festlegt (siehe Abschnitt 2.6)), Bachelier (1998b:133ff[16])). Ist es nicht unbedingt erforderlich, dass eine feste Anzahl von Elementen aus DV_{sec}^t in DV_{ter}^t übernommen wird, so kann eine einfache Heuristik verwendet werden, die zunächst alle Intervalle auf einer Relevanzskala anordnet und dann die ersten $\#M_{DV(m)}^{t=0}$ Intervalle auswählt. Sollten Intervalle existieren, die eine Überlappung mit den zuerst ausgewählten Intervallen besitzen, so werden diese ebenfalls ausgewählt, und die korrespondierenden Dokumentvektoren werden in die tertiäre Ergebnisliste aufgenommen. Auf diese Weise wird auf komplexere Verfahren verzichtet, mit denen die Dominanz zwischen Intervallen bestimmt wird, wobei diese Vorgehensweise die Eigenschaft besitzt, dass $\#M_{DV(m)}^{t=0}$ die Untergrenze der übernommenen Dokumentvektoren ist.

4.6.1.2) Polyrepräsentation von instanzbasierten Fehlermodellen

Grundlage der Modell-Polyrepräsentation soll eine künstliche Stimulus-Polyrepräsentation sein, d.h. aus der Gesamt-Stimulusmenge $M_{DV(m)}^{\leq t-1}$ werden durch Ziehen mit Zurücklegen δ Bootstraplisten $M_{DV(m),p}^{\leq t-1}$ erzeugt, die jeweils in Verbindung mit einem stützpunktbasierten Approximationsverfahren ein instanzbasiertes Relevanz-Approximationsmodell $AM(rel(x) \mid M_{DV(m),p}^{\leq t-1})$ festlegen. Mit den δ+1 Modellen wird jeweils eine Relevanzschätzung $rel(x_j^t \mid AM(rel(x) \mid M_{DV(m)}^{\leq t-1}))$ und $rel(x_j^t \mid AM(rel(x) \mid M_{DV(m),p}^{\leq t-1}))$, $p = 1, ..., \delta$, an der Stelle eines Kandidaten-Dokumentvektors aus DVM_{prim}^t vorgenommen. Die δ+1 Einzelschätzungen können zu einer Gesamtschätzung $\overline{rel}(x_j^t)^\wedge$ durch Bildung des arithmetischen Mittelwertes aggregiert werden, wobei die Gesamtschätzung als effektivere Relevanzschätzung betrachtet wird, welche die Rolle des richtigen Relevanzwertes in der Bias-Schätzung einnimmt. Die Bias-Schätzung an der Stelle x_j^t ergibt sich, wenn die δ Bootstrap-Einzelschätzungen zu einer Gesamtschätzung $rel(x_j^t)_{boot}^\wedge$ aggregiert werden, die mit der Schätzung $rel(x_j^t \mid AM(rel(x) \mid M_{DV(m)}^{\leq t-1}))$ des Gesamtmodells verglichen wird, wobei die 0,632-Heuristik angewendet werden kann:

$$bias(x_j^t)_{0,632}^\wedge = [rel(x_j^t \mid AM(rel(x) \mid M_{DV(m)}^{\leq t-1})) - rel(x_j^t)_{boot}^\wedge]/0,632. \quad (699)$$

Ist eine Menge von Einzelschätzungen vorhanden, so kann die Relevanz-Varianz direkt berechnet werden, indem $\overline{rel}(x_j^t)^\wedge$ oder $rel(x_j^t)_{boot}^\wedge$ als Mittelwert verwendet wird:

$$\sigma(rel(x_j^t))^{2\wedge} = 1/(\delta+1) * [(rel(x_j^t \mid AM(rel(x) \mid M_{DV(m)}^{\leq t-1})) - \overline{rel}(x_j^t)\wedge)^2$$
$$+ \sum_p (rel(x_j^t \mid AM(rel(x) \mid M_{DV(m),p}^{\leq t-1}) - \overline{rel}(x_j^t)\wedge)^2], \text{ bzw.}$$
$$\sigma(rel(x_j^t))^{2\wedge} = 1/\delta * \sum_p (rel(x_j^t \mid AM(rel(x) \mid M_{DV(m),p}^{\leq t-1}) - rel(x_j^t)_{boot}\wedge)^2. \quad (700)$$

Auf diese Weise ergibt sich wieder die unvollständige Datenstruktur $m_j^t = (x_j^t, \_, rel(x_j^t)\wedge, bias(x_j^t)\wedge, \sigma(rel(x_j^t))^{2\wedge})$ eines Kandidaten-Stimulus. Wurde dies für alle Elemente von DVM_{prim}^t durchgeführt, so wird nachfolgend auf der Basis von $rel(x_j^t)\wedge$, $bias(x_j^t)\wedge$ und $\sigma(rel(x_j^t))^{2\wedge}$ ein Ranking der Dokumentvektoren erzeugt, was zu der sekundären Ergebnisliste DV_{sec}^t führt.

Die Alternative der Erzeugung einer Polyrepräsentation von Bias- und Varianz-Approximationsmodellen $AM(bias(x))_p^{t-1}$ und $AM(\sigma(rel(x))^2)_p^{t-1}$ ist möglich, indem zunächst an den Stützpunkten x_j der Stimuli aus $M_{DV(m)}^{\leq t-1}$ mit Hilfe der instanzbasierten Relevanz-Approximationsmodelle $AM(rel(x) \mid M_{DV(m),p}^{\leq t-1})$ in Verbindung mit einem Gewinnerlisten-Verfahren Relevanzschätzungen $rel(x_j)_p\wedge$ abgegeben werden. Jedem Stützpunkt lassen sich δ richtige Biaswerte zuordnen, indem der bekannte richtige Relevanzwert $rel(x_j)$ mit einem der δ geschätzten Relevanzwert $rel(x_j)_p\wedge$ verglichen wird:

$$bias(x_j)_p = rel(x_j) - rel(x_j)_p\wedge. \quad (701)$$

Weiterhin können einem Stützpunkt x_j δ Varianzwerte zugeordnet werden, indem die Relevanzwerte $rel(x_k)$ aller Stützpunkte der Stimuli aus $M_{DV(m)}^{\leq t-1} \setminus \{m_j\}$ in Verbindung mit den geschätzten Relevanzwerten $rel(x_j)_p\wedge$ verwendet werden:

$$\sigma(rel(x_j))_p^2 = 1/\nu_j \sum_k h(d_{DVR}(x_j, x_k)) * (rel(x_k) - rel(x_j)_p\wedge)^2,$$
$$\nu_j = \sum_k h(d_{DVR}(x_j, x_k)), \forall m_k \in M_{DV(m)}^{\leq t-1} \setminus \{m_j\}. \quad (702)$$

Jeder Stimulus aus $M_{DV(m)}^{\leq t-1}$ besitzt somit die Datenstruktur:

$$m_j = (x_j, rel(x_j), rel(x_j)\wedge, rel(x_j)_p\wedge, bias(x_j), bias(x_j)_p, \sigma(rel(x_j))^2, \sigma(rel(x_j))_p^2 \mid p = 1, ..., \delta). \quad (703)$$

Zum einen besitzt dieses Verfahren einen deutlich höheren Erzeugungsaufwand im Vergleich zu dem zuerst dargestellten Verfahren. Zum anderen sind drei Schätzungen notwendig, um die Werte $rel(x_j^t)\wedge$, $bias(x_j^t)\wedge$ und $\sigma(rel(x_j^t))^{2\wedge}$ an der Stelle eines Kandidaten-Dokumentvektors aus DVM_{prim}^t zu ermitteln, im Gegensatz zu einer Schätzung, sowie der Bias- und der Varianz-Berechnung. Aus diesen Effizienzgründen soll dieses Verfahren nicht weiter ausgeführt werden.

4.6.2) Prototypbasierte Fehlermodelle

4.6.2.1) Monorepräsentation von prototypbasierten Fehlermodellen

Grundlage aller Verfahrensvarianten ist hierbei die Zuordnung einer Relevanz-, Bias- und Relevanz-Varianzschätzung zu jedem Prototypen, d.h. zu jedem Neuron $n_{DV,k}^{t-1}$ einer individuellen GNG-SOM N_{DV}^{t-1}, die durch die Stimuli aus $M_{DV(m)}^{\leq t-1}$ aus der nicht individuellen GNG-SOM N_{DV} erzeugt wurde. Die Relevanzschätzung wird durch das instanzbasierte Modell $AM(rel(x) \mid M_{DV(m)}^{\leq t-1})$ lokal oder global unter Verwendung aller vorliegenden Stimuli und eines LWR-Approximationsverfahrens bestimmt:

$$\begin{aligned} rel(w(x_{DV})_k^{t-1})\wedge &= rel(w(x_{DV})_k^{t-1} \mid AM(rel(x) \mid M_{DV(m)}^{\leq t-1})) \\ &= 1/\nu_k \sum_j h(d_X(w(x_{DV})_k^{t-1}, x_j)) * rel(x_j), \text{ mit} \\ \nu_k &= \sum_j h(d_X(w(x_{DV})_k^{t-1}, x_j)), \forall m_j \in M_{DV(m)}^{\leq t-1}. \end{aligned} \quad (704)$$

In analoger Weise lässt sich auch die Relevanz-Varianz bestimmen, indem die Vorgehensweise von Cohn et al. (1995[74]) durchgeführt wird, bei der die Relevanzschätzung $rel(w(x_{DV})_k^{t-1})\wedge$ als Mittelwert in der Varianzformel verwendet wird:

$$\begin{aligned} \sigma(rel(w(x_{DV})_k^{t-1}))^2 &= 1/\nu_k \sum_j h(d_X(w(x_{DV})_k^{t-1}, x_j)) * (rel(x_k) - rel(w(x_{DV})_k^{t-1})\wedge)^2, \\ \nu_k &= \sum_j h(d_X(w(x_{DV})_k^{t-1}, x_j)), \forall m_j \in M_{DV(m)}^{\leq t-1}. \end{aligned} \quad (705)$$

Die Festlegung einer Biasschätzung $bias(w(x_{DV})_k^{t-1})\wedge$ gelingt jedoch nicht in der gleichen geradlinigen Weise, da an der Stelle $w(x_{DV})_k^{t-1}$ kein richtiger Relevanzwert bekannt ist und auch kein richtiger Wert durch eine Bewertung des Agenten extern ermittelbar ist, da mit einer hohen Wahrscheinlichkeit kein Dokument in der Dokumentation vorliegt, das als Dokumentvektor $x := w(x_{DV})_k^{t-1}$ besitzt. Aus diesem Grunde muss die Bias an dieser Stelle in jedem Fall durch die Differenz zweier Relevanzschätzungen bestimmt werden. Zu diesem Zweck kann ein Bootstrap-Verfahren auf der Basis der Stimuli in $M_{DV(m)}^{\leq t-1}$ durchgeführt werden, indem δ mal eine Bootstrapliste $M_{DV(m),p}^{\leq t-1}$ durch Ziehen mit Zurücklegen aus $M_{DV(m)}^{\leq t-1}$ erzeugt wird, wobei diese Stimulusliste zusammen mit einem Approximationsverfahren ein instanzbasiertes Relevanz-Approximationsmodell $AM(rel(x) \mid M_{DV(m),p}^{\leq t-1})$ repräsentiert. Dieses Modell erzeugt eine Bootstrap-Einzelschätzung $rel(w(x_{DV})_k^{t-1})_p\wedge = rel(w(x_{DV})_k^{t-1} \mid AM(rel(x) \mid M_{DV(m),p}^{\leq t-1}))$, wobei die δ Schätzungen zu einer Bootstrap-Gesamtschätzung $rel(w(x_{DV})_k^{t-1})_{boot}\wedge$ aggregiert werden. Dieser Schätzwert, von dem angenommen wird, dass er effektiver ist als der Einzelschätzwert $rel(w(x_{DV})_k^{t-1})\wedge$ auf der Basis von $AM(rel(x) \mid M_{DV(m)}^{\leq t-1})$, übernimmt die Rolle des richtigen, unbekannten Relevanzwertes in der Biasformel, sodass sich unter Verwendung der 0,632-Heuristik ergibt:

$$bias(w(x_{DV})_k^{t-1})_{0,632}\wedge = [rel(w(x_{DV})_k^{t-1})_{boot}\wedge - rel(w(x_{DV})_k^{t-1} \mid AM(rel(x) \mid M_{DV(m)}^{\leq t-1}))]/0{,}632. \quad (706)$$

Die GNG-SOM-Monorepräsentation N_{DV}^{t-1} besitzt somit die folgende Datenstruktur, wobei festgelegt werden muss, ob die Relevanzschätzung $rel(w(x_{DV})_k^{t-1})\wedge$, $rel(w(x_{DV})_k^{t-1})_{boot}\wedge$ oder eine Bias-korrigierte Relevanzschätzung von $rel(w(x_{DV})_k^{t-1})\wedge$ mit

$$rel(w(x_{DV})_k^{t-1})\wedge' = rel(w(x_{DV})_k^{t-1})\wedge + bias(w(x_{DV})_k^{t-1})_{0,632}\wedge. \quad (707)$$

für die Relevanz-Approximationen verwendet werden soll:

$$\begin{aligned} N_{DV}^{t-1} = (n_{DV,k}^{t-1} = (&w(x_{DV})_k^{t-1}, rel(w(x_{DV})_k^{t-1})\wedge, rel(w(x_{DV})_k^{t-1})_{boot}\wedge, rel(w(x_{DV})_k^{t-1})\wedge', \\ &bias(w(x_{DV})_k^{t-1})_{0,632}\wedge, \sigma(rel(w(x_{DV})_k^{t-1}))^2, C_k^{t-1}, M_{DV,k}^{t-1}, M_{DV(m),k}^{t-1}) \mid i = 1, ..., \mu_{N,DV}). \end{aligned} \quad (708)$$

Wird die Bias-korrigierte Relevanzschätzung verwendet, so sind die drei Prototypmodelle $AM(rel(x) \mid N_{DV}^{t-1})$, $AM(bias(x) \mid N_{DV}^{t-1})$ und $AM(\sigma(rel(x))^2 \mid N_{DV}^{t-1})$ in Verbindung mit einem Approximationsverfahren festgelegt. Diese Modelle werden eingesetzt, um eine Relevanz-, eine Bias- und eine Varianz-Schätzung für jeden Kandidaten-Dokumentvektor x_j^t aus der Primärmenge DVM_{prim}^t zu erzeugen:

$$rel(x_j^t)^\wedge = 1/\nu_j^t * \sum_k h(d_{DVR}(x_j^t, w(x_{DV})_k^{t-1})) * rel(w(x_{DV})_k^{t-1})^\wedge,$$
$$bias(x_j^t)^\wedge = 1/\nu_j^t * \sum_k h(d_{DVR}(x_j^t, w(x_{DV})_k^{t-1})) * bias(w(x_{DV})_k^{t-1})_{0,632}^\wedge,$$
$$\sigma(rel(x_j^t))^{2\wedge} = 1/\nu_j^t * \sum_k h(d_{DVR}(x_j^t, w(x_{DV})_k^{t-1})) * \sigma(rel(w(x_{DV})_k^{t-1}))^2,$$
$$\nu_j^t = \sum_k h(d_{DVR}(x_j^t, w(x_{DV})_k^{t-1})), \forall n_{DV,k}^{t-1} \in N_{DV}^{t-1}. \quad (709)$$

Mit Hilfe dieser drei Schätzungen wird ein Ranking der Elemente aus DVM_{prim}^t gebildet, was zu der sekundären Ergebnisliste DV_{sec}^t führt, wobei das zugrunde liegende Sortierungsverfahren, das gleiche ist, wie bei den oben dargestellten instanzbasierten Verfahren.

Besitzen die Stimuli m_j aus $M_{DV(m)}^{\leq t-1}$ bereits eine Datenstruktur $m_j = (x_j, rel(x_j), rel(x_j)^\wedge, bias(x_j), \sigma(rel(x_j))^2)$, so besteht die Möglichkeit, die Bias- und die Relevanz-Varianz an der Stelle der Gewichtsvektoren der Neurone durch eine LWR-Approximation analog wie die Relevanzschätzung durchzuführen:

$$rel(w(x_{DV})_k^{t-1})^\wedge = 1/\nu_k \sum_j h(d_X(w(x_{DV})_k^{t-1}, x_j)) * rel(x_j),$$
$$bias(w(x_{DV})_k^{t-1})^\wedge = 1/\nu_k \sum_j h(d_X(w(x_{DV})_k^{t-1}, x_j)) * bias(x_j),$$
$$\sigma(rel(w(x_{DV})_k^{t-1}))^{\wedge 2} = 1/\nu_k \sum_j h(d_X(w(x_{DV})_k^{t-1}, x_j)) * \sigma(rel(x_j))^2, \text{ mit}$$
$$\nu_k = \sum_j h(d_X(w(x_{DV})_k^{t-1}, x_j)), \forall m_j \in M_{DV(m)}^{\leq t-1}. \quad (710)$$

4.6.2.2) Polyrepräsentation von prototypbasierten Fehlermodellen

Ziel der Polyrepräsentation ist hier die Spezifizierung von δ Relevanz-Approximationsmodellen $AM(rel(x) \mid N_{DV,p}^{t-1})$, Bias-Approximationsmodellen $AM(bias(x) \mid N_{DV,p}^{t-1})$ und Relevanz-Varianz-Approximationsmodellen $AM(\sigma(rel(x))^2 \mid N_{DV,p}^{t-1})$. Grundlage soll die individuelle GNG-SOM N_{DV}^{t-1} sein, aus der durch Stimulus-Bootstraping GNG-SOMs $N_{DV,p}^{t-1}$, $p = 1, ..., \delta$, gebildet werden. Hierzu werden aus $M_{DV(m)}^{\leq t-1}$ Bootstraplisten $M_{DV(m),p}^{\leq t-1}$ gebildet, die zu einer Reformulierung von N_{DV}^{t-1} verwendet werden, wobei eine reformulierte GNG-SOM mit $N_{DV,p}^{t-1}$ bezeichnet wird. Die Reformulierung kann wieder durch die zwei Strategien erfolgen (siehe Abschnitt 4.4.2.2)):

1) Reformulierung der Relevanzschätzungen bei Beibehaltung der Gewichtsvektoren.
2) Reformulierung der Gewichtsvektoren und der Relevanzschätzungen.

Aus Effizienzgründen soll die erste Möglichkeit betrachtet werden, d.h. korrespondierende Neurone $n_{DV,p,k}^{t-1}$ in den einzelnen Modellen $N_{DV,p}^{t-1}$ besitzen den gleichen Gewichtsvektor $w(x_{DV})_k^{t-1}$, die gleiche Dokumentvektorenmenge $M_{DV,k}^{t-1}$ und die gleichen Verbindungsvektoren C_k^{t-1}. Sie unterscheiden sich durch die lokale Bootstrap-Stimulusliste $M_{DV(m),p,k}^{t-1}$ und die daraus abgeleitete Relevanzschätzung $rel(w(x_{DV})_k^{t-1})_p^\wedge$:

$$N_{DV,p}^{t-1} = (n_{DV,p,k}^{t-1} = (w(x_{DV})_k^{t-1}, rel(w(x_{DV})_k^{t-1})_p^\wedge, C_k^{t-1}, M_{DV,k}^{t-1}, M_{DV(m),p,k}^{t-1}) \mid i = 1, ..., \mu_{N,DV}), p = 1, ..., \delta. \quad (711)$$

Die einzelnen Komponenten können auch in einer gemeinsamen Datenstruktur integriert werden:

$$N_{DV}^{t-1} = (n_{DV,k}^{t-1} = (w(x_{DV})_k^{t-1}, rel(w(x_{DV})_k^{t-1})\wedge, rel(w(x_{DV})_k^{t-1})_p\wedge, C_k^{t-1}, M_{DV,k}^{t-1}, M_{DV(m),k}^{t-1}, M_{DV(m),k}^{t-1} \mid p = 1, ..., \delta) \mid i = 1, ..., \mu_{N,DV}). \quad (712)$$

Einem Neuron $n_{DV,p,k}^{t-1}$ kann eine Relevanz-Varianz zugeordnet werden, indem die Relevanzschätzung $rel(w(x_{DV})_k^{t-1})_p\wedge$ als Mittelwert in der Varianzformel verwendet wird (Cohn et al. (1995[74])). Werden bei der Relevanzschätzung wie bei der Varianzbestimmung alle Elemente aus $M_{DV(m),p}^{t-1}$ verwendet, so ergibt sich:

$$\sigma(rel(w(x_{DV})_k^{t-1}))_p^2 = 1/\nu_{p,k} \sum_j h(d_X(w(x_{DV})_k^{t-1}, x_j)) * (rel(x_k) - rel(w(x_{DV})_k^{t-1})_p\wedge)^2,$$
$$\nu_{p,k} = \sum_j h(d_X(w(x_{DV})_k^{t-1}, x_j)), \forall m_j \in M_{DV(m),p}^{t-1}. \quad (713)$$

Im letzten Schritt muss jedem Neuron eine Biasschätzung zugeordnet werden, wobei zunächst eine Relevanzschätzung gebildet wird, von der angenommen werden kann, dass sie effektiver ist, als eine der Einzel-Schätzungen $rel(w(x_{DV})_k^{t-1})_p\wedge$. Dies gilt für die Bootstrap-Gesamtschätzung, bei der alle δ Einzelschätzungen aggregiert werden:

$$rel(w(x_{DV})_k^{t-1})_{boot}\wedge = 1/\delta \sum_p rel(w(x_{DV})_k^{t-1})_p\wedge. \quad (714)$$

Eine 0,623-Biasschätzung für das Neuron $n_{DV,p,k}^{t-1}$ ergibt sich, indem man die Differenz aus Bootstrap-Gesamtschätzung und Einzelschätzung bildet, dividiert durch 0,632:

$$bias(w(x_{DV})_k^{t-1})_{p,0,632}\wedge = [rel(w(x_{DV})_k^{t-1})_{boot}\wedge - rel(w(x_{DV})_k^{t-1})_p\wedge)]/0,632. \quad (715)$$

Die Datenstruktur einer Bootstrap-GNG-SOM $N_{DV,p}^{t-1}$ wird entsprechend den Relevanz- und den Biaswerten erweitert, sodass damit die drei Modelle $AM(rel(x) \mid N_{DV,p}^{t-1})$, $AM(bias(x) \mid N_{DV,p}^{t-1})$ und $AM(\sigma(rel(x))^2 \mid N_{DV,p}^{t-1})$ vorliegen:

$$N_{DV,p}^{t-1} = (n_{DV,p,k}^{t-1} = (w(x_{DV})_k^{t-1}, rel(w(x_{DV})_k^{t-1})_p\wedge, bias(w(x_{DV})_k^{t-1})_{p,0,632}\wedge, \sigma(rel(w(x_{DV})_k^{t-1}))_p^2, C_k^{t-1}, M_{DV,k}^{t-1}, M_{DV(m),p,k}^{t-1}) \mid i = 1, ..., \mu_{N,DV}), p = 1, ..., \delta. \quad (716)$$

Mit diesen Approximationsmodellen sollen die Dokumentvektoren x_j aus DVM_{prim}^t bewertet und in eine Rangfolge gebracht werden, d.h. jeder unvollständige Stimulus $m_j = (x_j, \_)$ wird um eine Relevanz-, Bias- und Relevanz-Varianz-Schätzung erweitert, die z.B. zu Relevanz-Intervallen aggregiert werden, mit denen ein Ranking aufgebaut wird. Der geradlinigste Weg hierzu ist die Bildung von je δ Einzelschätzungen:

$$rel(x_j)_p\wedge = 1/\nu_{p,j} \sum_k h(d_X(x_j, w(x_{DV})_k^{t-1})) * rel(w(x_{DV})_k^{t-1})_p\wedge,$$
$$bias(x_j)_p\wedge = 1/\nu_{p,j} \sum_k h(d_X(x_j, w(x_{DV})_k^{t-1})) * bias(w(x_{DV})_k^{t-1})_{p,0,632}\wedge,$$
$$\sigma(rel(x_j))_p\wedge^2 = 1/\nu_{p,j} \sum_k h(d_X(x_j, w(x_{DV})_k^{t-1})) * \sigma(rel(w(x_{DV})_k^{t-1}))_p^2, \text{ mit}$$
$$\nu_{p,j} = \sum_k h(x_j, d_X(w(x_{DV})_k^{t-1})), \forall n_{DV,p,k}^{t-1} \in N_{DV,p}^{t-1}, p = 1, ..., \delta. \quad (717)$$

Diese werden in einem nachfolgenden Schritt aggregiert, sodass sich die erweiterte, jedoch noch unvollständige Datenstruktur von m_j ergibt als:

$$m_j = (x_j, \_, rel(x_j)_{ges}\hat{}, bias(x_j)_{ges}\hat{}, \sigma(rel(x_j))_{ges}\hat{}^2), \text{ mit}$$
$$rel(x_j)_{ges}\hat{} = 1/\delta \sum_p rel(x_j)_p\hat{},$$
$$bias(x_j)_{ges}\hat{} = 1/\delta \sum_p bias(x_j)_p\hat{},$$
$$\sigma(rel(x_j))_{ges}\hat{}^2 = 1/\delta \sum_p bias(x_j)_p\hat{}. \quad (718)$$

Eine alternative Vorgehensweise kann bei einer Modell-Polyrepräsentation durchgeführt werden, indem darauf verzichtet wird, den Neuronen Bias- und Relevanz-Varianz-Schätzungen zuzuordnen, sondern die Bias- und Relevanz-Varianz-Schätzungen werden erst an der Stelle der Kandidaten-Dokumentvektoren gebildet. An der Stelle x_j^t werden die Relevanzschätzungen $rel(x_j^t)\hat{}$ und $rel(x_j^t)_p\hat{}$, p = 1, ..., δ, der Modelle N_{DV}^{t-1} und $N_{DV,p}^{t-1}$ erzeugt. Aus diesen Einzelschätzungen wird ein Relevanz-Mittelwert gebildet:

$$\overline{rel}(x_j^t)\hat{} = 1/(\delta+1) * [rel(x_j^t)\hat{} + \sum_p rel(x_j^t)_p\hat{}]. \quad (719)$$

Dieser Relevanzwert wird als Mittelwert in der allgemeinen Varianz-Formel eingesetzt:

$$\sigma(rel(x_j^t))^2 = 1/(\delta+1) * [(rel(x_j^t)\hat{} - \overline{rel}(x_j^t)\hat{})^2 + \sum_p (rel(x_j^t)_p\hat{} - \overline{rel}(x_j^t)\hat{})^2. \quad (720)$$

Für die Biasschätzung kann der Mittelwert der Relevanzschätzungen der Bootstrapmodelle gebildet werden:

$$rel(x_j^t)_{boot}\hat{} = 1/\delta \sum_p rel(x_j^t)_p\hat{}. \quad (721)$$

Eine 0,623-Biasschätzung ergibt sich, indem man die Differenz aus Bootstrap-Gesamtschätzung und Einzelschätzung $rel(x_j^t)\hat{}$ bildet, dividiert durch 0,632:

$$bias(x_j^t)_{0,632}\hat{} = [rel(x_j^t)_{boot}\hat{} - rel(x_j^t)\hat{})]/0,632. \quad (722)$$

Ein Kandidat m_j^t besitzt demnach die folgende Datenstruktur, mit der die Relevanzschätzung korrigiert und eine Rangfolge der Kandidaten gebildet wird:

$$m_j^t = (x_j^t, \_, rel(x_j^t)\hat{}, rel(x_j^t)_p\hat{}, bias(x_j^t)_{0,632}\hat{}, \sigma(rel(x_j^t))^2\hat{} \mid p = 1, ..., \delta). \quad (723)$$

4.7) Unterschiedliche Gewichtung von Relevanzmaximierung und Modellaufbau

Bislang wurden Retrieval-Strategien beschrieben, die ausschließlich das Ziel verfolgten, Dokumentvektoren mit hohen Relevanzwerten zu finden, ohne Rücksicht darauf, ob die sich ergebenden Stimuli gut lernbar sind. D.h. das vorliegende prototypbasierte Approximationsmodell muss bei seiner Aktualisierung die Aufgabe leisten, die Stimuli durch Erzeugung geeigneter Stützpunkte im DVR und im Relevanzraum zu lernen. Durch den Nachweis von Dokumentvektoren mit immer besseren Relevanzwerten kommt es zu einer Spezialisierung des betreffenden Approximationsmodells, da keine Stimuli mit schlechten Relevanzwerten nachgewiesen und somit gelernt werden. Diese systematische Verzerrung ist eine verfahrensimmanente Eigenschaft und somit eine Form von Bias, die dazu führen kann, dass Dokumentvektoren mit schlechten Relevanzwerten schlechter erkannt werden bzw. dass ihnen systematisch

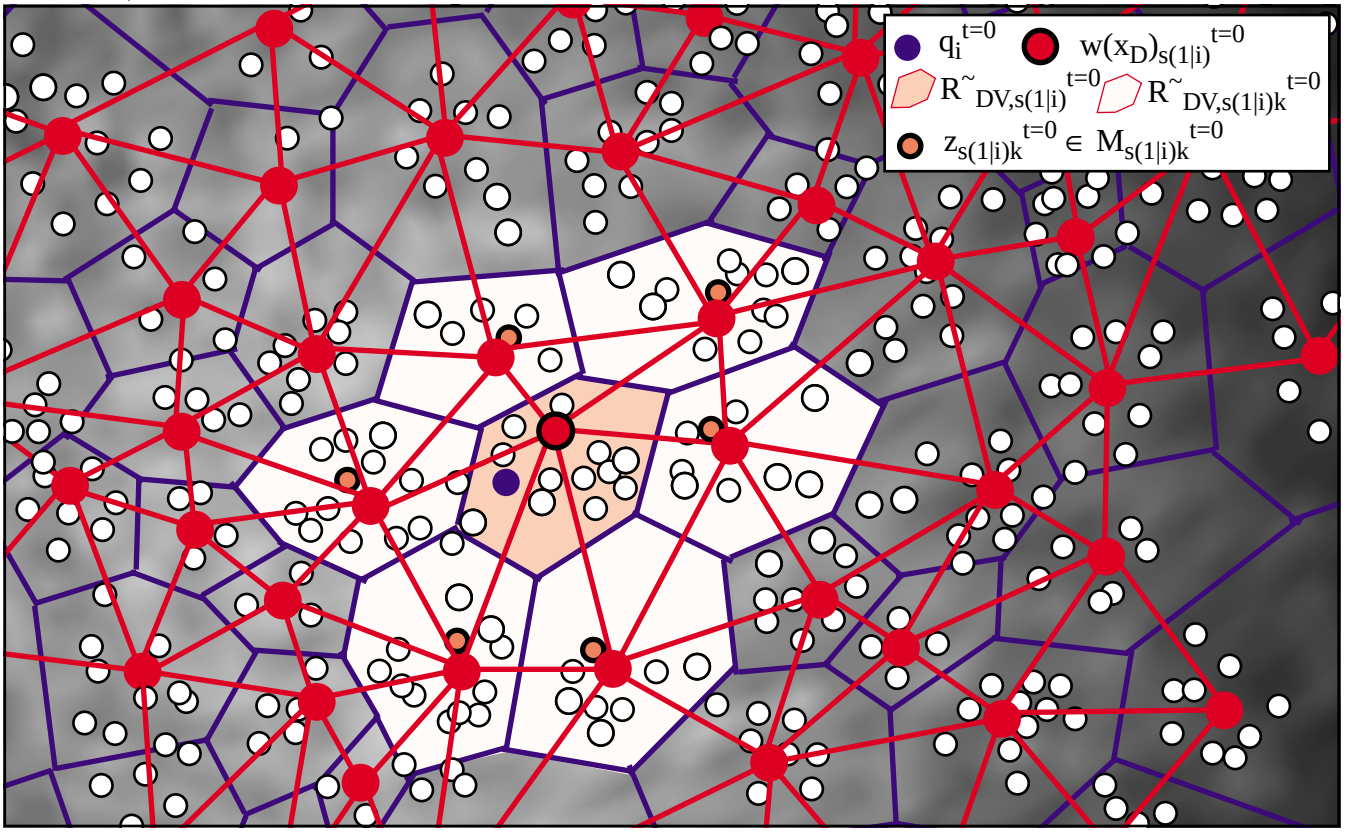
eine zu gute Relevanzschätzung zugeordnet wird. Dies ist jedoch im gegebenen Kontext nicht per se eine schlechte Eigenschaft, da keine absoluten Relevanzschätzungen für das Retrieval bestimmend sind, sondern bestimmend ist die Unterscheidung, ob ein Dokumentvektor ausgewählt wird, damit das korrespondierende Dokument dem Agenten präsentiert wird, bzw. es ist die Reihenfolge der Präsentation, d.h. das Ranking bestimmend. Werden Dokumentvektoren mit einem guten Relevanzwert adäquat geschätzt und werden Dokumentvektoren mit einem schlechten Relevanzwert zu gut geschätzt, so kann dies trotzdem zu einer adäquaten Reihenfolge bzw. einer Unterscheidung zwischen nachgewiesenen und nicht nachgewiesenen Dokumenten führen.

Sollen jedoch alle Klassen von Dokumentvektoren adäquat geschätzt werden, so erfordert dies gute wie schlechte Stützpunkte, was das Retrieval zu einem Zwei-Ziel-Entscheidungsproblem mit konkurrierenden Zielen macht. Zum einen existiert das Ziel den Agenten mit den Dokumenten zu versorgen, von dem das IRS annimmt, dass es diejenigen mit den besten Relevanzwerten sind, wobei die Anzahl der präsentierten Dokumente klein sein soll, um den Bewertungsaufwand des Agenten gering zu halten. Zum anderen existiert das Ziel, ein agentenspezifisches, adäquates Relevanz-Approximationsmodell aufzubauen, das für den Agenten oder eine Agentengruppe wiederverwendbar sein soll, wobei hier die Anzahl der Stimuli und somit der Stützpunkte nicht zu klein sein darf, damit die Relevanzschätzungen mit einem vertretbaren Fehlerwert durchführbar sind. Jedem der beiden Ziele kann ein eigener Verfahrensteil zugeordnet werden, in dem Dokumentvektoren entsprechend dem Ziel nachgewiesen werden, wobei der erste Teil als Relevanzmaximierung und der zweite Teil als Modellaufbau bzw. Modellmaximierung bezeichnet werden kann. Unterschiedliche Verfahrensvarianten ergeben sich aus der unterschiedlichen Gewichtung von Relevanz- und Modellmaximierung. Die Relevanzmaximierungs-Komponente konzentriert sich auf die kurzfristige Strategie des Nachweises der geschätzten besten Dokumentvektoren, während die Modellmaximierungs-Komponente sich auf eine längerfristige Strategie bezieht, die erst in der oder den nächsten Feedback-Iterationen greift, da dort mit dem neuen verbesserten Modell auch die Relevanzschätzungen besser werden. Wird die Gewichtung zugunsten des Modellaufbaus verschoben, so wird ewartet, dass die Effektivitätsteigerung in der nächsten Feedback-Iteration die anfallenden Kosten durch die aufgeschobene Befriedigung des Informationsbedürfnisses überwiegen wird. Diese Grundargumentation wird im Rahmen des aktiven Lernens eine wesentliche Rolle spielen (siehe Kapitel 5)), sodass die nachfolgenden Darstellungen den Übergang zum aktiven Lernen beschreiben.

Eine Strategie, die von den bislang vorgestellten Verfahren wenig abweicht, ermittelt neben den Dokumentvektoren in der lokalen Stimulusmenge $M_{DV,s(1|i)}^{t=0}$ des Gewinner-Neurons $n_{DV,s(1|i)}^{t=0}$ weitere Dokumentvektoren aus den lokalen Stimulusmengen $M_{DV,s(1|i)k}^{t=0}$ der Nachbarneurone aus $N(d_G=1 \mid G_{DV})_{s(1|i)}$. Welche Eigenschaften diese Dokumentvektoren besitzen müssen, um ausgewählt zu werden, kann unterschiedlich definiert werden, wobei eine sinnvolle Auswahl die Median-Dokumentvektoren verwendet. D.h. aus einer Menge $M_{DV,s(1|i)k}^{t=0}$ wird der Dokumentvektor $z_{s(1|i)k}^{t=0}$ ermittelt, dessen Distanz zum Gewichtsvektor $w(x_{DV})_{s(1|i)k}^{t=0}$ am geringsten ist (siehe Abb. 129)):

$$d_{DVR}(z_{s(1|i)k}^{t=0}, w(x_{DV})_{s(1|i)k}^{t=0}) = \min\{d_{DVR}(x_j^{t=0}, w(x_{DV})_{s(1|i)k}^{t=0}) \mid \forall x_j^{t=0} \in M_{DV,s(1|i)k}^{t=0}\}. \quad (724)$$

Abb. 129) Gewinner-Neuron und seine Nachbarn mit Median-Dokumentvektoren



Die primäre Ergebnismenge der Initialisierungs-Iteration ergibt sich somit als:

$$DVM(q_i^{t=0}) := M_{DV,s(1|i)}^{t=0} \cup \{z_{s(1|i)k}^{t=0} \mid \forall n_{DV,s(1|i)k}^{t=0} \in N(d_G=1 \mid G_{DV})_{s(1|i)}\}. \quad (725)$$

Nachdem die Bewertung durch den Agenten erfolgt ist, besitzt jeder Dokumentvektor aus $DVM(q_i^{t=0})$ einen Relevanzwert, sodass sich die Stimulusmenge $M_{DV}^{t=0}$ ergibt:

$$M_{DV}^{t=0} = (m_j^{t=0} = (x_j^{t=0}, rel(x_j^{t=0})), m_k^{t=0} = (z_{s(1|i)k}^{t=0}, rel(z_{s(1|i)k}^{t=0})) \mid$$
$$\forall x_j^{t=0} \in M_{DV,s(1|i)}^{t=0}, \forall n_{DV,s(1|i)k}^{t=0} \in N(d_G=1 \mid G_{DV})_{s(1|i)}\}. \quad (726)$$

Mit diesen Stimuli liegt ein instanzbasiertes Approximationsmodell vor, das als $AM(rel(x) \mid M_{DV}^{t=0})$ bezeichnet wird. Mit diesem Approximationsmodell werden Relevanzschätzungen an der Stelle der Gewichtsvektoren $w(x_{DV})_{s(1|i)}^{t=0}$ und $w(x_{DV})_{s(1|i)k}^{t=0}$, $\forall n_{DV,s(1|i)k}^{t=0} \in N(d_G=1 \mid G_{DV})_{s(1|i)}$, durchgeführt, d.h. es ergeben sich Stützpunkte $(w(x_{DV})_{s(1|i)}, rel(w(x_{DV})_{s(1|i)}^{t=0})\text{^})$ und $(w(x_{DV})_{s(1|i)k}, rel(w(x_{DV})_{s(1|i)k}^{t=0})\text{^})$, die ein prototypbasiertes Approximationsmodell $AM(rel(x) \mid N(d_G \leq 1 \mid G_{DV})_{s(1|i)})^{t=0}$ bilden.

Bei der zweiten Strategie, die eine Verschiebung der Gewichtung zu Gunsten des Modellaufbaus verwendet, wird in der Initialisierungs-Iteration auf die Menge $M_{DV,s(1|i)}^{t=0}$ in ihrer Gesamtheit verzichtet, und es wird nur der Median-Dokumentvektor $z_{s(1|i)}^{t=0}$ aus dieser Menge ausgewählt, der neben den anderen Medianvektoren die Ergebnismenge bildet:

$$DVM(q_i^{t=0}) := \{z_{s(1|i)}^{t=0}, z_{s(1|i)k}^{t=0} \mid \forall n_{DV,s(1|i)k}^{t=0} \in N(d_G=1 \mid G_{DV})_{s(1|i)}\}. \quad (727)$$

Nachdem die Bewertung durch den Agenten erfolgt ist, ergibt sich die Stimulusmenge $M_{DV}^{t=0}$:

$$M_{DV}^{t=0} = (m_{s(1|i)}^{t=0} = (z_{s(1|i)}^{t=0}, rel(z_{s(1|i)}^{t=0})), m_k^{t=0} = (z_{s(1|i)k}^{t=0}, rel(z_{s(1|i)k}^{t=0})) | \forall n_{DV,s(1|i)k}^{t=0} \in N(d_G=1 \mid G_{DV})_{s(1|i)}\}. \quad (728)$$

Das instanzbasierte Modell $AM(rel(x) \mid M_{DV}^{t=0})$ wird in der Nachfolge-Iteration t=1 verwendet, um den Dokumentvektoren jeweils eine Relevanzschätzung zuzuordnen, wobei z.B. die restlichen Dokumentvektoren in den lokalen Stimulusmengen $M_{DV,s(1|i)}^{t=0}$ und $M_{DV,s(1|i)k}^{t=0}$, $\forall n_{DV,s(1|i)k}^{t=0} \in N(d_G=1 \mid G_{DV})_{s(1|i)}$, verwendet werden können, d.h. die primäre Ergebnismenge $DVM_{prim}^{t=1}$ wird festgelegt durch:

$$DVM_{prim}^{t=1} = \{M_{DV,s(1|i)}^{t=0} \cup \{\bigcup_k M_{DV,s(1|i)k}^{t=0}\}\} \setminus M_{DV}^{t=0}. \quad (729)$$

Mit Hilfe von $AM(rel(x) \mid M_{DV}^{t=0})$ werden den Elementen $x_j^{t=1} \in DVM_{prim}^{t=1}$ Relevanzschätzungen zugeordnet, gefolgt von der Sortierung nach fallendem Relevanzwert, wodurch sich die sekundäre Ergebnisliste $DV_{sec}^{t=1}$ ergibt. Aus dieser werden die ersten Elemente in die tertiäre Liste $DV_{ter}^{t=1}$ übernommen, wobei z.B. $\#N(d_G \leq 1 \mid G_{DV})_{s(1|i)}$ Elemente übernommen werden können.

Andere Strategien verwenden neben dem Median-Dokumentvektor, als dem Dokumentvektor mit der kleinsten Distanz zu einem Gewichtsvektor, weitere Dokumentvektoren aus einer lokalen Stimulusmenge, wie z.B. der Dokumentvektor mit der zweitgeringsten Distanz, worauf jedoch nicht näher eingegangen werden soll.

Die lokale Zentrierung auf $n_{DV,s(1|i)}^{t=0}$ und seine Nachbar-Neurone ergibt sich durch die Erzeugung einer Query Q_i durch den Agenten und deren Indexierung zu $q_i^{t=0}$ durch das IRS. Im Rahmen eines Modellaufbaus beim Vorliegen einer Strukturierung der Dokumentvektorenmenge durch N_{DV} existiert jedoch die prinzipielle Möglichkeit in einer Initialisierung einen IRS-Output zu erzeugen, ohne dass ein Agent eine Query als Input liefert, d.h. es wird ein Retrieval ohne Query durchgeführt, die ausschließlich zum Modellaufbau dient. Dieser Initialisierungs-Output ist unabhängig von einem Agenten, seiner Problemstellung und seiner Query, und könnte somit standardisiert immer am Anfang einer Agent-IRS-Interaktion stehen. Zu diesem Zweck können alle Median-Dokumentvektoren, die zu N_{DV} gehören, ausgewählt werden:

$$DVM(N_{DV})^{t=0} := \{z_j^{t=0} \mid \forall n_{DV,j}^{t=0} \in N_{DV}\}. \quad (730)$$

Da die Anzahl der Neurone bei einer Dokumentvektoren-Clusterung in der Regel zu groß ist, damit ein Agent diese Menge bearbeiten kann, müssen Strategien eingeführt werden, um diese Menge deutlich zu verringern. Es können u.a. folgende Strategien hierzu angewendet werden:

1) Durchführung einer zufälligen, nicht lokalen Ziehung der Neurone aus N_{DV}, deren Median-Dokumentvektoren verwendet werden.
2) Aufzeichnung der Erzeugungshistorie der GNG-SOM N_{DV} (siehe Bachelier (1999d:171ff[22])), bei der bestimmte Zwischenzustände beim Wachstum gespeichert werden, z.B. die Netzstruktur, die nach einer Adaptionsphase und vor dem Einfügen des nächsten Neurons vorliegt. Besitzt das Netz N_{DV} nach dem Wachstumsabbruch $\mu_{N,D}$ Neurone, so können auf diese Art $\mu_{N,D}$ - 1 Zwischenzustände

erzeugt werden, da mit zwei Initialisierungs-Neuronen begonnen wird. Aus dieser Netzmenge kann ein Netz mit einer geeigneten Neuronenanzahl ausgewählt werden. Wird während des Netzaufbaus der Median-Dokumentvektor zu jedem Neuron bestimmt, so können diese Angaben direkt verwendet werden, und die Median-Dokumentvektoren werden in $DVM(N_{DV})^{t=0}$ aufgenommen.

3) Erweiterung der Strukturierungsgrundlage der Dokumentvektorenmenge zu einer Neuronen-Cluster-GNG-SOM (NC-GNG-SOM; siehe Bachelier (1998c:15ff[17])) bzw. einer Kombination aus Neuronen- und Stimulus-Cluster-GNG-SOM (NSC-GNG-SOM), indem die Gewichtsvektoren $w(x_{DV})_j$ aller Neurone $n_{DV,j}$ aus N_{DV} als Stimuli einer weiteren, unüberwachten Strukturierung durch eine SC-GNG-SOM verwendet werden. Es entsteht eine zusätzliche GNG-SOM N_{GV}, deren Neurone lokale Stimulusmengen besitzen, in denen Gewichtsvektoren der Neurone aus N_{DV} enthalten sind. Für jedes Neuron aus N_{GV} muss der Median-Dokumentvektor ermittelt werden, der in $DVM(N_{DV})^{t=0}$ aufgenommen wird.

5) Aktives Lernen in Mono- und Polyrepräsentations-IRS

Wie im Überblick in Kapitel 1) skizziert, wird im Rahmen dieser Arbeit rationales, aktives Lernen betrachtet, als die iterative Einflussnahme des Lernenden auf die Auswahl der als nächstes zu lernenden Stimuli mit dem Ziel, den Modellaufbau effizient, d.h. z.B. mit wenigen Stimuli, durchzuführen.

Zunächst wird in diesem Kapitel der Unterschied zwischen passivem und aktivem Lernen beschrieben (siehe Abschnitt 5.1)), gefolgt von der Dartsellung von drei prinzipiellen Szenarien des aktiven Lernens (siehe Abschnitte 5.1.2) bis 5.1.4)), sowie die Spezifizierung einer dieser Abläufe als Analogie zum Relevanz-Feedback im IR. Innerhalb dieses Szenarios folgt die detaillierte Darstellung von indirekten (siehe Abschnitt 5.2.1)) sowie direkten Vorgehensweisen (siehe Abschnitt 5.2.2)) des aktiven Lernens. Indirekte Verfahren werden auf Grund ihres vergleichsweise geringen Aufwandes betrachtet, während die implizite Annahme der Korrelation zwischen Eigenschaften von Kandidatenstimuli und Eigenschaften der Modelle, die erzeugt werden, wenn Kandidaten als reguläre Lernstimuli aufgenommen werden, ein Effektivitätsproblem beinhaltet. Dem gegenüber sind direkte Verfahren statistisch wie logisch begründbar, sodass sie ein solches Effektivitätsproblem nicht besitzen, doch sie besitzen ein Effizienzproblem, da ihr Aufwand ein Vielfaches dessen von indirekten Verfahren beträgt. In Anbetracht dieses großen Aufwandes, werden einige Strategien diskutiert, wie dieser verringert werden kann (siehe Abschnitt 5.2.3)), ohne jedoch erwarten zu können, dass der Aufwand auch nur annähernd dem der indirekten Verfahren angenähert werden kann. Eingeschoben wird zudem ein Vorschlag für Effektivitätsverbesserungen bei direkten Verfahren, da die vorgestellten Verfahren zur Bestimmung der Modellqualitäten ein anderes Effektivitätsproblem besitzen, da nur Mittelwerte (approximierte Integrale) verwendet werden. Effektiver wäre die Verwendung zusätzlicher Maße, welche die Verteilung der bestimmten Modellqualitäten umfassender beschreiben, was durch zentrale Momente höherer Ordnung erfolgt (siehe Abschnitt 5.2.4)).

Im Kontext des IR kann aktives Lernen allein nicht sinnvoll eingesetzt werden, da der Agent schnell relevante Dokumente erwartet und da von keiner großen Bereitschaft ausgegangen werden kann, viele Dokumente mit einem Relevanzwert zu bewerten. Aus diesem Grunde muss eine Integration des Relevanz-Maximierungskriteriums (Standard-Relevanz-Feedback) und des Modell-Maximierungskriterium (ausschließlich aktives Lernen) durchgeführt werden, sodass einige alternative Strategien der Integration vorgestellt werden (siehe Abschnitt 5.3)).

Zum Abschluss wird ein Lösungsansatz für das Kombinatorikproblems bei direkten Verfahren vorgeschlagen (siehe Abschnitt 5.4)). Das Kombinatorikproblem ergibt sich bei den direkten Verfahren, da für eine Kandidatenmenge deren Potenzmenge getestet werden müsste, um zu entscheiden, welche Kandidaten in eine momentane Lernmenge aufgenommen werden müssten, damit das best mögliche Modell erzeugt wird. Dies ist schon bei kleinen Kandidatenmengen nicht mehr möglich. Die Forderung von zwei Optimierungskriterien (Relevanz- und Modell-Maximierungskriterium) ermöglich jedoch einen neuen Lösungsansatz für dieses Problem.

5.1) Passives und aktives Lernen

Standardmäßig werden Formen von passivem Lernen in Machine-Learning-Systemen, Neuronalen Netzen oder statistischen Lernsystemen verwendet, die dadurch gekennzeichnet sind, dass eine Menge M von Stimuli extern gegeben ist. Neben der Lernmenge ist auch das Lernverfahren extern gegeben, d.h. ein Algorithmus, mit dem aus M ein Modell gebildet wird. Bei einem passiven Lernen besitzt der Lerner somit keinen Einfluss darauf was er lernt und wie er es lernt.

Demgegenüber steht das aktive Lernen, bei dem der Lerner Einfluss auf den Lernprozess besitzt. Das Spektrum der Entscheidungen eines aktiven Lerners umfasst Fragen wie „was soll gelernt werden“, „wie soll gelernt werden“, „wann soll das Lernen beendet werden“, usw. Zur Beendigung des Lernens wird ein Abbruchkriterium benötigt, das auf der Basis eines oder mehrerer Qualitätsmaße gebildet wird, sodass die Wahl der Qualitätsmaße in Verbindung mit Schwellenwerten o.ä. ebenfalls in eine umfassende Definition vom aktivem Lernen gehört.

Passives und aktives Lernen können verstanden werden als Punkte auf einer Skala, welche das Ausmaß des Einflusses des Lerners beschreibt. Passives Lernen bezeichnet den Nullpunkt dieser Skala und alle anderen Lernformen bilden mehr oder weniger ausgeprägte Formen des aktiven Lernens.

Wird ein Lernprozess charakterisiert durch eine Lernmenge und ein Lernverfahren, so ergeben sich drei Grundkonfigurationen des aktiven Lernens:
1) Einfluss auf die Lernmenge, d.h. auf das was gelernt wird, und kein Einfluss auf das Lernverfahren.
2) Einfluss auf das Lernverfahren, d.h. auf das wie gelernt wird, und kein Einfluss auf die Lernmenge.
3) Einfluss auf die Lernmenge und das Lernverfahren.

In den meisten Untersuchungen zum aktiven Lernen wurde sich auf den Einfluss auf die Lernmenge bei gegebenem Lernverfahren beschränkt. Dies ist im Hinblick auf die Auswahl von Dokumentvektoren in einer Iteration der Interaktion zwischen einem Agenten und einem IRS eine sinnvolle Einschränkung, die im weiteren übernommen werden soll. Das IRS als Lerner soll im Rahmen der weiteren Arbeit keinen Einfluss auf das Lernverfahren d.h. das stützpunktbasierte Approximationsverfahren nehmen, da eine instanz- oder prototypbasierte LWR-Variante verwendet werden soll.

Die Nutzung eines Approximationsverfahrens in Iterationen $t > 0$, wie sie im Kapitel 4) beschrieben wurden, ist dabei interpretierbar als eine einfache Form des aktiven Lernens des ersten Typs, da durch die Auswahl einer Dokumentvektorenliste DV_{ter}^{t}, von der das System annimmt, dass es sich um die korrespondierenden Dokumente mit den größten erwarteten Relevanzwerten handelt, die Lernmenge der nachfolgenden Iteration eindeutig festgelegt wird, wenn angenommen wird, dass der Agent alle vorgeschlagenen Dokumente bewertet. Der Aufbau des Modells $AM(rel(x))^{t=0}$ in der Initialisierungs-Iteration wird demgegenüber in den meisten der in Kapitel 4) dargestellten Verfahren als passives Lernen verstanden, wenn z.B. die Dokumentvektoren ausgewählt werden, die in einer ε-Umgebung oder in der Voronoi-Region des Gewinner-Neurons liegen.

Die Einflussnahme auf Lernmengen zukünftiger Lernphasen kann durch unterschiedliche Kriterien erfolgen, wobei die Auswahl der Stimuli mit den besten Outputschätzungen wie die besten Relevanzschätzungen eine Möglichkeit darstellt. Beim allgemeinen Modellaufbau durch aktives Lernen werden demgegenüber Kriterien verwendet, die dazu führen, dass das Modell möglichst schnell gute Qualitätswerte erreicht. D.h. die Auswahl nach besten Outputschätzungen kann als ein externes Kriterium betrachtet werden, da dadurch das Modell nicht besser werden muss. Im Gegenteil wird im Laufe der Iterationen eine systematische Bias eingeführt, indem das Modell darauf abgestimmt wird, die Stimuli mit den besten Outputwerten zu erkennen und zu bewerten. Ein internes Kriterium wählt Stimuli danach aus, welchen Einfluss die Aufnahme des Stimulus in die Lernmenge und das daraus aktualisierte Modell haben wird. Dies bedeutet, dass auch Stimuli mit schlechten Outputwerten ausgewählt werden, damit in das Modell keine systematische Bias eingeführt wird und es in der Lage ist, gute wie schlechte Inputvektoren zu erkennen und zu bewerten.

Externe und interne Kriterien sind somit in Bezug auf endliche Ressourcen konkurrierende Ansätze, d.h. im Kontext eines IRS dürfen nicht zu viele Dokumentvektoren nachgewiesen werden, die ausschließlich dem Modellaufbau dienen, und es müssen möglichst wenige Dokumentvektoren nachgewiesen werden, von dem das System annimmt, dass es sich um nicht relevante korrespondiernde Dokumente handelt, deren Bewertung durch den Agenten jedoch großen Einfluss auf die zu erwartende Modellqualität besitzen wird. Bei solchen konkurrierenden Ansätzen muss ein Gewichtungs- bzw. Aggregationsmodell entworfen werden, wenn beide Ansätze in einem Gesamtrahmen integriert werden sollen, was ein wesentliches Ziel dieses Kapitels sein wird.

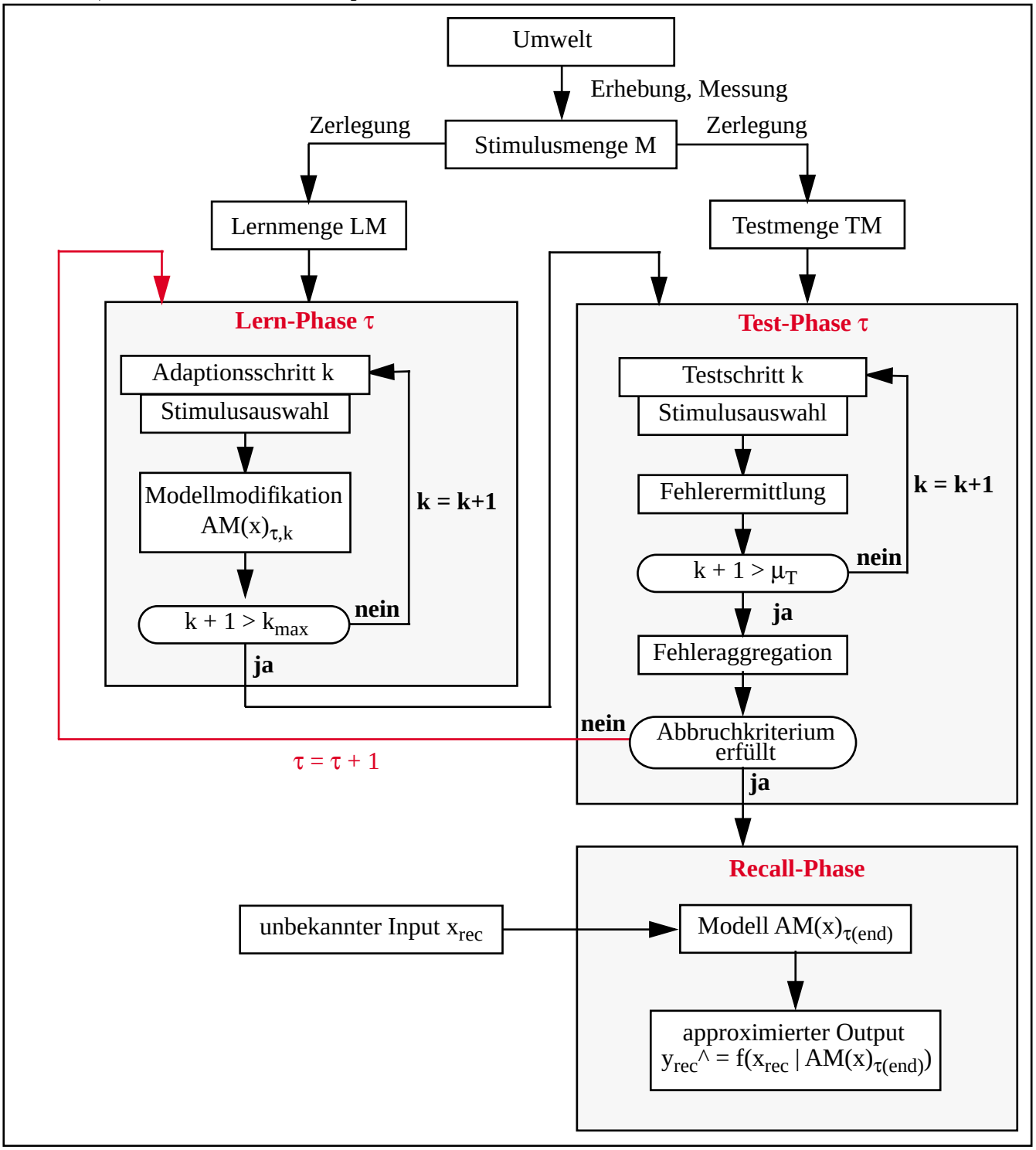
Im weiteren soll zunächst eine allgemeine, detaillierte Darstellung der Situation des passiven Lernens gegeben werden, von der aus drei Grundsituationen des aktiven Lernens beschrieben werden:
1) Aktives Lernen bei einer geschlossenen Stimulusmenge.
2) Aktives Lernen bei einem Stimulus-Strom (Query-Filtering).
3) Aktives Lernen bei einer offenen Stimulusmege (aktive Datenerhebung).

5.1.1) Passives Lernen

Die Standard-Stituation des passiven, überwachten Lernens geht von einer konstanten Menge M mit μ Stimuli m = (x, f(x)) aus, die z.B. zufällig aus einer Umgebung gewonnen wird, bzw. die einer vorhandenen Datensammlung entspricht. Soll der Modellaufbau durch ein Qualitätsmaß geregelt werden, so wird die Stimulusmenge M in eine Lernmenge LM mit μ_L Elementen und eine Testmenge TM mit μ_T Elementen disjunkt zerlegt, wobei der Lernprozess abwechselnd Lern- und Testphasen durchläuft. In der Lernphase τ wird das Modell entsprechend dem verwendeten Lernalgorithmus modifiziert, d.h. es wird in einem Lernschritt k ein Stimulus aus der Lernmenge ausgewählt, der auf der Basis eines Adaptions-Algorithmus das Modell von einem Zustand $AM(x)_{\tau,k-1}$ in einen Zustand $AM(x)_{\tau,k}$ überführt. Nach einer gegebenen Anzahl von k_{max} Adaptionsschritten, wird die τ'te Lernphase beendet, und die τ'te Testphase wird begonnen, in der für alle μ_T Elemente der Testmenge ein Fehlerwert auf der Basis des Modells $AM(x)_{\tau,k(max)}$ ermittelt wird, die zu einer Gesamtbewertung des Modells aggregiert werden. Mit dieser Gesamtbewertung wird das Abbruchkriterium des Gesamtverfahrens geprüft. Ist das Abbruchkriterium

erfüllt, dann wird das Lernen beendet, und die Recall- oder Anwendungsphase wird begonnen, bei der potentiell unbekannte Inputvektoren x_{rec} dem gelernten Modell $AM(x)_{\tau(end)}$ präsentiert werden, das eine Approximation $y_{rec}\hat{} = f(x_{rec} \mid AM(x)_{\tau(end)})$ des richtigen, aber unbekannten Outputs liefert. Ist das Kriterium nicht erfüllt, so wird die nächste Lern- und Testphase $\tau + 1$ begonnen.

Abb. 130) Standard-Stituation des passiven, überwachten Lernens



5.1.2) Aktives Lernen bei einer geschlossenen Stimulusmenge

Bei diesem Szenario ist eine Stimulusmenge M mit μ Elementen $m_j = (x_j, f(x_j))$ gegeben, die nicht erweitert werden kann. Der aktive Lerner hat hier die Auswahl, beginnend mit einer kleinen zufälligen Lernmenge $LM^{t=0}$ mit $\mu_L{}^{t=0}$ Elementen, sukzessive weitere Lernstimuli auszuwählen, und in seine Lernmenge zu integrieren. Es wird das Ziel verfolgt, eine geforderte Modellqualität mit einer so gering wie möglichen Stimulusanzahl zu erreichen.

Ansätze dieser Art werden auch als Active Data Subset Selection bezeichnet, wobei sie in Domänen eine Bedeutung besitzen, in denen ausreichende Lerndaten vorhanden sind, jedoch Bereiche der Domäne irrelevant oder irreführend sind, sodass die Verwendung aller verfügbaren Daten eine geringere Modellqualität ergeben würde. In Domänen, in denen ein Überangebot von Lerndaten existiert, die ebenfalls nicht nach ihrer Wichtigkeit klassifiziert sind, besitzt dieses Szenario ebenfalls eine Bedeutung, da das Lernen sehr großer Datenmengen allgemein Effizienzprobleme erzeugt.

Die Stimulusmenge M wird nicht zu Beginn des Lernprozesses irreversibel in LM und TM zerlegt, sondern es sind statische wie dynamische Festlegungen der Testmenge möglich. Sei LM^t die Lernmenge in der t'ten Iteration von Lern- und Testphase mit $\mu_L + t$ Elementen, so könnte die Testmenge TM^t dynamisch aus $M \setminus LM^t$ durch μ_T'faches zufälliges Ziehen ohne Zurücklegen ermittelt werden. Bei der nicht-konstanten Anzahl der Elemente der Testmenge könnte die jeweilige Restmenge $M \setminus LM^t$ als Testmenge verwendet werden, oder die Anzahl der Elemente in der Testmenge kann steigen, z.B. $\mu_T{}^t = \mu_T{}^{t-1} + 1$. Eine nicht-zufällige Auswahl von Teststimuli aus $M \setminus LM^t$ kann ebenfalls durch aktive Auswahlverfahren durchgeführt werden, was im weiteren jedoch nicht weiter verfolgt werden soll.

Wird die Testmenge aus der Restmenge $M \setminus LM^t$ gebildet, so kann sich im Verlauf des Verfahrens das Problem zu kleiner Testmengen, und somit unsicherer Qualitätsmaße ergeben. Im Extremfall kann keine Testphase durchgeführt werden, wenn in der Lernmenge alle Stimuli aus M aufgenommen wurden, d.h. wenn $M \setminus LM^t = \varnothing$ wird. Die Wahrscheinlichkeit, dass sich eine solche Situation im Rahmen des aktiven Lernens ergibt, ist jedoch gering, da aktives Lernen mit einer geschlossenen Stimulusmenge gerade voraussetzt, dass μ so groß ist, dass eine aktive Auswahl von Stimuli, die zum Lernen verwendet werden, sinnvoll wird. Andererseits wurde von Amari et al. (1996a[6], b[7]) gezeigt, dass die Testmenge nur wenige Prozent der Gesamtstimuli enthalten muss, um effektive Aussagen über die Modellqualität zu treffen. Es bleiben somit folgende prinzipielle Vorgehensweisen, um die Testmenge festzulegen:

1) $TM := TM^t := M, \forall t$.
2) TM wird zu Beginn des aktiven Lernens durch zufälliges Ziehen ohne Zurücklegen aus M ermittelt, und bleibt unabhängig von der sich verändernden Lernmenge für alle Iterationen des aktiven Lernens konstant.
3) TM^t wird für jede Iteration neu durch zufälliges Ziehen ohne Zurücklegen aus M ermittelt, was jedoch dazu führt, dass Qualitätsmaße aus unterschiedlichen Iterationen des aktiven Lernens nicht mehr direkt vergleichbar sind, da unterschiedliche Testmengen verwendet werden.

Im weiteren soll von der zweiten Variante ausgegangen werden, d.h. TM wird zu Beginn des aktiven Lernens erzeugt, und bleibt konstant während aller Iterationen, wobei die Ermittlung der Modellqualität

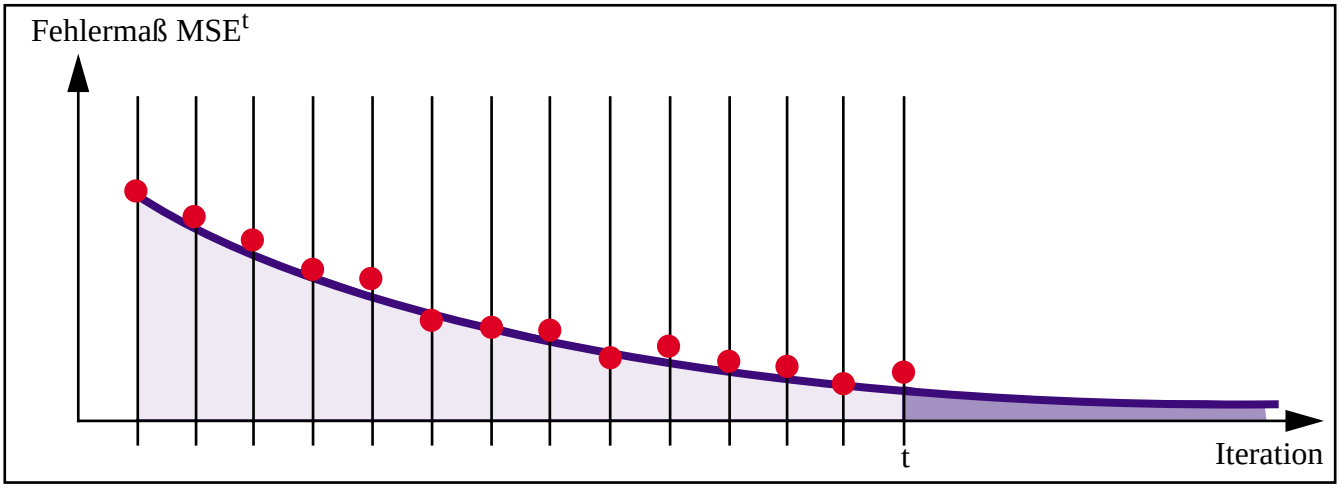
durch ein Gewinnerlisten-Verfahren durchgeführt werden soll. Für die Stimuli, die in TM und in einer momentanen Lernmenge LM^t liegen, wird ein LWR-Verfahren angewendet, das eine Schätzung f(x)^ z.B. ohne das erste Element in der Gewinner-Liste der Stützpunkte des Modells $AM(x)^t$ durchführt (siehe Abschnitt 2.3.8)).

Mit Hilfe der Lernmenge LM^t ist ein instanzbasiertes Modell $AM(x)^t = AM(x \mid LM^t)$ spezifiziert, wenn ein Approximationsverfahren vorliegt. Soll ein prototypbasiertes Modell verwendet werden, so muss dieses aus der Lernmenge durch Adaptionsoperationen erzeugt werden. Allgemein entspricht dies der Situation des passiven Lernens, d.h. eine Iteration des aktiven Lernens kann als eine Iteration bzw. mehrere Iterationen eines passiven Lernens interpretiert werden. Bei mehreren Iterationen wird jeweils eine Lern- und eine Testphase auf der Basis der temporär konstanten Lernmenge LM^t durchgeführt. Ein qualitätsbedingtes Abbruchkriterium kann eine Iteration des aktiven Lernens beenden, wobei ein Qualitätsmaß Anwendung finden kann, das im Verlauf des aktiven Lernens verschärft wird. Erfolgt ein Abbruch der Iteration, wenn ein Fehlermaß des Modells kleiner-gleich einem Schwellenwert ist, so kann das Abbruchkriterium des Modells $AM(x)^t$ auf der Basis von LM^t mit einem kleineren Fehlerschwellenwert geprüft werden, als $AM(x)^{t-1}$ auf der Basis von LM^{t-1}. Erfolgt auf der Basis von TM kein Abbruch des Gesamtverfahrens, so wird die nächste Iteration des aktiven Lernens durchgeführt, bei der aus der Restmenge $M \setminus LM^t$ ein oder mehrere Stimuli ausgewählt, und in die Lernmenge verschoben werden, die zu LM^{t+1} aktualisiert wird. Es folgt wiederum eine Phase des passiven Lernens, gefolgt von der Testphase t+1 auf der Basis von TM, bei der die Modellqualität ermittelt wird. Mit dem Qualitätsmaß wird entschieden, ob eine weitere Phase des passiven Lernens durchgeführt werden muss.

Vereinfachend soll von genau einer Phase passiven Lernens innerhalb einer Iteration des aktiven Lernens ausgegangen werden, sodass die Ermittlung der Modellqualität nicht für den Zweck verwendet wird, zu entscheiden, ob eine weitere Phase passiven Lernens durchgeführt werden soll. Der Abbruch des gesamten aktiven Lernens soll nicht auf der Basis eines einzelnen Modellzustandes und seinem Qualitätsmaß entschieden werden. Eine Strategie der Abbruchprüfung könnte demgegenüber die Sequenz der Modelle $AM(x)^t$, t = 1, ..., und der damit verbundenen Qualitätsmaße analysieren. Ein Abbruch könnte erfolgen, wenn der erwartete Zuwachs an Modellqualität eine bestimmte Relation zu dem Aufwand besitzt, der benötigt wird, um diesen Zuwachs zu erreichen, d.h. wenn erwartet wird, dass nur noch kleine Qualitätsverbesserungen in den weiteren Iterationen t + 1, t + 2, ... erreicht werden können.

