Correlation Features And A Structured SVM Family For Phoneme Classification And Automatic Speech Recognition

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Naturwissenschaftlich-Technischen Fakultät II
– Physik und Mechatronik –
der Universität des Saarlandes

von

Andreas Beschorner

Saarbrücken
2014

Tag des Kolloqiums:	28.11.2014
Dekan:	Univ. Prof. Dr. Christian Wagner

Mitglieder des Prüfungsausschusses:

Univ. Prof. Dr. Michael Möller	*Vorsitzender*
Univ. Prof. Dr. Dietrich Klakow	*Gutachter*
Univ. Prof. Dr. Romanus Dyczij-Edlinger	*Gutachter*
Dr. Felix Felgner	*akademischer Beisitzer*

Correlation Features And A Structured SVM Family For Phoneme Classification And Automatic Speech Recognition

Abstract – Zusammenfassung

Abstract

The foremost aim of this thesis is to introduce concepts targeting at improving both phoneme classification and in line with this automatic speech recognition. The most distinctive part of the herein presented, new approach is that the different stages of the analysis, from feature vector creation to classification, are all developed upon the common basis. This foundation becomes apparent by the interaction of correlation and the formal structure of a tristate phoneme model that manifests itself in short time weak stationary characteristic and transitions between such segments within phonemes. The tristate layout is a topology that partitions a phoneme, or more generally an observed frame, into three main sections, **s**tart, **m**iddle and **e**nd. In combination with the well known Hidden Markov Model (HMM) it targets at modeling the above mentioned states of transitions and stationarity.

On the base of weak stationarity and the tristate structure, our approach evolves as follows. A stochastic process such as a speech signal that is short time weak stationary has first and second order moments independent of time t, they are affected only by the timespan between observations. This effect is reflected by the (auto)covariance of the process and carries over to (auto)correlation and to some degree to cross correlation. In this light, based on common MFCC feature vectors, we first analyze potential improvements when using autocorrelation data and due to motivating results introduce both new MFCC autocorrelation- and later specific cross correlation features. In this context we note that, in contrast to different components (roughly representing the different frequency bands) of a single MFCC vector, identical components across different MFCC vectors in general are not decorrelated.

In a subsequent step, the cross correlation transform is integrated into support vector classifiers used for phoneme classification such that a specialized reproducing kernel utilized by the classifiers is deduced directly from the transform. The theoretical prerequisites for the new kernel to be established are derived and proven along with its necessary requirements. Concerning the support vector machines, in line with the new reproducing kernel a family of classifiers is introduced. The structure of the latter evolves around immanent aspects inherited from concepts of phoneme representation and their acoustic progression: The above mentioned tristate model. Based on the topology of the latter and the construction of the features, a specifically structured collection of classes and associated support vector classifiers is designed under additional integration of correlation. All this aims at developing a framework that represents and models both stationarity and transitions within acoustical events to a degree not achieved by recognition and classification systems hitherto.

To prove the success of this approach, experiments are conducted to demonstrate the improved recognition rates resulting from the new topology. Further on, the framework is integrated into a common automatic speech recognition system and evaluated in this context. Again, experiments that compare the new approach to a standard recognition system reveal its potentials. Finally, prospects and suggestions for further potential improvements seclude the thesis.

Zusammenfassung

Das Hauptziel dieser Arbeit ist, zur Verbesserung der Klassifikation von Phonemen und als direkte Folge davon zur Verbesserung automatischer Spracherkennung beizutragen. Die ausschlaggebende Innovation ist hierbei, dass unterschiedliche Phasen – von der Erstellung der Klassifikations-Merkmale über die innere Struktur der Klassifizierer bis hin zu deren Gesamttopologie – von ein und derselben Grundidee aus deduziert werden. Diese manifestiert sich vor allem in der Interaktion von Korrelation und der verwendeten Tristate-Modellierung von Phonemen. Basis ist dafür die Sprache eigene Charakteristik der (schwachen) Kurzzeitstationarität, repräsentiert durch Segmente mit dieser Eigenschaft und Übergänge zwischen solchen. Die Tristate-Topologie partitioniert dabei Phoneme, oder allgemeiner Beobachtungen, in drei Bereiche, **S**tarte, **M**itte und **E**nde, und simuliert in Verbindung mit den bekannten Hidden Markov Modellen eben jene Zustandsfolgen von quasi statischen Momenten und Transitionen.

Auf Basis der Stationarität und der Tristate Struktur entfaltet sich unser Ansatz wie folgt. Wir betrachten ein Sprachsignal als eine Realisierung eines Zufallsprozesses, welcher innerhalb kurzer Segmente o.g. Eigenschaften annimmt. Durch diese wird die Zeitunabhängigkeit der ersten beiden statistischen Momente determiniert, d.h. die Momente werden allein durch zeitliche Differenzen von Beobachtungen charakterisiertt.

Mit wechselnden Segmenten und Transitionen zwischen diesen ändern sich daher Auto- und Kreuzkorrelation und in infolgedessen die durch sie definierten, neu entwickelten Merkmale. In diesem Sinne analysieren wir, basierend auf herkömmlichen MFCC-Vektoren, in einem ersten Schritt mögliche Verbesserungen durch Verwendung von Autokorrelationsdaten und entwickeln aufgrund motivierender Resultate im Weiteren spezielle (Kreuz-) Korrelationsmerkmale. Dabei hilft die Tatsache, dass im Gegensatz zu verschiedenen MFCC-Vektorkomponenten ein und desselben Merkmalvektors (innerhalb dessen die unterschiedliche Komponenten verschiedene Frequenzbänder repräsentieren), gleiche Einträge unterschiedlicher Vektoren im Allgemeinen nicht dekorreliert sind.

Im darauffolgenden Schritt geht die Operation der Korrelation direkt in die für die Phonemklassifikation benutzten Support Vektor Klassifizierer insofern ein, als dass deren (reproduzierender) Kern gewonnen wird aus besagter Transformation. Die dafür theoretischen Voraussetzungen werden hergeleitet und die notwendigen Eigenschaften des neuen reproduzierenden Kernes wird bewiesen. Einhergehend mit diesem speziellen Kern wird eine Familie aus Klassifizierern eingeführt, deren Struktur, den Features folgend, direkt an das Tristatemodel angelehnt und ebenfalls von der Korrelation beeinflusst ist. In ihrer Gesamtheit zielen die Konzepte darauf ab, die stationaritären Phasen als auch Transitionen zwischen verschiedenen Sprachsegmenten adäquater zu modellieren als bisherige Verfahren.

Die Verbesserung der Erkennungsrate im Vergleich zum Standardansatz wird anschließend anhand von vergleichenden Experimenten gezeigt, und im weiteren Verlauf wird das Verfahren eingebunden in ein allgemeines automatisches Spracherkennungssystem und auf diesem ausgewertet. Vergleichende Experimente mit Standardverfahren demonstrieren dabei das Potential des neuen Ansatzes, und Vorschläge zu Verbesserungen und Weiterentwicklungen schließen die Arbeit ab.

Aknowledgements

First of all I want to thank my family, who with their continuous support and endurance of my diverse moods always gave the strongest and best support I could have possibly asked for and wanted. Next, this thesis could not have been realized without the LSV department and Prof. Dietrich Klakow, my doctoral supervisor. Furthermore, the knowledge and help I gained from the functional analysis lectures of Prof. Jörg Eschmeier and his former doctoral student Christoph Barbian were fundamental for proving the theoretic parts on the linear transform based reproducing kernel.

Big thanks also to Theo Brandmüller, a professor for composition at Saarland Music Academy, who accepted and invited my into his composition class for further postgraduate studies. To my deepest regret, Theo has meanwhile passed away.

Further thanks fly to the proofreaders, Christian Pich, Jörg-Uwe Pott and Wolfgang Reichel.

Contents

Introduction

Challenges in machine learning methods

(voice from transmitter) :	MAYDAY, MAYDAY. HELLO. CAN YOU HEAR US? CAN YOU HEAR US, CAN YOU ...*NOISE*... OVER! WE ARE SINKING... WE ARE SINK...*NOISE*
trainee :	Hello!? Zis is ze German coast guahd.
(voice from transmitter):	WE ARE SINKING, WE'RE SINKING!
trainee :	What are you zinking about?

– Berlitz TV ad

Phoneme classification and speech recognition are complex issues due to the vast amount of varieties of many of their characteristics. First, there are influences from the physical state of the speaker such as age, gender, size or health condition. Second, one has to deal with local dialects and differences between native and – various kinds of – non native speakers. Third, individual speech characteristics (speed, tendency to swallow vowels, mumbling and many, many more) need to be considered. Fourth, impediments like noise, reverberation, background music or voices, attenuation or diffusion directly affect and impurify the timbre – and in addition, when performing automatic speech recognition (ASR) or phoneme classification, different recording and recognition system alter the original data in unequal ways.

Before initiating whatever kind of analysis, one needs to have a certain idea about what exactly will be analyzed and where to start: How large is my vocabulary? Do I want to recognize just words or complete sentences? Once this is determined, what are adequate features and how are they effectively and suitably represented? Assuming we have decided for a recognizer or classifier that builds on phonemes, further questions arise during the process itself and cannot be answered in advance. At present, most recognition systems subdivide phonemes to still smaller snippets, so called subphonemes. We will not only adopt this approach in this work but also extend and combine it with other transforms.

The set of phonemes itself is clearly language dependent. Hence, recorded speech needs to be labeled in some manner beforehand, either automatically/ (semi)supervised or manually. In a similar way, considering the next levels of larger building blocks such as single words or utterances, speech recognition requires further steps, wherein both sequences of phonemes and words are analyzed. Additional statistical information is therefore used to compute acoustic and language models representative for the given set of training and test data on which the system works. This information covers certain aspects of the language in question such as frequently occurring word sequences or different ways of speaking and pronouncing a known utterance.

To clarify this a little, consider the short remark *That is bad.* Potential variants are for instance *That's bad!*, *Tha's bad* or even *Tha's baad* with a very long and dark spoken last vowel. The language model allows for at least the most common forms and deviations. It thus becomes clear immediately that the training data must be representative: The generated language model rarely reflects complex grammatical idioms but mostly statistical data about word combinations instead. In the same manner, the acoustic model has to cope with multiple above mentioned varieties of pronunciation, speed or other characteristics on the word-phoneme level. Hence, the training data must be encompassing enough to guarantee sound recognition even across different dialects and genders.

Each step in the overall process from collecting and labeling data to defining and computing features, training classifiers, acoustic and language models to the final step of automatic speech recognition has its own intrinsic kinds of challenges. In this thesis, one of the most important ideas is that distinct parts share common foundations. Practically, we hence focus strongly on developing a system that carries specific characteristics from one step to another: from feature definition over mathematical transforms to classifier topologies.

In the same manner as many real world applications, ASR has foundations in finding a suitable (mathematic) model for tasks and subtasks. The well known MFCC features in this context give a good impression of the extent to which accurate structures and models have been and are successfully developed. On one hand, the vocal tract of a speaker and the psycho-physiological aspects of speech reception were for instance used to develop the *mel frequency cepstral coefficients/* (MFCC) features. On the other hand, in the early 1960s, Baum and his co-authors ([1], [2]) introduced the *Hidden Markov Model* (HMM). In short time intervals, speech behaves quasi stationary and quasi periodic over short periods of time. A HMMs is a probabilistic finite state machine that can capture both the stationarity and the pronunciation differences to a certain degree. This is the foundation for the acoustic model described above. Shortly afterwards in ([3], [4]) it was demonstrated that HMMs can be computed efficiently by dynamic programming methods and thus quickly found their way into diverse areas of research.

This thesis is mainly concerned with improvements at the lower levels (or first steps) of speech recognition with the distinct aim of improving phoneme classification and continuous speech recognition. In the first place this work address the challenging topic of enhancing the well known and commonly used MFCC-$\Delta - \Delta\Delta$ feature vectors. As mentioned above, the main contribution is, to offer an improved framework where several operations are based on the same base concepts. In this light, in addition to the introduction of new features, a special topology based on those is educed. Both the new features and the new structure shall reflect the nature of the given task and modify underlying methods in an effective manner. To be more accurate, the MFCC feature vectors are replaced by correlation data computed in a specific way from the MFCC data. The motivation for this stems from the short-time quasi stationarity of speech.
In the subsequent step we stray from the standard path of current automatic speech recognition systems insofar as that we replace the common set of HMMs by a family of support vector classifiers and calculate posterior probabilities afterwards. The correlation operation in the feature vector computation is a special case of a linear transform, and a new reproducing kernel for the support vector classifiers, based on a given linear transform, is deduced. Furthermore, the classifier family organized following a very specific structure, which itself is derived from the typical three state (tristate) design of phonemes. This topology partitions its objects into *start*, *middle* and *end* sections. Together with HMMs it aims at emulating both the short time (quasi) stationarity of and transitions within speech, including variances in the data. In our approach, the classifier family hence replaces the HMM transition/state model.

Combining all this we are capable of creating a system of multiple classes per phoneme in line with an associated, rather big set of classifiers that offers a more substantial and better model of the above mentioned speech characteristics. To summarize, the contributions of the thesis to current research are...

- ... the introduction of new correlation features for both phoneme classification and continuous speech recognition.

- ... the deduction of a new reproducing kernel for a phoneme classifier network, based on the correlation operation.

- ... the development of a classifier family following a topology that integrates the correlation operation into the standard tristate phoneme concept to model transition aspects more adequately and replace the common HMM approach.
- ... the modification of a common ASR system to deal with the new data. This makes it possible to compare the new approach to standard methods.

Structure of the thesis

Chapter one covers the mathematical backgrounds for the thesis. In the first section, the minimal amount of measure theory necessary for subsequent sections is briefly reviewed. The second sections introduces reproducing kernels (RK) and their associated reproducing kernel Hilbert spaces (RHKS), or more general Hilbert function spaces are explained. A RK is an operator (or function) that is directly related to the inner products of a Hilbert function space. This relation is the most important characteristic used in recognition and classification together with the fact that a reproducing kernel spans the underlying Hilbert function space. Given those two properties it is possible to transform features via RKs into the function space, in this context called RKHS.

Section three explains the so called *Theorem of Mercer* and its consequences, which are the foundations for the success of reproducing kernel methods in pattern recognition tasks. Subsequently, section four continues with examples ranging from basic reproducing kernels over construction of new ones to proving characteristic properties and the detailed derivation of some very common reproducing kernels and their associated Hilbert function spaces. In the fifth section we derive the specific reproducing kernel motivated by the correlation operation for computing the aforementioned new features.

Sections six and seven deal with basics of convex optimization and its application in the context of support vector classifiers. Both the binary and the multi class case are depicted. The combination of reproducing kernels in support vector machines, based on the theory developed in the earlier sections, is the topic of section eight. The chapter closes with section nine, which shows common methods of producing probability output given (multi class) support vector classifier output. It is an indispensable step when progressing from phoneme classification to continuous speech recognition, as current recognizers rely on probabilistic output data such as generated for instance by HMMs or Gaussian Mixture Models.

Chapter two focuses on the more speech related aspects. Starting with a motivation for the MFCC features by brief summary of a typical model for the human vocal tract in section one, the chapter continues with a detailed explanation of the MFCC features and filterbank in the subsequent, second section. Section three reviews the theory of Hidden Markov Models followed by section four, which gives a short summary of simple acoustic and language models used in current speech recognition systems. Section five is the main section in this chapter. Evolving around the combination of the correlation operation and the specific kernel it gives a detailed description both the construction and topology of the classifier network and the individual steps necessary to train them. In this section, all aspects of our new approach cumulate. The final section of this chapter offers a short discussion of the impact of imbalanced training class sizes, that also affects our research.

Finally, chapter three illustrates and compares several experiments and their setups. One focus lies on the comparison of our new approach with common features and speech recognition setups. Detailed (confusion) tables and figures in appendices A to C give deeper insight. Section six elucidates the results compared to standard ASR systems and analyses potential reasons for disadvantageous influences. Finally, chapter three closes with conclusions and prospects.

The following figure 1 depicts interactions and relations of the three main areas and helps understanding the main path of the thesis when pursuing its path from top to bottom. Let us remark that the partitioning, given in the figure, into the *main areas* is clearly not reflecting the fact that several topics belong to more than one main area. For instance, Kernel methods imply a mathematical background to a certain degree. However, the main intention of the scheme is to clarify, how different ideas are related and how the main course from top to bottom utilizes the relations and combines their (partial) results.

Overview: Sequence diagram

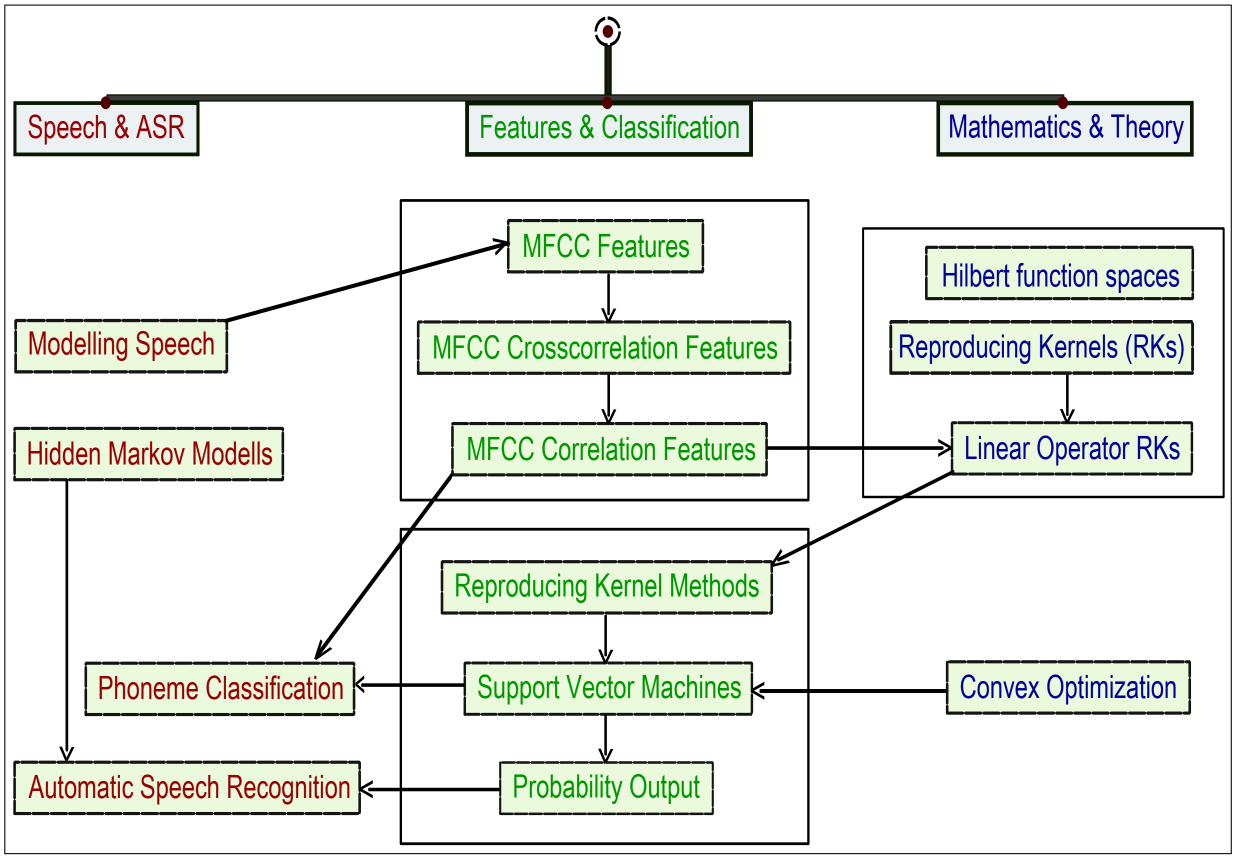
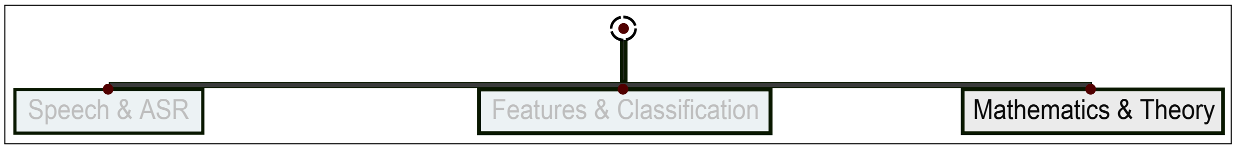


Figure 1: Overview of the thesis and important relations between the diverse parts.

Chapter 1 – Mathematical background



Overview

In the first section of this chapter we give a very quick and condensed review of the most important concepts of measure theory and Lebesgue integration to an extend necessary to understand the subsequent theorems on Hilbert function spaces and to understand the formulation of an important theorem of Mercer, which will be given in section 1.3. The theorem is a bridge between mathematical theory and applied pattern recognition as it states that reproducing kernels can easily be constructed using Eigenvalues of the underlying integral operator.

Reproducing kernels (RKs) and reproducing kernel Hilbert spaces will be presented afterwards. We start by giving basic definitions and results including their respective mathematical backgrounds such as algebraic properties useful for the construction of (new) kernels. The subsequent section portrays some examples of reproducing kernels themselves, the reproducing property in specific and finally demonstrates their use by applying the concepts to a well known axis transform and dimensionality reduction method: principal component analysis (PCA).

Section 1.5 concentrates on the aspects of RKs necessary for the theoretical results of our research. We present a way of integrating linear transforms and furtheron linear operators into reproducing kernels. The greater goal of this will become clear later, in section 2.5.3, where the approach of section 1.5 is combined with a new kind of feature, the latter being motivated by certain characteristics of short time speech intervals stemming from assumed stationarity: correlation. As already mentioned in the introduction, decent and meaningful results depend on the data as well as on an appropriate representation.

Section 1.6 reviews the main ideas of convex optimization, which are necessary to understand support vector (SV) classification, also called classification by support vector machines. The latter, being the (training and) classification method of our choice, is briefly portrayed in section 1.7. Support vector machines (SVMs) have become very popular within the last decades, mostly due to the fact that reproducing kernels have found their way into the SVMs' *dual* form, allowing for addressing non-linear problems.

Whereas the first main goal of this work, as outlined above, is the development of speech specific features and an associate mathematical space in the context of support vector classification and reproducing kernel methods, the next greater goal is the fusion of the aforementioned discriminative methods with and their integration into common speech recognition tools. Such tools make decisions using methods as Hidden Markov Models (see chapter 2, section 2.3) that rely on statistical data. Therefore it is indispensible to find a way of transforming the discriminative support vector classification results into appropriate probability data. To that effect section 1.9 presents methods developed for exactly this purpose: to generate probabilistic output based on support vector classification results.

Representing objects of one space in a different Hilbert function space comes along with a change of basis. This change of representation can be compared to the well known Fourier transform, where a function in time is transformed into a frequency based space[1]. The new function space often offers different advantages such as for instance a new dimension which allows either for a more compact data representation or guarantees linear separability for data not linearly separable in the original space. In the latter case, the reproducing kernel space is often of higher dimension than the original data space. In the case of Fourier transforms, the quick computation of convolutions in the original space is a very famous and frequently used characteristic. As many methods relying on metric relations make use of inner products or can be rephrased/ transformed to doing so, reproducing kernel methods can be applied to a huge collection of algorithms. Principal component analysis, data centering, data clustering and linear discriminant analysis are just a few to be mentioned.

[1]In fact, the Fourier transform is a special case of a reproducing kernel transform, which will be discussed in example 1.39 of section 1.4.3

1.1 Revisiting measure theory

Common one dimensional Riemannian integrals over bounded intervals $[a,b]$, $b > a$ are defined via Riemannian sums: Given a partitioning $a, a+\zeta, a+2\zeta, \ldots, a+L\zeta = b$ of the interval, where $\zeta = (b-a)/L$, the integral of a function $f : [a,b] \to \mathbb{R}$ is defined as the limit

$$\int_a^b f(x)dx = \lim_{L\to\infty} \sum_{z=0}^{L} f(x_z)(a_{(z+1)\zeta} - a_{z\zeta})$$

of a step function f evaluated at $x_z \in [a_{z\zeta}, a_{(z+1)\zeta}]$ arbitrary. If the limit exists and does not depend on the choice of the partition and the chosen x_z, f is said to be Riemann integrable over $[a,b]$.

One major problem arises when dealing with limits of sequences of integrable functions, when the latter themselves are integrable but this characteristic does not carry over to the limit. Lebesgue integration addresses this problem to a great extend by defining the integral over the image of a function rather than the domain: The image is, as before the domain, partitioned into small intervals and the limit of their lengths towards zero defines the integral. Given such a subinterval I_ζ, the preimage $\mathcal{D}_f\{I_\zeta\}$ is valued by a *measure* $\mu\left(\mathcal{D}_f\{I_\zeta\}\right)$ meeting certain requirements which are given in the definitions below. The Lebesgue-integral is then defined in an analogous manner, the ideas of which we will briefly depict well enough to understand its application within the main theorem of this subsection.

While at a first glance this might look trivial, a convincing formalization is far more complicated than it seems; a well known example showing one potential pitfall is the *Banach-Tarski paradox*. For more details, we refer the reader to [5], pages $3-6$. We reduce the material in this section to the few definitions and examples necessary for our purpose of formulating and comprehending the theorem of Mercer.

Definition 1.1
Let Ω be a set and $\mathcal{A}$ be a family of subsets of Ω. We call $\mathcal{A}$ a σ-**Algebra** if

(1) $\Omega \in \mathcal{A}$

(2) $A \in \mathcal{A} \Rightarrow A^c = \Omega \setminus A \in \mathcal{A}$ (complement)

(3) $A_i \in \mathcal{A}, i \in \mathbb{N} \Rightarrow \bigcup_i A_i \in \mathcal{A}$. (finite union)

■

Example 1.2 (Powerset)
To give a simple example consider any arbitrary $\Omega \subseteq \mathbb{N}$. Defining $\mathcal{A}$ as the set of all subsets of Ω, it is easily verified that $\mathcal{A}$ is a σ-Algebra. The set of all subsets is called **powerset**.

–●

Definition 1.3
A tuple $(\Omega, \mathcal{A})$, where $\mathcal{A}$ is a σ-Algebra over Ω, is called **measure space**. Given two measure spaces $(\Omega_1, \mathcal{A}_1)$ and $(\Omega_2, \mathcal{A}_2)$, a function $f : \Omega_1 \to \Omega_2$ is called **measurable** (or more specific $\mathcal{A}_1 - \mathcal{A}_2$–**measurable**), if $f^{-1}(\mathcal{A}_2) \subset \mathcal{A}_1$.

Looking back at the introduction of this section, the definition of measurability is very intuitive: if the pre-image $f^{-1}(\mathcal{A}_2)$ of the range in question (here the σ-Algebra $\mathcal{A}_2$) is a subset of the domain (here the σ-Algebra $\mathcal{A}_1$), the mapping is called measurable. In other words, each potential element of $\mathcal{A}_2$ can be *measured* w.r.t. the function's domain. What lacks is a way of measuring the function, of giving some amount of volume in the widest sense. This is covered by

Definition 1.4
A function $\mu : \mathcal{A} \to \mathbb{R}$ is called **measure** if

(1) $\forall A \in \mathcal{A} :\ \mu(A) \geq 0$

(2) $\mu(0) = 0$

(3) $\forall j, k \in \mathbb{N}, j \neq k, A_j \cap A_k = \emptyset : \mu\left(\bigcup_{i \in \mathbb{N}} A_i\right) = \sum_{i \in \mathbb{N}} \mu(A_i)$.

A measure μ on a family $\mathcal{A}$ of sets is called **strictly positive**, if for each $A \in \mathcal{A}$ we have $\mu(A) > 0$.