Sei MSE^t das verwendete Fehlermaß für $AM(x)^t$, so kann die Sequenz $MSE^{1->t}$ der MSE-Werte im Verlauf der Iterationen des aktiven Lernens gebildet werden. Die daraus abgeleiteten Wertepaare (MSE^1, 1), (MSE^2, 2), ..., (MSE^t , t) können verwendet werden, um eine Regressionsgleichung zu bilden, wobei die Form der Regressionsgleichung monoton fallend sein soll, sowie asymptotisch von oben gegen MSE = 0 fallen soll (siehe Abb. 131)). Das uneigentliche Integral in den Grenzen]t, -∞[ist ein Maß für die mögliche, zukünftige Verbesserung durch aktives Lernen, und kann somit für das Abbruchkriterium herangezogen werden. Beispielsweise kann der Quotient des Integralwertes in den Grenzen [1, t] und in den Grenzen]t, -∞[gebildet werden, wobei ein Abbruch erfolgt, wenn der Quotient einen bestimmten Schwellenwert wie 0,05 unterschreitet. Bei der asymptotischen Vorgehensweise ist jedoch zu bedenken, dass in der Grundmenge M der Stimuli nur eine endliche und nicht erweiterbare Anzahl von Stimuli liegen, sodass ein Verfahrensabbruch in jedem Fall nach μ Iterationsschritten erfolgen muss.

Abb. 131) Abbruch des aktiven Lernens durch erwartete Modellverbesserung

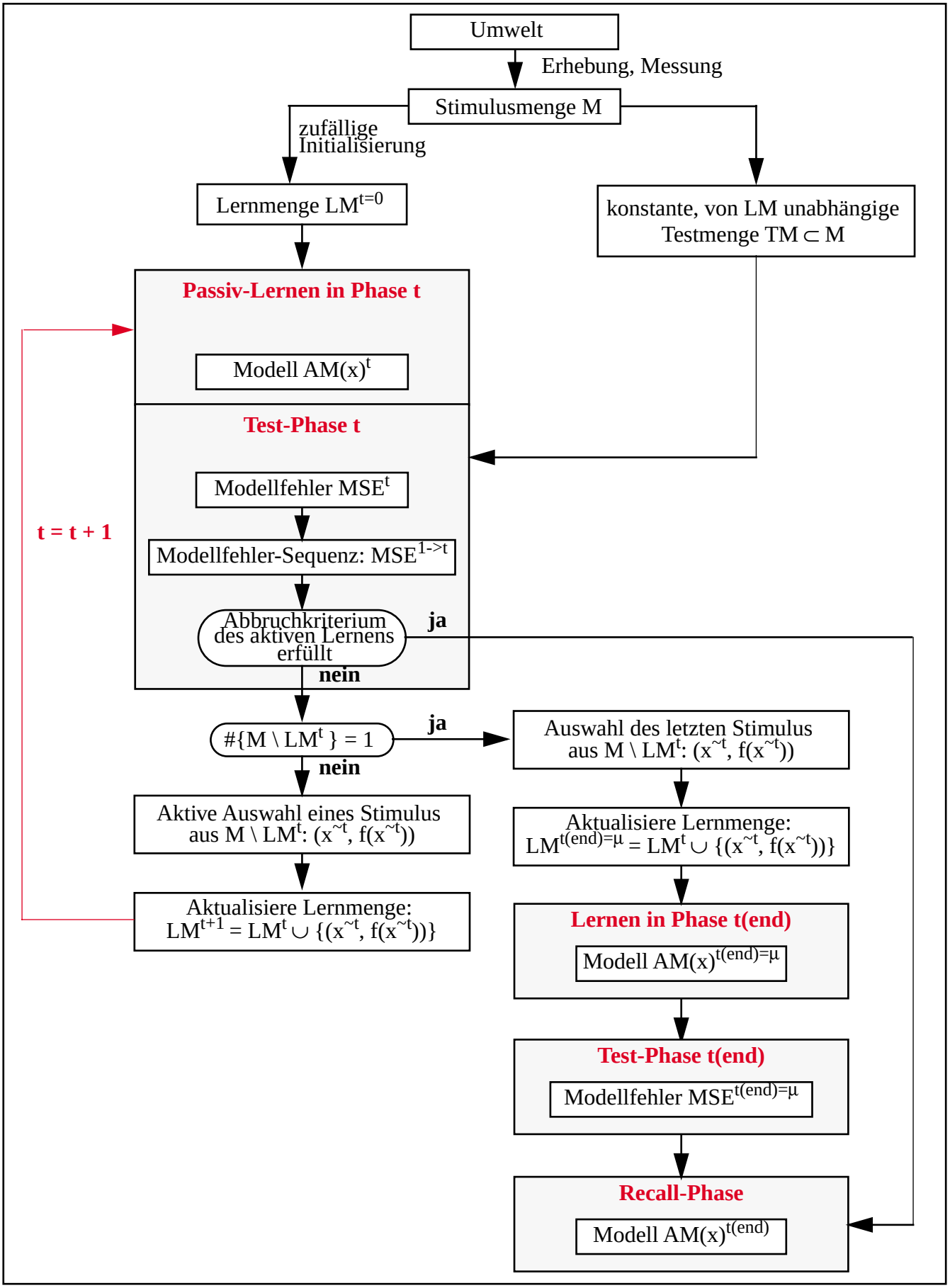


Das aktive Lernen mit geschlossener Stimulusmenge wird in Abb. 132) nochmals dargestellt, mit M als der Stimulusgrundmenge, die μ Elemente enthalten soll, wobei μ als sehr groß betrachtet wird. M wurde durch Erhebung bzw. Messung aus der Umwelt ermittelt, und ist entsprechend der hier betrachteten Variante des aktiven Lernens nicht erweiterbar, d.h. es besteht nicht die Möglichkeit zu einem x-Vektor einen richtigen Output f(x) zu ermitteln. Das aktive Lernen beginnt mit der zufälligen Initialisierung einer Lernmenge $LM^{t=0}$ mit $\mu_L{}^{t=0} << \mu$ Elementen. Mit Hilfe dieser Lernmenge wird in einer Phase des passiven Lernenes, d.h. mit einer konstanten, unveränderlichen Lernmenge, ein Modell $AM(x)^t$ erzeugt, dessen Qualität in der nachfolgenden Testphase mit MSE^t bestimmt wird. Grundlage für den Test ist die konstante Testmenge TM, die als Teilmenge von M durch zufälliges Ziehen ohne Zurücklegen von $\mu_T << \mu$ Elementen gewonnen wird, und während des gesamten aktiven Lernens konstant bleiben soll. Es soll angenommen werden, dass genau eine Lernphase pro Iterationsschritt des aktiven Lernens durchgeführt wird.

Der Modellfehler MSE^t wird in die geordnete Liste der Modellfehler $MSE^{1->t}$ eingeordnet, und es wird mit einem nicht weiter zu spezifizierenden Verfahren geprüft, ob das aktive Lernen beendet, und die Recall-Phase begonnen werden soll. Ist dies nicht der Fall, und sind in der Restmenge $M \setminus LM^t$ noch mehr als ein Stimulus vorhanden, so wird aus $M \setminus LM^t$ ein Stimulus $(x^{\sim t}, f(x^{\sim t}))$ durch ein Selektionsverfahren des aktiven Lernens ausgewählt, und in die Lernmenge verschoben, die zu LM^{t+1} wird. Beispiele dieser Selektionsverfahren wie „Query by dissagreement“ werden weiter unten beschrieben (siehe Abschnitt 5.2.1)), und sollen daher hier unspezifiziert bleiben. Die aktualisierte Lernmenge LM^{t+1} wird in der nächsten Iteration t + 1 des aktiven Lernens verwendet, um das Modell $AM(x)^t$ zu aktualisieren, das somit zu $AM(x)^{t+1}$ wird.

Sollte in $M \setminus LM^t$ genau ein Element vorhanden sein, so hat sich die Strategie des aktiven Lernens quasi als Fehlschlag erwiesen. Hätte man anstatt des aktiven Lernens ein passives Lernen auf der Basis der gesamten Stimulusmenge M durchgeführt, so hätte man den Mehraufwand des aktiven Lernens, der durch das Selektionsverfahren des jeweils nächsten Stimulus $(x^{\sim t}, f(x^{\sim t}))$, t = 1, ..., bestimmt ist, nicht aufbringen müssen. Sollte der unwahrscheinliche Fall eintreten, dass in $M \setminus LM^t$ noch genau ein Element vorliegt, so wird dieser Stimulus direkt in die Lernmenge aufgenommen, die zu $LM^{t(end)=\mu}$ wird, d.h. in der Lernmenge liegen alle μ Elemente aus M vor.

Abb. 132) Aktives Lernen bei geschlossener Stimulusmenge



Es wird im nächsten Schritt die letzte Lernphase begonnen, an deren Ende das Modell $AM(x)^{t(end)=\mu}$ steht, gefolgt von der letzten Testphase mit Hilfe von TM, aus der $MSE^{t(end)=\mu}$ ermittelt wird. Dieser Modellfehler ist wichtig für die nachfolgende Recall-Phase, da aus ihm die Unsicherheit der Approximationen der Recall-Stimuli x_{rec} abgeleitet wird, für die kein richtiger Output $f(x_{rec})$ bekannt ist.

Im Kontext einer IRS würde eine geschlossene Stimulusmenge bedeuten, dass zu jedem Dokument in der gegebenen Dokumentation eine Relevanzbewertung durch einen momentanen Agenten vorliegt, und dass nach einer geeigneten Teilmenge von Stimuli gesucht wird, mit der ein Relevanz-Approximationsmodell aufgebaut werden soll. Diese Situation entspricht jedoch nicht derjenigen einer IRS-Agent-Interaktion, da die Bewertungen von Dokumenten nicht bekannt sind, bzw. im Verlauf der Interaktion eine wachsende Teilmenge von Dokumenten eine Relevanzbewertung zugeordnet wird, wobei die Anzahl dieser Teilmenge sehr klein im Vergleich zur gegebenen Dokumentation ist. Aus diesem Grunde wird das aktive Lernen mit geschlossener Stimulusmenge als nicht adäquat im IRS-Kontext betrachtet und nicht weiter verfolgt.

5.1.3) Aktives Lernen bei einem Stimulusstrom

In diesem Szenario steht dem aktiven Lerner ein Datenstrom zur Verfügung, der ihn mit Stimuli versorgt, und dessen Quelle vom Lerner nicht beeinflusst werden kann. Die fehlende Beeinflussbarkeit entspricht der fehlenden Erweiterbarkeit im Fall der geschlossenen Stimulusmenge, und unterscheidet diese beiden Formen des aktiven Lernens von der erweiterbaren Stimulusmenge.

Die Entscheidung bezüglich der Lernstimuli, die der Lerner beim Stimulusstrom treffen kann, ist die aktive Selektion zum Lernen bzw. das aktive Verwerfen von Stimuli. Die aktiven Strategien zeichnen sich dadurch aus, dass die einzelnen Stimuli auf ihre Eignung als Lernstimulus mit Hilfe des momentanen Modells AM geprüft werden, wohingegen passive Strategien des Akzeptierens oder Verwerfens, pauschal z.B. alle Stimuli mit geradem Zeit-Index akzeptieren, und alle ungeraden verwerfen, d.h. es findet keine Prüfung der Eignung bezüglich des Modellaufbaus statt. Diese Situation besitzt strukturelle Analogien zu einem Information Filter System, da in beiden Fällen ein Datenstrom mit Dokumenten bei IFS bzw. Stimuli bei dieser Form des aktiven Lernens vorliegt, aus dem einzelne Elemente aufgenommen und die restlichen verworfen werden.

Bei einem Stimulusstrom existiert keine Stimulusmenge M, auf die Selektionsoperationen zugreifen können, und somit auch keine Möglichkeit der a priori Zerlegung in eine Lern- und Testmenge. Andererseits können die akzeptierten Stimuli im Lerner gesammelt werden, der auf diese Weise eine Lernmenge aufbaut. Durch die Verwendung eines Gewinnerlisten-Verfahrens kann die Testmengenproblematik gelöst werden (siehe Abschnitt 2.3.8)) , sodass die Forderung einer separaten Testmenge, die unabhängig vom Stimulusstrom vorliegt bzw. ebenfalls aus diesem generiert wird, unberücksichtigt bleiben kann.

Ein vorliegender Stimulus wird zunächst mit Hilfe des momentanen Modells $AM(x)^t$ geprüft, ob er als Lern-Stimulus geeignet erscheint. Wird er akzeptiert, dann wird der Stimulus darauf geprüft, ob er in der internen Lernmenge des Lerners bereits vorhanden ist, da der mangelnde Einfluss auf den Stimulusstrom

sich auch dadurch äussert, dass mehrfach vorliegende Stimuli oder periodische Präsentationen nicht ausgeschlossen werden können. Wird der akzeptierte Stimulus nicht in der Lernmenge gefunden, so wird er als neuer Lernstimulus in die Lernmenge aufgenommen, die zu LM^t aktualisiert wird. Es folgt eine Lernphase, bei der das Modell $AM(x)^{t-1}$ unter Verwendung des gegebenen Lernalgorithmus und der aktualisierten Lernmenge LM^t zu $AM(x)^t$ modifiziert wird.

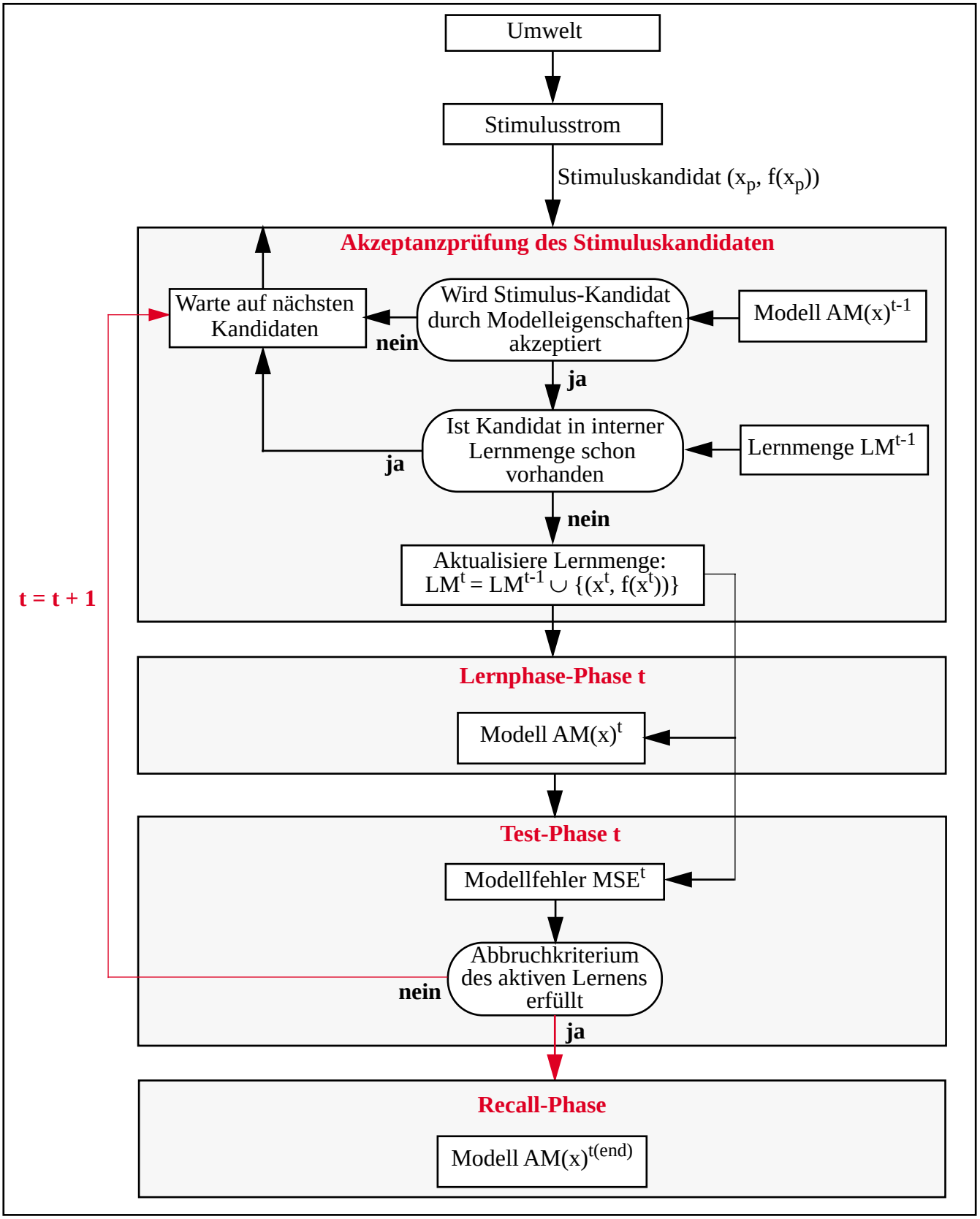
Nachdem eine bestimmte Anzahl von Lern-Stimuli akzeptiert und gelernt wurden, wird eine Testphase auf der Basis von $TM^t := LM^t$ und einem Gewinnerlisten-Verfahren eingeleitet., in der die momentane Modellqualität geschätzt wird. In der nachstehenden Abbildung Abb. 133) wird davon ausgegangen, dass nach jeder Aktualisierung der Lernmenge durch einen Stimulus und der nachfolgenden Modifikation des Modells eine Testphase durchgeführt wird.

Während der Testphase müssen alle Stimuli, die von dem Stimulusstrom präsentiert werden, verworfen werden, wenn kein Pufferspeicher in der Architektur des Lerners oder außerhalb integriert ist. Ein Test und Akzeptanzprüfungen, sowie möglicherweise weitere Lernoperationen sind nicht gleichzeitig durchführbar, da das Modell durch Lernen modifiziert wird, und der zuletzt erzeugte Test dadurch irrelevant wird. Erfolgt andererseits durch den Test der Abbruch des aktiven Lernens, so sind die Akzeptanzprüfungen und die Lernoperationen irrelevant. Da der Lerner keinen Einfluss auf die Quelle des Stimulusstroms besitzt, kann er diese somit nicht während der Testphase deaktivieren, sodass bei einem fehlenden Pufferspeicher nur die Ablehnung aller während der Testphase präsentierten Stimuli verbleibt, d.h. der Lerner wird während der Testphase sensorisch blind. Auf Möglichkeiten des Abbruchs der parallelen Verarbeitung von Akzeptanzprüfungen und der Testphase, sowie des frühzeitigen Abbruchs der Testphase aufgrund von neuen Stimuli soll hier jedoch nicht eingegangen werden.

In der Testphase wird eine Modellqualität wie $MSE^t := MSE(AM(x)^t \mid TM^t)$ ermittelt, und ein Abbruchkriterium geprüft, das entweder auf dieser Modellqualität oder auf der Sequenz $MSE^{1->t}$ der Modellqualitäten im Verlauf der Modellbildung basiert. Wird das Abbruchkriterium nicht erfüllt, so wird die Deaktivierung des Stimulus-Selektions-Algorithmus aufgehoben, und Stimuli, die durch den Stimulusstrom präsentiert werden, können wieder geprüft werden, ob sie als Lernstimuli zugelassen werden. Ist demgegenüber das Abbruchkriterium erfüllt, so wird die Recall-Phase eingeleitet.

Die bislang dargestellte Architektur des Lernsystems bei einem Stimulusstrom enthält mit der Lernmenge LM einen internen Speicher für akzeptierte Stimuli. Nicht akzeptierte Stimuli werden endgültig verworfen, da keine Möglichkeit besteht, sie zu einem späteren Zeitpunkt auf der Basis des dann vorliegenden Modells zu prüfen. Die einzige Möglichkeit solche Stimuli noch einmal zu prüfen besteht darin, dass der Stimulusstrom sie zu einem späteren Zeitpunkt nochmal präsentiert, worauf der Lerner jedoch keinen Einfluss besitzt, zumal der Lerner diese Stimuli nicht als schon einmal präsentierte Stimuli identifizieren kann. Ein variabler Stimuluspuffer, der die Elemente des Stimulusstroms in einem bestimmten Zeitintervall speichert, oder ein konstanter Puffer, der eine feste Anzahl von nachfolgenden Elementen speichert, unabhängig wann sie auftreten, ist eine Erweiterung der Architektur, die eine Reihe von Varianten des aktiven Lernens mit einem Stimulusstrom ermöglichen. Das Zwischenspeichern von Stimuli, die während internen Lern- und Testphasen präsentiert werden, wurde bereits angesprochen.

Abb. 133) Aktives Lernen bei einem Stimulusstrom



Andere Strategien sind besonders sinnvoll, wenn die Stimulusquelle bestimmte Eigenschaften bezüglich der Variabilität nachfolgender Stimuli in der Sequenz besitzt. Sind nachfolgende Stimuli nicht unabhängig voneinander, so kann eine Form der Stimulus-Clusterung durchgeführt werden. Hierbei wird eine Menge nachfolgender Stimuli, die sich entsprechend der Eigenschaft der Stimulusquelle ähnlich sind, in

einem sensornahen Puffer zwischengespeichert. Aus dieser Stimulusmenge wird ein Element ausgewählt, das als Vertreter für die betreffende Stimulusmenge betrachtet wird, und das auf Akzeptanz geprüft wird. Sollte es akzeptiert werden, so wird die gesamte Stimulusmenge in dem Puffer akzeptiert, ansonsten werden alle darin enthaltenen Stimuli verworfen. Eine solche Cluster-Akzeptanzprüfung ist insbesondere dann sinnvoll, wenn der Aufwand zur Prüfung eines Stimulus hoch ist.

Es ist auch die umgekehrte Situation durchführbar, bei der ähnliche Stimuli ohne eine Akzeptanzprüfung abgelehnt werden. Der zuletzt akzeptierte Stimulus kann dabei in einem Puffer gespeichert werden, wobei ein Ähnlichkeitswert mit nachfolgenden Stimuli aus dem Strom bestimmt wird. Sollte ein nachfolgender Stimulus ein Ähnlichkeitsmaß größer als ein Schwellenwert besitzen, so wird er direkt verworfen, ohne dass eine Akzeptanzprüfung durchgeführt wird. Der erste Stimulus, der einen Ähnlichkeitswert kleiner als der Schwellenwert besitzt, wird zur Akzeptanzprüfung zugelassen, und ersetzt den Stimulus in dem Pufferspeicher, falls er akzeptiert wird. Diese Vorgehensweise ist sinnvoll, wenn angenommen werden kann, dass gleiche bzw. hinreichend ähnliche Stimuli keinen bzw. einen vernachlässigbaren Effekt auf das Modell besitzen, wenn sie gelernt werden. Stimuli dieser Art besitzen keine neuen Informationen und führen dementsprechend auch zu keinen Veränderungen in dem sie repräsentierenden Modell, sodass auf sie verzichtet werden kann.

Ein System mit genau einem Datenstrom kann als Spezialfall einer allgemeineren Klasse interpretiert werden, die über mehrere parallele Kanäle Datenströme aufnehmen und bewerten. Bei einer solchen Architektur lassen sich erweiterte aktive Lernstrategien formulieren, die davon ausgehen, dass die einzelnen Stimulusströme nicht unabhängig sind, sodass Ereignisse in einem Kanal Auswirkungen auf die Akzeptanzprüfungsstrategien der anderen Kanäle haben. Wird beispielsweise auf einem Kanal ein sehr relevanter Stimulus entdeckt, so könnte dies die Prüfungsstrategien in den anderen Kanälen ändern, da erwartet wird, dass dort in Kürze ebenfalls wichtige Stimuli eintreffen werden. Aktive Lerner dieser Art sind typisch für biologische Systeme, die über mehrere unterschiedliche Sensorentypen verfügen und diese zu einem Gesamtbild integrieren (Merging of the senses; Stein & Meredith (1993[323])). Im Kontext von IRS und IFS würde dies eine Rolle spielen, wenn die Dokumente aus unterschiedlichen, nicht unkorrelierten Quellen stammen. In der Regel wird jedoch auf eine Unterscheidung von Dokumentquellen verzichtet, sodass alle Dokumente behandelt werden können, als wenn sie aus einer Quelle stammen. Aus diesem Grunde soll auf eine detailliertere Darstellung von aktivem Lernen mit mehreren Kanälen verzichtet werden.

Eine alternative Verwendung eines Datenstroms wird in Cohn et al. (1990[70]) und Freund et al. (1993[123]) mit dem „Query-Filtering“ vorgeschlagen. Es wird dabei kein natürlicher Datenstrom verwendet, sondern ein Zufallsprozess wählt mögliche Inputvektoren x aus der entsprechenden Domäne aus, und erzeugt somit einen künstlichen Datenstrom. Bei einem überwachten Lernen fehlt die Outputkomponente y, die von der Umgebung geliefert werden muss, d.h. der Lerner besitzt nun einen Einfluss auf externe Instanzen. Die sich ergebenden Stimuli (x, y) werden als Lern-Stimuli in LM aufgenommen, mit der ein internes Modell AM(x) aufgebaut bzw. modifiziert wird. Ausgangspunkt des Query-Filterings war die Erkenntnis, dass eine Form der Inputvektoren-Konstruktion (Membership-Query) nicht dazu führt, dass bei bestimmten Problemarten weniger Lern-Stimuli verwendet werden, als bei einer Zufallspräsentation. Dies führt zu dem Ergebnis, dass die zufällige Auswahl in Verbindung mit einem

Selektions-Algorithmus, der Akzeptanzprüfungen durchführt, Informationen über die zukünftigen Stimuli liefert (Freund et al. (1993: 484[123])). Diese Aussage gilt jedoch nicht für alle Formen konstruierter Inputvektoren, insbesondere für solche nicht, bei denen selbst Zufallsprozesse einfließen.

Das Query-Filtering ist der direkte Übergang zum dritten Szenario des aktiven Lernens, bei dem die Inputvektoren, zu denen der aktive Lerner einen zugehörigen Output wünscht, vom Lerner selbst erzeugt wird.

5.1.4) Aktives Lernen bei einer offenen Stimulusmenge

Dieses Szenario des aktiven Lernens zeichnet sich dadurch aus, dass der aktive Lerner Anfragen oder Experimente in Form eines Inputvektors entwerfen kann, die von einer externen Instanz in Form eines zugehörigen Outputvektors f(x) beantwortet wird. Ziel ist es auch hier, mit so wenigen Anfragen wie möglich ein Modell mit einem gewünschten Qualitätsmaß zu erzeugen. Die abzuwägenden Kosten sind dabei zum einen der Berechnungsaufwand zur Bestimmung der nächsten Anfrage, und zum anderen der Aufwand, den die externe Instanz aufwenden muss, um den angeforderten Output zu bestimmen.

Initialisiert wird ein solcher Lerner, indem eine Stimulusmenge M entweder zufällig aus einer Umgebung gewonnen wird, bzw. indem eine vorhandene Datensammlung als Stimulusmenge M verwendet wird. Die gesamte Stimulusmenge wird in der Initialisierungsphase als Lernmenge verwendet, um damit ein Initialisierungsmodell $AM(x)^{t=0}$ zu erstellen, das im weiteren durch das aktive Lernen verfeinert wird. Um die Probleme, die mit einer unabhängigen Testmenge verbunden sind, zu eliminieren, soll die Lernmenge in Verbindung mit einem Gewinnerlisten-Verfahren verwendet werden, um jeweils die Modellqualität zu schätzen.

Grundlage der erweiterten Stimulusmenge ist ein Verfahren, das Inputvektoren entsprechend ihrer Eignung als zukünftige Lernstimuli bewerten kann, wobei Beispiele für diese Bewertungsverfahren in den nächsten Abschnitten beschrieben werden.

1) Diskrete Vorgehensweise, bei der eine endliche Kandidatenmenge an Inputvektoren spezifiziert wird, die durch das Verfahren bewertet werden, und aus der mit Hilfe dieser Bewertung ein Inputvektor ausgewählt wird, dessen Outputvektor extern bestimmt wird.
2) Stetige Vorgehensweise, bei der durch ein Suchverfahren, das den stetigen Inputraum nutzt, eine oder mehrere Inputvektoren mit besonderen Eigenschaften festlegt. Verfahren dieser Art basieren meist auf einem Gradientenabstieg, der die erste Ableitung einer stetig differenzierbaren Bewertungsfunktion nutzt (siehe Cohn (1995[73]), Cohn et al. (1995[74])), oder es werden Verfahren genutzt, die Ableitungen höherer Ordnung nutzen.

Die diskrete Vorgehensweise ist der Grundsituation einer endlichen Dokumentmenge in einem IRS adäquat, sodass sich auf diese Möglichkeit in den weiteren Darstellungen beschränkt werden soll. Unabhängig von der diskreten oder stetigen Vorgehensweise wird ein oder mehrere Inputvektoren identifiziert, zu denen eine externe Instanz einen Outputwert oder -vektor ermittelt wird, sodass dadurch ein oder mehrere neue Stimuli erzeugt werden. Diese werden in die Lernmenge aufgenommen, gefolgt von der

Modifikation des Modells, der Schätzung der Modellqualität, sowie der Prüfung des Abbruchkriteriums des aktiven Lernens.

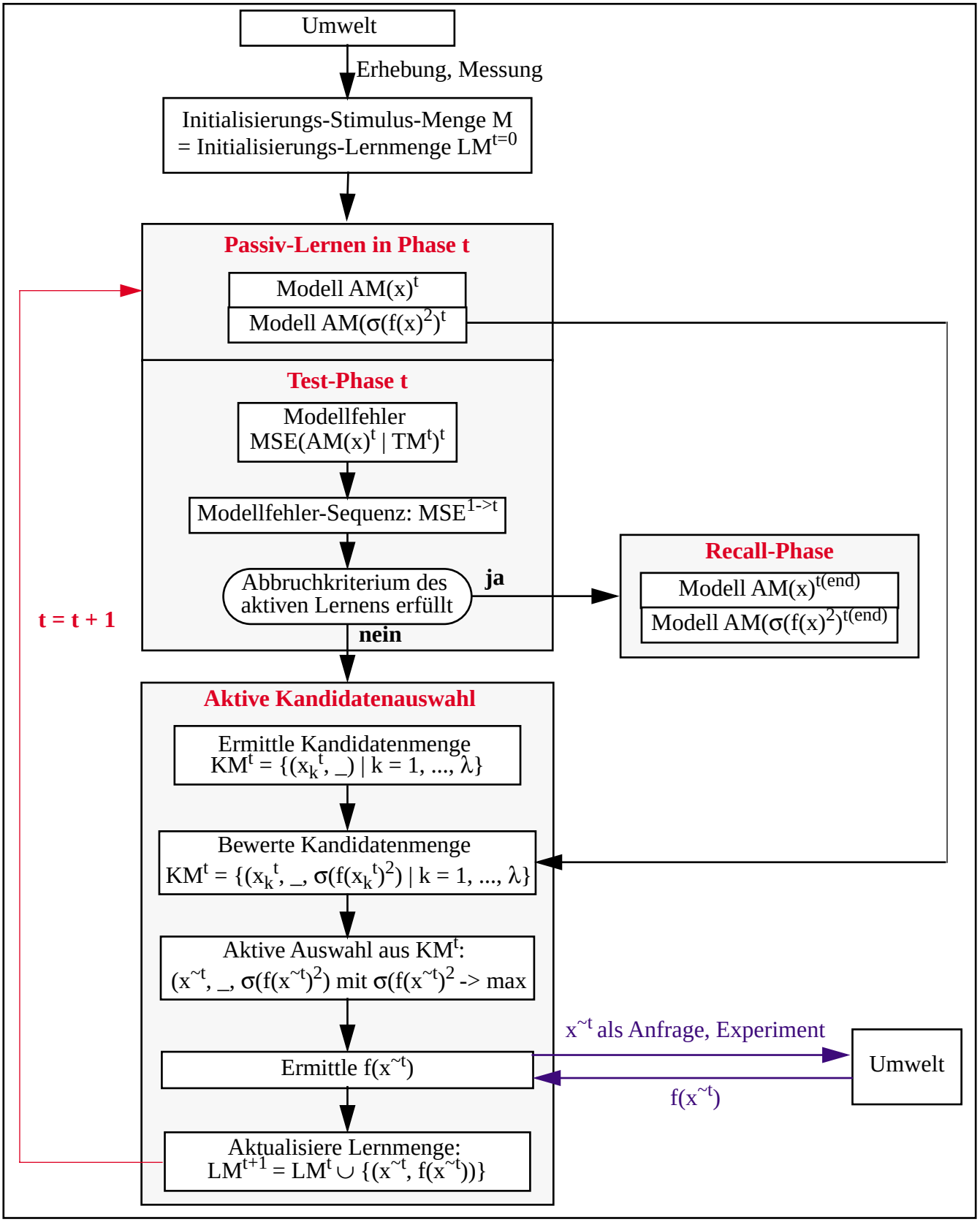
In der Abbildung Abb. 134) wird von einer Initialisierungs-Lernmenge $LM^{t=0}$ ausgegangen, die als Stimulusmenge M festgelegt wird, deren Elemente durch Erhebung oder Messung aus der Umwelt ermittelt wurden. Es folgt die Lernphase t, in der ein Output-Approximationsmodell $AM(x)^t$ auf der Basis von LM^t erzeugt wird. Es soll der Fall angedeutet werden, dass das aktive Lernen durch Schätzungen der Output-Varianz $\sigma(f(x))^2$ von Vektoren x durchgeführt wird, sodass ein Output-Varianz-Approximationsmodell $AM(\sigma(f(x))^2)^t$ benötigt wird, das diese Schätzungen erzeugt. Ein solches Modell kann aufgebaut werden, wenn einer Stützpunktverteilung richtige Outputwerte f(x) sowie Schätzungen f(x)^ durch $AM(x)^t$ zugeordnet werden.

Nachdem die beiden Modelle $AM(x)^t$ und $AM(\sigma(f(x))^2)^t$ in der Lernphase t erzeugt wurden, soll eine Testphase eingeschoben werden, in der das Abbruchkriterium des aktiven Lernens anhand einer Sequenz von Modellbewertungen geprüft wird. D.h. sinnvoll kann dieses Abbruchkriterium erst geprüft werden, wenn eine Mindestanzahl von Iterationen t des aktiven Lernens durchgeführt wurde. Bei dieser Prüfung wird zunächst die Modellqualität $MSE(AM(x)^t \mid TM^t)$ des Modells $AM(x)^t$ mit Hilfe von $TM^t := LM^t$ und einem Gewinnerlisten-Verfahren gebildet. Diese Einzelqualität wird in die geordnete Liste eingefügt, die somit zu $MSE^{1->t}$ aktualisiert wird. Es folgt die Analyse dieser Sequenz, auf deren Details hier nicht weiter eingegangen werden soll. Wird das Abbruchkriterium des aktiven Lernen bestätigt, so kann die Recall-Phase begonnen werden, ansonsten wird die Iteration t mit der Ermittlung einer Kandidatenmenge $KM^t = \{(x_k^t, \_) \mid k = 1, ..., \lambda\}$ weitergeführt.

Der nächste Schritt besteht darin, die Elemente der Kandidatenmenge für die Selektion im Rahmen des aktiven Lernens zu bewerten, indem den Inputvektoren x_k^t Output-Varianz-Schätzungen $\sigma(f(x_k^t))^{2\wedge}$ zugeordnet werden. Die aktive Auswahl aus der Kandidatenmenge KM^t erfolgt, indem das Element mit dem größten Wert der Output-Varianz-Schätzung ermittelt wird. Für dieses Element $(x^{\sim t}, \_, \sigma(f(x^{\sim t}))^{2\wedge})$ wird der richtige Outputwert $f(x^{\sim t})$ ermittelt, indem $x^{\sim t}$ als Anfrage an die externe Instanz gestellt wird, bzw. indem ein Experiment durchgeführt wird, dessen Spezifizierung durch $x^{\sim t}$ gegeben ist. Danach liegt der entsprechende Stimulus $(x^{\sim t}, f(x^{\sim t}))$ vor, der in die Lernmenge aufgenommen wird, die zu LM^{t+1} aktualisiert wird. Mit dieser neuen Lernmenge wird die nächste Iteration des aktiven Lernens begonnen, indem die Lernphase t + 1 durchgeführt wird.

Das aktive Lernen mit einer offenen Stimulusmenge besitzt einige partitielle Analogien zu Evolutionären Algorithmen in Verbindung mit einem Fitness-Approximationsmodell (siehe Bachelier (1999d[22])). Die Kandidatenmenge KM kann als Zwischenpopulation ZP betrachtet werden, deren Individuen mit Hilfe eines Modells bewertet werden. Auf der Basis dieser Bewertung wird ein Nachkomme bzw. eine Nachkommenpopulation durch sel_N ausgewählt, denen der richtige Fitnesswert f(x) durch eine externe Operation zugeordnet wird. Individuen sind somit Experimente in der Umwelt, deren Ausgang oder Bewertung nicht innerhalb des eigentlichen evolutionären Verfahrens liegt, so wie die Fitnessbewertung eines biologischen Individuums von der Umwelt vorgenommen wird, und nicht durch molekulare Eigenschaften oder Vorgänge in dem Individuum, wenn man von Phänomenen absieht, in denen eine Selbstbewertung eine Rolle spielen.

Abb. 134) Aktives Lernen bei offener Stimulusmenge



Bei der Erzeugung der Kandidaten könnten Reproduktionsoperationen Verwendung finden, bei denen die Elemente der Lernmenge als Eltern involviert sind, sodass hier ein reichhaltiges Repertoire an Verfahren anwendbar ist, mit denen eine Kandidatenmenge erzeugt werden kann.

In beiden Fällen existiert keine Obergrenze der Kosten für die Evaluation der Fitnessfunktion bzw. für die Durchführung eines Experimentes. Diese Tatsache wird bei den Untersuchungen der Evolutinären Algorithmen systematisch vernachlässigt, da die Testfunktionen, mit denen unterschiedliche EA-Varianten verglichen werden, explizit und meist in einer geschlossenen Form vorliegen. Mit zunehmenden Kosten der Evaluierung der Fitnessfunktionen tritt die Minimierung der Anzahl der erzeugten bzw. evaluierten Individuen in den Vordergrund, sodass Fitness-Approximationsmodelle und aktive Lernprozesse, mit denen diese Modelle generationsintern oder generationsübergreifend aufgebaut werden, an Bedeutung gewinnen.

Im Kontext von IRS würde eine offene Stimulusmenge in Verbindung mit der stetigen Vorgehensweise bedeuten, dass jeder Punkt im DVR vom IRS selektierbar ist. D.h. es könnte die Bewertung eines Dokumentes benötigt werden, das durch die Indexierungsfunktion $A_{IR(D)}$ auf einen Dokumentvektor abgebildet würde, welches das IRS ermittelt hat, wobei dieses Dokument jedoch nicht in der vorliegenden Dokumentation existieren muss. Im Kapitel 3) wurde bereits darauf hingewiesen, dass die Indexierungsfunktion $A_{IR(D)}$ nicht eindeutig umkehrbar ist, d.h. ein Dokument wird eindeutig auf einen Dokumentvektor abgebildet, aber ein Dokumentvektor kann nicht eindeutig einem Dokument zugeordnet werden, da mehrere Dokumente mit einer endlichen Anzahl von Zeichen auf ein Dokumentvektor abgebildet werden können. Die stetige Vorgehensweise kann als adäquat betrachtet werden, wenn zu jedem Vektor aus DVR, der durch den aktiven Lerner spezifiziert wird, der Dokumentvektor mit der kleinsten Distanz ermittelt, zu dem ein Dokument in der gegebenen Dokumentation existiert.

Die diskrete Vorgehensweise bei einer offenen Stimulusmenge kann demgegenüber direkt adäquat verwendet werden, wenn als Kandidaten ausschließlich Dokumentvektoren verwendet werden, zu denen ein korrespondierendes Dokument in der gegebenen Dokumentation existiert. Trotzdem ist eine offene Stimulusmenge in diesem Fall eine nicht vollständig adäquate Situation, da eine offene Menge impliziert, dass beliebig viele Elemente aufgenommen werden können, während eine momentane Dokumentation immer endlich ist. Dies ist jedoch unter der Grundprämisse des aktiven Lernens keine Beschränkung, da beim aktiven Lernen gefordert wird, dass der Modellaufbau so schnell und effizient wie möglich, d.h. mit so wenig wie möglichen Stimuli, durchgeführt wird.

Das aktive Lernen beim Retrieval mit Feedback-Operationen in einem IRS kann durch eine Kombination von geschlossener und offener Stimulusmenge modelliert werden. Die Eigenschaft, die von der geschlossenen Stimulusmenge übernommen wird, ist die Tatsache, dass eine endliche Menge von Dokumenten und Dokumentvektoren vorliegt, wobei die Dokumentmenge vom IRS nicht selbständig erweitert werden kann. Die Eigenschaft, die von der offenen Stimulusmenge übernommen wird, ist die Tatsache, dass zu den Dokumentvektoren keine Relevanzbewertungen durch den momentanen Agenten vorliegen, sodass es sich noch um keine vollständigen Stimuli für ein überwachtes Lernen handelt. Es können jedoch Dokumentvektoren zum aktiven Lernen ausgewählt und dem Agenten zur Bewertung präsentiert werden, was eine Eigenschaft des aktiven Lernens bei offener Stimulusmenge ist.

5.2) Indirekte und direkte Verfahren beim Modell-Maximierungskriterium

Bei einer offenen Stimulusmenge kann eine direkte und eine indirekte Klasse von Verfahren zum aktiven Lernen unterschieden werden:

1) Direkte Verfahren

   Es wird der Inputvektor gesucht, der, wenn er in die Lernmenge aufgenommen wird, zu der größten, erwarteten Verbesserung des Modells führt. Dabei werden Kandidaten zur Probe in die Lernmenge aufgenommen, gefolgt von einer Modelladaption und der Bestimmung der Modellqualität. Faktisch übernommen wird der Inputvektor, der zu dem Modell mit der besten geschätzten Qualität korrespondiert.

2) Indirekte Verfahren

   Es wird der Inputvektor gesucht, der die größte Fehler- bzw. Unsicherheitsschätzung besitzt, wobei angenommen wird, dass die Aufnahme des zugehörigen Stimulus in die Lernmenge zu einer großen Verbesserung des Modells führt.

Indirekte Verfahren erhalten ihre Bezeichnung dadurch, dass sie keine Modellverbesserungen für unterschiedliche Input-Alternativen vergleichen, d.h. es wird keine direkte Maximierung der Modellverbesserung durchgeführt, sondern eine indirekte Optimierung, die annimmt, dass die Integration von Inputvektoren, die dem momentanen Modell die größten Probleme bereiten, zu der größten Modellverbesserung im Verlauf des aktiven Lernens führen wird. Operationalisiert werden kann dies durch Bias- und Varianz-Schätzungen.

Direkte Verfahren gliedern sich in eine Lernphase und eine Testphase. In der Lernphase wird für jeden Kandidaten $(x_k^t, \_)$ eine eigene Lernmenge LM_k^t gebildet, gefolgt von dem Aufbau des Modells $AM(x)_k^t$ entweder on-scratch oder durch Adaption des vorliegenden Ausgangsmodells $AM(x)^t$ in der Iteration t+1, wobei im weiteren von einer Nachadaption ausgegangen werden soll. Bei einem überwachten Lernen wird eine Schätzung $f(x_k^t)\hat{}$ des Outputs an der Stelle x_k^t benötigt, um einen formal vollständigen Stimulus $(x_k^t, f(x_k^t)\hat{})$ zu erhalten, mit dem das Nachlernen des Modells zu $AM(x)_k^t$ durchgeführt wird, was jedoch nur einen sehr geringen Anteil am Gesamtaufwand erfordert. Bei einem instanzbasierten Approximationsmodell entfällt eine solche Lernphase vollständig, da mit der Festlegung einer Stützpunktmenge LM_k^t und einem Approximationsverfahren das Modell vollständig spezifiziert ist. Ein prototypbasiertes Verfahren erfordert eine Nachadaption von $AM(x)^t$ zu $AM(x)_k^t$, wobei die Verwendung einer SC-GNG-SOM besondere Vorteile bietet, da eine Nachadaption für eine monoton gewachsene Lernmenge problemlos und effizient durchgeführt werden kann (Bachelier (1998c[17])). Hierzu wird der Inputvektor x_k^t der SC-GNG-SOM genau einmal präsentiert, gefolgt von der Bestimmung des Gewinner-Neurons $n_{s(1|k)}$ und der Aufnahme des Stimulus $m_k^t = (x_k^t, f(x_k^t)\hat{})$ in die lokale Lernmenge $M_{s(1|k)}$ des Gewinner-Neurons. Werden streng lokale Operationen durchgeführt, so wird danach der Gewichtsvektor $w(x)_{s(1|k)}$ des Gewinner-Neurons auf der Basis der aktualisierten, lokalen Lernmenge adaptiert, gefolgt von der Neuschätzung der Relevanz an der Stelle des adaptierten Gewichtsvektors. Eine vereinfachte Vorgehensweise wäre nach der Relevanzschätzung beendet, während eine effektivere Vorgehensweise vorab prüft, ob eine lokale Umverteilung der Stimuli in den lokalen Lernmengen des Gewinner-Neurons und seiner verbundenen Nachbarn durchgeführt werden kann, da sich die Grenzen der Voronoi-Regionen durch die Adaption des Gewichtsvektors $w(x)_{s(1|k)}$ verändert haben.

Gegenüber der Lernphase erfordert die Testphase, in der die Modellqualität von $AM(x)_k^t$ geschätzt wird, einen wesentlich größeren Aufwand, wobei ein Extremfall die Verwendung von Eigenschafts-Integralen im Kontext des Optimal Experiment Design darstellt (Cohn (1995[73]), Cohn et al. (1995[74])). In Cohn (1995[73]) wird ein Bias-Integral und in Cohn et al. (1995[74]) wird ein Output-Varianz-Integral über den gesamten Inputraum X durch eine numerische Monte-Carlo-Approximation geschätzt, was eine Vielzahl von Stützpunkten in X benötigt, für die jeweils eine Biasschätzung, oder eine Output-Varianz-Schätzung, bzw. beide Schätzungen bestimmt werden müssen. Wird bei einem indirekten Verfahren beispielsweise nach der Bias-Maximierung für einen Kandidaten (x_k^t, _, $rel(x_k^t)$^) genau eine Bias-Schätzung $bias(x_k^t)$^ notwendig, so müssen bei dem direkten Verfahren mit der Minimierung des Bias-Integrals hunderte von Stützpunkten mit je einer Bias-Schätzung berechnet werden, aus der das Bias-Integral des Modells $AM(x)_k^t$ geschätzt wird.

Wird dies berücksichtigt, so kann der Mehraufwand des direkten Verfahrens gegenüber einem indirekten Verfahren zwei bis drei Zehnerpotenzen pro Iteration betragen. Ein Vergleich des Gesamtaufwandes beider Verfahrensarten erfordert jedoch die Berücksichtigung der Anzahl der Iterationen, wobei vermutet werden kann, dass direkte Verfahren aufgrund der expliziten Maximierung weniger Iterationen benötigen, um ein gegebenes Qualitätsniveau zu erreichen. Trotzdem kann durch die Verwendung von Eigenschafts-Integralen kaum der Fall eintreten, dass der Gesamtaufwand eines direkten Verfahrens kleiner wird als der Gesamtaufwand eines indirekten Verfahrens.

Weiterhin muss berücksichtigt werden, dass bei der Agent-IRS-Interaktion quasi Real-Time-Anforderungen bestehen, wenn der Agent bei einer Interaktion $t > 0$ nach der Bewertung der zuletzt präsentierten Dokumente auf die nächste Dokumentliste wartet. Der Gesamtaufwand ist in diesem Kontext weniger wichtig als der Aufwand einer Iteration, sodass direkte Verfahren weniger geeignet erscheinen. Trotzdem soll auf direkte Verfahren eingegangen werden, und es wird versucht, Varianten zu finden, die einen deutlich geringeren Aufwand erfordern (siehe Abschnitt 5.2.3)).

5.2.1) Indirekte Verfahren

5.2.1.1) Selektionskriterien bei indirekten Verfahren

Bei allen vorgestellten Verfahren zum Ablauf des aktiven Lernens wurde weitgehend der wichtigste Aspekt ausgeklammert, auf den in diesem Abschnitt eingegangen werden soll. Es handelt sich dabei um die Festlegung des Selektionskriteriums, mit dem entschieden werden soll, ob ein noch unvollständiger Stimulus, d.h ein Dokumentvektor im Fall eines IRS, in die Ergebnismenge einer Iteration aufgenommen werden soll.

In Kapittel 4) bis einschlisslich Abschnitt 4.5) wurde als Selektionskriterium ausschließlich die Relevanzschätzung durch das oder die vorliegenden Relevanz-Approximationsmodelle genutzt. Es wurden die Dokumente mit der größten Relevanzschätzung selektiert, sodass das Selektionskriterium als Relevanz-Maximierung bezeichnet werden kann. Im Abschnitt 4.6) wurden Fehlerschätzungen und Unsicherheitsschätzungen eingeführt, mit denen Relevanzschätzungen von Kandidaten-Dokumentvektoren korrigiert werden können. In diesem Kontext dienen Fehler und Unsicherheiten dem Ziel der Relevanz-

Maximierung, da ein Ranking gebildet wird, das auf dem Prinzip der korrigierten fallenden Relevanzschätzung basiert.

Im weiteren sollen Verfahren beschrieben werden, welche die gleichen Schätzungen wie bei der Relevanz-Maximierung verwenden, d.h. Relevanz-, Fehler- und Unsicherheitsschätzungen. Diese Schätzungen werden jedoch im Kontext von Selektionskriterien der Modell-Maximierung verwendet, wodurch ein Modellaufbau durch aktives Lernen möglich wird. Der Vorteil dieser Vorgehensweise besteht darin, dass aktives Lernen, d.h. die Modell-Maximierung, ohne einen Mehraufwand gegenüber der fehlerkorrigierten Relevanz-Maximierung durchgeführt werden kann. D.h. es werden die gleichen Kennzahlen nur im Kontext eines anderen Selektionskriteriums verwendet.

Von größerer Bedeutung ist jedoch die Möglichkeit eine fehlerkorrigierte Relevanz-Maximierung und eine Modell-Maximierung zu integrieren, ohne dass ein zusätzlicher Aufwand bezüglich der Bewertung von Kandidaten-Dokumentvektoren notwendig wird. Ein Mehraufwand gegenüber einem der beiden Einzelverfahren entsteht jedoch, da zwei Selektionskriterien zu einem Ranking aggregiert werden müssen, wenn als allgemeiner Constraint gelten soll, dass dem Agenten genau eine Liste präsentiert werden soll. Im Vergleich zu dem Aufwand der Bewertung der Kandidaten ist der Integrationsaufwand gemäß einer Meta-Strategie, welche die Integration statisch oder dynamisch im Verlauf der Iterationen regelt, jedoch klein, was im Abschnitt 5.3) detaillierter beschrieben wird.

5.2.1.2) Allgemeine Selektion durch fehlende Übereinstimmung

Hauptvertreter der Ansätze, welche die Unsicherheit von Modellen minimieren wollen, werden als „Query-By-Disagreement“ bezeichnet, wobei eine Polyrepräsentation der Approximationsmodelle vorausgesetzt wird. Ansätze, die eine Modell-Polyrepräsentation verwenden, werden auch als Query-by-Committee bezeichnet, wobei alle Query-by-disagreement-Ansätze auch Query-by-Committee-Ansätze sind. Dem gegenüber sind Query-by-Committee-Ansätze denkbar, die nicht auf fehlender Übereinstimmung basieren. Wird eine Modellmenge nur zur Schätzung von Outputwerten verwendet, indem Einzelschätzungen aggregiert werden, so spielt die fehlende Übereinstimmung der Einzelschätzungen keine Rolle, und ein entsprechendes Verfahren kann nicht als Query-by-disagreement-Ansatz interpretiert werden.

In Seung et al. (1992[308]) und Freund et al. (1993[123]) werden Query-By-Disagreement-Ansätze verwendet, d.h. eine Menge von Modellen bewerten Kandidaten-Stimuli, und es werden solche Stimuli ausgewählt, bei denen Uneinigkeit bezüglich der Bewertung beobachtet wird bzw. bei denen die Uneinigkeit am größten ist. In RayChaudhuri & Hamey (1995[274], 1996[275]) wird die Unsicherheit durch die Varianz der Outputschätzungen operationalisiert, was als Spezialfall der Verwendung von zentralen Momenten höherer Ordnung betrachtet wird.

Im weiteren soll zunächst das Verfahren der Outputvarianz-Maximierung nach RayChaudhuri & Hamey (1996[275]) allgemein ohne Bezug zu einem IRS dargestellt werden, gefolgt von der Anpassung an die Verhältnisse eines IRS.

5.2.1.3) Outputvarianz-Maximierung

Initialisiert wird das Verfahren von RayChaudhuri & Hamey (1996[275]) durch eine zufällige Menge von μ_L Inputvektoren x_j, zu denen der richtige Outputwert $f(x_j)$ bestimmt wird, sodass die Initialisierungslernmenge $LM^{t=0}$ vorliegt:

$$M^{t=0} = \{m_j^{t=0} = (x_j^{t=0}, f(x_j^{t=0})) \mid j = 1, ..., \mu_L\}. \quad (731)$$

Aus dieser Initialisierungsmenge werden δ Lernmengen durch ein Resamplingverfahren durch Ziehen mit Zurücklegen oder Leave-n-Out (Jackknife, siehe Efron & Tibshirani (1993[105])) erzeugt. Im weiteren soll ein Bootstrapverfahren unterstellt werden, d.h. es werden Lernlisten mit je μ_L Elementen erzeugt, die teilweise übereinstimmen:

$$LL_k^{t=0} = (m_{L,k,j}^{t=0} = (x_{L,k,j}^{t=0}, f(x_{T,k,j}^{t=0})) \mid j = 1, ..., \mu_L). \quad (732)$$

Mit diesen Lernmengen werden Approximationsmodelle $AM_k^{t=0}$ erzeugt, wobei RayChaudhuri & Hamey (1995[274]) als Modelltyp Feedforward-Netze verwenden. Nach dieser Lernphase folgt eine Testphase, bei der zunächst μ_T Inputvektoren zufällig ausgewählt werden, die nicht in der Lernliste vorliegen, für die jedoch kein richtiger Outputwert ermittelt wird:

$$TM^{t=0} = \{m_{T,j}^{t=0} = (x_{T,j}^{t=0}, \_) \mid j = 1, ..., \mu_T\}. \quad (733)$$

Jedes Modell $AM_k^{t=0}$ erzeugt eine Outputschätzung $f(x_{T,j}^{t=0})_k\hat{} = f(x_{T,j}^{t=0} \mid AM_k^{t=0})$ für jedes der Test-Inputvektoren $x_{T,j}^{t=0}$. Für jeden Test-Inputvektor wird der Mittelwert $\bar{f}(x_{T,j}^{t=0})$ und die Varianz $\sigma(f(x_{T,j}^{t=0})\hat{})^2$ der Outputschätzungen gebildet:

$$\sigma(f(x_{T,j}^{t=0})\hat{})^2 = 1/\delta * \sum_{k=1->\delta} (\bar{f}(x_{T,j}^{t=0}) - f(x_{T,j}^{t=0})_k\hat{})^2, \text{ mit}$$
$$\bar{f}(x_{T,j}^{t=0}) = 1/\delta * \sum_{k=1->\delta} f(x_{T,j}^{t=0})_k\hat{}. \quad (734)$$

An den Stellen mit einer hohen Varianz sind sich die Modelle uneinig über den entsprechenden Output, sodass es sinnvoll ist, an genau diesen Stellen den richtigen Outputwert zu ermitteln, und in die Lernmenge aufzunehmen. In RayChaudhuri & Hamey (1996[275]) wird bei jeder Iteration des aktiven Lernens genau ein Inputvektor festgelegt, wobei als Selektionskriterium die maximale Output-Varianz $\sigma(f(x_{T,j}^{t=0})\hat{})_{max}^2$ bezüglich der verwendeten Approximationsmodelle verwendet wird:

$$x_{max}^{t=0}: \sigma(f(x_{T,j}^{t=0})\hat{})_{max}^2 = \max\{\sigma(f(x_{T,j}^{t=0})\hat{})^2 \mid j = 1, ..., \mu_T\}. \quad (735)$$

Danach wird für $x_{max}^{t=0}$ der richtige Output extern ermittelt, sodass der Stimulus $m_{max}^{t=0} = (x_{max}^{t=0}, f(x_{max}^{t=0}))$ vollständig ist, der in die Gesamtstimulusmenge aufgenommen wird, die zu $M^{t=1}$ aktualisiert wird, d.h. vor der Aktualisierung wird die nachfolgende Iteration t=t+1 eingeleitet:

$$M^{t=1} = M^{t=0} \cup \{m_{max}^{t=0}\}. \quad (736)$$

Es folgt wiederum der Aufbau von δ Lernlisten $LL_k^{t=1}$ aus $M^{t=1}$, die unabhängig voneinander und unabhängig von den Lernlisten $LL_k^{t=0}$ gebildet werden. Durch die mangelnde Fähigkeit von Feedforward-Netzen mit veränderten Lernmengen umgehen zu können, müssen die Modelle $AM_k^{t=1}$ on-scratch neu gebildet werden, was die Vorgehensweise von RayChaudhuri & Hamey (1995[274], 1996[275]) sehr rechenintensiv macht. Nach der Bildung der Modelle $AM_k^{t=1}$ wird eine neue Testmenge $TM^{t=1}$ mit μ_T

Inputvektoren unabhängig von $TM^{t=0}$ zufällig ausgewählt, gefolgt von der Bildung der Outputschätzungen, der Outputschätzungs-Varianzen und der Auswahl des Elementes mit der größten Varianz. Als Abbruchkriterium wird ein Varianzschwellenwert S_{var} verwendet, d.h. ein Abbruch erfolgt, wenn alle Varianzen einer Testmenge kleiner als S_{var} sind.

Die Ergebnisse von RayChaudhuri & Hamey (1996[275]) zeigen, dass die Modellverbesserung mit Hilfe des Maximum-Outputvarianz-Verfahrens des aktiven Lernens mit weniger Lernstimuli erreicht werden kann, als eine reine Zufallsauswahl von Lernstimuli. Weiterhin zeigt sich, dass bei einem stochastischen externen Prozess, der die richtigen Outputwerte liefert, der Vorteil dieser Form von aktivem Lernen mit zunehmendem Rauschen des Prozesses kleiner wird, bis das aktive Lernen mehr Lernstimuli benötigt als die reine Zufallsauswahl. D.h. aktives Lernen wird mit zunehmendem Rauschen ineffektiver.

Ein Kritikpunkt des Verfahrens von RayChaudhuri & Hamey (1996[275]) basiert darauf, dass nur eine relative Outputvarianz-Maximierung durchgeführt wird, da ein Maximum auf der Basis einer zufälligen, endlichen Testmenge bestimmt wird. Eine effektive Outputvarianz-Maximierung erfordert demgegenüber die Suche nach einem Outputvarianz-Maximum im gesamten Inputraum durch ein Optimierungsverfahren. Z.B. kann ein Local-Search-Verfahren wie der Gradienten-Abstieg oder ein Hill-Climbing-Verfahren verwendet werden, um eines der nächsten lokalen Outputvarianz-Maxima zu finden. Es lassen sich für diesen Zweck auch populationsbasierte Verfahren, d.h. Evolutions-Algorithmen, einsetzen, die nicht lokal begrenzt sind.

5.2.1.4) Bias- und Varianz-Maximierung bei klassifizierten Dokumentvektoren

Der Ansatz der Outputvarianz-Maximierung soll in diesem Abschnitt an den IR-Kontext angepasst werden, d.h. es wird eine Relevanz-Varianz betrachtet, wobei gleichzeitig eine Bias-Maximierung als zweites Modell-Maximierungskriterium verwendet werden soll. Die Vorgehensweisen aus Abschnitt 4.6) lassen sich weitgehend übernehmen, wobei nur die Bildung des Rankings für die sekundäre Ergebnisliste verändert wird, indem anstatt des korrigierten Relevanz-Maximierungskriteriums jetzt die beiden Kriterien Bias- und Relevanz-Varianz-Maximierung verwendet werden, was zusätzlich eine Form von Integration der beiden Kriterien erfordert. Wie in Abschnitt 4.6) beschrieben, lässt sich zunächst die folgende Fallunterscheidung treffen:

1) Instanzbasierte Fehlermodelle
    1.1) Monorepräsentation von instanzbasierten Fehlermodellen
    1.2) Polyrepräsentation von instanzbasierten Fehlermodellen
2) Prototypbasierte Fehlermodelle
    2.1) Monorepräsentation von prototypbasierten Fehlermodellen
    2.2) Polyrepräsentation von prototypbasierten Fehlermodellen

Vereinfachend soll in diesem Abschnitt klassifizierte Dokumentvektoren betrachtet werden, die durch eine GNG-SOM klassifiziert wurden, sodass die Kombination mit einem prototypbasierten Fehlermodell naheliegend ist. Instanzbasierte Modelle werden implizit bei prototypbasierten verwendet, um Relevanzschätzungen an den Stellen der Gewichtsvektoren zu erzeugen, sodass bei der Darstellung eines prototypbasierten Modells beide Sichtweisen integriert sind.

D.h. im weiteren wird eine Dokumentvektoren-Klassifikation in Form einer globalen SC-GNG-SOM N_{DV} verwendet:

$$N_{DV} = \{n_{DV,j} = (w(x_{DV})_j, M_{DV,j}, C_{DV,j}) \mid j = 1, ..., \mu_{N,DV}\},$$
$$\bigcup_j M_{DV,j} = DV. \quad (737)$$

Aus dieser SC-GNG-SOM lässt sich ein prototypbasiertes Relevanz-Approximationsmodell direkt bilden, wenn einzelnen oder allen Neuronen ein Stützpunkt im Relevanzraum zugeordnet wird. Ein derartiges Gesamtmodell, bei der alle Neurone eine Relevanzschätzung zugeordnet bekommen, besitzt die nachfolgende Struktur, mit $M_{DV(m),j}^{t}$ als der neuronen-spezifischen Stimulusmenge, und $rel(w(x_{DV})_j^{t})\wedge$ als einer Relevanzschätzung, die durch das globale, instanzbasierte LWR-Modell $AM(rel(x) \mid M_{DV(m)}^{t})$ durchgeführt wurde:

$$N_{DV}^{t} = \{n_{DV,j}^{t} = (w(x_{DV})_j^{t}, rel(w(x_{DV})_j^{t})\wedge, M_{DV,j}^{t}, M_{DV(m),j}^{t}, C_{DV,j}^{t}) \mid j = 1, ..., \mu_{N,DV}\},$$
$$rel(w(x_{DV})_j^{t})\wedge = rel(w(x_{DV})_j^{t} \mid AM(rel(x) \mid M_{DV(m)}^{t})). \quad (738)$$

Die Verwendung eines stützpunktbasierten Approximationsmodells auf der Basis eines SC-GNG-SOM besitzt gegenüber der Vorgehensweise von RayChaudhuri & Hamey (1996[275]) den Vorteil, dass eine Nachadaption und kein vollständig neuer Aufbau der Relevanz-Approximationsmodelle in jeder Iteration durchgeführt werden kann.

Es soll eine Approximationsmodell-Polyrepräsentation verwendet werden, die durch Stimulus-Bootstrap-Verfahren erzeugt werden, d.h. aus einer streng monoton wachsenden Stimulusmenge $M_{DV(m)}^{t=0}$, $M_{DV(m)}^{t\leq 1}$, ..., werden jeweils δ Stimulus-Bootstraplisten $M_{DV(m),p}^{t}$ erzeugt, die ein SC-GNG-SOM-Modell $N_{DV,p}^{t}$ sowie in Verbindung mit einem stützpunktbasierten Approximationsverfahren ein Approximationsmodell $AM(rel(x))_p^{t}$ festlegen. Im Abschnitt 4.4.2.1.3) wurden die Verfahrensklassen „einmalige Erzeugung mit Identitätserhaltung“ und „Ableitung in jeder Iteration“ unterschieden, d.h. Stimulus-Bootstraplisten werden im ersten Fall in der Initialisierungsiteration einmal on-scratch erzeugt und in den nachfolgenden Iterationen entsprechend der wachsenden Gesamtstimulusmenge aktualisiert bzw. die Bootstraplisten werden in jeder Iteration vollständig neu gebildet. Diese beiden Verfahrensklassen können auch im betrachteten Kontext des aktiven Lernens durch eine Bias- und eine Outputvarianz-Maximierung verwendet werden, wobei im weiteren vereinfachend die zweite Verfahrensklasse betrachtet werden soll.

Die Modelle $N_{DV,p}^{t}$ können durch Nachadaption der Gewichtsvektoren und Neubestimmung der Relevanzschätzungen der Neurone mit Hilfe einer Stimulus-Bootstrapliste $M_{DV(m),p}^{t}$ erzeugt werden, oder die Gewichtsvektoren bleiben erhalten in Verbindung mit einer Neubestimmung der Relevanzschätzungen (siehe Abschnitte 4.4.2.1.3.3) und 4.4.2.1.3.4)). Die ausschließliche Neuschätzung der Relevanzwerte an der Stelle der Gewichtsvektoren der Neurone durch unterschiedliche Stimulus-Bootstraplisten ist ein rechen- wie speichereffizienterer Ansatz, und soll in diesem Abschnitt verwendet werden. Dies korrespondiert mit der oben beschriebenen „Ableitung in jeder Iteration“, da eine weiterlaufende Anpassung der Modelle über die Iterationen hinweg nur sinnvoll ist, wenn die Gewichtsvektoren adaptiert werden.

Ausgangspunkt eines Retrievals unter ausschließlicher Verwendung des Modell-Maximierungskriteriums ist wiederum ein Queryvektor $q_i^{t=0}$, der in N_{DV} das Gewinner-Neuron $n_{DV,s(1|i)}^{t=0}$ bestimmt. Die lokale Dokumentvektorenmenge $M_{DV,s(1|i)}^{t=0}$ wurde standardmäßig bei den Retrieval-Verfahren unter ausschließlicher Verwendung des Relevanz-Maximierungskriteriums im Kapitel 4) als primäre Ergebnismenge verwendet, die entsprechend der Distanzen zu $q_i^{t=0}$ als sekundäre und tertiäre Ergebnisliste geordnet wird. Dies ergibt sich dadurch, dass in der Initialisierungs-Iteration kein Relevanz-Approximationsmodell für ein anderes Sortierungskriterium zur Verfügung steht, wenn keine Modelle hinzu gezogen werden sollen, die aus der Vergangenheit des IRS stammen (siehe Abschnitt 4.5)). Im Kontext einer Relevanzvarianz-Maximierung ist zu entscheiden, ob genau ein Dokumentvektor aus der primären Ergebnismenge ausgewählt werden soll, analog der Vorgehensweise von RayChaudhuri & Hamey (1996[275]), oder ob pro Iteration mehrere Dokumentvektoren ausgewählt werden sollen. Im weiteren soll in Analogie zu den Verfahrensweisen der Relevanz-Maximierung im Kapitel 4) $M_{DV,s(1|i)}^{t=0}$ als primäre Ergebnismenge $DVM_{prim}^{t=0} = DVM(q_i^{t=0})$ gewählt werden, und alle Dokumentvektoren aus $M_{DV,s(1|i)}^{t=0}$ sollen in die sekundäre und tertiäre Ergebnisliste aufgenommen werden. Wie bei der Relevanz-Maximierung steht in t=0 kein Approximationsmodell bzw. Modell-Polyrepräsentation zur Verfügung, mit der Relevanzschätzungen für die einzelnen Dokumentvektoren und somit Relevanzvarianzen erstellt werden könnten. Als Sortierungskriterium bleibt somit wieder nur die steigende Distanz zum Queryvektor.

Nach der Bewertung der korrespondierenden Dokumente der Vektoren aus $DV_{ter}^{t=0}$ ergibt sich die erste Stimulus-Gesamtmenge $M_{DV(m)}^{t=0}$, die das instanzbasierte Relevanz-Approximationsmodell $AM(rel(x)) \mid M_{DV(m)}^{t=0})$ festlegt. Die Relevanzschätzungen an den Stellen ausgewählter Gewichtsvektoren soll in diesem Abschnitt durch alle zur Verfügung stehenden Stützpunkte durchgeführt werden, d.h. global durch die jeweiligen Modelle $AM(rel(x)) \mid M_{DV(m)}^{t})$, t = 0, 1, ... Ausgewählt werden in jeder Iteration die Neurone, in deren Voronoi-Region sich Dokumentvektoren befinden, die zu einem der Elemente in der momentan vorliegenden Stimulus-Gesamtmenge $M_{DV(m)}^{t}$ korrespondieren. In t=0 wird dem Gewichtsvektor $w(x_{DV})_{s(1|i)}$ des Gewinner-Neurons eine Schätzung zugeordnet, wobei im Fall der Erhaltung der Gewichtsvektoren ein Iterationsindex t nicht in Verbindung der Gewichtsvektoren stehen muss, sondern durch die Referenz-Stimulusmenge $M_{DV(m)}^{t=0}$ beschrieben werden kann:

$$\begin{aligned} rel(w(x_{DV})_{s(1|i)})^{t=0\wedge} &= rel(w(x_{DV})_{s(1|i)} \mid AM(rel(x) \mid M_{DV(m)}^{t=0}) \\ &= 1/\nu_{s(1|i)} \sum_j h(d_{DVR}(x_j^{t=0}, w(x_{DV})_{s(1|i)})) * rel(x_j^{t=0}), \\ \nu_{s(1|i)} &= \sum_j h(d_{DVR}(x_j^{t=0}, w(x_{DV})_{s(1|i)})), \forall m_j^{t=0} \in M_{DV(m)}^{t=0}. \end{aligned} \quad (739)$$

Die Iteration t = 0 wird als Sonderfall betrachtet, bei der zusätzlich für die Nachbarneurone aus $N(d_G=1 \mid G_{DV})_{s(1|i)}^{t=0}$ eine Relevanzschätzung bestimmt wird, obwohl in ihren Voronoi-Regionen noch keine Dokumentvektoren liegen, denen richtige Relevanzwerte durch den Agenten zugeordnet wurden, da kein prototypbasiertes Approximationsmodell mit nur einem Stützpunkt verwendet werden soll. In den nachfolgenden Iterationen werden den Nachbarneuronen keine Relevanzschätzungen mehr zugeordnet. Die Neuronenmenge $\Delta N_{DV}^{t=0}$, in der die Neurone gesammelt werden, deren Relevanzwert in der betrachteten Iteration verändert wurden, wird als $\Delta N_{DV}^{t=0} = N(d_G=1 \mid G_{DV})_{s(1|i)}^{t=0}$ bestimmt.

Es entsteht somit eine individuelle, globale GNG-SOM $N_{DV}^{t=0}$, zu der ein prototypbasiertes Relevanz-Approximationsmodell AM(rel(x) | $N_{DV}^{t=0}$) korrespondiert, was einer Modell-Monorepräsentation entspricht.

$$N_{DV}^{t=0} = \{n_{DV,j}^{t=0} = (w(x_{DV})_j, rel(w(x_{DV})_{s(1|i)})^{t=0\wedge}, M_{DV,j}, M_{DV(m),j}^{t=0}, C_{DV,j}) \mid j = 1, ..., \mu_{N,DV}\}. \quad (740)$$

Die Modell-Polyrepräsentation soll mit Hilfe eines Stimulus-Bootstrap erzeugt werden, d.h. aus $M_{DV(m)}^{t=0}$ werden jeweils Stimulus-Ziehungen mit Zurücklegen durchgeführt, sodass insgesamt δ Bootstrap-Stimuluslisten $M_{DV(m),p}^{t=0}$ erzeugt werden. Das Ursprungs-GNG-SOM $N_{DV}^{t=0}$ wird δ mal kopiert, die Kopien werden zu $N_{DV,p}^{t=0}$ umbenannt, und in den Kopien wird die lokale Stimulusmenge $M_{DV(m),s(1|i)}^{t=0}$ des Neurons $n_{DV,p,s(1|i)}^{t=0}$ durch die Bootstrap-Stimulusliste $M_{DV(m),p,s(1|i)}^{t=0}$ ersetzt. Da im weiteren der spezielle Fall betrachtet werden soll, dass die Gewichtsvektoren konstant bleiben und nur die Relevanzschätzungen verändert werden, werden die Elemente in der Menge $\Delta N_{DV}^{t=0}$ in allen δ GNG-SOMs $N_{DV,p}^{t=0}$ mit Hilfe von $M_{DV(m),p}^{t=0} = M_{DV(m),p,s(1|i)}^{t=0}$ aktualisiert. D.h. in t=0 wird die Relevanzschätzung des Neurons $n_{DV,s(1|i),p}^{t=0}$ in allen Bootstrap-GNG-SOMs $N_{DV,p}^{t=0}$ aktualisiert, indem alle Stimuli in der lokalen Stimulusmenge eine instanzbasierte LWR-Approximation durchführen, wobei einzelne Elemente mehrfach vorkommen können und andere Elemente gegenüber $M_{DV(m)}^{t=0} = M_{DV(m),s(1|i)}^{t=0}$ fehlen:

$$\begin{aligned} rel(w(x_{DV})_{s(1|i)})_p^{t=0\wedge} &= rel(w(x_{DV})_{s(1|i)} \mid AM(rel(x) \mid M_{DV(m),p}^{t=0}) \\ &= 1/\nu_{p,s(1|i)} \sum_j h(d_{DVR}(x_j^{t=0}, w(x_{DV})_{s(1|i)})) * rel(x_j^{t=0}), \text{ mit} \\ \nu_{p,s(1|i)} &= \sum_j h(d_{DVR}(x_j^{t=0}, w(x_{DV})_{s(1|i)})), \forall x_j^{t=0} \in M_{DV(m),p}^{t=0}. \end{aligned} \quad (741)$$

Für die Nachbarneurone aus $N(d_G=1 \mid G_{DV})_{s(1|i)}^{t=0}$ wird ebenfalls eine Relevanzschätzung an der Stelle der Gewichtsvektoren mit Hilfe der Modelle AM(rel(x)| $M_{DV(m),p}^{t=0}$) erfolgen, auch wenn in ihren Voronoi-Regionen keine Dokumentvektoren liegen, die bislang bewertet wurden. Nach den Relevanzschätzungen ergeben sich die Modelle $N_{DV,p}^{t=0}$ mit der folgenden Struktur:

$$\begin{aligned} N_{DV,p}^{t=0} = \{n_{DV,p,j}^{t=0} = (w(x_{DV})_j, rel(w(x_{DV})_{s(1|i)})_p^{t=0\wedge}, M_{DV,j}, M_{DV(m),p,j}^{t}, C_{DV,j}) \\ \mid j = 1, ..., \mu_{N,DV}\}, p = 1, ..., \delta. \end{aligned} \quad (742)$$

Je nach den verwendeten Vorgehensweisen bei der Polyrepräsentation von prototypbasierten Fehlermodellen (siehe Abschnitt 4.6.2.2)) sind die Datenstrukturen $N_{DV}^{t=0}$ und $N_{DV,p}^{t=0}$ vollständig, bzw. es müssen den ausgewählten Neuronen Bias- und Relevanz-Varianz-Schätzungen zugeordnet werden. Bei einer Modell-Polyrepräsentation bietet es sich an, die Bias- und Relevanz-Varianz-Schätzungen erst an den Stellen der Kandidaten-Dokumentvektoren zu erzeugen, sodass hier vereinbart werden soll, dass die Datenstrukturen $N_{DV}^{t=0}$ und $N_{DV,p}^{t=0}$ als vollständig zu betrachten sind, was eine Übereinstimmung mit der Vorgehensweise bei der Relevanz-Maximierung darstellt.

Nachdem die δ+1 Relevanz-Approximationsmodelle AM(rel(x) | $N_{DV,p}^{t=0}$) durch die Stützpunkte im DVR und im Relevanzraum in Verbindung mit einem LWR-Approximationsverfahren spezifiziert wurden, wird die Iteration t=1 eingeleitet, indem die primäre Ergebnismenge $DVM_{prim}^{t=1}$ bestimmt wird. Hierzu wird zunächst die Neuronenmenge $N_{DV,bew}^{t=1}$ als Teilmenge von N_{DV} gebildet, in deren Dokumentvektorenmenge mindestens ein Element enthalten ist, zu dem eine Relevanzbewertung durch den

Agenten vorliegt. Für t=1 ist $N_{DV,bew}^{t=1}$ gleich $\{n_{DV,s(1|i)}^{t=0}\}$. Die primäre Ergebnismenge wird als Vereinigung der Dokumentvektorenmengen der Neurone aus $N_{DV,bew}^{t}$ und ihrer verbundenen Nachbarn gebildet, korrigiert um die Dokumentvektoren, für die bereits der momentane Agent eine Relevanzbewertung abgegeben hat:

$$DVM_{prim}^{t=1} = \{\bigcup_k M_{DV,k}\} \setminus \{x_j^{t=0} \mid \forall m_j^{t=0} \in M_{DV(m)}^{t=0}\},$$
$$\forall n_{DV,k} \in N(d_G \leq 1 \mid G_{DV})_{s(1|i)}^{t=0}. \quad (743)$$

Die primäre Ergebnismenge DVM_{prim}^{t} übernimmt die Funktion der Test- bzw. Kandidatenmenge TM^t im Verfahren von RayChaudhuri & Hamey (1996[275]). Für jeden Kandidaten-Stimulus werden δ+1 Relevanzschätzungen durch das Gesamtmodell $N_{DV}^{t=0}$ und die Bootstrap-Modelle $N_{DV,p}^{t=0}$ gebildet:

$$rel(x_j^{t=1})\wedge = 1/\nu_j \sum_k h(d_X(x_j^{t=1}, w(x_{DV})_k)) * rel(w(x_{DV})_k)\wedge,$$
$$\nu_j = \sum_k h(d_X(x_j^{t=1}, w(x_{DV})_k)), \forall n_{DV,k} \in N_{DV}^{t=0},$$
$$rel(x_j^{t=1})_p\wedge = 1/\nu_{p,j} \sum_k h(d_X(x_j^{t=1}, w(x_{DV})_k)) * rel(w(x_{DV})_k)_p\wedge,$$
$$\nu_{p,j} = \sum_k h(x_j^{t=1}, d_X(w(x_{DV})_k)), \forall n_{DV,p,k} \in N_{DV,p}^{t=0}, p = 1, ..., \delta. \quad (744)$$

Für die Bias-Schätzung wird ein Bootstrap-Relevanz-Mittelwert und für die Relevanz-Varianz-Schätzung ein Gesamt-Mittelwert gebildet:

$$rel(x_j^{t=1})_{boot}\wedge = 1/\delta \sum_p rel(x_j^{t=1})_p\wedge,$$
$$\overline{rel}(x_j^{t=1})\wedge = 1/(\delta+1) * [rel(x_j^{t=1})\wedge + \sum_p rel(x_j^{t=1})_p\wedge]. \quad (745)$$

Eine 0,623-Biasschätzung und eine Relevanz-Varianz-Schätzung ergeben sich zu:

$$bias(x_j^{t=1})_{0,632}\wedge = [rel(x_j^{t=1})_{boot}\wedge - rel(x_j^{t=1})\wedge)]/0{,}632,$$
$$\sigma(rel(x_j^{t=1}))^2 = 1/(\delta+1) * [(rel(x_j^{t=1})\wedge - \overline{rel}(x_j^{t=1})\wedge)^2 + \sum_p (rel(x_j^{t=1})_p\wedge - \overline{rel}(x_j^{t=1})\wedge)^2. \quad (746)$$

Für alle Dokumentvektoren $x_j^{t=1}$ der primären Ergebnismenge wird ein Kandidaten-Stimulus $m_j^{t=1}$ mit den berechneten Schätzungen erstellt:

$$m_j^{t=1} = (x_j^{t=1}, \_, rel(x_j^{t=1})\wedge, rel(x_j^{t=1})_p\wedge, bias(x_j^{t=1})_{0,632}\wedge, \sigma(rel(x_j^{t=1}))^2\wedge \mid p = 1, ..., \delta),$$
$$\forall x_j^{t=1} \in DVM_{prim}^{t=1}. \quad (747)$$

Wird als Selektionskriterium ausschließlich die Modell-Maximierung verwendet, so werden die Kandidaten-Stimuli ausgewählt, welche die größten Fehlerschätzungen und die größten Unsicherheiten in der Relevanzschätzung, d.h. die größten Relevanz-Varianz-Schätzungen besitzen. Ist nur eine der beiden Ordnungskriterien vorhanden, so kann die sekundäre Ergebnisliste nach fallenden Bias- bzw. Relevanz-Varianzschätzungen geordnet werden. Liegen beide Ordnungskriterien vor und sollen beide integriert werden, so handelt es sich um ein Zwei-Ziel-Entscheidungsproblem, für das unterschiedliche Lösungsansätze existieren, die sich in zwei Klassen einteilen lassen:

1) Unabhängige Listenbildung

2) Abhängige Listenbildung

5.2.1.4.1) Unabhängige Listenbildung

Bei der unabhängigen Listenbildung werden für die beiden Ordnungskriterien zunächst unabhängig je eine sekundäre Ergebnisliste gebildet:

$$DV_{sec|bias}^{t=1} = (x_{b(j)}^{t=1} \mid bias(x_{b(j)}^{t=1})_{0,632}\text{^} > bias(x_{b(j+1)}^{t=1})_{0,632}\text{^};\ \forall x_j^{t=1} \in DVM_{prim}^{t=1}),$$
$$DV_{sec|\delta}^{t=1} = (x_{\sigma(j)}^{t=1} \mid \sigma(rel(x_{\sigma(j)}^{t=1}))^2 > \sigma(rel(x_{\sigma(j+1)}^{t=1}))^2;\ \forall x_j^{t=1} \in DVM_{prim}^{t=1}). \quad (748)$$

In einem weiteren Schritt wird durch ein Selektionskriterium aus beiden Listen eine Teilliste der ersten Elemente gebildet, die als tertiäre Liste $DV_{ter|bias}^{t=1}$ und $DV_{ter|\delta}^{t=1}$ verwendet werden, wobei als Heuristik wiederum die Anzahl $\#M_{DV(m)}^{t=0}$ als Referenz verwendet werden kann:

$$\#DV_{ter|bias}^{t=1} + \#DV_{ter|\delta}^{t=1} = \#M_{DV(m)}^{t=0}. \quad (749)$$

Problemlos ist diese Heuristik verwendbar, wenn $\#M_{DV(m)}^{t=0}$ gerade ist, da aus jeder der beiden Teillisten die ersten $\#M_{DV(m)}^{t=0}/2$ Elemente übernommen werden. Ist $\#M_{DV(m)}^{t=0}$ ungerade, so kann z.B. auf $\#M_{DV(m)}^{t=0}+1$ aufgerundet werden, da die Anzahl der zu präsentierenden Dokumente ein externer Parameter ist. Wichtiger als die Anzahl ist jedoch die Frage, wie die ausgewählten Elemente präsentiert werden. Bei der unabhängigen Listenbildung existieren hierfür die Ansätze der

1) Erhaltung der zwei Listen bzw. die
2) Integration zu einer Liste.

Bei der Listenerhaltung bleiben die beiden Listenstrukturen von $DV_{ter|bias}^{t=1}$ und $DV_{ter|\delta}^{t=1}$ erhalten, d.h. dem Agenten werden zwei Dokumentlisten präsentiert, wobei die eine die Dokumente enthält, die zu $DV_{ter|bias}^{t=1}$ korrespondiert und die andere Dokumentliste korrespondiert zu $DV_{ter|\delta}^{t=1}$. Zusätzlich zu den beiden Listen wird das Selektionskriterium benannt, d.h. der Agent wird darauf hingewiesen, dass in der ersten Liste Dokumente liegen, deren Relevanzschätzung größere Fehler aufweist, und dass in der zweiten Liste Dokumente vorliegen, deren Relevanzschätzung größere Unsicherheiten besitzen. Die Listenerhaltung kann geeignet sein, wenn wenige Ordnungskriterien vorliegen, wie die Bias- und die Varianz-Schätzung in dem dargestellten Fall. Werden jedoch mehrere Kriterien verwendet, die zudem zunehmend unintuitiver werden, wie die Verwendung von Verteilungsbeschreibungen durch Fehlermomente höherer Ordnung (siehe Abschnitt 2.3.7)), so besitzt die Darstellung mit mehreren Listen keine Differenzierungsfähigkeiten für den Agenten mehr.

Bei zwei Listen kann jedoch schon das Problem des zweifachen Auftretens von Dokumentvektoren und somit korrespondierender Dokumente auftreten, was Rückwirkungen auf die Anzahl der zu übernehmenden Elemente aus $DV_{sec|bias}^{t=1}$ und $DV_{sec|\delta}^{t=1}$ haben kann. Prinzipiell kann das mehrfache Auftreten akzeptiert oder korrigiert werden, wobei im zweiten Fall ein Kriterium formuliert werden muss, welches entscheidet, in welcher Liste ein zweifach auftretendes Element verbleiben bzw. gelöscht werden soll. Eine Korrektur macht das Verfahren nicht mehr unabhängig, da die beiden tertiären Listen in Abstimmung zueinander gebildet werden. Ein einfaches Kriterium ist die externe Auswahl einer Liste die dominiert, d.h. alle darin vorkommenden Elemente werden in der entsprechenden Reihenfolge übernommen. Wird z.B. $DV_{sec|bias}^{t=1}$ als dominierende Liste ausgewählt, so wird eine Sollanzahl der ersten Elemente als $DV_{ter|bias}^{t=1}$ übernommen, und die Elemente, die in $DV_{ter|bias}^{t=1}$ liegen, werden in $DV_{sec|\delta}^{t=1}$ eliminiert. Die Übernahme der ersten Elemente aus $DV_{sec|\delta}^{t=1}$ für $DV_{ter|\delta}^{t=1}$ ignoriert die entstandenen Lük-

ken in der Liste $DV_{sec|\delta}^{t=1}$, d.h. es wird die Sollanzahl von noch vorhandenen Dokumentvektoren aus der nicht dominierenden Liste $DV_{sec|\delta}^{t=1}$ übernommen. Eine solche Vorgehensweise funktioniert bei mehr als zwei Listen, wenn die Dominanz abgestuft wird, d.h. wenn eine Dominanz-Rangfolge der unabhängig erzeugten Listen extern vorgegeben ist.

Eine Dominanz-Rangfolge ist ein Zwischenschritt in Richtung Integration der unabhängigen Sekundärlisten in genau eine Tertiärliste, da bei einer Tertiärliste ebenfalls keine Dokumentvektoren mehrfach auftreten dürfen und ein Integrationskriterium notwendig ist. Die gleiche Problematik stellt sich, wenn bei der Integration der Relevanz- und der Modell-Maximierung drei Kriterien zu integrieren sind (siehe Abschnitt 5.3)), sodass hier nur kurz auf die Erzeugung einer gemeinsamen tertiären Liste eingegangen werden soll.

Ein Integrationskriterium muss aus dem Vergleich bzw. der Aggregation der Bias- und der Varianzschätzung abgeleitet werden, bzw. aus den beiden Rängen, die einem Kandidaten-Stimulus $m_j^{t=1}$ aus $DVM_{prim}^{t=1}$ in $DV_{sec|bias}^{t=1}$ und $DV_{sec|\delta}^{t=1}$ zugeordnet wird. In beiden Fällen handelt es sich um eine Zwei-Ziel-Entscheidung, bei der das Pareto-Kriterium oder eine Aggregationsfunktion der beiden Variablen angewendet werden muss, wie z.B. die ungewichtete Addition der Bias- und der Varianzschätzung bzw. der beiden Ränge. Bei der Aggregation wird aus einer Mehr-Ziel-Entscheidung ein Ein-Ziel-Entscheidungsproblem, da die verschiedenen Variablen zu einer Variablen aggregiert werden.

Werden die beiden Schätzungen Bias- und Varianzschätzung verwendet, so ist die Bildung der zwei Listen $DV_{sec|bias}^{t=1}$ und $DV_{sec|\delta}^{t=1}$ unnötig, da das Pareto-Kriterium oder eine Aggregationsfunktion direkt für die Elemente $m_j^{t=1}$ angewendet werden kann. Aus diesem Grunde soll sich das Pareto-Kriterium oder eine Aggregationsfunktion beim Vorliegen zweier Sekundärlisten auf die Ränge der Elemente $m_j^{t=1}$ beziehen, die mit $rang_{b(j)}^{t=1}$ und $rang_{\sigma(j)}^{t=1}$ bezeichnet und in die Datenstruktur von $m_j^{t=1}$ aufgenommen werden:

$$m_j^{t=1} = (x_j^{t=1}, \_, rel(x_j^{t=1})\hat{}, rel(x_j^{t=1})_p\hat{}, bias(x_j^{t=1})_{0,632}\hat{}, rang_{b(j)}^{t=1}, \sigma(rel(x_j^{t=1}))^2\hat{}, rang_{\sigma(j)}^{t=1} \mid p = 1, ..., \delta), \forall x_j^{t=1} \in DVM_{prim}^{t=1}. \quad (750)$$

Als Beispiel einer Aggregationsfunktion, die auf den Rängen basiert, wird vereinfachend die ungewichtete Summe betrachtet:

$$rang_j^{t=1} = rang_{b(j)}^{t=1} + rang_{\sigma(j)}^{t=1}. \quad (751)$$

Mit dieser rangbasierten Variablen wird eine neue sekundäre Liste $DV_{sec|b,\sigma}^{t=1}$ gebildet, welche die beiden Listen $DV_{sec|bias}^{t=1}$ und $DV_{sec|\delta}^{t=1}$ zusammenfasst, und mit der die tertiäre Liste $DV_{ter|b,\sigma}^{t=1}$ eindeutig gebildet werden kann, indem die Sollanzahl $\#M_{DV(m)}^{t=0}$ der ersten Listenelemente übernommen wird:

$$DV_{sec|b,\sigma}^{t=1} = (x_j^{t=1} \mid rang_j^{t=1} > rang_{j+1}^{t=1}; \forall x_j^{t=1} \in DVM_{prim}^{t=1}). \quad (752)$$

Wird die Paretomenge auf der Basis der beiden rangbasierten Variablen $(rang_{b(j)}^{t=1}, rang_{\sigma(j)}^{t=1})$ gebildet, so besitzt man zunächst keinen Einfluss auf die Anzahl der Elemente, d.h. die Paretomenge wird in den

meisten Fällen keine $\#M_{DV(m)}^{t=0}$ Elemente enthalten. Die $\#M_{DV(m)}^{t=0}$-Heuristik aufzugeben und die Paretomenge als Grundlage für die tertiäre Liste zu verwenden, ist jedoch keine geeignete Strategie, da zum einen die Paretomenge wenige bzw. genau ein Element enthalten kann, und zum anderen eine tertiäre Menge anstatt eine tertiäre Liste erzeugt wird, da die Elemente der Paretomenge per Definition gleichgewichtet sind. Anstatt einer einzelnen Paretomenge kann eine der unterschiedlichen Arten von Pareto-Hierarchien erzeugt werden (siehe Abschnitt 2.4.2); Bachelier (1998b:141ff[16])), wie z.B. die Hierarchiebildung durch sukzessive Deaktivierung von Paretomengen. Die einzelnen Hierarchieebenen sind als Ränge einer tertiären Ergebnisliste interpretierbar, wobei auf einem Rang mehrere Dokumentvektoren liegen, im Gegensatz zu der Bildung einer tertiären Liste mit genau einer Variablen, bei der auf einem Rang ein Dokumentvektor liegt, wenn man von dem Spezialfall absieht, dass mehrere Dokumentvektoren die gleiche Bewertung bezüglich der betrachteten Variable erhalten.

5.2.1.4.2) Abhängige Listenbildung

Bei der abhängigen Listenbildung wird auf den Zwischenschritt der Bildung von $DV_{sec|bias}^{t=1}$ und $DV_{sec|\delta}^{t=1}$ und deren Aggregation zu einer Liste verzichtet, und es wird direkt aus den beiden kadinalskalierten Variablen $(bias(x_j^{t=1})_{0,632}\wedge, \sigma(rel(x_j^{t=1}))^2\wedge)$ aus der Datenstruktur von $m_j^{t=1}$ genau eine sekundäre Ergebnisliste $DV_{sec|b,\sigma}^{t=1}$ gebildet. Hierzu kann wiederum eine Aggregationsfunktion, das Paretokriterium oder eine Intervallbildung verwendet werden, die bereits im Prinzip bei der fehler-korrigierten Relevanzschätzung eingesetzt wurde (siehe Abschnitt 4.6)).

Wird eine Aggregationsfunktion wie z.B. die ungewichtete Summenbildung verwendet, so werden die beiden kadinalskalierten Variablen zu einer neuen kadinalskalierten Variablen zusammengefasst:

$$g(x_j^{t=1}) = bias(x_j^{t=1})_{0,632}\wedge + \sigma(rel(x_j^{t=1}))^2\wedge. \tag{753}$$

Mit dieser neuen Variable wird eine sekundäre Liste $DV_{sec|g}^{t=1}$ gebildet, indem die Dokumentvektoren aus $DVM_{prim}^{t=1}$ nach fallendem Variablenwert geordnet werden:

$$DV_{sec|g}^{t=1} = (x_j^{t=1} \mid g(x_j^{t=1}) > g(x_{j+1}^{t=1}); \forall x_j^{t=1} \in DVM_{prim}^{t=1}). \tag{754}$$

Bei einem kombinierten Fehler- und Unsicherheits-Intervall wird die Biasschätzung als Intervallmittelpunkt und die Standardabweichung $\sigma(rel(x_j^{t=1}))$ bzw. der Betrag der Standardabweichung als linke und rechte Intervallgrenze verwendet. Allgemeiner kann dies in Verbindung mit einem Skalierungsfaktor α in Verbindung mit der Standardabweichung formuliert werden:

$$[bias(x_j^{t=1})_{0,632}\wedge - \alpha * |\sigma(rel(x_j^{t=1}))|, bias(x_j^{t=1})_{0,632}\wedge + \alpha * |\sigma(rel(x_j^{t=1}))|]. \tag{755}$$

Die Intervalle werden entsprechend ihrem Mittelpunkt auf einer Achse angeordnet, was als eine Form von Ranking interpretiert wird. Aus dieser sekundären Struktur muss eine Anzahl von Intervallen ausgewählt werden, welche die tertiäre Dokumentmenge bzw. -liste bildet, die dem Agenten präsentiert wird. Hierfür sind unterschiedliche Selektionsheuristiken anwendbar, wie z.B. die Auswahl der ersten $\#M_{DV(m)}^{t=0}$ Intervalle bezogen auf den Intervallmittelpunkt, sowie die Intervalle, die mit rechts davon liegen und die mit dem zuletzt selektierten Intervall eine Überlappung besitzen (siehe Abschnitt 2.6)). Der Nachteil dieser Heuristik besteht daran, dass die Anzahl der selektierten Intervalle nicht begrenzt

werden kann, mit der Folge, dass im Extremfall alle Intervalle übernommen werden müssen, wenn die Standardabweichungen der rechts liegenden Intervalle groß sind.

Die Verwendung des Paretokriteriums in Form einer Paretohierarchie nutzt die beiden kadinalskalierten Variablen ($bias(x_j^{t=1})_{0,632}\hat{}$, $\sigma(rel(x_j^{t=1}))^2\hat{}$) z.B. um sukzessiv Paretomengen aufzubauen und zu deaktivieren (siehe Abschnitt 2.4.2.2)), wobei jede dieser Paretomengen als Hierarchieebene und als Rang einer sekundären Liste interpretiert wird. Da jeder Rang mit mehreren Elementen belegt sein kann, werden die ersten Ränge in die tertiäre Struktur übernommen, bis der Rang erreicht ist, in der mehr Elemente liegen, als für die Sollanzahl $\#M_{DV(m)}^{t=0}$ notwendig sind, wenn alle Elemente dieses Rangs übernommen würden. Aus diesem Rang können durch Zufallsauswahl so viele Elemente gezogen werden, bis die Sollanzahl erreicht ist. Als effektivere Alternative können alle Elemente dieses letzten Ranges übernommen werden, da alle Elemente durch das Paretokriterium als gleichwertig betrachtet werden. Diese Alternative sollte angewendet werden, wenn pro Iteration nicht notwenig die gleiche Anzahl von Dokumenten nachgewiesen werden müssen, d.h. wenn $\#M_{DV(m)}^{t=0}$ ein Richtwert und kein Sollwert ist.

Unabhängig ob eine unabhängige oder eine abhängige Listenbildung verwendet wird, es wird eine tertiäre Listenstruktur $DV_{ter}^{t=1}$ erzeugt. Die zu diesen Dokumentvektoren korrespondierenden Dokumente werden dem Agenten präsentiert, sodass nach der Relevanzbewertung sich die Stimulusmenge $M_{DV(m)}^{t=1}$ ergibt, die zusammen mit $M_{DV(m)}^{t=0}$ die Gesamtmenge $M_{DV(m)}^{t\leq1}$ ergibt. Die Datenstruktur der Elemente $m_j^{t=1}$ aus $M_{DV(m)}^{t=1}$ wird entsprechend korrigiert, indem die Relevanzbewertung $rel(x_j^{t=1})$ eingetragen und die Schätzungen $rel(x_j^{t=1})\hat{}$, $rel(x_j^{t=1})_p\hat{}$, $bias(x_j^{t=1})_{0,632}\hat{}$, $\sigma(rel(x_j^{t=1}))^2\hat{}$, p = 1, ..., δ, gelöscht werden.

Im letzten Schritt in der Iteration t=1 wird mit $M_{DV(m)}^{t\leq1}$ oder nur mit $M_{DV(m)}^{t=1}$ die Modelle $AM(rel(x) \mid N_{DV}^{t=1})$ und $AM(rel(x) \mid N_{DV,p}^{t=1})$ aufgebaut bzw. die alten Modelle aktualisiert, wobei die Aktualisierung von $N_{DV}^{t=0}$ zu $N_{DV}^{t=1}$ und der On-scratch-Aufbau der Bootstrap-Modelle aus $N_{DV}^{t=1}$ betrachtet wird. Die Aktualisierung von $N_{DV}^{t=0}$ zu $N_{DV}^{t=1}$ kann durch Operationen im DVR und im Relevanzraum bzw. nur durch Operationen im Relevanzraum unter Beibehaltung der Gewichtsvektoren im DVR durchgeführt werden. Bei einer Aktualisierung in beiden Räumen wird zunächst die Gewichtsvektoren-Adaption durchgeführt, indem Elemente aus der verwendeten Stimulusmenge $M_{DV(m)}^{t\leq1}$ oder $M_{DV(m)}^{t=1}$ gezogen werden, für die der Gewichtsvektor des Gewinner-Neurons und seine Nachbar-Neurone nach den GNG-SOM-Regeln adaptiert wird. Darauf folgt die Aktualisierung im Relevanzraum, bei der alle Neurone, deren Gewichtsvektor adaptiert wurde, eine Relevanzschätzung zugeordnet bekommen. Werden während der Gewichtsvektor-Adaption Nachbar-Neurone adaptiert, so muss eine delokale instanzbasierte Approximation durchgeführt werden, da Nachbar-Neurone existieren, die adaptiert wurden, in deren Voronoi-Region jedoch kein Dokumentvektor liegt, der zu einem der Elemente in $M_{DV(m)}^{t\leq1}$ korrespondiert. Um Probleme dieser Art zu umgehen, sollte das jeweils gesamte Approximationsmodell $AM(rel(x) \mid M_{DV(m)}^{\leq t})$ verwendet werden, d.h. $AM(rel(x) \mid M_{DV(m)}^{t\leq1})$ in t=1.

Bei einer ausschließlichen Aktualisierung im Relevanzraum werden die Neurone ermittelt, in deren Voronoi-Region mindestens ein Dokumentvektor liegt, der eine Korrespondenz in $M_{DV(m)}^{t\leq1}$ besitzt, und diese Neurone erhalten durch $AM(rel(x) \mid M_{DV(m)}^{t\leq1})$ eine neue Relevanzschätzung.

Unabhängig welche der beiden Verfahren verwendet wird, es ergibt sich das aktualisierte SC-GNG-SOM-Modell $N_{DV}^{t=1}$, aus der mit Hilfe von Bootstrap-Stimuluslisten $M_{DV(m),p}^{t\leq1}$ die δ Bootstrap-GNG-SOMs $N_{DV,p}^{t=1}$ erzeugt werden. Bei der Aktualisierung im DVR und im Relevanzraum wird jeweils eine Kopie von $N_{DV}^{t=0}$ erzeugt, gefolgt von der Gewichtsvektoren-Adaption auf der Basis von $M_{DV(m),p}^{t\leq1}$ und der Neuschätzung der Relevanzwerte durch AM(rel(x) | $M_{DV(m),p}^{t\leq1}$). Bei der ausschließlichen Aktualisierung im Relevanzraum wird hierzu zunächst jeweils eine Kopie von $N_{DV}^{t=1}$ erzeugt, gefolgt von der Neuschätzung der Relevanzwerte durch AM(rel(x) | $M_{DV(m),p}^{t\leq1}$).

Mit der Fertigstellung von $N_{DV}^{t=1}$ und $N_{DV,p}^{t=1}$ wird t=1 beendet und t=2 begonnen, indem zunächst $N_{DV,bew}^{t=2}$ als Teilmenge von N_{DV} gebildet wird, in der die Neurone aufgenommen werden, in deren Voronoi-Region mindestens ein Dokumentvektor enthalten ist, zu dem eine Relevanzbewertung durch den Agenten vorliegt. Mit $N_{DV,bew}^{t=2}$ wird $DVM_{prim}^{t=2}$ gebildet, für deren Elemente Relevanzschätzungen durch die Modelle AM(rel(x) | $N_{DV}^{t=1}$) und AM(rel(x) | $N_{DV,p}^{t=1}$) gebildet werden, gefolgt von der Bildung der Biasschätzung und der Relevanz-Varianz.

5.2.2) Direkte Verfahren am Beispiel Optimal-Experiment Design

Im weiteren soll eine vereinfachte Darstellung des Optimal-Experiment-Designs (OED) beschrieben werden (siehe Fedorov (1972[113])), wobei sich auf Cohn (1995[73]) und Cohn et al. (1995[74]) bezogen wird. In Cohn (1995[73]) wird eine Bias- bzw. Bias-Quadrat-Minimierung und in Cohn et al. (1995[74]) eine Output-Varianz-Minimierung durch ein OED beschrieben, wobei eine gemeinsame Minimierung der Bias und der Output-Varianz nicht durchgeführt wurde.

5.2.2.1) Bias-Quadrat-Integral-Minimierung

In diesem Abschnitt soll die Bias-Minimierung ohne Varianz-Betrachtungen (Nur-Bias-Minimierung) als Kriterium des OED dargestellt werden. Hierzu wird in Cohn (1995[73]) eine Stimulusmenge M^t mit rauschfreien Stimuli $m_j^t = (x_j^t, y_j^t)$ angenommen, wobei als Initialisierung verwendet wird:

$$M^{t=0} = \{m_j^{t=0} = (x_j^{t=0}, y_j^{t=0})) \mid j = 1, ..., \mu_L\}. \quad (756)$$

Weiterhin wird ein deterministischer Lerner $AM(x)^t$ angenommen, der Output-Schätzungen $y_j^{t\wedge} = f(x_j^t \mid AM(x)^t)$ durchführen kann. In Cohn (1995[73]) wird ein instanzbasiertes LWR-Modell verwendet, d.h. Modelle der Form AM(x | M^t). Die Annahme bezüglich rauschfreien Stimuli dienen zur Herleitung von Termen und werden später in den experimentellen Untersuchungen von Cohn (1995[73]) in dieser Strenge aufgegeben, da bei einem entsprechenden Test der Input mit einem geringen Gauss-Rauschen von 1% überlagert wurde, ohne dass die Leistung der Nur-Bias-Minimierung schlechter wurde. Weiterhin wird angenommen, dass an jeder Stelle x_j des Inputraumes eine hinreichend gute Bias-Schätzung $bias(x_j)\wedge$ des Modells AM(x | M^t) erzeugt werden kann, um den richigen Funktionswert y_j gut zu schätzen:

$$y_j = f(x_j \mid AM(x \mid M^t)) - bias(x_j)\wedge. \quad (757)$$

Notwendig ist die Definition eines Qualitäts- bzw. Fehlermaß $Q[AM(x \mid M^t)]$ eines Modells, mit dem unterschiedliche Modellvarianten verglichen werden können. Allgemein wird hierfür der erwartete, quadratische Fehler eines Modells in einem Punkt x des Inputraumes über den gesamten Inputraum bzw. die Inputverteilung P(x) verwendet, und als Integral operationalisiert:

$$Q[AM(x \mid M^t)] = \int_x [f(x \mid AM(x \mid M^t)) - f(x)]^2 * P(x)\, dx. \quad (758)$$

Wird anstelle des quadratischen Fehlers die Bias bzw. die quadrierte Bias verwendet, wie bei der der Bias-Minimierung in Cohn (1995[73]), so ergibt sich als Modell-Qualität das Bias-Integral IB[.]. Da die Bias definiert ist als geschätzter Wert minus richtigem Wert, kann eine negative Bias bei einer Unterschätzung und eine positive Bias bei einer Überschätzung existieren. Integriert man über den Inputraum, so würden sich bei der Verwendung einer einfachen Bias Unter- und Überschätzungen rechnerisch ausgleichen und zu einer zu guten Modellqualität führen. Durch die Verwendung des Bias-Quadrates werden Richtungen wie Über- und Unterschätzung niveliert, doch eine Aufhebung wird verhindert:

$$IB[AM(x \mid M^t)] = \int_x bias(x \mid AM(x \mid M^t))^2 * P(x)\, dx. \quad (759)$$

Der Inputraum wird als ein hochdimensionaler, stetiger Vektorraum mit einer Distanzmetrik $d_X(.,.)$ angenommen, sodass ein Fehler- bzw. Bias-Quadrat-Integral numerisch geschätzt werden muss. In Cohn (1995[73]) wie in Cohn et al. (1995[74]) wird hierfür ein Monte-Carlo-Verfahren verwendet (Fishman (1995[114])), bei dem im Inputraum eine Menge zufälliger Punkte ausgewählt wird, für welche eine Bias-Schätzung erzeugt wird. Da an den ausgewählten zufälligen Stellen der richtige Outputwert unbekannt ist, muss der Outputwert geschätzt werden, wobei ein effektiverer Prozess verwendet werden muss, als die Schätzung durch das momentane Modell $AM(x \mid M^t)$ bzw. ein Modell-Kandidat. In Cohn (1995[73]) wird hierfür u.a. eine Bootstrap-Schätzung auf der Basis von Bootstrap-Restwertlisten (siehe Abschnitt 2.2.2)) verwendet. Diese kann ebenso durch eine Bootstrap-Schätzung auf der Basis von Bootstrap-Stimuluslisten durchgeführt werden.

Gegeben sei eine Stimulus-Kandidatenmenge KM^t analog zu der zufälligen Testmenge TM^t bei RayChaudhuri & Hamey (1996[275]) und der primären Dokumentvektoren-Ergebnismenge $DVM_{prim}{}^t$. Die Input-Komponenten $x_{K,j}{}^t$ der Elemente $m_{K,j}{}^t$ dieser Kandidaten-Stimulusmenge unterscheidet sich von allen Input-Komponenten der Elemente aus M^t, und der richtige Output $y_{K,j}{}^t$ ist zu dem betrachteten Zeitpunkt für alle Elemente unbekannt:

$$KM^t = \{m_{K,j}{}^t = (x_{K,j}{}^t, \_) \mid j = 1, ..., \mu_K\}. \quad (760)$$

Gesucht wird der Stimulus aus KM^t, der zu der besten erwarteten Modellqualität führen würde, wenn er in die Stimulusmenge aufgenommen würde. Hierzu wird für alle μ_K Elemente aus KM^t explizit die momentane Stimulusmenge M^t um $m_{K,j}{}^t$ erweitert, was zu dem instanzbasierten Modell $AM(x \mid M^t \cup \{m_{K,j}{}^t\})$ führt, für das eine Schätzung des gesamten Modellfehlers $IB[AM(x \mid M^t \cup \{m_{K,j}{}^t\})]$ durchgeführt wird. Ausgewählt wird das Modell mit der besten Modell-Qualität, d.h. mit dem kleinsten Bias-Integral. Wird ein Stützpunkt x der Monte-Carlo-Integral-Approximation betrachtet, so verändert die Aufnahme eines neuen Kandidaten-Stimulus $m_{K,j}{}^t$ in die Gesamt-Stimulusmenge die Outputschätzung und die Bias an dieser Stelle. Sei $\Delta y\hat{}$ die Veränderung der Outputschätzung an der Stelle x mit

$$\Delta y\hat{} = f(x \mid AM(x \mid M^t \cup \{m_{K,j}{}^t\}) - f(x \mid AM(x \mid M^t), \tag{761}$$

so ergibt sich die neue quadrierte Bias bias(x)'^{2} an der Stelle x durch (siehe Cohn (1995[73])):

$$\begin{aligned} bias(x)'^2 &= [f(x \mid AM(x \mid M^t \cup \{m_{K,j}{}^t\}) - y]^2 \\ &= [f(x \mid AM(x \mid M^t) + \Delta y\hat{} - y]^2 \\ &= \Delta y\hat{}^2 + 2 * \Delta y\hat{} * bias(x) + bias(x)^2. \end{aligned} \tag{762}$$

Die Bias bias(x) ist unabhängig von einem Stimulus-Kandidaten aus einer speziellen Kandidatenmenge KM^t, sodass die Minimierung von bias(x)'^{2} äquivalent zu der Minimierung von $\Delta y\hat{}^2 + 2 * \Delta y\hat{} * bias(x)$ ist. Der Wert $\Delta y\hat{}$ lässt sich mit einer LWR explizit durch $\Delta y\hat{} = f(x \mid AM(x \mid M^t \cup \{m_{K,j}{}^t\}) - f(x \mid AM(x \mid M^t)$ angeben, wobei Cohn (1995[73]) eine LOESS-Variante verwendet (Cleveland et al. (1988[68])), welche den gewichteten Input-Mittelwert μ_x, Output-Mittelwert μ_y, Input-Varianz σ_x und Covarianz σ_{xy} verwendet, ohne dass hierauf detailliert eingegangen werden soll. Wichtig ist jedoch, dass für die Bestimmung von $\Delta y\hat{}$ für einen Kandidaten-Stimulus $m_{K,j}{}^t = (x_{K,j}{}^t, \_)$ der richtige Outputwert $y_{K,j}{}^t$ bekannt sein müsste. Da dieser jedoch nicht bekannt ist, muss eine Schätzung durchgeführt werden, indem die Outputschätzung mit dem bekannten Modell $AM(x \mid M^t)$ um eine Bias-Schätzung $bias(x_{K,j}{}^t)\hat{}$ korrigiert wird:

$$y_{K,j}{}^t \approx f(x_{K,j}{}^t \mid AM(x \mid M^t)) - bias(x_{K,j}{}^t)\hat{}. \tag{763}$$

Diese Bias-Schätzung kann durch ein Bootstrap-Verfahren durchgeführt werden, indem Stimulus-Bootstraplisten $M_p{}^t$ durch μ_L faches zufälliges Ziehen mit Zurücklegen aus M^t gezogen werden. Eine solche Liste repräsentiert ein instanzbasiertes Modell $AM(x \mid M_p{}^t)$, mit dem eine Bootstrap-Output-Schätzung $f(x_{K,j}{}^t \mid AM(x \mid M_p{}^t))$ erzeugt wird. Werden δ Bootstraplisten und somit δ Einzelschätzungen erzeugt, so können diese zu einer Bootstrap-Gesamtschätzung aggregiert werden. Mit der Schätzung des Gesamtmodells $f(x_{K,j}{}^t \mid AM(x \mid M^t))$ und der 0,632-Heuristik ergibt dies eine Bias-Schätzung:

$$\begin{aligned} bias(x_{K,j}{}^t)_{0,632}\hat{} &= [f(x_{K,j}{}^t)_{boot}\hat{} - f(x_{K,j}{}^t \mid AM(x \mid M^t))]/0{,}632, \text{ mit} \\ f(x_{K,j}{}^t)_{boot}\hat{} &= 1/\delta * \sum_p f(x_{K,j}{}^t \mid AM(x \mid M_p{}^t)). \end{aligned} \tag{764}$$

Diese Vorgehensweise erfordert für jeden Kandidaten aus KM^t δ+1 Outputschätzungen, die jeweils aus μ_L gewichteten Summanden bestehen. Als wesentlich effizientere Alternative kann ein Bias-Approximationsmodell erzeugt werden, was bei Cohn (1995[73]) als „fitting cross-validated estimates“ bezeichnet wird. Hierbei wird jedem Element $m_j{}^t$ aus M^t zunächst eine Output-Schätzung durch $AM(x \mid M^t)$ zugeordnet, wobei bedacht werden muss, dass bei einem instanzbasierten Modell $x_j{}^t$ als Stützpunkt verwendet wird, sodass die Outputschätzung zu gut ist im Vergleich zu einem $x_{K,j}{}^t$, das weiter entfernt liegt. Aus diesem Grunde soll das Gewinnerlisten-Verfahren verwendet werden (siehe Abschnitt 2.3.8)), welches alle außer dem ersten Element der Gewinnerliste berücksichtigt, d.h. bei der Präsentation von $x_{K,j}{}^t$ wird aus M^t die Gewinnerliste $G(x_{K,j}{}^t \mid M^t, 2, \mu_L)$ gebildet, zu der das instanzbasierte Approximationsmodell $AM(x \mid G(x_{K,j}{}^t \mid M^t, 2, \mu_L))$ korrespondiert, das die folgende Output-Schätzung liefert:

$$\begin{aligned} f(x_j{}^t \mid AM(x \mid G(x_j{}^t \mid M^t, 2, \mu_L))) &= 1/\nu_j \sum_k h(d_X(x_j{}^t, x_k{}^t)) * f(x_k{}^t), \\ \nu_j &= \sum_k h(d_X(x_j{}^t, x_k{}^t)), \forall m_k{}^t \in G(x_j{}^t \mid M^t, 2, \mu_L). \end{aligned} \tag{765}$$

Die Bias an der Stelle x_j^t in Abhängigkeit von dem verwendeten Approximationsmodell kann exakt angegeben werden als:

$$bias(x_j^t) = f(x_j^t \mid AM(x \mid G(x_j^t \mid M^t, 2, \mu_L))) - y_j^t. \quad (766)$$

Wird dies für alle Elemente aus M^t durchgeführt, so ergibt sich die Datenstruktur:

$$M^t = \{m_j^t = (x_j^t, y_j^t, f(x_j^t \mid AM(x \mid G(x_j^t \mid M^t, 2, \mu_L)), bias(x_j^t)) \mid j = 1, ..., \mu_L\}. \quad (767)$$

Soll die Bias an der Stelle $x_{K,j}^t$ eines Kandidaten aus KM^t geschätzt werden, so kann dies mit einem instanzbasierten Bias-LWR-Approximationsmodell $AM(bias(x) \mid M^t)$ erfolgen, indem die Stützpunkte $(x_k^t, bias(x_k^t))$ der μ_L Elemente aus M^t verwendet werden:

$$bias(x_{K,j}^t)\wedge := bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)) = 1/\nu_j \sum_k h(d_X(x_{K,j}^t, x_k^t)) * bias(x_k^t),$$
$$\nu_j = \sum_k h(d_X(x_{K,j}^t, x_k^t)), \forall m_k^t \in M^t. \quad (768)$$

Diese Vorgehensweise erfordert pro Kandidat genau eine LWR-Approximation mit μ_L gewichteten Summanden als variabler Kostenfaktor. Es entsteht jedoch ein Mehraufwand bei der Erzeugung des Bias-Approximationsmodells $AM(bias(x) \mid M^t)$, da μ_L Output-Schätzungen mit je $(\mu_L - 1)$ gewichteten Summanden notwendig werden, was als fixer Kostenfaktor gesehen werden kann. Bei dem oben beschriebenen Bootstrap-Verfahren wird kein fixer Kostenfaktor benötigt, der variable Kostenfaktor ist jedoch deutlich höher. Für KM^t wird bei dem Bootstrap-Verfahren $[(\delta+1) * \mu_L] * \mu_K$ gewichtete Summanden verwendet, während bei dem Bias-Modell für KM^t $[\mu_L * (\mu_L - 1)] + [\mu_L * \mu_K]$ gewichtete Summanden benötigt werden. Welche der beiden Alternativen effizienter ist, ist somit von der Anzahl der Bootstrap-Listen durch den Parameter δ abhängig.

Das Gesamtverfahren soll im weiteren beschrieben werden, wobei von der Existenz einer Kandidatenmenge KM^t ausgegangen wird, deren Inputkomponenten durch einen Zufallsprozess oder einen Klassifikationsprozess im Fall der Dokumentvektoren erzeugt wird. Es soll ein Bias-Approximationsmodell verwendet werden, von dem ebenfalls ausgegangen werden soll, d.h. die Stimuli aus M^t besitzen die verkürzte Datenstruktur:

$$M^t = \{m_j^t = (x_j^t, y_j^t, bias(x_j^t)) \mid j = 1, ..., \mu_L\}. \quad (769)$$

Für einen Kandidaten $m_{K,j}^t$ wird zunächst die erweiterte Stimulusmenge $M^t \cup \{m_{K,j}^t\}$ gebildet, gefolgt von der Output-Schätzung $f(x_{K,j}^t \mid AM(x \mid M^t))$ durch $AM(x \mid M^t)$, sowie der Bias-Schätzung $bias(x_{K,j}^t \mid AM(bias(x) \mid M^t))$. Mit diesen beiden Schätzungen kann die bias-korrigierte Output-Schätzung $y_{K,bias,j}^t \approx f(x_{K,j}^t \mid AM(x \mid M^t)) - bias(x_{K,j}^t)\wedge$ erzeugt werden, die als temporäres Substitut des richtigen, unbekannten Outputs $y_{K,j}^t$ verwendet wird. Als temporäre Datenstruktur ergibt sich somit:

$$m_{K,j}^t = (x_{K,j}^t, \_, y_{K,bias,j}^t, f(x_{K,j}^t \mid AM(x \mid M^t)), bias(x_{K,j}^t \mid AM(bias(x) \mid M^t))). \quad (770)$$

Der nächste Schitt besteht in der Erzeugung einer Stützpunktmenge für die Monte-Carlo-Approximation des Bias-Integrals, wobei zwei prinzipielle Vorgehensweisen möglich sind:
1) Die Stützpunktmenge wird genau einmal bestimmt, und wird für alle Kandidaten aus KM^t verwendet.
2) Die Stützpunktmenge wird für jeden Kandidaten unabhängig neu bestimmt.

Im weiteren soll vom ersten Fall ausgegangen werden, d.h. es wird eine Stützpunktmenge SM^t mit μ_S Elementen durch einen gleichverteilten unabhängigen Zufallsprozess im Inputraum erzeugt:

$$SM^t = \{m_{S,i}{}^t = (x_{S,i}{}^t, \_) \mid i = 1, ..., \mu_S\}. \quad (771)$$

Die Anzahl μ_S der Stützpunkte wird als Funktion eines extern festgelegten Integrationsfehlers ε_{IB} der bestimmt und beträgt maximal $1/\varepsilon_{IB}{}^2$ für die Mehrfachintegration mit Hilfe der Monte-Carlo-Approximation (siehe Traub et al. (1988[338]), Traub & Wozniakowski (1994[338]), Wozniakowski (1996a[372], b[373]), Sloan & Wozniakowski (1996[314]), Werschulz & Wozniakowski (2000[358])). Bei einem Zufallsverfahren wie der Monte-Carlo-Approximation ist ε_{IB} keine sichere Aussage, d.h. der Fehler der Integration wird wahrscheinlich kleiner-gleich ε_{IB} sein. Das Bias-Quadrat-Integral über dem gesamten Inputraum wird approximiert, indem jedem der Stützpunkte ein Bias-Quadrat $bias(x_{S,i}{}^t \mid AM(x \mid M^t \cup \{m_{K,j}{}^t\}))^2$ auf der Basis des Kandidaten Modells $AM(x \mid M^t \cup \{m_{K,j}{}^t\})$ zugeordnet wird, und der arithmetische Mittelwert der Bias-Quadrat-Schätzungen gebildet wird, wobei die bezüglich $m_{K,j}{}^t$ individualisierte Stützpunktmenge $SM_{K,j}{}^t$ eingeführt werden soll:

$$\begin{aligned} IB[AM(x \mid M^t \cup \{m_{K,j}{}^t\})] &= \int_x bias(x \mid M^t \cup \{m_{K,j}{}^t\})^2 * P(x)\, dx, \\ IB[AM(x \mid M^t \cup \{m_{K,j}{}^t\})]\wedge &= 1/\mu_S * \sum_i bias(x_{S,i}{}^t)_{K,j}{}^2, \\ &= 1/\mu_S * \sum_i bias(x_{S,i}{}^t \mid AM(x \mid M^t \cup \{m_{K,j}{}^t\}))^2, \forall m_{S,i}{}^t \in SM_{K,j}{}^t. \end{aligned} \quad (772)$$

Die Bias-Quadrat-Schätzung an der Stelle eines Stützpunktes bei einer aktualisierten Stimulusmenge $M^t \cup \{m_{K,j}{}^t\}$ wurde als

$$bias(x_{S,i}{}^t)_{K,j}{}^2 = \Delta y_{S,i}{}^{t\wedge 2} + 2* \Delta y_{S,i}{}^{t\wedge} * bias(x_{S,i}{}^t) + bias(x_{S,i}{}^t)^2 \quad (773)$$

angegeben, sodass für $x_{S,i}{}^t$ die Output-Schätzungen $f(x_{S,i}{}^t \mid AM(x \mid M^t)$ und $f(x_{S,i}{}^t \mid AM(x \mid M^t \cup \{m_{K,j}{}^t\})$, sowie eine Bias-Schätzung notwendig werden. Die Bias-Schätzung kann mit $AM(bias(x) \mid M^t)$ durchgeführt werden. Denkbar wäre auch eine Schätzung mit $AM(bias(x) \mid M^t \cup \{m_{K,j}{}^t\})$, dabei wäre $bias(x_{K,j}{}^t)\wedge$ selbst eine Schätzung $bias(x_{K,j}{}^t \mid AM(bias(x) \mid M^t))$, sodass als Grundmenge M^t verwendet werden soll. Letztendlich bedeutet dies, dass $f(x_{S,i}{}^t \mid AM(x \mid M^t)$ und $bias(x_{S,i}{}^t \mid AM(bias(x) \mid M^t))$ unabhängig von einem Kandidaten $m_{K,j}{}^t$ ist, sodass diese beiden Schätzungen für alle Stützpunkte vor den Bias-Integral-Approximationen bestimmt und dann wiederverwendet werden können. Für die nicht individualisierte Stützpunktmenge ergibt dies eine erweiterte Datenstruktur:

$$SM^t = \{m_{S,i}{}^t = (x_{S,i}{}^t, \_, f(x_{S,i}{}^t \mid AM(x \mid M^t)), bias(x_{S,i}{}^t \mid AM(bias(x) \mid M^t))) \mid j = 1, ..., \mu_S\}. \quad (774)$$

Diese Stützpunktmenge wird durch $m_{K,j}{}^t$ zu $SM_{K,j}{}^t$ individualisiert, indem für jeden Stützpunkt $x_{S,i}{}^t$ eine Output-Schätzung $f(x_{S,i}{}^t \mid AM(x \mid M^t \cup \{m_{K,j}{}^t\})$ mit dem erweiterten Modell $AM(x \mid M^t \cup \{m_{K,j}{}^t\}$ gebildet wird:

$$\begin{aligned} SM_{K,j}{}^t = \{m_{S,i}{}^t = (x_{S,i}{}^t, \_, f(x_{S,i}{}^t \mid AM(x \mid M^t)), f(x_{S,i}{}^t \mid AM(x \mid M^t \cup \{m_{K,j}{}^t\})), \\ bias(x_{S,i}{}^t \mid AM(bias(x) \mid M^t))) \mid i = 1, ..., \mu_S\}. \end{aligned} \quad (775)$$

Damit kann das neue Bias-Quadrat $bias(x_{S,i}{}^t)_{K,j}{}^2$ an der Stelle des Stützpunktes $x_{S,i}{}^t$ geschätzt werden, indem $\Delta y_{S,i}{}^{t\wedge}$ durch $f(x_{S,i}{}^t \mid AM(x \mid M^t \cup \{m_{K,j}{}^t\}$ - $f(x_{S,i}{}^t \mid AM(x \mid M^t)$ berechnet wird, und $\Delta y_{S,i}{}^{t\wedge}$ sowie

$bias(x_{S,i}^t \mid AM(bias(x) \mid M^t))$ in die Bias-Quadrat-Gleichung (773) eingesetzt werden. Im nächsten Schritt wird der arithmetische Mittelwert der Bias-Quadrat-Schätzungen über alle μ_S Stützpunkte gebildet, der als Approximation $IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$ für das Bias-Quadrat-Integral verwendet wird. Dieser Wert wird der individualisierten Stützpunktmenge $SM_{K,j}^t$ zugeordnet, sodass sich die endgültige Datenstruktur ergibt:

$$SM_{K,j}^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\})), \Delta y_{S,i}^{t\wedge},$$
$$bias(x_{S,i}^t \mid AM(bias(x) \mid M^t)), bias(x_{S,i}^t)_{K,j}^2), IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge \mid i = 1, ..., \mu_S\}. \quad (776)$$

Wurde für jeden Kandidaten $m_{K,j}^t$ aus KM^t eine solche individuelle Stützpunktmenge $SM_{K,j}^t$ und ein approximiertes Bias-Quadrat-Integral erzeugt, so lassen sich die Kandidaten nach steigendem Wert des Integrals ordnen, sodass sich eine Kandidatenliste KL^t ergibt:

$$KL^t = (m_{K,j}^t = (x_{K,j}^t, \_, y_{K,bias,j}^t, f(x_{K,j}^t \mid AM(x \mid M^t)), bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)), SM_{K,j}^t,$$
$$IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge) \mid$$
$$IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge < IB[AM(x \mid M^t \cup \{m_{K,j+1}^t\})]^\wedge; j = 1, ..., \mu_K). \quad (777)$$

In Cohn (1995[73]) wird derjenige Kandidat $m_{K,min}^t$ mit dem kleinsten erwarteten Bias-Integral ausgewählt, d.h. $m_{K,j=1}^t$ aus KL^t, gefolgt von der externen Bestimmung des richtigen Outputwertes $y_{K,min}^t$ und dem Entfernen der anderen Komponenten aus der Datenstruktur von $m_{K,min}^t$, die nicht mehr benötigt werden. Der neue Stimulus $m_{K,min}^t = (x_{K,min}^t, y_{K,min}^t)$ wird in die Gesamt-Stimulusmenge aufgenommen, die zu M^{t+1} aktualisiert wird, wodurch das Approximationsmodell $AM(x \mid M^{t+1})$ spezifiziert ist. Daraufhin kann die Modell-Qualität durch $IB[AM(x \mid M^{t+1})]^\wedge$ neu bestimmt werden, da $IB[AM(x \mid M^t \cup \{m_{K,min}^t\})]^\wedge$ mit der Outputschätzung $y_{K,bias,min}^t$ anstatt dem richtigen Outputwert bestimmt wurde.

Die Auswahl genau eines Kandidaten ist die effektive, geradlinige Strategie, doch die Auswahl mehrerer Kandidaten pro Iteration wäre effizienter, da die vorliegende Berechnung der Modell-Qualitäten verwendet werden könnten, im Gegensatz zu der Einzelauswahl, bei der nicht übernommene Kandidaten in der nächsten Iteration nochmals vollständig neu bewertet werden müssen, wenn sie in der nachfolgenden Kandidatenmenge nochmals vorkommen. Im Kontext des Retrievals wäre die Auswahl mehrerer Dokumentvektoren ebenfalls sinnvoll, da ansonsten pro Iteration dem Agenten genau ein Dokument nachgewiesen wird, wenn ausschließlich das Modell-Maximierungskriterium verwendet wird. Dies relativiert sich, wenn eine Mischstrategie aus Relevanz-Maximierung und Modell-Maximierung verwendet wird, da dem Agenten mehrere Dokumente durch das Relevanz-Maximierungskriterium und ein Dokument durch das Modell-Maximierungskriterium präsentiert werden.

Das vorgestellte direkte Verfahren ist jedoch auf die Auswahl genau eines Kandidaten abgestimmt, d.h. es wurde mit $IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$ die Modellqualität geschätzt, die sich ergibt, wenn der Kandidat $m_{K,j}^t$ in die gegebene Stimulusmenge M^t aufgenommen würde. Dies ist zu unterscheiden von dem Fall, dass die Modellqualität geschätzt wird, wenn z.B. zwei neue Stimuli aufgenommen werden, was ein kombinatorisches Problem ergibt, da für alle Paare aus KM^t diese Bedingung umformuliert werden muss. D.h. es muss mit $IB[AM(x \mid M^t \cup \{m_{K,i}^t, m_{K,j}^t\})]^\wedge$ die Modellqualität geschätzt werden, die sich ergibt, wenn die beiden Kandidaten $m_{K,i}^t$ und $m_{K,j}^t$ in die gegebene Stimulusmenge M^t aufgenommen

würden. Bei μ_K Kandidaten in KM^t müssen $\mu_K * (\mu_K - 1)$ Paare gebildet werden, wobei für jedes Kandidatenpaar eine individuelle Stützpunktmenge $SM_{K,i,j}{}^t$ gebildet werden müsste, d.h. es müssten $\mu_K * (\mu_K - 1)$ Modellqualitäten in Form von Bias-Quadrat-Integralen gebildet werden. Allgemein lässt sich dies als Ziehen von k Elementen aus einer μ_K Elemente großen Grundmenge beschreiben, sodass die effektive Formulierung des direkten aktiven Lernens mehrerer Elemente nicht praktikabel ist. Diese Problemstellung wird im Abschnitt 5.4) behandelt.

5.2.2.2) Output-Varianz-Integral-Minimierung

In diesem Abschnitt soll die Output-Varianz-Minimierung ohne Bias-Betrachtungen (Nur-Varianz-Minimierung) als Kriterium des OED dargestellt werden. Verwendet wird auch hier eine Stimulusmenge M^t mit rauschfreien Stimuli $m_j{}^t = (x_j{}^t, y_j{}^t)$, sowie ein instanzbasiertes LWR-Modell $AM(x \mid M^t)$ (Cohn et al. (1995[74])). Weiterhin sei die Kandidatenmenge $KM^t = \{m_{K,j}{}^t = (x_{K,j}{}^t, \_) \mid j = 1, ..., \mu_K\}$ gegeben, sowie eine Stützpunktmenge $SM^t = \{m_{S,i}{}^t = (x_{S,i}{}^t, \_) \mid j = 1, ..., \mu_S\}$, wobei die Inputkomponenten der Stützpunkte durch einen Monte-Carlo-Prozess erzeugt werden.

Bei der Bias-Quadrat-Integral-Minimierung wurde jedem Stützpunkt $m_{S,i}{}^t$ eine Schätzung des Bias-Quadratwertes $bias(x_{S,i}{}^t)_{K,j}{}^2$ zugeordnet, unter der Annahme, dass ein Kandidat $m_{K,j}{}^t$ neues Element der Stimulusmenge ist und somit das instanzbasierte LWR-Modell $AM(x \mid M^t \cup \{m_{K,j}{}^t\})$ vorliegt. Auf diese Weise wird eine bezüglich $m_{K,j}{}^t$ individualisierte Stimulusmenge $SM_{K,j}{}^t$ erzeugt. Analog hierzu wird bei der Output-Varianz-Integral-Minimierung jedem Stützpunkt $m_{S,i}{}^t$ eine Schätzung der Output-Varianz $\sigma(x_{S,i}{}^t)_{K,j}{}^2$ zugeordnet, unter der Annahme, dass das instanzbasierte LWR-Modell $AM(x \mid M^t \cup \{m_{K,j}{}^t\})$ vorliegt.

Das Varianz-Integral über dem gesamten Inputraum wird approximiert, indem der arithmetische Mittelwert der Output-Varianzen $\sigma(x_{S,i}{}^t)_{K,j}{}^2$ über alle Stützpunkte gebildet wird:

$$\begin{aligned} IV[AM(x \mid M^t \cup \{m_{K,j}{}^t\})] &= \int_x \sigma(x \mid M^t \cup \{m_{K,j}{}^t\})^2 * P(x)\, dx, \\ IV[AM(x \mid M^t \cup \{m_{K,j}{}^t\})]\hat{} &= 1/\mu_S * \sum_i \sigma(x_{S,i}{}^t)_{K,j}{}^2, \\ &= 1/\mu_S * \sum_i \sigma(x_{S,i}{}^t \mid AM(x \mid M^t \cup \{m_{K,j}{}^t\}))^2, \forall m_{S,i}{}^t \in SM_{K,j}{}^t. \end{aligned} \quad (778)$$

Entsprechend der Darstellung in Cohn et al. (1995[74]) werden für $\sigma(x_{S,i}{}^t)_{K,j}{}^2$ die folgenden Terme benötigt:
1) Kernelwert an der Stelle des Kandidaten $m_{K,j}{}^t$ bezüglich des Stützpunktes $m_{S,i}{}^t$: $h(d_X(x_{S,i}{}^t, x_{K,j}{}^t))$.
2) Outputschätzung an der Stelle des Stützpunktes $m_{S,i}{}^t$ bezüglich der Stimulusmenge M^t:

$$\begin{aligned} f(x_{S,i}{}^t \mid AM(x \mid M^t)) = \mu(y)_{S,i}{}^t &= 1/\nu_{S,i} \sum_k h(d_X(x_{S,i}{}^t, x_k{}^t)) * f(x_k{}^t), \\ \nu_{S,i} &= \sum_k h(d_X(x_{S,i}{}^t, x_k{}^t)), \forall m_k{}^t = (x_k{}^t, f(x_k{}^t)) \in M^t. \end{aligned} \quad (779)$$

3) Output-Varianz an der Stelle des Stützpunktes $m_{S,i}{}^t$ bezüglich der Stimulusmenge M^t:

$$\begin{aligned} \sigma(x_{S,i}{}^t)^2 &= 1/\nu_{S,i} \sum_k h(d_X(x_{S,i}{}^t, x_k{}^t)) * (f(x_k{}^t) - f(x_{S,i}{}^t \mid AM(x \mid M^t)))^2, \\ \nu_{S,i} &= \sum_k h(d_X(x_{S,i}{}^t, x_k{}^t)), \forall m_k{}^t = (x_k{}^t, f(x_k{}^t)) \in M^t. \end{aligned} \quad (780)$$

4) Outputschätzung an der Stelle des Kandidaten $m_{K,j}^t$ bezüglich der Stimulusmenge M^t:

$$f(x_{K,j}^t \mid AM(x \mid M^t)) = \mu(y)_{K,j}^t = 1/\nu_{K,j} \sum_k h(d_X(x_{K,j}^t, x_k^t)) * f(x_k^t),$$
$$\nu_{K,j} = \sum_k h(d_X(x_{K,j}^t, x_k^t)), \forall m_k^t = (x_k^t, f(x_k^t)) \in M^t. \quad (781)$$

5) Output-Varianz an der Stelle des Kandidaten $m_{K,j}^t$ bezüglich der Stimulusmenge M^t:

$$\sigma(x_{K,j}^t)^2 = 1/\nu_{K,j} \sum_k h(d_X(x_{K,j}^t, x_k^t)) * (f(x_k^t) - f(x_{K,j}^t \mid AM(x \mid M^t)))^2,$$
$$\nu_{K,j} = \sum_k h(d_X(x_{S,i}^t, x_k^t)), \forall m_k^t = (x_k^t, f(x_k^t)) \in M^t. \quad (782)$$

Da die Outputschätzung und die Output-Varianz an der Stelle des Stützpunktes $m_{S,i}^t$ unabhängig von Kandidaten sind, genauso wie die Outputschätzung und die Output-Varianz an der Stelle des Kandidaten $m_{K,j}^t$ unabhängig von Stützpunkten sind, können die Datenstrukturen der Stützpunkte und der Kandidaten vor der Berechnung der μ_S kandidatenabhängigen Stützpunktmengen erweitert werden zu:

$$SM^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), \sigma(x_{S,i}^t)^2) \mid i = 1, ..., \mu_S\},$$
$$KM^t = \{m_{K,j}^t = (x_{K,j}^t, \_, f(x_{K,j}^t \mid AM(x \mid M^t)), \sigma(x_{K,j}^t)^2) \mid j = 1, ..., \mu_K\}. \quad (783)$$

Die Output-Varianz an der Stelle des Stützpunktes $m_{S,i}^t$ bei einer Stimulusmenge $M^t \cup \{m_{K,j}^t\}$ berechnet sich nach Cohn et al. (1995[74]) explizit zu:

$$\begin{aligned} \sigma(x_{S,i}^t)_{K,j}^2 &= [(\nu_{S,i} * \sigma(x_{S,i}^t)^2 + h(d_X(x_{S,i}^t, x_{K,j}^t)) * \sigma(x_{K,j}^t)^2)/(\nu_{S,i} + h(d_X(x_{S,i}^t, x_{K,j}^t))] \\ &+ [(\nu_{S,i} * h(d_X(x_{S,i}^t, x_{K,j}^t)) * (f(x_{K,j}^t \mid AM(x \mid M^t)) - f(x_{S,i}^t \mid AM(x \mid M^t)))^2)/ \\ &\quad (\nu_{S,i} + h(d_X(x_{S,i}^t, x_{K,j}^t))^2]. \end{aligned} \quad (784)$$

In Cohn et al. (1995[74]) wird eine korrigierte Output-Varianz $\sigma(x_{S,i}^t)_{K,j}^{2'}$ verwendet:

$$\sigma(x_{S,i}^t)_{K,j}^{2'} = \sigma(x_{S,i}^t)_{K,j}^2/(\nu_{S,i} + h(d_X(x_{S,i}^t, x_{K,j}^t))). \quad (785)$$

Die Stützpunktmenge wird durch den Bezug zu $m_{K,j}^t$ individualisiert zu:

$$SM_{K,j}^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), \sigma(x_{S,i}^t)^2, \sigma(x_{S,i}^t)_{K,j}^2, \sigma(x_{S,i}^t)_{K,j}^{2'}) \mid i = 1, ..., \mu_S\}. \quad (786)$$

Auf der Basis der korrigierten, individualisierten Output-Varianzen wird das Output-Varianz-Integral bestimmt zu:

$$IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge = 1/\mu_S * \sum_i \sigma(x_{S,i}^t)_{K,j}^{2'}, \forall m_{S,i}^t \in SM_{K,j}^t. \quad (787)$$

Wurde für jeden Kandidaten $m_{K,j}^t$ aus KM^t eine solche individuelle Stützpunktmenge $SM_{K,j}^t$ und das approximierte Output-Varianz-Integral erzeugt, so lassen sich die Kandidaten nach steigendem Wert des Integrals ordnen, sodass sich eine Kandidatenliste KL^t ergibt:

$$KL^t = (m_{K,j}^t = (x_{K,j}^t, \_, f(x_{K,j}^t \mid AM(x \mid M^t)), \sigma(x_{K,j}^t)^2, SM_{K,j}^t, IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge) \mid$$
$$IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge < IV[AM(x \mid M^t \cup \{m_{K,j+1}^t\})]^\wedge; j = 1, ..., \mu_K). \quad (788)$$

5.2.2.3) Kombinierte Bias-Quadrat- und Output-Varianz-Integral-Minimierung

Die Kombination von Bias-Quadrat- und Output-Varianz-Integral-Minimierung erfordert die Bildung der Vereinigungsmenge der jeweils benötigten Datenstrukturen der Stimulus-, der Kandidaten- und der Stützpunktmenge. Die Stimulusmenge M^t besitzt zu Beginn beider Verfahren die Basisstruktur

$$M^t = \{m_j^t = (x_j^t, y_j^t) \mid j = 1, ..., \mu_L\}. \tag{789}$$

Bei der Output-Varianz-Integral-Minimierung bleibt diese Struktur erhalten, wobei die Bezeichnung M_V^t verwendet werden soll. Bei der Bias-Quadrat-Integral-Minimierung wird neben dem Output-Approximationsmodell $AM(x \mid M^t)$ ein Bias-Approximationsmodell $AM(bias(x) \mid M^t)$ erforderlich, wenn die Bias-Schätzung an der Stelle von Stützpunkten nicht durch ein Resampling-Verfahren durchgeführt werden soll. Hierzu wird jedem Stimulus m_j^t eine Output-Schätzung wie z.B. $f(x_j^t \mid AM(x \mid G(x_j^t \mid M^t, 2, \mu_L))$ zugeordnet, d.h. es wird ein instanzbasiertes Gewinnerlisten-Verfahren verwendet. Die Differenz aus Schätzung und richtigem Output wird als Bias $bias(x_j^t)$ verwendet. Wird die Vereinigung der beiden Strukturen gebildet, so ergibt sich die Stimulusmenge M_{BV}^t, welche die gleiche Struktur wie M_B^t besitzt:

$$M_{BV}^t = M_B^t = \{m_j^t = (x_j^t, y_j^t, f(x_j^t \mid AM(x \mid G(x_j^t \mid M^t, 2, \mu_L)), bias(x_j^t)) \mid j = 1, ..., \mu_L\}. \tag{790}$$

Die Kandidatenmenge KM^t besitzt zu Beginn die Datenstruktur $KM^t = \{m_{K,j}^t = (x_{K,j}^t, \_) \mid j = 1, ..., \mu_K\}$. Bei der Output-Varianz-Integral-Minimierung wird die erweiterte Struktur mit Output-Schätzung und Output-Varianz gebildet, wobei die Stimuli aus M_V^t bzw. M_{BV}^t verwendet werden:

$$KM_V^t = \{m_{K,j}^t = (x_{K,j}^t, \_, f(x_{K,j}^t \mid AM(x \mid M^t)), \sigma(x_{K,j}^t)^2) \mid j = 1, ..., \mu_K\}. \tag{791}$$

Bei der Bias-Quadrat-Integral-Minimierung wird eine Output- und eine Bias-Schätzung durch $AM(x \mid M^t)$ und $AM(bias(x) \mid M^t)$ erzeugt. Hinzu kommt eine weitere Schätzung $y_{K,bias,j}^t$ des richtigen Outputwertes $y_{K,j}^t$, indem die Output-Schätzung um die Bias-Schätzung korrigiert wird. Die sich ergebende Datenstruktur ist somit:

$$KM_B^t = (m_{K,j}^t = (x_{K,j}^t, \_, y_{K,bias,j}^t, f(x_{K,j}^t \mid AM(x \mid M^t)), bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)) \mid j = 1, ..., \mu_K). \tag{792}$$

Als Vereinigung beider Strukturen ergibt sich:

$$KM_{BV}^t = \{m_{K,j}^t = (x_{K,j}^t, \_, y_{K,bias,j}^t, f(x_{K,j}^t \mid AM(x \mid M^t)), bias(x_{K,j}^t \mid AM(bias(x) \mid M^t), \sigma(x_{K,j}^t)^2) \mid j = 1, ..., \mu_K\}. \tag{793}$$

Die Stützpunktmenge SM^t besitzt zu Beginn die gleiche Basisstruktur wie KM^t, d.h. $SM^t = \{m_{S,i}^t = (x_{S,i}^t, \_) \mid j = 1, ..., \mu_S\}$. Bei der Output-Varianz-Integral-Minimierung wird diese Struktur um eine Output-Schätzung $f(x_{S,i}^t \mid AM(x \mid M^t))$ und um eine Output-Varianz $\sigma(x_{S,i}^t)^2$ erweitert:

$$SM_V^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), \sigma(x_{S,i}^t)^2) \mid i = 1, ..., \mu_S\}. \tag{794}$$

Bei der Bias-Quadrat-Integral-Minimierung wird die Basisstruktur durch eine Output-Schätzung und eine Bias-Schätzung durch das Bias-Approximationsmodell $AM(bias(x) \mid M^t)$ erweitert, sodass sich SM_B^t ergibt:

$$SM_B^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), bias(x_{S,i}^t \mid AM(bias(x) \mid M^t))) \mid i = 1, ..., \mu_S\}. \quad (795)$$

Als Vereinigung der beiden Strukturen ergibt sich SM_{BV}^t mit:

$$SM_{BV}^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), bias(x_{S,i}^t \mid AM(bias(x) \mid M^t)), \sigma(x_{S,i}^t)^2) \mid i = 1, ..., \mu_S\}. \quad (796)$$

Im weiteren folgt die μ_K'fache Individualisierung der Stützpunktmenge, indem SM^t unter Bezugnahme auf einen Kandidaten $m_{K,j}^t$ zu $SM_{K,j}^t$ transformiert wird. Die sich ergebenden Datenstrukturen sind bei der Output-Varianz- und der Bias-Quadrat-Integral-Minimierung unterschiedlich, sodass eine Kombination beider Minimierungen wieder die Vereinigung beider Datenstrukturen erfordert.