Let us look at the simple and detailed

Example 1.5
For a probability space we consider outcomes $\omega \in \Omega$ of a random process. Take, for instance, the trivial act of throwing a four sided pyramid-shaped dice, numbering the sides from 1 to 4. Possible outcomes shall be the number of the side the dice lands on and we devote the analysis to an experiment where we are interested in two specific unions of outcomes, $A = \{1, 2\}$ and $A^c = \{3, 4\}$ respectively. We let $\Omega = \{\{1, 2\}, \{3, 4\}\}$ be the union the two sets.

For the consideration of the σ-Algebra, we make the same choice as in example (1.2) and define $\mathcal{A}$ as the power set over A – in this case the set of all possible outcomes of throwing the dice. This choice guarantees that $\mathcal{A}$ is a σ-Algebra, as can easily be verified: As each potential subset of A is en element of $\mathcal{A}$, clearly all elements (the experiment's possible outcomes) of Ω are elements of $\mathcal{A}$. Second, $\mathcal{A} \ni A = \{1,2\}$ implies $\mathcal{A} \ni A^c = \{3,4\}$ and vice versa, proving (2). Clearly $\{\{1,2\} \cup \{3,4\}\} \in \mathcal{A}$.

Note also, that for any choice of the σ-Algebra two specific subsets are inevitably elements in $\mathcal{A}$ due to definition (1.1): $\{1,2,3,4\}$, which is called *certain event* and $\emptyset$, the *impossible event*, which in this example might for instance be the dice vanishing, not landing or coming to rest on one of its edges. Finally, demanding $\mu(\Omega) = 1$ is the last condition that enables us to define the (discrete) probability space $(\Omega, \mathcal{A}, \mu)$ for this example.

–●

As promised above, we will now be able to define the Lebesgue-integral similar to the Riemann-integral:

Definition 1.6 (Lebesgue Integral)
Given a measure space $(\Omega, \mathcal{A}, \mu)$ and a measurable real valued step function

$$f : \Omega \to \mathbb{R},\ f = \sum_{l=0}^{L} \alpha_l \chi_{A_l},$$

where $A_l \in \mathcal{A}$, $\alpha_i \geq 0$ for $1 \leq l \leq L$ and χ_{A_l} is the characteristic function

$$\chi_{A_l} : A_l \to \{0,1\}, a \mapsto \begin{cases} 1 & a \in A_l \\ 0 & a \notin A_l \end{cases}.$$

Then

$$\int_\Omega f d\mu = \sum_{l=0}^{L} \alpha_l \mu(A_l)$$

is the **μ-integral** or **Lebesgue integral** of f over Ω. The definition of the integral does not depend on the representation of f.

Definition 1.7
A characteristic of a function $f \in (\Omega, \mathcal{A}, \mu)$ is set to hold **μ-almost everywhere**, if it holds everywhere but on a set $A \subset \mathcal{A}$ such that $\mu(A) = 0$, i.o.w if it holds everywhere except on a μ-null set.

We have presented the idea of the Lebesgue integral and the underlying measure theory to a degree sufficient for our purposes. For more details and a thorough presentation of material including for instance the remaining steps from the integral definition over monotonic limits of nonnegative and finally all measurable functions, we refer the reader to [5] or any other book on measure- and integration theory. The last definition of this section briefly introduces Borel sets and Borel measures, needed as a further premise for theorem (1.28).

Definition 1.8 (Borel σ-Algebra and Borel measure)
Given a metric or topological space X and a system $\mathcal{O}$ of open subsets of X, the σ-Algebra $\sigma(\mathcal{O})$ spanned by $\mathcal{O}$ is called **Borel-σ-Algebra** or **σ-Algebra of Borel subsets of** X or, shorter, the **σ-Algebra of Borel sets** when X is clear from the context. A measure λ defined on such a $\sigma(\mathcal{O})$ is called **Borel measure**.

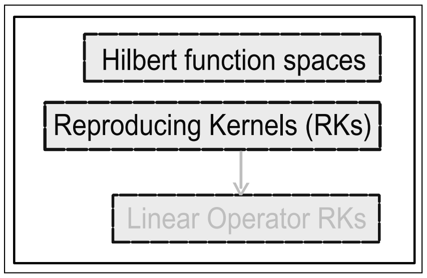
Example 1.9
For $X = \mathbb{R}^N$ and the common product topology

$$\lambda\left([a_1, b_1] \times \cdots \times [a_N, b_N]\right) = (b_N - a_N) \cdot \ldots \cdot (b_1 - a_1),$$

where $N \in \mathbb{N}$ and w.l.o.g. $a_n < b_n$ for $a_n, b_n \in \mathbb{R}$ and $1 \leq n \leq N$, λ is a Borel measure.

–●

1.2 Reproducing Kernels and Hilbert Function spaces



Reproducing kernels in their most general form are operators with specific characteristics between inner product spaces. They can be defined in both vector- and operator-valued spaces, but after a general definition we focus on the complex and later on the real valued case. On the basis of Hilbert spaces (the definition will follow below) we demonstrate that reproducing kernel Hilbert spaces (RKHS) are so called function spaces characterized by the property, that the point evaluations of their functionals are continuous. The *reproducing* property manifests itself in the fact that the reproducing kernel, uniquely determining an associated RKHS and vice versa, *reproduces* the continuous evaluation in a certain manner.

Kernels appear for instance in the special context of integral operators or, more general, in that of integral equations. In this light, Mercer's Theorem, in the form presented in section 1.3, is a bridge connecting the theoretical, integral operator based form of a kernel and a finite, discrete representation usable in applications .

Definition 1.10
A **Hilbert space** $\mathcal{X}$ is a normed vector space over a field $\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = \mathbb{C}$ endowed with an inner product $\langle \cdot, \cdot \rangle : \mathcal{X} \times \mathcal{X} \to \mathbb{K}$ that induces the norm $||x|| = \sqrt{\langle x, x \rangle}$ of $\mathcal{X}$ given $x \in \mathcal{X}$. Furthermore, $\mathcal{X}$ is complete w.r.t. to the same norm, that is every Cauchy Sequence of elements of $\mathcal{X}$ has a unique limit which is also an element of $\mathcal{X}$. Omitting completeness, $\mathcal{X}$ is called **pre-Hilbert space**.

Example 1.11
Simple examples are the euclidean reel and Hermitean complex spaces $\mathbb{K} = \mathbb{R}^n, \mathbb{C}^n$, $n \in \mathbb{N}$. A more complex example is the space of $\ell^2(\mathbb{K})$ of square summable sequences $(x_1, x_2, \ldots)$. *Square summable* means that $\sum_{m=0}^{\infty} |x_m|^2 < \infty$. For $\mathbb{K} = \mathbb{C}$ the inner product for two sequences $x, y \in \ell^2(\mathbb{K})$ is given by

$$\langle x, y \rangle = \sum_{m=0}^{\infty} x_m \overline{y_m}. \tag{1.1}$$

We will omit the easy proofs that equation (1.1) is well-defined and meets the requirements for an inner product as well as the proof that the space with the norm induced by the inner product

$$||x|| = \langle x, x \rangle^{\frac{1}{2}},$$

is complete. They are part of any book on introductory calculus or complex analysis.

–●

Definition 1.12
A **functional** is a mapping from a vector space V into its underlying scalar field $\mathbb{K}$.

In other words, a function space is simply what its name states: a $\mathbb{K}$-vector space the elements of which are functions. The functions thus take the role of ordinary points in spaces such as the common $\mathbb{R}^n$.

Definition 1.13
Given a set $X \neq \emptyset$ and a Hilbert Space $\mathcal{X}$ over X, the set of functionals on X with values in $\mathcal{X}$, $\mathcal{H} \subset \{\tau : X \to \mathcal{X}\} =: \mathcal{X}^X$, is called **Hilbert function space** if for all functionals

$f \in \mathcal{H}$ and for all $\nu \in X$ the **evaluation** δ_ν as follows is continuous:

$$\begin{aligned} \delta_\nu \ &: \ \mathcal{H} \to L(\mathcal{X}) \\ & \quad f \mapsto f(\nu), \end{aligned}$$

were $L(\mathcal{X})$ denotes the **linear space** (see remark (1.18) for details) of functionals in $\mathcal{X}^X$.

To countervail potential confusions of common spaces such as $\mathbb{R}^n$ and $\mathbb{C}^n$, $n \in \mathbb{N}$, function spaces and Hilbert function spaces we will give some more examples and point out differences. The basic concepts – vector spaces defined over a scalar field along with their characteristics – remain the same for either. We have already seen examples of Hilbert spaces and know that Hilbert function spaces are spaces of functions with images in the scalar field the function vector space is defined over.

However, for function spaces we illuminate the situation in more detail by means of the next example (1.14), introducing so called L^p-spaces. The key characteristic for function spaces, as the name readily implies, is the fact that the elements of function spaces are functions. Hilbert function spaces by definition fall into that category, given two additional qualities: An inner product, being a typical structuring element of any Hilbert spaces, and an evaluation functional according to definition 1.13.

Example 1.14 (L^p function spaces)
Given a measure space $(\Omega, \mathcal{A}, \mu)$, for $1 \le p \le \infty$, $p \in \mathbb{N}$ the $\mathcal{L}^p := \mathcal{L}(\Omega, \mathcal{A}, \mu)$ function space is comprised of all functions f such that the norm satisfies

$$||f||_p = \left(\int_\Omega |f|^p \right)^{\frac{1}{p}} < \infty. \tag{1.2}$$

As for any normed vector space, the set of such so called p-integrable functions needs to meet several requirements in order to form a $||\cdot||_p$-normed vector space of functions:

- Defining $(f+g)(x) = f(x)+g(x)$ and $(\alpha f)(x) = \alpha(f(x))$ for $f, g \in \mathcal{L}^p, \alpha \in \mathbb{K}$ ensures that addition and scalar multiplication are well-defined.
- As $||f||_p$ can be zero for functions $f = 0$ almost everywhere, $\mathcal{L}^p$ is clearly not a normed vector space. This "imperfection" is eluded by considering the quotient space $L^p = \mathcal{L}^p / \mathcal{N}_{\mathcal{L}^p}$, where[2] $\mathcal{N}_{\mathcal{L}^p} = \{f \in \mathcal{L}^p : f = 0 \ \mu \text{ - almost everywhere}\}$.

[2]We refer the interested reader to any standard book on measure theory such as ([5]) for the proof that

The quotient space L^p is a space of equivalence classes $[\cdot]$. Let us denote the equivalence class of f by $[f]$. Then, given any $[f] \in L^p$ and a representative element g of $[f]$, the definition of a quotient space implies $f \equiv g$ μ - almost everywhere, that is $(f - g) \in \mathcal{N}_{\mathcal{L}^p}$. It is clear that $||f||_p = 0 \Leftrightarrow f \in \mathcal{N}_{\mathcal{L}^p}$ and thus we can consider the norm "on" the equivalence class instead of it's representers. This shows that

$$|||\cdot|||_p : L^p \to \mathbb{R}_+ \cup \infty, \qquad [f] \mapsto ||f||_p$$

is well-defined and satisfies

$$|||[f]|||_p = ||f||_p = 0 \Leftrightarrow [f] = 0 \Leftrightarrow f \equiv 0 \ \mu \text{ - almost everywhere}$$

for $[f] \in L^p$.

The meaning of this is that functions that are identical $\mu-$almost everywhere are interpreted as being basically the same and thus are elements of one and the same equivalence class. Each of the equivalence classes elements is an equipollent representative. Experience shows that the (topological) differences (i.e. the equalization of potentially different limits in $\mathcal{L}^p$ by forming quotient spaces by sets of measure 0 and along with this the usage of equivalence classes instead of functions) are small enough to be neglected w.r.t. the underlying measure space.

–●

Example 1.15
Any norm $||\cdot||_p$, $1 \leq p < \infty$, defined on the respective L^p-spaces given above is a functional $L^p \to \mathbb{R}$.

–●

Definition 1.16
Let $\mathcal{H} \subset \mathcal{X}^X$ be a Hilbert Space of bounded functionals endowed with an inner product $\langle \cdot, \cdot \rangle$. A mapping $k : X \times X \to L(\mathcal{X})$ is called **reproducing kernel** if for all $\zeta \in \mathcal{X}$ and $\nu \in X$:

(i) $k(\cdot, \nu)\zeta \in \mathcal{H}$

(ii) $\forall f \in \mathcal{H} : \langle f(\nu), \zeta \rangle = \langle f, k(\cdot, \nu)\zeta \rangle$

Keep in mind that, as k maps into $L(\mathcal{X})$, property 1.16 (i) implies $k(\cdot, \nu) \in \mathcal{H}$. A closer look at the second property of those two definitions clarifies its meaning: for a kernel

$\mathcal{N}_{\mathcal{L}^p}$ is a sub vector space of $\mathcal{L}^p$ and the quotient space is thus a well-defined vector space over $\mathbb{K}$.

$k(\cdot,\nu)$ given as the right term of the inner product, for fixed $\nu \in X$, the inner product $\langle f, k(\cdot,\nu)\rangle$ equals the evaluation of the mapping given as a left term of the equation, here f, at ν. The kernel *reproduces* the evaluation. For this reason this characteristic is often called **reproducing property**.

As such a kernel function is itself an element of $\mathcal{H}$, we note

$$k(\mu,\nu) = \langle k(\cdot,\mu), k(\cdot,\nu)\rangle \;\; \forall \mu,\nu \in X. \tag{1.3}$$

Example 1.17
The most simple example of a reproducing kernel is a constant map $k(x,z) = c$, $c \in \mathbb{R}_+$. More elaborate examples will be given in section 1.4, after having dealt with the necessary theoretical background.

Remark 1.18
Let us have a quick glance at the concept of linear spaces, which are the foundation of function spaces. A linear space is basically a vector space V which is homogeneous and linear. In other words, for any $x, y \in V$ and any scalar $\alpha, \beta \in \mathbb{K}$ and operations $+$, $\cdot$, the following conditions hold:

- $(\alpha \cdot \beta)x = \alpha(\beta \cdot x)$
- $\alpha \cdot (x+y) = \alpha \cdot x + \alpha \cdot y$

Both requirements are valid for $\mathbb{C}$, hence the vector space of complex numbers is isomorphic to its linear space by identifying the arithmetic operations on $\mathbb{C}$ with their counterparts on $L(\mathbb{C})$.

Note that thereby it becomes clear that in the context of definition 1.13 a Hilbert function spaces $\mathcal{H}$ inherits its linearity from $\mathcal{X}^X$!

Example: $\ell^2(\mathbb{K})$, introduced in example (1.11), is a linear space. Homogeneity is verified easily, as scaling is an invariant w.r.t. square summability. Applying the triangle inequality can be used to proof the Minkowsky inequality, which leads to additivity and finalizes the proof that $\ell^2(\mathbb{K})$ is a linear space.

For the remainder of this section, we will focus on $\mathcal{X} = \mathbb{C}$ and make use of the fact that $L(\mathbb{C}) \simeq \mathbb{C}$, see remark 1.18. In this case definition (1.16) reduces to

Definition 1.19
Let $\mathcal{H}$ be a Hilbert Space of complex valued bounded functionals endowed with an inner product $\langle \cdot, \cdot \rangle$. A mapping $k : X \times X \to \mathbb{C}$ is called **reproducing kernel** if for all $\nu \in X$:

(i) $k(\cdot, \nu) \in \mathcal{H}$

(ii) $\forall f \in \mathcal{H} : \; f(\nu) = \langle f, k(\cdot, \nu) \rangle$

As a first result we show that Hilbert function spaces and reproducing Kernels in Hilbert spaces are equivalent concepts:

Theorem 1.20
Let $\mathcal{H}$ be a Hilbert space on X. The following properties are equivalent:

(i) $\mathcal{H}$ is a Hilbert function space

(ii) $\mathcal{H}$ has a reproducing kernel

The reproducing kernel is unique.

Proof. *ii* $\Rightarrow$ *i*) Given a reproducing kernel k, 1.19 (ii) guarantees the linearity of the evaluation functional δ_ν, see definition 1.13. Now,

$$\begin{aligned} |\delta_\nu(f)| = |f(\nu)| &= |\langle f, k(\cdot, \nu) \rangle| \le ||f|| \cdot ||k(\cdot, \nu)|| \\ &= ||f|| \, \langle k(\cdot, \nu), k(\cdot, \nu) \rangle^{1/2} = ||f|| \, k(\nu, \nu)^{1/2}, \end{aligned} \tag{1.4}$$

hence f is continuous in all $\nu \in X$ and thereby $\mathcal{H}$ a Hilbert function space. The first equality makes use of the reproducing property, number two is the Cauchy-Schwarz inequality and equality three holds as in a Hilbert space the norm is induced by the space's inner product. The very last equation again uses the reproducing property using identity 1.3.

i $\Rightarrow$ *ii*) Given $\lambda' \in \mathcal{H}'$, the Fréchet-Riesz Lemma proves the existence and uniqueness of a function $k(x, \nu) \in \mathcal{H}$ satisfying $\lambda'(x) = (f\nu)(x) = \langle f(x), k(x, \nu) \rangle \, \forall x \in \mathcal{H}$. Here $k(\cdot, \nu)$ is the reproducing kernel.

Finally, given two kernels $\tilde{k}(\cdot,\nu)$ and $k(\cdot,\nu)$, equations 1.3 and 1.7 yield

$$k(\mu,\nu) = \left\langle k(\cdot,\mu), \tilde{k}(\cdot,\nu) \right\rangle = \overline{\left\langle \tilde{k}(\cdot,\nu), k(\cdot,\mu) \right\rangle} = \overline{\tilde{k}(\nu,\mu)} = \tilde{k}(\mu,\nu),$$

which proves the uniqueness of the reproducing kernel.

□

The operator norm and equation 1.4 give an idea of what the norm of the evaluation functional looks like:

$$||\delta_\nu|| = \sup_{||f||\neq 0} \frac{|\delta_\nu(f)|}{||f||} = \frac{||f|| \cdot ||k(\cdot,\nu)||}{||f||} = k(\nu,\nu)^{1/2} \tag{1.5}$$

Equality holds as the operator norm implies $\leq$ and as the supremum itself is achieved when the functional is the kernel itself, again with fixed ν.

Let us stop here for a moment to get insight into the mapping defined by such a kernel by means of an example we will encounter frequently in this work, the Gaussian kernel (a special case of an rbf-kernel, see below) $k_{rbf}(y,x) : x \mapsto \exp\left(-\frac{||x-y||^2}{2\gamma^2}\right)$ for x fix. Figure 1.1 illustrates the kernel mapping, which maps single points into a space of Gaussian functions.

We continue by looking at reproducing kernels from a different but equivalent viewpoint:

Definition 1.21
A function $k : X \times X \to \mathbb{C}$ is called **positive semi-definite**, if for all $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_n) \in \mathbb{C}^n$ and all $\boldsymbol{\nu} = (\nu_1,\ldots,\nu_n) \in X^n$ the following inequality holds:

$$\overline{\boldsymbol{\alpha}} k \boldsymbol{\alpha} = \sum_{i,j=1}^{n} \overline{\alpha_i}\alpha_j k(\nu_i,\nu_j) \geq 0 \tag{1.6}$$

If the inequality is proper, k is called **positive definite**.

Every positive (semi)definite (psd) function basically is a RK and vice versa. This leads further to the matrix form representation of psd functions, which in the finite dimensional case is often called **Gram matrix** or **Gramian**. Details follow after some important characteristics of reproducing kernels, such as

Theorem 1.22
Reproducing kernels are positive semi-definite and satisfy

$$k(\mu,\nu) = \overline{k(\nu,\mu)} \quad \forall \mu,\nu \in X. \tag{1.7}$$

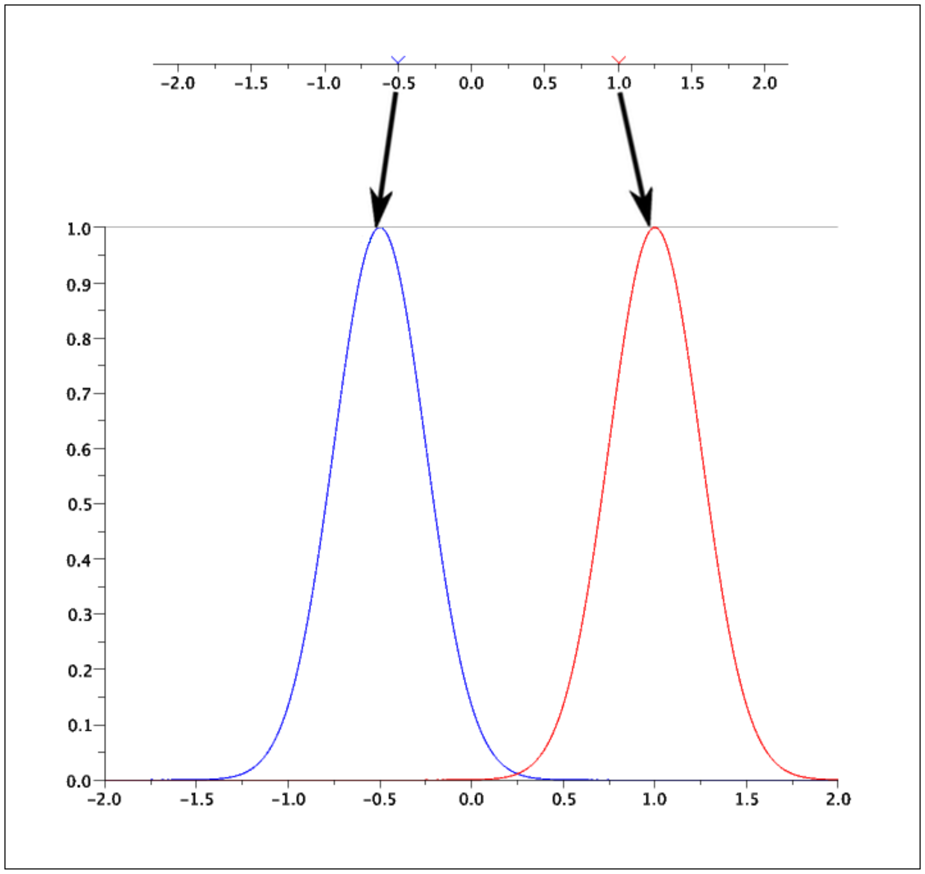


Figure 1.1: The kernel maps single points to functionals. The figure illustrates the process for two points $x_1 = -0.5, x_2 = 1$ and the Gaussian rbf-kernel $k_{rbf}(y, x_i) = \exp\left(-\frac{||x_i - y||^2}{2\gamma^2}\right)$, $i = 1, 2$ with parameter $\gamma = 0.75$.

Proof.

$$\begin{aligned}
\sum_{i,j=1}^{n} \overline{\alpha}_i \alpha_j k(\nu_i, \nu_j) &= \sum_{i,j=1}^{n} \overline{\alpha}_i \alpha_j \langle k(\cdot, \nu_i), k(\cdot, \nu_j) \rangle \\
&= \left\langle \sum_{j=1}^{n} \alpha_j k(\cdot, \nu_j), \sum_{i=1}^{n} \alpha_i k(\cdot, \nu_i) \right\rangle \\
&= \left|\left| \sum_{j=1}^{n} \alpha_j k(\cdot, \nu_j) \right|\right|^2 \geq 0, \qquad (1.8)
\end{aligned}$$

which proves the first claim. The second property holds because of

$$\langle k(\cdot, \nu), k(\cdot, \mu) \rangle = k(\mu, \nu) \text{ and } \langle k(\cdot, \nu), k(\cdot, \mu) \rangle = \overline{\langle k(\cdot, \mu), k(\cdot, \nu) \rangle}. \qquad (1.9)$$

$\square$

Vice versa, given a positive semi-definite mapping $k : X \times X \to \mathbb{C}$, we can interpret the latter as a reproducing kernel and construct a Hilbert function space upon it. The following important theorem will elaborate this link and show that there is a bijective relation between those objects:

Theorem 1.23
Given a positive semi-definite function $k : X \times X \to \mathbb{C}$, there exists exactly one Hilbert function space $\mathcal{H} \subset \mathbb{C}$ with reproducing kernel k.

Proof. The proof will be sketched only; for details see [6] and [7]. We define H to be the linear span of the kernel function on all elements of X, $H = \operatorname{Span}\{k(\cdot,\nu) \,|\, \nu \in X\}$. Given that definition, consider two points of that space $f, g \in H$, $f = \sum_{i=1}^{n} \alpha_i k(\cdot,\nu_i)$ and $g = \sum_{j=1}^{n} \beta_j k(\cdot,\mu_j)$. We endow H with an inner product defined by

$$\langle f, g\rangle = \left\langle \sum_{i=1}^{n} \alpha_i k(\cdot,\nu_i), \sum_{j=1}^{n} \beta_j k(\cdot,\mu_j) \right\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \overline{\beta_j} k(\mu_j,\nu_i). \tag{1.10}$$

It can easily be verified that 1.10 represents a well-defined sesquilinear form (i.e. that there are no unequal representations $\sum_{i=1}^{n} \alpha_i k(\cdot,\nu_i)$ and $\sum_{j=1}^{n} \beta_j k(\cdot,\nu_j)$ of one and the same element of the constructed space) and is furthermore positive semi-definite given the positive definiteness of k, see equation 1.8.

For $\langle f, f\rangle = 0$, property 1.16 (ii) and the Cauchy-Schwarz inequality prove $f \equiv 0$. Hence 1.10 defines a positive definite inner product and induces a norm $||f|| = \langle f, f\rangle^{\frac{1}{2}}$ in H, which altogether makes H a pre-Hilbert space. By definition,

$$\begin{aligned} \langle f, k(\cdot,\nu)\rangle &= \sum_{i=1}^{n} \alpha_i k(\nu,\mu_i) && (1.11)\\ &= f(\nu) && (1.12) \end{aligned}$$

and using the Cauchy-Schwarz inequality to bound $||\delta_\nu f||$ we see that the evaluation δ_ν is continuous. Finally, let $(\mathcal{H}, \langle\cdot,\cdot\rangle)$ denote the completion of H. It can then be shown that the elements of this complete Hilbert space can now be identified[3] with functions given as linear equations of k as above, where k is **the** reproducing kernel of $\mathcal{H}$. Theorem 1.20 ergo implies that $\mathcal{H}$ is a Hilbert function space.

[3] *Identified* in this context means, that the space is closed w.r.t. the norm defined by the inner product. There is thus no difference between functions and their potential representations through limits of elements of the (not yet complete) space.

Now, given that $\mathcal{H}$ is a Hilbert function space, let $\{e_i(\cdot)\}$ be an orthonormal basis of complex valued functions on $\mathcal{H}$. As $k(\cdot,\nu) \in \mathcal{H}, \nu \in X$, it has a Fourier series representation $\sum_{i=0}^{\infty} \alpha_I e_i(\cdot)$ with coefficients $\alpha_i = \langle k(\cdot,\nu), e_i(\cdot)\rangle$. The sesquilinearity of the inner product in $\mathbb{C}$ implies $\alpha_i = \langle k(\cdot,\nu), e_i(\cdot)\rangle = \overline{\langle e_i(\cdot), k(\cdot,\nu)\rangle} = \overline{\alpha_i}$, thereby proving:

$$\begin{aligned} k(\cdot,\nu) &= \sum_i \langle k(\cdot,\nu), e_i(\cdot)\rangle\, e_i(\cdot) = \sum_i \overline{\langle e_i(\cdot), k(\cdot\nu)\rangle} e_i(\cdot) & (1.13)\\ &= \sum_i \overline{e_i(\nu)} e_i(\cdot) & (1.14)\\ &\text{or} & \\ k(\mu,\nu) &= \sum_i \overline{e_i(\nu)} e_i(\mu) & (1.15)\end{aligned}$$

Along with this, the opposite direction of the last property extends naturally from ordinary Hilbert spaces to Hilbert function spaces. Given an orthonormal system $(e_i)_{i\in\mathbb{N}}$, the reproducing property and identity (1.15) in conjunction with each other prove

$$\Phi(\nu) = \langle \Phi, k(\cdot,\nu)\rangle = \sum_{i=0}^{\infty} e_i(\nu)\, \langle \Phi, e_i(\cdot)\rangle\,. \tag{1.16}$$

This holds for all $\Phi \in \mathcal{H}$ and for all $\nu \in X$ and we can thus generalize the last equation to

$$\Phi = \langle \Phi, k(\cdot,\nu)\rangle = \sum_{i=0}^{\infty} \langle \Phi, e_i(\cdot)\rangle\, e_i(\cdot), \tag{1.17}$$

which shows that the system is complete.

$\square$

Remark 1.24
The last derivation was made under the tacit assumption that – even in an infinite dimensional Hilbert function space – a countable set of linear combinations of (kernel)functions suffices and is capable of approximating any given functional in the considered space. The motivation for this lies in the fact that in any separable metric space $\mathcal{S}$, any subset dense in $\mathcal{H}$ contains itself a countable subset, which again is dense in $\mathcal{S}$. The proof for this can be found in most books on functional analysis such as [8]. In the case of a Hilbert function space space, any such countable set is comprised of linear combinations of kernels $k(\cdot,\nu)$, as the latter span the space.

Combining the result of theorem 1.23 with the reproducing property in the form of equation (1.3), in the next theorem we define a mapping from an element in the domain of a reproducing kernel to a function $x \mapsto k(\cdot, x)$ that relates the RK to another space, which may for instance be a feature space in the context of a pattern recognition task:

Corollary 1.25 (factorization corollary)
Given a psd function $k : X \times X \to \mathbb{C}$, there exist a Hilbert space $\mathcal{H}_k$ and a function $\Phi : X \to \mathcal{H}_k$ such that

$$k(\mu, \nu) = \langle k(\cdot, \mu), k(\cdot, \nu)\rangle = \langle \Phi(\mu), \Phi(\nu)\rangle \tag{1.18}$$

for allmost all $\nu, \mu \in X$. The index k denotes the relation between the matrix and the underlying hermitian form and we will skip it in this work from now onwards when there is no direct connection to a kernel.

As for any hermitean form we can define the matrix

$$G^k = \big(\langle \Phi(\nu_i), \Phi(\nu_j)\rangle\big)_{i,j} = \big(k(\nu_i, \nu_j)\big)_{i,j} \tag{1.19}$$

$$= \begin{pmatrix} k(\nu_1,\nu_1) & \cdots & k(\nu_1,\nu_n) \\ \vdots & \ddots & \vdots \\ k(\nu_n,\nu_1) & \cdots & k(\nu_n,\nu_n) \end{pmatrix} \tag{1.20}$$

for $1 \leq i, j \leq n$, which by equation (1.8) is positive semi-definite in the sense that $\forall \mathbf{x} \in X : \mathbf{x}^H G^k \mathbf{x} \geq 0$.

G^k is called **Gram matrix** or **Gramian** and by definition completely characterizes the kernel. As a matrix it can be used and implemented efficiently as a linear map into the Hilbert function space associated with the kernel k.
Let us look at a generalization of example 1.11 in more details to further illuminate and clarify definitions 1.13, 1.16 and 1.19.

Example 1.26
Consider a nonemtpy set M and define

$$l^2(M) = \{f : M \to \mathbb{C}, \; ||f||^2 = \sum_{m \in M} |f(m)|^2 < \infty.\}$$

First, by theorem 1.23 we conclude that the reproducing kernel for $l^2(M)$ is $K(m, n) = \delta_{mn}$, where δ is the Kronecker function. Being ordinary sums of products, the point evaluations are clearly continuous. Together, this makes $l^2(M)$ a reproducing kernel Hilbert space.

–●

Remark 1.27
Corollary 1.25 reflects one of the main concepts of the application of reproducing kernels within the field of pattern recognition, which is discussed to some extend necessary for our purposes in section 1.4.4: The Hilbert function space defined by such a reproducing kernel k can be interpreted as a new feature space based on a feature mapping Φ. Once again, the idea behind this approach is that all required metric information – that is information about distances or, in a wider sense, similarities – between features $\Phi(\mu), \Phi(\nu)$ of given data $\mu, \nu \in X$ can be computed by evaluating the reproducing kernel functional at (μ, ν).

1.3 The theorem of Mercer

The theorem of Mercer, which will be stated here in a rather general form, had major impact for machine learning and pattern recognition, offering a simple method for constructing reproducing kernels in practice. We will present it here without proof, but the interested reader can find theorem plus proof in similar form for instance in [9]. We mention, however, that the proof is in greater parts identical or similar to that of theorem 1.23.

Theorem 1.28 (Mercer)
Let V be a compact metric space endowed with a finite strictly positive Borel measure η, $k \in \mathcal{C}(V \times V)$ and $T_k : L^2_\eta(V) \to L^2_\eta(V)$ the integral operator

$$(T_k f)(\cdot) = \int_V k(\cdot, \nu) f(\nu) d\eta(\nu).$$

If T_k is positive definit in the $L^2_\eta(V)$-sense, that is

$$\int_V \int_V k(\mu, \nu) f(\nu) d\eta(\mu) d\eta(\nu) \geq 0, \; f \in L^2(V), \tag{1.21}$$

then equation (1.7) holds for all $\mu, \nu \in V$ (thus rendering T_k selfadjoint) and $k(\mu, \nu)$ can be represented as an absolutely and uniformly convergent series

$$k(\mu, \nu) = \sum_{i=1}^{\infty} \lambda_i \iota_i(\mu) \overline{\iota_i(\nu)} \quad \text{for almost all } \mu, \nu \in V, \tag{1.22}$$

where λ_i are the eigenvalues of T_k according to their geometric multiplicities and ι_i their respective orthogonal eigenfunctions.

Filling the Gap – What is so special about theorem 1.28?

Under certain, but in practice frequent conditions, it opens ways for a relatively easy construction of reproducing kernels:

- On the theoretical side, given rather non-strict preliminaries such as continuity of a kernel function k and the integrability in the L^2_η-sense of the associate transformation $T_k f$, the theorem guarantees the existence of a uniformly convergent series and furthermore also gives information on how it is constructed. Example 1.39 is, in this context, very interesting and illuminating. Here $f(\cdot)e^{-i\omega t}$ is such a kernel function $k(\cdot, t)$.

- Looking at equation (1.22) we realize that it equals the inner product in the sequence space given the mapping

$$\varphi : V \to \ell^2_\iota : x \mapsto \left(\sqrt{\lambda_i}\iota_i(x)\right)_{1\le i\le\infty} .$$

  The subscript ι illustrates the fact that the $\iota_i(\cdot)$, $1 \le i \le \infty$ compose the basis of the sequence space ℓ^2. We get the same result as in Corollary 1.25, thus once again closing the gap between the theoretical, operator based result and practical applications.

The next example illustrates the practicability of the Mercer theorem by interpreting the decomposition of a symmetric matrix A in its context:

Example 1.29
Consider a symmetric matrix $A \in \mathbb{R}^{N\times N}$, $A = (a_{m,n})_{1\le m,n\le N}$, nonnegative in the sense that $\mathbf{x}^T A\mathbf{y} \ge 0$ $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^N$. Let λ_i, $1 \le i \le N$ be the eigenvalues of A with associated eigenvectors ϕ_i, then A can be decomposed into the matrix product $A = \Phi\Lambda\Phi^T$. In the matrix product on the right side of the last equation Λ is the diagonal matrix with entries $\Lambda_{ii} = \lambda_i$ and Φ is the matrix composed of the corresponding eigenvectors. As A is nonnegative, $\lambda_i \ge 0$ and we can take the element wise square-root of the matrix.

–●

Example 1.30 (Finite dimensional Hilbert function spaces)
We will close this section with an example that illustrates the theory and clarifies its application in the finite dimensional case. In a manner similar to the second part of the proof of theorem 1.23, only finite dimensional this time, consider an N-dimensional Hilbert function space $\mathcal{H}$ and a basis $\{e_i(\cdot)\}_{i=1}^N$.

This setup implies

$$\forall f \in X \subset \mathcal{H} \,:\, f(\mu) = \sum_{i=1}^{N} \alpha_i e_i(\mu) \tag{1.23}$$

for all $\mu \in X$ with complex scalar coefficients α_i.

Subsequent to corollary 1.25 we have introduced the Gram matrix, the elements of which are the pairwise kernel evaluations. Hence given a reproducing kernel $k \in \mathcal{H}$ (we do not yet know what k looks like!), we have

$$G^k = \big(k\left(e_i(\cdot), e_j(\cdot)\right)\big)_{i,j}, \ 1 \le i, j \le N \tag{1.24}$$

We recall equation 1.22 as well as the fact that the reproducing property must hold for the kernel and look in detail at the coefficients α_i: in the context of equation 1.22 it becomes clear that we need to find elements $\zeta_{i,j}$ such that

$$\sum_{j=1}^{N} \alpha_{i,j}\overline{\zeta_{j,i}} = \delta_{i,j},$$

where

$$\delta_{i,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} .$$

$\zeta_{i,j} = \overline{\alpha_{i,j}^{-1}}$ does the job. Finally, by definition $\{e_i(\cdot)\}_{i=1}^{N}$ is an orthonormal basis and we conclude that

$$k(\mu, \nu) = \sum_{i,j=1}^{N} \zeta_{i,j} e_i(\mu) e_j(\nu)$$

meets all requirements and is **the** – due to the uniqueness of the RK defining its associated RKHS – sought reproducing kernel for the finite dimensional Hilbert function space $\mathcal{H}$.

–●

Remark 1.31
Naturally, classification tasks for computer systems are finite dimensional. This, however, is not necessarily the case any longer when working in the Hilbert function space determined by a reproducing kernel. The most famous (or infamous, depending of the point of view) examples for this are the exponential kernel k_e and the RBF-kernel k_{rbf} given in example 1.35, (3) and (4).

Theoretically, working in infinite dimensional spaces would burden us with all its problems and challenges compared to finite dimensional ones. The good news is that the functionals of most of the function spaces can be approximated arbitrarily exact by a countable (see remark 1.24) or even finite dimensional subsystem dense in the respective space. For the examples mentioned above, the Taylor series with n terms is such a subsystem, letting $n \to \infty$, as already mentioned in remark 1.24. Whenever this is not the case, care has to be taken.

1.4 Applications and Examples

Knowing the basics of Hilbert function spaces and reproducing kernels, we shed light on some of their properties. By showing that reproducing kernels are closed under a lot of common algebraic operations and making use of those, we will be able to derive some well known and frequently used reproducing kernels and common Hilbert function spaces. In the third subsection, our aim is to deduce the reproducing property for two specific Hilbert function spaces. The literature we studied did not include any such deduction, and in proving the reproducing property we were surprised and glad to realize that this process grants deeper insight into the spaces' structures and their functionals and operators. Clearly, if one does not yet happen to know the kernel of a space in question it is indispensable to first derive it. This subsection, subsequent to the examples of RKs, thus implicitly elucidates the Hilbert function space associated with the polynomial kernel of degree two and shows that the set of (bandlimited) Fourier transform is a reproducing Hilbert kernel space.

1.4.1 Kernel construction

We have seen that positive semi-definite matrices (or reproducing kernels) and reproducing kernel Hilbert spaces are equivalent concepts. In this section we give some examples of constructing reproducing kernels – and hence function spaces – by algebraic methods. A simple example immediately unveils itself through the fact that positive semi-definite matrices $M^{n\times n}, n \in \mathbb{N}$ are closed under addition: Let $k_1(\cdot,\cdot), k_2(\cdot,\cdot) \in X \times X$ be two reproducing kernels. Then $k_+(\cdot,\cdot) = k_1(\cdot,\cdot) + k_2(\cdot,\cdot)$ is a reproducing kernel. The next theorem gives further examples.

Theorem 1.32
Given a subset $S \subset X \in \mathbb{K}^n, n \in \mathbb{N}$, $\mathbf{x}, \mathbf{z} \in \mathbb{K}^n$, $0 \neq f : X \to \mathbb{K}$, $\psi : X \to \mathbb{K}^n$, $a \in \mathbb{K}$ and $A \in \mathbb{K}^{n\times n}$ a symmetric psd matrix, let $k, k_i \in X \times X, i \in \mathbb{N}$ be reproducing Kernels and $k_\psi \in X \times X$. Then

(1) $k_\oplus(\mathbf{x},\mathbf{z}) = \sum_{i=1}^n k_i(\mathbf{x},\mathbf{z})$

(2) $k_\otimes(\mathbf{x},\mathbf{z}) = \bigotimes_{i=1}^n k_i(\mathbf{x},\mathbf{z})$

(3) $k_{|S}(\mathbf{x},\mathbf{z}) = k(\mathbf{x},\mathbf{z})_{|S}$, the restriction of the kernel to the subset S.

(4) $k_*(\mathbf{x},\mathbf{z}) = ak(\mathbf{x},\mathbf{z})$

(5) $k_f(\mathbf{x},\mathbf{z}) = f(\mathbf{x})k(\mathbf{x},\mathbf{z})\overline{f(\mathbf{z})}$

(6) $k_A(\mathbf{x}, \mathbf{z}) = \mathbf{x}^H A \mathbf{z}$

(7) $k_\psi(\mathbf{x}, \mathbf{z}) = k(\psi(\mathbf{x}), \psi(\mathbf{z}))$

(8) $k_e = \exp k(\mathbf{x}, \mathbf{z})$

are reproducing kernels, where $\bigotimes_{i=1}^n$ denotes the product of kernels $k_i, i = 1, \dots n$.

Proof. We will proof (4) and (7), the latter being important in section 1.5.1. For (1) and (2) (restricted to two product terms), (3) and (5) the reader should consult [10]; (6) is proven in [9] but is also easily deduced as a special case of (5). Finally, the proof for (8) will be given in the next subsection, example 1.35 item (3).

(4) Referring to definition 1.21, the result of scaling $\boldsymbol{\alpha} \in \mathbb{K}^n$ by a constant scalar a is an element of $\mathbb{K}^n$ as well. The Gramian corresponding to the kernel k^ψ thereby remains psd.

(7) The proof is straight forward. As ψ is defined on X and maps to $\mathbb{K}^n$ we conclude that k_ψ is well-defined and a reproducing kernel.

□

Remark 1.33
As we do not make any use of kernels more sophisticated than the ones defined in this work, such as for instance RKs on graphs or trees, we refer the reader to [9] or [11] for both definitions and detailed analysis. The motivation for such kernels is no different from ours: Optimal RKs and their spaces' structures and metrics should to a certain degree reflect and represent structures and relations of both underlying data and methods used for its analysis.

1.4.2 Reproducing kernel examples

Example 1.34
Obviously, any constant $c \in \mathbb{R}_+$ is a reproducing Kernel on $\mathbb{C}_c$, the set of complex numbers endowed with the inner product $\langle x, z \rangle = \frac{1}{c} x \overline{z}$. This is a special case of example 1.30: As $c \in \mathbb{R}_+$, we have $\overline{c^{-1}} = c^{-1} = \frac{1}{c}$.

–●

In the context of numerical classification tasks – that is problems living in real or complex spaces $\mathbb{K}^n, n \in \mathbb{N}$ – the following kernels are amongst the most frequently used ones:

Example 1.35
Given a subset $X \subset \mathbb{K}^n$, variables c, n and $d \in \mathbb{N}$, a scalar $\sigma \in \mathbb{R}_+$, vectors $\mathbf{x}, \mathbf{z} \in X$ and a reproducing kernel $k \in X \times X$,

(1) the **linear kernel** $k_l(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle$ as a special case for $d = 1$ and $c = 0$ of

(2) the **polynomial kernel** $k_{p^d}(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + c)^d$

(3) the **exponential kernel** $k_e(\mathbf{x}, \mathbf{z}) = \exp\left(k(\mathbf{x}, \mathbf{z})\right)$

(4) the **exponential rbf kernel** $k_{rbf}(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{||(\mathbf{x}-\mathbf{z})||}{2\sigma}\right)$

are reproducing kernels.

$-\bullet$

Proof. (see also [9], page 77) (1) and (2) are obviously reproducing kernels due to the positive definiteness of the inner product of Hilbert spaces and theorem (1.32). The exponential function has a representation as a Taylor series – in other words, it can be approximated arbitrarily accurate by polynomials and is thus a reproducing kernel by theorem 1.32, (1) and (4). Finally, using the linear kernel scaled by $\frac{1}{\sigma^2}$ as the exponent in (3) and normalizing the kernel by $\frac{1}{\sqrt{\exp\left(||x||^2/\sigma^2\right)\exp\left(||z||^2/\sigma^2\right)}}$ results in expression (4):

$$\frac{\exp\left(\frac{\langle \mathbf{x},\mathbf{z}\rangle}{\sigma^2}\right)}{\sqrt{\exp\left(\frac{||x||^2}{\sigma^2}\right)\exp\left(\frac{||z||^2}{\sigma^2}\right)}} = \exp\left(-\left(\frac{\langle \mathbf{x},\mathbf{x}\rangle}{2\sigma^2} - \frac{\langle \mathbf{x},\mathbf{z}\rangle}{\sigma^2} + \frac{\langle \mathbf{z},\mathbf{z}\rangle}{2\sigma^2}\right)\right) = \exp\left(-\frac{||(\mathbf{x}-\mathbf{z})||}{2\sigma}\right),$$

where normalization is a well-defined operation by theorem 1.32, part (4).

□

Remark 1.36
Both the polynomial and the exponential rbf kernel given above are special cases. The first one can be generalized to

$$k_p(\mathbf{x}, \mathbf{z}) = p(k(\mathbf{x}, \mathbf{z})),$$

where p is a nonzero polynomial function with nonnegative coefficients. Further examples for kernels based on the polynomial one are **All-subset kernels** and **ANOVA kernels**, a detailed description and analysis of which is given in [9]. For the latter, the norm specializes a more general metric function $d \in \mathbb{K}^n \times \mathbb{K}^n$, where the kernel takes the form

$$\exp\left(\frac{-d(\mathbf{x}, \mathbf{z})}{2\sigma}\right).$$

In the remainder of this work we will, if not mentioned otherwise, make of use the exponential kernels with constant $\gamma = \frac{1}{2\sigma}$.

Example 1.37 (Hardy space)

$$H^2(\mathbb{D}_1) = \left\{ f : \mathbb{D}_1 \to \mathbb{C} \,\middle|\, f \text{ analytic and } \sup_{0 \le r < 1} \int_0^{2\pi} \frac{|f(re^{it})|^2}{2\pi} dt < \infty \right\}$$

is the space of square-integrable holomorphic functions defined on the unit disc

$$\mathbb{D}_1 = \{ z \in \mathbb{C} \,\big|\, |z| < 1 \}$$

and is called **Hardy space**. For $X \subset \mathbb{D}_1$ and $x, z \in X$,

$$k(x,z) = \frac{1}{\pi(1 - x\overline{z})^2}$$

is the reproducing kernel of $H^2(\mathbb{D}_1)$.

–●

1.4.3 The reproducing property

As mentioned in the introduction of this section, we want to give some examples of proving the reproducing property of reproducing kernel. As a side effect, we will gain interesting insight into the structure of the respective underlying reproducing kernel Hilbert spaces.

The first example builds on a reproducing kernel we already know, the polynomial kernel of degree 2, $k_{p^2}(\mathbf{x}, \mathbf{z})$. By proving its reproducing property, we moreover will deduce a basis for the reproducing kernel Hilbert space and thus know which elements it consists of.

Example 1.38
By equation (1.11), the evaluation of the RKHS determined by this kernel is spanned by linear combinations of the latter. After expanding the kernel we have

$$k_{p^2}(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \odot \mathbf{z} + c)^2 = x_2^2 z_2^2 + 2c x_1^1 z_1^1 + x_0^0 z_0^0 c^2 = x_2^2 z_2^2 + 2c x_1 z_1 + c^2, \quad \mathbf{x}, \mathbf{z} \in \mathbb{R}^3, \quad c \in \mathbb{R}_+,$$

where $\odot$ is the element wise product. This is – as expected – a polynomial, and obviously $\operatorname{span}\{k(\cdot, \mathbf{z})\} \subset \operatorname{span}\{1, \mathbf{x}, \mathbf{x}^2\}$. Hence, if we now can show that, by virtue of equation (1.11), we can express linear combinations of kernels by those of polynomials,

$$\sum_{i=0}^{2} \beta_i k(\mathbf{x}, \mathbf{z}_i) = \sum_{i=0}^{2} \alpha_i x^i$$

with $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \alpha_2)$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$, for $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^3$ the reproducing property is proven to be true. This, however, can easily be seen: We rewrite the equation as

$$\begin{pmatrix} \mathbf{z}_0 \\ \mathbf{z}_1 \\ \mathbf{z}_2 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \frac{\alpha_0}{c^2} \\ \frac{\alpha_1}{cz} \\ \alpha_2 \end{pmatrix}. \tag{1.25}$$

which always has a solution for any such $\boldsymbol{\alpha}$ – the vector we are looking for[4]. This implies, that all polynomials in $\mathbb{R}[x]$ of degree 2 are elements of the RKHS $\mathcal{H}_{k_{p^2}}$, which in this particular case is of dimension 3 and by the derivations above has the basis $\{1, x, x^2\}$.

–●

For the second example we will consider the set of unbounded Fourier transforms and also concretize on bandlimited transforms[5].

Example 1.39
Fouriertransforms can be defined in $L^2(\mathbb{R})$ on usual Borel sets of intervals in $\mathbb{R}$ using the common Lebesgue measure

$$M = \left\{ f : \mathbb{R} \to \mathbb{C} \quad | \quad \int_{-\infty}^{\infty} |f(t)|^2 \, dt < \infty \right\}, f(t) \mapsto \hat{f}(\omega) = \lim_{A \to \infty} \int_{-A}^{A} f(t) e^{-i\omega t} dt, \tag{1.26}$$

where $f \in L^2(-A, A)$. The inner product in this space is defined as

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(t)\overline{g(t)} dt, f, g \in M.$$

Assuming M to be a Hilbert function space with reproducing kernel k, let us look once again at the reproducing property

$$f(t) = \langle f, k(\cdot, t) \rangle \tag{1.27}$$

and its meaning in this situation. If such a kernel k exists, it must satisfy equation (1.27).

[4]Note that $\mathbf{z}_i$ are (row) vectors: Each $\mathbf{z}_i, i = 0..2$ has 3 components $z_2, z_1, z_0 = 1$. As (1.25) is a system of equations with vectors, writing out the elements of the matrix would result in an awkward and indecipherable amount of indices.

[5]Keep im mind that at this point we do not yet know whether or not this is a reproducing Hilbert space at all!

We can utilize the facts that Fourier transforms are isometric isomorphisms and that any function $f \in L^2(\mathbb{R})$ for which the Fourier transform exists can also be expressed via its inverse Fourier transform. By replacing $\hat{f}(\omega)$ by its integral transform and using s as the variable of integration, we get

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(s) e^{-i\omega s} ds \, e^{i\omega t} d\omega,$$

In the sense of the limit as given by equation (1.26), both transforms are well-defined and finite[6]. The theorem of Fubini allows us to change the order of integration:

$$\begin{aligned} f(t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(s) e^{-i\omega s} e^{i\omega t} ds d\omega = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(s) e^{-i\omega(t-s)} ds d\omega && (1.28) \\ &= \int_{-\infty}^{\infty} \frac{1}{2\pi} \left[\int_{-\infty}^{\infty} e^{i\omega(t-s)} d\omega \right] f(s) ds && (1.29) \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \delta(t-s) f(s) ds = f(t). && (1.30) \end{aligned}$$

Hence, switching back to our former form of notation, for $k_{\mathcal{F}}(\mu, \nu) = \int_{-\infty}^{\infty} e^{i\omega(\nu-\mu)t} dt$, the reproducing property $f(\nu) = \langle f, k(\cdot, \nu) \rangle$ is satisfied. Furthermore $k_{\mathcal{F}}$ is clearly positive semidefinite and thereby proven to be the reproducing kernel of the set of unbounded fourier transforms.

By definition, bandlimited functions vanish almost everywhere outside $]-\pi, \pi[$:

$$f(t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{f}(\omega) e^{i\omega t} d\omega, \hat{f} \in L^2(-\pi, \pi).$$

The reproducing kernel hence takes the form

$$k(\mu, \nu) = \int_{-\pi}^{\pi} e^{-i\omega(\nu-\mu)t} dt = \frac{\sin[\pi(\nu-\mu)]}{\pi(\nu-\mu)} = \begin{cases} 1 & \mu = \nu \\ \operatorname{sinc}[\pi(\nu-\mu)] & otherwise \end{cases}.$$

–●

This result is especially nice, given the fact that we can deduce the convergent series stated by Mercer's theorem (1.28) directly from the hitherto result:

[6]Note that the integral transform does not converge pointwise, but in the L^2-sense.

As $\sin(x) = \sum_{n=1}^{\infty} \frac{x^{2n-1}}{(2n-1)!}$ we conclude

$$k(\mu, \nu) = \begin{cases} 1 & \mu = \nu \\ 1 - \sum_{n=1}^{\infty} \frac{x^{2n}}{(2n+1)!} & otherwise \end{cases}$$

1.4.4 Reproducing kernels and pattern recognition

Reproducing kernels have been considered within the context of many common algorithmic procedures, the fundamental step always being the substitution of inner products by reproducing kernels. Numerous algorithms make use of inner products or can be rewritten in such a way, ranging from smaller tasks like data centering, clustering and projection to more complex algorithms such as canonical correlation analysis, linear regression and calculation of smallest enclosing hyperspheres for a collection of data points. [9] is a very good reference for algorithms based on kernel methods. The motivation is, as explained before, the interpretation of a RKHS as a feature space represented by metric relations (or, if available, the functionals forming the functional space).

As an example for an algorithm we present the basic steps of reproducing kernel principal component analysis (RKPCA), PCA[7] being one of the most widely used eigen-decompoisition methods. For more detailed derivations, analysis and comparison to other methods the reader is referred to [11], [9] or [12].

Example 1.40 (Kernel PCA)
Given N centered[8] data points $\mathbf{x}_n \in \mathbb{R}^p$, $1 \le n \le N$ and their approximated, positive definite covariance matrix $C = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T$, the principal components are found by diagonalization of C. This boils down to solving the eigenvalue problem $C\mathbf{z} = \lambda\mathbf{z}$ where $\lambda \in \mathbb{R}$ and $\mathbf{z} \in \mathbb{R}^p$. Substituting the eigenequation into the sum for computing C delivers $\lambda\mathbf{z} = \frac{1}{N}\sum_{n=1}^{N} \langle \mathbf{x}_n, \mathbf{z}\rangle \mathbf{x}_n$, or

$$\lambda \langle \mathbf{x}_n, \mathbf{z}\rangle = \langle \mathbf{x}_n, C\mathbf{z}\rangle, \quad 1 \le n \le N. \tag{1.31}$$

Given a feature mapping $T : \mathbb{R}^p \to F$ into a finite dimensional feature space F, the problem translates canonically:

[7]PCA is also known as the **Karhunen-Loève transform.**
[8]$\sum_{n=1}^{N} \mathbf{x}_n = 0$

Assuming the data is centered in the feature space, we need to compute

$$\lambda \langle T\mathbf{x}_n, \mathbf{z}\rangle = \langle T\mathbf{x}_n, C\mathbf{z}\rangle , \quad 1 \leq n \leq N. \tag{1.32}$$

As $F = \operatorname{span}\{T\mathbf{x}_1, \dots, T\mathbf{x}_N\}$ is finite dimensional, each vector $\mathbf{z} \in F$ can be written as a linear combination $\mathbf{z} = \sum_{n=1}^{N} \alpha_n T\mathbf{x}_n$, $\alpha_n \in \mathbb{R}$. Substitution of the last equation into 1.32 delivers N equations

$$\lambda \sum_{n=1}^{N} \alpha_n \langle T\mathbf{x}_n, T\mathbf{x}_i\rangle = \frac{1}{N} \sum_{n=1}^{N} \left\langle T\mathbf{x}_n, \sum_{l=1}^{N} T\mathbf{x}_l \underbrace{\langle T\mathbf{x}_l, T\mathbf{x}_n\rangle}_{k_{n,l}} \right\rangle . \tag{1.33}$$

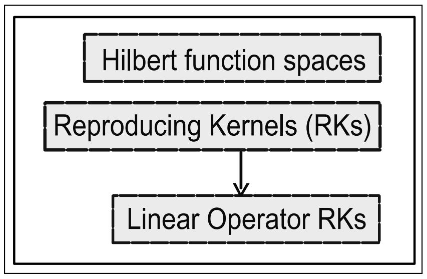
One point of substitution of the reproducing kernel for the inner product is indicated in the last equation. Substituting all occurrences and using the Gramian (oder kernel matrix) representation, the notation simplifies and the following eigenvalue problem remains to be solved:

$$\lambda G^k \boldsymbol{\alpha} = \frac{1}{N} G^k \boldsymbol{\alpha}, \tag{1.34}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_N)^T$. For details, we refer the reader to chapter 6.2 in [9].