Ausgehend von SM_V^t wird bei der Output-Varianz-Integral-Minimierung eine Output-Varianz $\sigma(x_{S,i}^t)_{K,j}^2$ an der Stelle eines Stützpunktes $m_{S,i}^t$ berechnet, wenn $m_{K,j}^t$ als neues Element in die Stimulusmenge aufgenommen würde. Dies geschieht mit Hilfe der Terme $f(x_{S,i}^t \mid AM(x \mid M^t))$, $f(x_{K,j}^t \mid AM(x \mid M^t))$, $\sigma(x_{S,i}^t)^2$ und $\sigma(x_{K,j}^t)^2$. Mit Hilfe von $\nu_{S,i}$ und $h(d_X(x_{S,i}^t, x_{K,j}^t))$ erfolgt die Korrektur zu $\sigma(x_{S,i}^t)_{K,j}^{2'}$, d.h. die beiden Komponenten $\sigma(x_{S,i}^t)_{K,j}^2$ und $\sigma(x_{S,i}^t)_{K,j}^{2'}$ werden in die Datenstruktur der Stützpunkte eingetragen, wodurch die individualisierte Stützpunktmenge $SM_{V|K,j}^t$ erzeugt wird. Die Werte $\sigma(x_{S,i}^t)_{K,j}^{2'}$ aller Stützpunkte aus $SM_{V|K,j}^t$ werden zu der Output-Varianz-Integral-Approximation $IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$ aggregiert, die der Stützpunktmenge $SM_{V|K,j}^t$ zugeordnet wird, sodass sich die Struktur ergibt:

$$SM_{V|K,j}^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), \sigma(x_{S,i}^t)^2, \sigma(x_{S,i}^t)_{K,j}^2, \sigma(x_{S,i}^t)_{K,j}^{2'}), IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge \mid i = 1, ..., \mu_S\}. \quad (797)$$

Wurden die μ_S individualisierten Stützpunktmengen $SM_{V|K,j}^t$ ergzeugt, so können den Kandidaten die Output-Varianz-Integral-Approximation zugeordnet werden, sodass sich die Kandidatenmenge mit der folgenden Datenstruktur ergibt:

$$KM_V^t = \{m_{K,j}^t = (x_{K,j}^t, \_, f(x_{K,j}^t \mid AM(x \mid M^t)), \sigma(x_{K,j}^t)^2, IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge) \mid j = 1, ..., \mu_K\}. \quad (798)$$

Bei der Bias-Quadrat-Integral-Minimierung wird eine Output-Schätzung $f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\}))$ auf der Basis der potentiell neuen Stimulusmenge $M^t \cup \{m_{K,j}^t\}$ gebildet. Die Differenz aus dieser neuen Output-Schätzung und der alten Schätzung $f(x_{S,i}^t \mid AM(x \mid M^t))$ wird als $\Delta y_{S,i}^{t\wedge}$ bezeichnet, die zusammen mit der Bias-Schätzung $bias(x_{S,i}^t \mid AM(bias(x) \mid M^t))$ den Bias-Quadratwert $bias(x_{S,i}^t)_{K,j}^2$ bilden. Es ergibt sich die individualisierte Stützpunktmenge $SM_{B|K,j}^t$, wobei die Aggregation der Bias-Quadratwerte über alle Stützpunkte die Bias-Quadrat-Integral-Approximation $IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$ ergibt, die der Stützpunktmenge zugeordnet wird:

$$SM_{B|K,j}^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\})), \Delta y_{S,i}^{t\wedge}, bias(x_{S,i}^t \mid AM(bias(x) \mid M^t)), bias(x_{S,i}^t)_{K,j}^2), IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge \mid i = 1, ..., \mu_S\}. \quad (799)$$

Als Kandidatenmenge mit erweiterter Datenstruktur ergibt sich:

$$KM_B^t = (m_{K,j}^t = (x_{K,j}^t, \_, y_{K,bias,j}^t, f(x_{K,j}^t \mid AM(x \mid M^t)), bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)), IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge \mid j = 1, ..., \mu_K). \quad (800)$$

Bei der Kombination beider Ansätze ergibt sich die Frage, ob bei KM_B^t die Bestimmung der korrigierten Schätzung $y_{K,bias,j}^t$ des richtigen, unbekannten Outputwertes $y_{K,j}^t$ nur von der Bias $bias(x_{K,j}^t \mid AM(bias(x) \mid M^t))$ abhängig gemacht werden soll, oder ob zusätzlich in die Korrektur die Output-Varianz $\sigma(x_{K,j}^t)^2$ bzw. die Standard-Abweichung $\sigma(x_{K,j}^t)$ einbezogen werden soll. Im ersten Fall ergibt sich:

$$y_{K,bias,j}^t = f(x_{K,j}^t \mid AM(x \mid M^t)) - bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)). \quad (801)$$

Bedacht werden muss, dass die Output-Varianz keine Richtung besitzt, d.h. es kann bei der Korrektur durch die Standard-Abweichung nicht sinnvoll angegeben werden, ob eine subtraktive oder additive Korrektur erfolgen soll. Mit Hilfe eines Intervalls, das die bias-korrigierte Outputschätzung $f(x_{K,j}^t \mid AM(x \mid M^t)) - bias(x_{K,j}^t \mid AM(bias(x) \mid M^t))$ als Mittelpunkt besitzt und durch Subtraktion von $\sigma(x_{K,j}^t)$ die untere Grenze bzw. durch Addition von $\sigma(x_{K,j}^t)$ die obere Grenze besitzt, kann jedoch eine adäquate Modellierung der zugrunde liegenden Unsicherheit eingeführt werden. Auf diese Weise ist $y_{K,bias,j}^t$ kein Punkt mehr, sondern ein Intervall:

$$y_{K,bias,j}^t = [f(x_{K,j}^t \mid AM(x \mid M^t)) - bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)) - \sigma(x_{K,j}^t), f(x_{K,j}^t \mid AM(x \mid M^t)) - bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)) + \sigma(x_{K,j}^t)] . \quad (802)$$

Genutzt wird $y_{K,bias,j}^t$ bei der neuen Output-Schätzung der Stützpunkte durch das Modell $AM(x \mid M^t \cup \{m_{K,j}^t\})$ bei den Stützpunktmengen $SM_{B|K,j}^t$, d.h. der Stimulus $(x_{K,j}^t, y_{K,bias,j}^t)$ wird als Substitut für $(x_{K,j}^t, y_{K,j}^t)$ bei den μ_S Output-Schätzungen $f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\}))$ verwendet. Wird $y_{K,bias,j}^t$ als Intervall erzeugt, so muss das Approximationsmodell $AM(x \mid M^t \cup \{m_{K,j}^t\})$ durch eine Intervall-Arithmetik so reformuliert werden, dass es Intervalle als Input aufnehmen kann, was zur Folge hat, dass auch der Output ein Intervall sein wird. Dies pfanzt sich fort, sodass $\Delta y_{S,i}^{t\wedge}$ und $bias(x_{S,i}^t)_{K,j}^2$ ebenfalls Intervalle werden, genauso wie die Aggregation der Bias-Quadratwerte zu der Approximation des Integrals $IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$. Da der Integral-Approximation jedoch ein Fehler als Funktion der Anzahl der Stützpunkte zugeordnet werden kann, ergibt die Verwendung von Intervallen hier keine Veränderung der Datenstruktur. Auf die Darstellung einer rigorosen Verwendung von Intervallen anstatt Skalaren für alle vorliegenden Variablen, soll im Rahmen dieser Arbeit jedoch verzichtet werden.

Wird die Korrektur ohne die Output-Standardabweichung $\sigma(x_{K,j}^t)$ durchgeführt, so kann die Output-Varianz- und die Bias-Quadrat-Integral-Minimierung unabhängig durchgeführt werden, während bei der Korrektur mit $\sigma(x_{K,j}^t)$ die Output-Varianz-Integral-Minimierung vor der Bias-Quadrat-Integral-Minimierung durchgeführt werden muss. Genauer betrachtet ist eine teilweise Überlappung möglich, wobei die Berechnung von $f(x_{K,j}^t \mid AM(x \mid M^t))$ und $\sigma(x_{K,j}^t)^2$ für alle Kandidaten $m_{K,j}^t$ als erstes erfolgen kann, gefolgt von $bias(x_{K,j}^t \mid AM(bias(x) \mid M^t))$, sodass aus diesen Werten $y_{K,bias,j}^t$ durch die zwei-komponentige Korrektur berechenbar wird, mit der Folge, dass KM_{BV}^t vollständig vorliegt. Parallel zu der Bestimmung von KM_{BV}^t kann SM_{BV}^t bestimmt werden, da keine dieser Komponenten von Komponenten aus KM_{BV}^t abhängig ist. Nachdem KM_{BV}^t und SM_{BV}^t vorliegen, können die μ_K Individualisierungen von

SM_{BV}^t zu $SM_{BV|K,j}^t$ für alle $m_{K,j}^t$ aus KM_{BV}^t erfolgen. Hierzu wird zunächst die Outputschätzung $f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\}))$ bestimmt, gefolgt von der Differenz $\Delta y_{S,i}^{t\wedge}$ und dem Bias-Quadratwert $bias(x_{S,i}^t)_{K,j}^2$. Es folgt die Output-Varianz $\sigma(x_{S,i}^t)_{K,j}^2$ und ihre Korrektur zu $\sigma(x_{S,i}^t)_{K,j}^{2'}$, sodass sich die Struktur ergibt:

$$SM_{BV|K,j}^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\})), \Delta y_{S,i}^{t\wedge}, bias(x_{S,i}^t \mid AM(bias(x) \mid M^t)), \sigma(x_{S,i}^t)^2, bias(x_{S,i}^t)_{K,j}^2, \sigma(x_{S,i}^t)_{K,j}^2, \sigma(x_{S,i}^t)_{K,j}^{2'}) \mid i = 1, ..., \mu_S\}. \quad (803)$$

Werden die Bias-Quadratwerte zu dem approximierten Integral $IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$ und die Output-Varianzen $\sigma(x_{S,i}^t)_{K,j}^{2'}$ zu dem approximierten Integral $IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$ aggregiert, so können diese beiden Integralwerte der individuellen Stützpunktmenge $SM_{BV|K,j}^t$ zugeordnet werden:

$$SM_{BV|K,j}^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\})), \Delta y_{S,i}^{t\wedge}, bias(x_{S,i}^t \mid AM(bias(x) \mid M^t)), \sigma(x_{S,i}^t)^2, bias(x_{S,i}^t)_{K,j}^2, \sigma(x_{S,i}^t)_{K,j}^2, \sigma(x_{S,i}^t)_{K,j}^{2'}) IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge, IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge \mid i = 1, ..., \mu_S\}. \quad (804)$$

Sind alle individuellen Stützpunktmengen gebildet worden, so wird die Datenstruktur der Kandidaten in KM_{BV}^t erweitert, indem eine Referenz $SM_{BV|K,j}^t$ sowie die Integralwerte hinzu gefügt werden:

$$KM_{BV}^t = \{m_{K,j}^t = (x_{K,j}^t, \_, y_{K,bias,j}^t, f(x_{K,j}^t \mid AM(x \mid M^t)), bias(x_{K,j}^t \mid AM(bias(x) \mid M^t), \sigma(x_{K,j}^t)^2, SM_{BV|K,j}^t, IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge, IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge) \mid j = 1, ..., \mu_K\}. \quad (805)$$

Wie aus der Kandidatenmenge KM_{BV}^t eine Kandidatenliste KL_{BV}^t mit Hilfe der Bias-Quadrat- und Output-Varianz-Integrale gebildet wird, muss im weiteren spezifiziert werden. Bei einer solchen Mehr-Ziel-Entscheidung sind prinzipiell zwei Strategieklassen möglich:

1) Aggregation der beiden Funktionen zu einer einzelnen Funktion, sodass aus der Mehr-Ziel-Entscheidungssituation eine Ein-Ziel-Entscheidung wird.
2) Verwendung von expliziten Mehr-Ziel-Entscheidungsverfahren, wie der Bildung einer Pareto-Hierarchie (siehe Abschnitt 2.4.2)).

Bei der Aggregation der beiden Funktionen liegt die Bildung der Summe der beiden Integralwerte am nächsten, da bei der MSE-Zerlegung in eine Bias-Quadrat- und eine Varianz-Komponente ebenfalls eine einfache additive Zerlegung verwendet wird (siehe Abschnitt 2.3.3); Geman et al. (1992[142])):

$$IBV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge = IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge + IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge. \quad (806)$$

Besitzen die Integrale definierte Fehler wie ε_{IB} und ε_{IV}, so werden diese durch entsprechende Operationen der Intervall-Arithmetik ebenfalls addiert (siehe Bauch et al. (1987[30]), Adams & Kulisch (1993[3]), Klatte et al. (1993[182])), d.h. der IBV[.]-Wert besitzt ein größeres Fehler-Intervall als die beiden Einzelwerte, wenn eine einfache additive Aggregation durchgeführt wird. Es wird somit genau ein Ordnungskriterium erzeugt, sodass die Kandidaten aus KM_{BV}^t nach steigendem Wert der Integralsumme $IBV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$ sortiert werden:

$$KL_{BV}^{t} = (m_{K,j}^{t} = (x_{K,j}^{t}, \_, y_{K,bias,j}^{t}, f(x_{K,j}^{t} \mid AM(x \mid M^{t})), bias(x_{K,j}^{t} \mid AM(bias(x) \mid M^{t}), \sigma(x_{K,j}^{t})^{2},$$
$$SM_{BV|K,j}^{t}, IB[AM(x \mid M^{t} \cup \{m_{K,j}^{t}\})]\hat{}, IV[AM(x \mid M^{t} \cup \{m_{K,j}^{t}\})]\hat{}, IBV[AM(x \mid M^{t} \cup \{m_{K,j}^{t}\})]\hat{}) \mid$$
$$IBV[AM(x \mid M^{t} \cup \{m_{K,j}^{t}\})]\hat{} < IBV[AM(x \mid M^{t} \cup \{m_{K,j+1}^{t}\})]\hat{}; j = 1, ..., \mu_{K}). \quad (807)$$

Wird ein Mehr-Ziel-Entscheidungsverfahren wie der Bildung einer Pareto-Hierarchie verwendet, die z.B. durch die sukzessive Deaktivierung von Pareto-Mengen gebildet wird (siehe Abschnitt 2.4.2.2)), so wird zunächst die Paretomenge $PM(KM^{t})_{1}$ der Kandidaten aus KM^{t} gebildet, indem eine Minimierung von Bias-Quadrat- und Varianz-Integral unterstellt wird. Diese Paretomenge bildet den ersten Rang der Pareto-Hierarchie, d.h. im Gegensatz zu der Verwendung genau eines Ordnungskriteriums, ist es hierbei möglich, dass auf einem Rang eine Kandidatenmenge mit mehr als einem Element liegt. Bei genau einem Ordnungskriterium können auf einem Rang mehrere Kandidaten liegen, wenn der IBV[.]-Wert jeweils gleich ist, wobei das jeweilige Fehler-Intervall bei der Entscheidung berücksichtigt werden muss, ob gleiche Were vorliegen. Entsprechend dem Verfahren der sukzessiven Deaktivierung von Paretomengen werden die Elemente von $PM(KM^{t})_{1}$ aus KM^{t} entfernt, gefolgt von der erneuten Bildung der Paretomenge $PM(KM^{t})_{2}$ aus der Restmenge, bis alle Kandidaten aus KM^{t} einer Paretomenge $PM(KM^{t})_{k}$ zugeordnet sind, d.h. bis die Restmenge leer wird. Die Kandidatenliste KL^{t} besteht in diesem Fall aus einer Liste von sukzessiv erzeugten Paretomengen:

$$KL^{t} = (PM(KM^{t})_{k} \mid k = 1, ...). \quad (808)$$

5.2.3) Effizienzverbesserungen bei direkten Verfahren

Es existieren eine Reihe von Ansatzpunkten, mit denen versucht werden kann, den großen Aufwand in einer Iteration eines direkten Verfahrens zu verringern, ohne dass erwartet werden kann, dass der Aufwand pro Iteration auch nur annähernd so gering wird wie bei einem indirekten Verfahren. Prinzipiell zu unterscheiden sind Vorschläge, welche das Modell-Qualitätsmaß „Eigenschafts-Integral“ wie Bias- und des Output-Varianz-Integrals weiter verwenden und Vorschläge, die alternative Verfahren von Modell-Qualitäten verwenden.

5.2.3.1) Eigenschafts-Integrale mit weniger Stützpunkten durch Fehlerauswahl

Wird die Modellqualität durch ein oder mehrere Eigenschafts-Integrale bestimmt, so ist der Verfahrensaufwand wesentlich von der Anzahl μ_{S} der Stützpunkte bestimmt, die für die numerische Integration benötigt werden. Die Anzahl der Stützpunkte ist prinzipiell von drei Faktoren abhängig (siehe Traub et al. (1988[337]), Traub & Wozniakowski (1994[338]), Wozniakowski (1996a[372], b[373]), Sloan & Wozniakowski (1996[314]), Werschulz & Wozniakowski (2000[358])):

1) von dem Fehlermaß ε_{I}, mit dem ein Integral approximiert werden soll.
2) von der Dimension n des Inputraumes.
3) von der Glattheit der zu integrierenden Funktion, operationalisiert durch den Parameter r, der die Anzahl der beschränkten, partiellen Ableitungen angibt, mit der die zu integrierende Funktion ableitbar ist.

Bei zufallsbestimmten Verfahren berechnet sich die Anzahl der Stützpunkte proportional zu dem Fehlerwert ε_I, der als Funktion der Inputraum-Dimension und der Funktionsglattheit bestimmt wird:

$$1/\varepsilon_I^s, \text{ mit } s = 2/(1+2r/n) \in [0, 2]. \tag{809}$$

D.h. bei einem zufallsbestimmten Verfahren ist der maximale Aufwand kleiner-gleich $1/\varepsilon_I^2$, um ein Integral mit dem Fehler von ε_I zu approximieren, wobei der Extremfall $1/\varepsilon_I^2$ bei einer stetigen aber nirgends differenzierbaren Funktion (r = 0) in Verbindung mit einer Monte-Carlo-Approximation notwendig wird.

Die zu integrierende Funktion besteht im gegebenen Kontext aus einer Summe gewichteter Kernelfunktionen h((.,.)), da ein LWR-Bias-Approximationsmodell unterstellt wird, sodass die Glattheit der Gesamtfunktion direkt abhängig ist von der Glattheit der Kernelfunktion. In Cohn (1995[73]) und Cohn et al. (1995[74]) wird als Kernel eine Gaussfunktion mit k als einem externen Parameter verwendet,

$$h((x_i,x_j)) = \exp(-k * (x_i - x_j))^2), \text{ bzw. allgemeiner}$$
$$h(d_X(x_i,x_j)) = \exp(-k * d_X(x_i,x_j)). \tag{810}$$

Cleveland et al. (1988[68]) verwenden im Gegensatz hierzu eine trikubische Kernelfunktion, die im weiteren jedoch nicht betrachtet werden soll. In diesem Kontext besitzt die Gaussfunktion den entscheidenden Vorteil, da die Exponentialfunktion unendlich oft differenzierbar ist. Die Dimension des Inputraumes, d.h. DVR im Kontext des Retrieval, ist sehr groß, was jedoch durch die Glattheit der zu integrierenden Funktion überkompensiert wird.

Soll der Aufwand der Integration verringert werden, so kann dies durch die Wahl eines Fehlerparameters ε_I erfolgen, der so groß ist, dass die Erzeugung einer Rangfolge der Kandidaten aus KM^t ermöglicht wird, bzw. dass ein Element aus KM^t als Gewinner mit dem kleinsten Eigenschafts-Integral ermittelt werden kann. Durch die Kenntnis eines Fehlers bei einer gegebenen Anzahl von Stützpunkten wird einem Kandidaten nicht ein einzelner Wert für ein Eigenschafts-Integral zugeordnet, sondern ein Eigenschafts-Intervall, das als Konfidenz-Intervall interpretiert werden kann, und z.B. für das Bias-Integral eines Kandidaten $m_{K,j}^t \in KM^t$ die Form besitzt:

$$KI_{IB(K,j)} = [IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]\wedge - \varepsilon_{IB}, IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]\wedge + \varepsilon_{IB}]. \tag{811}$$

Die Kandidaten können entsprechend dem Mittelpunkt bzw. dem linken Rand ihres Konfidenz-Intervalls geordnet werden, wobei ein eindeutiger Gewinner $m_{K,min}^t$ existiert, wenn kein anderes Intervall mit $KI_{IB(K,min)}$ eine Überschneidung besitzt. Sollte der Kandidat auf dem ersten Rang mit einem oder mehreren anderen Kandidaten eine Intervall-Überlappung besitzen, so sind die folgenden, prinzipiellen Strategien anwendbar:

1) Ignorierung der Überlappungen und Auswahl des ersten Kandidaten.
2) Auswahl des ersten Kandidaten und aller anderen, die eine Intervall-Überlappung zum ersten Kandidaten besitzen.
3) Versuch, durch Erhöhung der Anzahl der Stützpunkte bei der Integral-Approximation der sich überlappenden Kandidaten, einen eindeutigen Gewinner zu erzeugen.

Die beiden ersten Strategien erfordern keinen weiteren Aufwand, besitzen jedoch unterschiedliche Effektivitätseigenschaften, da im ersten Fall nicht garantiert werden kann, dass der beste Kandidat ausgewählt wird. Im zweiten Fall ergibt sich die Problematik der Mehrfachauswahl in Verbindung mit einem Entscheidungskriterium, d.h. der Schätzungen durch ein Approximationsmodell, dessen Stimulusmenge um genau einen Stimulus erweitert wurde. Der dritte Fall erfordert weitere Stützpunkte, d.h. die kandidatenspezifischen Stützpunktmengen $SM_{K,j}{}^{t}$ werden durch neu erzeugte Objekte $m_{S,j,i}{}^{t}$ erweitert, wobei der Inputvektor $x_{S,j,i}{}^{t}$ gleichverteilt zufällig und unabhängig von den vorhandenen Inputvektoren erzeugt wird. Dies bedeutet, dass die Mengen $SM_{K,j}{}^{t}$ für alle Kandidaten zwischengespeichert werden müssen, bis letztendlich entsprechend dem Ranking entschieden werden kann, ob Intervall-Überlappungen vorliegen und welche Kandidaten somit betroffen sind. Die Stützpunktmengen der anderen Kandidaten können dann gelöscht werden, gefolgt von der Erweiterung der verbleibenden Mengen, wobei für die neuen Stützpunkte jeweils Outputschätzungen mit und ohne den Kandidaten und die notwendigen Biasschätzungen bzw. Output-Varianzen erzeugt werden. Mit den einzelnen Biasschätzungen und Output-Varianzen wird das kandidatenspezifische Eigenschafts-Integral neu bestimmt, gefolgt von der Neuanodnung der verbleibenden Kandidaten entsprechend dem Mittelpunkt bzw. dem linken Rand ihres Konfidenz-Intervalls. Durch die Erhöhung der Stützpunktanzahl wird nicht nur die Intervall-Position verändert, sondern es verringert sich auch der Intervalldurchmesser, der durch den Fehlerwert wie ε_{IB} bestimmt wird. Würde sich die Intervall-Position nicht verändern bei gleichzeitiger Verringerung des Intervalldurchmessers, so würde bei einer bestimmten Stützpunktanzahl die Intervall-Überlappung zwischen dem ersten Rang und dem oder den folgenden Rängen verschwinden. Verändert sich jedoch zusätzlich die Intervall-Position, so kann sich die Rangfolge der verbleibenden Kandidaten noch verändern. Als potentiell weitere Kandidaten bleiben jeweils der Kandidat auf dem ersten Rang und alle Kandidaten, die mit diesem eine Intervall-Überlappung besitzen, d.h. rechts davon liegende Intervalle, die keine Überlappung mit dieser Gruppe besitzen, werden ausgeschlossen.

Die Gesamtstrategie mit einer streng monoton wachsenden Stützpunktanzahl besteht darin, mit einer kleinen Stützpunktanzahl zu beginnen, die für alle Kandidaten aus KM^{t} angewendet wird. Entsprechend dem Eigenschafts-Integral und dessen Konfidenz-Intervall werden die ersten Elemente der Kandidatenliste KL^{t} ausgewählt, die als Gewinnercluster bezeichnet werden können, und die eine neue Kandidatenmenge KM'^{t} bilden. Für die darin enthaltenen Kandidaten wird eine neue erweiterte Stützpunktmenge gebildet, gefolgt von der kandidatenspezifischen Individualisierung der Stützpunktmenge und der Neuberechnung des Eigenschafts-Integrals und dessen Konfidenz-Intervalls. Kandidaten, die eine Intervall-Überlappung mit dem jeweiligen Gewinnercluster verlieren, werden entfernt und treten somit in der Kandidatenmenge der nächsten Iteration nicht mehr auf. Die Erhöhung der Stützpunktanzahl wird iterativ solange durchgeführt, bis der Gewinnercluster zum ersten Mal genau ein Element enthält, d.h. wenn zum ersten Mal genau ein eindeutiger Gewinner vorliegt.

Das Problem dieses Ansatzes besteht darin, dass nicht garantiert werden kann, dass nach einer vertretbaren Anzahl von Stützpunkten ein solcher einzelner Gewinner-Kandidat identifiziert werden kann. Vereinfachend könnte daher der erste Kandidat verwendet werden, wenn nach einer bestimmten Anzahl von Iterationen die Rangfolge der ersten Elemente unverändert bleibt.

5.2.3.2) Eigenschafts-Integrale mit weniger Stützpunkten durch Häufigkeitsverteilung

Ein alternativer Ansatz Eigenschafts-Integrale mit weniger Stützpunkten zu erzeugen, könnte sich durch die Verwendung von Häufigkeitsverteilungen ergeben. Hierzu wird wie bei der Monte-Carlo-Integration eine Menge von Stützpunkten gleichverteilt zufällig und unabhängig voneinander im Inputraum erzeugt, wobei die Anzahl der Stützpunkte deutlich geringer sein soll als bei einer Monte-Carlo-Integration, was durch die Bezeichnung $\mu_S' < \mu_S$ und $SM_K^{t'}$ beschrieben werden soll. Jedem Stützpunkt wird eine Outputschätzung $f(x_{S,i}^t \mid AM(x \mid M^t)$ und im Kontext der Bias-Integrale eine Bias-Schätzung $bias(x_{S,i}^t \mid AM(bias(x) \mid M^t))$ zugeordnet. Im Kontext eines Kandidaten $m_{K,j}^t$ aus der Kandidatenmenge KM^t wird die Stützpunktmenge zu $SM_{K,j}^{t'}$ individualisiert, indem jedem Stützpunkt eine Outputschätzung $f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\})$ zugeordnet wird. Damit kann das neue Bias-Quadrat $bias(x_{S,i}^t)_{K,j}^2$ an der Stelle des Stützpunktes $x_{S,i}^t$ geschätzt werden, indem $\Delta y_{S,i}^{t\wedge}$ durch $f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\} - f(x_{S,i}^t \mid AM(x \mid M^t)$ berechnet wird, und $\Delta y_{S,i}^{t\wedge}$ sowie $bias(x_{S,i}^t \mid AM(bias(x) \mid M^t))$ in die Bias-Quadrat-Gleichung

$$bias(x_{S,i}^t)_{K,j}'^2 = \Delta y_{S,i}^{t\wedge 2} + 2 * \Delta y_{S,i}^{t\wedge} * bias(x_{S,i}^t) + bias(x_{S,i}^t)^2 \tag{812}$$

eingesetzt werden. Bei einer Monte-Carlo-Integration wird der arithmetische Mittelwert der neuen Bias-Quadrate über alle Stützpunkte gebildet, während im weiteren zunächst die Häufigkeitsverteilung der neuen Bias-Quadrate bestimmt werden soll. Im ersten Schritt wird der Stützpunkt mit dem kleinsten Bias-Quadrat-Wert $bias(x_{S,min}^t)_{K,j}'^2$ und der Stützpunkt mit dem größten Bias-Quadrat-Wert $bias(x_{S,max}^t)_{K,j}'^2$ bestimmt, die zusammen ein Bias-Quadrat-Intervall $B^2I_{K,j}^t$ der individualisierten Stützpunktmenge $SM_{K,j}^{t'}$ bilden:

$$B^2I_{K,j}^t = [bias(x_{S,min}^t)_{K,j}'^2, bias(x_{S,max}^t)_{K,j}'^2], \text{ mit}$$
$$bias(x_{S,min}^t)_{K,j}'^2 = \min\{bias(x_{S,i}^t)_{K,j}'^2 \mid \forall m_{S,i}^t \in SM_{K,j}^{t'}\},$$
$$bias(x_{S,max}^t)_{K,j}'^2 = \max\{bias(x_{S,i}^t)_{K,j}'^2 \mid \forall m_{S,i}^t \in SM_{K,j}^{t'}\}. \tag{813}$$

Das Gesamtintervall wird in γ gleich breite Teilintervalle $B^2I(l)_{K,j}^t$, $l = 1, ..., \gamma$, zerlegt, das den Durchmesser $\Delta bias^2{}_{K,j}^t$ besitzt:

$$\Delta bias^2{}_{K,j}^t = (bias(x_{S,max}^t)_{K,j}'^2 - bias(x_{S,min}^t)_{K,j}'^2)/\gamma. \tag{814}$$

Es ergibt sich die geordnete Liste der γ Teilintervalle:

$$B^2IL_{K,j}^t = (B^2I(l)_{K,j}^t \mid l = 1, ..., \gamma), \text{ mit}$$
$$B^2I(1)_{K,j}^t = [bias(x_{S,min}^t)_{K,j}'^2, bias(x_{S,min}^t)_{K,j}'^2 + \Delta bias^2{}_{K,j}^t[,$$
$$B^2I(2)_{K,j}^t = [bias(x_{S,min}^t)_{K,j}'^2 + \Delta bias^2{}_{K,j}^t, bias(x_{S,min}^t)_{K,j}'^2 + 2 * \Delta bias^2{}_{K,j}^t[,$$
$$...,$$
$$B^2I(\gamma)_{K,j}^t = [bias(x_{S,min}^t)_{K,j}'^2 + (\gamma-1)\,\Delta bias^2{}_{K,j}^t, bias(x_{S,max}^t)_{K,j}'^2]. \tag{815}$$

Für jedes der Teilintervalle $B^2I(l)_{K,j}^t$ wird die Menge $SM(l)_{K,j}^{t'}$ der Stützpunkte aus $SM_{K,j}^{t'}$ ermittelt, deren Bias-Quadrat-Wert innerhalb des Teilintervalls liegt:

$$SM(l)_{K,j}^{t'} = \{m_{S,i}^t \in SM_{K,j}^{t'} \mid bias(x_{S,i}^t)_{K,j}'^2 \in B^2I(l)_{K,j}^t\}. \tag{816}$$

Die relative Anzahl der Stützpunkte in einem Teilintervall $B^2I(l)_{K,j}^t$ soll mit s(.) bezeichnet werden, d.h. die absolute Anzahl $\#SM(l)_{K,j}^{t'}$ wird durch die Gesamtzahl der Elemente in $SM_{K,j}^{t'}$, d.h. durch μ_s' dividiert:

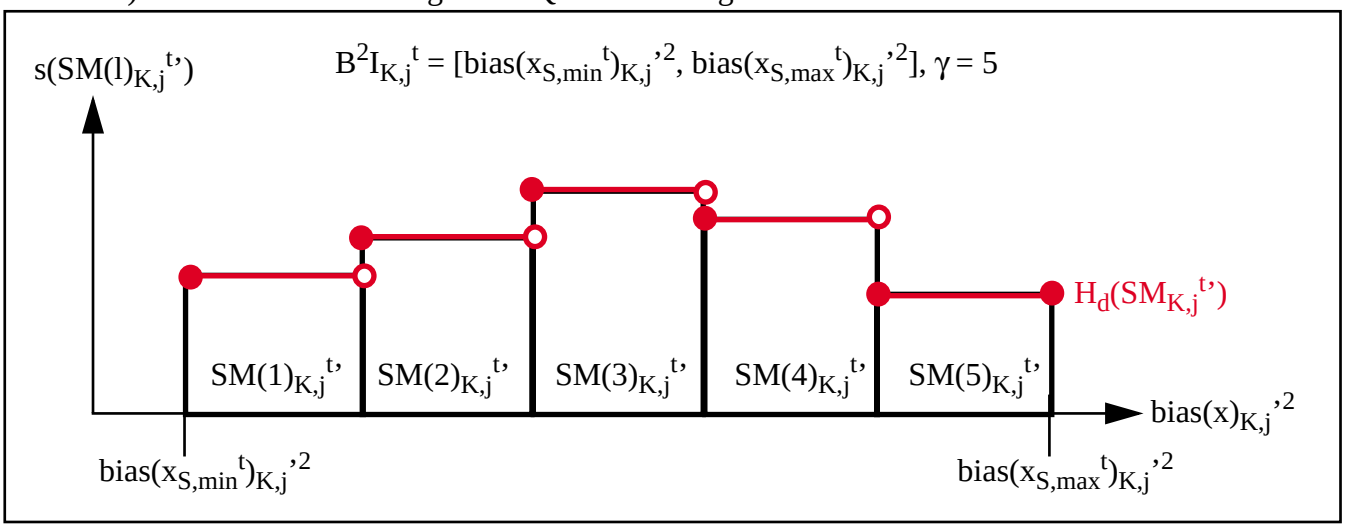
$$s(SM(l)_{K,j}^{t'}) := \#SM(l)_{K,j}^{t'}/\mu_s'. \tag{817}$$

Die diskrete Häufigkeitsfunktion $H_d(.)$ ordnet jedem Teilintervall $B^2I(l)_{K,j}^t$ die relative Anzahl $s(SM(l)_{K,j}^{t'})$ der Stützpunkte in der entsprechenden Teilmenge $SM(l)_{K,j}^{t'}$ zu:

$$H_d(SM_{K,j}^{t'})\text{: } SM(l)_{K,j}^{t'} \mapsto s(SM(l)_{K,j}^{t'}). \tag{818}$$

Wird jedem Punkt eines Intervalls der s(.)-Wert zugeordnet, so gelangt man zu einer punktweise nicht stetigen Stufenfunktion, wobei die Grenzen so gewählt sind, wie die Teilintervalle $B^2I(l)_{K,j}^t$ in der geordneten Liste $B^2IL_{K,j}^t$ (siehe Abb. 135)).

Abb. 135) Punktweise nicht stetige Bias-Quadrat-Häufigkeitsfunktion



Eine punktweise nicht stetige Dichtefunktion wird in eine geglättete, über der gesamten Definitionsmenge $[bias(x_{S,min}^t)_{K,j}'^2, bias(x_{S,max}^t)_{K,j}]$ stetige Dichtefunktion umgewandelt, indem Stützpunkte ausgewählt werden, die als Input in ein Glättungs- oder ein Regressionsverfahren verwendet werden. D.h. zunächst wird eine Anzahl von Outputwerten aus dem Intervall $[bias(x_{S,min}^t)_{K,j}'^2, bias(x_{S,max}^t)_{K,j}]$ spezifiziert, denen jeweils ihr s(.)-Wert zugeordnet wird. Sinnvoll ist es, aus jedem Teilintervall genau ein Wert zu spezifizieren, da alle Punkte innerhalb eines Teilintervalls den gleichen s(.)-Wert besitzen. Denkbar wäre es, die Mittelpunkte der Teilintervalle auszuwählen und ihnen den s(.)-Wert des Teilintervalls zuzuordnen. Die Mitte eines Intervalls $SM(1)_{K,j}^{t'}$ soll mit $SM(1)_{K,j}^{t'}/2 = bias(x_{S,min}^t)_{K,j}'^2 + (l\text{-}1) * \Delta bias^2{}_{K,j}^t + 1/2 * \Delta bias^2{}_{K,j}^t = bias(x_{S,min}^t)_{K,j}'^2 + (l\text{-}1/2) * \Delta bias^2{}_{K,j}^t$ bezeichnet werden, sodass dem Intervall $IM(l)_i^t$ der Stützpunkt $(SM(1)_{K,j}^{t'}/2, s(SM(l)_{K,j}^{t'}))$ zugeordnet wird. Mit Hilfe der γ Stützpunkte und einem Regressions- oder Glättungsverfahren wird eine stetige Funktion $H_s(SM_{K,j}^{t'})$ erzeugt, die jedem Outputwert $bias(x)_{K,j}'^2$ aus dem Intervall $B^2I_{K,j}^t = [bias(x_{S,min}^t)_{K,j}'^2, bias(x_{S,max}^t)_{K,j}'^2]$ einen relativen Häufigkeitswert $s(bias(x)_{K,j}'^2)$ zuordnet:

$$H_s(SM_{K,j}^{t'})\text{: } bias(x)_{K,j}'^2 \mapsto s(bias(x)_{K,j}'^2),\ bias(x)_{K,j}'^2 \in B^2I_{K,j}^t. \tag{819}$$

Eine Anwendung einer stetigen Funktion $H_s(SM_{K,j}^{t'})$ besteht darin, μ_s - μ_s' Zufallsexperimente durchzuführen, bei denen jeweils ein künstlicher Stützpunkt erzeugt wird, der ausschließlich aus einem Bias-Quadrat-Wert besteht. Diese Biaswerte werden zusammen mit den Biaswerten der Stützpunktmenge $SM_{K,j}^{t'}$ zur Approximation des Integrals $IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$ verwendet, indem der arithmetische Mittelwert der Bias-Quadrate gebildet wird. Der Effizienzvorteil gegenüber der Bildung von μ_s Stützpunkten besteht darin, dass die Ermittlung eines Bias-Quadrat-Wertes aus einer Verteilung $H_s(SM_{K,j}^{t'})$ wesentlich geringere Rechen- und Speicherressourcen erfordert, als die reguläre Bestimmung eines Stützpunktes $m_{S,i}^t$, bei dem zwei Output-Schätzungen $f(x_{S,i}^t \mid AM(x \mid M^t)$ und $f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\})$, sowie eine Bias-Schätzung $bias(x_{S,i}^t \mid AM(bias(x) \mid M^t))$ berechnet werden müssen.

Anstatt eine diskrete Anzahl μ_s - μ_s' von Zufallsexperimenten zu erzeugen, kann mit Hilfe des Integrals über einer Funktion mit absoluten Häufigkeiten der Fall beschrieben werden, dass eine Anzahl $\mu_s \to \infty$ Stützpunkte verwendet wird. Dies ist dann relevant, wenn die Verteilungsfunktion in einer analytisch integrierbaren Form vorliegt, d.h. wenn keine numerische Integration durchgeführt wird, die keinen Effizienzvorteil gegenüber der numerischen Approximation von $IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$ besitzen würde.

5.2.3.3) Deterministische Integration im average case setting

Anstatt eine stochastische Integration wie eine Monte-Carlo-Approximation durchzuführen, wurde im Rahmen der Forschung zur Information-based Complexity (siehe Traub et al. (1988[338]), Traub & Wozniakowski (1994[338]), Wozniakowski (1996a[372], b[373]), Sloan & Wozniakowski (1996[314]), Werschulz & Wozniakowski (2000[358])) gezeigt, dass deterministische Verfahren existieren, die im typischen Fall, d.h. im average case setting, eine Integration mit einer Fehlerkomplexität von $1/\varepsilon_I$ durchführen können im Gegensatz zu der Fehlerkomplexität von bis zu $1/\varepsilon_I^2$ beim Monte-Carlo-Verfahren. Beim average case setting kann für einen speziellen Fall nicht garantiert werden, dass der richtige Integralwert wirklich innerhalb eines Intervalls $[IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge - \varepsilon_I, IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge + \varepsilon_I]$ liegt. Da jedoch viele dieser Integrale berechnet werden, um aus einer Kandidatenmenge einen Gewinner zu ermitteln, liegen die meisten Integralwerte innerhalb der erzeugten Intervalle, sodass eine Rangbildung sinnvoll anwendbar ist.

Als Voraussetzung der deterministischen Verfahren gilt, dass die zu integrierende Funktion aus der Grundgesamtheit einer Wiener-Verteilung stammen muss, was eine Verallgemeinerung der Gauss-Verteilung auf Räume von Funktionen darstellt. Die deterministischen Verfahren können auch dann eingesetzt werden, wenn die Glattheit der Integrationsfunktion gleich Null ist, d.h. wenn $r = 0$, was der Fall ist, wenn die Funktion stetig jedoch nirgends differenzierbar ist. Die Integrationsfunktion ist im betrachteten Kontext der LWR-Modelle eine gewichtete Summe von Kernelfunktionen, die nach Cohn (1995[73]) und Cohn et al. (1995) als Gaussfunktion $h(x_i,x_j)$ bzw. $h(d_X(x_i,x_j))$ mit einem Glättungsfaktor k definiert wird. Die Integrationsfunktion ist somit stetig und auf Grund der Eigenschaft der enthaltenen e-Funktion unendlich oft differenzierbar, sodass die Anwendbarkeit der deterministischen Verfahren gewährleistet ist.

Die Integrations-Verfahren werden als deterministisch bezeichnet, da die Stützpunkte der Integration nicht mehr gleichverteilt zufällig im Inputraum erzeugt werden wie bei der Monte-Carlo-Integration,

sondern es werden deterministisch Stützpunkte erzeugt, wobei Hammersley-Punkte oder hyperbolische Kreuzungspunkte verwendet werden können (Traub & Wozniakowski (1994:67[338])).

5.2.3.4) Weniger Eigenschafts-Integrale durch Kandidatencluster und Approximation

Ein anderer Ansatz, um den Aufwand einer Iteration zu verringern, besteht darin, nicht allen Kandidaten aus KM^t ein Eigenschafts-Integral durch eine Integral-Approximation zuzuordnen, d.h. es wird eine Teilmenge von Kandidaten selektiert, für die ein Eigenschafts-Integral auf die oben dargestellte Weise durch eine Monte-Carlo-Approximation berechnet wird, und die restlichen Kandidaten erhalten eine Schätzung ihres Eigenschafts-Integrals auf der Basis der approximierten Werte. Dieser Ansatz wird somit durch zwei Faktoren spezifiziert:

1) die disjunkte Zerlegung der Kandidatenmenge KM^t in $KM_{MC}{}^t$ und eine Restmenge, wobei in $KM_{MC}{}^t$ die Kandidaten liegen, die mit der Monte-Carlo-Approximation berechnet werden.
2) ein Approximationsverfahren, mit dem die Integralwerte der Kandidaten aus der Restmenge geschätzt werden.

Die Schätzungen sollen durch ein LWR-Verfahren erfolgen, d.h. sie sind distanzabhängig im Inputraum d.h. im DVR. Zwei Kandidaten $m_{K,j}{}^t$ und $m_{K,i}{}^t$, deren Inputvektoren $x_{K,j}{}^t$ und $x_{K,i}{}^t$ nahe beieinander liegen, erhalten tendenziell ähnlichere Werte des Eigenschafts-Integrals wie z.B. $IB[AM(x \mid M^t \cup \{m_{K,j}{}^t\})]_{LWR}\wedge$ und $IB[AM(x \mid M^t \cup \{m_{K,i}{}^t\})]_{LWR}\wedge$, als wenn die beiden Inputvektoren weiter auseinander liegen.

Bei der Zerlegung von KM^t in $KM_{MC}{}^t$ und eine Restmenge, soll diese Restmenge bei der Verwendung einer LWR-Approximation als $KM_{LWR}{}^t$ bezeichnet werden. Als Zerlegungsverfahren können stochastische Verfahren wie das zufällige Ziehen ohne Zurücklegen eingesetzt werden, oder es werden Kandidatencluster gebildet, aus denen jeweils ein Element bzw. wenige Elemente als Prototypen deterministisch ausgewählt werden. Von besonderem Interesse ist dabei eine vorhandene Strukturierung der Kandidaten auf der Basis ihrer Inputvektoren, die übernommen werden könnte. Dies ist der Fall bei Dokumentvektoren, die durch eine globale GNG-SOM N_{DV} strukturiert sind, wobei die Kandidatenmenge als $DVM_{prim}{}^t$ auf der Basis der Positionen der Dokumentvektoren bezüglich der Voronoi-Regionen ausgewählt werden. In der Iteration t=0 wird die lokale Dokumentvektorenmenge $M_{DV,s(1|i)}{}^{t=0}$ des Gewinner-Neurons $n_{DV,s(1|i)}{}^{t=0}$ bezüglich des Queryvektors q_i als Ergebnismenge $DVM_{prim}{}^{t=0}$ ausgewählt. Da alle Elemente aus $DVM_{prim}{}^{t=0}$ präsentiert werden und für t=0 noch kein Relevanz-Approximationsmodell vorliegt, werden keine Eigenschafts-Integrale berechnet. Nach der Bewertung der Elemente aus $DVM_{prim}{}^{t=0}$ liegt $M_{DV(m)}{}^{t=0}$ vor, das $AM(rel(x) \mid M_{DV(m)}{}^{t=0})$ direkt und $AM(rel(x) \mid N_{DV}{}^{t=0})$ indirekt spezifiziert. Mit der Einführung der Neuronenmenge $N_{DV,bew}{}^t$, in der alle Neurone aus N_{DV} liegen, denen bislang mindestens ein Dokumentvektor mit einer richtigen Relevanzbewertung zugeordnet wurde, wird in der nächsten Iteration t=1 die nächste Ergebnis- bzw. Kandidatenmenge $DVM_{prim}{}^{t=1}$ festgelegt als die Vereinigung aller Dokumentvektorenmengen der Neurone aus $N_{DV,bew}{}^{t=0}$ und ihrer Nachbarneurone korrigiert um die Dokumentvektoren, denen bereits ein richtiger Relevanzwert zugeordnet wurde:

$$DVM_{prim}^{t=1} = \{\bigcup_r M_{DV,r}\} \setminus \{x_j \mid \forall m_j \in M_{DV(m)}^{\leq t-1}\},$$
$$\forall n_{DV,r} \in N(d_G \leq 1 \mid G_{DV})_k, \forall n_{DV,k} \in N_{DV,bew}^{t=0}. \quad (820)$$

Da $N_{DV,bew}^{t=0} = \{n_{DV,s(1|i)}\}$ ist, und allen Dokumentvektoren aus $M_{DV,s(1|i)}^{t=0}$ ein richtiger Relevanzwert zugeordnet wurde, sind alle Nachbarneurone an der Erzeugung von $DVM_{prim}^{t=1}$ beteiligt, d.h. alle $n_{DV,r}$ aus $N(d_G=1 \mid G_{DV})_{s(1|i)}$. Aus jeder Dokumentvektorenmenge $M_{DV,r}$ wird ein oder mehrere Elemente ausgewählt, denen ein Eigenschafts-Integral durch die Monte-Carlo-Methode berechnet wird, und die zusammen die Menge bzw. $DVM_{MC,prim}^{t=1}$ bilden. Als Auswahlkriterium wird die Distanz eines Dokumentvektors zu dem Prototypen-Gewichtsvektor $w(x_{DV})_r$ des Neurons $n_{DV,r}$ verwendet, d.h. es kann zunächst eine Gewinnerliste $G(w(x_{DV})_r \mid M_{DV,r})$ gebildet werden, welche die Elemente aus $M_{DV,r}$ nach steigender Distanz zu $w(x_{DV})_r$) ordnet:

$$G(w(x_{DV})_r \mid M_{DV,r}) = (x_{r,j} \mid d_{DVR}(x_{r,j}, w(x_{DV})_r) < d_{DVR}(x_{r,j+1}, w(x_{DV})_r), \forall x_{r,j} \in M_{DV,r}) \quad (821)$$

Das erste Element $x_{r,j=1}$ dieser Liste besitzt die geringste Distanz zu $w(x_{DV})_r$, und kann als Substitut ausgewählt werden, d.h. der Vektor repräsentiert die Verteilung der Dokumentvektoren in der Voronoi-Region des Neurons $n_{DV,r}$ am besten. Die Menge $DVM_{MC,prim}^{t=1}$ kann aus diesen Elementen gebildet werden:

$$DVM_{MC,prim}^{t=1} = \{x_{r,j=1} \mid \forall n_{DV,r} \in N(d_G=1 \mid G_{DV})_{s(1|i)}\}. \quad (822)$$

Werden die Dokumentvektoren als Komponente von Stimuli betrachtet, so kann die Kandidatenmenge $KM_{MC}^{t=1}$ entsprechend gebildet werden:

$$KM_{MC}^{t=1} = \{m_{r,j=1} = (x_{r,j=1}) \mid \forall x_{r,j=1} \in DVM_{MC,prim}^{t=1}\}. \quad (823)$$

Jedem Kandidaten wird eine individuelle Stützpunktmenge $SM_{r,j=1}^{t=1}$ zugeordnet, mit der eine Eigenschafts-Integral-Approximation durch das Monte-Carlo-Verfahren berechnet werden kann. Wird ein Bias-Integral unterstellt, so ergibt sich für das zu testende Relevanz-Approximationsmodell $AM(rel(x) \mid M_{DV(m)}^{\leq t} \cup \{m_{r,j=1}^{t=1}\})$ die Bewertung $IB[AM(rel(x) \mid M_{DV(m)}^{\leq t} \cup \{m_{r,j=1}^{t=1}\})]_{MC}\wedge$, die vereinfachend mit $IB_{MC,r,j=1}^{t=1\wedge}$ bezeichnet werden soll. Die Datenstruktur der Kandidaten wird entsprechend erweitert zu:

$$KM_{MC}^{t=1} = \{m_{r,j=1}^{t=1} = (x_{r,j=1}^{t=1}, SM_{r,j=1}^{t=1}, IB_{MC,r,j=1}^{t=1\wedge}) \mid \forall x_{r,j=1} \in DVM_{MC,prim}^{t=1}\}. \quad (824)$$

Der n-dimensionale Dokumentvektor $x_{r,j=1}$ wird als Inputvektor und der Skalar $IB_{MC,r,j=1}^{t=1\wedge}$ als eindimensionaler Outptvektor verwendet, sodass ein instanzbasiertes Approximationsmodell aufgebaut werden kann, das als Input Dokumentvektoren aufnimmt, und als Output eine Schätzung eines Bias-Integrals ausgibt, d.h. es wird das Bias-Integral des instanzbasierten Relevanz-Approximationsmodells geschätzt, das $m_{r,j=1}$ neben den Elementen aus $M_{DV(m)}^{\leq t}$ aufnimmt. Dieses instanzbasierte Bias-Integral-Approximationsmodell soll mit $AM(IB \mid KM_{MC}^{t=1})$ bezeichnet werden, da IB die zu schätzende Variable ist und die Elemente aus $KM_{MC}^{t=1}$ die Instanzen des Modells aufbauen, wobei von einem LWR-Approximationsverfahren ausgegangen wird. Die Dokumentvektoren $x_j^{t=1}$ aus der Restmenge $DVM_{prim}^{t=1} \setminus DVM_{MC,prim}^{t=1} := DVM_{LWR,prim}^{t=1}$ erhalten durch $AM(IB \mid KM_{MC}^{t=1})$ eine Bias-Integral-Schätzung:

$$IB(x_j^{t=1} \mid AM(IB \mid KM_{MC}^{t=1})) = IB_{LWR,j}^{t=1\wedge} = 1/\nu_j \sum_k h(d_X(x_j^{t=1}, x_k^{t=1})) * IB_{MC,k}^{t=1\wedge},$$
$$\nu_j = \sum_k h(d_X(x_j^{t=1}, x_k^{t=1})), \forall m_k^{t=1} \in KM_{MC}^{t=1}. \quad (825)$$

Analog den Dokumentvektoren aus $DVM_{MC,prim}^{t=1}$ können den Dokumentvektoren aus $DVM_{LWR,prim}^{t=1}$ eine Kandidatenmenge zugeordnet werden:

$$KM_{LWR}^{t=1} = \{m_j^{t=1} = (x_j^{t=1}, IB_{LWR,j}^{t=1\wedge}) \mid \forall x_j \in DVM_{LWR,prim}^{t=1}\}. \quad (826)$$

Besitzen alle Elemente aus $DVM_{prim}^{t=1}$ eine Bias-Integral-Schätzung, unabhängig ob diese durch eine Monte-Carlo-Approximation oder durch eine LWR-Schätzung gebildet wurde, so werden beide Schätzungen als gleichwertig behandelt, und es wird eine Kandidatenliste nach steigendem Wert der Bias-Integral-Schätzungen gebildet. In der gleichen Weise kann die sekundäre Dokumentvektorenliste $DV_{sec}^{t=1}$ aus der Menge $DVM_{prim}^{t=1}$ gebildet werden:

$$KL^{t=1} = (m_j^{t=1} \mid IB_j^{t=1\wedge} < IB_{j+1}^{t=1\wedge}, \forall m_j^{t=1} \in KM^{t=1}),$$
$$DV_{sec}^{t=1} = (x_j^{t=1} \mid IB_j^{t=1\wedge} < IB_{j+1}^{t=1\wedge}, \forall x_j^{t=1} \in DVM_{prim}^{t=1}). \quad (827)$$

Entsprechend der strengen Variante der Vorgehensweise beim direkten und ausschließlichen aktiven Lernen wird das erste Element der Liste $DV_{sec}^{t=1}$ ausgewählt, und das korrespondierende Dokument wird dem Agenten präsentiert, der eine Relevanzbewertung abgibt. Der neue Stimulus erweitert die vorhandene Stimulusmenge $M_{DV(m)}^{\leq t-1}$, die zu $M_{DV(m)}^{\leq t}$ aktualisiert wird. Ein ausschließliches aktives Lernen im Rahmen eines Retrievals ist jedoch nicht sinnvoll, und wird immer von dem Kriterium der Relevanz-Maximierung begleitet, d.h. es wird durch einen Prozess Dokumentvektoren durch das Relevanz-Maximierungs-Kriterium ermittelt, und genau ein Dokumentvektor durch das direkte Modell-Maximierungs-Kriterium (siehe Abschnitt 5.3)).

Auf eine Darstellung mehrerer Bewertungskriterien und ihrer Integralbildung wie z.B. die Kombination von Bias- und Output-Varianz-Integralen soll verzichtet werden.

Sinnvoll aus Effizienzsicht ist die Erzeugung eines Bias-Integral-Approximationsmodells dann, wenn der Aufwand der Modellerzeugung und der Modellanwendung kleiner ist als die Summe aller Bias-Integral-Approximationen durch das Monte-Carlo-Verfahren. Die Modellerzeugung besitzt bei instanzbasierten Modellen keinen zusätzlichen Aufwand, da die Bestimmung der Bias-Integrale der Elemente aus KM_{MC}^{t} in beiden Verfahrensvarianten anfallen. Somit verbleibt der Vergleich des Aufwandes für eine Integral-Schätzung nach der Monte-Carlo-Methode und nach der Integral-Approximationsmodell-Methode. Bei der Monte-Carlo-Methode müssen für μ_S Stützpunkte je zwei Relevanzschätzungen und eine Bias-Schätzung durch ein LWR-Modell erzeugt werden. Wird jeweils ein instanzbasiertes Modell auf der Basis aller bekannten Stimuli unterstellt, d.h. auf $M_{DV(m)}^{\leq t}$ für t-1, so werden $3 * \mu_S * \#M_{DV(m)}^{\leq t}$ gewichtete Summanden gebildet und addiert, gefolgt von der Mittelwertsbildung der Bias-Schätzungen zu einem Bias-Integral. Bei der Integral-Approximationsmodell-Methode wird eine Integralschätzung durch $\#KM_{MC}^{t}$ gewichtete Summanden gebildet. Die Effizienzverbesserung des Integral-Approximationsmodell-Methode kann somit mehr als zwei Zehnerpotenzen betragen.

Der Effektivitätsnachteil der Integral-Approximationsmodell-Methode besteht jedoch darin, dass den Schätzungen $IB_{LWR,j}^{t=1\wedge}$ kein Fehler ε_I auf die gleiche Weise zuordenbar ist, wie den Schätzungen $IB_{MC,j}^{t=1\wedge}$ durch die Monte-Carlo-Methode. Dem Integral-Approximationsmodell könnten jedoch empirisch Fehlermodelle zugeordnet werden, indem ausgewählten Kandidaten beide Schätzungstypen zugeordnet werden und indem das LWR-Verfahren mit einem Gewinnerlisten-Verfahren (siehe Abschnitt 2.3.8)) kombiniert wird. Als Kandidaten bieten sich die Elemente aus $KM_{MC}^{t=1}$ an, die eine entsprechend erweiterte Datenstruktur erhalten:

$$KM_{MC}^{t=1} = \{m_{r,j=1}^{t=1} = (x_{r,j=1}^{t=1}, SM_{r,j=1}^{t=1}, IB_{MC,r,j=1}^{t=1\wedge}, IB_{LWR,r,j=1}^{t=1\wedge}) \mid \forall x_{r,j=1} \in DVM_{MC,prim}^{t=1}\}, \text{ mit}$$
$$IB_{LWR,r,j=1}^{t=1\wedge} = 1/\nu_{r,j=1} \sum_k h(d_X(x_{r,j=1}^{t=1}, x_{k,j=1}^{t=1})) * IB_{MC,k,j=1}^{t=1\wedge},$$
$$\nu_{r,j=1} = \sum_k h(d_X(x_{r,j=1}^{t=1}, x_{k,j=1}^{t=1})), \forall m_{k,j=1}^{t=1} \in KM_{MC}^{t=1} \setminus \{m_{r,j=1}^{t=1}\}. \quad (828)$$

Alternativ sind effektivere Schätzungen von $IB_{LWR,r,j=1}^{t=1\wedge}$ durch Resampling-Verfahren der Stützpunkte aus der Grundmenge $KM_{MC}^{t=1} \setminus \{m_{r,j=1}^{t=1}\}$ möglich, wenn eine Mindestanzahl von Stützpunkten vorliegt, worauf jedoch nicht explizit eingegangen werden soll.

An den Stellen $x_{r,j=1}^{t=1}$ lässt sich somit die Bias und die Output-Varianz bestimmen, wobei die Output-Varianz durch das Verfahren von Cohn et al. (1995[74]) berechnet werden kann:

$$bias(x_{r,j=1}^{t=1})_{IB} = IB_{MC,r,j=1}^{t=1\wedge} - IB_{LWR,r,j=1}^{t=1\wedge},$$
$$\sigma(x_{r,j=1}^{t=1})_{IB}^2 = 1/\nu_{r,j=1} \sum_k h(d_X(x_{r,j=1}^{t=1}, x_{k,j=1}^{t=1})) * (IB_{MC,k,j=1}^{t=1\wedge} - IB_{LWR,r,j=1}^{t=1\wedge})^2,$$
$$\nu_{r,j=1} = \sum_k h(d_X(x_{r,j=1}^{t=1}, x_{k,j=1}^{t=1})), \forall m_{k,j=1}^{t=1} \in KM_{MC}^{t=1} \setminus \{m_{r,j=1}^{t=1}\}. \quad (829)$$

Die Bias- und Output-Varianz-Werte werden in die Datenstruktur der Kandidaten aus $KM_{MC}^{t=1}$ aufgenommen:

$$KM_{MC}^{t=1} = \{m_{r,j=1}^{t=1} = (x_{r,j=1}^{t=1}, SM_{r,j=1}^{t=1}, IB_{MC,r,j=1}^{t=1\wedge}, IB_{LWR,r,j=1}^{t=1\wedge}, bias(x_{r,j=1}^{t=1})_{IB}, \sigma(x_{r,j=1}^{t=1})_{IB}^2) \mid \forall x_{r,j=1} \in DVM_{MC,prim}^{t=1}\}. \quad (830)$$

Analog dem LWR-Approximationsmodell $AM(IB \mid KM_{MC}^{t=1})$ der Bias-Integral-Schätzung kann somit ein LWR-Approximationsmodell $AM(bias_{IB} \mid KM_{MC}^{t=1})$ der Bias, der Bias-Integral-Schätzung sowie ein LWR-Approximationsmodell $AM(\sigma_{IB}^2 \mid KM_{MC}^{t=1})$ der Output-Varianz der Bias-Integral-Schätzung gebildet werden, wobei alle drei Modelle instanzbasiert sind.

Wird ein Kandidat $m_j^{t=1}$ aus der Menge $KM_{LWR}^{t=1}$ betrachtet, so wird diesem eine Bias-Integral-Schätzung $IB_{LWR,j}^{t=1\wedge}$, eine Schätzung des Fehlers der Integral-Schätzung in Form der Bias $bias(x_j^{t=1})_{IB}\wedge$, sowie die Varianz $\sigma(x_j^{t=1})_{IB}^2$ der Integral-Schätzung durch die drei LWR-Approximationsmodelle $AM(IB \mid KM_{MC}^{t=1})$, $AM(bias_{IB} \mid KM_{MC}^{t=1})$ und $AM(\sigma_{IB}^2 \mid KM_{MC}^{t=1})$ zugeordnet. Bzw. es werden die beiden Modelle $AM(IB \mid KM_{MC}^{t=1})$ und $AM(bias_{IB} \mid KM_{MC}^{t=1})$ verwendet und die Output-Varianz wird durch das Verfahren aus Cohn et al. (1995[74]) direkt berechnet, sodass anstatt der Schätzung $\sigma(x_j^{t=1})_{IB}^{2\wedge}$ der Wert $\sigma(x_j^{t=1})_{IB}^2$ verwendet wird:

$$IB_{LWR,j}^{t=1\wedge} = 1/\nu_j \sum_k h(d_X(x_j^{t=1}, x_k^{t=1})) * IB_{MC,k,j=1}^{t=1\wedge},$$
$$bias(x_j^{t=1})_{IB}\wedge = 1/\nu_j \sum_k h(d_X(x_j^{t=1}, x_k^{t=1})) * bias(x_{k,j=1}^{t=1})_{IB},$$
$$\sigma(x_j^{t=1})_{IB}^{2\wedge} = 1/\nu_j \sum_k h(d_X(x_j^{t=1}, x_k^{t=1})) * \sigma(x_{k,j=1}^{t=1})_{IB}^2, \text{ bzw.}$$
$$\sigma(x_j^{t=1})_{IB}^2 = 1/\nu_j \sum_k h(d_X(x_j^{t=1}, x_k^{t=1})) * (IB_{MC,k,j=1}^{t=1\wedge} - IB_{LWR,j}^{t=1\wedge})^2,$$
$$\nu_j = \sum_k h(d_X(x_j^{t=1}, x_k^{t=1})), \forall m_k^{t=1} \in KM_{MC}^{t=1}. \quad (831)$$

Die Stimulusobjekte $m_j^{t=1}$ aus $KM_{LWR}^{t=1}$ erhalten somit die erweiterte Datenstruktur:

$$KM_{LWR}^{t=1} = \{m_j^{t=1} = (x_j^{t=1}, IB_{LWR,j}^{t=1\wedge}, bias(x_j^{t=1})_{IB}\wedge, \sigma(x_j^{t=1})_{IB}^2) \mid \forall x_j \in DVM_{LWR,prim}^{t=1}\}. \quad (832)$$

Analog der Vorgehensweisen bei den fehlerkorrigierten Relevanzschätzungen in Abschnitt 4.6.1.1) kann die Schätzung $IB_{LWR,j}^{t=1\wedge}$ durch $bias(x_j^{t=1})_{IB}\wedge$ korrigiert werden, sodass der sich ergebende Bias-Integralwert $IB_{LWR,j}^{t=1\wedge}$' als Mittelpunkt eines Integral-Intervalls $KI_{IB(LWR,j)}$ betrachtet werden kann, dessen Grenzen durch die Standardabweichung $\sigma(x_j^{t=1})_{IB}$ gebildet werden. Der Skalierungsfaktor α, mit dem die Standardabweichung im allgemeinen Fall multipliziert wird, wird vereinfachend als $\alpha := 1$ gesetzt:

$$KI_{IB(LWR,j)} = [IB_{LWR,j}^{t=1\wedge}' - \sigma(x_j^{t=1})_{IB}, IB_{LWR,j}^{t=1\wedge}' + \sigma(x_j^{t=1})_{IB}], \text{ mit}$$
$$IB_{LWR,j}^{t=1\wedge}' = IB_{LWR,j}^{t=1\wedge} + bias(x_j^{t=1})_{IB}\wedge. \quad (833)$$

Die Datenstruktur der Kandidaten aus $KM_{LWR}^{t=1}$ kann entsprechend umstrukturiert werden, indem nur noch der Dokumentvektor $x_j^{t=1}$ und das Konfidenz-Intervall verwendet werden, wenn das Intervall die einzige Grundlage für die Bildung des Rankings der Kandidaten aus $KM_{LWR}^{t=1}$ und $KM_{MC}^{t=1}$ sein soll:

$$KM_{LWR}^{t=1} = \{m_j^{t=1} = (x_j^{t=1}, KI_{IB(LWR,j)}) \mid \forall x_j \in DVM_{LWR,prim}^{t=1}\}. \quad (834)$$

Dabei ist zu beachten, dass die Elemente aus $KM_{MC}^{t=1}$ zwar Bias- und Output-Varianzwerte, jedoch noch keine Konfidenz-Intervalle besitzen, wobei festzulegen ist, wie diese im gegebenen Kontext berechnet werden sollen. Im vorangegangenen Abschnitt wurden Konfidenz-Intervalle für Kandidaten, die mit Hilfe der Monte-Carlo-Approximation Bias-Integrale erhalten haben, durch den Fehler ε_{IB} erzeugt, sodass für ein Element $m_{r,j=1}^{t=1}$ aus $KM_{MC}^{t=1}$ ein Intervall $KI_{IB(MC,r,j=1)}$ berechnet werden kann:

$$KI_{IB(MC,r,j=1)} = [IB_{MC,r,j=1}^{t=1\wedge} - \varepsilon_{IB}, IB_{MC,r,j=1}^{t=1\wedge} + \varepsilon_{IB}]. \quad (835)$$

Andererseits kann ein Intervall auch mit der Standardabweichung berechnet werden, die sich aus der Bias-Quadrat-Varianz $\sigma(x_{r,j=1}^{t=1})_{IB}^2$ ergibt:

$$KI_{IB(MC,r,j=1)} = [IB_{MC,r,j=1}^{t=1\wedge} - \sigma(x_{r,j=1}^{t=1})_{IB}, IB_{MC,r,j=1}^{t=1\wedge} + \sigma(x_{r,j=1}^{t=1})_{IB}]. \quad (836)$$

Die Varianz $\sigma(x_{r,j=1}^{t=1})_{IB}^2$ kann berechnet werden, indem das Integral $IB_{MC,r,j=1}^{t=1\wedge}$ als Mittelwert der Bias-Quadratwerte über alle Monte-Carlo-Stützpunkte aus $SM_{K,r,j=1}^t$ verstanden wird, und die Bias-Quadrat-Varianz aus der Summe der quadrierten Differenzen aus dem Mittelwert und den Bias-Quadratwerten berechnet wird:

$$IB_{MC,r,j=1}{}^{t=1\wedge} = 1/\mu_S * \sum_i bias(x_{S,i}{}^t)_{K,r,j=1}{}^2,$$
$$\sigma(x_{r,j=1}{}^{t=1})_{IB}{}^2 = 1/\mu_S * \sum_i (IB_{MC,r,j=1}{}^{t=1\wedge} - bias(x_{S,i}{}^t)_{K,r,j=1}{}^2)^2, \forall m_{S,i}{}^t \in SM_{K,r,j=1}{}^t. \quad (837)$$

Unabhängig wie ein Konfidenz-Intervall bestimmt wird, es wird Teil der Datenstruktur der Elemente aus der Kandidatenteilmenge $KM_{MC}{}^{t=1}$:

$$KM_{MC}{}^{t=1} = \{m_{r,j=1}{}^{t=1} = (x_{r,j=1}{}^{t=1}, SM_{r,j=1}{}^{t=1}, IB_{MC,r,j=1}{}^{t=1\wedge}, IB_{LWR,r,j=1}{}^{t=1\wedge}, bias(x_{r,j=1}{}^{t=1})_{IB},$$
$$\sigma(x_{r,j=1}{}^{t=1})_{IB}{}^2, KI_{IB(MC,r,j=1)}) \mid \forall x_{r,j=1} \in DVM_{MC,prim}{}^{t=1}\}. \quad (838)$$

Somit sind allen Elementen aus $KM_{MC}{}^{t=1}$ und $KM_{LWR}{}^{t=1}$ ein Bias-Integral-Konfidenz-Intervall zugeordnet, sodass die Elemente entsprechend dem Intervall-Mittelwert oder der linken Intervall-Grenze geordnet werden können. Aus dieser Liste wird das erste bzw. die ersten Elemente ausgewählt, d.h. es werden die Elemente mit den kleinsten, geschätzten Bias-Integralen ausgewählt. Bei der Verwendung von Intervallen sind angepasste Selektionskriterien notwendig, wie z.B. der Auswahl des ersten Elementes und aller anderen Elemente, die mit dem Intervall des ersten Elementes eine Überschneidung besitzen.