–●

1.5 Reproducing kernels and linear operators



In this section we derive the theoretical results necessary for section 2.5.3, where we introduce correlation features and integrate them directly into the reproducing kernel. We first present the theory in the context of linear mappings in scalar-valued Hilbert spaces (section 1.5.1) and afterwards generalize to operator-valued situations (section 1.5.3).

1.5.1 Embedding linear mappings into reproducing kernels

Consider a continuous, linear mapping $T : X \to Y$ between two **Banach**[9] **spaces** X, Y. Let $X \oplus Y$ denote the product space of elements in X and Y, that is $X \oplus Y = \{(x, y) \mid x \in X,\, y \in Y\}$. A basic result from functional analysis is that the space $X \oplus Y$ with its norm given by $||(x, y)|| = \sqrt{||x||^2 + ||y||^2}$, $x \in X, y \in Y$ is again a Banach space.

As an application of this, the closed graph theorem ([13], pages 156 and 157) implies that the graph of T,

$$G(T) = \{(x, Tx) \mid x \in X\},$$

is a closed subspace of $X \oplus Y$ with norm

$$||x||_T = \sqrt{||x||^2 + ||Tx||^2} \geq ||x|| \,. \tag{1.35}$$

In the context of Hilbert spaces, $G(T)$ as a closed subspace is itself a Hilbert space, where the inner product is defined on the concatenation of the components. Let $\mathcal{H}, \mathcal{H}_T$ be Hilbert spaces, $p, q \in \mathcal{H}$ and $T : \mathcal{H} \to \mathcal{H}_T$ with respective inner products $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and $\langle \cdot, \cdot \rangle_{\mathcal{H}_T}$.

[9] A Banach space B is a normed vector space over a field $\mathbb{R}$ or $\mathbb{C}$ that is complete w.r.t its norm. It is a more general space than a Hilbert space, the latter being a Banach space with an inner product induced by the norm.

Then $G(T) \subset \mathcal{H} \oplus \mathcal{H}_T$ and

$$\langle (p, Tp), (q, Tq) \rangle_{G(T)} = \langle p, q \rangle_{\mathcal{H}} + \langle Tp, Tq \rangle_{\mathcal{H}_T} . \tag{1.36}$$

Given the above mentioned situation, the well-known representation lemma named after Riesz (see for instance [13], pages 109–112 and 235 –239) can be applied. It ensures that for every bounded, linear and continuous operator (mapping) $T : \mathcal{H} \to \mathcal{H}_T$ between a finite dimensional Hilbert space $\mathcal{H}$ and a Hilbert space $\mathcal{H}_T$ there exists exactly one adjoint operator (mapping) $T^* : \mathcal{H}_T \to \mathcal{H}$ such that for all $p \in \mathcal{H}$, $q' \in \mathcal{H}_T$ the equation $\langle Tp, q' \rangle_{\mathcal{H}_T} = \langle p, T^* q' \rangle_{\mathcal{H}}$ holds.

In our work we build on this theorem and, using the bilinearity of inner products in $\mathbb{R}$ (the complex case behaves analogous using the inner product's sesquilinearity instead) and recasting equation (1.36) as follows:

$$\begin{aligned} \langle (p, Tp), (q, Tq) \rangle_{G(T)} &= \langle p, q \rangle_{\mathcal{H}} + \langle Tp, Tq \rangle_{\mathcal{H}_T} \\ &= \langle p, q \rangle_{\mathcal{H}} + \langle p, T^* T q \rangle_{\mathcal{H}} \\ &= \langle p, q + T^* T q \rangle_{\mathcal{H}} \\ &= \langle p, (I_{\mathcal{H}} + T^* T) q \rangle_{\mathcal{H}} , \end{aligned} \tag{1.37}$$

where $I_{\mathcal{H}}$ is the neutral element (that is, the identity matrix) of the endomorphisms of $\mathcal{H}$ and the last inner product is defined on $\mathcal{H} \times \mathcal{H}$. T^*T is positive semi-definite (p.s.d), as for any $z \in \mathbb{C}$ we have $\overline{z}(T^*T)z = (\overline{z}T^*)(Tz) = (Tz)^*(Tz) \geq 0$, and whenever the trace of T^*T does not equal zero it is even positive definite (p.d.). In the latter case, $(I_{\mathcal{H}} + T^*T)$ will thus be p.d., too, and a new reproducing kernel stemming upon T is given by

$$k_{T^*T}(p, q) = \overline{p}(I_{\mathcal{H}} + T^*T)q. \tag{1.38}$$

The effect of T on the kernel can be controlled further by scaling the inner products $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and $\langle \cdot, \cdot \rangle_{\mathcal{H}_T}$ by positive weights $w_{\mathcal{H}}$ and $w_{\mathcal{H}_T}$ satisfying $w_{\mathcal{H}} + w_{\mathcal{H}_T} = 1$. Using basic linear algebra, we compute

$$\begin{aligned} w_{\mathcal{H}} \langle p, q \rangle + w_{\mathcal{H}_T} \langle Tp, Tq \rangle &= \langle p, w_{\mathcal{H}} q \rangle + \langle p, w_{\mathcal{H}_T} T^* T q \rangle \\ &= \langle p, w_{\mathcal{H}} q + w_{\mathcal{H}_T} T^* T q \rangle \\ &= \langle p, (w_{\mathcal{H}} I_{\mathcal{H}} + w_{\mathcal{H}_T} T^* T) q \rangle_{\mathcal{H}} \end{aligned} \tag{1.39}$$

and thus

$$k^w_{T^*T}(p,q) = \overline{p}(w_{\mathcal{H}} I_{\mathcal{H}} + w_{\mathcal{H}_T} T^* T)q. \tag{1.40}$$

Note that $w_{\mathcal{H}}, w_{\mathcal{H}_T} > 0$ will ensure positive definiteness.

Finally, for T unitary, the weighting in equation (1.37) reduces to scaling q, as

$$\langle (p, Tp), (q, Tq) \rangle_{G(T)} \;\; = \;\; \langle p, ((w_{\mathcal{H}} + w_{\mathcal{H}_T}) I_{\mathcal{H}})\, q \rangle_{\mathcal{H}} \,. \tag{1.41}$$

Remark 1.41
A crucial characteristic of the derived kernel $k^w_{T^*T}(p,q)$ is that it is defined on the original set $X \times X$. We thus can, as depicted in section 2.5.3 , make use of theorem 1.32, for which this characteristic is a necessary requirement.

1.5.2 Correctness of the linear operator RK

It is still not clear that the integration of the a linear mapping T into the inner product delivers a bounded operator. To prove this, note that $T : \mathcal{H} \to \mathcal{H}_T$ is defined on a subset of a space that is itself a reproducing kernel space. Now, considering the first line of equation 1.36, we have

$$\langle (p, Tp), (q, Tq) \rangle_{G(T)} = \langle p, q \rangle_{\mathcal{H}} + \langle Tp, Tq \rangle_{\mathcal{H}_T} \,, \tag{1.42}$$

which offers a good point for our estimations.

Theorem 1.42
The reproducing kernel defined by equation (1.38) is bounded.

Proof. We prove this by showing that both inner products on $\mathcal{H}$ and $\mathcal{H}_T$ are finite. Given finite $p, q \in \mathcal{H}$ it is clear that $\langle p, q \rangle$ and $||\langle p, q \rangle||$ are finite as well. The operator norm for bounded linear operators is, as already noted in equation (1.5), given by

$$\sup_{||h|| \neq 0} \frac{||Th||}{||h||} = \sup_{||h||=1} ||Th|| \,.$$

It satisfies

$$||Th|| \leq ||T|| \, ||h|| \,,$$

and for a proof the reader can consult for instance [14], pages 91 to 93.

Now, as T is bounded and the inner product homogeneous in both terms, we conclude

$$||\langle Tp, Tq\rangle|| \leq ||T||^2 \, ||\langle p, q\rangle|| \leq \infty, \tag{1.43}$$

which proves finiteness of the inner products.

□

1.5.3 Generalization to linear operators

Let $T : \mathcal{H}_k \to \mathcal{H}$ be a bounded linear operator from a RKHS $\mathcal{H}_k$ on a set E with kernel k and let $\mathcal{H}$ be an ordinary Hilbert space. In such a function space, the graph is, in the same manner as before, defined as $G(T) = \{(f, Tf) \,|\, f \in \mathcal{H}_k\}$ and again – as a closed subspace – a Hilbert space itself, its inner product being

$$\langle (f, Tf), (g, Tg)\rangle_{G(T)} = \langle f, g\rangle_{\mathcal{H}_k} + \langle Tf, Tg\rangle_{\mathcal{H}}\,. \tag{1.44}$$

In the same way as before with additionally making use of the reproducing property in step (1.45) we deduce

$$\begin{aligned}
\langle (f, Tf), (g, Tg)\rangle_{G(T)} &= \langle f, k(\cdot, g)\rangle + \langle Tf, Tk(\cdot, g)\rangle \\
&= \langle f + T^*Tf, k(\cdot, g)\rangle \\
&= \langle (I + T^*T)f, k(\cdot, g)\rangle \\
&= \langle f, (I + T^*T)k(\cdot, g)\rangle
\end{aligned} \tag{1.45}$$

A new reproducing kernel k_{T^*T} in $G(T)$ can now be defined in terms of the known kernel k as follows:

$$k_{T^*T}(f, g) = f^*(I + T^*T)^{-1}k(\cdot, g). \tag{1.46}$$

Note that $(I + T^*T)$ is psd and symmetric and thus an invertible operator.

Saitoh ([10], pp. 80 – 81) implicitly uses results similar to (1.46) in an example where $T : \mathbb{C}^n \to \mathbb{C}$ is the evaluation functional, which is necessarily an element of any RKHS. For this special case of $(T^*T)(\cdot)$ he elaborates an analytic equation for the new kernel and states that the associate RKHS is a strict subspace of $\mathcal{H}_k$.

This is not entirely correct, especially in our particular setting. Instead we prove the following

Theorem 1.43
$T : \mathcal{H}_k \to G(T)$ as defined above is a topological vector space isomorphism of Hilbert spaces.

Proof. We have to show that there exists a mapping $G(T) \to \mathcal{H}_k$ that is both one-to-one and onto as well as continuous and linear and has an continuous inverse mapping. Let $||\cdot||_T$ denote the norm for T as given in equation (1.35). Then the identity mapping (or alternatively we may call it projection onto the first component)

$$\begin{aligned} G(T) \to \mathcal{H}_k \quad : \quad & (f, Tf) \mapsto f \\ & (p, ||\cdot||_T) \mapsto (p, ||\cdot||) \end{aligned}$$

clearly meets the claimed conditions, thereby proving the theorem.

□

Remark 1.44
Cave! For both operators and maps, care must be taken that $\langle Tf, Tg\rangle$ and $\langle Tp, Tq\rangle$ remain inner products. Differentiation for instance renders the latter indefinite! Finally, the structure of Hilbert spaces is not even necessary.

Theorem 1.43 has important implications: Being a topological isomorphism, open sets remain open due to the continuity of the inverse mapping. As both domain and range are Banach oder Hilbert spaces, limits are carried over. For instance, Cauchy sequences in the domain map to Cauchy sequences in the range.

We close this chapter by giving another simple example of a topological vector space isomorphism, which is well known from linear algebra.

Example 1.45
Given a finite dimensional Banach space E over a field $\mathbb{K}$ with base $\{e_1, \dots, e_n\}$, the mapping

$$T : E \to \mathbb{K}^n, \qquad x = \sum_{i=1}^{n} x_i e_i \mapsto (a_1, \dots, a_n),$$

$a_1, \dots, a_n \in \mathbb{K}$ is a topological vector space isomorphism. Linearity, well-definiteness and bijectivity are clear. The fact that the inverse mapping is an element of $L(\mathbb{K}^n, E)$ can easily be seen using the Cauchy Schwarz inequality:

$$\left|\left|T^{-1}\mathbf{x}\right|\right| \leq \sum_{i=1}^{n} |x_i| \, ||e_i|| \leq \left(\sum_{i=1}^{n} |x_i|^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{n} ||e_i||^2\right)^{\frac{1}{2}} \leq c\,|\mathbf{x}|\,,$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ and $c = \left(\sum_{i=1}^{n} ||e_i||^2\right)^{\frac{1}{2}}$.

—●

1.6 Optimization theory

Convex Optimization

This section offers a short introduction to the theory of (convex) optimization, covering enough material to understand the basics necessary for support vector classification, which as an important utility for our research is presented in section 1.7. Most of this section as well as of section 1.6.1 is based on [15] and hence we recommend this (downloadable for free) book of Stephen Boid for further details and proofs. Note however, that the notation given here is quite different as the chapter includes some results of our own, and we preferred our deviating variant.

An **optimization problem** in a very general form is a mathematical problem for which, in most cases under certain constraints, an optimal solution in the form of a minimum or maximum[10] is sought:

$$\begin{aligned} &\underset{\mathbf{x}\in\mathcal{D}\subset\mathbb{R}^n}{\text{minimize}} && f(\mathbf{x}) \\ &\text{s.t.} && \begin{cases} g(\mathbf{x}) = 0 \\ h(\mathbf{x}) \leq 0 \end{cases} \end{aligned} \tag{1.47}$$

Here, f is called **objective function** and $g : \mathbb{R}^n \to \mathbb{R}^k$, $h : \mathbb{R}^n \to \mathbb{R}^l$ are called **equality** and **inequality constraints**, respectively. The subscript – in this case $\mathbf{x} \in \mathcal{D}$ – indicates, with respect to which variable(s) f is to be optimized. The set of $\mathbf{x}^* \in \mathcal{D}$ leading to an optimal solution amongst all vectors in $\mathcal{D}$ is called the set of **objective values** or **solutions** of the optimization problem, where $\mathcal{D} = \text{dom} f \bigcap_{i=1}^{k} \text{dom} g_i \bigcap_{j=1}^{l} \text{dom} h_j$, the intersection of the domains of objective function and all constraints, is the problem's **domain**. If the problem does not include any constraints at all, it is called **unconstrained**.

For the sake of clearness we will often write $g_i(\cdot)$ and $h_j(\cdot)$ with $i = 1, \ldots, k$ $j = 1, \ldots, l$, where $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$. A point $\mathbf{y} \in \mathcal{D}$ is said to be **feasible**, if it satisfies all constrains of a problem, and the problem itself is feasible if its set of feasible points is nonempty. It is clear by definition that an objective value is necessarily feasible.

[10]Problem 1.47 is a given as a minimization problem. In the same way, a maximization problem can be defined by essentially just replacing the 'minimize' instruction by a 'maximize' instruction and evidently seeking a maximum instead of a minimum.

Problem 1.47 is the **standard form** representation. In this form, the righthand side of both kinds of constraints are zero and the inequality constraints are of $\leq$ - kind. Note that the first characteristic can always be achieved by subtracting the nonzero righthand side values, the second one by switching the signature of any $\geq$ - kind of inequality constraint. **Boxed constraints** of the form $c_i \leq h_i(\mathbf{x}) \leq b_i$, $b_i, c_i \in \mathbb{R}$, $i = 1, \ldots, l$ can be split into two sets of inequality constraints $c_i - h_i(\mathbf{x}) \leq 0$ and $h_i(\mathbf{x}) - b_i \leq 0$.

Example 1.46
Given $\mathbf{c}, \mathbf{x}, \mathbf{a}_i \in \mathbb{R}^n$, and $b_i \in \mathbb{R}$ for $i = 1, \ldots, m, \quad m \in \mathbb{N}$, the optimization problem

$$\begin{array}{cl} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} & \mathbf{a}_i^T \mathbf{x} \leq b_i \end{array} \tag{1.48}$$

is an example of a **linear program**, where the objective function as well as all constraints are linear in the optimization variable(s).

−●

Convex optimization problems play an important part especially in the context of support vector machines and dualization of optimization problems (see later this section). Reasons for this are, that even on a standard desktop computer, problems of relatively large size (hundreds of optimization variables and thousands of constraints) can be solved in a minute's or two time. In addition, many nonconvex problems can be approximated well enough by convex ones. Another application is the computation of lower bounds by constraint relaxation, that is converting nonconvex constraints into weaker but convex ones. The term *relaxation* will be introduced later in this section.

We define (the class of) convex optimization problems as follows:

Definition 1.47
An optimization problem

$$\begin{array}{cl} \underset{\mathbf{x} \in \mathcal{D} \subset \mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{s.t.} & h_i(\mathbf{x}) \leq b_i, \end{array} \tag{1.49}$$

with convex objective function f, $\mathbf{x} \in \mathbb{R}^n$, convex inequality constraints $h_i : \mathbb{R}^n \to \mathbb{R}$ (i.o.w. $h_i(\alpha \mathbf{x} + \beta \mathbf{y}) \leq \alpha h_i(\mathbf{x}) + \beta h_i(\mathbf{y})$ for $\mathbf{y} \in \mathbb{R}^n$ and $\alpha, \beta \in \mathbb{R}$) and affine equality constraints $\mathbf{a}_i^T \mathbf{x} \leq b_i$, $b_i \in \mathbb{R}$, is called **convex optimization problem**.

Note that linear programs such as given in example (1.46) are a special case of convex optimization problems. Due to the convex nature of support vector classification and the fact that dualization leads to convex optimization problems, we will restrict ourselves to the convex case in the mathematical review of this thesis.

1.6.1 Theoretical background

Definition 1.48
A set $D \subset \mathbb{R}^n$ is set to be **affine**, if for any $\mathbf{x}_1, \mathbf{x}_2 \in D$ and $\alpha \in \mathbb{R}$ we have $\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \in D$. Given any (not necessarily affine) set $C \subset \mathbb{R}^n$, $\sum_{i=1}^{k} \alpha_i \mathbf{x}_i$ is called a **affine combination** of the points $\mathbf{x}_i \in C$ if $\sum_{i=1}^{k} \alpha_i = 1$, $\alpha_i \in \mathbb{R}$.

Definition 1.49
A set $D \subset \mathbb{R}^n$ is set to be **convex**, if for any $\mathbf{x}_1, \mathbf{x}_2 \in D$ and $\alpha \in \mathbb{R}$, $0 \leq \alpha \leq 1$, we have $\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \in D$. Given any (not necessarily convex) set $C \subset \mathbb{R}^n$, $\sum_{i=1}^{k} \alpha_i \mathbf{x}_i$ is called a **convex combination** of the points $\mathbf{x}_i \in C$ if $\sum_{i=1}^{k} \alpha_i = 1$, $\alpha_i \in \mathbb{R}_+$.

Both affine and convex combinations are a special case of linear combinations.

Definition 1.50
A convex set D with $\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 \in C$ for any $\mathbf{x}_1, \mathbf{x}_2 \in C$ and $\alpha_1, \alpha_2 \in \mathbb{R}_+$ is called **convex cone**. Given any set $C \subset \mathbb{R}^n$, $\sum_{i=1}^{k} \alpha_i \mathbf{x}_i$ is called a **conic combination** of the points $\mathbf{x}_i \in C$ if $\alpha_i \in \mathbb{R}_+$. If $\mathbf{x}_i \in D$, $i = 1, \dots, k$, then so will be the conic combination.

The idea of convex cones is important insofar, as the constraints of a convex optimization problem form a convex cone. In addition, convexity is closed under a lot of common algebraic operations. Together, this opens ways of formulating a number of problems in such a way that both the problem itself as well as its complete computations preserve convexity, or it allows for recasting nonconvex problems into convex ones.

Operations preserving convexity of a set are for instance infinite intersections, mappings through affine functions and Möbius transforms. Proofs for these statements can be found in [15], pages 35 to 42.

Example 1.51 (from [15], page 35)
In the set of symmetric matrices $S^n = \{X \in \mathbb{R}^{n \times n} | X = X^T\}$, which is a vector space of dimension $\frac{n(n-1)}{2}$, we consider the two subsets of positive semidefinite and positive denifite matrices $S^n_{psd} = \{X \in S^n | X \geq 0\}$ and $S^n_{pd} = \{X \in S^n | X > 0\}$. Both are convex cones.

For a proof, consider the definition of a positive semidefinite matrix. Given $\alpha, \beta \in \mathbb{R}_+$ and $X, Y \in S^n_{psd}$, we have for any $\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{x}^T(\alpha X + \beta Y)\mathbf{x} = \alpha \mathbf{x}^T X \mathbf{x} + \beta \mathbf{x}^T Y \mathbf{x} \geq 0.$$

The proof for S^n_{pd} is identical. –●

Remark 1.52

Example 1.51 and the fact that each p.d. matrix represents a (unique) reproducing kernel inducing its associated Hilbert function space imply that reproducing kernels over the same set form a convex cone. This is an important fact for our approach, as we will formulate our discriminative classifier given a kernel combination over a set. As a consequence, the resulting optimization problem is guaranteed to remain within a convex cone and will thereby be uniquely solvable.

By definition, convex optimization problems have convex inequality constraints and we quickly revisit the definition of convex functions:

Definition 1.53

A function $f : D \to \mathbb{R}$ defined on a convex set $D \subset \mathbb{R}^n$ is **convex**, if for all $\alpha \in \mathbb{R}$, $0 \leq \alpha \leq 1$ and $\mathbf{x}, \mathbf{y} \in D$ we have

$$f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}). \tag{1.50}$$

If $-f$ is convex, we say that f is **concave**.

The geometric interpretation of a function f being convex is, that the line segment from $(\mathbf{x}, f(\mathbf{x}))$ to $(\mathbf{y}, f(\mathbf{y})$ lies completely above the graph of f. If the inequality in equation (1.50) is strict, f is called **strictly convex** (or **strictly concave**). An aspect of utter importance is, that for convex functions one can conclude from local information to a global characteristic:

Theorem 1.54

Given a convex, differentiable function $f : D \to \mathbb{R}$, $D \subset \mathbb{R}^n$ convex, the following inequality holds for all $\mathbf{x}, \mathbf{y} \in D$:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}). \tag{1.51}$$

Vice versa, if inequality (1.51) holds for a function f defined on a convex set $D \subset \mathbb{R}^n$, f is a convex function.

Proof. See [15], page 70. □

The term on the righthand side is immediately recognized as the first order Taylor approximation of f. The local information in this context are the values of the function and its derivative at a given point, the global characteristic is, that the first order Taylor approximation is a lower bound of f. Inequality (1.51) is known as the **first order condition** for a convex function. As one might expect, there is a similar **second order condition**, which we also present without proof.

Theorem 1.55
Assuming that f is twice differentiable, $f : D \subset \mathbb{R}^n \to \mathbb{R}$ is

(i) convex if and only if D is convex and its Hessian matrix is positive semidefinite,

(ii) concave, if it is negative semidefinite.

Example 1.56
Any norm $||\cdot||$ on $\mathbb{R}^n$ is a convex function. Its positive homogeneity and the triangle inequality immediately prove the statement.

–●

Example 1.57 (Important for optimization problems such as support vector classification)
Given $x_n \in \mathbb{R}, 1 \leq n \leq N \in \mathbb{N}$, $f : \mathbb{R}^N \to \mathbb{R}$, the mapping $\mathbf{x} \mapsto \max\{x_1, \ldots, x_N\}$ is convex. The proof is straightforward. For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$, $0 \leq \alpha \leq 1, \alpha \in \mathbb{R}$, we have

$$\begin{aligned} f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) &= \max_n(\alpha x_n + (1-\alpha)y_n) \\ &= \max_n(\alpha x_n) + \max((1-\alpha)y_n) \\ &= \alpha \max_n(x_n) + (1-\alpha)\max(y_n) \\ &= \alpha f(\mathbf{x}) + (1-\alpha)f(\mathbf{y}). \end{aligned}$$

The same proof shows that the min function is convex, and in a similar manner it can easily be seen that pointwise maximum and minimum of (a set or family of) pairwise real valued convex functions are convex:

For $g, h : \mathbb{R}^N \to \mathbb{R}$ and $f(\mathbf{x}) = \max\{g(\mathbf{x}), h(\mathbf{x})\}$ the convexity of g and h and the pointwise convexity of the maximum itself prove

$$\begin{aligned} f(\alpha\mathbf{x} + (1-\alpha)\mathbf{x}) &= \alpha \max\{g(\mathbf{x}), h(\mathbf{x})\} + (1-\alpha)\max\{g(\mathbf{y}), h(\mathbf{y})\} \\ &= \alpha f(\mathbf{x}) + (1-\alpha)f(\mathbf{y}). \end{aligned}$$

–●

Definition 1.58
In an n-dimensional space $\mathcal{H}$ equipped with an inner product $\langle \cdot, \cdot, \rangle$, writing $\mathbf{x} = (x_1 \cdots x_n)$, $\mathbf{w} = (w_1 \cdots w_n)$ and letting $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{H}$, $\mathbf{w} \in \mathcal{H}$ and $b \in \mathbb{R}$, a **hyperplane** is a subspace $\{\langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \,|\, \mathbf{x} \in \mathcal{H}\}$, determined by a vector $\mathbf{w}$ orthogonal to the hyperplane and an offset b.

From linear algebra we know that a dot product in a Hilbert space represents the length of the projection of either component onto the direction of the remaining one. Thus $\langle \mathbf{w}, \mathbf{x} \rangle$ is the length of the projection of $\mathbf{x}$ along the direction of $\mathbf{w}$ scaled by $||\mathbf{w}||$ – and vice versa. Geometrically, the definition shows that a hyperplane is the (sub)set of points with a constant inner product, analytically being determined by the solution set of a linear equation of an affine set over $\mathbf{x}$.

1.6.2 Lagrangian- and Kuhn Tucker multipliers

When we introduced optimization problems at the beginning of this section, we distinguished between equality and inequality constraints $g_i(\cdot)$ and $h_i(\cdot)$. Now we will examine how such constraints can be handled given the task of finding solutions to the problem. In this context we will introduce **Lagrangian multipliers**, which come into play when dealing with equality constraints, and **Kuhn Tucker multipliers** – an equivalent to the Lagrangian multipliers for inequality constraints.

Naturally, both kinds of multipliers comprise the foundation for solving problems with mixed constraints and for so called **dualization**, which will be introduced in section 1.6.3. We will also give an interpretation of the Lagrangian multipliers in subsection 1.6.4. The theorems of this sections are presented without proofs, which can either be found in any book on analysis and advanced calculus or in works covering the topics of (convex) optimization or numerical analysis, such as [15].