5.2.3.5) Kombination von stetiger und diskreter Vorgehensweise

Im Abschnitt 5.1.4) wurde zwischen der diskreten und der stetigen Vorgehensweise beim aktiven Lernen mit einer offenen Stimulusmenge unterschieden. Diskrete Verfahren untersuchen eine endliche Kandidatenmenge, während stetige Verfahren den Inputvektor aus dem gesamten Inputraum suchen, der zu dem Modell mit der erwarteten besten Modellqualität führen wird. Bei einem stetigen Verfahren kann ein deterministisches lokales Suchverfahren wie ein Gradientenabstieg durchgeführt werden, das die erste Ableitung einer Bias-Quadratfunktion nutzt (siehe Cohn (1995[73]), Cohn et al. (1995[74])), es können auch Ableitungen höherer Ordnung verwendet werden, wenn die betreffende Funktion mehrfach ableitbar ist, wie z.B. das Levenberg-Marquardt-Verfahren (Press et al. (1988[264])).

Die stetige und die diskrete Vorgehensweise können kombiniert werden, indem zunächst durch das stetige Verfahren ein Inputvektor als lokales Optimum spezifiziert wird. Zu diesem Inputvektor wird das Element aus der Kandidatenmenge KM^t ermittelt, dessen Inputvektor-Komponente den geringsten Abstand d_X im Fall des aktiven Lernens bzw. im Kontext des Retrievals den geringsten Abstand d_{DVR} von dem Optimum besitzt,. Alternativ wird eine Kandidatenliste KL^t aus KM^t erzeugt, indem die Kandidaten nach steigender Distanz zu dem Optimum sortiert werden. Je nach dem verwendeten Selektionskriterium wird der erste Kandidat bzw. die ersten Kandidaten zum aktiven Lernen ausgewählt, d.h. ihnen wird durch einen externen Prozess, wie die Bewertungen durch einen Agenten, ein Output- bzw. Relevanzwert zugeordnet. Effizient ist diese Vorgehensweise, wenn das iterative stetige Verfahren für deutlich weniger Punkte aus dem Inputraum ein Eigenschafts-Integral berechnen muss, als Kandidaten in KM^t vorhanden sind.

Ein Effektivitätsnachteil dieser Vorgehensweise besteht darin, dass ein einfacher Vergleich mit Hilfe der Distanzen im Inputraum nur bei einer monotonen bzw. lokal monotonen Bias-Quadrat-Funktion effektiv ist. Besitzt die Bias-Quadratfunktion lokale Minima, so können Kandidaten existieren, die im Inputraum

weiter von dem Punkt entfernt sind, der beim stetigen Verfahren ermittelt wurde, die jedoch andere Bias-Quadratwerte besitzen, als die Rangfolge der Distanzen dies vermuten lässt. Es handelt sich dabei im Prinzip um die gleiche Argumentation, die bei der Einführung expliziter Relevanz-Approximationsmodelle im Vergleich zur impliziten Schätzung durch die Distanz-Relevanzfunktion dargelegt wurde (siehe Abschnitt 4.1.2)).

Wird berücksichtigt, dass der Gradienten-Abstieg oder das Levenberg-Marquardt-Verfahren iterativ ist, so wurden für mehrere Punkte im Inputraum das Eigenschafts-Integral berechnet. Diese Punkte können als Stützpunkte einer LWR-Approximation verwendet werden, welche die Eigenschafts-Integrale an der Stelle von Kandidaten aus KM^t schätzen, wie dies im Prinzip im vorangegangenen Abschnitt 5.2.3.4) mit Stützpunkten aus $KM_{MC}{}^t$ beschrieben wurde.

5.2.3.6) Neurone als Stützpunkte der Approximation und der Integration

Grundlage der Monte-Carlo-Integration ist die Verwendung von gleichverteilt zufällig erzeugten, d.h. voneinander unabhängigen Stützpunkten, die entweder für jeden Kandidaten neu gebildet werden, oder für eine ganze Kandidatenmenge KM^t und somit für die gesamte Iteration t verwendet werden. Im Gegensatz hierzu stehen die Verfahren der Average-Case-Integration (siehe Traub et al. (1988[338]), Traub & Wozniakowski (1994[338]), Wozniakowski (1996a[372], b[373]), Sloan & Wozniakowski (1996[314]), Werschulz & Wozniakowski (2000[358])), bei denen die Stützpunkte deterministisch z.B. als Hammersley-Punkte oder hyperbolische Kreuzungspunkte erzeugt werden. Im Kontext des Retrievals bei einer klassifizierten Gesamtmenge von Dokumentvektoren, könnten als Stützpunkte die Gewichtsvektoren aller Neurone aus N_{DV} bzw. einer geeigneten Teilmenge verwendet werden. Auf diese Weise werden Neurone zu Stützpunkten der Approximation und zu Stützpunkten der Integration. Dies bedeutet, dass jeder Voronoi-Region genau ein Eigenschaftswert zugeordnet wird, d.h. eine Voronoi-Region trägt mit diesem Wert an der Eigenschafts-Integral-Approximation bei. Das n-dimensionale Volumen der Voronoi-Regionen ist abhängig von der lokalen Dichte der Dokumentvektoren, d.h. bei einer großen Dichte werden lokal kleinere Voronoi-Regionen durch den Wachstums- und Adaptionsprozess der SC-GNG-SOM gebildet.

Verbunden ist damit die Erwartung einer Effizienzverbesserung, da die Stützpunkte iterationsübergreifend für alle Integrationsprozesse genutzt werden können. Werden keine Adaptionen der Gewichtsvektoren durchgeführt, so müssten bestimmte Eigenschaften nicht immer wieder neu berechnet werden, was im weiteren untersucht werden soll. Wie sich zeigen wird, können jedoch keine Effizienzverbesserungen bezüglich der Datenstrukturen erreicht werden. Eine potentielle Effizienzverbesserung ergibt sich höchstens dann, wenn die Anzahl der als Stützpunkt verwendeten Neurone kleiner ist, als die Anzahl der Stützpunkte, die bei einem anderen Verfahren, wie der Monte-Carlo-Integration, benötigt werden., wovon jedoch ausgegangen werden kann.

Daneben existiert auch die Erwartung einer Effektivitätsverbesserung, da die Gewichtsvektoren die Dokumentvektorenverteilung in besonderer Weise repräsentieren, wenn als Wachstumskriterium der SC-GNG-SOM die Minimierung des Quantifizierungsfehlers verwendet wurde. Die entstehende Verbindungsstruktur der Gewichtsvektoren ist eine Delaunay-Triangulation bzw. ein Teilgraph einer Delaunay-

Triangulation. Stützpunkte dieser Art zeichnen sich für die Approximation besonders aus (siehe Omohundro (1990[241])), auch wenn in der betreffenden Literatur Interpolation anstatt eine Regression wie eine LWR-Approximation sowie die Triangulation auf der Ebene der Instanzen anstatt auf der Ebene von Prototypen untersucht wurde. Die Verwendung von ausgewählten Repräsentanten der zugrunde liegenden Verteilung sollte für die Integration über dieser Verteilung ebenfalls besonders geeignet sein.

Die Darstellungen sollen die Bezeichnungen im Kontext des iterativen Retrievals verwenden, d.h. zu Beginn der Iteration t liegt eine Gesamtstimulusmenge $M_{DV(m)}^{\leq t-1}$ vor, die als Vereinigungsmenge der Stimulusmengen erzeugt wird, die bis einschließlich der Iteration t-1 gebildet wurden, d.h.

$$M_{DV(m)}^{\leq t-1} = \bigcup_{k=0,...,t-1} M_{DV(m)}^{k}. \tag{839}$$

Elemente von $M_{DV(m)}^{\leq t-1}$ werden in der Iteration t mit m_j^t bezeichnet, und sie besitzen die Basisstruktur $m_j^t = (x_j^t, rel(x_j^t))$.

Im allgemeinen Fall der Output-Varianz-Integration in Verbindung mit einem instanzbasierten Approximationsmodell $AM(x \mid M^t)$ müssen die Stützpunkte der Approximation nur die Basisstruktur $m_j^t = (x_j^t, y_j^t)$ besitzen, während bei der Bias-Quadrat-Integration ein Bias-Approximationsmodell $AM(bias(x) \mid M^t)$ notwendig ist, sodass die Stützpunkte der Approximation zusätzlich eine Output-Schätzung und ein Biaswert besitzen müssen, wie z.B. $m_j^t = (x_j^t, y_j^t, f(x_j^t \mid AM(x \mid G(x_j^t \mid M_{DV(m)}^{\leq t-1}, 2, \mu_L)), bias(x_j^t))$. Im Retrievalkontext besitzen die Elemente m_j^t die erweiterte Basisstruktur

$$M_{DV(m)}^{\leq t-1} = \{m_j^t = (x_j^t, rel(x_j^t), rel(x_j^t \mid AM(rel(x) \mid G(x_j^t \mid M_{DV(m)}^t, 2, \mu_L^t)), bias(x_j^t)) \mid j = 1, ..., \mu_L^t\}, \tag{840}$$

wodurch die beiden instanzbasierten Modelle $AM(rel(x) \mid M_{DV(m)}^{\leq t-1})$ und $AM(bias(x) \mid M_{DV(m)}^{\leq t-1})$ spezifiziert sind. Es wird eine iterationabhängige Anzahl μ_L^t von Stimuli verwendet, da diese Stimulusmenge streng monoton im Verlauf der Iterationen t wächst.

Alternativ kann die Relevanzschätzung der Stimuli durch das prototypbasierte Modell $AM(rel(x) \mid N_{DV}^t)$ erfolgen, sodass keine Entscheidung notwendig wird, welche Art von Gewinnerlisten-Verfahren angewendet werden soll. Diese Alternative ist zudem konsistent mit der Standardvorgehensweise, bei der Stimuli und Kandidaten durch das prototypbasierte Modell bewertet werden und Neurone durch ein instanzbasiertes Modell. Da weniger Neurone als Instanzen vorliegen, besitzt die Erzeugung von $rel(x_j^t \mid N_{DV}^t)$ zudem einen Effizienzvorteil gegenüber $rel(x_j^t \mid AM(rel(x) \mid G(x_j^t \mid M_{DV(m)}^t, 2, \mu_L^t))$. Unabhängig welche der beiden Alternativen verwendet wird, es soll die allgemeine Bezeichnung $rel(x_j^t)\hat{}$ verwendet werden:

$$M_{DV(m)}^{\leq t-1} = \{m_j^t = (x_j^t, rel(x_j^t), rel(x_j^t)\hat{}, bias(x_j^t)) \mid j = 1, ..., \mu_L^t\}. \tag{841}$$

Werden Neurone als Stützpunkte einer Approximation verwendet, so übernimmt der Gewichtsvektor die Funktion des Stützpunktes im Inputraum. Bei einem Approximationsmodell $AM(rel(x) \mid N_{DV}^t)$ muss jedes Neuron ein Stützpunkt im Output- bzw. Relevanzraum besitzen, wobei angenommen wird, dass der richtige Wert an der Stelle des Gewichtsvektors unbekannt ist, sodass eine Approximation $rel(w(x_{DV})_j^t)\hat{}$

notwendig ist. Diese Output-Schätzung soll von dem instanzbasierten, globalen Modell AM(rel(x) | $M_{DV(m)}^{\leq t-1}$) durchgeführt werden. Bei einem Approximationsmodell AM(bias(x) | N_{DV}^{t}) muss jedes Neuron einen Stützpunkt bias($w(x_{DV})_j^t$)^ im Biasraum besitzen, der durch das instanzbasierte Modell AM(bias(x) | $M_{DV(m)}^{\leq t-1}$) bestimmt wird. Dementsprechend besitzt das SC-GNG-SOM die folgende Struktur, wenn angenommen wird, dass jedem Neuron eine Relevanz- und eine Biasschätzung zugeordnet werden, unabhängig von der Distanz der Approximationsstützpunkte zu den Gewichtsvektoren:

$$N_{DV}^{t} = \{n_{DV,j}^{t} = (w(x_{DV})_j^t, rel(w(x_{DV})_j^t)\wedge, bias(w(x_{DV})_j^t)\wedge, M_{DV,j}^{t}, M_{DV(m),j}^{t}, C_{DV,j}) \mid j = 1, ..., \mu_{N,DV}\}, \text{ mit}$$

$$rel(w(x_{DV})_j^t)\wedge = rel(w(x_{DV})_j^t \mid AM(rel(x) \mid M_{DV(m)}^{\leq t-1})),$$

$$bias(w(x_{DV})_j^t)\wedge = bias(w(x_{DV})_j^t \mid AM(bias(x) \mid M_{DV(m)}^{\leq t-1})). \quad (842)$$

Mit dieser Struktur sind die Neurone geeignet, als Stützpunkte der Approximation verwendet zu werden. Sie sind zudem geeignet als Stützpunkte der Bias-Quadrat-Integration, d.h. $N_{DV}^{t} = N_{DV,B}^{t}$, jedoch nicht für eine Output-Varianz-Integration, da hierfür noch eine Output-Varianz an der Stelle der Gewichtsvektoren berechnet werden muss, wenn eine Varianz-Schätzung durch ein LWR-Verfahren durchgeführt werden soll. Wird als Output die Relevanz verwendet, so mus dementsprechend eine Relevanz-Varianz bestimmt werden, die mit $\sigma(w(x_{DV})_j^t)^2$ bezeichnet werden soll und die nach dem Verfahren von Cohn et al. (1995[74]) in Verbindung mit einer Kernel-Regression berechnet wird durch:

$$\sigma(w(x_{DV})_j^t)_{rel}^2 = 1/\nu_j \sum_k h(d_X(w(x_{DV})_j^t, x_k^t)) * (rel(x_k^t) - rel(w(x_{DV})_j^t)\wedge)^2,$$

$$\nu_j = \sum_k h(d_X(w(x_{DV})_j^t, x_k^t)), \forall m_k^t \in M_{DV(m)}^{\leq t-1}. \quad (843)$$

Die vollständige Datenstruktur für Neurone als Stützpunkte der Approximation und der Integration wird durch $N_{DV,BV}^{t}$ beschrieben:

$$N_{DV,BV}^{t} = \{n_{DV,j}^{t} = (w(x_{DV})_j^t, rel(w(x_{DV})_j^t)\wedge, bias(w(x_{DV})_j^t)\wedge, \sigma(w(x_{DV})_j^t)_{rel}^2, M_{DV,j}^{t}, M_{DV(m),j}^{t}, C_{DV,j}) \mid j = 1, ..., \mu_{N,DV}\}. \quad (844)$$

Im allgemeinen Fall wird von einer Kandidatenmenge KM^t ausgegangen, die eine Basisstruktur $KM^t = \{m_{K,j}^t = (x_{K,j}^t, \_) \mid j = 1, ..., \mu_K\}$ besitzt. Im Falle des Retrievals übernimmt die primäre Dokumentvektorenmenge DVM_{prim}^{t} die Rolle der Kandidatenmenge, wobei die Anzahl der darin enthaltenen Dokumentvektoren als μ_K^t bezeichnet werden soll. Die Datenstruktur der Kandidaten-Dokumentvektoren, die hier mit $m_{K,j}^t$ bezeichnet werden sollen, muss im Fall der Relevanz-Varianz-Integration um eine Relevanzschätzung $rel(x_{K,j}^t)\wedge$ und einen Relevanz-Outputwert $\sigma(x_{K,j}^t)_{rel}^2$ erweitert werden. Bei einem instanzbasierten Approximationsmodell wird die Relevanzschätzung der Art $rel(x_{K,j}^t \mid AM(rel(x) \mid M_{DV(m)}^{\leq t-1}))$ durchgeführt, während beim Vorliegen der angepassten GNG-SOM $N_{DV,B}^{t}$ bzw. $N_{DV,BV}^{t}$ die Relevanzschätzung auf der Basis der Gewichtsvektoren erfolgt. Das gleiche gilt für die Relevanz-Varianz:

$$rel(x_{K,j}^t \mid AM(rel(x) \mid N_{DV,BV}^{t})) = 1/\nu_{K,j} \sum_k h(d_X(x_{K,j}^t, w(x_{DV})_k^t)) * rel(w(x_{DV})_k^t)\wedge,$$

$$\sigma(x_{K,j}^t)_{rel}^2 = 1/\nu_{K,j} \sum_k h(d_X(x_{K,j}^t, w(x_{DV})_k^t)) * (rel(w(x_{DV})_k^t)\wedge - rel(x_{K,j}^t \mid AM(rel(x) \mid N_{DV,BV}^{t})))^2,$$

$$\nu_{K,j} = \sum_k h(d_X(x_{K,j}^t, w(x_{DV})_k^t)), \forall n_{DV,k}^t \in N_{DV,BV}^{t}. \quad (845)$$

Die primäre Dokumentvektorenmenge wird angepasst zu:

$$DVM_{prim,V}^{t} = \{m_{K,j}^{t} = (x_{K,j}^{t}, \_, rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})), \sigma(x_{K,j}^{t})_{rel}^{2}) \mid j = 1, ..., \mu_{K}^{t}\}. \quad (846)$$

Bei der Bias-Quadrat-Integration ist eine Relevanzschätzung $rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t}))$ und eine Bias-Schätzung $bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t}))$ notwendig, mit

$$bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t})) = 1/\nu_{K,j} \sum_{k} h(d_X(x_{K,j}^{t}, w(x_{DV})_k^{t})) * bias(w(x_{DV})_k^{t})\wedge,$$
$$\nu_{K,j} = \sum_{k} h(d_X(x_{K,j}^{t}, w(x_{DV})_k^{t})), \forall n_{DV,k}^{t} \in N_{DV,BV}^{t}. \quad (847)$$

Es folgt die Berechnung der korrigierten Relevanzschätzung

$$rel(x_{K,j}^{t})_{bias} = rel(x_{K,j}^{t})\wedge - bias(x_{K,j}^{t})\wedge, \quad (848)$$

sodass sich die primäre Dokumentvektorenmenge ergibt:

$$DVM_{prim,B}^{t} = \{m_{K,j}^{t} = (x_{K,j}^{t}, \_, rel(x_{K,j}^{t})_{bias}, rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})),$$
$$bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t}))) \mid j = 1, ..., \mu_{K}^{t}\}. \quad (849)$$

Als Vereinigung beider Datenstrukturen ergibt sich

$$DVM_{prim,BV}^{t} = \{m_{K,j}^{t} = (x_{K,j}^{t}, \_, rel(x_{K,j}^{t})_{bias}, rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})),$$
$$bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t})), \sigma(x_{K,j}^{t})_{rel}^{2}) \mid j = 1, ..., \mu_{K}^{t}\}. \quad (850)$$

Grundsätzlich stellt sich wiederum die Frage, ob bei einer kombinierten Relevanz-Varianz- und Bias-Quadrat-Integration die korrigierte Relevanzschätzung $rel(x_{K,j}^{t})_{bias}$ ausschließlich durch die Bias erzeugt werden soll, oder ob die Standardabweichung $\sigma(x_{K,j}^{t})_{rel}$ berücksichtigt werden soll, was zu einem Intervall führt:

$$rel(x_{K,j}^{t})_{bias} = rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})) - bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t})), \text{ oder}$$
$$rel(x_{K,j}^{t})_{bias,\sigma} = [rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})) - bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t})) - \sigma(x_{K,j}^{t})_{rel},$$
$$rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})) - bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t})) + \sigma(x_{K,j}^{t})_{rel}]. \quad (851)$$

Wird die zweite Option verwendet, so muss die Reihenfolge der berechneten Terme der Sequenz $(rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})), bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t})), \sigma(x_{K,j}^{t})_{rel}^{2}, rel(x_{K,j}^{t})_{bias})$ oder $(rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})), \sigma(x_{K,j}^{t})_{rel}^{2}, bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t})), rel(x_{K,j}^{t})_{bias})$ entsprechen.

Die Prüfung jedes Kandidaten $m_{K,j}^{t}$ aus KM_{BV}^{t} erfordert die Individualisierung der Stützpunktmenge von SM_{BV}^{t} zu $SM_{BV|K,j}^{t}$, was im Kontext des Retrievals bedeutet, dass die GNG-SOM $N_{DV,BV}^{t}$ bezüglich des Kandidaten $m_{K,j}^{t}$ individualisiert wird zu $N_{DV,BV|K,j}^{t}$. Hierzu wird zunächst die Individualisierung zu $SM_{B|K,j}^{t}$ bzw. $N_{DV,B|K,j}^{t}$ und zu $SM_{V|K,j}^{t}$ bzw. $N_{DV,V|K,j}^{t}$ betrachtet, gefolgt von der Vereinigung der beiden Datenstrukturen zu $SM_{BV|K,j}^{t}$ bzw. $N_{DV,BV|K,j}^{t}$.

Bei der Bias-Quadrat-Integration muss eine Relevanz-Schätzung der Stützpunkte auf der Basis der potentiell neuen Stimulusmenge gebildet werden. In der Iteration t wird aus der primären Menge $DVM_{prim,B}^{t}$ eine tertiäre Liste $DV_{ter,BV}^{t}$ gebildet, die nach der Bewertung durch den Agenten die Stimulusmenge $M_{DV(m)}^{t}$ bildet. Vereinigt mit $M_{DV(m)}^{\leq t-1}$ wird so die neue Gesamtstimulusmenge $M_{DV(m)}^{\leq t}$ erzeugt. D.h. genau genommen müsste das Modell $AM(x \mid M_{DV(m)}^{\leq t-1} \cup M_{DV(m)}^{t})$ bzw. $AM(x \mid M_{DV(m)}^{\leq t})$ betrachtet werden, wobei $M_{DV(m)}^{t}$ unbekannt ist. Wird eine Sollanzahl μ_L von Elementen aus $DVM_{prim,B}^{t}$ extern festgelegt, so ergeben sich "μ_L aus μ_K^{t}" mögliche Teilmengen, die als eigentliche Kandidatenmengen getestet werden müssten. Wie im Kontext der Einführung der direkten Verfahren bzw. dem Optimal Experiment Design dargestellt wurde, ist der Test von Kandidaten-Teilmengen aus kombinatorischen Gründen kaum tragbar, sodass das Verfahren so aufgebaut werden soll, als ob immer genau ein Kandidat pro Iteration ausgewählt wird, auch wenn letztlich aus der sich ergebenden Kandidatenliste mehrere Elemente ausgewählt werden. Auf diese Problematik wird im Abschnitt 5.4) noch einmal eingegangen.

Die Gewichtsvektoren als Stützpunkte erhalten ihre Relevanz- und Bias-Schätzungen sowie ihre Relevanz-Varianzwerte durch die Instanzen aus der jeweiligen Stimulusmenge. Dies bedeutet, dass das Modell $AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})$ betrachtet wird, um jedem Stützpunkt der Integration eine zweite Relevanzschätzung zuzuordnen. Als Substitut des unbekannten Relevanzwertes des Kandidaten $m_{K,j}^{t}$ wird der korrigierte Wert $rel(x_{K,j}^{t})_{bias}$ bzw. $rel(x_{K,j}^{t})_{bias,\sigma}$ verwendet:

$$rel(w(x_{DV})_i^{t})_{K,j}\hat{} = 1/\nu_i * [h(d_X(w(x_{DV})_i^{t}, x_{K,j}^{t})) * rel(x_{K,j}^{t})_{bias,\sigma} + \sum_k h(d_X(w(x_{DV})_i^{t}, x_k^{t})) * rel(x_k^{t})],$$

$$\nu_i = h(d_X(w(x_{DV})_i^{t}, x_{K,j}^{t})) + \sum_k h(d_X(w(x_{DV})_i^{t}, x_k^{t})), \forall m_k^{t} \in M_{DV(m)}^{\leq t-1}. \quad (852)$$

Die Differenz aus dieser neuen Relevanz-Schätzung $rel(w(x_{DV})_i^{t})_{K,j}\hat{}$ und der alten Schätzung $rel(w(x_{DV})_i^{t})$ wird als $\Delta rel(w(x_{DV})_i^{t})$ bezeichnet, die zusammen mit der vorliegenden Bias-Schätzung $bias(w(x_{DV})_i^{t} \mid AM(bias(x) \mid M_{DV(m)}^{\leq t-1}))$ den Bias-Quadratwert $bias(w(x_{DV})_i^{t})_{K,j}^{2}$ bilden:

$$\begin{aligned} bias(w(x_{DV})_i^{t})_{K,j}^{2} \quad = \quad & \Delta rel(w(x_{DV})_i^{t})^2 \\ & + 2 * \Delta rel(w(x_{DV})_i^{t}) * bias(w(x_{DV})_i^{t} \mid AM(bias(x) \mid M_{DV(m)}^{\leq t-1})) \\ & + bias(w(x_{DV})_i^{t} \mid AM(bias(x) \mid M_{DV(m)}^{\leq t-1}))^2. \end{aligned} \quad (853)$$

Es ergibt sich die individualisierte Stützpunktmenge $N_{DV,B|K,j}^{t}$, wobei die Aggregation der Bias-Quadratwerte über alle Stützpunkte als Bias-Quadrat-Integral-Approximation $IB[AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})]\hat{}$ der Stützpunktmenge zugeordnet wird:

$$\begin{aligned} N_{DV,B|K,j}^{t} = \{ & n_{DV,i}^{t} = (w(x_{DV})_i^{t}, rel(w(x_{DV})_i^{t})\hat{}, rel(w(x_{DV})_i^{t})_{K,j}\hat{}, \\ & bias(w(x_{DV})_i^{t} \mid AM(bias(x) \mid M_{DV(m)}^{\leq t-1})), bias(w(x_{DV})_i^{t})_{K,j}^{2}, M_{DV,i}^{t}, M_{DV(m),i}^{t}, C_{DV,i}), \\ & IB[AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})]\hat{} \mid j = 1, ..., \mu_{N,DV}\}. \end{aligned} \quad (854)$$

Den Kandidaten kann die Referenz für die individualisierte Stützpunktmenge sowie der Bias-Quadrat-Integralwert zugeordnet werden:

$$DVM_{prim,B}^{t} = \{m_{K,j}^{t} = (x_{K,j}^{t}, \_, rel(x_{K,j}^{t})_{bias}, rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})),$$
$$bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t})), N_{DV,B|K,j}^{t}, IB[AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})]^{\wedge})$$
$$\mid j = 1, ..., \mu_{K}^{t}\}. \quad (855)$$

Bei der Relevanz-Varianz-Integration muss eine Relevanz-Varianz an der Stelle eines Stützpunktes $n_{DV,i}^{t}$ berechnet werden, wenn $m_{K,j}^{t}$ als neues Element in die Stimulusmenge aufgenommen würde. Dies geschieht mit Hilfe der Terme $rel(w(x_{DV})_{i}^{t})^{\wedge} = rel(w(x_{DV})_{i}^{t} \mid AM(rel(x) \mid M_{DV(m)}^{\leq t-1}))$, $rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t}))$, $\sigma(w(x_{DV})_{i}^{t})_{rel}^{2}$ und $\sigma(x_{K,j}^{t})_{rel}^{2}$. Der erste und der dritte Term sind in der Datenstruktur $n_{DV,i} \in N_{DV,V}^{t}$ enthalten. Der zweite und der vierte Term sind in der Datenstruktur $m_{K,j}^{t} \in DVM_{prim,V}^{t}$ enthalten, sodass keine neuen Berechnungen der Terme durchgeführt werden müssen, und der Relevanz-Varianzwert berechnet werden kann durch:

$$\sigma(w(x_{DV})_{i}^{t})_{rel|K,j}^{2} =$$
$$[(\nu_{i} * \sigma(w(x_{DV})_{i}^{t})_{rel}^{2} + h(d_{X}(w(x_{DV})_{i}^{t}, x_{K,j}^{t})) * \sigma(x_{K,j}^{t})_{rel}^{2})/(\nu_{i} + h(d_{X}(w(x_{DV})_{i}^{t}, x_{K,j}^{t}))]$$
$$+ [(\nu_{i} * h(d_{X}(w(x_{DV})_{i}^{t}, x_{K,j}^{t})) * (rel(x_{K,j}^{t})^{\wedge} - rel(w(x_{DV})_{i}^{t})^{\wedge})^{2})/(\nu_{i} + h(d_{X}(w(x_{DV})_{i}^{t}, x_{K,j}^{t})))^{2}],$$
$$\text{mit } \nu_{i} = \sum_{k} h(d_{X}(w(x_{DV})_{i}^{t}, x_{k}^{t})), \forall m_{k}^{t} \in M_{DV(m)}^{\leq t-1}. \quad (856)$$

Mit Hilfe von ν_{i} und $h(d_{X}(w(x_{DV})_{i}^{t}, x_{K,j}^{t}))$ erfolgt die Korrektur zu $\sigma(w(x_{DV})_{i}^{t})_{rel|K,j}^{2}$, d.h. die beiden Komponenten $\sigma(w(x_{DV})_{i}^{t})_{rel|K,j}^{2}$ und $\sigma(w(x_{DV})_{i}^{t})_{rel|K,j}^{2}$ werden in die Datenstruktur der Stützpunkte eingetragen, wodurch die individualisierte Stützpunktmenge $N_{DV,V|K,j}^{t}$ erzeugt wird. Die Werte $\sigma(w(x_{DV})_{i}^{t})_{rel|K,j}^{2}$ aller Stützpunkte werden zu der Relevanz-Varianz-Integral-Approximation $IV[AM(rel(x) \mid M^{t} \cup \{m_{K,j}^{t}\})]^{\wedge}$ aggregiert, die der Stützpunktmenge $N_{DV,V|K,j}^{t}$ zugeordnet wird:

$$N_{DV,V|K,j}^{t} = \{n_{DV,i}^{t} = (w(x_{DV})_{i}^{t}, rel(w(x_{DV})_{i}^{t})^{\wedge}, \sigma(w(x_{DV})_{i}^{t})_{rel}^{2}, \sigma(w(x_{DV})_{i}^{t})_{rel|K,j}^{2}, \sigma(w(x_{DV})_{i}^{t})_{rel|K,j}^{2},$$
$$M_{DV,i}^{t}, M_{DV(m),i}^{t}, C_{DV,i}), IV[AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})]^{\wedge}$$
$$\mid j = 1, ..., \mu_{N,DV}\}. \quad (857)$$

Den Kandidaten kann die Referenz für die individualisierte Stützpunktmenge sowie der Referenz-Varianz-Integralwert zugeordnet werden:

$$DVM_{prim,V}^{t} = \{m_{K,j}^{t} = (x_{K,j}^{t}, \_, rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})), \sigma(x_{K,j}^{t})_{rel}^{2}, N_{DV,V|K,j}^{t},$$
$$IV[AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})]^{\wedge}) \mid j = 1, ..., \mu_{K}^{t}\}. \quad (858)$$

Werden die beiden Datenstrukturen $N_{DV,B|K,j}^{t}$ und $N_{DV,V|K,j}^{t}$ zu $N_{DV,BV|K,j}^{t}$ kombiniert, so ergibt sich:

$$N_{DV,BV|K,j}^{t} = \{n_{DV,i}^{t} = (w(x_{DV})_{i}^{t}, rel(w(x_{DV})_{i}^{t})^{\wedge}, rel(w(x_{DV})_{i}^{t})_{K,j}^{\wedge}, \sigma(w(x_{DV})_{i}^{t})_{rel}^{2},$$
$$bias(w(x_{DV})_{i}^{t} \mid AM(bias(x) \mid M_{DV(m)}^{\leq t-1})), bias(w(x_{DV})_{i}^{t})_{K,j}^{2}, \sigma(w(x_{DV})_{i}^{t})_{rel|K,j}^{2},$$
$$\sigma(w(x_{DV})_{i}^{t})_{rel|K,j}^{2}, M_{DV,i}^{t}, M_{DV(m),i}^{t}, C_{DV,i}),$$
$$IB[AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})]^{\wedge}, IV[AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})]^{\wedge}$$
$$\mid j = 1, ..., \mu_{N,DV}\}. \quad (859)$$

Eine Kombination von $DVM_{prim,B}^{t}$ und $DVM_{prim,V}^{t}$ zu $DVM_{prim,BV}^{t}$ ergibt:

$$DVM_{prim,BV}^{t} = \{m_{K,j}^{t} = (x_{K,j}^{t}, \_, rel(x_{K,j}^{t})_{bias}, rel(x_{K,j}^{t} \mid AM(rel(x) \mid N_{DV,BV}^{t})), bias(x_{K,j}^{t} \mid AM(bias(x) \mid N_{DV,BV}^{t})), \sigma(x_{K,j}^{t})_{rel}^{2}, N_{DV,B|K,j}^{t}, N_{DV,V|K,j}^{t}, IB[AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})]\wedge, IV[AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})]\wedge)) \mid j = 1, ..., \mu_{K}^{t}\}. \quad (860)$$

Im nächsten Schritt wird die Kandidatenmenge in eine Kandidatenliste transformiert, d.h. $DVM_{prim,BV}^{t}$ wird in $DV_{sec,BV}^{t}$ mit Hilfe der Bias-Quadrat- und Relevanz-Varianz Integrale transformiert. Dies kann durch Aggregation der beiden Integralwerte erfolgen, was im einfachsten Fall durch die Addition der beiden Werte geschieht, oder es wird ein Mehr-Ziel-Entscheidungsverfahren angewendet, bei dem eine geordnete Kandidatenliste oder eine geordnete Liste von Kandidatenmengen gebildet wird. In Abhängigkeit von der Struktur der Kandidatenliste und der Anzahl der Kandidaten bzw. Rangplätze der Liste wird ein geeignetes Selektionverfahren verwendet, um aus der Liste $DV_{sec,BV}^{t}$ eine tertiäre Teilliste $DV_{ter,BV}^{t}$ zu erzeugen. Die korrespondierenden Dokumente der Dokumentvektoren aus $DV_{ter,BV}^{t}$ wird dem Agenten in der vorgegebenen Strukturierung präsentiert, der Relevanzbewertungen abgibt. Auf diese Weise wird die Stimulusmenge $M_{DV(m)}^{t}$ erzeugt, die vereinigt mit $M_{DV(m)}^{\leq t-1}$ die neue Gesamtstimulusmenge $M_{DV(m)}^{\leq t}$ bildet. Damit sind die Operationen in der Iteration t beendet, und es wird t+1 begonnen, wenn ein Abbruchkriterium nicht greifen sollte bzw. wenn der Agent die Interaktion nicht beendet.

Zu Beginn von t+1 erfolgt eine Erweiterung der Datenstruktur der Stimuli sowie eine umfangreiche Reinitialisierung der Datenstruktur der Neurone auf der Basis der neuen Gesamtstimulusmenge $M_{DV(m)}^{\leq t}$. Zunächst wird die Datenstruktur der Neurone korrigiert, indem alle Komponenten bis auf $w(x_{DV})_{j}^{t}$, $M_{DV,j}^{t}$, $M_{DV(m),j}^{t}$ und $C_{DV,j}$ gelöscht werden. Der Gewichtsvektor $w(x_{DV})_{j}^{t}$ wird direkt zu $w(x_{DV})_{j}^{t+1}$ umbenannt, da ein Verfahren unterstellt wird, bei dem während des Retrievals keine Gewichtsvektorenadaption erfolgen soll. Die lokale Dokumentvektorenmenge $M_{DV,j}^{t}$ wird ebenfalls direkt zu $M_{DV,j}^{t+1}$ umbenannt, da vorausgesetzt wird, dass in t bzw. allgemein, während einer Retrievaloperation keine Indexierungsoperationen erfolgen, und somit keine neuen Dokumentvektoren auftreten und in eine lokale Menge einzuordnen sind. Es folgt die Erweiterung der lokalen Stimulusmengen $M_{DV(m),j}^{t}$ auf der Basis von $M_{DV(m)}^{\leq t}$, sodass $M_{DV(m),j}^{t+1}$ erzeugt wird. Die Verbindungsstruktur in Form des Verbindungsvektors $C_{DV,j}$ bleibt ebenfalls erhalten, da keine neuen Gewichtsvektoren oder Verbindungen eingeführt und keine Gewichtsvektoradaptionen durchgeführt wurden. Der nächste Schritt besteht in der globalen Relevanzschätzung durch alle Stimuli aus $M_{DV(m)}^{t+1}$, d.h es wird $rel(w(x_{DV})_{j}^{t+1})\wedge = rel(w(x_{DV})_{j}^{t} \mid AM(rel(x) \mid M_{DV(m)}^{t+1}))$ gebildet, sodass sich die vorläufige Datenstruktur ergibt:

$$N_{DV}^{t+1} = \{n_{DV,j}^{t+1} = (w(x_{DV})_{j}^{t+1}, rel(w(x_{DV})_{j}^{t+1})\wedge, M_{DV,j}^{t+1}, M_{DV(m),j}^{t+1}, C_{DV,j}) \mid j = 1, ..., \mu_{N,DV}\}. \quad (861)$$

Um den Neuronen Biaswerte zuordnen zu können, müsste der richtige Relevanzwert der Gewichtsvektoren bekannt sein, was jedoch ausgeschlossen wurde, oder es müssen Biasschätzungen auf der Basis der Stimuli erzeugt werden. Hierzu muss jedoch zunächst die Datenstruktur der Stimuli erweitert werden, indem Relevanzschätzungen auf der Basis von $M_{DV(m)}^{\leq t}$ und einem Gewinnerlisten-Verfahren oder auf

der Basis von N_{DV}^{t+1} durchgeführt werden. Unabhängig von dem Verfahren werden Schätzungen $rel(x_j^{t+1})\hat{}$ erzeugt, die zusammen mit $rel(x_j^{t+1})$ Biaswerte $bias(x_j^{t+1})$ bilden, sodass sich die Datenstruktur ergibt:

$$M_{DV(m)}^{\leq t} = \{m_j^{t+1} = (x_j^{t+1}, rel(x_j^{t+1}), rel(x_j^{t+1})\hat{}, bias(x_j^{t+1})) \mid j = 1, ..., \mu_L^{t+1}\}. \quad (862)$$

Mit den richtigen Biaswerten der Stimuli lassen sich Biasschätzungen an den Stellen der Gewichtsvektoren in der gleichen Weise durch ein LWR-Verfahren erzeugen, wie die Relevanzschätzungen an der Stelle der Gewichtsvektoren. Für die Neurone ergibt sich somit die Datenstruktur:

$$N_{DV}^{t+1} = \{n_{DV,j}^{t+1} = (w(x_{DV})_j^{t+1}, rel(w(x_{DV})_j^{t+1})\hat{}, bias(w(x_{DV})_j^{t+1})\hat{}, M_{DV,j}^{t+1}, M_{DV(m),j}^{t+1}, C_{DV,j}) \mid j = 1, ..., \mu_{N,DV}\}. \quad (863)$$

Werden Neurone als Stützpunkte der Integration verwendet, so muss für eine Output-Varianz-Integration den Stützpunkten eine Relevanz-Varianz zugeordnet werden, was durch eine Kernel-Regression mit Hilfe der Relevanzwerte der Stimuli und der Relevanzschätzung $rel(w(x_{DV})_j^{t+1})\hat{}$ erfolgt:

$$\sigma(w(x_{DV})_j^{t+1})_{rel}^2 = 1/\nu_j \sum_k h(d_X(w(x_{DV})_j^{t+1}, x_k^{t+1})) * (rel(x_k^{t+1}) - rel(w(x_{DV})_j^{t+1})\hat{})^2,$$
$$\nu_j = \sum_k h(d_X(w(x_{DV})_j^{t+1}, x_k^{t+1})), \forall m_k^{t+1} \in M_{DV(m)}^{\leq t}. \quad (864)$$

Die vollständige Datenstruktur für Neurone als Stützpunkte der Approximation und der Integration wird durch $N_{DV,BV}^{t+1}$ beschrieben:

$$N_{DV,BV}^{t+1} = \{n_{DV,j}^{t+1} = (w(x_{DV})_j^{t+1}, rel(w(x_{DV})_j^{t+1})\hat{}, bias(w(x_{DV})_j^{t+1})\hat{}, \sigma(w(x_{DV})_j^{t+1})_{rel}^2, M_{DV,j}^{t+1}, M_{DV(m),j}^{t+1}, C_{DV,j}) \mid j = 1, ..., \mu_{N,DV}\}. \quad (865)$$

Es folgt die Bestimmung der Ergebnismenge DVM_{prim}^t der Stimuluskandidaten, sowie die sukzessive Erweiterung der Datenstrukturen zu $DVM_{prim,BV}^t$ und entsprechende Erweiterungen der Datenstruktur der Neurone in $N_{DV,BV}^{t+1}$.

Als Ergebnis der Darstellung zur Verwendung von Neuronen als Stützpunkte der Approximation und der Integration zeigt sich, dass keine unmittelbare Effizienzverbesserung erreicht werden kann, da individualisierte Neuronenmengen $N_{DV,BV|K,j}^t$ analog zu den individualisierten Stützpunktmengen $SM_{BV|K,j}^t$ erzeugt werden müssen. Da die Neurone aus $N_{DV,BV|K,j}^t$ mehr Komponenten besitzen als z.B. zufällig erzeugte Stützpunkte bei einer Monte-Carlo-Integration, ist die Verwendung von Neuronen als Stützpunkte der Integration aus dieser Sicht weniger speichereffizient. Eine potentielle Effizienzverbesserung bezüglich der Rechen- und der Speicherressourcen ergibt sich jedoch dann, wenn die Anzahl der Neuronen-Stützpunkte kleiner ist, als die Anzahl der Stützpunkte, die bei einem anderen Verfahren benötigt werden. Dies ist insbesondere dann der Fall, wenn regionale Neuronengruppen verwendet werden. D.h. analog zu den regionalen Approximationen von Eigenschaften von Dokumentvektoren aus DVM_{prim}^t durch eine regionale Neuronengruppe, kann auch für die Integration eine Teilmenge der Neurone ausgewählt werden, welche die Region mehr oder weniger großräumig umschreiben, in der die bisherigen Stimuli aus $M_{DV(m)}^{\leq t-1}$ und die jetzigen Stimuluskandidaten aus DVM_{prim}^t liegen.

Der Nachteil der Verwendung einer regionalen Neuronengruppe als Stützpunkte einer Integration besteht darin, dass die Modellqualität von AM(rel(x) | $M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^t\}$) nicht über den gesamten Inputraum sondern nur über eine Teilregion geschätzt wird. Je nach der verwendeten Methode zur Bestimmung der nächsten Dokumentvektoren-Ergebnismenge DVM_{prim}^{t+1} muss dies jedoch kein gravierender Nachteil sein. Wird ein Modell in der Region getestet, in der die Dokumentvektoren liegen, die in der nächsten Iteration in der primären Ergebnismenge nachgewiesen werden, so ist dies eine hinreichende Teststrategie.

5.2.4) Effektivitätsverbesserungen bei direkten Verfahren durch zentrale Momente

Die numerische Berechnung von Bias-Quadrat- und Output-Varianz-Integralen sind Beispiele für Fehler-Mittelwerte, da bei der Approximation der Integrale der arithmetische Mittelwert der jeweiligen Fehlerart bestimmt wird. Mittelwerte haben als Qualitätsmaß jedoch entscheidende Effektivitätsnachteile, da es sich dabei nur um ein Maß handelt, mit dem eine Verteilung beschrieben wird. Andere Verteilungsmaße wurden bei den vorgestellten Vorgehensweisen der direkten Verfahren zum aktiven Lernen nicht berücksichtigt. Diese können jedoch ebenso direkt aus der Datenstruktur einer individualisierten Stützpunktmenge $SM_{BV|K,j}^t$ abgeleitet werden, wie die Mittelwertsbildung. Werden mehrere Verteilungsmaße verwendet, so entspricht dies einer Mehr-Ziel-Entscheidungssituation, die bereits vorliegt, wenn ein Bias-Quadrat- und ein Output-Varianz-Integral pro Kandidat $m_{K,j}^t$ bestimmt wird, sodass die Verfahren, die zur Bildung einer geordneten Kandidatenliste verwendet werden, von einem Zwei-Ziel-Verfahren auf ein Mehr-Ziel-Verfahren mit mehr als zwei Zielen umgestellt werden müssen. Wird das Pareto-Kriterium verwendet, um eine Hierarchie von disjunkten Kandidatenmengen zu erzeugen, so ist keine konzeptuelle Umstellung erforderlich, da das Pareto-Kriterium bei einer beliebigen Anzahl von Zielfunktionen in der gleichen Weise arbeitet. Eine ungewichtete Aggregation von Einzel-Zielfunktionen arbeitet ebenfalls in der gleichen Art bei beliebig vielen Zielfunktionen, während andere Formen der Aggregation, wie Formen der gewichteten Aggregation, möglicherweise konzeptuell erweitert werden müssten.

Anstatt nur ein Bias-Quadrat- und Output-Varianz-Integral zu berechnen, können die ersten, zentralen Momente der Bias-Quadrat- und der Output-Varianz-Verteilung berechnet werden (siehe auch Abschnitt 2.3.7)). Zunächst soll die Bias-Quadrat-Verteilung betrachtet werden, wobei μ_S Stützpunkte $m_{S,i}^t$ Bias-Quadratwerte $bias(x_{S,i}^t)_{K,j}^2$ besitzen, die angeben, wie sich der erwartete Bias-Quadratwert an der Stelle $x_{S,i}^t$ entwickelt, wenn $m_{K,j}^t$ in die Stimulusmenge aufgenommen wird, d.h. wenn eine Stimulusmenge $M^t \cup \{m_{K,j}^t\}$ vorliegen würde. Der arithmetische Mittelwert $\overline{bias}(x_{S,i}^t)_{K,j}^2$ der μ_S Stützpunkte ist die Approximation IB[AM(x | $M^t \cup \{m_{K,j}^t\}$)]^ des Bias-Quadrat-Integrals und gleichzeitig die Initialisierung der Momentbildung mit r = 1, d.h. mom(1,0) der Stützpunktmenge $SM_{BV|K,j}^t$:

$$\overline{bias}(x_{S,i}^t)_{K,j}^2 = IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge = mom(1,0 \mid bias(.)^2, SM_{BV|K,j}^t)$$
$$= 1/\mu_S * \sum_j bias(x_{S,i}^t)_{K,j}^2. \quad (866)$$

Das zweite zentrale Moment der Stützpunktmenge ist gleich der Varianz der Bias-Quadratwerte:

$$mom(2, \overline{bias}(x_{S,i}^t)_{K,j}^2 \mid SM_{BV|K,j}^t) = 1/\mu_S * \sum_j (bias(x_{S,i}^t)_{K,j}^2 - \overline{bias}(x_{S,i}^t)_{K,j}^2)^2. \quad (867)$$

Allgemein wird das r'te zentrale Moment, mit r > 0, bezüglich der Stützpunktmenge berechnet durch:

$$mom(r, \overline{bias}(x_{S,i}^t)_{K,j}^2 \mid SM_{BV|K,j}^t) = 1/\mu_S * \sum_j (bias(x_{S,i}^t)_{K,j}^2 - \overline{bias}(x_{S,i}^t)_{K,j}^2)^r. \quad (868)$$

Die drei ersten Momente entsprechen dem Mittelwert, der Varianz und der Schiefe der Verteilung, wobei diese drei Maße eine Verteilung bereits gut charakterisieren. Beispielsweise werden im Rahmen eines speziellen Bootstrap-Verfahrens, dem Wild-Bootstrap (Mammen (1992[206]), Efron & Tibshirani (1993[105]), Shao & Tu (1995[309])), die drei ersten Momente einer gegebenen, diskreten Verteilung ermittelt, um eine stetige Verteilung mit den gleichen drei Momenten festzulegen, mit der Resampling-operationen durchgeführt werden. Für die Output-Varianz ergibt sich:

$$mom(r, \overline{\sigma}(x_{S,i}^t)_{K,j}^{2'} \mid SM_{BV|K,j}^t) = 1/\mu_S * \sum_j (\sigma(x_{S,i}^t)_{K,j}^{2'} - \overline{\sigma}(x_{S,i}^t)_{K,j}^{2'})^r, \text{ mit}$$
$$\overline{\sigma}(x_{S,i}^t)_{K,j}^{2'} = IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]\hat{} = mom(1,0 \mid \sigma(.)^2, SM_{BV|K,j}^t) = 1/\mu_S * \sum_j \sigma(x_{S,i}^t)_{K,j}^{2'}. \quad (869)$$

Werden die drei ersten zentralen Momente für die Bias-Quadrat- und die Output-Varianz-Verteilung als Substitut für die beiden Integrale gebildet, so ergibt sich die Datenstruktur der Kandidatenmenge bzw. Stützpunktmenge:

$$KM_{BV}^t = \{m_{K,j}^t = (x_{K,j}^t, \_, y_{K,bias,j}^t, f(x_{K,j}^t \mid AM(x \mid M^t)), bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)), \sigma(x_{K,j}^t)^2,$$
$$SM_{BV|K,j}^t, mom(r, \overline{bias}(x_{S,i}^t)_{K,j}^2 \mid SM_{BV|K,j}^t), mom(r, \overline{\sigma}(x_{S,i}^t)_{K,j}^{2'} \mid SM_{BV|K,j}^t) \mid r = 1, 2, 3)$$
$$\mid j = 1, ..., \mu_K\}. \quad (870)$$

$$SM_{BV|K,j}^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\})), \Delta y_{S,i}^t\hat{},$$
$$bias(x_{S,i}^t \mid AM(bias(x) \mid M^t)), \sigma(x_{S,i}^t)^2, bias(x_{S,i}^t)_{K,j}^2, \sigma(x_{S,i}^t)_{K,j}^2, \sigma(x_{S,i}^t)_{K,j}^{2'}),$$
$$mom(r, \overline{bias}(x_{S,i}^t)_{K,j}^2 \mid SM_{BV|K,j}^t), mom(r, \overline{\sigma}(x_{S,i}^t)_{K,j}^{2'} \mid SM_{BV|K,j}^t)$$
$$\mid r = 1, 2, 3; i = 1, ..., \mu_S\}. \quad (871)$$

Die Transformation der Kandidatenmenge in eine Kandidatenliste bzw. Liste von disjunkten Kandidatenmengen erfolgt durch eine 6-Ziel-Entscheidungsstrategie auf der Basis der Momente mom(r, $\overline{bias}(x_{S,i}^t)_{K,j}^2 \mid SM_{BV|K,j}^t)$, mom(r, $\overline{\sigma}(x_{S,i}^t)_{K,j}^{2'} \mid SM_{BV|K,j}^t)$, r = 1, 2, 3. Da Bias und Varianz Fehlermaße sind, die minimiert werden sollen, bedeutet dies auch für die zentralen Momente eine Minimierung durch einen 6-dimensionalen Vektor.

Ähnlich der Vorgehensweise beim Wild-Bootstrap (Mammen (1992[206]), Efron & Tibshirani (1993[105]), Shao & Tu (1995[309])) könnte aus der diskreten Häufigkeitsverteilung der Bias-Quadratwerte und der diskreten Häufigkeitsverteilung der Output-Varianzwerte, die aus der Stützpunktmenge bezüglich eines Kandidaten hervorgehen, jeweils die ersten drei zentralen Momente berechnet werden. Mit diesen drei Parametern wird eine stetige Häufigkeitsverteilung für die Bias-Quadrat- und Output-Varianzwerte festgelegt. Die Dominanz zweier Kandidaten wird im folgenden durch den Vergleich der beiden stetigen Verteilungen entschieden, worauf jedoch nicht näher eingegangen werden soll.

5.3) Integration von Relevanz- und Modell-Maximierungskriterium

Die Erzeugung der primären Ergebnismenge ist unabhängig von einem der beiden Selektionskriterien, sodass DVM_{prim}^{t} als gemeinsame Grundlage verwendet wird, um Integrationsstrategien zu formulieren. Ziel der Integration ist zunächst die Erzeugung der beiden sekundären Listen $DV_{Rel,sec}^{t}$ und $DV_{Mod,sec}^{t}$ entsprechend dem Relevanz- und dem Modell-Maximierungskriterium. Im weiteren sind folgende Vorgehensweisen unterscheidbar, die sich auf die Präsentation der Ergebnisse beziehen:

1) Bekannte Präsentationsstrategie mit genau einer tertiären Liste $DV_{RelMod,ter}^{t}$, was eine Integration der beiden Listen $DV_{Rel,sec}^{t}$ und $DV_{Mod,sec}^{t}$ erfordert.
2) Neue Präsentationsstrategie mit einer tertiären Liste $DV_{Rel,ter}^{t}$ aus $DV_{Rel,sec}^{t}$ und einer tertiären Liste $DV_{Mod,ter}^{t}$ aus $DV_{Mod,sec}^{t}$, was die Festlegung einer Integrationsstrategie erübrigt.
3) Neue Präsentationsstrategie, welche die Schnittmenge aus $DV_{Rel,ter}^{t}$ und $DV_{Mod,ter}^{t}$ als eigene Datenstruktur neben den beiden Restlisten darstellt.

5.3.1) Erzeugung einer gemeinsamen tertiären Ergebnisliste

Grundlegend muss die Struktur der Ränge der Listen $DV_{Rel,sec}^{t}$ und $DV_{Mod,sec}^{t}$ in Abhängigkeit der Anzahl der verwendeten Attribute unterschieden werden. Eine sekundäre Liste kann auf einem Rang

1) einen einzelnen Dokumentvektor x_j^t, oder
2) eine Menge von Dokumentvektoren besitzen.

Im ersten Fall wurde die Liste durch ein einzelnes Attribut, d.h. durch ein Ein-Ziel-Entscheidungskriterium erzeugt, bzw. durch die Integration mehrerer Attribute zu einem einzelnen Kriterium, wie z.B. der Addition eines Bias-Quadrat- und eines Varianzwertes zu einem MSE-Wert. Im zweiten Fall wird ein Mehr-Ziel-Entscheidungskriterium wie das Pareto-Kriterium unterstellt, wobei die Hierarchie disjunkter Dokumentvektorenmengen durch eine Liste dargestellt wird. Jede Menge $PM(DVM_{prim}^{t})_k$ kann dabei mehr als ein Element besitzen. Dementsprechend existieren vier Szenarien, die bei der Erzeugung einer gemeinsamen tertiären Dokumentvektorenliste berücksichtigt werden müssen:

1) $DV_{Rel,sec}^{t} = (x_j^t \mid \forall x_j^t \in DVM_{prim}^{t})$ und $DV_{Mod,sec}^{t} = (x_j^t \mid \forall x_j^t \in DVM_{prim}^{t})$.
2) $DV_{Rel,sec}^{t} = (x_j^t \mid \forall x_j^t \in DVM_{prim}^{t})$ und $DV_{Mod,sec}^{t} = (PM(DVM_{prim}^{t})_k \mid k = 1, ...)$.
3) $DV_{Rel,sec}^{t} = (PM(DVM_{prim}^{t})_k \mid k = 1, ...)$ und $DV_{Mod,sec}^{t} = (x_j^t \mid \forall x_j^t \in DVM_{prim}^{t})$.
4) $DV_{Rel,sec}^{t} = (PM(DVM_{prim}^{t})_k \mid k = 1, ...)$ und $DV_{Mod,sec}^{t} = (PM(DVM_{prim}^{t})_k \mid k = 1, ...)$.

Je nachdem, ob Ränge von Einzel-Elementen, Einzel-Elementen und Elementmengen oder von Elementmengen aggregiert werden sollen, müssen unterschiedliche Verfahren angewendet werden.

Liegen die beiden Listen $DV_{Rel,sec}^{t}$ und $DV_{Mod,sec}^{t}$ vor, so sind die beiden folgenden weiteren Vorgehensweisen zu unterscheiden, wenn als Ziel eine gemeinsame tertiäre Liste $DV_{RelMod,ter}^{t}$ angenommen werden soll:

1) Unabhängige Erzeugung der tertiären Listen $DV_{Rel,ter}^{t}$ und $DV_{Mod,ter}^{t}$ aus $DV_{Rel,sec}^{t}$ und $DV_{Mod,sec}^{t}$, sowie Integration zu der gemeinsamen tertiären Liste $DV_{RelMod,ter}^{t}$.
2) Direkte Integration der beiden sekundären Listen zu einer gemeinsamen tertiären Liste $DV_{RelMod,ter}^{t}$.

Allgemein soll gelten, dass bei einer unabhängigen Erzeugung der tertiären Listen die Struktur der sekundären Liste auf die Struktur der tertiären Liste vererbt wird. Bestehen die Komponenten der sekundären Liste aus Dokumentvektoren, so werden die Komponenten der tertiären Liste ebenfalls aus Dokumentvektoren bestehen. Bestehen die Komponenten der sekundären Liste aus Dokumentvektorenmengen, so werden die Komponenten der tertiären Liste ebenfalls aus Mengen bestehen.

Bei der unabhängigen Erzeugung kann jeweils eine Soll- bzw. Muss-Anzahl für die Dokumentvektoren der beiden tertiären Listen gegeben sein, mit λ_{Rel} für $DV_{Rel,ter}^{t}$ und λ_{Mod} für $DV_{Mod,ter}^{t}$. D.h. aus $DV_{Rel,sec}^{t}$ werden die besten λ_{Rel} Dokumentvektoren in $DV_{Rel,ter}^{t}$ übernommen, und aus $DV_{Mod,sec}^{t}$ werden die besten λ_{Mod} Dokumentvektoren in $DV_{Mod,ter}^{t}$ übernommen. Die Übernahme geschieht in Abhängigkeit von der Datenstruktur der Listen. Besteht eine Liste aus Dokumentvektoren, so werden einfach die ersten λ_{Rel} bzw. λ_{Mod} Dokumentvektoren in der gleichen Reihenfolge übernommen, wobei die tertiäre Liste somit ebenfalls aus Dokumentvektoren besteht. Besteht eine Liste aus Dokumentvektorenmengen, so werden diese beginnend mit der ersten Menge $PM(DVM_{prim}^{t})_1$ übernommen, bis die Menge erreicht ist, bei deren Übernahme die Anzahl λ_{Rel} bzw. λ_{Mod} erreicht bzw. überschritten wird.

Beim Erreichen der Anzahl wird die weitere Übernahme abgebrochen, während bei einer Überschreitung ein zusätzliches Kriterium eingeführt werden muss, da alle Elemente innerhalb einer Menge als gleichwertig gelten. Handelt es sich bei λ_{Rel} bzw. λ_{Mod} um eine Muss-Anzahl, so werden aus der betrachteten Menge so viele Dokumente durch eine zufällige Ziehung ohne Zurücklegen ausgewählt, bis die Anzahl in der tertiären Liste erreicht ist. Handelt es sich bei λ_{Rel} bzw. λ_{Mod} um eine Soll-Anzahl, so kann einfach die ganze Menge übernommen werden, sodass eine iterationsspezifische Anzahl λ_{Rel}^{t} bzw. λ_{Mod}^{t} erzeugt wird. Durch die Vererbung der Struktur der Listen besteht die tertiäre Liste ebenfalls aus Mengen, unabhängig ob die letzte Menge ganz oder teilweise übernommen wurde.

Anstatt der Darstellung der Dokumentvektorenlisten, soll im weiteren eine Darstellung auf der Basis von Stimuli m_j^t verwendet werden, d.h. es werden die beiden Listen $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ verwendet, unabhängig von den Datenstrukturen, die den Komponenten zugeordnet werden. Ziel der Erzeugung einer gemeinsamen Ergebnisliste $ML_{Rel\cup Mod,ter}^{t}$. Im allgemeinen Fall kann zunächst eine Vereinigungsmenge $M_{Rel\cup Mod,ter}^{t}$ der Listenkomponenten erzeugt werden, unabhängig ob die Komponenten Stimuli oder Stimulusmengen enthalten.

$$M_{Rel\cup Mod,ter}^{t} = ML_{Rel,ter}^{t} \cup ML_{Mod,ter}^{t}. \tag{872}$$

Es folgt ein on-scratch Aufbau der Liste $ML_{Rel\cup Mod,ter}^{t}$ aus der Menge $M_{Rel\cup Mod,ter}^{t}$, indem die vorhandenen Attribute der Stimuli zusammengenommen werden und ein hierarchieerzeugendes Mehr-Ziel-Verfahren angewendet wird. Unabhängig ob jedem Stimulus aus $ML_{Rel,ter}^{t}$ bzw. $ML_{Mod,ter}^{t}$ ein oder mehrere Attribute zugeordnet sind, es ergibt sich in $ML_{Rel\cup Mod,ter}^{t}$ immer eine Liste disjunkter Stimulusmengen, da mindestens zwei Attribute vorliegen werden, sodass ein Mehr-Ziel-Verfahren angewendet werden muss. Diese Vorgehensweise bietet sich insbesondere dann an, wenn Dokumentvektoren als Komponenten in beiden Listen vorliegen, da keine anderweitige Strukturierung wie Mengen nicht-dominanter Stimuli vorliegt. Es sollen zusätzlich Verfahren beschrieben werden, welche die Struktur der Listen

$ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ verwenden, um $ML_{Rel\cup Mod,ter}^{t}$ aufzubauen, anstatt die Strukturen durch die Bildung der Vereinigungsmenge zu zerstören und dann wieder aufzubauen.

5.3.1.1) Dokumentvektoren als Komponenten in beiden Listen

Hierbei soll gelten, dass den Dokumentvektoren aus DVM_{prim}^{t} beispielhaft jeweils eine fehlerkorrigierte Relevanzschätzung $rel(x_j^t)_{korrig}^{\wedge}$ und ein MSE-Integral $IMSE[AM(rel(x) \mid M_{DV(m)}^{\leq t-1} \cup \{m_{K,j}^{t}\})]^{\wedge} \equiv IMSE(x_j^t)$ als Attribut für die Relevanz- bzw. Modell-Maximierung zugeordnet wurden. Für die beiden Stimuluslisten $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ bedeutet dies:

$$ML_{Rel,ter}^{t} = (m_{Rel,j}^{t} = (x_{Rel,j}^{t}, rel(x_{Rel,j}^{t})_{korrig}^{\wedge}, IMSE(x_{Rel,j}^{t})) \mid$$
$$rel(x_{Rel,j}^{t})_{korrig}^{\wedge} > rel(x_{Rel,j+1}^{t})_{korrig}^{\wedge}; j = 1, ..., \lambda_{Rel}),$$
$$ML_{Mod,ter}^{t} = (m_{Mod,j}^{t} = (x_{Mod,j}^{t}, rel(x_{Mod,j}^{t})_{korrig}^{\wedge}, IMSE(x_{Mod,j}^{t})) \mid$$
$$IMSE(x_{Mod,j}^{t}) < IMSE(x_{Mod,j+1}^{t}); j = 1, ..., \lambda_{Mod}). \quad (873)$$

Da die sekundären Dokumentvektorlisten aus der primären Menge DVM_{prim}^{t} unter Beibehaltung der Anzahl der Dokumentvektoren gebildet werden, bedeutet dies, dass die Wahrscheinlichkeit groß ist, dass die beiden Listen $DV_{Rel,ter}^{t}$ und $DV_{Mod,ter}^{t}$ nicht disjunkt sind. D.h. in der Regel wird davon ausgegangen, dass eine nicht leere Schnittmenge der tertiären Listen auf Stimulusbasis existiert:

$$M_{Rel\cap Mod,ter}^{t} = \{m_{RelMod,j}^{t} \mid m_{RelMod,j}^{t} \in ML_{Rel,ter}^{t} \wedge m_{RelMod,j}^{t} \in ML_{Mod,ter}^{t}\}. \quad (874)$$

Eine Vereinigung von $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ bedeutet somit, dass die Menge $ML_{Rel\cup Mod,ter}^{t}$ mit λ_{RelMod} Dokumentvektoren weniger Elemente besitzen wird als die Summe aus λ_{Rel} und λ_{Mod}:

$$M_{Rel\cup Mod,ter}^{t} = ML_{Rel,ter}^{t} \cup ML_{Mod,ter}^{t},$$
$$\lambda_{RelMod} \leq \lambda_{Rel} + \lambda_{Mod}. \quad (875)$$

Die Strukturierung der Vereinigungsmenge $M_{Rel\cup Mod,ter}^{t}$ in eine Liste $ML_{Rel\cup Mod,ter}^{t}$ kann durch unterschiedliche Kriterienklassen durchgeführt werden, wobei im weiteren interne Kriterien bzw. die Kombination eines internen und externen Kriteriums betrachtet werden sollen:

1) externe Kriterien wie z.B. eine Zufallsauswahl mit einer erhöhten Auswahlwahrscheinlichkeit aus $ML_{Rel,ter}^{t}$.
2) interne Kriterien, welche vorliegende Attribute der Stimuli bzw. deren Rangplätze in den beiden tertiären Listen nutzen.

Eine Verwendung von Rangplätzen muss die nur teilweise Überdeckung der tertiären Listen berücksichtigen, d.h. es existieren Stimuli, die nur in einer tertiären Liste vorliegen und die somit nur einen Rangwert besitzen würden. Dies erschwert die Nutzung der Rangplätze, sodass als Grundlage die Rangplätze der sekundären Listen verwendet werden sollen, da jeder Dokumentvektor bzw. Stimulus aus den tertiären Listen in den sekundären Listen enthalten ist und somit einen Rangplatz besitzt. Rangplätze können durch die Bildung der Summe der Rangplätze integriert werden, d.h. jeder Stimulus erhält einen Summenrangplatz aus der Menge $\{2, 3, ..., \lambda_{Rel} + \lambda_{Mod}\}$ zugeordnet. Stimuli mit kleineren Rangplätzen dominieren dabei Stimuli mit einer größeren Rangsumme. Einzelne Stimuli können die gleiche Rangsumme

besitzen, sodass diese Stimuli untereinander als nicht-dominant bewertet werden müssen. Eine Ergebnisliste $ML_{Rel \cup Mod,ter}^{t}$, die auf der Basis von Rangplätzen gebildet wird, besitzt somit die Struktur einer Liste von Stimulusmengen, auch wenn die Basislisten $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ Stimuluslisten sind. Nur im Sonderfall, dass alle Stimuli abweichende Rangsummen besitzen, kann $ML_{Rel \cup Mod,ter}^{t}$ als Stimulusliste dargestellt werden.

Differenzierter lässt sich eine Ergebnisliste $ML_{Rel \cup Mod,ter}^{t}$ auf der Basis der Attribute $rel(x_j^t)_{korrig}\hat{}$ und $IMSE(x_j^t)$ bilden, da es sich in beiden Fällen um reelle Werte handelt, im Gegensatz zu den jeweils natürlichen Zahlen der Rangplätze. Wird ein Zwei-Ziel-Entscheidungsverfahren angewendet, so muss beachtet werden, dass die Relevanzschätzung maximiert und die MSE-Integralwerte minimiert werden müssen. Bei einer unterstellten Normierung von $rel(x)_{korrig}\hat{} \in [0, 1]$ kann die Maximierung in eine Minimierung überführt werden durch die Setzung von $rel(x)_{korrig}\hat{}^{-1}$, sodass eine Zwei-Ziel-Minimierungsstrategie mit Hilfe des Pareto-Kriteriums erzeugt werden kann. Da alle Elemente aus $M_{Rel \cup Mod,ter}^{t}$ in der Liste $ML_{Rel \cup Mod,ter}^{t}$ vertreten sein sollen, ist eine Pareto-Hierarchie z.B. durch sukzessive Deaktivierung von Paretomengen (siehe Abschnitt 2.4.2.2)) oder durch eine Pareto-Wettkampf-Hierarchie (siehe Abschnitt 2.4.2.3)) durchzuführen. In jedem Fall besitzt die Liste $ML_{Rel \cup Mod,ter}^{t}$ genau wie bei der Verwendung von Rangplätzen die Struktur einer Liste von disjunkten Stimulusmengen.

Eine besonders hervor zu hebende Klasse von Kombinationen von interner und externer Kriterien sind Verfahren, bei denen die Gewichtung von Relevanz- und Modell-Maximierung durch eine externe Metastrategie geregelt wird. Z.B. kann es als sinnvoll erachtet werden, wenn zu Beginn der Interaktion zwischen IRS und Agent das Modell-Maximierungskriterium höher gewichtet werden soll, da der Aufbau eines effektiven Relevanz-Approximationsmodells unterstützt werden soll, mit dem validere Relevanzschätzungen in den nachfolgenden Iterationen der IRS-Agent-Interaktion gebildet werden können. Dieser Ansatz führt zu einer dynamischen Regelung der Gewichtung zwischen Relevanz- und Modell-Maximierung, da im Verlauf der Iterationen die Übergewichtung für die Modell-Maximierung schnell abgebaut werden soll.

Eine Möglichkeit einer externen Gewichtungsregelung besteht in der Vorgabe der Anzahl von Stimuli, die aus den sekundären Listen in die tertiären Listen $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ übernommen werden. Die Anzahl λ_{Rel}^{t} bzw. λ_{Mod}^{t} ist somit interaktionsspezifisch und extern vorgegeben, entweder durch ein festes Scheduling oder dynamisch in Abhängigkeit von der erreichten Modellqualität. Überschreitet die Modellqualität ein Qualitätsschwellenwert, so wird die Übergewichtung der Modell-Maximierung aufgehoben, bzw. durch eine Übergewichtung der Relevanz-Maximierung ersetzt, wobei im Extremfall $\lambda_{Mod}^{t} := 0$ für die weiteren Iterationen gesetzt werden kann. Eine externe Setzung der Parameter λ_{Rel}^{t} bzw. λ_{Mod}^{t} hat den Vorteil, dass die Integrationsoperation, mit der die beiden tertiären Listen $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ zu einer gemeinsamen Ergebnisliste $ML_{Rel \cup Mod,ter}^{t}$ integriert werden, nicht verändert werden muss.

Die Veränderung dieser Integrationsoperation bei konstanten Parametern λ_{Rel} bzw. λ_{Mod} ist eine weitere potentielle Möglichkeit der Gewichtung von Relevanz- und Modell-Maximierung, die jedoch problematischer zu realisieren ist, da dies einen Eingriff in das verwendete Mehr-Ziel-Entscheidungsverfahren bedeutet. Prinzipiell wäre dies möglich, wenn eine gewichtete Summenfunktion als Aggregationsfunk-

tion der beiden verwendeten Attribute für die Relevanz- und Modell-Maximierung verwendet wird, durch welche eine Mehr-Ziel-Optimierung in eine Ein-Ziel-Optimierung transformiert wird. Eine Modifikation des Pareto-Kriteriums als einer wirklichen Mehr-Ziel-Strategie sollte jedoch unterlassen werden, da dies unabsehbare Folgen für den Optimierungsprozess haben könnte.

5.3.1.2) Dokumentvektormengen als Komponenten in beiden Listen

Werden den Dokumentvektoren aus DVM_{prim}^{t} jeweils mehr als ein Attribut für das Relevanz- und Modell-Maximierungskriterium zugeordnet, und werden die sekundären Listen $DV_{Rel,sec}^{t}$ und $DV_{Mod,sec}^{t}$ bzw. $ML_{Rel,sec}^{t}$ und $ML_{Mod,sec}^{t}$ durch ein hierarchiesches Mehr-Ziel-Verfahren ohne Aggregation der Einzel-Attribute gebildet, so besitzen die sekundären Listen Mengen in ihren Komponenten. Naheliegende Attribute für die Relevanz-Maximierung sind Relevanzschätzung, Relevanz-Bias und Relevanz-Varianz. Beispiel-Attribute für die Modell-Maximierung sind Bias- und Varianzschätzung bei indirekten Verfahren und Bias-Quadrat- und Relevanz-Varianz-Integral bei direkten Verfahren des aktiven Lernens.

Aus den sekundären Listen werden tertiäre Listen gebildet, indem die Mengen auf den ersten Rängen übernommen werden. Es werden beginnend mit dem ersten Rang die Mengen übernommen, bis eine Soll-Anzahl λ_{Rel} bzw. λ_{Mod} erreicht bzw. überschritten wird, wobei die Menge, bei der dieses Abbruchkriterium eintritt, wenn sie ganz übernommen würde, auch ganz übernommen wird. Diese Vorgehensweise ist dadurch effektiv, da die Stimuli innerhalb einer Menge untereinander nicht dominant sind, sodass keine begründbare Auswahl getroffen werden kann. Auf diese Weise ergeben sich die iterationsspezifischen Ist-Mächtigkeiten der Stimuli von λ_{Rel}^{t} und λ_{Mod}^{t}.

Korrespondiert zu der primären Dokumentvektorenmenge DVM_{prim}^{t} eine Stimulusmenge M_{prim}^{t}, und wird eine sukzessive Deaktivierung von Paretomengen verwendet, um eine Hierarchie zu erzeugen, die durch die Listen $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ beschrieben sind, so werden die tertiären Listen durch die sukzessiven Paretomengen $PM(M_{prim}^{t})_{Rel,k}$ und $PM(M_{prim}^{t})_{Mod,k}$ beschrieben, die vereinigt λ_{Rel}^{t} bzw. λ_{Mod}^{t} Stimuli enthalten:

$$ML_{Rel,ter}^{t} = (PM(M_{prim}^{t})_{Rel,k} \mid \#\{\bigcup_{k} PM(M_{prim}^{t})_{Rel,k}\} = \lambda_{Rel}^{t}; k = 1, ..., k_{Rel,max}),$$
$$ML_{Mod,ter}^{t} = (PM(M_{prim}^{t})_{Mod,k} \mid \#\{\bigcup_{k} PM(M_{prim}^{t})_{Mod,k}\} = \lambda_{Mod}^{t}; k = 1, ..., k_{Mod,max}) \quad (876)$$

Ziel ist wiederum die Integration der tertiären Listen, d.h. die Integration von $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ zu $ML_{Rel\cup Mod,ter}^{t}$. Ein Lösungsvorschlag ist die komponentenweise Vereinigung korrespondierender Paretomengen, d.h. $PM(M_{prim}^{t})_{Rel,k}$ und $PM(M_{prim}^{t})_{Mod,k}$ werden für gleiche k vereinigt, sodass sich eine integrierte Liste ergibt:

$$ML_{Rel\cup Mod,ter}^{t} = (PM(M_{prim}^{t})_{RelMod,k} = PM(M_{prim}^{t})_{Rel,k} \cup PM(M_{prim}^{t})_{Mod,k} \mid k = 1, ...). \quad (877)$$

Problemlos funktioniert dieser Ansatz, wenn in beiden Listen $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ die gleiche Anzahl von Komponenten vorliegen, d.h. wenn $k_{Rel,max} = k_{Mod,max}$. Doch auch eine abweichende Anzahl von Komponenten erfordert keine prinzipielle Änderung, da eine vorhandene Menge immer mit der leeren Menge vereinigt werden kann, d.h. es wird eine Komponentenanzahl $k_{RelMod,max}$ unterstellt,

die gleich dem Maximum der beiden Listen ist. Belegt werden die neuen Komponenten bei der vorher kleineren Liste jeweils mit der leeren Menge. Dies kann beschrieben werden durch:

$$ML_{Rel\cup Mod,ter}{}^{t} = (PM(M_{prim}{}^{t})_{RelMod,k} = PM(M_{prim}{}^{t})_{Rel,k} \cup PM(M_{prim}{}^{t})_{Mod,k} \mid k = 1, ...k_{RelMod,max} = \max\{k_{Rel,max}, k_{Mod,max}\}). \quad (878)$$

Durch die Vereinigung werden doppelt auftretende Stimuli korrigiert, wobei Elemente einer Menge zunächst als untereinander nicht-dominant betrachtet werden müssen. Ein Nachteil dieser Vereinigungsstrategie besteht darin, dass Stimuli in zwei Mengen auftreten können, sodass die Mengen nicht mehr disjunkt sind. Da keine semantische Begründung existiert, warum ein Dokument in einem Ranking an zwei Stellen vorkommen soll, muss diese Eigenschaft korrigiert werden, indem z.B. das zweite Auftreten eines Stimulus entfernt wird, d.h. tritt ein Stimulus z.B. in $PM(M_{prim}{}^{t})_{RelMod,2}$ und $PM(M_{prim}{}^{t})_{RelMod,4}$ auf, so wird er in $PM(M_{prim}{}^{t})_{RelMod,4}$ entfernt.

Der zweite Nachteil einer solchen Vereinigung besteht jedoch darin, dass das Ranking-Prinzip weiter aufgeweicht wird, indem potentiell größere Mengen, d.h. eine unstrukturierte Datenstruktur, erzeugt wird. Entsprechend dieser Argumentation kann gefordert werden, dass die einzelnen Mengen $PM(M_{prim}{}^{t})_{RelMod,k}$ intern strukturiert werden. Eine binäre Strukturierung ergibt sich, wenn Stimuli vorliegen, die in der Schnittmenge $PM(M_{prim}{}^{t})_{Rel,k} \cap PM(M_{prim}{}^{t})_{Mod,k}$ liegen. Diese Stimuli können als erste Komponente einer zwei-elementigen Liste verwendet werden, durch die jede Menge $PM(M_{prim}{}^{t})_{RelMod,k}$ ersetzt wird. Die zweite Komponente bildet die Restmenge, d.h. $PM(M_{prim}{}^{t})_{RelMod,k}$ ohne die Schnittmenge:

$$ML_{Rel\cup Mod,ter}{}^{t} = (PM(M_{prim}{}^{t})_{Rel,k} \cap PM(M_{prim}{}^{t})_{Mod,k}, PM(M_{prim}{}^{t})_{RelMod,k} \setminus \{PM(M_{prim}{}^{t})_{Rel,k} \cap PM(M_{prim}{}^{t})_{Mod,k}\} \mid k = 1, ..., k_{RelMod,max}). \quad (879)$$

Eine wesentlich feinere Strukturierung kann sich ergeben, wenn aus den Elementen der Menge $PM(M_{prim}{}^{t})_{RelMod,k}$ eine neue Pareto-Hierarchie erzeugt wird, d.h. $PM(M_{prim}{}^{t})_{RelMod,k}$ wird als Grundmenge betrachtet, aus der z.B. durch sukzessive Deaktivierung Paretomengen $PM(PM(M_{prim}{}^{t})_{RelMod,k})_{p}$ erzeugt werden, sodass sich eine allgemeine Struktur aus zweifach geschachtelten Listen ergibt:

$$ML_{Rel\cup Mod,ter}{}^{t} = ((PM(PM(M_{prim}{}^{t})_{RelMod,k})_{p} \mid p = 1, ...) \mid k = 1, ..., k_{RelMod,max}). \quad (880)$$

5.3.1.3) Dokumentvektoren als Komponenten der ersten und Dokumentvektormengen als Komponenten der zweiten Liste

Eine solche Situation setzt voraus, dass in der ersten tertiären Liste Stimuli genau ein Attribut besitzen, und somit ein eindeutiges Ranking auf Elementbasis möglich ist, während in der zweiten tertiären Liste Stimuli mehrere Attribute besitzen, sodass eine Hierarchie von disjunkten Mengen untereinander nicht-dominanter Stimuli durch ein entsprechendes Mehr-Ziel-Entscheidungsverfahren erzeugt wird. Beispielhaft soll $ML_{Rel,ter}{}^{t}$ mit dem Attribut „korrigierte Relevanzschätzung“ die erste Liste und $ML_{Mod,ter}{}^{t}$ mit den Attributen Bias-Qudrat- und Relevanz-Varianz-Integral die zweite Liste beschreiben. Ziel ist wiederum die Integration zu $ML_{Rel\cup Mod,ter}{}^{t}$, wobei die Datenstruktur der gemeinsamen Liste wieder eine

Hierarchie von Mengen sein muss, da als Grundlage deren Aufbaues Stimuli mit mehreren Attributen vorliegen, d.h. insgesamt drei Attribute im betrachteten Beispiel.

Als prinzipieller Lösungsvorschlag der Integration soll eine Modifikation der Vereinigung korrespondierender Komponenten aus dem vorangegangenen Abschnitt verwendet werden, wobei das Problem unterschiedlicher Komponentenanzahl in diesem Kontext besonders gravierend wird. Entsprechend dem Beispiel liegen in der ersten Liste $ML_{Rel,ter}^{t}$ λ_{Rel} Komponenten mit je einem Stimulus und in der zweiten Liste $ML_{Mod,ter}^{t}$ $k_{Mod,max}$ Komponenten mit jeweils einer Stimulusmenge vor, mit $\lambda_{Rel} >> k_{Mod,max}$:

$$ML_{Rel,ter}^{t} = (m_{Rel,j}^{t} \mid j = 1, ..., \lambda_{Rel}),$$
$$ML_{Mod,ter}^{t} = (PM(M_{prim}^{t})_{Mod,k} \mid k = 1, ..., k_{Mod,max}). \quad (881)$$

Bezüglich der Stimulusmengen können zwei Verfahrensklassen unterschieden werden:
1) Beibehaltung der Anzahl $k_{Mod,max}$ von Stimulusmengen.
2) Veränderung der Anzahl von Stimulusklassen.

Eine bevorzugte Strategie im Kontext der Beibehaltung der Anzahl von Stimulusmengen verteilt die Stimuli aus $ML_{Rel,ter}^{t}$ auf die $k_{Mod,max}$ Stimulusmengen $PM(M_{prim}^{t})_{Mod,k}$, wobei das Prinzip sinngemäß erhalten bleiben soll, dass korrespondierende Komponenten vereinigt werden. Dies bedeutet jedoch nicht, dass $m_{Rel,j=1}^{t}$ mit $PM(M_{prim}^{t})_{Mod,k=1}$, $m_{Rel,j=2}^{t}$ mit $PM(M_{prim}^{t})_{Mod,k=2}$ usw. vereinigt werden sollen. Vielmehr sollen die ersten Elemente aus $ML_{Rel,ter}^{t}$ mit $PM(M_{prim}^{t})_{Mod,k=1}$ vereinigt werden, die nachfolgenden mit $PM(M_{prim}^{t})_{Mod,k=2}$ usw., wobei die Anzahl der Elemente aus $ML_{Rel,ter}^{t}$ durch das Verhältnis von λ_{Rel} und $k_{Mod,max}$ ermittelt werden kann. Problemlos kann die Zuordnung erfolgen, wenn $\lambda_{Rel}/k_{Mod,max}$ restlos teilbar ist, da somit die ersten $\lambda_{Rel}/k_{Mod,max}$ Elemente von $ML_{Rel,ter}^{t}$ mit $PM(M_{prim}^{t})_{Mod,k=1}$ vereinigt werden, usw. Lässt sich λ_{Rel} und $k_{Mod,max}$ nicht restlos teilen, so muss eine Zuordnungsregelung in Verbindung mit einer Rundungsoperation festgelegt werden. Sei IntegerPart[$\lambda_{Rel}/k_{Mod,max}$] der Integer-Anteil des Bruches $\lambda_{Rel}/k_{Mod,max}$, so könnten die ersten IntegerPart[$\lambda_{Rel}/k_{Mod,max}$] + 1 Elemente aus $ML_{Rel,ter}^{t}$ mit der Menge $PM(M_{prim}^{t})_{Mod,k=1}$ vereinigt werden, gefolgt von dem Elementen IntegerPart[$\lambda_{Rel}/k_{Mod,max}$] + 2 bis 2 * IntegerPart[$\lambda_{Rel}/k_{Mod,max}$] + 1, die mit $PM(M_{prim}^{t})_{Mod,k=2}$ vereinigt werden, usw. bis alle Elemente aus $ML_{Rel,ter}^{t}$ zugeordnet sind. Es ergibt sich für die gemeinsame Liste $ML_{Rel\cup Mod,ter}^{t}$ eine Datenstruktur:

$$ML_{Rel\cup Mod,ter}^{t} = (PM(M_{prim}^{t})_{Mod,1} \cup \{m_{Rel,j}^{t} \mid j = 1 , ..., IntegerPart[\lambda_{Rel}/k_{Mod,max}] + 1\},$$
$$PM(M_{prim}^{t})_{Mod,k} \cup \{m_{Rel,j}^{t} \mid j = (k-1) * IntegerPart[.] + 2, ..., k * IntegerPart[.] + 1\}$$
$$\mid k = 2, ..., k_{Mod,max}), \text{ mit } IntegerPart[.] := IntegerPart[\lambda_{Rel}/k_{Mod,max}]. \quad (882)$$

Durch die Vereinigung werden zweifach auftretende Stimuli korrigiert, wobei trotzdem tendenziell größere Mengen gebildet werden, die intern strukturiert werden können, indem z.B. die verwendeten Attribute eine Pareto-Hierarchie der Stimuli erzeugen.

Ein Beispiel der Veränderung der Anzahl der Stimulusklassen ist die Vergrößerung der Anzahl der Mengen in $ML_{Mod,ter}^{t}$ von $k_{Mod,max}$ auf λ_{Rel}, wobei die neuen Komponenten mit der leeren Menge initialisiert werden. Im weiteren findet eine komponentenweise Vereinigung statt, d.h. $m_{Rel,1}^{t}$ wird mit

$PM(M_{prim}^t)_{Mod,1}$ vereinigt, usw. bis die ersten $k_{Mod,max}$ Paare gebildet wurden. Danach wird der jeweils nächste Stimulus mit der leeren Menge vereinigt, bis alle λ_{Rel} Stimuli gepaart wurden. Es ergibt sich für die gemeinsame Liste $ML_{Rel \cup Mod,ter}^t$ eine Datenstruktur:

$$ML_{Rel \cup Mod,ter}^t = (PM(M_{prim}^t)_{Mod,k} \cup \{m_{Rel,k}^t\}, \{m_{Rel,j}^t\} \mid k = 1, ..., k_{Mod,max}; j = k_{Mod,max} + 1, ..., \lambda_{Rel}). \quad (883)$$

5.3.2) Erzeugung zweier tertiären Ergebnislisten

Im vorangegangenen Abschnitt bestand das Hauptproblem darin, aus den beiden tertiären Listen $ML_{Rel,ter}^t$ und $ML_{Mod,ter}^t$ eine gemeinsame Liste $ML_{Rel \cup Mod,ter}^t$ in Abhängigkeit von der Art der Komponenten der Listen zu generieren. Die Integrationsproblematik entfällt, wenn auf das bekannte Präsentationsverfahren verzichtet wird, bei dem genau eine Dokumentliste gefordert wird, und wenn stattdessen zwei Dokumentlisten dem Agenten präsentiert werden. D.h. die zu $ML_{Rel,ter}^t$ und $ML_{Mod,ter}^t$ korrespondierenden Dokumentlisten werden dem Agenten explizit mit dem Hinweis präsentiert, dass es sich bei der ersten Liste um Dokumente handelt, deren erwartete Relevanz am größten ist, während es sich bei der zweiten Liste um Dokumente handelt, deren Relevanzwerte für den effektiven und effizienten Aufbau eines Nutzermodells in Form eines oder mehrerer Relevanz-Approximationsmodelle benötigt werden.

Die Darstellung der beiden Listen kann nicht vollständig unabhängig gestaltet sein, da in den beiden Listen $ML_{Rel,sec}^t$ und $ML_{Mod,sec}^t$ die gleichen Elemente enthalten sind, sodass eine hohe Wahrscheinlichkeit besteht, dass die beiden tertiären Listen $ML_{Rel,ter}^t$ und $ML_{Mod,ter}^t$ nicht disjunkt sind. D.h. es muss geklärt werden, was mit den Dokumentvektoren geschehen soll, die in beiden tertiären Listen vorliegen. Im Kontext des Retrievals mit dem Relevanz- und dem Modell-Maximierungskriterium ist eine nahe liegende Lösung die Dominanz der Dokumentvektoren in $ML_{Rel,ter}^t$, da erwartet werden kann, dass der Agent vorwiegend Dokumente bewertet, die voraussichtlich seinen Informationsbedürfnissen genügen. Stimuli, die in $ML_{Rel,ter}^t$ und $ML_{Mod,ter}^t$ liegen, d.h. in der Schnittmenge $M_{Rel \cap Mod,ter}^t$, werden aus $ML_{Mod,ter}^t$ entfernt, sodass eine neue tertiäre Liste $ML_{Mod,ter}^{t'}$ der Modell-Maximierung erzeugt wird:

$$ML_{Mod,ter}^{t'} = ML_{Mod,ter}^t \setminus M_{Rel \cap Mod,ter}^t. \quad (884)$$

Durch die Entfernung von Elementen aus einer Liste entstehen unerwünschte Lücken, sodass $ML_{Mod,ter}^{t'}$ so korrigiert werden soll, dass die entstandenen Lücken einfach gelöscht werden, und somit eine Liste mit $\lambda_{Mod} - \#M_{Rel \cap Mod,ter}^t$ Komponenten entsteht.

Spezialfälle ergeben sich, wenn die Soll-Anzahl der Elemente in $ML_{Mod,ter}^t$ deutlich kleiner ist als die Anzahl in $ML_{Rel,ter}^t$, und wenn alle Stimuli aus $ML_{Mod,ter}^t$ ebenfalls in $ML_{Rel,ter}^t$ liegen, sodass die Restliste $ML_{Mod,ter}^{t'}$ leer wird. Z.B. wird bei den direkten Verfahren des aktiven Lernens so vorgegangen, dass die Stimulusmenge immer nur mit genau einem Kandidaten erweitert wird, um die kombinatorische Explosion der Auswahl einer größeren Teilmenge aus der Kandidatenmenge zu umgehen. Dementsprechend dürfte nur der erste Rang der Liste $ML_{Mod,sec}^t$ in $ML_{Mod,ter}^t$ übernommen werden. Bei einer größeren Anzahl von Elementen, die von $ML_{Rel,sec}^t$ in $ML_{Rel,ter}^t$ übernommen werden, ist die Chance groß, dass der Dokumentvektor aus $ML_{Mod,ter}^t$ in $ML_{Rel,ter}^t$ enthalten ist, sodass $ML_{Mod,ter}^{t'}$ leer wird.

Eine leere Liste $ML_{Mod,ter}^{t‘}$ ist jedoch problemlos, da die ursprünglich in $ML_{Mod,ter}^{t}$ enthaltenen Elemente in $ML_{Rel,ter}^{t}$ liegen, sodass die Dokumentvektoren in jedem Fall einen Relevanzwert durch die Bewertung der korrespondierenden Dokumente enthalten. Die daraus hervorgehenden Stimuli bilden die iterationsspezifische Stimulusmenge $M_{DV(m)}^{t}$, die zusammen mit der bisherigen Gesamtstimulusmenge $M_{DV(m)}^{\leq t-1}$ zu der neuen Gesamtstimulusmenge $M_{DV(m)}^{\leq t}$ vereinigt wird. D.h. es ist unerheblich in welcher Liste die Dokumentvektoren liegen, wenn ihnen ein Relevanzwert zugeordnet wird, und sie somit zu Stimuli werden, mit denen das instanzbasierte Basis-Approximationsmodell $AM(rel(x) \mid M_{DV(m)}^{\leq t})$ direkt bzw. das prototypbasierte Modell $AM(rel(x) \mid N_{DV}^{t})$ indirekt aktualisiert wird. Zudem überführt eine leere Liste $ML_{Mod,ter}^{t‘}$ eine neue Präsentationsstrategie mit zwei Listen in die bekannte Präsentationsstrategie mit genau einer Liste, sodass dies keine Umstellungen bei Agenten erfordert.

5.3.3) Erzeugung dreier tertiären Ergebnislisten

Diese ebenfalls neue Präsentationsstrategie ergibt sich als Erweiterung der Erzeugung zweier Ergebnislisten, wenn die Schnittmenge $M_{Rel\cap Mod,ter}^{t}$ anders behandelt wird. Vorgeschlagen wurde die Entfernung der Elemente der Schnittmenge aus $ML_{Mod,ter}^{t}$, sodass sich $ML_{Mod,ter}^{t‘}$ ergibt, unter Beibehaltung der Struktur von $ML_{Rel,ter}^{t}$. Als Alternative kann $M_{Rel\cap Mod,ter}^{t}$ als eigene Datenstruktur verwendet werden, die als Menge oder als Liste $ML_{Rel\cap Mod,ter}^{t}$ verwendet wird, um eine erweiterte Präsentationsstrategie zu erzeugen. Präsentiert werden dem Agenten drei Listen von Dokumenten, die zu den Elementen der Listen $ML_{Rel\cap Mod,ter}^{t}$, $ML_{Rel,ter}^{t‘}$ und $ML_{Mod,ter}^{t‘}$ korrespondieren, mit

$$M_{Rel\cap Mod,ter}^{t} = ML_{Rel,ter}^{t} \cap ML_{Mod,ter}^{t},$$
$$ML_{Rel,ter}^{t‘} = ML_{Rel,ter}^{t} \setminus M_{Rel\cap Mod,ter}^{t},$$
$$ML_{Mod,ter}^{t‘} = ML_{Mod,ter}^{t} \setminus M_{Rel\cap Mod,ter}^{t}. \quad (885)$$

Die Entfernung von Elementen aus den Listen $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ erzeugt wiederum unerwünschte Lücken, die geschlossen werden, indem die Lücken einfach entfernt werden, sodass $ML_{Rel,ter}^{t‘}$ nun λ_{Rel} - $\#M_{Rel\cap Mod,ter}^{t}$ Komponenten und $ML_{Mod,ter}^{t‘}$ λ_{Mod} - $\#M_{Rel\cap Mod,ter}^{t}$ Komponenten besitzt.

Wird gefordert, dass alle Ergebnisse als Listen vorliegen sollen, so muss die Schnittmenge $M_{Rel\cap Mod,ter}^{t}$ in eine Liste $ML_{Rel\cap Mod,ter}^{t}$ umgewandelt werden, was mit Hilfe der Ränge der Elemente aus $M_{Rel\cap Mod,ter}^{t}$ in den beiden tertiären Listen $ML_{Rel,ter}^{t}$ und $ML_{Mod,ter}^{t}$ erfolgen kann, oder durch die Attribute $rel(x_j^t)_{korrig}$^ und $IMSE(x_j^t)$) in Verbindung mit einem Zwei-Ziel-Entscheidungsverfahren, das eine Hierarchie von disjunkten Stimulusmengen mit untereinander nicht dominanten Elementen erzeugt, wie z.B. die sukzessive Deaktivierung von Paretomengen (siehe Abschnitt 2.4.2.2)).

5.4) Lösungsansatz des Kombinatorikproblems bei direkten Verfahren durch die Integration eines Output- und eines Modell-Maximierungskriteriums

Das Kombinatorikproblem ist ein gravierendes Problem im Rahmen der direkten Verfahren zum aktiven Lernen (siehe Abschnitte 5.2.2) und 5.2.3)). Für jeden Kandidaten $m_{K,j}^t$ aus KM^t wird ein Modell $AM(x \mid M^t \cup \{m_{K,j}^t\})$ gebildet, dessen Eigenschaften bewertet werden, sodass die Auswahl des Modells mit den besten Eigenschaften nur die Auswahl genau dieses Kandidaten pro Iteration des aktiven Lernens rechtfertigen würde. Sollen z.B. zwei Kandidaten pro Iteration ausgewählt werden, so müssen alle Modelle $AM(x \mid M^t \cup \{m_{K,i}^t, m_{K,j}^t\})$ erzeugt werden, mit $m_{K,i}^t \neq m_{K,j}^t$ aus der Kandidatenmenge KM^t. Bei μ_K Kandidaten müssen $\mu_K * (\mu_K - 1)$ Paare gebildet werden, wobei für jedes Kandidatenpaar eine individuelle Stützpunktmenge $SM_{K,i,j}^t$ gebildet werden müsste, d.h. es müssten $\mu_K * (\mu_K - 1)$ Modellqualitäten gebildet werden, wenn genau eine Eigenschaft für die Modellqualität ermittelt wird. Werden mehrere Eigenschaften ermittelt, so erhöht sich diese Anzahl entsprechend. Sollen drei Kandidaten ausgewählt werden, so müssten $\mu_K * (\mu_K - 1) * (\mu_K - 2)$ Tripel gebildet und eine entsprechende Anzahl von Modellen getestet werden. Allgemein lässt sich dies als Ziehen ohne Zurücklegen von k Elementen aus einer μ_K Elemente großen Grundmenge beschreiben, ohne Berücksichtigung einer Reihenfolge, sodass die effektive Formulierung des direkten aktiven Lernens mehrerer Elemente nicht praktikabel ist.

Wird jedoch neben dem Modell-Maximierungskriterium in Form des direkten aktiven Lernens ein weiteres Kriterium wie ein Output-Maximierungskriterium verwendet, so sind Strategien anwendbar, um das Kombinatorikproblem zu entschärfen. Der grundlegende Ansatz besteht darin, eine Vorstrukturierung der Kandidatenmenge entsprechend dem zweiten Kriterium, d.h. dem Output-Maximierung, durchzuführen. Diese Vorstrukturierung wird verwendet, um eine handhabbare Anzahl ausgewählter Kandidatenmengen mit mehr als einem Element zu spezifizieren. Für diese Kandidatenmengen wird jeweils das erweiterte Modell ermittelt, gefolgt von der Ermittlung der Modellqualität.

5.4.1) Output-Maximierung

Im Kontext des IR bedeutet die Output-Maximierung die Maximierung der Relevanzschätzungen. Die weiteren Darstellungen sollen Bezug nehmen zu der kombinierten Bias-Quadrat- und Output-Varianz-Integral-Minimierung (siehe Abschnitt 5.2.2.3)). In diesem Rahmen wird die Kandidatenmenge KM^t zu KM_{BV}^t erweitert, wenn Bias-Quadrat- und Output-Varianz-Integrale verwendet werden sollen, mit der folgenden Datenstruktur:

$$KM_{BV}^t = \{m_{K,j}^t = (x_{K,j}^t, \_, y_{K,j}^{t\wedge}, f(x_{K,j}^t \mid AM(x \mid M^t)), bias(x_{K,j}^t \mid AM(bias(x) \mid M^t),$$
$$\sigma(x_{K,j}^t)^2 \mid j = 1, ..., \mu_K\}, \text{ mit } y_{K,j}^{t\wedge} \in \{y_{K,bias,j}^t, y_{K,bias,\sigma,j}^t\},$$
$$y_{K,bias,j}^t = f(x_{K,j}^t \mid AM(x \mid M^t)) - bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)),$$
$$y_{K,bias,\sigma,j}^t = [f(x_{K,j}^t \mid AM(x \mid M^t)) - bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)) - \sigma(x_{K,j}^t), f(x_{K,j}^t \mid AM(x \mid M^t)) -$$
$$bias(x_{K,j}^t \mid AM(bias(x) \mid M^t)) + \sigma(x_{K,j}^t)]. \quad (886)$$

Neben den Kandidaten wird eine nicht-individuelle Stützpunktmenge erzeugt, die für alle Modellbewertungen als Ausgangsbasis dient, und welche die folgende Datenstruktur besitzt:

$$SM_{BV}^{t} = \{m_{S,i}^{t} = (x_{S,i}^{t}, \_, f(x_{S,i}^{t} \mid AM(x \mid M^{t})), bias(x_{S,i}^{t} \mid AM(bias(x) \mid M^{t})), \sigma(x_{S,i}^{t})^{2}) \mid i = 1, ..., \mu_{S}\}. \quad (887)$$

Jedem Kandidaten $m_{K,j}^{t}$ wird bei der standardmäßigen Vorgehensweise eine individuelle Stützpunktmenge $M_{BV|K,j}^{t}$ zugeordnet, mit deren Auswertung Attribute für das Modell $AM(x \mid M^{t} \cup \{m_{K,j}^{t}\})$ geschätzt werden.

5.4.2) Vorstrukturierung der Kandidatenmenge

Im Gegensatz hierzu soll im weiteren zunächst eine Vorstrukturierung der Kandidatenmenge auf der Basis des zweiten Kriteriums, d.h. der Output-Maximierung, erfolgen. Als einfachstes Beispiel soll die korrigierte Output-Schätzung $y_{K,j}^{t\wedge}$ verwendet werden, d.h. es wird genau ein Attribut verwendet, um eine Vorstrukturierung zu erzeugen, sodass entsprechend dem Maximierungskriterium eine geordnete Liste der Kandidaten gebildet werden kann:

$$KL_{BV}^{t} = (m_{K,j}^{t} \mid y_{K,j}^{t\wedge} > y_{K,j+1}^{t\wedge}; j = 1, ..., \mu_{K}). \quad (888)$$

Werden mehrere Attribute im Rahmen eines zweiten Maximierungskriteriums verwendet, wie Output-Schätzung sowie verschiedene Formen der Fehlerschätzung an dem entsprechenden Inputvektor, die nicht zu einer gemeinsamen Funktion aggregiert werden sollen bzw. können, so wird die Anwendung eines Mehr-Ziel-Entscheidungsverfahrens notwendig. Naheliegend ist die Bildung einer Pareto-Hierarchie z.B. durch die sukzessive Deaktivierung von Paretomengen, d.h. es werden Paretomengen $PM(KM_{BV}^{t})_{k}$ gebildet, deren Elemente $m_{K,kj}^{t}$ und $m_{K,k,i}^{t}$ paarweise untereinander nicht dominant bezüglich der verwendeten Attribute sind:

$$KL_{BV}^{t} = (PM(KM_{BV}^{t})_{k} = \{m_{K,k,i}^{t}, m_{K,k,j}^{t} \mid dom(m_{K,k,i}^{t}, m_{K,k,j}^{t}) = 0; i, j = 1, ..., \mu_{K,k}\} \mid k = 1, ..., k_{max}^{t}). \quad (889)$$

5.4.3) Bildung von Kandidatenteilmengen

Es existieren mehrere sinnvolle Nutzungen dieser Vorstrukturierungen bezüglich der Definition von Kandidatenteilmengen, auf die sich die Erweiterung der Stimulusmenge M^{t} bezieht. Es soll davon ausgegangen werden, dass eine Anzahl von Kandidatenteilmengen vorgegeben wird, was jedoch nicht zwangsläufig eine externe Definition bedeutet. Bei der standardmäßigen Vorgehensweise werden μ_{K} Mengen mit jeweils genau einem Kandidaten verwendet, wobei die Kandidaten aus der primären Dokumentvektorenmenge DVM_{prim}^{t} hervorgehen, die z.B. bei einer GNG-SOM-klassifizierten Dokumentvektorenmenge selbstorganisiernd durch Voronoi-Regionen und nicht durch externe Faktoren festgelegt wird. Die Anzahl μ_{K} von Kandidatenmengen kann als Richtgröße beibehalten werden, wobei die Kandidatenmengen im gegebenen Kontext jeweils mehrere Kandidaten enthalten können.

Wird die Kandidatenliste $KL_{BV}^{t} = (m_{K,j}^{t} \mid y_{K,j}^{t\wedge} > y_{K,j+1}^{t\wedge}; j = 1, ..., \mu_{K})$ unterstellt, so kann die Bildung von Kandidatenmengen allgemein als Auswahl einer zusammenhängenden Teilliste oder mehrerer, unzu-

sammenhängender Teillisten operationalisiert werden. Bei einer zusammenhängenden Teilliste kann eine Terminologie analog den Gewinnerlisten-Verfahren verwendet werden (siehe Abschnitt 2.3.8)), indem die grundlegende Liste, das erste Element i_{start} und das letzte Element i_{end} spezifiziert wird:

$$L(KL_{BV}^t, i_{start}, i_{end}) = (m_{K,i(start)}^t, ..., m_{K,i(end)}^t). \quad (890)$$

Als zusätzliches Kriterium für die Auswahl von Stimuli soll gelten, dass Stimuli mit einem kleineren Rang innerhalb der Kandidatenliste entsprechend dem Output-Maximierungskriterium, d.h. Stimuli mit einer großen Relevanzschätzung, Stimuli mit einem größen Rang vorzuziehen sind.

Zu unterscheiden sind Auswahlverfahren, die Kandidatenteilmengen mit jeweils der gleichen Anzahl von Elementen erzeugen, oder die eine unterschiedliche Elementanzahl zulassen.

Ein Beispiel für die Erzeugung von Kandidatenmengen mit einer steigenden Anzahl von Stimuli verwendet immer den ersten Rang $m_{K,1}^t$ aus KL_{BV}^t als Startposition i_{start} , während als Endposition alle Ränge aus KL_{BV}^t verwendet werden, sodass insgesamt μ_K geordnete Kandidatenlisten $L(KL_{BV}^t, 1, k)$ entstehen, mit einem bis μ_K Elementen. Diese Kandidatenlisten werden in einer Menge KMM_{BV}^t zusammengefasst, welche die Rolle der Kandidatenmenge KM_{BV}^t übernimmt:

$$KMM_{BV}^t = \{L(KL_{BV}^t, 1, k) = (m_{K,1}^t, ..., m_{K,k}^t) \mid k = 1, ..., \mu_K\}. \quad (891)$$

Werden mehrere Attribute wie Output-Schätzung sowie Formen der Fehlerschätzung verwendet, die nicht aggregiert werden, so können Paretomengen $PM(KM_{BV}^t)_k$ gebildet werden. Diese sind natürliche Beispiele für Kandidatenmengen, um welche die Stimulusmenge M^t erweitert werden könnte, wobei auch hier gilt, dass Paretomengen mit einem kleinen Index k Mengen mit einem gösseren Index vorzuziehen sind. Das entsprechende Auswahlverfahren gehört zu denen mit einer unterschiedlichen Anzahl von Elementen, da k_{max}^t Paretomengen aus KM_{BV}^t gebildet werden, was deutlich weniger ist, als die Gesamtmenge von μ_K oder μ_K^t Kandidaten in KM_{BV}^t.

5.4.4) Direktes aktives Lernen bei vorstrukturierten Kandidatenteilmengen

In den weiteren Darstellungen soll unterstellt werden, dass eine Menge KMM_{BV}^t von Kandidatenmengen bzw. -listen $L(KL_{BV}^t, 1, k)$ vorliegen. Jede dieser Listen definiert eine neue Stimulusmenge $M^t \cup L(KL_{BV}^t, 1, k)$ und somit ein Modell $AM(x \mid M^t \cup L(KL_{BV}^t, 1, k))$, für das eine Bewertung ermittelt werden soll. Der erste Schritt hierfür ist die Bildung einer individuellen Stützpunktmenge $SM_{BV|L(KL(t),1,k)}^t$, deren Komponenten analog zu $SM_{BV|K,j}^t$ gestaltet sind, wobei die beiden Terme $bias(x_{S,i}^t)_{L(KL(t),1,k)}^2$ und $\sigma(x_{S,i}^t)_{L(KL(t),1,k)}^2$ die zentrale Bedeutung besitzen, da aus ihnen die Inregral-Approximationen als Mittelwert über alle Stützpunkte gebildet werden:

$$SM_{BV|L(KL(t),1,k)}^t = \{m_{S,i}^t = (x_{S,i}^t, \_, f(x_{S,i}^t \mid AM(x \mid M^t)), f(x_{S,i}^t \mid AM(x \mid M^t \cup L(KL_{BV}^t, 1, k))),$$
$$\Delta y_{S,i}^{t\wedge}, bias(x_{S,i}^t \mid AM(bias(x) \mid M^t)), \sigma(x_{S,i}^t)^2, bias(x_{S,i}^t)_{L(KL(t),1,k)}^2,$$
$$\sigma(x_{S,i}^t)_{L(KL(t),1,k)}^2, \sigma(x_{S,i}^t)_{L(KL(t),1,k)}^2,'),$$
$$IB[AM(x \mid M^t \cup L(KL_{BV}^t, 1, k))]^\wedge, IV[AM(x \mid M^t \cup L(KL_{BV}^t, 1, k))]^\wedge \mid i = 1, ..., \mu_S\}. \quad (892)$$

Das Bias-Quadrat kann problemlos auf den Fall einer Kandidatenmenge mit mehr als einem Kandidaten erweitert werden, indem $\Delta y_{S,i}^{t\wedge}$ als Differenz zwischen Output-Schätzung auf der Basis des neuen Modells und der Output-Schätzung auf der Basis des alten Modells berechnet wird:

$$bias(x_{S,i}^t)_{L(KL(t),1,k)}^2 = \Delta y_{S,i}^{t\wedge 2} + 2* \Delta y_{S,i}^{t\wedge} * bias(x_{S,i}^t) + bias(x_{S,i}^t)^2 \text{ , mit}$$
$$\Delta y_{S,i}^{t\wedge} = f(x_{S,i}^t \mid AM(x \mid M^t \cup L(KL_{BV}^t, 1, k))) - f(x_{S,i}^t \mid AM(x \mid M^t)). \quad (893)$$

Die Voraussetzung für die Output-Schätzung $f(x_{S,i}^t \mid AM(x \mid M^t \cup L(KL_{BV}^t, 1, k)))$ besteht in der Spezifizierung von Outputwerten für die Stimuli m_j^t aus $L(KL_{BV}^t, 1, k)$, für welche die richtigen Outputwerte y_j^t gegenwärtig unbekannt sind. In Cohn (1995[73]) wird im Kontext der Bias-Quadrat-Integral-Minimierung eine bias-korrigierte Output-Schätzung $y_j^{t\wedge}$ als Substitut für y_j^t verwendet, sodass Stimuli mit der Datenstruktur $m_j^t = (x_j^t, y_j^{t\wedge})$ vorliegen:

$$y_j^{t\wedge} = f(x_j^t \mid AM(x \mid M^t)) - bias(x_j^t)^\wedge, \text{ mit}$$
$$f(x_j^t \mid AM(x \mid M^t)) = 1/\nu_j \sum_k h(d_X(x_j^t, x_k^t)) * f(x_k^t),$$
$$bias(x_j^t)^\wedge = 1/\nu_j \sum_k h(d_X(x_j^t, x_k^t)) * bias(x_k^t),$$
$$\nu_j = \sum_k h(d_X(x_j^t, x_k^t)), \forall m_k^t = (x_k^t, f(x_k^t), f(x_k^t \mid AM(x \mid M^t \setminus \{m_k^t\})), bias(x_k^t)) \in M^t. \quad (894)$$

Bei der Kombination von Bias-Quadrat- und Varianz-Integral-Minimierung (siehe Abschnitt 5.2.2.3)) wurde beim Teilbereich Bias-Quadrat-Integral-Minimierung auf die Mögichkeit verwiesen, die Schätzung $y_j^{t\wedge}$ nicht nur mit Hilfe der Bias-Schätzung an der Stelle des Kandidaten m_j^t zu berechnen, sondern auch die Output-Varianz zu berücksichtigen. Wird die Outputschätzung durch das Modell $AM(x \mid M^t)$ verwendet, so ergibt sich das Intervall:

$$y_j^{t\wedge} = [f(x_j^t \mid AM(x \mid M^t)) - bias(x_j^t)^\wedge - \sigma(x_j^t), f(x_j^t \mid AM(x \mid M^t)) - bias(x_j^t)^\wedge + \sigma(x_j^t)], \text{ mit}$$
$$f(x_j^t \mid AM(x \mid M^t)) = 1/\nu_j \sum_k h(d_X(x_j^t, x_k^t)) * f(x_k^t),$$
$$bias(x_j^t)^\wedge = 1/\nu_j \sum_k h(d_X(x_j^t, x_k^t)) * bias(x_k^t),$$
$$\sigma(x_j^t)^2 = 1/\nu_j \sum_k h(d_X(x_j^t, x_k^t)) * (f(x_k) - f(x_j^t \mid AM(x \mid M^t))^2,$$
$$\nu_j = \sum_k h(d_X(x_j^t, x_k^t)), \forall m_k^t = (x_k^t, f(x_k^t), f(x_k^t \mid AM(x \mid M^t \setminus \{m_k^t\})), bias(x_k^t)) \in M^t. \quad (895)$$

Die Qualität der Schätzung $y_j^{t\wedge}$ kann weiterhin durch Bootstrap-Verfahren erhöht werden, indem Bootstrap-Stimulusmengen M_p^t erzeugt werden, mit denen Bootstrap-Einzelschätzungen $y_{j,p}^{t\wedge}$ berechnet werden, die zu einer Bootstrap-Gesamtschätzung $y_j^{t\wedge}$ aggregiert werden, worauf hier jedoch nicht explizit eingegangen werden soll. Die Outputschätzung $f(x_{S,i}^t \mid AM(x \mid M^t \cup L(KL_{BV}^t, 1, k)))$ berechnet sich dann zu:

$$f(x_{S,i}^t \mid AM(x \mid M^t \cup L(KL_{BV}^t, 1, k))) = 1/\nu_{S,i} [\sum_k h(d_X(x_{S,i}^t, x_k^t)) * f(x_k) + \sum_j h(d_X(x_{S,i}^t, x_j^t)) * y_j^{t\wedge}],$$
$$\nu_{S,i} = \sum_k h(d_X(x_{S,i}^t, x_k^t)) + \sum_j h(d_X(x_{S,i}^t, x_j^t)),$$
$$\forall m_k^t \in M^t \text{ und } \forall m_j^t = (x_j^t, y_j^{t\wedge}) \in L(KL_{BV}^t, 1, k). \quad (896)$$

Weiterhin ist für die Berechnung des neuen Bias-Quadrates $bias(x_{S,i}^t)_{L(KL(t),1,k)}^2$ an der Stelle des Stützpunktes $m_{S,i}^t$ die Bias $bias(x_{S,i}^t)$ an $m_{S,i}^t$ bezüglich der alten Stimulusmenge M^t notwendig, die durch

eine Kernel-Regression auf der Basis der Elemente aus M^t berechnet wird:

$$bias(x_{S,i}^t)^\wedge = 1/\nu_{S,i} \sum_k h(d_X(x_{S,i}^t, x_k^t)) * bias(x_k^t),$$
$$\nu_{S,i} = \sum_k h(d_X(x_{S,i}^t, x_k^t)), \forall m_k^t = (x_k^t, f(x_k^t), f(x_k^t \mid AM(x \mid M^t \setminus \{m_k^t\})), bias(x_k^t)) \in M^t. \quad (897)$$

Problematischer ist jedoch die Berechnung von $\sigma(x_{S,i}^t)_{L(KL(t),1,k)}^2$ bzw. $\sigma(x_{S,i}^t)_{L(KL(t),1,k)}^{2'}$, da das Verfahren von Cohn et al. (1995[74]) ausschließlich auf der Verwendung eines einzelnen Kandidaten $m_{K,j}^t$ basiert, und somit nur eine explizite Darstellung von $\sigma(x_{S,i}^t)_{K,j}^2$ in geschlossener Form liefert. Aus diesem Grunde soll die grundlegende Definition der Output-Varianz an der Stelle eines Stützpunktes $m_{S,i}^t$ bezüglich einer bestimmten Stimulusmenge verwendet werden, was durch Cohn (2000[75]) als gangbarer Weg verifiziert wurde. Wird die Output-Varianz an der Stelle des Stützpunktes $m_{S,i}^t$ bezüglich der alten Stimulusmenge M^t verwendet, so ergibt sich:

$$\sigma(x_{S,i}^t)^2 = 1/\nu_{S,i} \sum_j h(d_X(x_{S,i}^t, x_j^t)) * (f(x_j^t) - f(x_{S,i}^t \mid AM(x \mid M^t)))^2,$$
$$\nu_{S,i} = \sum_j h(d_X(x_{S,i}^t, x_j^t)), \forall m_j^t = (x_j^t, f(x_j^t)) \in M^t. \quad (898)$$

Wird als Stimulusmenge $M^t \cup \{m_{K,j}^t\}$ verwendet, so müsste der Stimulus $m_{K,j}^t = (x_{K,j}^t, y_{K,j}^t)$ vorliegen, wobei der Output $y_{K,j}^t$ bislang jedoch unbekannt ist. Hierfür kann die oben beschriebene bias-korrigierte Output-Schätzung bzw. das Output-Intervall $y_{K,j}^{t\wedge}$ als Substitut für $y_{K,j}^{t\wedge}$ verwendet werden. Weiterhin muss festgelegt werden, welches Modell verwendet werden soll, um die Output-Schätzung an der Stelle des Stützpunktes zu erzeugen, wobei das Ursprungsmodell $AM(x \mid M^t)$ bzw. das Modell auf Basis der erweiterten Stimulusmenge verwendet werden soll. Wird jeweils die erweiterte Stimulusmenge verwendet, so ergibt sich die Output-Varianz an der Stelle von $m_{S,i}^t$ bezüglich $M^t \cup \{m_{K,j}^t\}$:

$$\sigma(x_{S,i}^t)_{K,j}^2 = 1/\nu_{S,i} [\sum_j h(d_X(x_{S,i}^t, x_j^t)) * (f(x_j^t) - f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\})))^2 +$$
$$h(d_X(x_{S,i}^t, x_{K,j}^t)) * (y_{K,j}^{t\wedge} - f(x_{S,i}^t \mid AM(x \mid M^t \cup \{m_{K,j}^t\})))^2],$$
$$\nu_{S,i} = \sum_j h(d_X(x_{S,i}^t, x_j^t)) + h(d_X(x_{S,i}^t, x_{K,j}^t)), \forall m_j^t \in M^t \text{ und } m_{K,j}^t = (x_{K,j}^t, y_{K,j}^{t\wedge}). \quad (899)$$

In analoger Weise kann die Output-Varianz an der Stelle von $m_{S,i}^t$ bezüglich der Stimulusmenge $M^t \cup L(KL_{BV}^t, 1, k)$ bestimmt werden, indem allen Stimuli $m_{K,j}^t$ aus $L(KL_{BV}^t, 1, k)$ eine korrigierte Output-Schätzung $y_{K,j}^{t\wedge}$ zugeordnet wird:

$$\sigma(x_{S,i}^t)_{L(KL(t),1,k)}^2 = 1/\nu_{S,i} [\sum_k h(d_X(x_{S,i}^t, x_k^t)) * (f(x_k^t) - f(x_{S,i}^t \mid AM(x \mid M^t \cup L(KL_{BV}^t, 1, k))))^2 +$$
$$\sum_j h(d_X(x_{S,i}^t, x_{K,j}^t)) * (y_{K,j}^{t\wedge} - f(x_{S,i}^t \mid AM(x \mid M^t \cup L(KL_{BV}^t, 1, k))))^2],$$
$$\nu_{S,i} = \sum_k h(d_X(x_{S,i}^t, x_k^t)) + \sum_j h(d_X(x_{S,i}^t, x_{K,j}^t)),$$
$$\forall m_k^t \in M^t \text{ und } \forall m_{K,j}^t = (x_{K,j}^t, y_{K,j}^{t\wedge}) \in L(KL_{BV}^t, 1, k). \quad (900)$$

Nachdem jeder Stützpunkt $m_{S,i}^t$ aus der Stützpunktmenge $SM_{BV|L(KL(t),1,k)}^t$ einen Bias-Quadratwert $bias(x_{S,i}^t)_{L(KL(t),1,k)}^2$ und eine Output-Varianz $\sigma(x_{S,i}^t)_{L(KL(t),1,k)}^2$ zugeordnet bekommen hat, werden diese Werte über alle Stützpunkte aggregiert. Wird als Aggregationsfunktion in beiden Fällen der arithmetische Mittelwert, d.h. das r=1'te Moment, verwendet, so ergeben sich die beiden Integral-Approximationen $IB[AM(x \mid M^t \cup L(KL_{BV}^t, 1, k))]^\wedge$ und $IV[AM(x \mid M^t \cup L(KL_{BV}^t, 1, k))]^\wedge$.

Nachdem alle Kandidatenmengen bzw. -listen aus KMM_{BV}^{t} eine solche Bewertung erhalten haben, ergibt sich eine Datenstruktur:

$$KMM_{BV}^{t} = \{(L(KL_{BV}^{t}, 1, k), IB[AM(x \mid M^{t} \cup L(KL_{BV}^{t}, 1, k))]^{\wedge}, IV[AM(x \mid M^{t} \cup L(KL_{BV}^{t}, 1, k))]^{\wedge}) \mid k = 1, ..., \mu_{K}\}. \quad (901)$$

Aus dieser Menge kann eine Liste KML_{BV}^{t} erzeugt werden, indem die beiden Integrale additiv zu $IBV[AM(x \mid M^{t} \cup L(KL_{BV}^{t}, 1, k))]^{\wedge}$ aggregiert werden, sodass eine eindeutige Ein-Ziel-Entscheidung möglich wird. Werden die Listen $(L(KL_{BV}^{t}, 1, j), IBV[AM(x \mid M^{t} \cup L(KL_{BV}^{t}, 1, j))]^{\wedge})$ nach steigender Integral-Approximation geordnet, so ergibt sich:

$$KML_{BV}^{t} = ((L(KL_{BV}^{t}, 1, j), IBV[AM(x \mid M^{t} \cup L(KL_{BV}^{t}, 1, j))]^{\wedge}) \mid IBV[AM(x \mid M^{t} \cup L(KL_{BV}^{t}, 1, j))]^{\wedge} < IBV[AM(x \mid M^{t} \cup L(KL_{BV}^{t}, 1, j+1))]^{\wedge}; j = 1, ..., \mu_{K}). \quad (902)$$

Alternativ können die Bewertungen als Attribute eines Mehr-Ziel-Verfahrens wie einem hierarchischen Pareto-Verfahren verwendet werden, sodass als Datenstruktur von KML_{BV}^{t} eine Liste von disjunkten Mengen gebildet wird.

6) Zusammenfassung

Im Kapitel 1) wurde zunächst der Anwendungbereich dieser Arbeit spezifiziert, indem Information-Retrieval-Systeme als Spezialfall von Informationssystemen formal beschrieben wurden (siehe Abschnitt 1.1)). Es wurde daraufhin gezeigt, dass IR ein sehr komplexes Problem darstellt (siehe Abschnitt 1.2)), da mehrere Einzelkomponenten zusammen wirken, die jede für sich bereits ein komplexes Problem darstellt. Dazu gehören hochdimensionale Zusammenhänge (siehe Abschnitt 1.2.1)), nichtlineare und multimodale Zusammenhänge (siehe Abschnitt 1.2.2)), nicht-stationäre Zusammenhänge (siehe Abschnitt 1.2.3)), Unsicherheit und Vagheit (siehe Abschnitt 1.2.4)), Diversität der Agenten und ihrer Ziele (siehe Abschnitt 1.2.5)) sowie Mehrziel-Anforderungen (siehe Abschnitt 1.2.6)). Im weiteren wurde gezeigt, wie ein vielschichtiger Methodentransfer aus einer großen Anzahl von Disziplinen genutzt wurde, um diese Probleme des IR zu lösen (siehe Abschnitt 1.3)). Es folgte die Darstellung der allgemeinen Zielsetzung adaptiver Informationssysteme (siehe Abschnitt 1.4)), da angenommen wurde, dass nur auf diese Weise die dargestellten Probleme lösbar werden.

Im Rahmen der Einführung wurde daraufhin die Einbettung externer Informationsbeschaffung in ein Modell der allgemeinen Intelligenz beschrieben (siehe Abschnitt 1.5)), da geklärt werden musste, warum und wann in einem Problemlösungsprozess der Wechsel von internen, kognitiven Problemlösungsversuchen zu einer externen Informationsbeschaffung mit der Interaktion des Agenten mit einem IRS erfolgt.

Es folgte der Überblick über die weitere Arbeit, indem die drei Hauptbereiche Polyrepräsentation (siehe Abschnitt 1.6)), Relevanz-Approximationsmodelle (siehe Abschnitt 1.7)) und aktives Lernen (siehe Abschnitt 1.8)) verbal beschrieben werden, um damit die formalen Beschreibungen in den nachfolgenden Kapiteln zu erleichtern. Zum Abschluss des Überblickes wurden die Beziehungen der drei Hauptbereiche verbal und in einem semantischen Netz dargestellt (siehe Abschnitt 1.9)).

Im Abschnitt 1.6) wurde zunächst die Polyrepräsentation in Informationssystemen allgemein und für IRS im speziellen beschrieben (siehe Abschnitt 1.6.1)), wobei von einer Einschränkung auf eine einkomponentige Polyrepräsentation bezüglich der Definition des Information Retrievals ausgegangen wird, um die kombinatorische Explosion einer mehrkomponentigen Polyrepräsentation zu vermeiden. Es folgte die Unterscheidung zwischen Inter- und Intraparadigmen-Polyrepräsentation (siehe Abschnitt 1.6.2)). Daraufhin wurden Gründe beschrieben, warum eine Intraparadigmen-Polyrepräsentation im Kontext des Vektorraummodells ausreicht, um die erwarteten Vorteile der Polyrepräsentation zu nutzen (siehe Abschnitt 1.6.3)), gefolgt von einer Spezifizierung anderer Gründe der Verwendung der Polyrepräsentation (siehe Abschnitt 1.6.4)), die vor allem durch das Machine Learning begründet sind.

Im Abschnitt 1.7) wurde zunächst der Begriff der Approximation beschrieben, gefolgt von der Spezifizierung von Interpolation und Regression. Es wurde dargestellt, dass im Kontext des IR ein Relevanz-Approximationsmodell direkt als ein Nutzermodell interpretiert werden kann, und dass eine Polyrepräsentation von solchen Modellen Effektivitätsvorteile besitzt, sowie für bestimmte Formen des aktiven Lernens von Bedeutung ist.

Im Abschnitt 1.8) wurde zunächst passives und aktives Lernen differenziert (siehe Abschnitt 1.8.1)), gefolgt von der Unterscheidung zwischen geschlossener und offener Lernmenge (siehe Abschnitt 1.8.2)), sowie zwischen direkten und indirekten Verfahren des aktiven Lernens (siehe Abschnitt 1.8.3)). Es wurde auf die unterschiedlichen Charakteristika von direkten und indirekten Verfahren bezüglich Effizienz- und Effektivitätseigenschaften hingewiesen (siehe Abschnitt 1.8.4)). Im Kontext des IR wurde darauf hingewiesen, dass die Maximierung der Relevanz und die Maximierung des Modellaufbaus konkurrierende Ziele sind (siehe Abschnitt 1.8.5)). Im letzten Abschnitt wurde die Bedeutung der Modell-Polyrepräsentation beim aktiven Lernen vertieft (siehe Abschnitt 1.8.6)), wobei dies für indirekte Verfahren unbedingt notwendig ist und für direkte Verfahren optional anwendbar ist.

Im Kapitel 2) wurden die methodischen Grundlagen beschrieben, um die vorliegende Arbeit weitgehend selbsterklärend zu gestalten. Hierzu wurden zunächst die Approximationsverfahren formal beschrieben, beginnend bei der Unterscheidung zwischen Interpolation und Regression (siehe Abschnitt 2.1.2)), der Unterscheidung zwischen symbolischer und stützpunktorientierter Approximation (siehe Abschnitt 2.1.2.1)) und der Unterscheidung zwischen instanz- und prototypbasierter Approximation (siehe Abschnitt 2.1.2.2)). Es wurde ein Framework des überwachten und unüberwachten Lernens beschrieben, indem insbesondere überwachtes Lernen in einen unüberwachten und einen überwachten Teilprozess zerlegt werden kann, mit dem Batch-Lernprozesse (siehe Abschnitt 2.1.7)) erzeugt werden können. Im weiteren wurde die Verfahrensklasse der Local-Weighted-Regression als überwachtes Lernverfahren eingeführt (siehe Abschnitt 2.1.3)), sowie die sensorische-SOM (siehe Abschnitt 2.1.4)) und die beiden Erweiterungen GNG-SOM (siehe Abschnitt 2.1.5)) und SC-GNG-SOM (siehe Abschnitt 2.1.6)) als unüberwachte Lernverfahren. Entsprechend dem Framework aus Abschnitt 2.1.2.3) konnte die SC-GNG-SOM zu einem überwachten Lernverfahren durch einen Batch-Lernprozess erweitert werden (siehe Abschnitt 2.1.7)), das in dieser Arbeit an vielen Stellen genutzt wird. Im Hinblick auf die Query-Polyrepräsentation (siehe Abschnitt 3.6.1.2)) und der Query-Modifikation bzw. -Expansion mit Hilfe von Merkmals-SOM-Graphen (siehe Abschnitte 3.8.5) und 3.8.6)) wurden an dieser Stelle die methodischen Grundlagen in Form der Aktivitätsausbreitung in GNG-Graphen gelegt (siehe Abschnitt 2.1.8)).

Nach den Darstellungen zur Approximation wurden Basis-Verfahren des Resamplings formal beschrieben (siehe Abschnitt 2.2)), darunter der einfache Stimulus-Bootstrap (siehe Abschnitt 2.2.1)), der Restwert-Bootstrap (siehe Abschnitt 2.2.2)), sowie der Moving-Blocks-Bootstrap (siehe Abschnitt 2.2.3)) als ein Bootstrapverfahren, das für sequentielle Daten wie Zeitreihen oder Zeichenketten geeignet ist.

Grundlage jeder Modell-Selektion ist die Bewertung der Kandidaten-Modelle, sodass eine Darstellung von Verfahren zur Bestimmung von Modellqualitäten folgte (siehe Abschnitt 2.3)). Zunächst wurde mit dem Quantifizierungsfehler ein Qualitätsmaß für unüberwachte Modelle wie die SC-GNG-SOM beschrieben (siehe Abschnitt 2.3.1)), gefolgt von Qualitätsmaßen für überwachte Modelle wie lokale MSE-Werte (siehe Abschnitt 2.3.2)), MSE-Integral (siehe Abschnitt 2.3.3)), Varianz- und Bias-Integrale (siehe Abschnitt 2.3.4)). Stimuli, die Stützpunkte in einem Input- und einem Outputraum besitzen, können in Verbindung mit einem Output-Approximationsmodell bzw. einer Menge von Output-Approximationsmodellen Bias- und Output-Varianzwerte zugeordnet werden, die zusammen mit einem Verfahren der stützpunktbasierten Approximation Bias- und Varianz-Approximationsmodelle bilden (siehe Abschnitt 2.3.5)), die als Fehlermodelle bzw. Unsicherheitsmodelle genutzt werden können. Um eine

Spannweite vorliegender Outputfehler eines Output-Approximationsmodells angeben zu können, wäre eine Suche nach Inputvektoren mit extremalen Outputwerten sinnvoll, d.h. es wird eine Form der aktiven Stimulusgewinnung unterstellt, mit der ein Modell getestet werden kann (siehe Abschnitt 2.3.6)). Der Quantifizierungsfehler, MSE-Werte sowie die unterschiedlichen Integraltypen stellen quasi eine Mittelwertsbildung dar, die lokale Abweichungen eliminiert. Es wird gezeigt, dass effektivere Qualitätsmaße gebildet werden können, wenn neben einem Mittelwert als zentrales Moment erster Ordnung Momente höherer Ordnung verwendet werden (siehe Abschnitt 2.3.7)). Für diese Arbeit von besonderer Bedeutung war die Einführung der Bestimmung der Modellqualität durch ein Gewinnerlisten-Verfahren (siehe Abschnitt 2.3.8)), da auf diese Weise keine Zerlegung der Stimulusmenge in Lern- und Test-Menge erforderlich ist, sodass alle Stimuli an Lern- wie Testverfahren teilnehmen können.

Im Kontext der Darstellung warum IR als ein komplexes Problem gesehen werden muss (siehe Abschnitt 1.2)), wurde die Mehr-Ziel-Anforderung beschrieben (siehe Abschnitt 1.2.6)). Im weiteren wurde beschrieben, wie hierarchische Strukturen bei einer Mehr-Ziel-Optimierung erzeugt werden können, mit denen eine Selektion von Alternativen durchgeführt werden kann, die jeweils durch mehrere Attribute definiert werden (siehe Abschnitt 2.4)), was z.B. bei einer Modell-Selektion notwendig wird, die mehrere Qualitätsmaße verwendet. In diesem Kontext wurde das Konzept des Paretokriteriums und der Paretomenge vorgestellt (siehe Abschnitt 2.4.1)). Weiterhin wurden Verfahren zur Erzeugung von Hierarchien mit Hilfe des Paretokriteriums vorgestellt (siehe Abschnitt 2.4.2)). Hervorzuheben ist dabei die sukzessive Deaktivierung von Paretomengen (siehe Abschnitt 2.4.2.2)), sowie die Anwendung der Paretodominanz auf eine Wettkampf-Hierarchie (siehe Abschnitt 2.4.2.3)). Es folgten Darstellungen, bei denen gezeigt wurde, wie bei einer Mehr-Ziel-Optimierung Abbruchkriterien formuliert werden können.

Adaptive Systeme erfordern ein Optimierungsverfahren, mit dessen Hilfe sie ihre eigenen Systemparameter an externe Anforderungen anpassen können. Als Beispiel sehr robuster Parameteroptimierungsverfahren wurden im weiteren grundlegende Modelle der Evolutions-Strategien für eine Ein-Ziel-Optimierung (siehe Abschnitt 2.5.1)) und eine Mehr-Ziel-Optimierung (siehe Abschnitt 2.5.2)) vorgesellt.

Im Kontext der Darstellung von IR als einem komplexen Problem (siehe Abschnitt 1.2)), wurden Unsicherheit und Vagheit beschrieben (siehe Abschnitt 1.2.4)). Eine Möglichkeit mit der Unsicherheiten quantifiziert werden können sind Intervalle, die mit Intervallarithmetiken verarbeitet werden können. Werden Objekte durch intervallbasierte Attribute beschrieben, so müssen spezielle Selektions-Operatoren eingeführt werden, mit denen ein Vergleich und somit ein Ranking durchgeführt werden kann (siehe Abschnitt 2.6)). Eine Klasse von Möglichkeiten eine Dominanz zwischen zwei Intervallen festzustellen, ist der Vergleich eines ausgewählten Punktes im Intervall wie z.B. der Intervallmittelpunkt (siehe Abschnitt 2.6.1)). Eine andere Möglichkeit ist die Verwendung eines Teils des Methodenrepertoires der Fuzzylogik, indem Zugehörigkeitsfunktionen verwendet werden (siehe Abschnitt 2.6.2)), oder indem intervallbasierte Dominanzfunktionen und Distanzmetriken definiert werden (siehe Abschnitt 2.6.3)).

Im Kapitel 3) wurde das Konzept der Polyrepräsentation in Kontext des Information Retrievals, das im Abschnitt 1.6) eingeführt wurde, formal ausdifferenziert. Ausgehend von der allgemeinen Betrachtung von Dokumenten, Merkmalen und Queries als Zeichenketten, wurden die Auswirkungen auf die Daten-

strukturen bei der Indexierung gezeigt, wenn eine einkomponentige Polyrepräsentation betrachtet wird (siehe Abschnitt 3.4)). Weiterhin wurde gezeigt, welche Retrievalstrategien möglich und notwendig sind, um polyrepräsentierte Queries (siehe Abschnitt 3.6.1)), Queryvektoren (siehe Abschnitt 3.6.2)), Dokumentvektoren (siehe Abschnitt 3.6.4)), Retrievalregionen (siehe Abschnitt 3.6.5)) und polyrepräsentierte Ergebnismengen zu verarbeiten.

Eingeführt wurde das Konzept positiver und negativer Queries und Queryvektoren, wobei negative Queries als eine Form von Gegenbeispiel zu interpretieren sind, wodurch komplexe Retrievalstrategien im Vektorraummodell möglich werden, die Analogien zu einem boolschen Retrieval mit der NOT-Funktion besitzen (siehe Abschnitt 3.6.6)).

Die Clusterung von Informationsobjekten (siehe Abschnitt 3.7)), insbesondere von Dokumentvektoren, wurde im Hinblick auf die Darstellungen in den beiden nachfolgenden Kapiteln detailliert beschrieben, wobei die Clusterung mit Hilfe von unüberwachten Verfahren aus der Klasse der Self-Organizing Maps (SOMs, siehe Abschnitte 2.1.4) bis 2.1.8)) formal dargelegt wurde (siehe Abschnitt 3.8)). Diese Entwicklungslinie knüpft somit direkt an die Arbeit von Bachelier (1995[14]) an, bei der die Indexierung mit einem neuen SOM-Typ formal beschrieben wurde. Neben der Clusterung von Dokumentvektoren wurde die Clusterung von Merkmalen durch eine GNG-SOM beschrieben, sowie die Query-Modifikation mit Hilfe dieses Merkmalsgraphen, wobei der Fall einer Queryvektor-Monorepräsentation (siehe Abschnitt 3.8.5)) und einer Queryvektor-Polyrepräsentation (siehe Abschnitt 3.8.6)) dargestellt wurde.

Nach der Darstellung von Retrievalstrategien, die genau einen Interaktionsakt zwischen Agent und IRS umfassen, wurden interaktive Retrievalstrategien in Form des Relevanz-Feedbacks beschrieben (siehe Abschnitt 3.9)). Grundlage war die Darstellung der Relevanzproblematik im IR ((siehe Abschnitt 3.9.1))), bei der gezeigt werden konnte, dass die Relevanzbewertungen des Agenten während des Interaktionsprozesses mit dem IRS den Status von Schätzungen besitzen, da der Agent zu diesen Zeitpunkten sein zugrunde liegendes Problem noch nicht gelöst hat, und somit nicht angeben kann, welchen Beitrag bestimmte Inhalte (Dokumente) zu der Lösung beitragen werden.

Im Kontext des Relevanz-Feedbacks konnte gezeigt werden, dass eine erweiterte Sichtweise möglich ist, bei der nicht nur eine Modifikation der Query- und Dokumentvektoren durch (binäre) Relevanzurteile des Agenten durchgeführt werden kann, sondern bei der die Relevanzurteile allgemeiner genutzt werden können, um andere Datenstrukturen wie Gewichtsvektoren (siehe Abschnitt 3.9.4)), Retrievalregionen (siehe Abschnitt 3.9.5)) oder die Indexierungsfunktion (siehe Abschnitt 3.9.6)) zu modifizieren, um dem Ideal eines adaptiven Informationssystems näher zu kommen.

Im Rahmen der Darstellung der Adaption von Query- und Dokumentvektoren wurde formal gezeigt, wie alternative Adaptionsverfahren, deren Ursprung die Adaption von Gewichtsvektoren in GNG-SOMs sind, eingesetzt werden können, um Queryvektoren (siehe Abschnitt 3.9.2.2)) und Dokumentvektoren (siehe Abschnitt 3.9.3.2)) unter Verwendung der binären Relevanzwerte des Agenten zu adaptieren. Der konzeptuelle Schwerpunkt bildete jeweils die Darstellung des Queryvektor- und des Dokumentvektor-Feedbacks im Kontext von Dokumentvektoren-GNG-SOMs (siehe Abschnitte 3.9.2.7) und 3.9.3.3)).

Weiterhin wurde ein neues Queryvektorsplitting-Verfahren beschrieben, das sich in natürlicher Weise aus der Verwendung GNG-SOMs und der SOM-orientierten Queryvektor-Adaption ergibt (siehe Abschnitt 3.9.2.5)). Die bewerteten relevanten und nicht relevanten Dokumentvektoren werden dabei als Stimuli eines unüberwachten Lernprozesses genutzt, um mit Adaptionsoperationen eine wachsende Queryvektor-GNG-SOM aufzubauen. Nach dem Abbruch des Aufbaus kann jeder Gewichtsvektor dieser GNG-SOM als eigener Queryvektor interpretiert und in der nächsten Feedback-Iteration weiter verwendet werden.

Auf der Basis des Konzeptes der Reformulierungsrelevanz, die im Kontext der Diskussion zur Relevanzproblematik beschrieben wurde, wurden mit dem Reformulierungs-Relevanz-Feedback neue Vorgehensweisen eingeführt (siehe Abschnitt 3.9.7)). Diese basieren zum Teil darauf, dass der Agent aufgefordert wird, seine Query zu reformulieren, nachdem ihm Dokumente auf der Basis seiner vorangegangenen Query präsentiert wurden. Durch die Kenntnis der sich ergebenden zwei Queryvektoren lassen sich Reformulierungs-Relevanzwerte berechnen, die ihrerseits für das gesamte Spektrum der Relevanz-Feedbackverfahren einsetzbar sind. Diese Reformulierungs-Relevanzwerte werden als reelle Relevanzwerte angenommen, wodurch eine Brücke zu Kapitel 4) geschlagen wird, indem reelle Relevanzwerte generell verwendet werden.

Nach der Begründung des Nutzens von Relevanz-Approximationsmodellen im Gegensatz zu Rankingverfahren mit einer monoton fallenden Distanzfunktion (siehe Abschnitt 4.1.2)), wurde zunächst der Fall unklassifizierter Dokumentvektoren betrachtet. In diesem Kontext wurden Verfahren zur Queryvektor-Adaption bei reellen Relevanzbewertungen bei Mono- und Polyrepräsentation der Queryvektoren formal beschrieben (siehe Abschnitt 4.2.1)). Im Mittelpunkt standen hierbei wiederum Adaptionsverfahren, die aus der SOM-Adaption abgeleitet wurden.