Theorem 1.59 (Lagrangian multipliers)
Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ (thus $g : \mathbb{R}^n \to \mathbb{R}^k$, $g = (g_1, \ldots, g_k)$, $1 \leq i \leq k$), $n \in \mathbb{N}_+$, be continuously differentiable functions and $f(\mathbf{x}^*)$, $\mathbf{x}^* \in \mathcal{D}$, an optimal solution of f defined on an open subset of the problem's domain $\mathcal{D}$. If the Jacobi matrix $\mathcal{J}g$ has full image space rank k, there exists a vector $\lambda = (\lambda_1, \ldots, \lambda_k) \in \mathbb{R}^k$ such that

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^{k} \lambda_i \nabla g_i(\mathbf{x}^*) = 0, \tag{1.52}$$

where $\nabla f = \left(\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n}\right)$ and $\nabla g_i = \left(\frac{\partial g_i}{\partial x_1}, \cdots, \frac{\partial g_i}{\partial x_n}\right)$, $1 \leq i \leq n$, are the gradient vectors of f and g_i respectively. λ is called (the vector of) **Lagrangian multiplier(s)** or **Lagrangian multiplier(s) for equality constraints**.

Remark 1.60
As will become clear immediately, the additional term – the linear sum of the Lagrangian multipliers – which is added to the necessary condition for an optimal solution is itself zero and thus does not modify the condition itself.

The theorem formulates a condition based on the first derivative of the objective function with respect to the equality constraints. For this reason, the conditions are also known as first-order necessary conditions. Comparing the results to that of finding extremal values in standard real analysis, the situation might suggest the existence of (sufficient) second-order conditions. And indeed, there are – again, as there were for the first and second order conditions, see theorem 1.55 – equivalent requirements.
Looking at the formulation of the next theorem, where $L_{\mathbf{x},\lambda}$ is used as a short form for the linear term $f(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i g_i(\mathbf{x})$, one will immediately recognize the similarity to necessary and sufficient conditions for extremal values of unconstrained problems.

Theorem 1.61
Consider an optimal solution $\mathbf{x}^* \in \mathcal{D}$ of an at least two times continuously differentiable function $\mathcal{C}^2 \ni f : \mathbb{R}^n \to \mathbb{R}$ and a vector $\lambda \in \mathbb{R}^k$ of Lagrangian multipliers such that the Jacobi matrix Dg, $g \in \mathcal{C}^2$, has full image space rank k and satisfies the first order condition $\nabla f(\mathbf{x}^*) + \sum_{i=1}^{k} \lambda_i \nabla g_i(\mathbf{x}^*) = 0$. Then for any such f, g and $\forall \mathbf{z}_{\mathbf{x}^*} \in \mathcal{K}(\mathbf{x}^*) = \{\tilde{\mathbf{z}} \in \mathbb{R}^n | \left(\nabla g_i(\mathbf{x}^*)\right) \tilde{\mathbf{z}} = 0\}$, the following implications hold

(1a) $f(\mathbf{x}^*)$ is a local minimum on $\mathcal{D} \Rightarrow \mathbf{z}_{\mathbf{x}^*}^T \nabla^2 L_{\mathbf{x},\lambda} \mathbf{z}_{\mathbf{x}^*} \leq 0$. For $\mathbf{z}_{\mathbf{x}^*} \neq 0$, the minimum is strict, if the inequality is strict.

(1b) $f(\mathbf{x}^*)$ is a local maximum on $\mathcal{D} \Rightarrow \mathbf{z}_{\mathbf{x}^*}^T \nabla^2 L_{\mathbf{x},\lambda} \mathbf{z}_{\mathbf{x}^*} \geq 0$. For $\mathbf{z}_{\mathbf{x}^*} \neq 0$, the maximum is strict, if the inequality is strict.

Here $\nabla^2 L_{\mathbf{x},\lambda}$ denotes the Hessian Matrix of L at $\mathbf{x}$ given the multiplier vector λ.

As indicated above, there is a formulation for optimization problems with inequality constraints resembling the first order conditions of the Lagrangian multipliers.

Theorem 1.62 (Kuhn Tucker multipliers)
Let $f : \mathbb{R}^n \to \mathbb{R}$ and $h_j : \mathbb{R}^n \to \mathbb{R}$, $1 \leq j \leq l$ (thus $h : \mathbb{R}^n \to \mathbb{R}^l$, $h = (h_1, \ldots, h_l)$), $n \in \mathbb{N}_+$, be continuously differentiable functions.

Suppose that $f(\mathbf{x}^*)$, $\mathbf{x}^* \in \mathcal{D}$, is an optimal solution of f defined on an open subset of the problem's domain $\mathcal{D}$. If the Jacobi matrix Dh has full rank l there exists a vector $\zeta \in \mathbb{R}^l$ such that

(1) $\zeta \geq 0$, (This inequality is to be understood elementwise.)

(2) $\zeta_j h_j(\mathbf{x}^*) = 0$, $j = 1, \ldots, l$

(3a) $\nabla f(\mathbf{x}^*) + \sum_{j=1}^{l} \zeta_j \nabla h_j(\mathbf{x}^*) = 0$ for a maximum.

(3b) $\nabla f(\mathbf{x}^*) - \sum_{j=1}^{l} \zeta_j \nabla h_j(\mathbf{x}^*) = 0$ for a minimum.

$\zeta \in \mathbb{R}^l$ is called (the vector of) **Kuhn Tucker multiplier(s)** or **Kuhn Tucker multiplier(s) for inequality constraints**.

Remark 1.63
It should be clear that we only consider points that meet the required conditions of the respective (in)equality constraints such as for instance $g_i(\cdot) = 0$. Furthermore, if any of the conditions for the theorem is not met, the implications will in general not hold – we will give an example where the rank of the Jacobian is not full and, as a consequence, the theorem in question fails.

1.6.3 Dualization

The concepts introduced above directly lead (or are directly connected) to dualization. As the following considerations hold for both equality constrained and inquality constrained optimization problems as well as those including both constraint types, we will jump right in an consider the all encompassing, latter case. Thus, given an optimization problem

$$\underset{\mathbf{x}\in\mathcal{D}\subset\mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) \qquad \text{s.t.} \quad \begin{cases} g(\mathbf{x}) = 0 \\ h(\mathbf{x}) \leq 0 \end{cases}, \tag{1.53}$$

$g : \mathbb{R}^n \to \mathbb{R}^k$, $h : \mathbb{R}^n \to \mathbb{R}^l$, let us look back at equation (1.52) and its counterpart (3) in theorem 1.62. Ignoring the differentiation operation for a moment, what basically is done in the subsequent considerations is that the constraints are integrated into the objective function in form of weighted linear sums.

Definition 1.64

$$L_{x,\lambda,\zeta} = f(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{l} \zeta_j h_j(\mathbf{x}) \tag{1.54}$$

is called the **Lagrangian** associated with the optimization problem (1.53). The vectors λ and ζ are, in this context, also called the **dual variables** associated with the optimization problem. The minimum of $L_{x,\lambda,\zeta}$ over $\mathbf{x} \in \mathcal{D}$ is called the **dual function**.

Theorem 1.65
The dual function associated with an optimization problem as above is concave even if the optimization problem itself is not convex.

Proof. The dual function is the pointwise minimum of a set of affine functions in the dual variables. Example (1.57) can be carried over to concavity with a nearly identical proof. This is all we need.

□

Remark 1.66
Let us summarize: Solving the dual problems requires maximization of a concave function (pointwise min/max) given convex constraints, which makes the optimization problem convex.

Example 1.67
We will derive the dual form for the inequality constrained optimization problem

$$\begin{aligned} \underset{\mathbf{x}\in\mathcal{D}\subset\mathbb{R}^n}{\text{maximize}} \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \leq b. \end{aligned} \tag{1.55}$$

We reformulate this as a minimization problem,

$$\begin{aligned} \underset{\mathbf{x}\in\mathcal{D}\subset\mathbb{R}^n}{\text{minimize}} \quad & -\mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \leq b, \end{aligned} \tag{1.56}$$

and minimize the problem's Lagrangian to get

$$\inf_{\mathbf{x}\in\mathcal{D}\subset\mathbb{R}^n} \{-\mathbf{c}^T\mathbf{x} + \zeta^T(A\mathbf{x} - b)\} = \inf_{\mathbf{x}\in\mathcal{D}\subset\mathbb{R}^n} \{-\mathbf{c} + (A^T\zeta)^T\mathbf{x} - \zeta^T b\}.$$

Before exploring some examples of constrained optimization problems, let us derive a meaningful interpretation of the multipliers.

1.6.4 Interpretation of the multipliers

An important question immediately arising is the sensitivity of the solution to the multipliers itself. How much will the latter deviate for (small) changes? The following considerations will give an idea of the 'meaning' of the Lagrangian multipliers and will directly answer the question.

Without loss of generality, let us fix all but one constraints of an optimization problem with equality constraints $g_i = c_i$, $i = 1, \dots, k$, while substituting the single remaining constraint g_ι for some $\iota \in [1, \dots, k]$ by an "unknown" parameter u. Now the optimal solution $\mathbf{x}^*$ depends on u, which we make clear by writing $\mathbf{x}^*(u)$. Furthermore, following theorem 1.59, the objective function f and the constraints g_i are evaluated with respect to the (new) optimal solution, hence we have to consider $f(\mathbf{x}^*(u))$, $g_i(\mathbf{x}^*(u))$. As the constraints must be met for any optimal solution, we get

$$g_i(\mathbf{x}^*(u)) = c_i, i = 1, \dots, k, i \neq \iota$$

and

$$g_\iota(\mathbf{x}^*(u)) = u.$$

The chainrule is used to compute the derivatives of the objective function and the constraints with respect to u:

$$\frac{\partial f(\mathbf{x}^*(u))}{\partial u} = \nabla f(\mathbf{x}^*(u)) \cdot \frac{\partial \mathbf{x}^*(u)}{\partial u} \tag{1.57}$$

and

$$\frac{\partial g_i(\mathbf{x}^*(u))}{\partial u} = \nabla g_i(\mathbf{x}^*(u)) \cdot \frac{\partial \mathbf{x}^*(u)}{\partial u} = \begin{cases} 1 & i = \iota \\ 0 & i \neq \iota \end{cases} \tag{1.58}$$

for $i = 1, \dots, k$. The first order condition yields

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^{k} \lambda_i \nabla g_i(\mathbf{x}^*) = 0 \Leftrightarrow \nabla f(\mathbf{x}^*) = -\sum_{i=1}^{k} \lambda_i \nabla g_i(\mathbf{x}^*), \tag{1.59}$$

and substituting this into equation (1.57) and applying the results of (1.58) delivers

$$\begin{aligned}\frac{\partial f(\mathbf{x}^*)}{\partial u} &= -\frac{\partial \mathbf{x}^*(u)}{\partial u}\sum_{i=1}^{k}\lambda_i g_i(\mathbf{x}^*(u)) = -\frac{\partial \mathbf{x}^*(u)}{\partial u}\lambda_\iota g_\iota(\mathbf{x}^*(u)) \\ &= -\lambda_\iota\left(g_\iota(\mathbf{x}^*(u))\cdot\frac{\partial \mathbf{x}^*(u)}{\partial u}\right) = -\lambda_\iota\end{aligned}$$

The essence of this is that the value of the objective function (in the neighbourhood of an optimal solution) is directly related to changes of the ith constraint, and the degree of this sensitivity is measured by the associated Lagrangian multiplier λ_i.

1.6.5 Constrained optimization examples

Example 1.68 (Equality constraints)
Consider the problem of maximizing the function $f:\mathbb{R}^2\to\mathbb{R}$, $(x_1,x_2)\mapsto(x_1x_2)^2$ given the equality constraint $g:\mathbb{R}^2\to\mathbb{R}$, $x_1^2+x_2^2-1=0$. Rewriting the latter to $x_1^2+x_2^2=1$ we can interpret this as finding optimal maximal solution on the boundary of the unit circle $\mathcal{S}^2$. As this is the first example, we will describe all steps taken in details. First, note that we are given one equality constraint and, following definition 1.59, thus have $k=1$. By definition of f, the dimension of the domain of f is $n=2$. We thereby know that

(a) If Lagrangian multipliers satisfying equation (1.52) exist[11], it must be exactly one multiplier $\lambda\in\mathbb{R}$ for the single given constraint.

(b) It is clear that $f,g\in\mathcal{C}^2$, guaranteeing the differentiability conditions of both theorems 1.59 and 1.61.

(c) Given n,k we have to solve a system of $n+k=2+1=3$ equations in three unknowns (x_1,x_2,λ), once again assuming that λ exists.

Before solving the above mentioned optimization problem, we can draw further helpful conclusions: As f is a continuous function defined on a compact subset $\mathcal{S}^2\subset\mathbb{R}^2$ (the overall domain $\mathcal{D}$ of the given optimization problem), f takes its maximum (and minimum) in $\mathcal{D}$. We can also exclude the cases $x_i=0$, $i=1,2$, as those are clearly not candidates for maximal extremal points.

[11] As noted before, conclusions of the theorems may not hold when any of the preliminaries is not met. It will become clear in example 1.70, that for instance $\text{rank}(g)<k$ may imply that the (vector of) Lagrangian multipliers satisfying equation (1.52) may not even exist.

To start with, we determine the set of critical points of $L_{x,\lambda}$ by solving $\nabla L_{x,\lambda} = 0$. For this task we compute

$$\nabla f = \begin{pmatrix} 2x_1x_2^2 \\ 2x_1^2x_2 \end{pmatrix}, \quad \nabla g = \begin{pmatrix} -2x_1 \\ -2x_2 \end{pmatrix}.$$

As $x_1 = 0 \Rightarrow x_2 \neq 0$ and $x_2 = 0 \Rightarrow x_1 \neq 0$ (otherwise $x_1^2 + x_2^2 = 0 \neq 1$ in contradiction to the constraint!) we have $\operatorname{rank}(g) = 1$. This ensures the existence of the Lagrangian multiplier λ and our system of equations is hence given by

$$\begin{aligned} 2x_1x_2^2 - 2\lambda x_1 &= 0 && (1.60)\\ 2x_1^2x_2 - 2\lambda x_2 &= 0 && (1.61)\\ 1 - x_1^2 - x_2^2 &= 0. && (1.62) \end{aligned}$$

The first line can be simplified to $x_2^2 = \lambda$, the second one to $x_1^2 = \lambda$, which implies $\lambda = x_1^2 = x_2^2$, or $x_1 = x_2 = \lambda^2$. Due to the constraint we get $x_1 = x_2 = \frac{1}{2}$ and along with this $\lambda = \sqrt{\frac{1}{2}} = \frac{1}{\sqrt{2}}$. Thus $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$ is the only critical point being a potential candidate for a maximal point. Now

$$f\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) = \frac{1}{4} > 0 = f(0,\cdot) = f(\cdot,0)$$

and $\mathcal{D}$ compact prove the critical point to be a maximum without even the necessity of checking the second order conditions.

–●

The next example has more than one inequality constraint. However, we will not present a solution, but set the main focus here on showing how the system of equations will look like for the given problem. As the domain of the objective function is $\mathbb{R}^3$, we switch to $x - y - z -$ notation instead of using indices as in the previous sample.

Example 1.69 (Inequality constraints)

$$\begin{aligned} \underset{x,y,z\in\mathbb{R}_{\geq 0}}{\text{maximize}} \quad & f : \mathbb{R}^3 \to \mathbb{R}, (x,y,z) \mapsto xyz && (1.63)\\ \text{s.t.} \quad & x + y + z \leq a \in \mathbb{R}_+. \end{aligned}$$

In this formulation, the constraints $x \geq 0$, $y \geq 0$, $z \geq 0$ and the (bounding) parameter

$a > 0$ are implicitly given. We first rewrite the problem into standard form:

$$\begin{aligned} &\underset{x,y,z\in\mathbb{R}}{\text{maximize}} \quad f:\mathbb{R}^3\to\mathbb{R},(x,y,z)\mapsto xyz \\ &\text{s.t.} \quad \begin{cases} x+y+z-a\le 0 \\ -x\le 0 \\ -y\le 0 \\ -z\le 0. \end{cases}\end{aligned} \tag{1.64}$$

Having four constraints we are given the four one dimensional Kuhn-Tucker multipliers by

$$L_{x,y,z,\lambda_i} = xyz - \lambda_1(x+y+z-a) + \lambda_2 x + \lambda_3 y + \lambda_4 z,$$

$i = 1,2,3,4$ those are with partial derivatives

$$\frac{\partial L_{x,y,z,\lambda_i}}{\partial x} = yz - \lambda_1 + \lambda_2, \quad \frac{\partial L_{x,y,z,\lambda_i}}{\partial y} = xz - \lambda_1 + \lambda_3, \quad \frac{\partial L_{x,y,z,\lambda_i}}{\partial z} = xy - \lambda_1 + \lambda_4.$$

Condition (2) of theorem 1.62 gives

$$\lambda_1(x+y+z-a) = 0, \quad \lambda_2 x = 0, \quad \lambda_3 y = 0, \quad \lambda_4 z = 0. \tag{1.65}$$

Finally, from (1) of the Kuhn-Tucker theorem that $\lambda_i \ge 0$, $i = 1,\dots,4$.

—●

As mentioned in the introduction to the example, we stop here for emphasizing something of more importance than the solution. Equations (1.65) illustrate an interesting fact stemming from condition (2) for the Kuhn-Tucker multipliers in theorem 1.62: The product of any multiplier and its associated inequality constraint needs to be zero. If either of the two multiplicants is not zero, the other one has to be. This is often refered to as **complementary slackness**, where **slackness** can be seen as a kind of relaxation of constraints. We therefore can also say that, if either the constraint or its associated objective variable is slack, the other cannot be. We close this susbection with an example showing that, under violated conditions, the existence of multipliers is not even guaranteed:

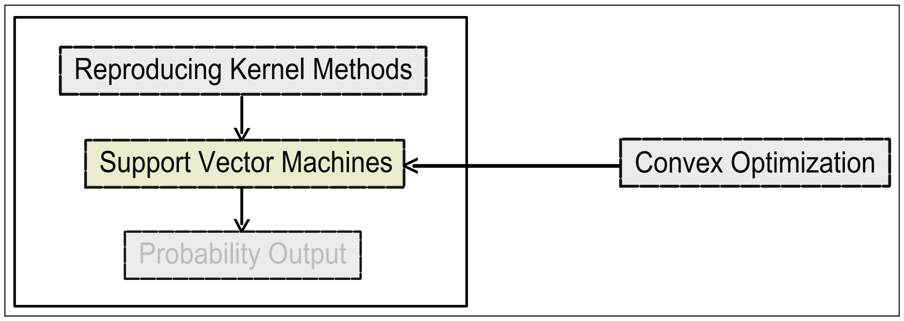
Example 1.70 (Equality constraints: Violated rank condition)
Let $f:\mathbb{R}^2\to\mathbb{R}$, $f(x,y) = -y$ be the projection onto the second component and the $g(x,y): y = x^2 \Leftrightarrow g(x,y) = y - x^2$. We analyze the optimization problem

$$\begin{aligned} \underset{(x,y)\in\mathcal{D}}{\text{maximize}} \quad & f(x,y) \\ \text{s.t.} \quad & y = x^2. \end{aligned} \tag{1.66}$$

First, $x^2 \geq 0$ implies $y \geq 0$ and $y = 0 \Leftrightarrow x = 0$, thus $(0,0)$ is immediately recognized as a (global and local) maximum. However, $\nabla g = (0,0)^T$ at $(0,0)$ and henceforth $\operatorname{rank}(\nabla g(0,0)) = 0 < 1$. Moreover, $\forall (x,y) \in \mathbb{R}^2 : \nabla f = (0,-1)^T$, and we conclude that there cannot exist any vector λ of Lagrangian multipliers satisfying the condition $\nabla f(\mathbf{x}^*) + \sum_{i=1}^{k} \lambda_i \nabla g_i(\mathbf{x}^*) = (0,0)$.

–●

1.7 Support vector classification



We give a brief summarization of the principle of using so-called **support vectors**. Starting with two class problems, we proceed to multi class situations and introduce the idea of using kernels as means of both a way of dealing with the case of not linearly separable classes and using combinations of several kernels to handle data comprised of (inhomogeneous) features of diverse possibly sources. This section is mostly based on [11] and [16].

1.7.1 The binary case

1.7.1.1 Hard-Margin classification

Support vector classification is a method to separate data by a decision function which is optimal in the sense of generalization for unseen data: the margin – the distance of the decision function to the closest point of both datasets – should be maximized to allow for optimal classification especially in adjacent regions. As this formulation already suggests, we are for now considering the case of (linearly) separable data.

Following definition (1.58) and the geometric interpretation of a hyperplane, it makes sense to use, for any $\mathbf{x}_i \in \mathcal{H}$, the orientation of $\langle \mathbf{w}, \mathbf{x}_i \rangle + b$ as a decision criterion, which we will denote by $d(\mathbf{x}_i|\mathbf{w}) = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$. The result is a value $y_i (\operatorname{sgn}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)) = 1$ for class labels $y_i \in \{\pm 1\}$, $i = 1, \dots, m$. In other words, for $||\mathbf{w}|| = 1$, $y_i d(\mathbf{x}_i|\mathbf{w})$ is the distance of $\mathbf{x}_i$ to the hyperplane and $d(\mathbf{x}_i|\mathbf{w})$ classifies samples $\mathbf{x}_i$ into either class 1 or -1.

The basic idea behind support vector classification is to maximize the margin from samples to the hyperplane separating the classes, where the margin is the smallest distance of $\mathbf{w}$ to any (training) sample. One can show that this is equivalent to minimizing $||\mathbf{w}||$ and leads to the constrained optimization problem

$$\begin{aligned} \underset{\mathbf{w}\in\mathcal{H},\, b\in\mathbb{R}}{\text{minimize}} \quad & \frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2 \\ \text{s.t.} \quad & y_i \left(\operatorname{sgn}\left(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right)\right) \geq 1,\ i = 1, \ldots, m\ , \end{aligned} \tag{1.67}$$

where – in this case – $f_0 = \frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2$ is the objective function to be minimized subject to the given constraints[12] and m the number of training samples. Forming the Lagrangian Dual and its derivations w.r.t. b and $\mathbf{w}$ (with the aim of canceling them out) and substituting those results back into the Lagrangian dual, setting $\mathbf{y} = (y_1 \cdots y_m)$ we get the dual form of the optimization problem

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle \\ \text{s.t.} \quad & \begin{cases} \boldsymbol{\alpha} \geq 0 \\ \langle \boldsymbol{\alpha}, \mathbf{y}\rangle = 0. \end{cases} \end{aligned} \tag{1.68}$$

The vector inequalities $\boldsymbol{\alpha} \geq 0$ and $\zeta \geq 0$ are to be understood elementwise – $\alpha_i \geq 0$ and $\zeta_i \geq 0$ for $i = 1, \ldots, m$ – and the decision criterion given a (new) pattern $\mathbf{x}$ for the dual form in this context becomes

$$d(\mathbf{x}|\mathbf{w}) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \left\langle \mathbf{x}, \mathbf{x}_i\right\rangle + b\right). \tag{1.69}$$

1.7.1.2 Soft-Margin classification

Hard-Margin classification lacks the ability of generalization especially for samples near the separating hyperplane; the most widely used and as a standard accepted improvement was suggested by [17]. They introduced slack variables $\zeta_i \geq 1$, $i = 1, \ldots, m$ – one for each sample – which relax the constraints and thus allow for a better generalization of the optimization problem. The decision criteria changes accordingly to $y_i d(\mathbf{x}_i|\mathbf{w}) \geq 1 - \zeta_i$, $i = 1, \ldots, m$. In this setting, training samples $\mathbf{x}_i$ with slacks $0 < \zeta_i < 1$ are classified correctly even though not satisfying the maximum margin property. Data with $\zeta_i \geq 1$ is classified incorrectly by the (optimal) hyperplane. Clearly, if the slack variables tend to zero, this converges towards the hard margin classifier.

[12]The reader might ask why 1 is used as the right side of the inequality in 1.7.1.1, whereas the hyperplane is defined as $\{\langle \mathbf{w}, \mathbf{x}\rangle + b = 0 \,|\, \mathbf{x} \in \mathcal{H}\}$. In fact, any positive number would work, and this is simply a kind of normalization.

On the other hand they can always be chosen large enough to make the inequalities feasible, and to prevent this from happening, the slack variables – together with a weight factor c which serves as a trade-off between the generalization capability or number of misclassified samples and the deviation of samples from the maximum margin – are used as a penalty term in the objective function. The primal form is then given as

$$\begin{aligned} &\underset{\mathbf{w}\in\mathcal{H},\boldsymbol{\zeta}\in\mathbb{R}^m,b\in\mathbb{R}}{\text{minimize}} && \frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2 + \frac{c}{m}\left\langle \mathbf{1},\boldsymbol{\zeta}\right\rangle \\ &\text{s.t.} && \begin{cases} y_i(\langle \mathbf{w},\mathbf{x}_i\rangle + b) \geq 1-\zeta_i,\ i=1,\dots,m \\ \boldsymbol{\zeta}\geq 0 \end{cases}, \end{aligned} \tag{1.70}$$

and dualization leads to the form

$$\begin{aligned} &\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} && \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left\langle \mathbf{x}_i,\mathbf{x}_j\right\rangle \\ &\text{s.t.} && \begin{cases} \langle \mathbf{y},\boldsymbol{\alpha}\rangle = 0 \\ 0 \leq \boldsymbol{\alpha} \leq C \end{cases}., \end{aligned} \tag{1.71}$$

where $C \in \mathbb{R}^m$ is a (vector valued) boundary parameter. The α_i are the Lagrangian multipliers introduced for solving the optimization problem under the given constraints and eliminating $\mathbf{w}$, b and $\boldsymbol{\zeta}$ via differentiation and substitution. In this form of the SVMs we can substitute the inner product of the objective function by a reproducing kernel k, and the new decision function becomes

$$d(\mathbf{x}|\mathbf{w}) = \sum_{i=1}^{m}\alpha_i y_i k(\mathbf{w},\mathbf{x}_i) + b = 0. \tag{1.72}$$

Again, the sign of $d(\mathbf{x}|\mathbf{w})$ equals the label of the classification of $\mathbf{x}$, which is unclassifiable for $d(\mathbf{x}|\mathbf{w}) = 0$. The main reason for using kernels stems from the fact that the RKHS determined by the reproducing kernel in general has a dimension different from that of the original space. In the context of support vector classification this is of utmost importance, as data which is not linearly separable in the original space can potentially be linearly separable in the (new) RKHS. Figure 1.2 illustrates this effect.

1.7.2 Learning SV classifiers for large datasets

In addition to this standard method, that it solving the above described optimization problem, further training methods have been developed, most of them focusing on the problem of large training datasets. Though we deal with large training datasets in the

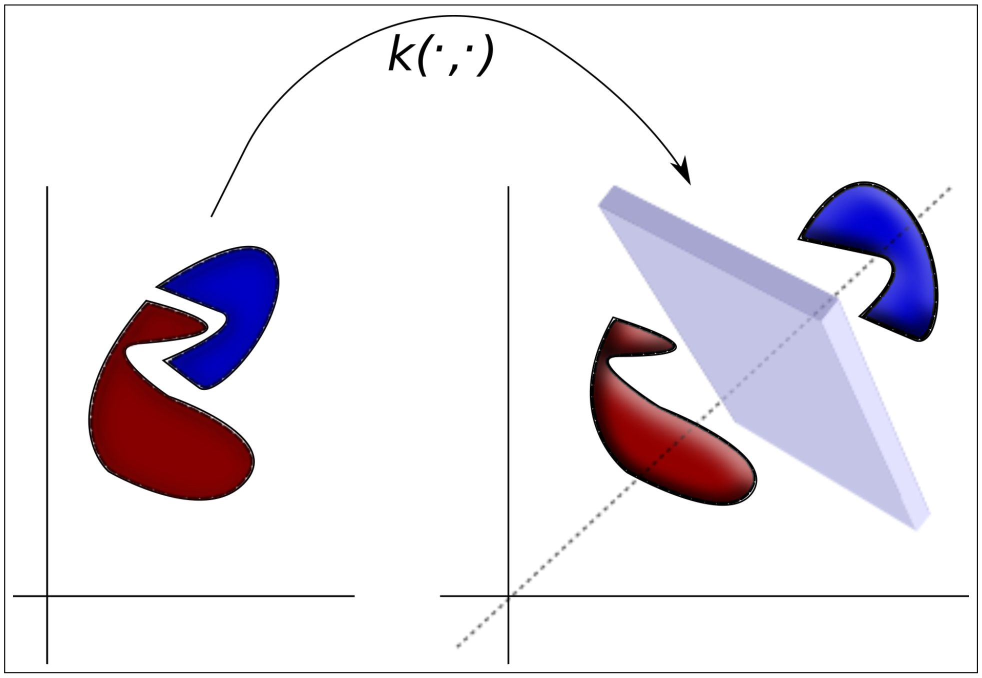


Figure 1.2: The figure shows two classes that are interwoven and thus not linearly separable in the two dimensional plane. Using an appropriate transform such as a reproducing kernel $k(\cdot,\cdot)$, linear separability of the data, here by a two dimensional hyperplane which for the sake of illustration appears three dimensional, can be achieved. In general the shapes of such clusters in the image space changes with k, but for reasons of clarity we present them basically unaltered.

herein addressed topics of phoneme classification and speech recognition, those methods have not been part of our research so far. Nevertheless we wanted mention this issue here and refer the interested reader to [16] for a detailed and great presentation of the topic.