Es folgte die Beschreibung des grundlegenden Verfahrens zum Einsatz eines instanzbasierten Relevanz-Approximationsmodells, bei dem die Relevanzwerte aus der Initialisierungsiteration für den Modellaufbau genutzt werden, wobei zunächst der Fall einer unklassifizierten Dokumentvektorenmenge betrachtet wurde (siehe Abschnitt 4.2)). Mit diesem instanzbasierten Modell werden in der nachfolgenden Iteration alle oder eine Teilmenge der noch nicht durch den Agenten bewerteten Dokumentvektoren eine Relevanzschätzung zugeordnet, wobei diese Schätzungen ein Ranking der Dokumentvektoren und ein korrespondierendes Ranking der Dokumente aufbauen. Der Queryvektor der Initialisierungsiteration kann im einfachsten Fall für alle weiteren Iterationen unberücksichtigt werden (siehe Abschnitt 4.2.2)), oder er kann durch zusätzliche Queryvektor-Adaptionen verändert werden (siehe Abschnitt 4.2.3)). Es wurden daraufhin Vorschläge zur Effizienzverbesserung des grundlegenden Verfahrens gemacht (siehe Abschnitt 4.2.4)), wobei Verfahren zur Begrenzung der Anzahl der Dokumentvektoren, denen eine Relevanzschätzung zugeordnet wird, im Vordergrund stehen (siehe Abschnitt 4.2.4.2)).

Der nächste Entwicklungsschritt bestand in der Erzeugung einer GNG-SOM der Dokumentvektoren, die durch das Relevanz-Approximationsmodell selektiert wurden. Mit dieser GNG-SOM wurden neue Retrievalstrategien formuliert, die ebenfalls das Ziel haben, die Anzahl der zu schätzenden Dokumentvektoren lokal zu begrenzen (siehe Abschnitt 4.2.5)). Der Aufbau eines GNG-SOMs der Ergebnis-Dokumentvektorenmenge bezeichnet den natürlichen Übergang zu einem prototypbasierten Relevanz-

Approximationsmodell, indem den einzelnen Neuronen einer solchen GNG-SOM ein Stützpunkt im Relevanzraum durch ein instanzbasiertes Relevanz-Approximationsmodell zugeordnet wird, da den Gewichtsvektoren selbst kein richtiger Relevanzwert zuordenbar ist (siehe Abschnitt 4.2.6)).

Nach der Einführung von GNG-SOMs auf der Basis von Ergebnis-Dokumentvektorenmengen, die somit jede Iteration on-scratch neu gebildet werden müssen, wurde die konsequente Weiterentwicklung beschrieben, bei der eine Klassifikation der gesamten Dokumentvektorenmenge durch eine GNG-SOM verwendet wurde (siehe Abschnitt 4.3)). Zunächst wurde nur die Voronoi-Zerlegung des Dokumentvektorenraumes verwendet, um Retrievalstrategien zu beschreiben, wobei die durch den Agenten bewerteten Dokumentvektoren direkt als Stützpunkte eines instanzbasierten Modells verwendet werden (siehe Abschnitt 4.3.1)). Im nächsten Schritt konnte gezeigt werden, wie der Gewichtsvektoren-Graph der Gesamt-GNG-SOM in ein prototypbasiertes Relevanz-Approximationsmodell erweitert werden kann, indem den Neuronen ein zusätzlicher Stützpunkt im Relevanzraum zugeordnet wurde, der durch ein instanzbasiertes Modell erzeugt wurde (siehe Abschnitt 4.3.2)). Die Gesamt-GNG-SOM wurde bis zu diesem Punkt ohne Nachadaption verwendet, sodass der nächste Schritt die Darstellung von Nachadaptions-Operationen ist (siehe Abschnitt 4.3.3)), d.h. Gewichtsvektoren der Gesamt-GNG-SOM werden durch Präsentation der bewerteten Dokumentvektoren adaptiert. Hierbei ist die Nachadaption ohne Wachstumsoperationen (siehe Abschnitt 4.3.3.1)) bzw. mit Wachstumsoperationen durchführbar (siehe Abschnitt 4.3.3.2)), was zu einer stärker werdenden Individualisierung der GNG-SOM und des darauf aufbauenden Approximationsmodells führt.

Bislang wurde ausschließlich eine Approximationsmodell-Monorepräsentation betrachtet, sodass im Abschnitt 4.4) die Darstellungen bezüglich der Modell-Polyrepräsentation folgen. Hierbei wurde zunächst die Polyrepräsentation instanzbasierter Modelle betrachtet (siehe Abschnitt 4.4.1)), wobei gezeigt wurde, wie diese durch eine Queryvektor-Polyrepräsentation (siehe Abschnitt 4.4.1.1)) bzw. durch Bootstrap-Verfahren erzeugt werden können (siehe Abschnitt 4.4.1.2)). Der Schwerpunkt der Darstellungen zur Modell-Polyrepräsentation bildet jedoch die Polyrepräsentation prototypbasierter Modelle (siehe Abschnitt 4.4.2)), wobei zwischen unklassifizierten Dokumentvektoren (siehe Abschnitt 4.4.2.1)) und dem Vorliegen einer Gesamt-GNG-SOM der Dokumentvektoren (siehe Abschnitt 4.4.2.2)) unterschieden wird. Liegt eine Dokumentvektor-GNG-SOM vor, so können aus der Gesamt-GNG-SOM individualisierte GNG-SOMs gebildet werden, die als Modell-Polyrepräsentation verwendet werden können. Es wurde gezeigt, wie dies durch Adaption der Stützpunkte im Relevanzraum und Erhaltung im Dokumentvektorenraum (siehe Abschnitt 4.4.2.2.1)) bzw. durch Adaption der Stützpunkte im Dokumentvektoren- und im Relevanzraum durchgeführt werden kann (siehe Abschnitt 4.4.2.2.2)). Besonders hervorzuheben sind die Speicher- und Berechnungs-Effizienzvorteile im ersten Fall, da die Modelle im hochdimensionalen Dokumentvektorraum die gleichen Stützpunkte besitzen, und sich nur in den Stützpunkten im eindimensionalen Relevanzraum unterscheiden. Im letzten Schritt wurde wiederum der Fall betrachtet, dass eine Adaption mit Wachstumsoperationen verwendet wird, um individuelle Modelle zu erzeugen (siehe Abschnitt 4.4.2.2.3)).

Der nächste größere Abschnitt beschäftigte sich mit der Nutzung von Ergebnissen vergangener Interaktionen anderer Agenten oder des gleichen Agenten, wobei zwischen diesen beiden Fällen nicht unterschieden werden soll (siehe Abschnitt 4.5)). Nach der formalen Definition von Interaktionsobjekten,

welche u.a. Queryvektor, nachgewiesene und bewertete Dokumentvektoren, und eventuell eine individualisierte GNG-SOM enthält, wurde gezeigt, wie man diese Objekte aus der Vergangenheit des IRS, bezogen auf die momentane Situation, selektieren kann (siehe Abschnitt 4.5.1)). Im weiteren wurde dargestellt, wie man die unterschiedlichen Datenstrukturen selektierter Interaktionsobjekte nutzen kann. Konkret wurde die Nutzung von Stimulusmengen (siehe Abschnitt 4.5.2)), nachadaptierten Queryvektoren (siehe Abschnitt 4.5.3)), Relevanz-Approximationsmodellen (siehe Abschnitt 4.5.4)) und Suchregionen (siehe Abschnitt 4.5.5)) vergangener Interaktionen untersucht.

Bei allen bis zu diesem Punkt dargestellten Verfahren der Nutzung von Relevanz-Approximationsmodellen wurde nicht berücksichtigt, dass die Relevanzschätzung selbst einem Fehler unterliegt, d.h. es wurde nicht die Modellqualität berücksichtigt. Im Abschnitt 4.6) wurde gezeigt, wie Relevanzschätzungen um Fehlerschätzungen korrigiert werden können, indem explizite Fehlermodelle aufgebaut wurden. Dabei wurden nicht nur Relevanz-Bias-Modelle, sondern auch Relevanz-Varianz-Modelle betrachtet, d.h. es wurde ein Fehlermodell und ein Unsicherheitsmodell formal beschrieben. Diese zwei Typen von Modellen sollen die gleiche Struktur wie die Relevanz-Approximationsmodelle besitzen, d.h. es werden instanzbasierte Fehlermodelle (siehe Abschnitt 4.6.1)), sowie prototypbasierte Fehlermodelle (siehe Abschnitt 4.6.2)) betrachtet. Dabei wurde jeweils zwischen Mono- und Polyrepräsentation der Fehlermodelle unterschieden.

Im letzten Abschnitt von Kapitel 4) wurde aufgezeigt, welche Metastrategien möglich sind, um die Zielsetzung der Relevanzmaximierung beim Retrieval und des Aufbaus von Relevanz-Approximationsmodellen zu integrieren (siehe Abschnitt 4.7)). Eine Heuristik den Modellaufbau zu verbessern, ist die zusätzliche Auswahl von Median-Dokumentvektoren aus der lokalen Dokumentvektorenmenge von Nachbar-Neuronen des Gewinner-Neurons einer GNG-SOM bezüglich eines Queryvektors in der Initialisierungs-Iteration. Solche Heuristiken sind begründet, da den Gewichtsvektoren der Nachbar-Neurone eine bessere Relevanzschätzung zugeordnet werden können, da die Distanz zu einem Stützpunkt mit einem Referenzwert durch den Agenten kleiner wird, doch diese Heuristiken sind nicht in dem Maß begründbar wie bestimmte Formen des aktiven Lernens, bei denen im Kontext des IR ebenfalls die Zielsetzung der Integration von Relevanz- und Modell-Maximierung vorgegeben ist.

Im Kapitel 5) wurden Methoden zum aktiven Lernen in IRS formal beschrieben, die im Abschnitt 1.8) skizziert wurden. Zunächst wurde die Unterscheidung zwischen passivem und aktivem Lernen vertieft (siehe Abschnitt 5.1.1)), wobei drei grundlegende Szenarien des aktiven Lernens mit geschlossener Stimulusmenge (siehe Abschnitt 5.1.2)), mit einem Stimulusstrom (siehe Abschnitt 5.1.3)) und mit einer offenen Stimulusmenge (siehe Abschnitt 5.1.4)) beschrieben wurden.

Die wesentliche Unterscheidung zwischen Verfahren des aktiven Lernens, die in dieser Arbeit behandelt wurde, bezieht sich jedoch auf die Unterscheidung zwischen indirekten und direkten Verfahren (siehe Abschnitt 5.2)). Indirekte Verfahren nutzen Eigenschaften von Kandidatenstimuli als Selektionskriteren, wobei das Kriterium der fehlenden Übereinstimmung der Outputschätzung in einer Menge von Approximationsmodellen eine Modell-Polyrepräsentation erfordert, was durch die Modellklasse Query-by-Committee umschrieben wird (siehe Abschnitt 5.2.1.2)). Es wurde gezeigt, dass eine Möglichkeit der Operationalisierung dieses Selektionskriteriums mit Hilfe der Outputvarianz-Maximierung möglich ist

siehe Abschnitt 5.2.1.3). Im Kontext des IR wurde diese Sichtweise erweitert, indem eine Bias-Maximierung hinzu genommen wurde, wobei eine Darstellung im Rahmen klassifizierter Dokumentvektoren mit Hilfe einer GesamtDokumentvektoren-GNG-SOM gewählt wurde (siehe Abschnitt 5.2.1.4)).

Direkte Verfahren nutzen Eigenschaften von Modellen mit erweiterten Stimulusmengen als Selektionskriterien, wobei als Modellqualität das Bias-Quadrat-Integral (siehe Abschnitt 5.2.2.1)), das Output-Varianz-Integral (siehe Abschnitt 5.2.2.2)), bzw. eine Kombination aus Bias-Quadrat- und Output-Varianz-Integral (siehe Abschnitt 5.2.2.3)) verwendet wurde. Es wurde dargestellt, dass direkte Verfahren Effektivitätsvorteile gegenüber indirekten Verfahren besitzen, da indirekte Verfahren von einer nicht Korrelation von Kandidateneigenschaften und Modelleigenschaften ausgehen. Direkte Verfahren besitzen demgegenüber einen erheblichen Effizienznachteil, sodass Verfahren vorgeschlagen wurden, die zu einer Effizienzverbesserung führen können (siehe Abschnitt 5.2.3)). Wird davon ausgegangen, dass Eigenschafts-Integrale zur Modellbewertung weiter verwendet werden sollen, so besitzen die Verfahren zur Effizienzverbesserung das Ziel, weniger Stützpunkte zu verwenden (siehe Abschnitte 5.2.3.1) bis 5.2.3.3)), da der Aufwand der Integration proportional zu der Anzahl der Stützpunkte ist. Ein alternativer Ansatz ist die Verringerung der Anzahl der Integrale, indem nur ausgewählten Prototyp-Kandidaten ein Integral durch Integration zugeordnet wird, und den restlichen Kandidaten wird eine Schätzung ihres Integrals zugeordnet (siehe Abschnitt 5.2.3.4)). Von besonderer Bedeutung ist das Verfahren der Nutzung von Neuronen als Stützpunkt der Approximation und der Integration (siehe Abschnitt 5.2.3.6)), das zu der Klasse der Verfahren mit verringerten Stützpunkten gehört, da auf diese elegante Weise ein lokales Qualitätsmaß eines stützpunktbasierten Approximationsmodells erzeugt wird, anstatt ein globales Maß, wenn die Integrationsstützpunkte im gesamten Inputraum liegen können. Prinzipiell könnten andere Verfahren zur Effizienzverbesserung angegeben werden, die nicht auf der Operationalisierung der Modellqualität durch ein Eigenschafts-Integral basieren, doch darauf wurde zugunsten einer Überlegung verzichtet, die Effektivitätsverbesserungen beinhaltet. Da ein Integral eine Mittelwertsbildung ist (zentrales Moment erster Ordnung), die lokale Schwankungen nivelliert, wurde dargelegt, dass die zusätzliche Verwendung von zentralen Momenten höherer Ordnung die Differenzierungsfähigkeit von Modellen erhöhen kann (siehe Abschnitt 5.2.4)), was seinerseits im Potential mündet, dass weniger Stützpunkte benötigt werden.

Im weiteren wurde die Integration des Relevanz- und Modell-Maximierungskriteriums beim IR im Kontext des aktiven Lernens dargestellt (siehe Abschnitt 5.3)), wobei allgemein eine sekundäre Dokumentvektorenliste für das Relevanz-Maximierungskriterium und eine sekundäre Liste für das Modell-Maximierungskriterium erstellt wurde. Es wurden drei Fälle unterschieden, die sich in der Anzahl der tertiären Ergebnislisten unterscheiden. Wird eine gemeinsame tertiäre Ergebnisliste gefordert (siehe Abschnitt 5.3.1)), so müssen Verfahren definiert werden, welche die beiden sekundären Listen integrieren, was einer Combining-Strategie entspricht. Können dem Agenten zwei Listen präsentiert werden, so entfällt die Notwendigkeit einer Combining-Strategie, wobei jedoch die Notwendigkeit besteht, doppelt auftretende Dokumentvektoren zu korrigieren (siehe Abschnitt 5.3.2)). Diese Elemente der Schnittmenge beider Listen besitzen das Potential eine eigene Liste zu bilden, sodass drei tertiäre Listen erzeugt würden (siehe Abschnitt 5.3.3)).

Als letzter Punkt des Kapitels 5) wurde ein Lösungsansatz für das Kombinatorikproblem bei direkten Verfahren des aktiven Lernens vorgeschlagen (siehe Abschnitt 5.4)), das sich ergibt, wenn mehrere Stimuli aus einer Stimulus-Kandidatenmenge übernommen werden sollen, da im Extremfall eine Auswahl aus der Potenzmenge der Kandidatenmenge getroffen werden müsste. Der Lösungsvorschlag setzt voraus, dass zwei Ordnungskriterien verwendet werden, wie das Output- und das Modell-Maximierungskriterium allgemein bzw. das Relevanz- und das Modell-Maximierungskriterium beim IR. Die beiden Kriterien müssen sequentialisiert werden, wobei das Output- bzw. Relevanz-Maximierungskriterium als erstes angewendet wird (siehe Abschnitt 5.4.1)), d.h. die Kandidatenmenge wird mit Hilfe des ersten Kriteriums in eine Liste vorstrukturiert (siehe Abschnitt 5.4.2)). Aus dieser Liste werden zusammenhängende Teilleisten als Kandidatenmengen ausgewählt (siehe Abschnitt 5.4.3)), wobei die Modelle bewertet werden, die sich ergeben, wenn eine solche Kandidatenmenge die vorliegende Stimulusmenge erweitert (siehe Abschnitt 5.4.4)).

Verzeichnis ausgewählter Symbole

Kapitel 1)

$\Theta = (d_k \mid k = 1, ..., \#\Theta \in N)$	Endliches Zeichenalphabet.
$D_i = (d_{ij} \mid \forall d_{ij} \in \Theta)$	Dokument als Zeichensequenz.
$D(\Theta \mid n)$, $D(\Theta \mid \leq n)$	Menge aller Dokumente mit einer Sequenzlänge von n. bzw. von kleiner-gleich n.
$D^t = (D_i^t \mid i = 1, ..., m^t)$	Geordnete Dokumentliste zum Zeitpunkt t.
DM^t	Dokumentmenge zum Zeitpunkt t.
$P^{DM(t)}$	Potenzmenge der Dokumentmenge.
$DVR \subseteq R^n$ bzw. $DVR \subseteq [0, 1]^n$.	n-dimensionaler Dokumentvektorenraum.
$x_i^t = (x_{ij}^t \in R$ bzw. $\in [0, 1] \mid j = 1, ..., n^t)$	Dokumentvektor von Dokument D_i^t mit Gewichtungen.
$DV^t = (x_i^t \in DVR \mid i = 1, ..., m^t)$	Geordnete Dokumentvektorenliste zum Zeitpunkt t.
DVM^t	Dokumentvektorenmenge zum Zeitpunkt t.
$P^{DVM(t)}$	Potenzmenge der Dokumentvektorenmenge.
$d_{DVR}(., .)$	Metrik im Dokumentvektorenraum DVR.
Γ_{DVR}	Menge aller Metriken im Dokumentvektorenraum DVR.
$A_{IR(D)}: D(\Theta) \rightarrow DVR: D_i \mid\rightarrow x_i^t$.	Indexierungsfunktion der Dokumente.
$F^t = (F_i^t \mid i = 1, ..., n^t)$	Geordnete Term-, Deskriptor-, Merkmals- oder allgemein Featureliste zum Zeitpunkt t.
$FVR \subseteq R^m$ bzw. $FVR \subseteq [0, 1]^n$	m-dimensionaler Merkmalsvektorenraum.
$f_i^t = (f_{ij}^t \in R$ bzw. $\in [0, 1] \mid j = 1, ..., m^t)$	Merkmalsvektor von F_i^t in Abhängigkeit von D^t.
$FV^t = (f_i^t \in FVR \mid i = 1, ..., n^t)$	Geordnete Merkmalsvektorenliste zum Zeitpunkt t.
$d_{FVR}(., .)$	Metrik im Merkmalsvektorenraum FVR.
Γ_{FVR}	Menge aller Metriken im Merkmalsvektorenraum FVR.
$Q_i^t = (d_{ij} \mid \forall d_{ij} \in \Theta)$	Query als Zeichensequenz.
$Q(\Theta)$	Menge aller Queries mit unspezifizierter Sequenzlänge.
$q_i^t = (q_{ij}^t \mid j = 1, ..., n_Q{}^t)$	Queryvektor von Query Q_i^t mit Gewichtungen.
$QR \subseteq R^{n(Q)}$	n(Q)-dimensionaler Queryvektorenraum.
$A_{IR(Q)}: Q(\Theta) \rightarrow QR: Q_i \mid\rightarrow q_i^t$.	Indexierungsfunktion der Queries (Monorepräsentation).
$ret(q_i^t, DMM^t, d_{DVR})$	Retrievalfunktion (Monorepräsentation).
$DV(q_i^t) \subset DV^t$	Dokumentvektoren-Retrieval-Liste bezügl. q_i^t.
$D(q_i^t) \subset D^t$	Dokument-Retrieval-Liste bezügl. q_i^t.
$U(q_i^t \mid \varepsilon)$	ε-Umgebung um Queryvektor q_i^t.
$DV_n{}^t = DV_{n\&rel}{}^t \cup DV_{n\&\overline{rel}}{}^t$	Liste der nachgewiesenen Dok-Vektoren x^t bezügl. q^t.
$DV_{n\&rel}{}^t$	Liste der nachgewiesenen und relevanten x^t.
$DV_{\bar{n}\&rel}{}^t$	Liste der nicht nachgewiesenen und relevanten x^t.
$DV_{n\&\overline{rel}}{}^t$	Liste der nachgewiesenen und nicht relevanten x^t.
$D_{\bar{n}\&\overline{rel}}{}^t$	Liste der nicht nachgewiesenen und nicht relevanten x^t.

$DV_{\neg n}^t = DV^t \setminus DV_n^t$	Liste der nicht nachgewiesenen x^t bezügl. q^t.
$Recall = \#DV_{n\&rel}^t / \#DV_{rel}^t$	Recall-Definition.
$Precision = \#DV_{n\&rel}^t / \#DV_n^t$	Precision-Definition.

Kapitel 2)

$X \subseteq R^n$	Inputraum.
$Y \subseteq R^m$	Outputraum.
$f(x): X \rightarrow Y: x_j \mid\rightarrow y_j$	Abbildung vom Inputraum in den Outputraum.
$f(x)\wedge: X \rightarrow Y: x_j \mid\rightarrow y_j\wedge$	Approximation der Abbildung f(x).
$M = \{m_j = (x_j, f(x_j)) \mid j = 1, ..., \mu\}$	Gesamtmenge überwachter Stimuli.
$AM(x \mid M)$	instanzbasiertes Approximationsmodell durch überw. M.
$S = ad(M) = \{m_{S,j} \mid j = 1, ..., \mu_S < \mu\}$	Prototypmenge durch Adaptionsfunktion aus M.
$AM(x \mid S)$	prototypbasiertes Approximationsmodell durch überw. M.
$d_X(x_k, x_j)$	Distanzfunktion im Inputraum.
$h(x_k, x_j) = h(d_X(x_k, x_j)) := h_{k,j}$	Distanzabhängige Kernelfunktion.
$\nu_k = \sum_j h(x_k,x_j)$	Normbildung.
$f(x_k)\wedge = 1/\nu_k \sum_j h(x_k,x_j) * f(x_j)$	Approximation von $f(x_k)$ durch Kernel Regression.
$\sigma(f(x_k))^2 = 1/\nu_k \sum_j h_{k,j} * (f(x_k)\wedge - f(x_j^t))^2$	Output-Varianz-Schätzung durch Kernel Regression.
$M = \{m_j = (x_j) \mid j = 1, ..., \mu_S\}$	Gesamtmenge unüberwachter Stimuli.
$N^t = \{n_i^t = (w_i^t, C_i^t, E_i^t) \mid i = 1,..., \mu_N^t\}$	Neuronenstruktur einer GNG-SOM zum Zeitpunkt t.
$w_i^t (= w(x)_i^t) \in X$	Gewichtsvektor bzw. Stützpunkt im Inputraum X.
C_i^t	Verbindungsvektor, der Netzstruktur definiert.
E_i^t	lokaler Fehlerwert der Netzstruktur.
$\tilde{R}_i^t \subseteq X$	Voronoi-Region von Neuron n_i^t.
$N(d_G=1 \mid G^t)_i$	Nachbarschaftsmenge mit Graphenabstand $d_G=1$ im Graph G^t um das Neuron n_i^t.
$N^t = \{n_i^t = (w_i^t, M_i^t, C_i^t, E_i^t) \mid i = 1, ..., \mu_N^t\}$	Neuronenstruktur einer SC-GNG-SOM zum Zeitpunkt t.
$M_i^t = \{m_{ij} = (x_{ij}) \mid j = 1, ..., \mu_{S,i}^t\}$	Menge unüberw. Stimuli, deren Vektor x in $\tilde{R}_i^t$ liegt.
$QF_i^t = 1/\mu_{S,i}^t \sum_j (w_i^t - x_{ij})^2$	lokaler Quantifizierungsfehler der Netzstruktur.
$QF(N^t) = 1/\mu_N^t \sum_i QF_i^t, \forall n_i^t \in N^t$	globaler Quantifizierungsfehler der Netzstruktur.
$M^t = \{m_j = (x_j, f(x_j)) \mid j = 1, ..., \mu_S^t\}$	Gesamtmenge überwachter Stimuli.
$N^t = \{n_i^t = (w(x)_i^t, w(y)_i^t, M_i^t, C_i^t) \mid i = ...\}$	Neuronenstruktur einer überwachten SC-GNG-SOM.
$w(y)_i^t \in Y$	Stützpunkt im Outputraum Y.
$M_i^t = \{m_{ij} = (x_{ij}, f(x_{ij})) \mid j = 1, ..., \mu_i^t\}$	Menge überw. Stimuli, deren Inputvektor in $\tilde{R}_i^t$ liegt.
$f(w(x)_i^t \mid M_i^t) = w(y)_i^t\wedge$	lokale, instanzbasierte Approximation des Outputs.
$AM(x \mid N^t)$	globales, prototypbasiertes AM durch Neuronen-Attribute.

$f(x_{ij} \mid AM(x \mid N^t)) =$ $1/\mu_N{}^t \sum_i h(x_{ij}, w(x)_i{}^t) * f(w(x)_i{}^t \mid M_i{}^t)$	Output-Schätzung eines lokalen Stimulus $x_{ij} \in M_i{}^t$ durch Kernel-Regression mit globalem Modell $AM(x \mid N^t)$.
$MSE(n_i{}^t \mid AM(x \mid N^t)) = 1/\mu_i{}^t *$ $\sum_j (f(x_{ij}{}^t) - f(x_{ij} \mid AM(x \mid N^t)))^2, \forall m_{ij} \in M_i{}^t.$	lokaler Mean-Square-Error (MSE) des Neurons $n_i{}^t$.
$MSE(AM(x \mid N^t)) = 1/\mu_N{}^t *$ $\sum_i MSE(n_i{}^t \mid AM(x \mid N^t)), \forall n_i{}^t \in N^t$	Gesamtfehler des überwachten SC-GNG-SOM-Modells.
$bias(x_i) = f(x_i) - f(x_i \mid AM_k)$	Bias der Outputschätzung bezügl. des Modells AM_k.
$f(x_i \mid B) = 1/\delta * \sum_{k=1->\delta} f(x_i \mid AM_k)$	Bootstrap-Outputschätzung.
$f(x_i \mid AM)$	Outputschätzung bezügl. des Gesamtmodells AM.
$bias(x_i)_{0,632} =$ $(f(x_i \mid B) - f(x_i \mid AM))/0,632$	0,632-Bootstrap-Schätzung.
$n_i = (w_i, net_i, z_i, S(z_i), \Delta w_i)$	Formales Neuron.
$w_i \in R^n$	Gewichtsvektor.
net_i $(net_i(c_{ij}, S(z_{ij}{}^t) \in R)$	Inputfunktion.
z_i (z.B. $z_i = net_i{}^{t+1} - T_i$)	Aktivitätsfunktion.
$S(z_i) = y_i \in R$	Outputfunktion.
Δw_i	Lernregel.
$N_{AA}{}^t = \{n_i{}^t = (w_i, M_i, C_i, net_i{}^t, T_i, z_i{}^t, S(z_i{}^t))$ $\mid cd_X, net, z(net), k, S(z), t_{max}; i = 1, ..., \mu_N\}$	GNG-SOM mit Aktivitätsausbreitung
w_i	Vektor von Gewichtungsfaktoren, die Verbindungskante zu einem benachbarten Neuron repräsentieren.
cd_X	Distanzgewichtungsfunktion.
net	Inputfunktion bei einem externen Vektor.
z(net)	Aktivitätsfunktion bei einem externen Vektor.
S(z)	Outputfunktion bei einem externen Vektor.
k	Parameter bei externer, sigmoider Outputfunktion.
t_{max}	maximale Anzahl der lateralen Iterationsschritte.
$mom(r,c) = 1/\mu_M * \sum_i (x_i - c)^r, r \in N_0.$	allgemeine Moment r'ter Ordnung bezogen auf Vektor c.
$mom(r,\bar{x}), \bar{x} = 1/\mu_M * \sum_i x_i$	zentrale Moment r'ter Ordnung mit $\bar{x}$ als Mittelwert.
$mom(r, M, \bar{c})$, mit $\bar{c} \in \{\bar{f}(x), \bar{f}(x)\hat{}, \Delta\bar{f}^2\}$	Output- und Fehlermomente eines Modells AM(x) bezüglich einer Stimulusliste M.
$\bar{f}(x) = 1/\mu_M * \sum_i f(x_i)$	Outputmittelwert.
$\bar{f}(x)\hat{} = 1/\mu_M * \sum_i f(x_i \mid AM(x))$	Mittelwert der Outputschätzungen.
$\Delta\bar{f}^2 = 1/\mu_M * \sum_i \Delta f_i^2$	Mittelwert der Outputfehler.
$C(n, M, AM(x)_k)$ $= (mom(r, M, \Delta\bar{f}(x_i)^2) \mid r = 1, ..., n)$	Liste der ersten n Fehlermomente eines Modells $AM(x)_k$ bezügl. der Stimulusmenge M.
$dom(r)_{k,p} := dom(c(r)_k, c(r)_p)$	Dreiwertige Dominanzfunktion zweier Momente.

$G(x_j \mid M)$ $= (m_{s(i\|j)} \mid d_X(x_j, x_{s(i\|j)}) < d_X(x_j, x_{s(i+1\|j)}))$	Gewinnerliste der Präsentation von x_j in der Menge M.
$G(x_j \mid M, i_{start}, i_{end})$	Zusammenhängender Ausschnitt von $G(x_j \mid M)$, beginnend mit dem i_{start}'ten und endend mit dem i_{end}'ten Element.
$x_i = (x_{ij} \mid j = 1, ..., n) \in X \subseteq R^n$	n Inputvariablen (decision vector).
$f_k(x), k = 1, ..., m \in Y \subseteq R^m$	m Zielfunktionen (objective vector).
$g_l(x), l = 1, ..., p$	p Constraints.
X_f	anwendbare Teilmenge von X (feasible set).
Y_f	anwendbare Region im Zielraum Y.
$dom(x_i, x_j)$	Dreiwertige Dominanzfunktion zweier Inputvektoren.
PM(A)	Menge der untereinander nicht dominanten Elemente aus
$\{a \in A \mid a$ ist nicht-dominant bezüglich $A\}$	A, die gegenüber A \ PM(A) dominant sind.
$PM(X_f) = X_{PM}$	Pareto-Menge bezüglich X_f.
$PH(A) = (PRM(A)_k \mid k = 1, ...)$	Pareto-Hierarchie als Liste disjunkter Mengen $PRM(A)_k$.
PWH(A)	Pareto-Wettkampf-Hierarchie
$P^t = \{a_i^t = (x_i^t, f(x_i^t), \sigma_i^t) \mid i = 1, ..., \mu\}$	ES-Elternpopulation mit μ Individuen.
$x_i^t = (x_{ij}^t \in R \mid j = 1, ..., n)$	Inputvektor.
$f(x_i^t) \in R$	Fitnessbewertung
$\sigma_i^t = (\sigma_{ij}^t \in R \mid j = 1, ..., n)$	Mutationsschrittweitenvektor.
$sel_R(P^t) = E_{N(k)}^t$	Selektion zur Reproduktion mit Elternmenge $E_{N(k)}^t$.
$rec(\sigma_{E(1,k)}^t, \sigma_{E(2,k)}^t)$	Rekombinationsfunktion zweier Mutationsschrittweiten.
$\sigma_{N(k)}^{t'} = m(\sigma_{N(k)}^t \mid \sigma_{N(k)}^t)$	Selbstmutation des Mutationsschrittweitenvektors.
$x_{N(k)}^t = rec(x_{E(1,k)}^t, x_{E(2,k)}^t)$	Rekombinationsfunktion zweier Inputvektoren.
$x_{N(k)}^{t'} = m(x_{N(k)}^t \mid \sigma_{N(k)}^{t'})$	Mutation des Inputvektors.
$ZP^t = \{a_{N(k)}^t \mid i = 1, ..., \lambda\}$	Zwischenpopulation.
$sel_N(ZP^t) = P^{t+1}$	Selektion zur Übernahme in die Nachfolgepopulation.
I	Menge aller reellen Intervalle.
$I_f(x_j) = [\underline{f}(x_j), \bar{f}(x_j)]$	Output-Intervall bezügl. eines reellen Inputs x.
$I_f(x_i) \leq_+ I_f(x_j) <=> \bar{f}(x_i) \leq \underline{f}(x_j)$	Relation „stark kleiner-gleich".
$I_f(x_i) \leq_{++} I_f(x_j) <=> \underline{f}(x_i) \leq \bar{f}(x_j)$	Relation „schwach kleiner-gleich".
$d_I(I_f(x)_1, I_f(x)_2)$	Distanzmetrik zw. Intervallvektoren $I_f(x)_1, I_f(x)_2 \in I^n$.

Kapitel 3)

$DMM^t = (x_{ij}^t \mid i = 1, ..., m^t; j = 1, ...n^t)$	Dokument-Merkmals-Matrix.
$f_j^t = (f_{ji}^t \mid i = 1, ..., m^t)$	Merkmalsvektor.
$DL_{i,\#D(i)-l+1}^t$ $= (d_{i,\#D(i)-l}^t, d_{i,\#D(i)-l+1}^t, ..., d_{i,\#D(i)}^t)$.	Zusammenhängende Teillisten aus D_i^t mit je l Zeichen.
$BQL_i^t = \{DL_{ip}^t \mid p = 1, ..., \#Q_i^t - l + 1\}$	Bootstrapmenge der Teillisten.

$movB: D(\Theta) \rightarrow D(\Theta): D_i^t \mid\rightarrow D_{ik}^t$	Moving-Blocks-Funktion, erzeugt künstl. Zeichensequenz.
$BD^t = \{D^t, BD_k^t \mid k = 1, ..., \delta\}$	Polyrepräsentation von Dokumentlisten.
$A_{IR(Q),k}: Q(\Theta) \rightarrow DVR: Q_i \mid\rightarrow q_{i,k}$	Polyrepräsentation der Indexierungsfunktion.
$A_{IR(D)}: D(\Theta) \rightarrow DVR:$	Stochastische Indexierungsfunktion.
$D_i^t \mid\rightarrow x_{ik}^t = x_i^t + \sigma_k^t; \sigma_k^t \in R^n$	
$QVM_i^t = \{q_{ik}^t \in DVR \mid j = 1, ..., \delta\}$	Menge der polyrepräsentierten Queryvektoren.
$U(q_{i,k}^t \mid \varepsilon)$	ε-Umgebung des Queryvektors $q_{i,k}^t$.
$DVM_{i,k}^t$	Ergebnis-Dokumentvektorenmenge von $q_{i,k}^t$.
$ret(.): P^{DVM(t)} \times DVR \times \Gamma_{DVR} \times R^+$ $\rightarrow P^{DVM(t)}: (DVM^t, q_{i,k}, d_{DVR}, \varepsilon) \mid\rightarrow DVM_{i,k}^t$	Retrievalfunktion (Polyrepräsentation).
Q_i^{-t}	negative Query (im Sinne eines Gegenbeispiels).
q_i^{-t}	negativer Queryvektor.
$U(q_i^{-t} \mid \varepsilon^-)$	ε-Umgebung eines negativen Queryvektors.
$DVM(q_i^{-t})$	Ergebnis-Dokumentvektorenmenge von q_i^{-t}.
$QVM_i^{-t} = \{q_{ij}^{-t} \in DVR \mid j = 1, ..., m_{iQ^-}^t\}$	Menge der polyrepräsentierten negativen Queryvektoren.
$DV(QVM_i^{+-t})$	Liste der Dokumentvektoren, die in mindestens einer neg. oder einer pos. Umgebung liegen.
$DV(\neg QVM_i^{+t} \wedge QVM_i^{-t})$	Dokumentvektoren, die in mindestens einer negativen Umgebung, jedoch in keiner pos. Umgebung liegen.
$DV(QVM_i^{+t} \wedge \neg QVM_i^{-t})$	Dokumentvektoren, die in mindestens einer positiven Umgebung, jedoch in keiner neg. Umgebung liegen.
$F(D_j)^t$	Menge der Merkmale aus F^t, die in D_j auftreten.
$D(F_i)^t$	Menge der Dokumente aus D^t, in denen F_i enthalten ist.
$h(F_i \mid D_j)$	Absolute Häufigkeit des Merkmals F_i im D_j.
$h(F_i \mid D^t)$	Absolute Häufigkeit desMerkmals F_i in D^t.
$\bar{h}(F_i \mid D^t) = 1/m^t \sum_j h(F_i \mid D_j)$	Mittlere Häufigkeit des Merkmals F_i in D^t.
$h(F^t \mid D_j) = \#D_j$	Absolute Häufigkeit aller Merkmale aus F^t in D_j.
$F_{max\|j}$	Merkmal, das am häufigsten in D_j auftritt.
$h(F_{max\|j} \mid D_j)$	Absolute Häufigkeit des Merkmals $F_{max\|j}$ in D_j.
$F_{max\|D(t)}$	Merkmal, das am häufigsten in D^t auftritt.
$h(F_{max\|D(t)} \mid D^t)$	Absolute Häufigkeit des Merkmals $F_{max\|D(t)}$ in D^t.
$w(F_i \mid D^t) = \log_e[1+ m^t/h(F_i \mid D^t)]$	Logarithmische $w(F_i \mid D^t)$-Gewichtung:
$w(F_i \mid D^t) =$ $\log_e[1 + h(F_{max\|D(t)} \mid D^t)/h(F_i \mid D^t)]$	Normalisierte logarithmische $w(F_i \mid D^t)$-Gewichtung:
$w(F_i \mid D^t) = 1/h(F_i \mid D^t)$	Hyperbolische $w(F_i \mid D^t)$-Gewichtung.
$w(F_i \mid D^t) = \log_e[(m^t - h(F_i \mid D^t))/h(F_i \mid D^t)]$	Normalisierte hyperbolische $w(F_i \mid D^t)$-Gewichtung:
$v(F_i \mid D^t) = \sum_{D(j)} [-h(F_i \mid D_j)/h(F_i \mid D^t) *$ $\log_2(h(F_i \mid D_j)/h(F_i \mid D^t))], \forall D_j \in D(F_i)^t$	Rauschen.

$sig(F_i \mid D^t) = \log_2[h(F_i \mid D^t) - \nu(F_i \mid D^t)]$	Signal.
$w(F_i \mid D^t)_{Entropie} = 1 - \nu(F_i \mid D^t)/\log_2[m^t]$	Signal-Rauschen-Gewichtung 1 (Entropie).
$w(F_i \mid D^t) = sig(F_i \mid D^t)$	Signal-Rauschen-Gewichtung 2.
$w(F_i \mid D^t) = sig(F_i \mid D^t)/\nu(F_i \mid D^t)$	Signal-Rauschen-Gewichtung 3.
$w(F_i \mid D^t) = \nu(F_i \mid D^t)_{max} - \nu(F_i \mid D^t)$, mit $\nu(F_i \mid D^t)_{max} = \max\{\nu(F_i \mid D^t) \mid \forall F_i \in F^t\}$.	Signal-Rauschen-Gewichtung 4.
$DURAEN^t = c * \sum_j s(\bar{x}, x_j)$, mit $\bar{x} = 1/m^t * \sum_j x_j, \forall D_j \in D^t$, c als Parameter.	Dichte der Dokumentmenge D^t im DR.
$DURAEN_{-i}^t$.	Dichte von D^t im $(n^t - 1)$-dim. $DR_{-i} \subseteq R^{n(t)-1}$ (d.h. ohne Berücksichtigung von Merkmal F_i):
$DW_i^t = DURAEN_{-i}^t - DURAEN^t$	Diskriminanzwert:
$w(F_i \mid D_j) = h(F_i \mid D_j) * DW_i^t$	Gewichtungsfaktor mit Diskriminanzwert:
$w(F_i \mid D_j) = V_i^t\, var_i^t/\bar{h}(F_i \mid D^t)$, mit $var_i^t = 1/(m^t - 1) * \sum_j (h(F_i \mid D_j) - \bar{h}(F_i \mid D^t))^2$, und $\bar{h}(F_i \mid D^t) = 1/m^t * \sum_j h(F_i \mid D_j), \forall D_j \in D(F_i)^t$.	Gewichtungsfaktor mit Variationskoeffizient:
$S^t = \{S_j^t = (s_j^t) \mid j = 1, ..., \}$	Grundmenge aller Objekte S_j^t mit Attributsvektor s_j^t.
$SV^t = \{s_j^t \mid j = 1, ..., \}$	Grundmenge aller Objektvektoren s_j^t.
$OK^t = \{SC_i^t \mid i = 1, ..., \#OK^t\}$, mit $SC_i^t = \{S_{ij}^t \mid j = 1, ..., \#SC_i^t\}$	Objekt-Klassifikation.
$OVK^t = \{SVC_i^t \mid i = 1, ..., \#OVK^t\}$, mit $SVC_i^t = \{s_{ij}^t \mid j = 1, ..., \#SC_i^t\}$.	Objektvektoren-Klassifikation.
$\bar{s}_{C(i)}^t = 1/\#SVC_i^t * \sum_j s_{ij}^t, \forall s_{ij}^t \in SVC_i^t$	Arithmetischer Zentroid-Vektor.
$z_{C(i)}^t$: $d(z_{C(i)}^t, \bar{s}_{C(i)}^t) = \min\{d(s_{ij}^t, \bar{s}_i^t) \mid \forall s_{ij}^t \in SVC_i^t\}$	Zentroid-Objektvektor.
$Z_{C(i)}^t \in SC_i^t$	Zentroid-Objekt.
$DVK^t = \{DVC_i^t \mid i = 1, ..., m_C^t\}$, mit $DVC_i^t = \{x_{ij}^t \mid j = 1, ..., m_{C(i)}^t\}$	Dokumentvektoren-Klassifikation.
$DK^t = \{DC_i^t \mid i = 1, ..., m_C^t\}$, mit $DC_i^t = \{D_{ij}^t \mid j = 1, ..., m_{C(i)}^t\}$	Dokument-Klassifikation.
DMM_{DC}^t	Rekodierung der DMM^t zur $(m_C^t \times n^t)$-Matrix DMM_{DC}^t.
FVR_C	neuer Merkm.-Vektorenraum, der zu DMM_{DC}^t korresp.
$cl(DV^t) = DVK^t$ -> $rec(DMM^t) = DMM_{DC}^t$	Dokumentvektoren-Clusterungsfunktion cl(.) und Rekodierungsfunktion rec(.).
$\bar{x}_{C(i)}^t = 1/\#DVC_i^t * \sum_j x_{ij}^t, \forall x_{ij}^t \in DVC_i^t$	Arithmetischer Zentroid-Dokumentvektor von DVC_i^t.
$z_{C(i)}^t$: $d_{DR}(z_{C(i)}^t, \bar{x}_{C(i)}^t) = \min\{d(x_{ij}^t, \bar{x}_i^t) \mid \forall x_{ij}^t \in DVC_i^t\}$	Zentroid-Dokumentvektor.
$Z_{C(i)}^t$	Zentroid-Dokument von DC_i^t.
$FVK^t = \{FVC_i^t \mid i = 1, ..., n_C^t\}$	Merkmalsvektoren-Klassifikation.
$FVC_i^t = \{f_{ij}^t \mid j = 1, ..., n_{C(i)}^t\}$	Merkmalsvektoren-Cluster.
DMM_{FC}^t	Rekodierung der DMM^t zur $(m^t \times n_C^t)$-Matrix DMM_{FC}^t.

DVR_C	neuer Dok.-Vektorenraum, der zu DMM_{FC}^t korrespondiert.
$cl(FV^t) = FVK^t \rightarrow rec(DMM^t) = DMM_{FC}^t$	Merkmalsvektoren-Clusterungsfunktion cl(.) und Rekodierungsfunktion rec(.).
$cl(FV^t) = FVK^t \rightarrow rec(DMM^t) = DMM_{FC}^t \rightarrow cl(DV_C^t) = DVK_C^t \rightarrow rec(DMM_{FC}^t) = DMM_{DFC}^t$	Merkm.-Vektoren-Clusterung, gefolgt von Dok.-Vektoren-Clusterung.
$N_F^t = \{n_{F,i}^t = (w(x_F)_i^t, M_{F,i}^t, C_{F,i}^t) \mid i = 1, ..., \mu_{N,F}^t\}$	GNG-SOM im Merkmalsvektorenraum.
$N_D^t = \{n_{D,i}^t = (w(x_D)_i^t, M_{D,i}^t, C_{D,i}^t) \mid i = 1, ..., \mu_{N,D}^t\}$	GNG-SOM im Dokumentvektorenraum.
QVM_i^{+t}	Menge von positiven Queryvektoren.
QVM_i^{-t}	Menge von negativen Queryvektoren.
$M_{Q+,p}^t$	Menge von positiven Queryvektoren q_i^{+t}, die in der Voronoi-Region $\tilde{R}_p^t$ eines Neurons $n_{D,p}^t$ liegen.
$M_{Q-,p}^t$	Menge von negativen Queryvektoren q_i^{-t}, die in der Voronoi-Region $\tilde{R}_p^t$ eines Neurons $n_{D,p}^t$ liegen.
N_{D+}^t	Menge von Neuronen, in deren $\tilde{R}_p^t$ nur q_i^{+t} liegen.
N_{D-}	Menge von Neuronen, in deren $\tilde{R}_p^t$ nur q_i^{-t} liegen.
N_{D+-}^t	Menge von Neuronen, in deren $\tilde{R}_p^t$ q_i^{+t} und q_i^{-t} liegen.
$N_{D/}^t$	Menge von Neuronen, in deren $\tilde{R}_p^t$ weder q_i^{+t} noch q_i^{-t} liegen.
$DV(QVM_i^{+-t})$	Vereinigung aller lokalen Dok-Vektoren- bzw. Stimulusmengen von Neuronen aus N_{D+-}^t.
G_{Q+-}^t	Graph der q_i^{+t} und q_i^{-t} (Delaunay-Triangulation).
$RF_{IR}: D(\Theta) \rightarrow \{0, 1\}$:	binäre Relevanz-Funktion.
$D_{ij} \mid\rightarrow rel(D_{ij}) \equiv rel(x_{ij}), D_{ij} \in D_i \equiv D(q_i) \equiv D(Q_i)$	
$DVM(q_i^{t=0})_{rel}$	relevante nachgewiesene Dok.-Vektoren bezügl. $q_i^{t=0}$.
$f_{rel}^{t=0}$	Anzahl der Elemente in $DVM(q_i^{t=0})_{rel}$.
$\bar{s}_{DVM(i,rel)}$	arithmetischen Zentroid-Vektoren aus $DVM(q_i^{t=0})_{rel}$.
$DVM(q_i^{t=0})_{\overline{rel}}$	nicht relevante Dok.-Vektoren bezügl. $q_i^{t=0}$.
$f_{\overline{rel}}^{t=0}$	Anzahl der Elemente in $DVM(q_i^{t=0})_{\overline{rel}}$.
$\bar{s}_{DVM(i,\overline{rel})}$	arithmetischen Zentroid-Vektoren aus $DVM(q_i^{t=0})_{\overline{rel}}$.
$q_i^{t+1} = rfb(q_i^t, DVM(q_i^t)_{rel}, DVM(q_i^t)_{\overline{rel}}, \alpha, \beta, \chi)$	Beispiel einer Queryvektor-Relevanzfeedback-Funktion.
$q_i^{t+1} = q_i^t + \Delta_{rel}q_i^t$, $\Delta_{rel}q_i^{t=0} = \varepsilon_{rel} * d_{DVR}(q_i^{t=0}, \bar{s}_{DVM(i,rel)}^{t=0})$	Beispiel einer SOM-basierten Queryvektor-rfb-Fkt mit ausschl. positiver Adaption.
$ad^+(q_i^t)$ bzw. $ad^-(q_i^t)$	positiven bzw. negative Adaption.
$\Delta_{\overline{rel}}q_i^t$	Verschiebevektor bei negativer Adaption.
$DV(QVM_i^{+-t})_{rel}$, $DV(QVM_i^{+-t})_{\overline{rel}}$	relevante bzw. nicht relevante nachgewiesene Dok.-Vektoren bezügl. pos. und negativer Queryvektoren.

$\bar{s}_{Q+-(i,rel)}^{t}$, $\bar{s}_{Q+-(i,\overline{rel})}^{t}$	korrespondierende Mittelwertsvektoren.
$\Delta_{rel}q_{ij}^{+t}$, $\Delta_{rel}q_{ij}^{-t}$	Verschiebevektoren, die pos. bzw. neg. Queryvektoren durch relevante Dok.-Vektoren adaptieren.
$\Delta_{\overline{rel}}q_{ij}^{+t}$, $\Delta_{\overline{rel}}q_{ij}^{-t}$	Verschiebevektoren, die pos. bzw. neg. Queryvektoren durch nicht relevante Dok.-Vektoren adaptieren.
$DV(U(\bar{s}_{DV(i,rel)}^{t} \mid \varepsilon^{+}))$,	Dok.-Vektoren, die in ε-Umgebung vom Fixpunkt $\bar{s}_{DV(i,rel)}^{t}$ der Adaption mit rel. Dok.-Vektoren liegen.
$DV(U(\bar{s}_{DV(i,\overline{rel})}^{t=0} \mid \varepsilon^{-}))$	Dok.-Vektoren, die in ε-Umgebung vom Fixpunkt $\bar{s}_{DV(i,rel)}^{t}$ der Adaption mit nicht rel. Dok.-Vektoren liegen.
$x_{ik+}^{t,neu} = x_{ik+}^{t,alt} + \Delta x_{ik+}^{t,alt}$	SOM-basierte positive Adaption von Dok.-Vektor.
$K(U(q_i \mid \varepsilon^{t-1}) \mid \Delta\varepsilon_t)$	$\Delta\varepsilon_1$-Kragen um $U(q_i \mid \varepsilon^{t-1})$.
FCM	Menge von Fitness-Cases bei Indexierungs-Fkt.-Feedback.
$f(A_{IR,j}^{t})$	Fitnessmaß einer Indexierungs-Fkt. $A_{IR,j}^{t}$.

Kapitel 4)

$RF_{IR}: D(\Theta) \rightarrow [0, 1]$:	reelle Relevanz-Funktion.
$D_{ij} \mid\rightarrow rel(D_{ij}) \equiv rel(x_{ij})$, $D_{ij} \in D_i \equiv D(q_i) \equiv D(Q_i)$	
$rel(x)^{\wedge}: DVR \rightarrow RVR := [0, 1]$:	Schätzung durch Relevanz-Approximationsmodell.
$x \mid\rightarrow rel(x)^{\wedge} = rel(x \mid AM(rel(x)))$, $\forall x \in DVR$.	
N_{DV}	unüberwachte, nicht individuelle SC-GNG-SOM
$M_{DV(m)}^{t}$	Menge bewerteter Dok.-Vektoren aus Iteration t.
$M_{DV(m)}^{\leq t}$	Menge bewerteter Dok.-Vektoren aus allen Iterationen bis einschl. t.
$AM(rel(x) \mid M_{DV(m)}^{t})$	zu $M_{DV(m)}^{t}$ korresp. Instanz-Approximationsmodell.
$M_{DV(m),i}^{t}$ bzw. $M_{DV(m),i}^{\leq t}$,	Menge bewerteter Dok.-Vektoren in Voronoi-Region $\tilde{R}_i^{t}$.
$N_{DV,bew}^{t}$	Neuronenmenge aus N_{DV}, für die $M_{DV(m),i}^{\leq t-1}$ nicht leer ist.
$AM(rel(x) \mid N_{D,bew}^{t})$	zu $N_{D,bew}^{t}$ korresp. Prototyp-Approximationsmodell.
N_{DV}^{t}	individuelle SC-GNG-SOM, mit Relevanzschätzungen für die Neurone, in deren Voronoi-Region mindestens ein bewerteter Dok.-Vektor liegt.
$M_{DV(m),i}^{t}$	Menge bewerteter Dok.-Vektoren in Voronoi-Region $\tilde{R}_i^{t}$.
$\overline{rel}_{D,k}^{t}$	Relevanzmittelwert eines Gewichtsvektors $w_{D,k}^{t}$.
$var_{rel,D,k}^{t}$	Relevanzvarianz.
$KI_{rel,D,k}$	Relevanz-Konfidenzintervall.
$rel(w(x_{D,bew})_i^{t} \mid AM(rel(x) \mid M_{DV(m)}^{t}))$	Rel-Schätzung an Gewichtsvektor durch Instanz-Modell.
$rel(x_j^{t+1} \mid AM(rel(x))^{t})$	Rel-Schätzung an unbewertetem Dok.-Vektor durch Modell aus vorangegangener Iteration.

$N_{D,p,bew}^{t}$	individuelle SC-GNG-SOM bei einer Polyrepräsentation von Prototyp-Approximationsmodellen.
$IRel_k^t = [rel(w(x_D)_k^t)_{min}\hat{}, rel(w(x_D)_k^t)_{max}\hat{}]$	Relevanz-Intervall mit kleinster und größter Relevanzschätzung mit einer Menge von Approximationsmodellen.
$H_d(RelM_k^t)$: $I \rightarrow N$: $IRel(1)_k^t \mid\rightarrow s(RelM(1)_k^t)$	diskrete Relevanz-Häufigkeitsfunktion.
$H_s(rel(w(x_D)_k)\hat{}) : R^+ \rightarrow R^+$: $rel(w(x_D)_k)\hat{} \mid\rightarrow s(rel(w(x_D)_k)\hat{})$	stetige Relevanz-Häufigkeitsfunktion.
ΔN_{DV}^{t}	Neuronenmenge, deren Gewichtsvektor und/oder Relevanzschätzung in der Iteration t verändert wurden.
$N_{DV}^{t} = N_{DV} \oplus \Delta N_{DV}^{t}$.	GNG-SOM-Individualisierung, indem im nicht indiv. Netz die veränderten Neurone ersetzt werden.
$I^{\leq T-1} = (Interaktion(\tau) \mid \tau = 1, ..., T-1)$	geordnete Interaktionsliste eines IRS.
$Interaktion(\tau) = (Q_\tau, q_\tau, M_{DV(m),\tau}, N_{DV,\tau})$	Struktur einer Interaktion mit
Q_τ, q_τ,	Query, Queryvektor bei Iterationsende,
$M_{DV(m),\tau}$	Menge aller Dok.-Vektoren mit Relevanz-Bewertung,
$N_{DV,\tau}$	indiv. GNG-SOM-Approx.-modell nach Iterationsende.
$IL_T^{\leq T-1} = (Interaktion(p) \in I^{\leq T-1} \mid d_{DVR}(q_T^{t=0}, q_p) < d_{DVR}(q_T^{t=0}, q_{p+1}))$	Interaktionsliste nach steigender Distanz zu dem Initialis.-Queryvektor der momentanen Interaktion T.
$Interaktion(\tau) = (..., q_\tau, n_{DV,s(1\mid\tau)}, ...)$	erweiterte Interaktionsstruktur bei globaler GNG-SOM, indem Neuron angegeben wird, in dessen Voronoi-Region der Queryvektor bei Iterationsende liegt.
$m_j = (x_j, rel(x_j), rel(x_j)\hat{}, bias(x_j), \sigma(rel(x_j))^2)$	Stimulusstruktur bei fehlerkorrigierten Instanz-Relevanz-Approximationsmodellen.
$n_{DV,k}^{t-1} = (w(x_{DV})_k^{t-1}, rel(w_k^{t-1})\hat{},$ $rel(w_k^{t-1})_{boot}\hat{}, rel(w_k^{t-1})\hat{}', bias(w_k^{t-1})_{0,632}\hat{},$ $\sigma(rel(w_k^{t-1}))^2, C_k^{t-1}, M_{DV,k}^{t-1}, M_{DV(m),k}^{t-1})$	Neuronenstruktur bei fehlerkorrigierten Prototyp-Relevanz-Approximationsmodellen.

Kapitel 5)

DVM_{prim}^{t}	Primäre Ergebnismenge.
$DV_{sec\mid bias}^{t}$	sekundäre Ergebnisliste auf der Basis der Biasschätzungen.
$DV_{sec\mid\delta}^{t}$	sek. Ergebnisliste auf der Basis der Varianzschätzungen.
$DV_{sec\mid g}^{t}$	sek. Ergebnisliste auf der Basis der aggregierten Bias- und Varianzschätzungen.
$SM^t = \{m_{S,i}^t \mid i = 1, ..., \mu_S\}$	Stützpunktmenge bei Monte-Carlo-Integration.
$KL^t = (m_{K,j}^t \mid j = 1, ..., \mu_K)$	Kandidatenliste.

$IB[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$	geschätztes Bias-Integral des Modells AM(x), wenn ein Kandidat $m_{K,j}^t$ neu in Stimulusmenge aufgenommen wird.
$bias(x_{S,i}^t)_{K,j}^2$	Bias-Quadrat-Schätzung an der Stelle eines Stützpunktes bei einer aktualisierten Stimulusmenge.
$SM_{K,j}^t$	individualisierte Stützpunktmenge, wenn Kandidat $m_{K,j}^t$ neu in Stimulusmenge aufgenommen wird.
$IV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$	geschätztes Varianz-Integral des Modells AM(x), wenn ein Kandidat $m_{K,j}^t$ neu in Stimulusmenge aufgenommen wird.
$\sigma(x_{S,i}^t)_{K,j}^2$	Varianz-Schätzung an der Stelle eines Stützpunktes bei einer aktualisierten Stimulusmenge.
$SM_{V\|K,j}^t$	Stützpunktmenge bei ausschl. Varianzbetrachtung.
$SM_{B\|K,j}^t$	Stützpunktmenge bei ausschl. Biasbetrachtung.
$SM_{BV\|K,j}^t$	Stützpunktmenge bei Bias- und Varianzbetrachtung.
$IBV[AM(x \mid M^t \cup \{m_{K,j}^t\})]^\wedge$	geschätztes Bias- und Varianz-Integral von AM(x), wenn Kandidat $m_{K,j}^t$ neu in Stimulusmenge aufgenommen wird.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Aalbersberg, IJsbrand Jan: Incremental relevance feedback. In: Belkin et al. (1992[37]), SIGIR 1992, S. 11 - 22.{ 37} (http://www.acm.org/pubs/articles/proceedings/ir/133160/p11-aalbersberg/p11-aalbersberg.pdf).

[2] Abell, M.L.; Braselton, P.: Statistics with Mathematica. 1999.{ 23, 34}

[3] Adams, E.; Kulisch, U. (Eds.): Scientific Computing with automatic Result Verification. Academic Press, Boston, 1993.{ 34, 228, 464}

[4] Allan, James: Relevance feedback with too much data. In: Fox et al. (1995[118]), SIGIR 1995, S. 337 - 343.{ 37} (http://www.acm.org/pubs/articles/proceedings/ir/215206/p337-allan/p337-allan.pdf).

[5] Allan, James: Incremental relevance feedback for information filtering. In: Frei et al. (1996[121]), SIGIR 1996, S. 270 - 278. { 33, 37, 143, 193} (http://www.acm.org/pubs/articles/proceedings/ir/243199/p270-allan/p270-allan.pdf).

[6] Amari, S.; Murara, N.; Müller, K.-R.; Finke, M.; Yang, H.: Asymptotic Statistical Theory of Overtraining and Cross-Validation. Dep. of Mathematical Engineering, University of Tokyo, Technical Report METR 95-06, September 1996. { 108, 428} (ftp://archive.cis.ohio-state.edu/pub/neuroprose/amari.overtraining.ps.Z).

[7] Amari, S.; Murara, N.; Müller, K.-R.; Finke, M.; Yang, H.: Statistical Theory of Overtraining - Is Cross-Validation Effective? In: Touretzky et al. (1996[336]), NIPS 8, S. 176-182. { 108, 428} (http://www.first.gmd.de/persons/Mueller.Klaus-Robert/nips_crossval.ps.Z).

[8] Anderson, John R.: The Adaptive Character of Thought. Hillsdale, Erlbaum, 1990. { 23, 26, 28, 38, 39, 42, 143, 167, 232}

[9] Anderson, John R.: The Place of Cognitive Architectures in a Rational Analysis. In VanLehn (1991[345]), S. 1- 24. { 23, 26, 28, 36, 38, 39, 42, 232}

[10] Anderson, John R.: Rules of the mind. Hillsdale, Erlbaum, 1993. { 23, 26, 28, 38, 39, 42, 143, 167, 232}

[11] Ando, Rie Kubota: Latent semantic-space: iterative scaling improves precision of inter-document similarity measurement. In: Belkin et al. (2000[40]), SIGIR 2000, S. 216 - 223. { 16, 25}

[12] Angeline, P. J.; Pollack, J. B.: Competitive Environments envolve better solutions for complex tasks. In: Forrest (1993[117]), S. 264 - 270. { 118, 119, 120}

[13] Audehm, D.: Systematische Ideenfindung. Expert Verlag, 1995. { 165}

[14] Bachelier, Günter: PSS-Kohdex: Konzepte zur integrierten Indexierung und Visualisierung von Textdaten auf der Basis Pseudo-Stetiger-Sensorischer-Kohonen-Karten. Magisterarbeit, Fachrichtung Informationswissenschaft, Saarbrücken, 1995 (Neuerscheinung, Marburg, ISBN 3-8288-8221-4, 1998). { 16, 17, 18, 25, 29, 33, 37, 48, 85, 152, 212, 213, 241, 502}

[15] Bachelier, Günter: Einführung in Selbstorganisierende Karten. Marburg, ISBN 3-8288-5017, 1998a. { 16, 33, 44, 48, 52, 59, 68, 71, 73, 77, 182, 241, 279}

[16] Bachelier, Günter: Einführung in Evolutionäre Algorithmen. Marburg, ISBN 3-8288-5019, 1998b. { 35, 36, 96, 108, 116, 127, 128, 131, 132, 182, 287, 334, 414, 451}

[17] Bachelier, Günter: Cluster-Verfahren im Rahmen der Selbstorganisierenden Karten. Marburg, ISBN 3-8288-5025, 1998c. { 59, 68, 76, 77, 340, 361, 363, 366, 423, 440}

[18] Bachelier, Günter: Eigenschaftserhaltungen in Selbstorganisierenden Karten. Marburg, ISBN 3-8288-5026, 1998d. { 68, 78, 95}

[19] Bachelier, Günter: Pseudo-Stetige-Karten. Marburg, ISBN 3-8288-5038, 1999a. { 212}

[20] Bachelier, Günter: Populationsfortschritt in Evolutions-Algorithmen. Marburg, ISBN 3-8288-5042, 1999b. { 130, 282, 294}

[21] Bachelier, Günter: Anwendungsregionen stützpunktbasierter SOM-Approximationsmodelle. Marburg, ISBN 3-8288-5045, 1999c. { 59, 74, 108, 112}

[22] Bachelier, Günter: Fitness-Approximationsmodelle in Evolutions-Strategien. Marburg, ISBN-3-8288-5055-3, 1999d. { 39, 59, 91, 96, 128, 130, 188, 197, 198, 204, 236, 246, 355, 359, 412, 422, 437}

[23] Bäck, Thomas; Schwefel, Hans-Paul: An overview of evolutionary algorithms for parameter optimization. In: Evolutionary Computation, 1 (1), S. 1 - 23, 1993. { 129, 287} (ftp://alife.santafe.edu/pub/USER-AREA/EC/ES/papers/sys93.ps.gz).

[24] Bäck, Thomas (ed.): Proceedings of the Seventh International Conference on Genetic Algorithms, ICGA-97. Morgan Kaufmann Publishers. 1997. { 542}

[25] Baker-Kearfott, R.; Kreinovich, V. (eds.): Applications of Interval Computations. Kluwer, 1996. { 34, 528}

[26] Banzhaf, W.; Nordin, P.; Keller, R.E.; Francone, F.D.: Genetic Programming. Heidelberg, 1998. { 37, 67, 169, 236, 292, 309}

[27] Bartell, Brian T.; Cottrell, Garrison W.; Belew, Richard K.: Latent semantic indexing is an optimal special case of multidimensional scaling. In: Belkin et al. (1992[37]), SIGIR 1992, S. 161 - 167 . { 16, 25}
(http://www.acm.org/pubs/articles/proceedings/ir/133160/p161-bartell/p161-bartell.pdf).

[28] Bartell, Brian T.: Optimizing Ranking Functions: A Connectionist Approach to Adaptive Information Retrieval. Ph.D. thesis, Department of Computer Science and Engineering, University of California, San Diego, 1994. { 22, 31, 38}
(http://www.bartell.com/bbartell/thesis_1sp.ps).

[29] Batori, Istvan S.; Lenders, Winfried; Putschke, Wolfgang (Hersg.): Computer Linguistik: Ein internationales Handbuch zur computergestützten Sprachforschung und ihrer Anwendungen. Berlin, 1989. { 26}

[30] Bauch, H.; Jahn, K.-U.; Oelschlägel, D.; Süsse, H; Wiebigke, V.: Intervallmathematik. Teubner, Leipzig, 1987. { 34, 136, 137, 140, 142, 228, 334, 464}

[31] Baum, Eric B.: Neural net algorithms that learn in polynomial time from examples and queries. In: IEEE Transactions on Neural Networks 2(1), 1991, S. 5 - 19. { 33}

[32] Baum, Eric B.; Lang, K. J.: Constructing hidden units using examples and queries. In: Lippmann et al. (1991[201]), NIPS 3, 1991, S. 904 - 910. { 33}
(http://nips.djvuzone.org/djvu/nips03/0904.djvu).

[33] Baumgarten, Christoph: A probabilistic solution to the selection and fusion problem in distributed information retrieval. In: Hearst et al. (1999[160]), SIGIR 1999, S. 246 - 253. { 31, 38}
(http://www.acm.org/pubs/articles/proceedings/ir/312624/p246-baumgarten/p246-baumgarten.pdf).

[34] Belew, Richard K.; Booker, L. B. (eds.): Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA 4). San Mateo, CA, Morgan Kaufmann, 1991. { 541}

[35] Belew, Richard K.; Mitchell, Melanie: Adaptive Individuals in Evolving Populations: Models and Algorithms. Santa Fe Institute Studies in the Science of Complexity, Proceedings Volume XXVI, Addison-Wesley, 1996. { 37}

[36] Belkin, Nicholas J.; Croft, W.B.: Information Filtering and Information Retrieval: Two Sides of the Same Coin? In: Communications of the ACM 35 (1992) 12 , S. 29-38. { 33, 143, 145, 193, 389}
(http://www.acm.org/pubs/articles/journals/cacm/1992-35-12/p29-belkin/p29-belkin.pdf).

[37] Belkin, Nicholas; Ingwersen, Peter; Pejtersen, Annelise Mark (eds.): SIGIR 1992, Proceedings of the 15th Annual International Conference on Research and Development in Information Retrieval. New York, NY, 1992. { 525, 527, 531, 537, 538, 544} (http://www.acm.org/pubs/contents/proceedings/ir/133160/index.html).

[38] Belkin, Nicholas J.; Kantor, P.; Fox, E.A.; Shaw, J.A.: Combining evidence of multiple query representations for information retrieval. In: Information Processing and Management, 31 (3): S. 431 - 448, 1995. { 31}

[39] Belkin, Nicholas J.; Narasimhalu, D.; Willett, P. (eds.): SIGIR 1997, Proceedings of the 20'th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Philadelphia, 1997. { 530, 539, 541, 542} (http://www.acm.org/pubs/contents/proceedings/ir/258525/index.html).

[40] Belkin, N.J.; Ingwersen, P.; Leong, M.-K.: SIGIR 2000, Proceedings of the 23'th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Athen, 2000. { 525, 539}

[41] Bellman, R.E.: Dynamische Programmierung und selbstanpassende Regelprozesse. München, 1967 (Aus dem Engl. übers. von Fred Behringer, Original: Adaptive control Processes. Princeton University Press. 1961). { 16, 25}

[42] Bellmann, K. (ed.): Molecular Genetic Information Systems: Modelling and Simulation. Akademie-Verlag, Berlin, 1983. { 529}

[43] Berleant, D.: Automatically verified arithmetic on probability distributions and intervals. In: Baker-Kearfott & Kreinovich (1996[25]), S. 227 - 244. { 34}

[44] Bernier, C. L.: Abstracts and abstracting. In: Dym (1985[102]), S. 423 - 444. { 29, 149}

[45] Beyer, Uwe; Smieja, Frank: Data Exploration with Reflective Adaptive Models. GMD, St. Augustin, 1994. { 42, 57} (ftp://borneo.gmd.de/pub/as/janus/ref94_1.ps).

[46] Binh, T.; Korn, U.: An evolution strategy for the multiobjective optimization. In: Proceedings of the 2nd International Conference on Genetic Algorithms (MENDEL 1996), Brno, Czech Republic, S. 23 - 28. { 35, 112}

[47] Binh, T.; Korn, U.: MOBES: A Multiobjective Evolution Strategy for Constrained Optimization Problems. In: Proceedings of the 3rd International Conference on Genetic Algorithms (MENDEL 1997), Brno, Czech Republic, 1997, S. 176 - 182. { 35, 112}

[48] Biron, Paul V.; Kraft, Donald H.: New methods for relevance feedback: improving information retrieval performance. In: Proceedings of the 1995 ACM symposium on Applied computing, 1995, S. 482 - 487. { 37}
(http://www.acm.org/pubs/articles/proceedings/sac/315891/p482-biron/p482-biron.pdf).

[49] Blair, D.C.: Language and Representation in Information Retrieval. Amsterdam, 1990. { 36}

[50] Blazewicz, Jacek; Ecker, Klaus H.; Pesch, Erwin; Schmidt, Günter; Weglarz, Jan: Scheduling Computer and Manufacturing Processes. Springer, Berlin, 1996. { 23}

[51] Bookstein, Abraham; Chiaramella, Yves; Salton, Gerard; Raghavan, Vijay: SIGIR 1991, Proceedings of the 14'th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Chicago, Illinois, 1991. { 531, 540}
(http://www.acm.org/pubs/contents/proceedings/ir/122860/index.html).