1.7.3 Interpreting results and performing classification

Let us substantiate and state a little more precisely the aspect of what, after optimization, the support vectors really are and how they serve for classifying samples. We remember the constraints of optimization problem 1.7.1.1,

$$y_i \left(\operatorname{sgn}\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right)\right) \geq 1, \; i = 1, \ldots, m \; . \tag{1.73}$$

Support Vectors are exactly those vectors $\mathbf{x}_i$ satisfying the constraints. Being orthogonal to the optimal separating hyperplane, they *support* it in a figurative sense. To see this, for euclidean metrics the distance from any training data $\mathbf{x}$ to the hyperplane is

$$\frac{|\langle \mathbf{w}, \mathbf{x} \rangle + b|}{||\mathbf{w}||},$$

which remains unaltered when leaving out those training vectors in equation 1.73 that meet the inequality only. For details, see [16], chapter 2.1.

1.7.4 Implementational aspects

Optimizing a support vector classifier as described above delivers a quadratic programming optimization problem. Solving this naïve the way is in general infeasible due to the size of most classification situations. Currently, most common SVM implementations are based on **S***equential* **M***inimal* **O***ptimization* (SMO), which was introduced by Platt in [18]. The author relates his solution to both, a publication by Osuna ([19]) and the algorithmic ideas of *Bregman-* and *row-action* methods ([20], [21]).

The first cited work splits the original optimization problem into smaller subproblems, the other solve convex programs with linear constraints, reducing the number of constraints to one per iteration step. Improvements to Platt's algorithm are for instance suggested by [22], overcoming the problems resulting from the use of a single adaptive threshold β throughout the iterations, where the main problems lie in Platt's ways of computing and using β. For details, the reader is suggested to consult the mentioned works. As an important consequence we note that, thanks to those concepts, large SVM problems can now be addressed and managed computationally.

1.7.5 The multiclass case

When progressing to classification problems covering more then two classes, the question of how to extend from binary to n-ary support vector classifiers arises naturally. Following our intuition, one might approach the problem by simply extending the binary class to a multiclass decision problem, modifying the optimization problem itself accordingly. The result for K classes would thus be the quadratic program

$$\begin{aligned} \underset{\mathbf{w}_k \in \mathcal{H}, \boldsymbol{\zeta}^k \in \mathbb{R}^m, b_k \in \mathbb{R}, 1 \le k \le K}{\text{minimize}} \quad & \frac{1}{2} \sum_{k=1}^{K} ||\mathbf{w}_k||^2 + \frac{c}{m} \sum_{i=1}^{m} \sum_{k \neq i} \zeta_i^k \\ \text{s.t.} \quad & \begin{cases} \langle \mathbf{w}_i, \mathbf{x}_i \rangle + b_i \ge \langle \mathbf{w}_k, \mathbf{x}_i \rangle + b_k + 2 - \zeta_i^k \\ \boldsymbol{\zeta}^k \ge 0 \end{cases} . \end{aligned} \tag{1.74}$$

The drawback of this concept is, that the optimization has to consider all support vectors simultaneously throughout its process. Not only does computation time needed for problems with larger amounts of samples and classes grow considerably, but also does the final set of support vectors in general tend to be very large. Thus, given the amount of time and space necessary for the direct way, most situations require a different approach.

Well known methods for discriminative decisions are **one-vs-all** (often also called **one versus the rest**) and **one-vs-one** (sometimes also referred to as a **pairwise** approach), both of which can be applied within the support vector context. In the first case, K hyperplanes are build that separate the individual classes – one at a time – from the remaining, pooled ones. The vote will be for that decision function (or class) which returns the highest value: given $d_k(\cdot|\mathbf{w}_k)$, $k = 1, \ldots, K$ a (new) sample $\mathbf{x}$ will be classified into class

$$\text{argmax}_{k=1}^{K} d_k(\mathbf{x}|\mathbf{w}_k). \tag{1.75}$$

When applying the latter rule, one set of support vectors for each of the possible unordered tuples of classes is computed, resulting in $\frac{K(K-1)}{2}$ decision functions. The overall decision is that of the maximum of the sum of positive decisions for one class.

Both methods have certain advantages and disadvantages and in general yield areas, where a decision is not possible without further assumptions. While the number of decision functions in the *one-vs-one* strategy grows large quickly (quadratic in K), the sets of training samples is small compared to the *one-vs-all* setting. As a consequence, the number of support vectors defining the final decision hyperplane of each binary problem is smaller as well. In addition to that, collating the samples often leads to considerably imbalanced decision systems. For detailed discussion of multiclass support vector classification, drawbacks and possible solutions or workarounds of those approaches, the reader is referred to the literature mentioned at the start of section 1.7, [11] and [16].

1.8 Combining kernels for SVM-classification

In section 1.4.1 we illustrated how reproducing kernels can be constructed for instance by mathematical operations to create new function spaces, and section 1.4.4 introduced concepts for using reproducing kernel Hilbert spaces to address problems in areas of classification such as of pattern recognition and data discrimination. In the context of support vector classification, the diverse ways of construction (summation, composition, using positive definit matrices, ...) for different types of data have been successfully implemented for and applied to various tasks. Learning with such kernels is frequently referred to as *mutiple kernel learning* (MKL), and its motivation derives from the fact that similarity of different kinds of data is often best represented by specific reproducing kernels and their induced metrics. Reproducing kernel methods for feature vectors comprised of heterogeneous information have to take the kernel aggregation into account and eventually need to be modified. In this section we briefly review some MKL-concepts and their alterations to the SVM-optimization problem.

A basic prerequisite is that the reproducing kernels live in the same function space. In other words, they need to be defined on the same set – a necessary condition that we already were confronted with when pointing out the application of the linear transform reproducing kernel in section 1.5.1. Considering M reproducing kernels (or their respective Gramians) $k_j \in \mathbb{K}^{n \times n}$, $j = 1, ..., M$, we state the MKL problem as finding a (linear or convex) combination $\sum_{j=1}^{M} \kappa_j k_j(\cdot,\cdot), \kappa_j \geq 0$ optimal in the sense of data separation. To get a visual impression of the impact of convex kernel combination on decision boundaries, we examine the effect for a simple binary support vector classification task upon the two most frequent phonemes in the TIMIT database, *ae* and *ix*. We trained a separating hyperplane on the first two components of the feature vectors, chosen due to results of a principal component analysis of the data. The images in table 1.8 illustrate the alterations of the decision boundary for convex combinations of an exponential rbf kernel and a linear one:

$$k(\mathbf{x}, \mathbf{z}) = \alpha k_{rbf}(\mathbf{x}, \mathbf{z}) + (1 - \alpha) k_l(\mathbf{x}, \mathbf{z}), \quad \alpha \in [0, 1].$$

The larger the parameter γ of the rbf kernel, the more the decision boundary fits the training set (see [11] or [9]), resulting in absolute overfitting by learning each single sample for high values. This is due to the basic nature of exponential kernels: As the dimension of the corresponding reproducing Hilbert space is infinite, the hyperplane can adapt to the training set, leading to bad generalization for unseen samples.

The figure also points out the main qualities of the relation between the weight α and the rbf-parameter γ:

1) It can clearly be seen that the linear kernel "linearizes" the decision boundary when its weight is increased, thereby lessening the overfitting effect of the exponential kernel.

2) The figure reflects a "global" versus "local" characteristic. It depicts the synergy of the combinatorial approach in general and unveils the reasons why it is attractive for research on a larger scale.

For a good choice of the parameters, the balance between global accuracy and local generalization capability needs to guarantee good characteristics not only considering separation but also in the sense of classifying unseen samples.

Section 3 shows results of experiments that compare single kernel to two convex combined reproducing kernels, performed on a subset of the TIMIT phoneme dataset. Its main purpose is to evaluate the quality of both the single/multiple kernel learning and the MFCC (auto)correlation (see section 2.5.2) versus the common MFCC features and to conclude which setup to use for approaching continuous speech recognition.

1.8.1 Multiple kernel learning

The most common concept for optimizing kernel combinations within the SVM context follows the strategy of optimizing coefficients of a weighted, convex or linear sums of the kernels in question such, that the margin of the resulting kernel is optimal in the common SVM-sense.

In [23], the authors derive a solution for learning such an optimal linear combination. Given I data samples (x_i, y_i), $1 \leq i \leq I$, where x_i are elements of some feature space and $y_i \in \{+1, -1\}$ class labels, we seek to optimize the linear combination

$$k^M = \sum_{j=1}^{M} \kappa_j k_j, k_j \in \mathbb{R}^{n \times n} \tag{1.76}$$

of M reproducing kernels k_j and weighting coefficients $\kappa_j \in \mathbb{R}_+$ constraint to $\text{trace}\{\sum_{j=1}^{M} \kappa_j k_j\} = c$, $c \in \mathbb{R}_+$ constant. For this problem, the authors formulate the following quadratic programming optimization problem:

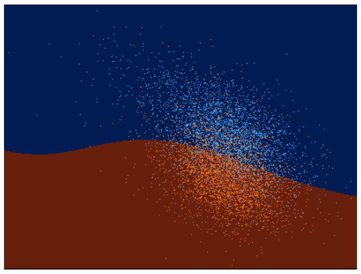
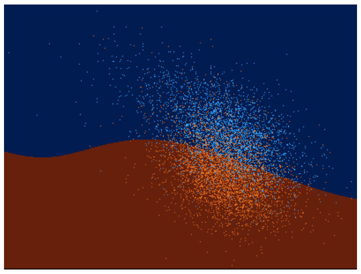
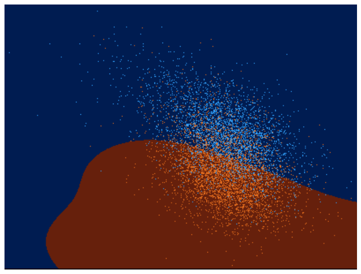
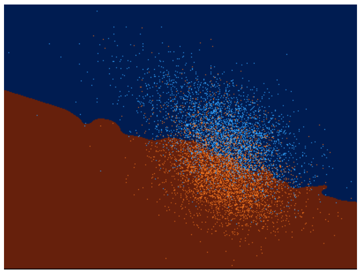
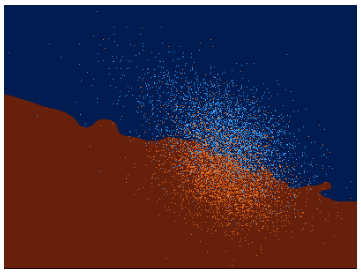
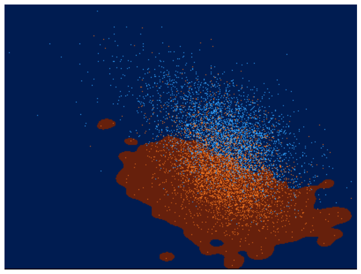
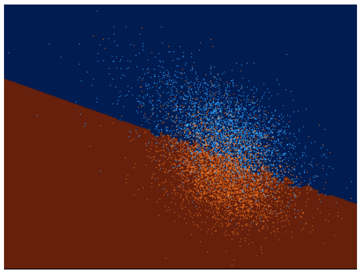
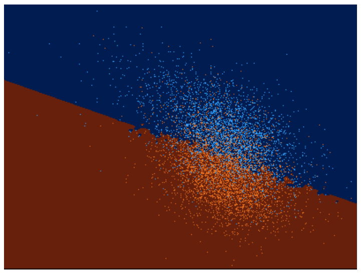
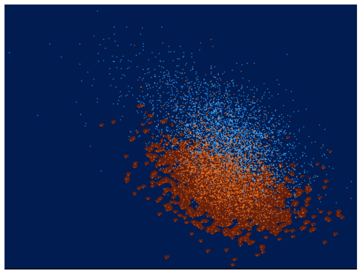
	$\alpha = 0.95$	$\alpha = 0.975$	$\alpha = 1$
$\gamma = 0.01$			
$\gamma = 1$			
$\gamma = 11$			

Table 1.1: Visualization of the decision boundaries for the training data of two phonemes from TIMIT database, *ae* (red) and *ix* (blue). The table of figures depicts the combined effects of the kernel parameter γ of the exponential rbf kernel and the weight parameter α of the convex kernel combination.

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^n,\theta\in\mathbb{R}}{\text{minimize}} \quad & \theta - 2\boldsymbol{\alpha}^T\mathbf{1}_n \\ \text{s.t.} \quad & \begin{cases} 0 \leq \boldsymbol{\alpha} \leq C, \boldsymbol{\alpha}^T y = 0, \\ \boldsymbol{\alpha}^T \text{diag}(\mathbf{y}) k_j \text{diag}(\mathbf{y}) \boldsymbol{\alpha} \leq \frac{\text{trace}\{k_j\}}{c}\theta, j \in \{1, \ldots, M\}. \end{cases} \end{aligned} \tag{1.77}$$

Here $\mathbf{y}$ is the collocate label vector and diag($\mathbf{y}$) the diagonal matrix formed by $\mathbf{y}$. The coefficients κ_j from equation 1.76 are computed as the Lagrangian multipliers given the M constraints

$$\boldsymbol{\alpha}^T \text{diag}(\mathbf{y}) k_j \text{diag}(\mathbf{y}) \boldsymbol{\alpha} \leq \frac{\text{trace}\{k_j\}}{c}\theta.$$

It should be pointed out that this approach actually learns a compound kernel matrix by parametrization of given subkernels and can be cast into a convex semidefinite optimization problem (SDP) by dualization. By further constraining the kernel weights κ_j to be non-negative, normalizing the kernel matrices k_j to 1 and considering conic kernel combination, the SDP reduces to a quadratically constrained quadratic program (QCDP), which is subject of research in [24]. The final formulation derived in their work is

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^n, t\in\mathbb{R}}{\text{minimize}} \quad & 2\boldsymbol{\alpha}^T\mathbf{1}_n - ct \\ \text{s.t.} \quad & \begin{cases} \frac{1}{I}\boldsymbol{\alpha}^T\text{diag}(\mathbf{y})k_j\text{diag}(\mathbf{y})\boldsymbol{\alpha} \leq t, \\ \boldsymbol{\alpha}^T\mathbf{y} = 0, 0 \leq \boldsymbol{\alpha} \leq C, \end{cases} \end{aligned} \tag{1.78}$$

where I is again the number of samples and $t \in \mathbb{R}_+$ bounds the terms

$$\boldsymbol{\alpha}^T\text{diag}(\mathbf{y})k_j\text{diag}(\mathbf{y})\boldsymbol{\alpha}.$$

The authors of [25] extend those concepts and propose a slightly modified problem formulation, the dual of which is equivalent to (1.77). Casting this into a convex form and using a special kind of regularization (as the problem itself is not differentiable), the work furthermore provides an SMO-like approach for solving the optimization problem. Nonetheless, the MKL-problem in all those approaches boils down to a grid parameter search and has cubic complexity with respect to the amount of samples. Due to this fact as well as both, the size of the final combined kernel matrix k^m and the complexity of the optimization problem, those techniques are intractable when dealing with large number of kernels and especially for large training sets.

Diego et al. ([26]) suggested a kind of averaging function taking into account distance information of different kernels on given datasets. For two kernels, they proposed to define a new kernel

$$k^{1,2} = \frac{1}{2}(k_1 + k_2) + f(k_1 - k_2). \tag{1.79}$$

As before, $\mathbf{y}$ is the compound vector of all labels. One specialization for f is

$$f(k_1, k_2) = \tau\text{diag}(\mathbf{y})\left|k_1 - k_2\right|\text{diag}(\mathbf{y}),$$

which is tantamount to common *minimum* and *maximum* functions

$$\min, \max(x, y) = \frac{1}{2}(x + y) \mp \frac{1}{2}\left|x - z\right|.$$

For samples i, j of different classes we have $y_i y_j = -1$, and the new kernel equals the minimum $\min(k_1(i,j), k_2(i,j))$. For identical classes the function delivers $y_i y_j = 1$ and henceforth produces a maximum-like choice. τ is an additional weighting factor and influences the final decision boundary; further choices of f are discussed in the article as well as the extension to three or more kernels. This approach however does not address the problem of really optimizing the margin with respect to the kernel combination, and especially for several kernels and heterogeneous data it has to be assumed that the approach will not adequately represent the data metrics and separate/ classify well. The results presented in the publication give reason to this judgment.

In [27], the authors propose two ideas. The first one optimizes an MKL-problem using an interclass score which *is the ratio of the total within-class standard deviation in the direction between the class means to the distance between the class means*,

$$FSM(k) = \frac{\sigma_+ + \sigma_-}{||\phi_+^k - \phi_-^k||}.$$

σ_+ and σ_- are the standard deviations of the positive and negative class, ϕ_+^k and ϕ_-^k the centers in the feature space. The new problem unveils itself as

$$\begin{aligned} \underset{\kappa_j}{\text{minimize}} \quad & FSM\Big(\sum_{j=1}^{M} \kappa_j k_j\Big) \\ \text{s.t.} \quad & \sum_{j=1}^{M} \kappa_j = 1, \kappa_j \in \mathbb{R}_+ \end{aligned} \tag{1.80}$$

It has quadratic runtime complexity and the FSM-ratio is easily and quickly computed.

The second approach builds on choosing a subset of "important" kernels from the set of all kernels in use. The weights are used in a convex combination and chosen proportionally to a quality estimation from a 10 fold cross validation, applying the kernels one by one to the given task.

Based on the comparison to a threshold[13] δ, the authors decide whether to keep or to reject the kernel in question. Setting

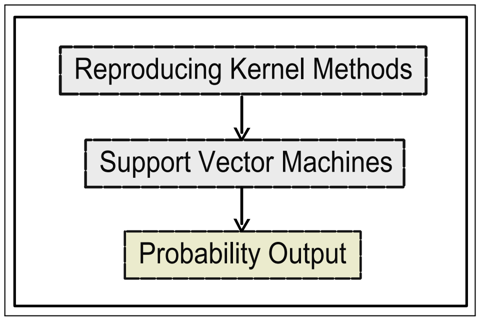
[13]The authors do not give any information about how they achieved the value of the threshold δ.

$$\begin{cases} \kappa_i = \frac{\mathrm{FSM}(k_i)-\delta}{\sum_{j=1}^{M}\mathrm{FSM}(k_j)-M\delta} & \mathrm{FSM}(k_i) > \delta \\ \kappa_i = 0 & \text{otherwise}, \end{cases} \tag{1.81}$$

the new **proportionally weighted multiple kernel** is defined as

$$k^{\mathrm{pwmk}} = \sum_{j=1}^{M} \kappa_j k_j. \tag{1.82}$$

1.9 Probabilistic output for support vector classifiers



Evaluating samples by support vector classifiers produces vectors or scalars according to the target space of the underlying decision function $d(\mathbf{x}|\mathbf{w})$ such as given for instance in equations (1.72), (1.69) and (1.75). The signature of the decision function's result for a given input vector $\mathbf{x}$ serves as a final classification criteria, see section 1.7 and subsequent subsections.

Many structures and algorithms however are based on probabilistic data rather than on deterministic values computed by a decision function. We will see that this is also true for Hidden Markov Models (HMMs), which play an important role in speech recognition as well as in other areas of research and will be revisited in section 2.3. SVMs on the other hand neither deliver any kind of probabilistic output, nor do they in general distinguish any kind of states within nonstatic or (non)stationary processes[14].

In this chapter we present concepts of generating probabilistic output given the result of such $d(\mathbf{x}|\mathbf{w})$ with the aim of using those as probabilities within HMM classification. Focussing on the binary case, section 1.9.1 introduces an algorithm developed by Platt ([28]), making use of a sigmoidal function in the neighbourhood of the SVM's (linear) hyperplane to serve as a probability estimator. The authors of [29] improve this approach by both changing the optimization method to guarantee convergence and eliminating certain numerical cancellation problems leading to instabilities within Platt's algorithm. Following this, the chapter closes with section 1.9.2, which reviews the methods presented in [30],

[14]As mentioned before, there are certain kernel types like HMM-kernels developed to address this problem. Standard numeric kernels such as those used in this work do however not include any information about nonstatic, nonstationary or state-transitional aspects of the data. This is the main reason and motivation for introducing the correlation features in section 2.5. Due to their way of creation, they include a certain amount of inter-timeframe and phoneme-length related information.

which take the mentioned approach one step further to estimation of SVM-based probabilistic output in multiclass classification environments.

1.9.1 Probabilistic output for SVMs using a sigmoid function

Again, let $\mathbf{x}_i, i = 1, \dots, N$ enumerate the trainingset, $\mathbf{x}$ be an arbitrary vector in the classification input space (including training-, validation- and testset), $y, y_i \in \{+1, -1\}$ be class labels and N_+, N_- denote the number of training samples of the respective classes. Platt's idea is to approximate posterior probabilities based on a sigmoid probability function

$$P_{a,b}(d(\mathbf{x}|\mathbf{w})) = \frac{1}{1 + \exp(a \cdot d(\mathbf{x}|\mathbf{w}) + b)} \approx p(y = +1|x), \tag{1.83}$$

using the results of the SVM's decision function $d(\cdot|\mathbf{w})$. Writing $p_i = P_{a,b}(d(\mathbf{x}_i|\mathbf{w}))$, the parameters a, b are computed by minimizing the negative log-likelihood via a gradient descent method in order to solve the minimization problem

$$\underset{a,b\in\mathbb{R}}{\text{minimize}} \quad -\sum_{i=1}^{N} \left[t_i \log(p_i) + (1 - t_i) \log(1 - p_i)\right], \tag{1.84}$$

where

$$t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} & y_i = +1 \\ \frac{1}{N_- + 2} & y_i = -1 \end{cases}, \quad i = 1, \dots, N.$$

He furthermore suggests two methods to prevent (or to keep as small as possible) the distribution bias of the decision functions evaluated on the training set parameter estimation in the context of his algorithm. The first one splits the training data 70% to 30% into two sets of vectors. The larger set is used for SVM-training whereas the smaller one serves for estimation of the probability function's parameters a, b or even for further kernel parameter optimization and a follow-up retraining of the SVM afterwards.

Platt favors a three-fold crossvalidation method. Splitting the training data into three sets, the classifiers are trained on all possible tuples of permutation, the evaluation being performed on the remaining respective third set. The union of those evaluations serve for parameter estimation of the sigmoid function. While this method is about 1.5 times slower ([28], page 6) than the first one, the amount of data used for parameter estimation is greater and thus the overall variance estimate smaller. As described in the experiments section 3.3, we follow both bias suggestion in our work, splitting the training set 70%

to 30% for SVM training and parameter estimation and performing training applying *leave-one-out* cross validation strategy. However, we do not use the smaller dataset for any additional kernel or SVM-parameter optimization.

In [29] it is pointed out that Platt's implementation basically equals an unconstrained optimization problem where the stepsize of the gradient descent in each iterative step is deduced and used directly. The authors elude that this can lead to convergence problems and modify the optimization method to implicitly control the stepsize. The fact that the Hessian matrix representing the optimization process and used in the gradient direction computation is positive semidefinite motivates their decision for a Newton method with backtracking line search. At the cost of some additional complexity, this method guarantees convergence.

Another problem in Platt's formulation is the numerical problem invoked by the cancellation when calculating $1 - \frac{1}{1+\exp(a \cdot d(\mathbf{x}_i|\mathbf{w})+b)} = 1 - p_i$. The authors suggest to rewrite the summand of the objective function in problem (1.84) as

$$\begin{aligned} & [t_i \log(p_i) + (1-t_i)\log(1-p_i)] && (1.85) \\ = \; & (t_i - 1)(a \cdot d(\mathbf{x}_i|\mathbf{w}) + b) + \log\left(1 + \exp(a \cdot d(\mathbf{x}_i|\mathbf{w}) + b)\right) && (1.86) \\ = \; & t_i(a \cdot d(\mathbf{x}_i \mathbf{w}) + b) + \log\left(1 + \exp(-a \cdot d(\mathbf{x}_i|\mathbf{w}) - b)\right), && (1.87) \end{aligned}$$

where the problematic term $1 - p_i$ does not occur any longer. Furthermore, p_i can be replaced by the equivalent and numerically more accurate expression

$$\frac{\exp(a \cdot d(\mathbf{x}_i|\mathbf{w}) + b)}{1 + \exp(a \cdot d(\mathbf{x}_i|\mathbf{w}) + b)}.$$

1.9.2 Probabilistic output for SVM based multiclass systems

The goal of the algorithm presented in [30] is the estimation of probabilities

$$p_i = P(y = i|\mathbf{x})$$

for an (unseen) given sample $\mathbf{x}$ belonging to class $i, i = 1, \ldots, M$. Given (estimates of) pairwise class probabilities

$$r_{i,j} = P(y = i|\mathbf{x}, y = i \text{ or } j),$$

the authors discuss various existing and introduce modifications leading to two new methods, the second of which we use in this work. The pairwise class probabilities suggested are those presented in the work of [29], which also is implemented for the experiments conducted for this thesis.

The proposed algorithm is based on the idea of minimizing the Kullback-Leibler distance

$$\sum_{i=1}^{M} \sum_{j\neq i, j=1}^{M} (M_i + M_j) r_{i,j} \log \frac{r_{i,j}}{\mu_{i,j}}$$

between the approximations $r_{i,j}$ and the real class probabilities $\mu_{i,j} = \frac{p_i}{p_i+p_j}$. M_i and M_j are the numbers of training samples of classes i, j respectively. This approach was introduced in [31], and after partial derivation with respect to the class probabilities p_i, $i = 1, \ldots, M$, the problem can be reformulated as finding a point such that

$$\sum_{i=1}^{M} \sum_{j\neq i, j=1}^{M} (M_i + M_j) r_{i,j} = \sum_{i=1}^{M} \sum_{j\neq i, j=1}^{M} (M_i + M_j) \mu_{i,j}, \tag{1.88}$$

constrained to $\sum_{i=1}^{M} p_i = 1$ and $p_i > 0$. An iterative gradient algorithm is proposed, where in each iteration step only a single component i is updated while others remain unchanged.