[52] Borko, H.; Bernier, C.L.: Abstracting Concepts and Methods. New York, 1975. { 29, 149}

[53] Born, J.: Evolutionsstrategien zur numerischen Lösung von Adaptionsaufgaben. Dissertation, Humboldt-Universität, Berlin, 1978. { 131, 132}

[54] Born, J.; Bellmann, K.: Numerical Adaption of Parameters in Simulation Models using Evolution Strategies. In: Bellmann (1983[42]), S. 291 - 320. { 131, 132}

[55] Born, J.; Voigt, H.-M.; Santibanez-Koref, I.: Alternative Evolution Strategies to Global Optimization. In: Männer & Manderick (1992[209]), S. 187 - 195. { 131, 132}

[56] Borodin, A.; Kerr, L.; Lewis, F.: Query Splitting in Relevance Feedback Systems. In: Salton (1971[293]), S. 394 - 402. { 37, 43, 250, 251}

[57] Brauen, T.L: Dokument Vector Modifikation. In: Salton (1971[293]), S. 456 - 485. { 37, 43, 268}

[58] Brause, Rüdiger: Neuronale Netze; Teubner-Verlag, Stuttgart, 1991. { 33, 44, 48}

[59] Brodley, C.E.: Recursive Automatic Algorithm Selection for Inductive Learning. Ph.D. Dissertation, Dept. of Computer Science, University of Massachusetts, Amherst, MA, 1994. { 45, 51, 56}
(ftp://ftp.cs.umass.edu/pub/techrept/techreport/1994/UM-CS-1994-061.ps).

[60] Buchholz, Peter: Die strukturierte Analyse Markovscher Modelle. Berlin, 1991. { 26, 169}

[61] Buchholz, Peter; Dunkel, Jürgen; Müller-Clostermann, Bruno; Szczittnick, Michael and Zäske, Siegmar: Quantitative Systemanalyse mit Markovschen Ketten. Stuttgart, 1994. { 26, 169}

[62] Buckley, Chris; Salton, Gerard; Allan, James: The effect of adding relevance information in a relevance feedback environment. In: Croft & Rijsbergen (1994[82]), SIGIR 1994, S. 292 - 300. {37}
(http://www.acm.org/pubs/articles/proceedings/ir/188490/p292-buckley/p292-buckley.pdf).

[63] Buckley, Chris; Salton, Gerard: Optimization of relevance feedback weights. In: Fox et al. (1995[118]), SIGIR 1995 S. 351 - 357. {19, 37}
(http://www.acm.org/pubs/articles/proceedings/ir/215206/p351-buckley/p351-buckley.pdf).

[64] Bundy, Alan (Ed.): Artificial Intelligence Techniques. 4. Aufl., Berlin 1997. { 26}

[65] Chen, Hsinchun; Ramsey, Marshall; Smith, Terry R.: Visual SOM (abstract). In: Belkin et al. (1997[39]), SIGIR 1997, S. 336. { 33, 37}

[66] Chiaramella, Yves: SIGIR 1988, Proceedings of the 11'th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Grenoble, 1988. { 536}
(http://www.acm.org/pubs/contents/proceedings/ir/62437/index.html).

[67] Chiaramella, Y.; Chevalett, J.P.: About retrieval models and logic. In: Computer Journal, 35 (3), 1992, S. 233 - 242. { 27}

[68] Cleveland, W.; Devlin, S.; Grosse, E.: Regression by local fitting. In: Journal of Econometrics 37 (1988): S. 87 - 114. { 69, 455, 466}

[69] Cobb, H.G.: Grefenstette, J.J.: Genetic Algorithms for Tracking Changing Environments. In: Forrest (1993[117]), ICGA5, 1993, S. 523 - 530. { 18}

[70] Cohn, David A.; Atlas, L.: Ladner, R.: Training connectionist networks with queries and selective sampling. In: Touretzky (1990[335]), NIPS 2, S. 566 - 573. { 33, 435}
(http://nips.djvuzone.org/djvu/nips02/0566.djvu).

[71] Cohn, David A.: Queries and exploration using optimal experiment design. In: Cowan et al. (1994[80]), NIPS 6, S. 679 - 686. { 33, 42, 50, 58, 59, 318}
(http://nips.djvuzone.org/djvu/nips06/0679.djvu).

[72] Cohn, David A.; Atlas, Les; Ladner, Richard: Improving generalization with active learning. In: Machine Learning 15(2), 1994, S. 201 - 221. { 42, 50, 58, 59}

[73] Cohn, David A.: Minimizing Statistical Bias with Queries. MIT AI Memo Nr. 1552, 1995. { 33, 42, 50, 58, 59, 61, 97, 98, 99, 235, 308, 411, 436, 441, 453, 454, 455, 458, 466, 470, 476, 499}
(http://www.ai.mit.edu/people/cohn/psyche/bias.ps.Z).

[74] Cohn, David A.; Ghahramani, Zoubin; Jordan, Michael I.: Active Learning with Statistical Models. In: Tesauro et al. (1995[331]), NIPS 7, S. 705 - 712. { 69, 70, 97, 98, 99, 101, 235, 308, 318, 411, 413, 416, 418, 436, 441, 453, 454, 459, 460, 466, 470, 474, 476, 479, 500} (http://nips.djvuzone.org/djvu/nips07/0705.djvu).

[75] Cohn, David A.: Persönliche Mitteilungen. 2000. { 3, 500}

[76] Collins, A.M.; Loftus, E.F.: A spreading-activation theory of semantic processing. In: Psychological Review 82, 1975, S. 407 - 428. { 28, 84}

[77] Cool, Colleen; Belkin, Nicholas J.; Koenemann, Jürgen: On the potential utility of negative relevance feedback in interactive information retrieval. In: Frei et al. (1996[121]), SIGIR 1996, S. 341. { 19}

[78] Cooper, William S.: Some inconsistencies and misnomers in probabilistic information retrieval. In: Bookstein et al. (1991[51]), SIGIR 1991, S. 57 - 61. { 24} (http://www.acm.org/pubs/articles/proceedings/ir/122860/p57-cooper/p57-cooper.pdf).

[79] Cooper, William S.; Gey, Fredric C.; Dabney, Daniel P.: Probabilistic retrieval based on staged logistic regression. In: Belkin et al. (1992[37]), SIGIR 1992, S. 198 - 210. { 25, 55} (http://www.acm.org/pubs/articles/proceedings/ir/133160/p198-cooper/p198-cooper.pdf).

[80] Cowan, Jack D.; Tesauro, Gerald; Alspector, Joshua (eds.): Advances in Neural Information Processing Systems 6, NIPS 6. San Mateo, CA, 1994. { 530, 535}

[81] Crestani, F.; Lalmas, M.; Rjisbergen, C.J. van: Information Retrieval: Uncertainty and Logik. Kluwer, 1998. { 23}

[82] Croft, Bruce W.; Rijsbergen, C.J. van: SIGIR 1994, Proceedings of the 17'th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Dublin, 1994. { 530, 536, 539, 541, 547} (http://www.acm.org/pubs/contents/proceedings/ir/188490/index.html).

[83] Croft, Bruce W.; Moffat, Alistair; Rijsbergen, C.J. van; Wilkinson, Ross; Zobel, Justin: SIGIR 1998, Proceedings of the 21'th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Melbourne, 1998. { 535, 542, 545, 548} (http://www.acm.org/pubs/contents/proceedings/ir/290941/index.html).

[84] Crouch, D.: A Clustering Algorithm for Large and Dynamic Document Collections. Ph.D. thesis, Southern Methodist Univ., Dallas, Texas, 1972. { 16, 17, 161, 162}

[85] Cunningham, Sally Jo; Holmes, Geoffrey; Littin, Jamie; Beale, Russell; Witten, Ian H.: Applying connectionist models to information retrieval. Department of Computer Science, The University of Waikato Hamilton, New Zealand, 1997a. { 25, 34}
(http://www.cs.waikato.ac.nz/~ml/publications/1997/SJC-GH-JL-RB-IHW97.pdf).

[86] Cunningham, Sally Jo; Littin, James; Witten, Ian H.: Applications of Machine Learning in Information Retrieval. Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1997b. { 25, 30}
(http://www.cs.waikato.ac.nz/~ml/publications/1997/SJC-Littin-Witten97.pdf).

[87] Damiani, E.; Fugini, M. G.: Automatic thesaurus construction supporting fuzzy retrieval of reusable components. In: Proceedings of the 1995 ACM symposium on Applied computing, 1995, S. 542 - 547.{ 30}
(http://www.acm.org/pubs/articles/proceedings/sac/315891/p542-damiani/p542-damiani.pdf).

[88] Daniels, P.J.: Cognitive models in information retrieval - an evaluative overview. Technical Reprort, Dept. Information Science, City University, London, 1986. { 29}

[89] Dasgupta, D.; McGregor, D.R.: Nonstationary function optimization using structured genetic algorithm. In: Männer & Manderic (1992[209]), PPSN II, S. 145 - 154. { 18}

[90] Dasgupta, Pallab; Chakrabarti, P.P.; DeSarkar, S.C.: Multiobjective Heuristic Search. Wiesbaden, 1999. { 21, 34}

[91] Davis, Ernest: Constraint Propagation with Interval Labels. In: Artificial Intelligence, 32, 1987, S. 281 - 331. { 34}

[92] Davis, M.H.A.: Markov models and optimization. London, 1993. { 26, 169}

[93] Deb, Kalyanmoy: Multi-objective genetic algorithms: Problem difficulties and construction of test functions. TR CI-49/98, Dep. of Computer Science / XI, Universität Dortmund, 1998. { 35, 115}
(http://www.iitk.ac.in/kangal/papers/mobj.testfn.ps.gz).

[94] Diamantaras, K.I.; Kung, S.Y.: Principal Component Neural Networks. New York, 1996. { 16}

[95] Diamond, Ted: Information retrieval using dynamic evidence combination. Unveröffentlichter Ph. D. Vorschlag, 1996 (nach Vogt (1999[352])). { 31}

[96] Dorffner, Georg: Konnektionismus. Stuttgart, 1991. { 49}

[97] Dozier, Gerry: Staty-State Evolutionary Path Planning, Adaptive Replacement, and Hyper-Diversity. In: Schoenauer et al. (2000[300]), PPSN VI, S. 561 - 570. { 18}

[98] Drucker, H.; Cortes, C.; Jackel, L.D.; LeCun, Y.; Vapnik, V.: Boosting and other ensemble methods. In: Neural Computation 6 (6), 1994, S. 1289–1301. { 45, 56}

[99] Drucker, H.; Cortes, C.: Boosting Decision Trees. In: Touretzky et al. (1996[336]), NIPS 8, S. 479 – 485. { 45, 51, 56}
(http://nips.djvuzone.org/djvu/nips08/0479.djvu).

[100] Dumais, S. T.; Furnas, G. W.; Landauer, T. K.; Deerwester, S.; Harshman, R.: Using latent semantic analysis to improve access to textual information. In: Conference proceedings on Human factors in computing systems, 1988, S. 281 - 285. { 16, 25}
(http://www.acm.org/pubs/articles/proceedings/chi/57167/p281-dumais/p281-dumais.pdf).

[101] Dunlop, Mark D.: The effect of accessing nonmatching documents on relevance feedback. ACM Trans. Inf. Syst. 15 (2), 1997, S. 137 - 153. { 19, 37}
(http://www.acm.org/pubs/articles/journals/tois/1997-15-2/p137-dunlop/p137-dunlop.pdf).

[102] Dym, Eleanor D. (ed.): Subject and Information Analysis. 1985. { 528, 542}

[103] Eckes, T.; Rossbach, H.: Clusteranalysen. Stuttgart, 1990. { 25}

[104] Edelman, S.: Representation, Similarity, and the Chorus of Prototypes. In: Minds and Machines 5 (1995), S. 45 – 68. { 31}

[105] Efron, B.; Tibshirani, R.: An introduction to the bootstrap. Chapman & Hall, New York, 1993. { 46, 47, 59, 83, 91, 92, 137, 149, 358, 443, 486}

[106] Efthimiadis, Efthimis N.: A user-centred evaluation of ranking algorithms for interactive query expansion. In: Korfhage (1993[187]), SIGIR 1993, S. 146 - 159. { 29, 226}
(http://www.acm.org/pubs/articles/proceedings/ir/160688/p146-efthimiadis/p146-efthimiadis.pdf).

[107] Egan, James P.: Signal detection theory and ROC analysis. New York, 1975. { 22}

[108] Eiben, Agoston E.; Bäck, Thomas; Schoenauer, Marc; Schwefel, Hans-Paul: Proceedings of the Fifth Conference on Parallel Problem Solving from Nature, PPSN V, Berlin, 1998. { 543}

[109] Elliott, Robert J.; Aggoun, Lakhdar; Moore, John B.: Hidden Markov models: estimation and control. Berlin, 1995. { 26, 169}

[110] Eshelman, Larry J. (ed.): Proceedings of the Sixth International Conference on Genetic Algorithms, ICGA-95. Morgan Kaufmann Publishers, 1995. { 539}

[111] Facaoaru, C.: Kreativität in Wissenschaft und Technik. Hans Huber Verlag, 1985. { 165}

[112] Fan, Weiguo; Gordon, Michael D.; Pathak, Praveen: Automatic Generation of a Matching Function by Genetic Programming for Effective Information Retrieval. In: Proceedings of the 1999 Americas Conference on Information Systems, August 13-15, 1999, Milwaukee, WI. { 37} (http://www-personal.umich.edu/~wfan/paper/Amcis_final.pdf).

[113] Fedorov, V.: Theory of Optimal Experiments. Academic Press, New York, 1972. { 33, 42, 50, 59, 235, 453}

[114] Fishman, George S.: Monte Carlo: concepts, algorithms and applications. Springer Verlag, New York, 1995. { 16, 454}

[115] Fonseca, Carlos M; Fleming, Peter: Genetic Algorithms for Multiobjective Optimization. In: Forrest (1993[117]), ICGA5, S. 416 - 423. { 35, 115, 116}

[116] Fonseca, Carlos M; Fleming, Peter: An overview of evolutionary algorithms in multiobjective optimization. In: Evolutionary Computation 3 (1), S. 1 - 16, 1995. { 35, 131}

[117] Forrest, Stephanie (ed.): Proceeding of the Fifth International Conference on Genetic Algorithms (ICGA5). San Mateo, CA, Morgan Kaufmann, 1993. { 525, 530, 534}

[118] Fox, Edward A.; Ingwersen, Peter; Fidel, Raya: SIGIR 1995, Proceedings of the 18'th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, New York, NY, 1995. { 525, 530} (http://www.acm.org/pubs/contents/proceedings/ir/215206/index.html).

[119] Francone, F.D.; Nordin, P. Banzaf, W.: Benchmarking the Generalization Capabilities of a Compiling Genetic Programming System using Sparse data Sets. In: Koza (1996[190]), S. 72 - 80. {67}

[120] Fraunhofer-Institut für Systemtechnik und Innovationsforschung (Hersg.): Delphie'98 Umfrage. Karlsruhe, 1998. { 29, 234}

[121] Frei, H.P.; Harman, D.; Schäuble, P.; Wilkinson, R.: SIGIR 1996, Proceedings of the 19'th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Zürich, 1996. { 525, 531, 540, 547} (http://www.acm.org/pubs/contents/proceedings/ir/243199/index.html).

[122] Freund, Yoav: Data filtering and distribution modelling algorithms for machine learning. Ph.D. thesis, Computer and Information Sciences, University of California, Santa Cruz, 1993. { 33, 49} (ftp://ftp.cse.ucsc.edu/pub/tr/ucsc-crl-93-37.ps.Z).

[123] Freund, Yoav; Seung, H.; Shamir, E.; Tishby, N.: Information, prediction, and query by committee. In: Hanson et al. (NIPS 5[156]), 1993, S. 483 - 490. { 33, 58, 435, 436, 442} (http://nips.djvuzone.org/djvu/nips05/0483.djvu).

[124] Freund, Yoav: Boosting a weak learning algorithm by majority. 1995. { 45, 49, 51, 56, 353} (http://www.research.att.com/~yoav/papers/majp.ps.Z).

[125] Freund, Yoav; Schapire, R.E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In: Proceedings of the Second European Conference on Computational Learning Theory. Barcelona, Spain. Springer Verlag, 1995. S. 23–37. { 45, 49, 51, 56, 353} (http://www.research.att.com/~schapire/papers/FreundSc95.ps.Z).

[126] Freund, Yoav; Schapire, R.E.: Experiments with a New Boosting Algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, 1996. { 45, 49, 51, 56, 353} (http://www.research.att.com/~schapire/papers/FreundSc96.ps.Z).

[127] Friedman, S.R.; Maceyak, J.A.; Weiss, S.F.: A Relevance Feedback System based on Dokument Transformations. In: Salton (1971[293]), S. 447 - 455. { 37, 43, 268}

[128] Fritzke, Bernd: Wachsende Zellstrukturen - Ein selbstorganisierendes Neuronales Netzwerkmodell. Arbeitsberichte des Instituts für mathematische Maschinen und Datenverarbeitung der Friedrich Alexander Universität Erlangen-Nürnberg, Dissertation, 1992. { 73, 76} (http://pikas.inf.tu-dresden.de/~fritzke/papers/thesis.ps.gz).

[129] Fritzke, Bernd: Superviced learning with growing cell structures. In: Cowan et al. (1994[80]), S. 255 - 262, 1994. { 76} (http://pikas.inf.tu-dresden.de/~fritzke/papers/fritzke.nips93.ps.gz).

[130] Fritzke, Bernd: A Growing Neural Gas Network Learns topologies. In: Tesauro et al. (1995[331]), NIPS 7, S. 625 - 632. { 73, 76} (ftp://ftp.neuroinformatik.ruhr-uni-bochum.de/pub/manuscripts/articles/fritzke.nips94.ps.gz).

[131] Fritzke, Bernd: Growing Self-organizing Networks - Why? In: Verleysen (1996[349]): S. 61 - 72. { 73} (http://pikas.inf.tu-dresden.de/~fritzke/ftppapers/fritzke.esann96.ps.gz).

[132] Fritzke, Bernd: Vektorbasierte Neuronale Netze. Habilitation, Shaker Verlag, 1998. { 59, 73} (http://pikas.inf.tu-dresden.de/~fritzke/papers/habil.ps.gz).

[133] Fuhr, Norbert; Gövert, Norbert; Rölleke, Thomas: DOLORES: a system for logic-based retrieval of multimedia objects. In: Croft et al. (1998[83]), SIGIR 1998, S. 257 - 265. { 27} (http://www.acm.org/pubs/articles/proceedings/ir/290941/p257-fuhr/p257-fuhr.pdf).

[134] Führing, T.; Jacoby, K.; Michaelis, R.; Panyr, J.: Kontextgestaltgebung: Eine Metapher zur Visualisierung von und Interaktion mit komplexen Wissensbeständen. In: Rauch et al. (1994[272]). { 213}

[135] Fung, R. M.; Crawford, S. L.; Appelbaum, L. A.; Tong, R. M.: An architecture for probabilistic concept-based information retrieval. In: Vidick (1990[350]), SIGIR 1990, S. 455 - 467. { 27} (http://www.acm.org/pubs/articles/proceedings/ir/96749/p455-fung/p455-fung.pdf).

[136] Furnas, G. W.; Deerwester, S.; Dumais, S. T.; Landauer, T. K.; Harshman, R. A.; Streeter, L. A.; Lochbaum, K. E.: Information retrieval using a singular value decomposition model of latent semantic structure. In: Chiaramella (1988[66]), SIGIR 1988, S. 465 - 480. { 16, 25} (http://www.acm.org/pubs/articles/proceedings/ir/62437/p465-furnas/p465-furnas.pdf).

[137] Gabbay, Dov M.; Hogger, C. J.; Robinson, J. A.; Siekmann, J. (eds.): Handbook of Logic in Artificial Intelligence and Logic Programming / Vol. 1: Logical Foundations. Oxford 1993. { 27}

[138] Gabbay, Dov M.; Hogger, C. J.; Robinson, J. A.; Siekmann, J. (eds.): Handbook of Logic in Artificial Intelligence and Logic Programming / Vol. 2: Deduction methodologies. Oxford 1994a. { 27}

[139] Gabbay, Dov M.; Hogger, C. J.; Robinson, J. A.; Siekmann, J. (eds.): Handbook of Logic in Artificial Intelligence and Logic Programming / Vol. 3: Nonmonotonic reasoning and uncertain reasoning. Oxford 1994b. { 27}

[140] Gabbay, Dov M.; Hogger, C. J.; Robinson, J. A.; Siekmann, J. (eds.): Handbook of Logic in Artificial Intelligence and Logic Programming / Vol. 4: Epistemic and temporal reasoning. Oxford 1995. { 27}

[141] Gama, Joao Manuel Portela da: Combining Classication Algorithms. Dissertation, Computer Science, Universität Porto, 1999. { 45, 49, 51, 56, 353} (http://www.ncc.up.pt/~jgama/tese.ps.gz).

[142] Geman, S.; Bienenstock, E. Doursat, R.: Neural Networks and the Bias/Variance Dilemma. In: Neural Computation, 4, S. 1 -58, 1992. { 50, 51, 97, 102, 411, 464}

[143] Gerdes, I.; Klawonn, F.; Kruse, R.: Evolutionäre Algorithmen. Vieweg-Verlag, 2001. { 35}

[144] Gey, Fredric C.: Inferring probability of relevance using the method of logistic regression. In: Croft & Rijsbergen (1994[82]), SIGIR 1994, S. 222 - 231. { 25, 55} (http://www.acm.org/pubs/articles/proceedings/ir/188490/p222-gey/p222-gey.pdf).

[145] Goldberg, D.E.; Smith, R. E.: Nonstationary Function Optimization Using Genetic Dominance and Diploidy. In: Grefenstette (1987[152]), ICGA87, S. 59 - 68. { 18}

[146] Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley, 1989.{ 35, 36, 115, 127}

[147] Goldberg, David E; Nichols, D.; Oki, B. M; Terry, D.: Using Collaborative Filtering to Weave an Information Tapestry. In: Communications of the ACM 35 (1992) 12 , S. 61-70. { 33, 143, 193, 389}
(http://www.acm.org/pubs/articles/journals/cacm/1992-35-12/p61-goldberg/p61-goldberg.pdf).

[148] Goldstein, Jade; Kantrowitz, Mark; Mittal, Vibhu; Carbonell, Jaime: Summarizing Text Dokuments: Sentence Selection and Evaluation Metrics. In: Hearst et al. (1999[160]), SIGIR 1999, S. 121 - 128. { 149}
(http://www.acm.org/pubs/articles/proceedings/ir/312624/p121-goldstein/p121-goldstein.pdf).

[149] Göppert, Josef: Die topologisch interpolierende selbstorganisierende Karte in der Funktionsapproximation. Dissertation, Universität Tübingen, Shaker Verlag, Aachen, 1997. { 54, 65, 66, 108}

[150] Gordon, M.: Probabilistic and genetic algorithms for document retrieval. In: Communications of the ACM 31 (10), 1988, S. 1208 - 1218. { 36}
(http://www.acm.org/pubs/articles/journals/cacm/1988-31-10/p1208-gordon/p1208-gordon.pdf).

[151] Görtz, Günter (Hersg.): Einführung in die künstliche Intelligenz. Bonn, 1993. { 26, 543, 544}

[152] Grefenstette, John J. (ed.): Proceedings of the Second International Conference on Genetic Algorithms and their Applications, ICGA87. Hillsdale, New Jersey, 1987. { 536}

[153] Grefenstette, J.J.: Genetic Algorithms for Changing Environments. In: Männer & Manderic (1992[209]), PPSN II, S. 137 - 144. { 18}

[154] Grossman, D.A.; Frieder, O.: Information Retrieval: Algorithms and Heuristics. Kluwer, 1999. {15, 25, 28}

[155] Hansen, Elden R.: Global Optimization Using Interval Analysis. New York, 1992. { 34}

[156] Hanson, Steve J.; Cowan, Jack D.; Giles, C. Lee (eds.): Advances in Neural Information Processing Systems 5, NIPS 5. San Mateo, Calif., 1993. { 535}

[157] Harman, Donna: Relevance feedback revisited. In: Belkin et al. (1992[37]), SIGIR 1992, S. 1 - 10. { 19, 37, 226, 232}
(http://www.acm.org/pubs/articles/proceedings/ir/133160/p1-harman/p1-harman.pdf).

[158] Harman, Donna (ed.): TREC-4, The 4'th Text Retrieval Conference, Gaithersberg, MD, National Institute of Standards and Technology (NIST), 1996. { 548}
(http://trec.nist.gov/pubs/trec4/t4_proceedings.html).

[159] Harter, P.S.: A Probabilistic Approach to Automatic Keyword Indexing (Part 1, Part 2). In: Journal of the ASIS 26 (1975), S. 197-206 (Teil 1), S. 280-289 (Teil 2). { 23, 48}

[160] Hearst, Marti; Gey, Fredric; Tong, Richard: SIGIR 1999, Proceedings of the 22'th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Berkeley, 1999. { 527, 537, 538, 541, 542, 548}
(http://www.acm.org/pubs/contents/proceedings/ir/312624/).

[161] Henrion, Max; Suermondt, Henri J.; Heckerman David E.: Probabilistic and Bayesian Reprensention of Uncertainty in Information Systems: A Pragmatic Introduction. In: Motro & Smets (1997[228]), S. 255 - 284. { 24}

[162] Hess, Michael: An incrementally extensible document retrieval system based on linguistic and logical principles. In: Belkin et al. (1992[37]), SIGIR 1992, S. 190 - 197. { 27}
(http://www.acm.org/pubs/articles/proceedings/ir/133160/p190-hess/p190-hess.pdf).

[163] Hijorth, J.S. Urban: Computer Intensive Statistical Methods: Validation, Modell Selection and Bootstrap. London, 1994. { 46, 47, 58}

[164] Hinton, Geoffrey; Sejnowski, Terrence J.: Unsuperviced Learning: Foundations of Neural Computation. Cambridge, 1999. { 30}

[165] Hofmann, Thomas: Probabilistic latent semantic indexing. In: Hearst et al. (1999[160]), SIGIR 1999, S. 50 - 57. { 16, 25}
(http://www.acm.org/pubs/articles/proceedings/ir/312624/p50-hofmann/p50-hofmann.pdf).

[166] Höhle, U.; Rodabaugh, St. E.: Mathematics of Fuzzy Sets. Kluwer, 1999. { 30, 53, 139}

[167] Höppner, F.; Klawonn, F.; Kruse, R.: Fuzzy-Clusteranalyse. Vieweg-Verlag, 1997. { 30}

[168] Hordijk, Wim: A Measure of Landscapes. The Santa Fe Institute Technical Report SFI-TR-95-05-045, 1995. { 51}
(http://www.santafe.edu/sfi/publications/Working-Papers/95-05-049.ps).

[169] Horn, Jeffrey: Genetic Algorithms, Problem Diffculty and the Modality of Fitness Landscapes. IlliGAL Report No. 95004, 1995. { 51}
(ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/95004.ps.Z).

[170] Hyvönen, Eero: Constraint reasoning based on intervall arithmetic: the tolerance propagation approach*. In: Artificial Intelligence 58, 1992, S. 71 - 112. { 34}

[171] Ingwersen, Peter: Information Retrieval Interaction. London, 1992. { 33, 36, 46}

[172] Ingwersen, Peter: Polyrepresentation of Information Needs and Semantic Entities. In: Croft & Rijsbergen (1994[82]), SIGIR 1994, S. 101 - 110. { 33, 36, 46}
(http://www.acm.org/pubs/articles/proceedings/ir/188490/p101-ingwersen/p101-ingwersen.pdf).

[173] Iwayama, Makoto: Relevance feedback with a small number of relevance judgements: incremental relevance feedback vs. document clustering. In: Belkin et al. (2000[40]), SIGIR 2000, S. 10 - 16. { 37}

[174] Jacob, Christian: Principia Evolvica Simulierte Evolution mit Mathematica. dpunkt-Verlag, Heidelberg, 1997. { 35, 36, 52, 116, 127, 182}

[175] Jain, A. K.; Murty, M. N.; Flynn, P. J.: Data clustering: a review. In: ACM Comput. Surv. 31, 3 (Sep. 1999), S. 264 - 323. { 25}
(http://www.acm.org/pubs/articles/journals/surveys/1999-31-3/p264-jain/p264-jain.pdf).

[176] Jaquard, A.: The Genetic Structure of Populations. In: Biomathematics, Vol. 5. Springer-Verlag, Berlin, 1974. { 131, 132}

[177] Jones, Terry; Forrest, Stephanie: Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In: Eshelman (1995[110]), ICGA-95. { 51}
(ftp://ftp.cs.unm.edu/pub/forrest/fdc-icga95.pdf).

[178] Jüttner, Gerald; Güntzer, Ulrich: Methoden der Künstlichen Intelligenz für Information Retrieval. München, 1988. { 27}

[179] Kappen, Bert; Gielen, Stan: Neural Networks: Artificial Intelligence and Industrial Applications. Proceedings of the Third Annual SNN Symposium on Neural Networks, Nijmegen, Springer Verlag, New York, 1995. { 542}

[180] Kaszkiel, Marcin; Zobel, Justin: Passage retrieval revisited. In: Belkin et al. (1997[39]), SIGIR 1997, S. 178 - 185. { 43}
(http://www.acm.org/pubs/articles/proceedings/ir/258525/p178-kaszkiel/p178-kaszkiel.pdf).

[181] Kim, Yu-Hwan; Hahn, Shang-Yoon; Zhang, Byoung-Tak: Text filtering by boosting naive bayes classifiers. In: Belkin et al. (2000[40]), SIGIR 2000, S. 168 - 175. { 31, 38}

[182] Klatte, R., Kulisch, U.; Wiethoff, A.; Lawo, C.; Rauch, M.: C-XSC. Berlin, 1993. { 34, 228, 334, 464}

[183] Kohonen, Teuvo: Median Strings. In: Pattern Recognition Letters 3, 1985, S. 309 - 313. { 301}

[184] Kohonen, Teuvo: Self-organization and associative memory. Heidelberg, 3. Aufl., 1989. { 16, 30, 33, 48, 69, 71, 241}

[185] Kohonen, Teuvo: Self-Organizing Maps. Berlin, 1995. { 16, 30, 33, 48, 69, 71, 241}

[186] Kong, E. B.; Dietterich, T. G.: Error-Correcting Output Coding Corrects Bias and Variance. In: Proceedings of the 12th International Conference on Machine Learning, Tahoe City, CA. San Francisco, Morgan Kaufmann, 1995, S. 313 - 321. { 45, 56} (ftp://ftp.cs.orst.edu/pub/tgd/papers/ml95-why.ps.gz).

[187] Korfhage, Robert R.: SIGIR 1993, Proceedings of the 16th Annual International Conference on Reasearch and Development in Information Retrieval. Pittsburgh, PA, USA, 1993. { 533} (http://www.acm.org/pubs/contents/proceedings/ir/160688/index.html).

[188] Koza, J.R.: Genetic Programming I. Cambridge, 1992. { 37, 67, 169, 236, 292, 309}

[189] Koza, J.R.: Genetic Programming II. Cambridge, 1994. { 37, 67, 169, 236, 292, 309}

[190] Koza, J.R. (ed.): International Conference on Genetic Programming 96. Standford, California, MIT Press, 1996. { 534}

[191] Koza, John R.; Bennett, Forrest H. III; Andre, David; Keane, Martin A.: Genetic Programming III. Morgan Kaufmann, San Francisco, CA, 1999. { 37, 67, 169, 236, 292, 309}

[192] Kulisch, U.: Scientific Computing with Automatic Result Verification. London. 1993. { 228, 334}

[193] Kwok, K. L.: Query modification and expansion in a network with adaptive architecture. In: Bookstein et al. (1991[51]), SIGIR 1991, 1991, S. 192 - 201. { 29, 226} (http://www.acm.org/pubs/articles/proceedings/ir/122860/p192-kwok/p192-kwok.pdf).

[194] Kwok, K. L.: A network approach to probabilistic information retrieval. In: ACM Trans. Inf. Syst. 13, 3 (Jul. 1995), S. 324 - 353. { 24, 29, 48, 226} (http://www.acm.org/pubs/articles/journals/tois/1995-13-3/p324-kwok/p324-kwok.pdf).

[195] Lancaster, F.W.: Indexing and Abstracting in Theory and Practice. London, 1991. { 29, 149}

[196] Larkey, Leah S.; Croft, W. Bruce: Combining classifiers in text categorization. In: Frei et al. (1996[121]), SIGIR 1996, S. 289 - 297. { 24} (http://www.acm.org/pubs/articles/proceedings/ir/243199/p289-larkey/p289-larkey.pdf).

[197] Lee, Jonghoon; Dubin, David: Context-sensitive vocabulary mapping with a spreading activation network. In: Hearst et al. (1999[160]), SIGIR 1999, S. 198 - 205. { 28, 84} (http://www.acm.org/pubs/articles/proceedings/ir/312624/p198-lee/p198-lee.pdf).

[198] Lee, Joon Ho: Properties of extended Boolean models in information retrieval. In: Croft & Rijsbergen (1994[82]), SIGIR 1994, S. 182 - 190. { 25} (http://www.acm.org/pubs/articles/proceedings/ir/188490/p182-lee/p182-lee.pdf).

[199] Lee, Joon Ho: Analyses of multiple evidence combination. In: Belkin et al. (1997[39]), S. 267 - 276. { 31, 38} (http://www.acm.org/pubs/articles/proceedings/ir/258525/p267-lee/p267-lee.pdf).

[200] Lee, Te-Won: Independent Component Analysis. Kluwer, 1998. { 16}

[201] Lippmann, Richard P.; Moody, John E.; Touretzky, David S. (eds.): Advances in Neural Information Processing Systems 3, NIPS 3. San Mateo, Calif., 1991. { 527}

[202] Lipsitch, Marc: Adaption on Rugged Landscapes Generated by Iterated Local Interactions of Neighboring Genes. In: Belew & Booker (1991[34]), S. 128 - 135. { 51}

[203] Lundquist, Carol; Grossman, David A.; Frieder, Ophir: Improving relevance feedback in the vector space model. In: Belkin et al. (1997[39]), SIGIR 1997, S. 16 - 23. { 19, 37, 232} (http://www.acm.org/pubs/articles/proceedings/cikm/266714/p16-lundquist/p16-lundquist.pdf).

[204] MacKay, David: Information-based objective functions for active data selection. In: Neural Computation 4 (4), 1992, S. 590 - 604. { 33} (http://wol.ra.phy.cam.ac.uk/mackay/selection.nc.ps.gz).

[205] Macready, William G.; Wolpert, David H.: What Makes An Optimization Problem Hard? The Santa Fe Institute Technical Report SFI-TR-95-05-046, 1996. { 50} (ftp://ftp.santafe.edu/pub/wgm/hard.ps).

[206] Mammen, Enno: When does Bootstrap work? Springer-Verlag, New York, 1992. { 46, 47, 486}

[207] Manderick, Bernard: Weger, Mark de; Spiessens, Piet: The Genetic Algorithm and the Structure of the Fitness Landscape. In: Belew & Booker (1991[34]), ICGA 4, 1991, S. 143 - 150. { 51}

[208] Mani, I.; Maybury, M.T. (eds.): Advances in automatic Text Summarization. MIT Press, 1999. { 29, 149}

[209] Männer, Reinhart; Manderick, Bernhard (eds.): Proceedings of the Second Conference on Parallel Problem Solving from Nature, PPSN II. Amsterdam, 1992. { 529, 532, 537}

[210] Manning, Christopher D.; Schütze, Hinrich: Foundation of statistical natural language processing. MIT Press, 1999. { 26}

[211] Martignon, Laura; Laskey, Kathryn Blackmond: Bayesian strategies for machine learning: Rule Extraction and Concept Detection. In: Kappen & Gielen (1995[179]), S. 85 - 90. { 24}

[212] Martinetz, Thomas: Selbstorganisierende neuronale Netzwerkmodelle zur Bewegungssteuerung. St. Augustin, 1992. { 73, 279}

[213] Mathis, B.A.; Rush, J.E.: Abstracting. The Nature and Definition of Abstracting. In: Dym (1985[102]), S. 445 - 484. { 29, 149}

[214] McLachlan, G.J.; Basford, K.E.: Mixture Models. New York, 1987. { 17, 25, 49}

[215] Merkl, Dieter: Exploration of text collections with hierarchical feature maps. In: Belkin et al. (1997[39]), SIGIR 1997, S. 186 - 195. { 33, 37}
(http://www.acm.org/pubs/articles/proceedings/ir/258525/p186-merkl/p186-merkl.pdf).

[216] Merz, Christopher John: Classification and Regression by Combining Models. Ph.D. Dissertation, University of California, 1998. { 45, 49, 51, 56, 353}
(http://www.ics.uci.edu/~cmerz/thesis.ps).

[217] Metzinger, Thomas: Subjekt und Selbstmodell. F. Schöningh, Paderborn, 1993. { 235}

[218] Metzinger, Thomas (Hersg.): Bewußtsein. 3 erw. Auflage, F. Schöningh, Paderborn, 1996. {234}

[219] Miller, David R. H.; Leek, Tim; Schwartz, Richard M.: A hidden Markov model information retrieval system. In: Hearst et al. (1999[160]), SIGIR 1999, S. 214 - 221. { 26}
(http://www.acm.org/pubs/articles/proceedings/ir/312624/p214-miller/p214-miller.pdf).

[220] Minsky, Marvin: The society of mind. New York, 1986. { 48}

[221] Minsky, Marvin: Commonsense-Based Interfaces. In: Communications of the ACM, 43 (8), 2000. { 48}
(http://www.acm.org/pubs/articles/journals/cacm/2000-43-8/p66-minsky/p66-minsky.pdf).

[222] Mitra, Mandar; Singhal, Amit; Buckley, Chris: Improving automatic query expansion. In: Croft et al. (1998[83]), SIGIR 1998, S. 206 - 214. { 226}
(http://www.acm.org/pubs/articles/proceedings/ir/290941/p206-mitra/p206-mitra.pdf).

[223] Moody, John E.; Hanson, Steve J.; Lippmann, Richard P. (eds.): Advances in Neural Information Processing Systems 4, NIPS 4. San Mateo, Calif., 1992. { 541}

[224] Mori, N.; Imanishi, S.; Kita, H.; Nishikawa, Y.: Adaption to Changing Environments by Means of a Memory Based Thermodynamical Genetic Algorithm. In: Bäck (1997[24]), ICGA-97, S. 299 - 306. { 18}

[225] Mori, N.; Kita, H.; Nishikawa, Y.: Adaption to Changing Environments by Means of a the Feedback Thermodynamical Genetic Algorithm. In: Eiben et al. (1998[108]), PPSN V, S. 149 - 158. { 18}

[226] Morik, Katharina: Maschinelles Lernen. In: Görz (1993[151]), S. 246 - 301. { 30} (http://www-ai.cs.uni-dortmund.de/LEHRE/VORLESUNGEN/MLRN/SKRIPT/handbuch_ki-ml.pdf).

[227] Mostafa, J.; Mukhopadhyay, S.; Palakal, M.; Lam, W.: A multilevel approach to intelligent information filtering: model, system, and evaluation. In: ACM Trans. Inf. Syst. 15, 4 (Oct. 1997), S. 368 - 399. { 29, 33, 143, 193} (http://www.acm.org/pubs/articles/journals/tois/1997-15-4/p368-mostafa/p368-mostafa.pdf).

[228] Motro, A.; Smets, P. (eds.): Uncertainty Management in Information Systems. Kluwer, 1997. {19}

[229] Mühlenbein, Heinz: Open Worlds, Reflective Statistics and Stochastic Modelling. GMD, St. Augustin, 1994. { 18, 42, 45, 49, 57} (ftp://borneo.gmd.de/pub/as/janus/ref94_14.ps).

[230] Mühlenbein, Heinz: Algorithms, data and hypotheses - Learning in open worlds. GMD, St. Augustin, 1995. { 18, 42, 45, 49, 57} (ftp://borneo.gmd.de/pub/as/janus/ref95_4.ps).

[231] Nadel, L.; Stein, D.: 1992 Lectures in Complex Systems. Addison-Wesley, 1993. { 544}

[232] Natke, H. Guenther; Ben-Haim, Yakov (eds.): Uncertainty: Models and Measures. Berlin, 1997. { 19}

[233] Newell, A.: Unified Theories of cognition. Cambridge, Cambridge University Press, 1990. { 23, 26, 28, 36, 38, 39, 40, 41, 42, 43, 44, 50, 52, 143, 237}

[234] Nguyen, H.T.: Fuzzy Systems. Kluwer, 1999. { 30, 53, 139}

[235] Nicolis, G.; Prigogine, I.: Self-Organisation in Non-Equilibrium Systems, New York, 1977. { 18}

[236] Nie, Jian-Yun: Towards a probabilistic modal logic for semantic-based information retrieval. In: Belkin et al. (1992[37]), SIGIR 1992, S. 140 - 151. { 27} (http://www.acm.org/pubs/articles/proceedings/ir/133160/p140-nie/p140-nie.pdf).

[237] Nissen, Volker: Evolutionäre Algorithmen, Wiesbaden, 1994. { 35, 36, 115, 182, 287}

[238] Nordin, Peter: Evolutionary Programm Induction of Binary Machine Code and its Applications. Münster, 1997. { 37, 67, 169, 236, 292, 309}

[239] Novak, V.; Perfilieva, I.: Mathematical Principles of Fuzzy Logik. Kluwer, 1999. { 30, 53, 139}

[240] Oja, E.: PCA: Algorithms and Applications. In: Proceeding of World Congress on Neural Networks. Portland, Vol. 2, 1993, S. 396 - 400. { 16}

[241] Omohundro, Stephen M.: The Delaunay triangulation and function learning. TR-90-001, International Computer Sciene Institute, Berkley, 1990. { 478} (http://www.icsi.berkeley.edu/ftp/pub/techreports/1990/tr-90-001.pdf).

[242] Overbeck-Larisch, Maria; Dolejsky, Wolfgang: Stochastik mit Mathematica. Vieweg Verlag, 1998. { 23}

[243] Owsnicki-Klewe, Bernd; Luck, Kai v.; Nebel, Bernhard: Wissensrepräsentation und Logik. In: Görz (1993[151]), S. 3 - 204. { 27}

[244] Paaß, Gerhard; Kindermann, Jörg: Bayesian query construction for neural network models. In: Tesauro et al. (1995[331]), NIPS 7, 1995a, S. 443 - 450. { 33} (http://nips.djvuzone.org/djvu/nips07/0443.djvu).

[245] Paaß, Gerhard; Kindermann, Jörg: Konstruktive Datenerhebung mit reflektiven neuronalen Netzen. In: GMD-Spielgel, 2, 1995b, S. 37 - 44. { 33}

[246] Paaß, Gerhard; Kurfeß, Franz (Hrsg.): Wissensverarbeitung mit Neuronalen Netzen. GMD-Studie 221, GMD, Sankt Augustin, 1993. { 49}

[247] Panyr, Jiri: Automatische Klassifikation und Information Retrieval, Tübingen, 1986a. { 14, 16, 64, 161, 192, 213}

[248] Panyr, Jiri: Relevanzproblematik in Information-Retrieval-Systemen. In: Nachrichten für Dokumentation 37, S. 2 - 4, 1986b. { 43, 159}

[249] Panyr, Jiri: Probabilistische Modelle in Information-Retrieval-Systemen. In: Nachrichten für Dokumentation 37, S. 60 - 66, 1986c. { 23, 48}

[250] Panyr, Jiri: Die Theorie der Fuzzy-Mengen und Information-Retrieval-Systeme. In: Nachrichten für Dokumentation 37, S. 163 - 168, 1986d. { 30}

[251] Panyr, Jiri: Vektorraum-Modell und Clusteranalyse in Information-Retrieval-Systemen. In: Nachrichten für Dokumentation 38, S. 13 - 20, 1987a. { 25, 48, 159, 161}

[252] Panyr, Jiri: Interaktive Retrievalstrategien: Relevanzfeedback. In: Nachrichten für Dokumentation 38, S. 145 - 152, 1987b. { 19, 37, 43, 232, 240, 250, 251, 264, 284}

[253] Panyr, Jiri: Informationsfilterung, Informationsmüll, Informationsökologie. Interner Bericht der Siemens AG, München, 1994. { 33, 143, 193, 233}

[254] Pareto, V.: Cours D'Economie Politique, Vol. 1. Lausanne, 1896 (nach Zitzler (1999[375])). {113}

[255] Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar: Fraktale: Bausteine des Chaos. Springer, Berlin, 1992. { 227}

[256] Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar: Chaos: Bausteine der Ordnung. Springer, Berlin, 1994. { 227}

[257] Pfeiffer-Jäger, G.: Referat und Referieren. Linguistische Beiträge zu ihrer Applikation in der Information und Dokumentation. In: Germanistische Linguistik, 1/2, 1980, S. 1 - 180. { 29, 149}

[258] Plutowski, M.; White, H.: Selecting concise training sets from clean data. In: IEEE Transactions on Neural Networks 4 (2), 1993, S. 305 - 318. { 33}

[259] Plutowski, M.: Selected Training Exemplars for Neural Network Learning. PhD thesis, University of California, San Diego, 1994. { 33}

[260] Poggio, Tomaso; Girosi, Federico: A theory of networks for approximation and learning. Technical Report AIM-1140, Artificial Intelligence Laboratory and Center for Biological Information Processing, Whitaker College, Massachusetts Institute of Technology, 1989. { 315} (ftp://publications.ai.mit.edu/ai-publications/1000-1499/AIM-1140.ps.Z).

[261] Ponte, J.M.; Croft, W.B.: A Language Modeling Approach to Information Retrieval. In: Croft et al. (1998[83]), SIGIR 1998, S. 275 - 281. { 170} (http://www.acm.org/pubs/articles/proceedings/ir/290941/p275-ponte/p275-ponte.pdf).

[262] Popper, Karl R.: Objektive Erkenntnis. Hamburg, 2. Aufl. 1994. { 18, 38, 41, 45, 49, 50, 237}

[263] Popper, Karl R.: Alles Leben ist Problemlösen. Zürich, 4. Aufl. 1995. { 18, 38, 41, 45, 49, 50, 237}

[264] Press, W.; Flannery, B.; Teukolsky, S.; Vetterling, W.: Numerical Recipes in C - the Art of scientific Computing. Cambridge, 1988. { 476}

[265] Preskill, John: Quantum Information and Computation. Lecture Notes for Physics 229, Springer, 1998. { 153}

[266] Pylyshin, Zenon W.: The Role of Cognitive Architectures in Theories of Cognition. In: VanLehn (1991[345]), S. 189 - 223. { 28, 36}

[267] Quinlan, J.R: C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993. { 31, 45, 49, 51, 56, 353}

[268] Quinlan, J.R: Bagging, boosting, and c4.5. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence. AAAI Press/MIT Press, 1996, S. 725 – 730. { 31, 45, 49, 51, 56, 353}

[269] Radecki, T.: Fuzzy set theoretical approach to document retieval. In: Information Processing and Management, 15, 1979, S. 247 - 259. { 30}

[270] Ram, Ashwin; Moorman, Kenneth (eds.): Understanding Language Understanding. MIT Press, 1999. { 26}

[271] Rao, R. Bharat; Gordon, Diana; Spears, William: For Every Generalization Action, Is There Really an Equal and Opposite Reaction? Analysis of the Conservation Law for Generalization Performance. 1995. { 50, 51}
(http://www.aic.nrl.navy.mil:80/~spears/papers/mlc95.ps).

[272] Rauch, W.; Strohmeier, F.; Hiller, H.; Schlögl, C. (Hersg.): Mehrwert von Information - Professionalisierung der Informationsarbeit, Proceedings des 4. Internationalen Symposiums für Informationswissenschaft (ISI94), Graz, 1994. { 536}

[273] Rawlings, J.O.; Pantula, S.G.; Dickey, D.A.: Applied Regression Analysis. 2 Auflage, New York, 1998. { 25}

[274] RayChaudhur, Tirthankar; Hamey, Leonard G.C.: An Algorithm for Active Data Collection for Learning; Feasibility Study with Neural Networks. Macquarie University Technical Report No. 95-173C, 1995. { 33, 58, 442, 443}
(ftp://ftp.mpce.mq.edu.au/pub/comp/techreports/950173.raychaudhuri.ps)

[275] RayChaudhuri, T.; Hamey, Leonard G.C: Cost Effective Querying Leading to Dual Control. Macquarie Computing Report 96-07, 1996. { 33, 58, 442, 442, 443, 444, 445, 446, 448, 454}
(ftp://ftp.mpce.mq.edu.au/pub/comp/techreports/960007.raychaudhuri.ps.Z).

[276] RayChaudhuri, T.: Seeking the Valuable Domain - Query Learning in a Cost-Optimal Perspective. Dissertation, Macquarie Universität, Sidney, 1997. { 33, 58} (ftp://ftp.mpce.mq.edu.au/pub/comp/papers/raychaudhuri.phd_thesis.97.ps.Z).

[277] Rechenberg, Ingo: Evolutionsstrategie ‘94. Stuttgart, 1994. { 35, 36, 52, 115, 127, 182}

[278] Reibnitz, Ute von: Szenarien: Optionen für die Zukunft. McGraw-Hill, Hamburg, 1987. { 165}

[279] Rijsbergen, C. J. van: Information Retieval. London, 1979. { 15, 147}

[280] Rijsbergen, C. J. van: A non-classical logic for information retrieval. In: Computer Journal, 29 (6), 1986, S. 481 - 485. { 27}

[281] Ringuest, J.L.: Multiobjective Optimization: Behavioral and Computational Considerations. Kluwer, 1992. { 21, 34}

[282] Ritter, Helge; Martinetz, Thomas; Schulten, Klaus: Neuronale Netze - Einführung in die Neuroinformatik selbstorganisierender Netzwerke. Addison-Wesley, Bonn, 2. Auflage, 1991. { 16, 33, 44, 48, 71, 241}

[283] Robertson, S. E.: Progress in Documentation, Theories and Models in Information Retrieval. In: Journal of Docum. 33, 126-148, 1977a. { 16, 23, 48}

[284] Robertson, S. E.: The Probabilistic Character of Relevance. In: Inform. Process. & Management 13, 247-251, 1977b. { 16, 23, 48}

[285] Robertson, S.E.: The Probability Ranking Principle in IR. In: Journal of Docum. 33, 294-304, 1977c. { 16, 23, 24, 48}

[286] Robertson S.E.: Indexing Theory and Retrieval Effectivness. In: Drexel Library Quartely 14, 40-56, 1979. { 16, 23, 48}

[287] Robertson, S.E.; Walker, S.: Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In: Croft & Rijsbergen (1994[82]), SIGIR 1994, S. 232 - 241. {25, 55} (http://www.acm.org/pubs/articles/proceedings/ir/188490/p232-robertson/p232-robertson.pdf).

[288] Rölleke, Thomas; Fuhr, Norbert: Retrieval of complex objects using a four-valued logic. In: Frei et al. (1996[121]), SIGIR 1996, S. 206 - 214. { 27} (http://www.acm.org/pubs/articles/proceedings/ir/243199/p206-rolleke/p206-rolleke.pdf).

[289] Rosenbloom, Paul S.; Laird, John E.; Newell, Allen (eds.): The Soar papers: research on integrated intelligence. Cambridge, Mass., 1993. { 28, 39}

[290] Rosenthal, Richard E.: Concepts, Theory, And Techniques: Principles of Multiobjective Optimization. In: Decision Science, 16, 1985, S. 133 - 152. { 21, 34}

[291] Roussinov, Dmitri; Tolle, Kristine; Ramsey, Marshall; McQuaid, Michael; Chen, Hsinchun: Visualizing Internet search results with adaptive self-organizing maps (demonstration abstract). In: Hearst et al. (1999[160]), SIGIR 1999, S. 336. { 33, 37} (http://www.acm.org/pubs/articles/proceedings/ir/312624/p336-roussinov/p336-roussinov.pdf).

[292] Ryan, Thomas: Modern Regression Methods. New York, 1997. { 25}

[293] Salton, Gerard (Herg.): The Smart Retrieval System: Experiments in Automatic Document Processing. Prentice Hall, 1971. { 16, 25, 48, 240, 529, 535}

[294] Salton, Gerard; McGill, Michael J.: Information Retrieval - Grundlegendes für Informationswissenschaftler. New York, MacGraw-Hill, 1987. { 15, 159, 160, 161}

[295] Santo, Y.; Kita, H.: Optimization of Noisy Fitness Functions by Means of Genetic Algorithms using History of Search. In: Schoenauer et al. (2000[300]), PPSN VI, S. 571 - 580. { 18}

[296] Savoy, J.; Ndarugendawmo, M.; Vrajitoru, D.: Report on the TREC-4 experiment: combining probabilistic and vector-space schemes. In: Harman (1996[158]), TREC-4. { 32} (http://trec.nist.gov/pubs/trec4/papers/savoy.ps.gz).

[297] Schaffer, C.: A conservation law for generalization performance. In: Proceedings of the Eleventh International Conference on Machine Learning. 1994, S. 259 - 265. { 50}

[298] Schank, R.: Conceptual Information Processing. Oxford, 1975. { 28}

[299] Schapire, Robert E.; Singer, Yoram; Singhal, Amit: Boosting and Rocchio Applied to Text Filtering. In: Croft et al. (1998[83]), SIGIR 1998, S. 215 - 223. { 31, 34, 38, 143, 193} (http://www.acm.org/pubs/articles/proceedings/ir/290941/p215-schapire/p215-schapire.pdf).

[300] Schoenauer, Marc; Deb, Kalyanmoy; Rudolph, Günter; Yao, Xin; Lutton, Evelyne; Merelo, Juan Julian; Schwefel, Hans-Paul: Proceedings of the Sixth Conference on Parallel Problem Solving from Nature, PPSM VI, Berlin, 2000. { 532, 548}

[301] Scholtes, Johannes Cornelis: Neural Networks in Natural Language Processing and Information Retrieval. Ph.D. Dissertation, University of Amsterdam, 1993. { 16, 18, 25, 26, 29, 33, 37, 152}

[302] Schöneburg, Eberhard; Hansen, Nikolaus; Gawelczyk, Andreas: Neuronale Netze. Haar bei München, 1990. { 33, 44, 48, 241}

[303] Schuchmann, Marco; Sanns, Werner: Statistik mit Mathematica. R. Oldenbourg Verlag, 1999a. {23, 25, 34}

[304] Schuchmann, Marco; Sanns, Werner: Nichtparametrische Statistik mit Mathematica. R. Oldenbourg Verlag, 1999b. { 23, 34}

[305] Schwefel, Hans-Paul: Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. Birkhäuser Verlag, Basel/Stuttgart, 1977. { 35, 36, 115, 127, 182, 287}

[306] Schwefel, Hans-Paul: Numerical optimization of computer models. Chichester, 1981. { 35, 36, 115, 127, 129, 182, 287}

[307] Schwefel, Hans-Paul: Evolution and Optimum seeking. New York, 1995. { 35, 36, 115, 127, 182}

[308] Seung, S.; Opper, M.; Smopolinsky, H.: Query by committee. In: Proceedings of the Fifth Workshop of Computational Learning Theory. Morgan Kaufmann, San Mateo, California, S 287 - 294, 1992. { 33, 58, 442}
(http://hebb.mit.edu/people/seung/papers/query.ps.gz).

[309] Shao, J.; Tu, D.: The Jackknife and Bootstrap. Springer-Verlag, New York, 1995. { 46, 47, 91, 486}

[310] Siegelmann, Hava T.: Neural Networks and Analog Computation - Beyond the Turing Limit. Birkhäuser Verlag, 1999. { 148, 153}

[311] Sinclair, Alistair: Algorithms for random generation and counting: a Markov chain approach. Basel, 1992. { 26, 169}

[312] Singhal, Amit K.: Term Weighting Revisited. Ph.D. thesis, Cornell University, 1997. { 19, 29}

[313] Skalak, David B.: Prototype Selection for Composite Nearest Neighbor Classifiers. Dissertation, Department of Computer Science, University of Massachusetts, 1996. { 45, 49, 51, 56, 353}
(http://www.cs.cornell.edu/Info/People/skalak/thesis-header-dspace.ps.gz).

[314] Sloan, I.H.; Wozniakowski, H.: An intractability result for multiple integration. Columbia University Computer Science Technical Report CUCS-019-96, 1996. { 457, 465, 470, 477}
(http://www.cs.columbia.edu/~library/TR-repository/reports/reports-1996/cucs-019-96.ps.gz).

[315] Soboroff, Ian M.; Nicholas, Charles K.; Kukla, James M.; Ebert, David S.: Visualizing document authorship using n-grams and latent semantic indexing. In: Proceedings of the workshop on New paradigms in information visualization and manipulation, 1998, S. 43 - 48. { 16, 25}
(http://www.acm.org/pubs/articles/proceedings/cikm/275519/p43-soboroff/p43-soboroff.pdf).

[316] Sollich, Peter: Asking Intelligent Questions - the Statistical Mechanics of Query Learning. PhD thesis, University of Edinburgh, 1993. { 33} (ftp://archive.cis.ohio-state.edu/pub/neuroprose/Thesis/sollich.thesis.tar.Z).

[317] Sollich, Peter: Query construction, entropy and generalisation in neural network models. In: Physical Review E 49, 1994, S. 4637 - 4651. { 33} (http://www.mth.kcl.ac.uk/~psollich/papers/QueryPREIV.ps.gz).

[318] Song, Fei; Croft, Bruce W.: A general Language Model for Information Retrieval. In: Hearst et al. (1999[160]), SIGIR 1999, S. 279 - 280. { 170} (http://www.acm.org/pubs/articles/proceedings/ir/312624/p279-song/p279-song.pdf).

[319] Sparck Jones, K: Automatic Keyword Classification for Information Retrieval. London: Butterworths 1971. { 16}

[320] Spink, Amanda: Term relevance feedback and query expansion: relation to design. In: Croft & Rijsbergen (1994[82]), SIGIR 1994, S. 81 - 90. { 226} (http://www.acm.org/pubs/articles/proceedings/ir/188490/p81-spink/p81-spink.pdf).

[321] Stadler, Peter F.: Landscapes and Their Correlation Functions. The Santa Fe Institute Technical Report SFI-TR-95-05-049, 1995. { 51} (http://www.santafe.edu/sfi/publications/Working-Papers/95-05-049.ps).

[322] Stagge, P.: Averaging Efficiently in the presence of Noise. In: Eiben et al. (1998[108]), PPSN V, S. 188 - 197. { 18}

[323] Stein, Barry E.; Meredith, M. Alex: The Merging of the senses. MIT Press, 1993. { 435}

[324] Steuer, R.E.: Multiple Criteria Optimization: Theory, Computation and Application. New York, 1986. { 21, 34}

[325] Strampp, Walter: Höhere Mathematik mit Mathematica, Band 1: Grundlagen, Lineare Algebra. Vieweg Verlag, 1996. { 25}

[326] Strube, Gerhard; Habel, Christopher; Hemforth, Barbara; Konieczny, Lars; Becker, Barbara: Kognition. In: Görz (1993[151]), S. 303 - 365. {) 28}

[327] Sun, Ron; Bookman, Lawrence A.: Computational Architectures integrating Neural and Symbolic Processes. Bosten, 1995. { 49}

[328] Swanson, Don R.: Information Retrieval als Versuch-Irrtum-Prozess. Studientexte Fachrichtung Informationswissenschaft, Universität des Saarlandes, Herausgeber: Wolfgang von Keitz, 1987 (Original: Information Retrieval as an Trial-and-Error Process. Library Quaterly, Vol 47, H. 2, 1977). { 15, 55, 233, 236, 237, 238}

[329] Swets, John A.: Signal detection theory and ROC analysis in psychology and diagnostics: collected papers. Mahwah, NJ, 1996. { 22}

[330] Tahani, V.: A fuzzy model of document information retrieval systems. In: Information Processing and Management, 12 (3), 1976, S. 177 - 188. { 30}

[331] Tesauro, Gerald; Touretzky, David S.; Leen, Todd (eds.): Advances in Neural Information Processing Systems 7, NIPS 7. Cambridge, 1995. { 531, 535, 544}

[332] Thrun, Sebastian; Möller, K.: Active exploration in dynamic environments. In: Moody et al. (1992[223]), NIPS 4, S. 531 - 538. { 33}
(http://www.cs.cmu.edu/~thrun/papers/thrun.nips91.ps.gz).

[333] Tong, R.M; Shapiro, D.G.; McCune, B.P.; Dean, J.S.: A rule-based approach to information retrieval: Some results and comments. In: Proceedings of the National Conference on Artificial Intelligence, 1983, S. 411 - 415. { 27}

[334] Tong, R.M; Shapiro, D.G.: Experimental investigations of uncertainty in a rule-based (system for information retrieval. In: International Journal of Man-Machine Studies, 22, 1985, S. 265 - 282. { 27}

[335] Touretzky, David S. (ed.): Advances in Neural Information Processing Systems NIPS 2, Morgan Kaufmann, San Francisco, CA., 1990. { 521}

[336] Touretzky, David S.; Mozer, Michael C.; Hasselmo, Michael E. (eds.): Advances in Neural Information Processing Systems 8, NIPS 8. Cambridge, 1996. { 523}

[337] Traub, J.F.; Wasilkowski, G. W.; Wozniakowski, H.: Information-Based Complexity. Academic Press, 1988. { 16, 457, 465, 470, 477}

[338] Traub, J. F.; Wozniakowski, H.: Wege aus der Unberechenbarkeit. Spektrum der Wissenschaft., 4/1994, S. 64- 69. { 457, 465, 470, 471, 477}

[339] Trostel, Rudolf: Mathematische Grundlagen der technischen Mechanik, Band 1: Vektor- und Tensoralgebra. Braunschweig, 1993. { 25}

[340] Trostel, Rudolf: Mathematische Grundlagen der technischen Mechanik, Band 2: Vektor- und Tensoranalysis. Braunschweig, 1997. { 25}

[341] Turtle, Howard; Croft, W. Bruce: Inference networks for information retrieval. In: Vidick (1990[350]), SIGIR 1990, S. 1 - 24. { 28} (http://www.acm.org/pubs/articles/proceedings/ir/96749/p1-turtle/p1-turtle.pdf).

[342] Turtle, Howard; Croft, W. Bruce: Evaluation of an Inference network-based retrieval model. In: ACM Trans. Inf. Syst. 9, 3 (Jul. 1991), S. 187 - 222. { 24, 28} (http://www.acm.org/pubs/articles/journals/tois/1991-9-3/p187-turtle/p187-turtle.pdf).

[343] Turtle, Howard R.; Croft, W. Bruce: Uncertainty in Information Retrieval Systems. In: Motro & Smets (1997[228]), S. 189 - 224. { 19, 20, 27, 52}

[344] Turtle, Howard: Inference Networks for Document Retrieval. Ph.D. Dissertation, Computer Science Department, University of Massaschusetts, 1991. { 28}

[345] VanLehn, Kurt (ed.): Architectures for Intelligence. The Twenty-Second Carnegie Mellon Symposium on Cognition. Hillsdale, 1991. { 23, 28, 546}

[346] Vavak, F.; Fogarty, T.C.; Jukes, K.: A Genetic Algorithm with Variable Range of Local Search for Tracking Changing Environments. In: Voigt et al. (1996[353]), PPSN IV, S. 376 - 385. { 18}

[347] Veldhuizen, David A. Van: Multiobjective Evolutionary Algorithms: Classification, Analyses and new Innovations. Dissertation, Graduate School of Engineering of the Air Force Institute of Technology, 1999. { 21, 34, 35, 112, 131} (http://www.lania.mx/~ccoello/EMOO/veldhuizen99a.ps.gz).

[348] Venkatesh, Santosh S.; Snapp, Robert R.; Psaltis, Demetri: Bellman strikes again!: the growth rate of sample complexity with dimension for the nearest neighbor classifier. In: Proceedings of the fifth annual ACM workshop on Computational learning theory, 1992, S. 93 - 102. { 16} (http://www.acm.org/pubs/articles/proceedings/colt/130385/p93-venkatesh/p93-venkatesh.pdf).

[349] Verleysen, M. (ed.): ESANN96 European Symposium on Artificial Neural Networks, D-Facto Publishers, Brussels, 1996. { 535}

[350] Vidick, J.-L.: SIGIR 1990, Proceedings of the 13'th annual international ACM-SIGIR conference on Research and development in information retrieval. Brussel, Belgien, 1990. { 536} (http://www.acm.org/pubs/contents/proceedings/ir/96749/index.html).

[351] Vogt, Christopher C.; Cottrell, Garrison W.: Predicting the performance of linearly combined IR systems. In: Croft et al. (1998[83]), SIGIR 1998, S. 190 - 196. { 31, 38} (http://www.acm.org/pubs/articles/proceedings/ir/290941/p190-vogt/p190-vogt.pdf).

[352] Vogt, Christopher C.: Adaptive Combination of Evidence for Information Retrieval. Dissertation, Computer Science, University of California, San Diego, 1999. { 22, 31, 32, 38, 56, 532} (http://www.chapman.edu/~vogt/papers/thesis.pdf).

[353] Voigt , Hans-Michael; Ebeling, Werner; Rechenberg, Ingo; Schwefel, Hans-Paul: Proceedings of the Fourth Conference on Parallel Problem Solving from Nature, PPSN IV, Berlin, 1996. { 552}

[354] Waller, W.G.; Kraft, D.H.: A mathematical Modell for weighted Boolean retrieval systems. In: Information Procesing and Management, 15 (5), 1979, S. 235 -2 45. { 25}

[355] Walley, P. : Statistical Reasoning with Imprecise Probabilities. London, 1991. { 34}

[356] Walter, Jörg A.: Rapid Learning in Robotics. Dissertation, Technische Fakultät der Universität Bielefeld, Bielefeld, 1996. { 68} (http://www.techfak.uni-bielefeld.de/~walter/pub/book/jwBook.ps.gz).

[357] Warwick, K.; Kárny, M. (eds.): Computer Intensive Methods in Control and Signal Processing: The Curse of Dimensionality. Birkhäuser Verlag, Basel/Stuttgart, 1997. { 16, 25}

[358] Werschulz, A. G.; Wozniakowski, H.: What is the complexity of surface integration? Columbia University Computer Science Technical Report CUCS-004-99, Revised version: August 21, 2000. { 457, 465, 470, 477} (http://www.cs.columbia.edu/~library/TR-repository/reports/reports-1999/cucs-004-99.pdf).

[359] Wilke, Fabiana Zamora; Franciosi, Beatriz Regina Tavares; Oliveira, Paulo Werlang; Claudio, Dalcidio Moraes: Modelling the Measurement Uncertainty by Intervals. In: Journal of Universal Computer Science, vol. 4, no. 1 (1998), S. 82-88. { 34}

[360] Williams, Hugh E.: Cafe: An indexed approach to searching genomic databases. In: Croft et al. (1998[83]), SIGIR 1998, S. 389. { 147} (http://www.acm.org/pubs/articles/proceedings/ir/290941/p389-williams/p389-williams.pdf).

[361] Witten, Ian H.; Smith, Tony C.: Learning Language using Genetic Algorithms. Department of Computer Science, The University of Waikato Hamilton, New Zealand, 1996. { 23} (http://www.cs.waikato.ac.nz/~ml/publications/1996/TCS-IHW96.pdf)

[362] Witten, Ian H.; Frank, Eibe: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, 1999. { 30, 53}

[363] Witten, Ian H.; Frank, Eibe; Trigg, Len; Hall, Mark; Holmes, Geoffrey; Cunningham, Sally Jo: Weka: Practical Machine Learning Tools and Techniques with Java Implementations. Department of Computer Science, University of Waikato Hamilton, New Zealand, 1999. { 30} (http://www.cs.waikato.ac.nz/~ml/publications/1999/99IHW-EF-LT-MH-GH-SJC-Tools-Java.pdf).

[364] Witten, Ian H.; Ting, Kai Ming: Issues in Stacked Generalization. Department of Computer Science, The University of Waikato Hamilton, New Zealand, 1999; auch in: Journal of Artificial Intelligence Research 10 (1999) 271-289. { 31, 45, 49, 51, 56, 353} (http://www.cs.waikato.ac.nz/~ml/publications/1999/99KMT-IHW-Issues.pdf).

[365] Wolpert, David H.: Stacked Generalization. Los Alamos, NM, 1990; auch in: Neural Networks, 5, 1992, S. 241 - 259. {} (ftp://archive.cis.ohio-state.edu/pub/neuroprose/wolpert.stack_gen.ps.Z).

[366] Wolpert, David H.: On the connection between in-sample testing and generalization error. In: Complex Systems 6, 1992, S. 47 - 94. { 31, 45, 49, 50, 51, 56, 353}

[367] Wolpert, David H.: Combining Generalizers using Partitions of the Learning set. In: Nadel & Stein (1993[231]), S. 489 - 500. { 31, 45, 49, 51, 56, 353}

[368] Wolpert, David H.: On Bias plus Variance. Santa Fe Institute Technical Report TR 95-007, 1995a; auch in: Neural Computation, 9: 6, S. 1211 - 1243, 1997. { 98, 102} (http://www.santafe.edu/sfi/publications/Working-Papers/95-08-074.ps)

[369] Wolpert, David H. (ed.): The Mathematics of Generalization. Addison-Wesley, 1995b. { 30}

[370] Wolpert, David H.; Macready, William G.: No Free Lunch Theorems for Search. The Santa Fe Institute Technical Report SFI-TR-95-02-010, 1996. { 50} (ftp://ftp.santafe.edu/pub/wgm/nfl.ps).

[371] Wong, S. K. M.; Yao, Y. Y.: On modeling information retrieval with probabilistic inference. In: ACM Transactions on Information Systems, Volume 13 , Issue 1 (1995), S. 38-68. { 23} (http://www.acm.org/pubs/articles/journals/tois/1995-13-1/p38-wong/p38-wong.pdf).

[372] Wozniakowski, Henryk: Overview of Information-Based Complexity. Columbia University Computer Science Department Report CUCS-024-96, 1996a.{ 16, 457, 470, 477} (http://www.cs.columbia.edu/~library/TR-repository/reports/reports-1996/cucs-024-96.ps.gz).

[373] Wozniakowski, Henryk: Computational Complexity of Continuous Problems. Columbia University Computer Science Department Report CUCS-025-96, 1996b. { 16, 457, 470, 477} (http://www.cs.columbia.edu/~library/TR-repository/reports/reports-1996/cucs-025-96.ps.gz).

[374] Yang, J.-J.; Korfhage, R.R.: Query modifikation using genetic algorithms in vector space model. In: International Journal of Expert Systems 7 (2), 1994, S. 165 - 191. { 36}

[375] Zitzler, Eckart: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Dissertation, Institut für Technische Informatik und Kommunikationsnetze, TH Zürich, 1999. { 21, 34, 35, 112, 113, 121, 122, 124, 126, 131, 545}
(ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/Zitz1999.pdf).

[376] Zobel, Justin; Moffat, Alistair: Exploring the Similarity Space. In: SIGIR Forum, Vol. 32, 1998, Nr. 1, S. 18 - 34. { 19, 36, 47, 159, 160, 163, 292, 329}