The sequence of points generated in this algorithm has strictly decreasing Kullback-Leibler distance but does not guarantee the convergence to a point satisfying equation (1.88), see also [32]. Under the premise of rather balanced class sizes[15], $M_i + M_j = 2/M$, denoting the solution as a vector $\mathbf{p}$ of multi-class probability estimates, Wu et al prove that solving the optimization problem

$$\begin{aligned} \underset{\mathbf{p}}{\text{minimize}} \quad & \sum_{i=1}^{M} \sum_{j\neq i, j=1}^{M} (r_{j,i} p_i - r_{i,j} p_j)^2 \\ \text{s.t.} \quad & \sum_{i=1}^{M} p_i = 1 \end{aligned} \tag{1.89}$$

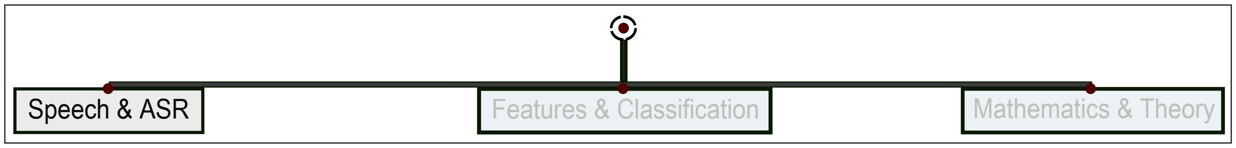
guarantees a unique solution in the sense of equation (1.88). The classifier's decision function in this case is as simple as

$$\text{argmax}_i(p_i). \tag{1.90}$$

[15]See section 2.6 for a short comment concerning this problem.

The proposed new algorithm boils down to solving a system of linear equations using Gaussian elimination or, with small modifications to meet the preliminarity of positive definiteness, Cholesky factorization – for further details, the reader is referred to the above mentioned publications.

Chapter 2 – Speech Features and Phoneme Classification



When performing speech synthesis, speech recognition or phoneme classification it is indispensable to have knowledge about the basics of the nature of speech production. The reason for this is simply that being aware of certain characteristics offers both a starting point and paths to follow when analyzing speech. This knowledge stems from both mathematical models of the glottal system or the human auditory system and from research in psycho-acoustic aspects of the reception of sound, pitch and volume.

This chapter begins by offering a brief overview over the concepts in the field of speech such as acoustic data modeling and extraction of characteristic features. The presented material is detailed enough to include all material necessary to understand the foundations of the ideas of our work without going to deep. The first section revisits facts and characteristics of speech resulting from mathematical models simulating the physical aspects of the glottal and vocal systems of human beings. The results had an immediate impact on the way today's features for speech recognition and phoneme classification were developed and are used.

The first section briefly illustrates certain base aspects when modeling natural, human speech and illuminates certain aspects of its production. Characteristics, such as short time stationarity, that determine or influence concepts described in subsequent sections, thereby building their foundations, are deduced.

Section 2.2 is dedicated to the description of one of those features, Mel-Frequency Cepstrum Coefficients (MFCCs). They also serve as a base for the new kind of features developed in this work and above that are widely used in areas of research not related to speech recognition and alike. Based on a filter bank addressing both characteristics of the human vocal tract as described in section 2.1 and psycho-acoustic aspects of human sound perception, they aim at catching the nature of speech production and recognition, or – within contexts different from speech recognition or similar topics – more general that of short time quasi stationary processes and their interpretation.

Following this, **H**idden **M**arkov **M**odels are introduced in section 2.3. They offer a reasonable way of representing many of the characteristics derived in section 2.1 and are common and widely used in the field of speech processing as well as in other areas that involve modeling state-like behaviour of some kind. The subsequent section shortly reviews some fundamental basics of acoustical and language models utilized in automatic speech recognition systems. This is only done to an extent sufficient for our research, which aims at the integration of specifically computed probability values from the *SME* setup into such a system, allowing to compare the results to the common MFCC variant.

We proceed with section 2.5, which details the main concepts and train of thoughts. Revealing the main motivation for the path this thesis pursues, the first subsection prepares the reader for the succeeding section by clarifying the pivotal keynote and its application leading to correlation data. Consequently, the chapter concludes with its final and main part. Starting with the substitution of the Δ and $\Delta\Delta$ parts of MFCC vectors with autocorrelation features and linearization of the latter by correlation with specific fix, representative vectors, the section unifies the distinct layers of the correlation feature itself, a classifier topology built (both topics covered in section 2.5.3) in the light of a tristate topology and a specific reproducing kernel (section 1.5) deduced from the correlation operation as well into one, thereby putting together the individual pieces.

2.1 The nature of speech

This subsection is basically a short description of the derivations given in [33] describing the modeling of the vocal tract and the consequences for speech representation under certain simplifying assumptions on the model.

The book's model follows concepts given by Sondhi [34] and Portnoff [35]. Here, the vocal tract is roughly separated into the glottal source, the oral cavity and the acoustic impedance resulting from the lips of the speaker. The oral cavity is assumed to be a lossless acoustic tube, the cross-sectional areas of which vary slowly both in time and space, referenced by t, x respectively. Further restricting the model to wave propagation in one direction, pressure $p(x, t)$ and velocity $u(x, t)$ of the speech produced can be described by two differential equations,

$$-\frac{\delta p}{\delta x} = \frac{\delta u}{\delta t}\frac{\varrho}{A(x,t)} - \frac{\delta u}{\delta x} = \frac{\delta p}{\delta t}\frac{A(x,t)}{\varrho c^2}, \tag{2.1}$$

where ϱ reflects the equilibrium density[1] of air in the tube and c is the corresponding sound velocity. Differentiation of the equations 2.1 leads to the *Webster equation*

$$\frac{\delta^2 p}{\delta x^2} + \frac{\delta p}{\delta x}\frac{\delta A}{\delta x}\frac{1}{A(x,t)}. \tag{2.2}$$

Based on this, Levinson (cf. [33]) shows that a (discrete) sinusoidal steady-state transfer function can be derived for the speech signal in the acoustic tube, which even includes the effects of thermal, viscous and wall losses under certain constraints such as boundary conditions for the mouth model.

As a function of time, the speech $p(t)$ signal stemming from the pressure function is a solution of the Webster equation. A detailed analysis, given in the above mentioned resource, reveals, that $p(t)$ in general is non-stationary due to the fact that $A(x, t)$ is a continuously time-varying function including random influences and change. However, according to the model, the variation of $A(x, t)$ is slow with respect to $p(t)$ – that is, $\left|\frac{\delta A}{\delta t}\right| \ll \left|\frac{\delta p}{\delta t}\right|$. As a result of this, $p(t)$ is approximately piecewise stationary and allows for speech to be regarded and treated as a sequence of short-time stationary chunks.

[1]The (air)pressure inside the tube is the ratio of the surface force acting onto the air and the tube area. In this simplified model it decomposes into two components: the equilibrium density component, ϱ, which is a constant, and a pressure disturbance varying in time and space, implicitly given by $p(x, t)$.

2.2 MFCC features

In the 1980s, **Mel-Frequency Cepstrum Coefficient** features have been developed akin to both psycho-acoustic and physiologic reception characteristics of the human ear. The filterbank used for their computation partitions the frequency band into several subbands. Let $k = 1, \dots, K$ denote the filters forming the filter bank, the distances of the subbands' centers, c_k, are decreasing at a logarithmic scale with decreasing frequency; accordingly, the same holds for the bandwidths. The motivation for this is, that the human ear perceives sound, frequencies and dynamic due to a logarithmic scaling.

Figure 2.1 depicts the filter bank, which is defined as

$$Mel_k[f] = \begin{cases} 0 & f < c_{k-1} \\ \frac{f-c_{k-1}}{c_k-c_{k-1}} & c_{k-1} \leq f \leq c_k \\ \frac{c_{k+1}-f}{c_{k+1}-c_k} & c_k \leq f \leq c_{k+1} \\ 0 & k > c_{k+1} \end{cases}. \tag{2.3}$$

The filters have a triangular structure, where the concept of overlapping segments adheres to relations and correlations of adjacent frequency subbands.

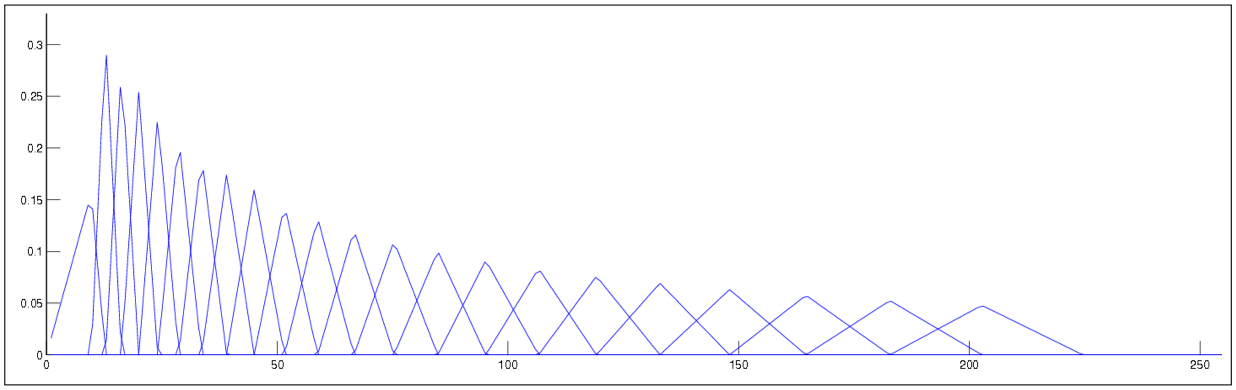


Figure 2.1: The unnormalized MEL filterbank with lower and upper filter frequencies of $133\frac{1}{3}$ and 6855.4976 Hz, respectively, showing frequency (Hz, horizontal axis) versus amplitude (vertical axis). Further parameters are the number of filters (40), the FFT-size (256) and the sampling rate of 16kHz. The (overlapping) triangles cover the different frequency subbands.

Given the N-point discrete Fourier transform $X[f] = \sum_{n=0}^{N-1} x[n]$ of a discrete (for instance sampled) time signal $x[n]e^{\frac{-2\pi i\omega nf}{N}}$, the MFC coefficients are computed by first calculating the log-energy

$$E[k] = \ln \sum_{n=0}^{N-1} Mel_k[f] \, |X[k]|^2$$

and afterwards applying a K-point discrete cosine transform (DCT)

$$C[n] = p_K \sum_{k=0}^{K} S[k] \cos\left(\frac{\pi n(2k+1)}{2K}\right),$$

that decorrelates the resulting coefficients among the frequency subbands of each vector. p_K is a normalization factor and frequently either set to 1 or, more often, chosen to be $\sqrt{\frac{2}{K}}$. In the latter case, the coefficient $C[0]$ is additionally scaled by $\frac{1}{\sqrt{2}}$ to guarantee the orthogonality of the transform.

Both the definition of the DCT and its various equivalent forms of representation and the analysis of some of its characteristics are given in more details for instance in [36]. MFCCs, similar features or such based on them are used in different areas of research but have been proven to be successful especially for speech recognition and phoneme classification tasks. The interested reader is also referred to [37], [33] or [12].

2.3 Markov Models

Hidden **M**arkov **M**odels (HMMs) basically are stochastic finite state machines used for modeling certain aspects of stochastic sequences. They are especially suitable when non stationary processes are comprised of piecewise (quasi) stationary signals, as HMMs represent both the stationary situation and transforms or transitions between the latter by probabilities. The meaning of the word *hidden* will become clear after a short review of the theory behind HMMs and their definition. We restrict introduction of Hidden Markov Models in this section (and work) to the discrete value case, as this suffices for our purposes. HMMs can be considered as a special case of both so called **Observable Operator Models** (OOM) and **Markov chains**. OOMs were introduced by Jäger (cf. [38]). In contrast to HMMs, they do not present a model based on hidden and emission states where output probabilities are generated, but describe processes as a sequence of (linear) operators. The most distinctive difference is that OOMs consider the observed variable directly, see also remark 2.1 for clarification. The training algorithm shows some advantages over that of HMMs and the class of OOMs is more encompassing. For more details, the interested reader should consider [38] or [39].

We restrain from explaining the training and estimation process of computing the models from data, as it is not essential for our research; the interested can consult [37] or [33].

2.3.1 Markov chains

Markov chains are special cases of stochastic processes, a family of random sequences where the sequences are pairwise stochastically independent and their states depend on past states only. In a Markov chain, the probability of a state depends solely on the preceding state[2]. For a sequence $Q_1, \cdots, Q_T$ of random variables over a finite and discrete output alphabet $\Omega = \{o_1, \ldots, o_M\}$, the joint probability can be computed using Bayes' Rule,

$$P(Q_1, \ldots, Q_T) = P(Q_1) \prod_{t=2}^{T} P(Q_t | Q_1, \ldots, Q_{t-1}). \tag{2.4}$$

Applying the **Markov assumption** and restricting the dependency to the preceding point of time only provides the first the **first oder Markov chain**, which simplifies 2.4 to

$$P(Q_1, \ldots, Q_T) = P(Q_1) \prod_{t=2}^{T} P(Q_t | Q_{t-1}). \tag{2.5}$$

Statistical properties of the process modeled by a Markov chain can be visualized and formalized in a graphical way as well as in a state machine like notation. Given a (finite) number of possible states, at each point of time the Markov chain has to be in a well defined state where **transition probabilities** (including auto-transitions) define the statistical behaviour. Figure 2.2 is an example of a graphical representation, where ellipses depict the states and arrows the transition probabilities. This graph shows a specific *left to right* situation without skips and backwards transitions. As can be seen from the figure, probabilities a_{ii} reflect chances for loops and hence serve as representers for moments where the sound is (quasi) static. More elaborate models also include transitions $|j - i| > 1$ and $j < i$ and even combinations, which constitute skips of states and backward jumps, respectively.

2.3.2 Hidden Markov Models

HMMs are a special, extended kind of Markov chains and include a variety of variables. In the context of speech recognition we can, as already done without being mentioned for

[2]In some literature you will instead find the equivalent definition, that in Markov chains the probability of all future states depend on none but the current state, i.e. that they are memoryless.

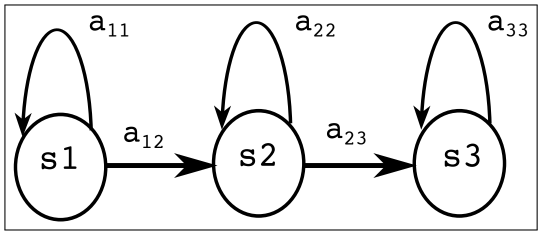


Figure 2.2: Graphical representation of a Markov chain. The ellipses $s1, s2, s3$ represent the (three) states the process modeled by this graph can take and a_{ij} are transition probabilities.

Markov chains above, restrict the models to finite and discrete sets as well as discrete probability distributions. Let us start by looking at the Finite State Machine based definition of Hidden Markov Models. A HMM is a tuple $\lambda = \{S, O, \Pi, A, B\}$ where

- A set $S = \{s_1, s_2, \ldots, s_N\}$ of states $(n = 1, \ldots, N)$.
- A set $O = \{o_1, o_2, \ldots, o_M\}$, the output alphabet.
- A vector $\Pi = (\pi_1, \pi_2, \ldots, \pi_n)$ of initial probabilities.
- A matrix $A = \begin{pmatrix} a_{11} & \cdots & a_{N1} \\ \vdots & \ddots & \vdots \\ a_{1N} & \cdots & a_{NN} \end{pmatrix}$ of transition probabilities.
- A matrix $B = \begin{pmatrix} b_{11} & \cdots & b_{N1} \\ \vdots & \ddots & \vdots \\ b_{1M} & \cdots & b_{NM} \end{pmatrix}$ of **emission probabilities**.

  For each output $o_m \in O, m \in [1, \ldots, M]$ the respective entry in the matrix equals the probability that, given state s_n, o_m is emitted. For the matrix B, the probabilities are therefore given by $b_{nm} = P(w = o_m | v = s_n)$, $1 \leq n \leq N$, $1 \leq t \leq T$.

This model is defined for a state sequence $v_1, \ldots, v_T$ and an associated sequence $w_1, \ldots, w_T$ of outputs. Figure 2.3 gives a graphical impression of the formal definition. The notation equals that of the theoretical section 1.5 or section 3.4, which describes initial experiments. The $1 \times M$ vector $\mathbf{b}_l, l \in \{s, m, e\}$ is itself time dependent, which becomes clear when looking at the definition of the matrix B and the probabilities its entries represent.

We note that in general transition probabilities correspond naturally to the adjacency matrices of the directly (connected) acyclic graphs (DAG) of the automaton / Markov

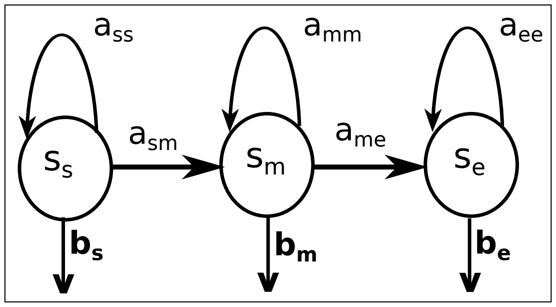


Figure 2.3: Graphical representation of a typical HMM modeling a spoken phoneme, illustrating the specific situation in our work. The depicted HMM realizes a standard tristate model, where the (empty) start and end state are omitted for reasons fo clarity. The foundation are three states denoted by indices s, m, e. Roughly spoken, they represent *start, middle* and *end* positions in a phoneme. The figure elucidates furthermore that the state of a variable only depends on the most recent predecessor. The indices of the transitions emphasize the change of states from s to m and m to e.

model given. Further probabilities related to actions such as state skips or backward jumps (not part of the model shown in figure 2.3) have zero probability.

For phoneme classification tasks, each phoneme is partitioned into subphonemes. Usually, the partition consists of three states, which is the reason why this is known as a **tristate** scheme. The states are often called *begin*, *middle* and *end* – or *start* instead of *begin*. In the same manner, words are segmented into phonemes when processing continuous speech. Here, sequences of three phonemes have been established as well, and this kind of modeling is commonly known as the **triphone** scheme. Observations which is analyzed segmentwise and for recognition/ classification the HMM for which the probability of generating this observation is highest is chosen.

Remark 2.1

To close this section, let us clarify what the *hidden* part is. The name stems upon the underlying process: By definition, each state can emit each of the output symbols and each such output has a certain probability given by the matrix B. What we **can** observe are the results, the emissions. What we **can not** observe is the process itself, that is the sequence of states that generated the output sequence. Hence the word *hidden*.

2.4 modeling speech and language

This section very briefly describes the two kinds of models used in common speech recognizers, the acoustic and the language model. The acoustic model is also shortly viewed in the general context of pattern analysis to clarify important concepts and denote the respective, relevant aspects in the specific task of speech recognition.

2.4.1 Acoustic modeling

The Hidden Markov Model described in the previous section is a very effective way of modeling substantial basic building blocks of multistage patterns. Figure 2.3 illustrates this for speech, where the components to be trained are phonemes. The latter are rather short and can very adequately be represented and modeled[3] by three to five states.

Depending on the task given and especially on the amount of different words (or in general the amount of patterns build from the basic blocks) to be recognized, the objects trained can also be subphonemes. The *SME* structure and feature partitioning in this context can be seen as a transition from modeling phonemes to classification on exactly this level of detail. Alternatively, one can train recognizers on words themselves, which is useful and delivers high recognition rates for small vocabularies. The very good book on speech recognition by Huang ([37]) covers this topic on pages 427 to 429. Also the references cited within this book are worth to be studied.

In order to guarantee an effective and stable classification or recognition, a chosen model needs to meet certain criteria, and the design and topology of the model greatly contribute to the overall quality. In general, for pattern analysis those requirements[4] are

- **Computational efficiency**: For both training and classification the amount of used resources such as processing time and space should be reasonable and scalable w.r.t. a potentially increasing amount of data such as new training classes.

- **Accuracy and robustness**: The model must guarantee accurate classification, i.e. pattern variation within and between classes must be taken into account and should disturb classification to the lowest possible degree.

[3]This statement holds for languages where the total amount of phonemes is in the range of 30 to 50, such as German, English or French, see [37], section 9.4.1, pp. 427 to 429 and therein cited references.

[4]See [9], pages 12 and 13. Our short summary given here deviates in some aspects from his elucidation, however.

- **Generalizability and stability**: New, unseen patterns with potential deviations must be recognized as expected.

For speech recognition and phoneme classification, the flexibility of an acoustic model needs to encompass for instance the deviations originating from the manifold differences in the ways of pronunciation, from varying lengths, spectral development or other characteristics. In practice, for either of the approaches (word, phoneme or subphoneme based recognizer) the acoustic contextual information of the observed object is extremely relevant. For phonemes, this is even more sensitive than for words, as single (sub)phonemes are for instance frequently completely swallowed or merge into each other. HMMs are very often the model of choice, as they can handle both the temporary stationarity as well as stochastic influences by uncertainties such as those mentioned above. Common modifications and alterations are for instance the possibility of skipping states, and the reader can get detailed information on variations of HMMs in the above mentioned book of Huang ([37]).

2.4.2 The N-gram language model

Similar to the acoustic topology, language specific information can help decoding utterances. Most common are so called N-**gram** models, where the probability of a word given $N-1$ predecessors is crucial for the decoder when deciding for a word within an utterance. Looking at the previous section, an N-gram design is a kind of a simple version of a context sensitive model representing potential word combinations at a very basic level. The probabilities are computed easily by counting occurrence frequencies: Considering a Trigram model ($N = 3$), let $o = (w_{t-2}, w_{t-1}, w_t)$ be an observation at time t comprised of the sequence $w = w_{t-2}w_{t-1}w_t$ of the word in question and its two predecessors. Denoting by $w :_{t-1}$ the subsequence restricted to w_{t-2}, w_{t-1} and by $|w|$ the number of occurrences of a sequence, the probability of a word w_t given two predecessors w_{t-2}, w_{t-1} is time independent and computed by

$$p(w_t|w_{t-2}, w_{t-1}) = \frac{|w|}{|w :_{t-1}|}. \tag{2.6}$$

While an N-gram language model therefore does not need any further grammatical information it becomes clear that a sufficiently large amount of training utterances with representative word sequences is necessary. Also, the choice of N clearly influences the degree of freedom a grammar allows for. A detailed discussion of this topic can be found for instance in [37], chapter 11, and is not part of this work.

2.4.3 Perplexity

The quality of a language model can be evaluated by performing a complete recognition run for a given testset. There is however also some intrinsic quality measure depending on the complexity of a language model (given a testset) that can be deduced without the necessity of a complete recognition run. It is based on the entropy given the language model and is called **perplexity**. Informally, it tells us how many choices in average the recognizer will have to take into account for each word.

More formally, consider a dictionary W of words $w, v \in W$ and an associated probability distribution p such that $p(w|v)$ denotes the probability that the word w is a direct successor of word v and where $p(w, v)$ is the overall probability that, given any position in a text, word v is followed by word w. The latter can be computed by simply counting the number of pairs v, w and dividing by all different word pairings. The (conditional) entropy of the word distribution given the context v as the previous word is then defined as

$$H_p(w, v) = -\sum_{w,v \in W} p(w, v) \log_2 p(w|v). \tag{2.7}$$

Then the perplexity is defined as

$$\Pi(H_p) = 2^{H_p(w,v)}. \tag{2.8}$$

The exponent can be interpreted as the averaged state entropy and equations (2.7) and (2.8) furthermore show that the entropy is the (base 2) logarithm of the perplexity. The definition can be extended naturally to sequences of word, see for instance [40], chapter 7.

Depending on the size of the dictionary, the complexity and representativity of a learning set and the similarity of test and training data, the perplexity may vary from small values such as ten or even less, to large ones up to 1000. In addition to [40], the interested reader may want to review [37], chapter 3.4, pp. 121 to 123 and chapter 11.3, pp. 554 to 556 or [41], pp. 317 to 318 for more details.

As a final note we remark that the distribution p and probabilities $p(w, v)$ and $p(w|v)$ are clearly determined by the set of training sentences and along with it heavily depend the on the choice of the language mode – hence so does the perplexity.

2.5 MFCC correlation features

In this section we introduce new features based on usual MFCC-vectors. Subsection 2.5.2 deals with auto correlation features and motivates their usage, whereas the subsequent subsection 2.5.3 modifies the approach and proposes cross-correlation features instead as well as a new family of SVM classifiers, where the structure of the latter reflects both the partitioning of phonemes into subphoneme and the (HMM)-states, see section 2.3. The motivation for the latter section is manifold:

After depicting the links between speech, stationarity and (auto)correlation, as a first application classifications using auto-correlation features show improvements over common MFCC-$\Delta - \Delta\Delta$ setups. Section 3.1 illustrates the results, contributing to further research interest with correlation features in general. Second, cross-correlation with a fixed vector $\mathbf{y}$ is a linear operation in $\mathbf{y}$ and allows integration of the operation itself into the SVM's reproducing kernel by applying the theory of section 1.5. Third, being developed in direct connection to the way the cross-correlation is realized, the SVM-based classification method in the spirit of subphoneme state representation as modeled for example by HMMs is a first step from pure phoneme classification towards speech recognition, using hitherto unknown, new features and methods.

2.5.1 Stationarity and correlation

From a statistical standpoint, each spoken utterance can be seen as a realization of an underlying stochastic process generating each utterance. Over short periods of time, say 5 to 12 ms, such frames of speech are approximately **wide sense** or **weakly stationary** or **covariance stationary**, see for instance [37] and [42]. The base idea of regarding speech signals as sections of successive weak stationarity blending into each other as potential non-stationary segments is a keystone of this thesis: The autocorrelation (ac) and later on cross correlation (cc) features developed originate from this concept.

Concerning the latter, different stationary segments are assumed to represent different phonemes and different time states within them. (Potentially non stationary) transitions between them signal phonetic changes and should thus drive features into a different direction, as new speech frames and changing states differ in the lag dependencies determined by the degree of stationarity. Combined, this should help us telling different phonemes and changes between them apart. Let us summarize these milestones:

The conceptual and theoretic foundations and main assumptions for the development and research of correlation features for phoneme classification and speech recognition are...

- ... that the time independency and lag dependency of first and second order moments of a quasi short time weak stationarity signal ...
- ... that the variation of the lag dependencies ...
- ... that different speech frames and transitions between them vary in the degree of stationarity, and that this differences...

are reflected to a certain, sound degree by (auto)correlation. The aim of this section is to clarify, why correlation features are justified by the assumption of weak short time stationarity of speech, see section 2.1.

Using *ac* for autocorrelation and *cc* for cross correlation, concerning the development of the new features we depict the sequence of milestones in our research process thus evolves as

$$\text{MFCC features} \xrightarrow{\text{degree of stationarity}} \text{MFCC } ac \text{ features} \xrightarrow{\text{linearization}} cc \text{ features}$$

Following [43], let us briefly review weak stationarity. Given a realization of a stochastic process in form of a random variable Y_t, weak stationarity means that the first (mean) and second (auto covariance) statistical moments are independent of the point of time and influenced only by the interval separating any observations, that is

$$E[Y_t] = \mu \tag{2.9}$$

$$E[(Y_t - \mu)(Y_{t-j} - \mu)] = \gamma_j = \mathrm{Cov}_Y(t, t-j) \tag{2.10}$$

$\forall t, k$, where μ and γ_j are the mean and autocovariance – both independent of t – and $E[\cdot]$ is the expectation. As we can see, the second moment – that is the autocovariance – is basically the covariance of Y_t with its dilated (or lagged) value Y_{t-j}.

By definition, the autocorrelation

$$\gamma(t_1, t_2) = \frac{\mathrm{Cov}_Y(t_1, t_2)}{\sigma^Y_{t_1} \cdot \sigma^Y_{t_2}} \tag{2.11}$$

inherits certain characteristics from the autocovariance, amongst them most importantly that it depends only on the lag $j = t_2 - t_1$. The first stage of experiments elaborated in the next section (2.5.2) gives details of the computation of features using autocorrelation of neighboured speech frames and MFCC features computed from those.

For the subsequent linearization it is important to ensure that statistically this method is well defined and meaningful. With $E[\cdot]$ and $\mathrm{Cov}_{\{\}}(\cdot, \cdot)$ not depending on t, the same holds for the correlation of two random variables X_t, Y_t:

$$\mathrm{Corr}(X_t, Y_t) = \frac{\mathrm{Cov}_{X,Y}(t, t)}{\sigma^X_t \cdot \sigma^Y_t}. \tag{2.12}$$

2.5.2 Autocorrelation features

While different components of MFCC feature vectors (and thus the different frequency subbands they represented) are decorrelated via a discrete cosine transform in the process of their computation, correlation remains within sequences of the same component. In our auto-correlation approach, correlation between the same components of adjacent standard MFCC-vectors of length L replace the widely used Δ and $\Delta\Delta$, resulting in features of total length $L + 3L = 4L$ (details are given below). For SVM training and classification, both parts – the standard MFCC-vectors and the correlation vectors – of the compound features are convex combined via reproducing kernels. As the experiments are intended to be comparable to standard speech recognition experiments, we keep the number of features in a similar range by correlating only three directly adjacent MFCC features of appropriate length L – concerning this, the reader is also suggested to consult the conclusions in section 3.7.

Formalizing all this, let $m^l_{n-1}, m^l_n, m^l_{n+1}, m^l_{n+2}$, $n \in \mathbb{N}$ be a sequence of adjacent MFCC vectors, where $l = 1, \ldots, L$ references a component of the vectors and the subindex n the speech frame the MFCC features were computed from. Using $\times$ to indicate cross correlation and forming two vectors, each of length three, of the same components of adjacent vectors, we get L cross correlation vectors $\tilde{m}_l$ by letting

$$\tilde{m}_l = \left(m^l_{n-1}, m^l_n, m^l_{n+1}\right) \times \left(m^l_n, m^l_{n+1}, m^l_{n+2}\right).$$

Normalizing and stacking the $\tilde{m}_l$ finalizes the computation of the autocorrelation feature vector $\left(\tilde{m}^1, \ldots, \tilde{m}^L\right)$. Figure 2.4 illustrates the computation graphically.

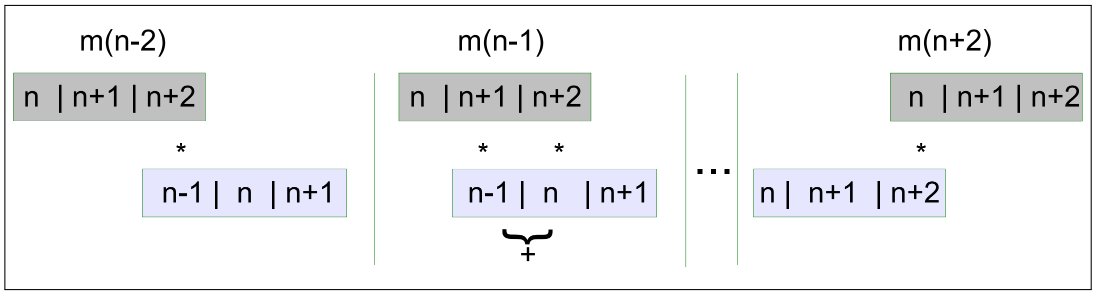


Figure 2.4: Computing autocorrelation features from MFCC feature vectors of a phoneme at frame n. To keep things clear, the values of the MFCC vector components are represented directly by their indices $n-1, \ldots, n+2$.

2.5.3 Linearization and a phoneme state like approach

In analogy to the concept of partitioning phonemes into smaller subsections, so called subphonemes, we propose new features in the spirit of both a split of the features akin to subphonemes and correlation information in MFCC vectors of the same frequency band but adjacent frames. A new approach is also taken by not only using the proposed new features but also applying the concept of the subphoneme like structure to a family of support vector classifiers. In contrast to the autocorrelation operation described above, given a fixed vector $\mathbf{x}$ of finite length, correlation with any finite vector $\mathbf{y}$ is a transform linear in $\mathbf{y}$. Hence, as a further implication, we can apply the theory presented in section 1.5, which allows us to integrate the feature computation into that of the kernel.

Before explaining the details of the individual steps, let us look at figure 2.5, which summarizes the interactions of the diverse elements in a less detailed way than figure 1 from the introduction. Instead, it addresses main foci and illustrates influences and flow paths in a less abstract manner, making the subsequent explanations clearer.

To start with, we again consider MFCC-vectors without Δ and $\Delta\Delta$. The simulation of a subphoneme kind of representation includes several steps:

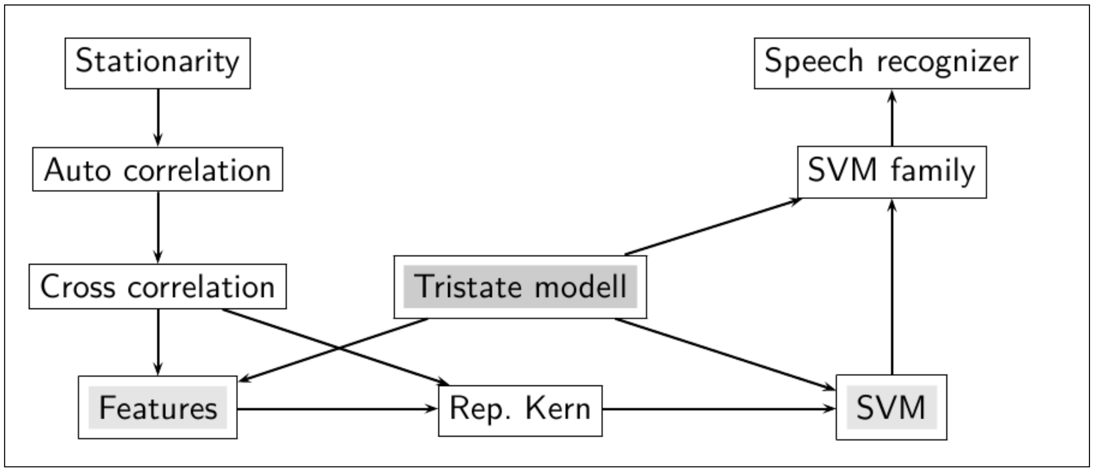


Figure 2.5: Overview of the thesis and important relations between the diverse parts.

(1) – ***SME*-Partitioning of observations/ samples**

Each phoneme sample, from now on called observation and denoted by o, is partitioned into three subsections. The overall length s, that is the number s of MFCC-vectors (or frames) the sample is comprised of, determines the split: Processing the feature vectors in their original order, we divide the data into start- and end section (S and E respectively) of length $s \div 3$ and each and a middle section (M) of length $(s \div 3) + (s \mod 3)$. The choice for this split into parts of nearly equal length is not based on any kind of optimization or made due to results of comparisons to other partitioning schemes; it is used because this approach is new, and we are aiming at getting initial results by following a baseline that is constructed, where possible, in a simple manner. We assume that a more sophisticated strategy will provide substantial improvements, see the prospects given in section 3.7. As an example, consider a phoneme sample comprised of 13 MFCC vectors. Then the start section covers the first four vectors, the next five determine the middle section and the final four vectors comprise the end section.

(2) – **Computing subphoneme center vectors**

For each phoneme class p in the set of phonemes we create three corresponding subsets by partitioning all members (observations) of the class and grouping the respective vectors of their S, M and E-sections determined by the scheme described above. Next, for each class the component wise averages of the feature vectors of each of the three subsets are computed and called **center vectors**. For the sake of clarity, the following notation does not include the class information p. Thus, we denote the three centers of a phoneme class $\mathbf{c}$ by $\mathbf{c}_s$, $\mathbf{c}_m$ and $\mathbf{c}_e$, respectively and will also later restrict mathematical formalization to one class. This, however, does not pose a problem, as the computations themselves do not vary amongst different classes.

Figure 2.6 illustrates both partitioning and computation of the center vectors.

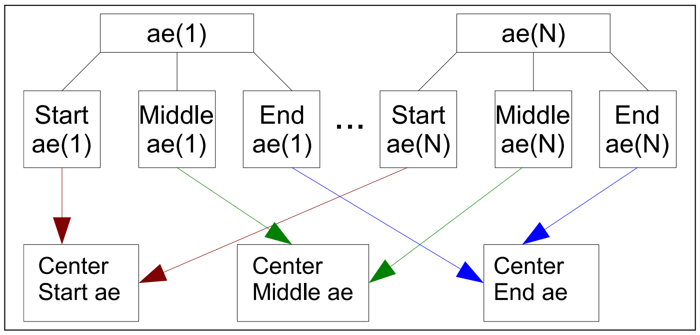


Figure 2.6: Partitioning and clustering of N samples of class *ae*.

(3) – **Structuring observations based on the *SME*-partition**

Using the partitioning scheme of (1), we introduce another *time alignment*, by grouping directly neighboured vectors based on their position within the sequence representing a phoneme. For a more detailed explication, let us focus on a single observation o. We consider a sequence of three adjacent MFCC-frames at each time step and a frame shift of one feature vector between two consecutive steps. Processing the complete sample, an observation comprised of for instance $M = 13$ MFCC-vectors results in $M-2 = 11$ sequences $\omega^1, \ldots, \omega^{M-2}$ of three vectors each, as for instance ω^1 includes vectors one to three, ω^2 vectors two to four and so forth. Again for reasons of clarity, for now we discard the index and just write ω. From steps (1) and (2), each of the three vectors of any ω is an element of either (class specific) partition S, M or E, depending on its position within the observation.

For every sequence of three vectors generated as described above, this ”windowing” leads to one of the following possible combinations: *SME, SSS, SSM, SMM, MMM, MME, MEE, EEE* – for each phoneme, these sets will form our new classes! Figure 2.8 illustrates the process graphically. Note that in this work we only consider observations o of length ≥ 4. Thus, the case of an *SME* group, which occurs for observations o of length 3 (thus $o = \omega$), will not be of further interest for us.

The seven groups constructed can be interpreted as sets of vectors representing a rather specific time- (or position-) based subphoneme state of the phoneme being observed.

(4) – **Creating subclasses from the original phoneme classes**

The process described in step (3) determines which three-vector subsequence ω of an observation o contributes to which of the new classes. We want to point out that, depending on the length of the classes' observations, o will potentially not contribute to all of the seven classes. The alignment and class setup described so far results in a structure which from now on will be called *SME*-structure.

(5) – **Linearization utilizing the *SME*-structure**

The center vectors (see step (2)) serve as representative anchors and foundations for the individual classes. In addition, we remember from section 2.5 that, for computing the auto-correlation features from an observation o, two adjacent sequences ω_n, ω_{n+1} of three MFCC-frames each were convoluted. This operation is now replaced by cross correlation of two vectors, one of which is a three vector sequence as constructed above. The second vector is a fix vector determined by the (class specific) group *SME, SSS, …, EEE* the sequence belongs to, and each of the groups itself contributes to a time/position specific combination C_ω of center vectors. The details will be explained right away, but we want to emphasize the key idea before:

The important steps in this process are the coupling of the concept of the center vectors with the idea of the *SME*-structure and along with that the **how to** of combining the center vectors to get C_ω: If we simply used a stacked vector $\left(\mathbf{c}_s, \mathbf{c}_m, \mathbf{c}_e\right)^T$ and computed its correlation with a given ω, the within-phoneme position information – intrinsic within the *SME* partition now – would remain disregarded. Our notation C_ω already points out that the correlation vector is indeed not only related to the class by being formed utilizing the fixed center vector. It also ought to depend on – and thus should include information on – the current observation's sequence of three vectors ω. It thus is a twofold context dependent feature.

The next step therefore also involves within-phoneme position information from the new class structure for the construction of C_ω: Given a phoneme p, let ω be an observation and assume we have formed its associated partitions $\{$*SSS, SSM, SMM, MMM, MME, MEE, EEE*$\}_p$. With l denoting the size of the MFCC vectors of p and $\omega s, \omega m, \omega e \in \{S, M, E\}$ denoting position[5] specific states, we define $C_\omega = \left(c^l_{\omega s}, c^l_{\omega m}, c^l_{\omega e}\right)_o$, given the current observation o.

The following example illustrates and clarifies the construction of the center vector combination C_ω:

[5]The position in this context is referring to the complete sequence described in (3), from which the subsequences of three vectors each are formed

Let $\omega \in SSM$ (remember this is a partition and class!) and let $c^l_{\omega s}$ and $c^l_{\omega m}$ have values from component l of the cluster center vector $\mathbf{c}_s$ representing the two S-state phases, whereas $c^l_{\omega e}$ is set to the respective component of $\mathbf{c}_m$ standing for the M-state phase. Now the cross correlation of the combined center vectors with the three MFCC vectors at positions $n, \ldots, n+2$ is performed, graphically depicted by figure 2.7.

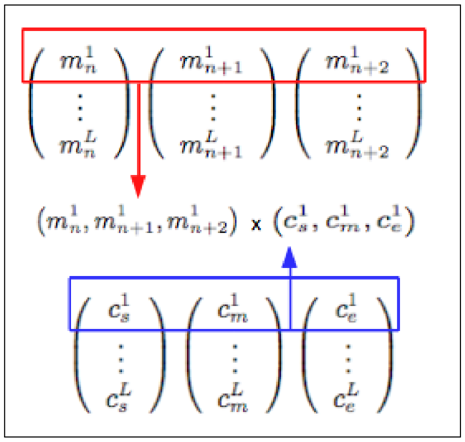


Figure 2.7: Component wise computation of the cross correlation features by three class and state dependent center vectors and three frames. The $\times$-symbol denotes the correlation operation.

It is clear that this cross correlation is indeed performed using data in a way similar to the that of the computation of the autocorrelation described in section 2.5.2, where the combined center vectors now replace one of the two adjacent sequences of three MFCC features vectors. Summing all this up, given the sequence $\omega = \omega^l_1, \omega^l_2, \omega^l_3$, the linear cross-correlation mapping equals

$$S\left(c^l_{\omega s}, c^l_{\omega m}, c^l_{\omega e}\right) \times \left(\omega^l_1, \omega^l_2, \omega^l_3\right). \tag{2.13}$$

Rewriting cross correlation with C_ω (equation 2.13) in matrix form omitting the component index l for the sake of clarity, we have

$$T = \begin{pmatrix} 0 & 0 & c_{\omega s} \\ 0 & c_{\omega s} & c_{\omega m} \\ c_{\omega s} & c_{\omega m} & c_{\omega e} \\ c_{\omega m} & c_{\omega e} & 0 \\ c_{\omega e} & 0 & 0 \end{pmatrix}.$$

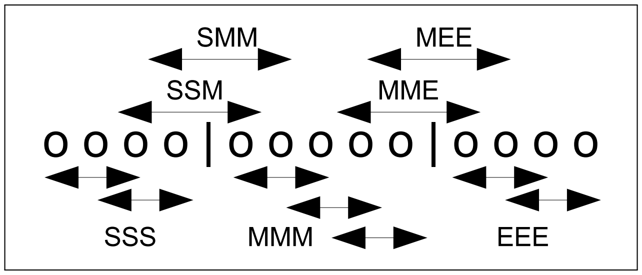


Figure 2.8: Example of a phoneme sample comprised of 13 MFCC-vectors/ frames and its split into *SME*-based parts. Two vector sequences are used for the classes *SSS* and *EEE*, three for *MMM* and one each for *SSM, SMM, MME* and *MEE.* In a consistent manner, the cumulative center vectors C_ω for the linear transforms necessary to compute the new cross-correlation features from the sequences are formed based on the same classes the respective sequences are categorized into.

In the light of the derivation given in section 1.5.1, we substitute T into equation (1.40) and finally get

$$T^*T = \begin{pmatrix} c_{\omega s}^2 + c_{\omega m}^2 + c_{\omega e}^2 & c_{\omega s}c_{\omega m} + c_{\omega m}c_{\omega e} & c_{\omega s}c_{\omega e} \\ c_{\omega s}c_{\omega m} + c_{\omega m}c_{\omega e} & c_{\omega s}^2 + c_{\omega m}^2 + c_{\omega e}^2 & c_{\omega s}c_{\omega m} + c_{\omega m}c_{\omega e} \\ c_{\omega s}c_{\omega e} & c_{\omega s}c_{\omega m} + c_{\omega m}c_{\omega e} & c_{\omega s}^2 + c_{\omega m}^2 + c_{\omega e}^2 \end{pmatrix},$$

where T^*T is positive definite and hence k_{T^*T} a reproducing kernel if

$$\text{trace}(T^*T) = \left(c_{\omega s}^l\right)^2 + \left(c_{\omega m}^l\right)^2 + \left(c_{\omega e}^l\right)^2 \neq 0$$

is satisfied for all $c_{\omega s}, c_{\omega m}$ and $c_{\omega e}$.

2.5.4 The bigger picture

Let us briefly get back to section 2.5.1. First and without proof we note some important facts esp. about non stationary signals and correlation. For details, the interested reader can consult for instance [44] or [45] and [46], where some special cases of non stationary signals (random walks) are considered. The key in their elucidations, which are combined here, lies in the application of both the continuous mapping theorem and the (functional) central limit theorem to the closed integral form of the statistical moments. Both theorems guarantee that certain characteristics of continuous functions are preserved when taking the limit even if their arguments are random processes. In the light of such random processes, the main point of interest is, how ensemble correlation compares to sample correlation, as we need to be able to analyze and characterize realizations of such a process.

For reasons of clarity we focus on the simple case of time series or signals in $\mathbb{R}$ or $\mathbb{C}$, which is without much effort generalized to multidimensional or function spaces. In the case of (weak) stationary series or signals, all or at least some moments such as variance, expectation etc. converge to constant values, which implies their time invariance and carries over to (sample) cross correlation. For non stationary signals, the situation is a little more complicated. Here, statistical moments and along with them covariance and correlation are varying with time. Under the above mentioned restrictions, it can be shown that correlation for random processes is still mathematically well defined. The sample correlation, the measure we are interested in, is approximated by taking the limit of the distribution that is the foundation of the random walk, and it turns out to basically converge towards a random variable in contrast to the stationary process.

This at least is mathematically meaningful, as random variables take values (from their domain) with a certain probability determined by the underlying probability distribution. In practice this translates to getting values from correlation that are based on this distribution (i.e. on the product space given by the product of the underlying probability space with itself). Hence for both stationary and non stationary signals, (sample) correlation is – in the sense of limits – well defined. However, in the latter case it is clear that the result does not measure linear dependency and is thereby not interpretable in the way it is for stationary signals.

What does this difference between stationary and non stationary cross correlation imply for features based on them? Basically, the impact of the non stationarity is, as we have seen above, that no matter how much (training) data we have, the outcome of cross correlation is different each and every time. Clearly, if we had to deal with non stationary data, cross correlation would be of no use for our classification system, as we do not match data to distributions or distribution parameters, which are the foundations of the random variables. As mentioned in previous chapters, we therefore rely on the short time weak stationarity of speech. This guarantees the time invariance of the statistical moments and keeps features comparable and classifiable, as the moments vary to different degrees for distinct phonemes and speech frames.

One of the main reasons for the elaboration given in section 2.5.1 is hence that it makes clear why we use short speech frames in the first place, whereas here we try to underline why stationary and non stationary cases need to be interpreted differently. Non stationarity within speech originates for instance from the flexibility of the vocal tract, which accounts for continuous changes of parameters such as timbre, air pressure and volume, but also from the different sounds that influence those parameters in particular. The stationarity is strongest when there are fewest changes, and our partitioning approach helps separating unequal speech sections: sections, where statistical moments change. Altogether, this emphasizes the importance of our choice to structure data further by partitioning phonemes already on feature level. In other words, it is supposed to reflect the mentioned difference between stationary and non stationary segments and transitions.

2.6 The problem of imbalanced training class sizes

We want to close this chapter with a very quick discussion addressing the problem of training SVMs on classes varying strongly in size. Our multi class SVM training is – following the theory given in section 1.9.2 – performed under the assumption of rather balanced class sizes. Previous sections in this chapter have made clear that this premise is clearly violated considering for instance the phoneme sets *aa_ao*,...,*z* and their *SME* subclasses. As mentioned before, in the latter case in particular longer phonemes produce for instance far more members for the *MMM* class than for *SSS* or *EEE* classes.

Even though the class imbalance problem has direct consequences for the experiments conducted and described in chapter 3, where a reduction to fewer *SME* classes is carried out for the sake of comparability to common speech recognition systems, we do not deal with this issue here any further. Nevertheless, we want to mention a few publications tackling the problem.

Whereas the authors of [47] focus on the SVM case, most articles are more encompassing. In [48] the authors include concepts for methods such as random forests or other ensemble classifiers in their research and reviews. A profound, detailed study is given in [49]. In [50] the authors review still more publications showing research results. They give a quick overview and compare differences and common issues in the reviewed approaches.

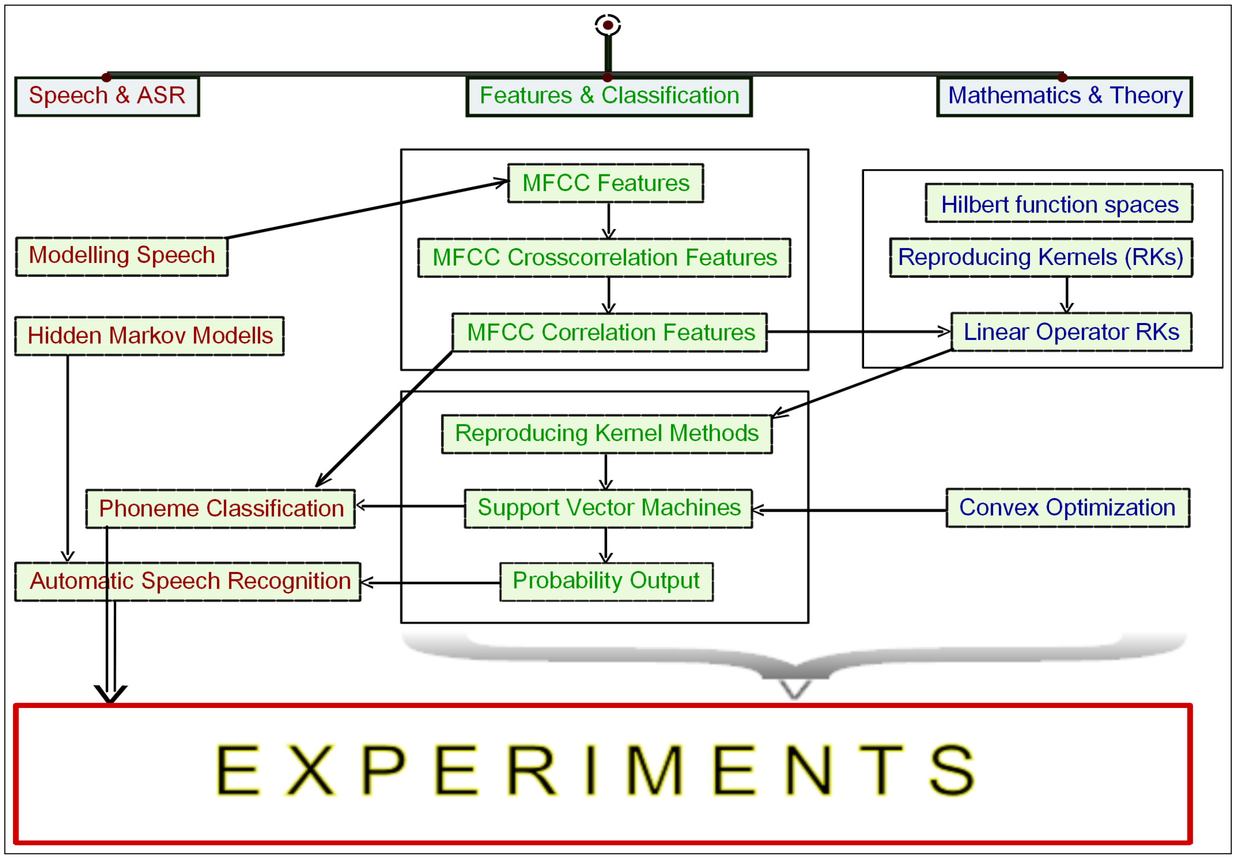
We close this chapter with a short remark on one potential effect of imbalanced training class sizes for SVMs: The optimal parameter(s) for the kernel of choice are clearly data dependent. As will be described in chapter 3, our parameter for the exponential kernel was chosen due to a rough grid search[6] on a small subset of all phonemes on which multi

[6] As an alternative one can also numerically solve the problem of minimizing for instance the leave-one-out error. In practice this anyway needs further fine tuning; optimizing does, however, give the general

class SVMs were trained. In an optimal evaluation, each binary classifier would have its own parameter. For slightly imbalanced class sizes one could for instance use (fix) size ratios for the different binary classifiers and try optimizing the (average) penalty over a separate testset. For greater imbalances, this becomes increasingly problematic: For the optimization it is necessary to compute the derivative of a loss function, for instance when determining the penalty. This derivative however is in general no longer continuous and the optimal kernel parameter(s) tend to be very small. As a result of that, some SV classifiers might converge towards ridge regression models thus posing a new burden.

direction.

Chapter 3 – Experiments



In the first two sections we present results from two sets of multiclass classification results representing the initial correlation feature and *SME*-struct experiments. Section 3.1 refers to the auto-correlation features introduced in section 2.5.2, whereas part 3.2 illustrates the results of the first cross-correlation experiments described in section 2.5.3. The features were extracted via HTK3.3 with a framesize of 25ms and an overlap of 10ms. Training and test for both experiments were performed on the eleven most frequent phonemes *aa, ae, ay, eh, ey, ih, ix, iy, n, s, z* of the TIMIT training dataset. For this, we modified svmlight 6.10 ([51]) to allow weighted multiple kernel training and classification. If not mentioned otherwise, *svmlight*-parameters remained unchanged. Also, parameters for SVMs trained on the new vectors were not optimized but chosen due to results from partially rough grid tests. Evaluating on finer grids and, in the case of kernel combination, solving the convex kernel combination SVM optimization problem will very likely improve results further.

Kernel combinations for SVM phoneme classification

We conducted experiments aiming at the observation of the effect of kernel combinations, using specific pairs of reproducing kernels k_1, k_2 on a fraction of the subset comprised of eleven phonemes shown in the introduction to this chapter. The reason for restricting the experiments to a fraction of this subset – to be exact half of the amount of training samples of each class – lies in the fact that in addition to a grid search on the parameters of the individual kernel each such search needs to be performed within another search for an optimization of the convex combination parameter α where $k_\Sigma = \alpha k_1 + (1-\alpha)k_2$, $\alpha \in]0,1[$. Due to this complexity, we decided for this computationally feasible setup.

3.1 Autocorrelation Features

The first stage consisted of comparing several single- and two kernel classification results using both common MFCC$-\Delta-\Delta\Delta$ and MFCC- auto correlation features (cf. to section 2.5.2 and figure 2.4 on pages 79 and 80). The comparison includes

(1) no kernel combination, one single rbf kernel k_{rbf} for the complete vector comprised of MFCC and autocorrelation data .

(2) one rbf kernel k_{rbf} and one linear kernel k_l for MFCC$-\Delta-\Delta\Delta$ features,

(3) one rbf kernel k_{rbf} and one polynomial kernel k_p of degree 2 for MFCC$-\Delta-\Delta\Delta$ features,

(4) two rbf kernels k_{rbf1}, k_{rbf2} used for the 13 MFCC and the $\Delta-\Delta\Delta$ components, respectively,

(5) two rbf kernels k_{rbf1}, k_{rbf2} used for the 13 MFCC and the autocorrelation components, respectively,

(6) MFCC$-\Delta-\Delta\Delta$ features using one rbf kernel k_{rbf}.

Given the rbf parameters $\gamma_{1,2}$ and the polynomial coefficients $c_0,\ldots,c_3$ for $k_p(x) = \sum_{i=0}^{3} c_i x^i$ we let $\alpha = 0.05, 0.10, \ldots, 0.85, 0.90, 0.91, 0.92, ..., 0.99$; the outcome is summarized by Table 3.1.

The table shows the common MFCC$-\Delta-\Delta\Delta$ features (item (6)) followed by classification using one rbf kernel for the MFCC values plus a linear kernel (item (2)), a polynomial kernel (item (3)) and a second rbf kernel (item (4)). The second best result was achieved by rbf kernel combinations of MFCC plus autocorrelation features (item (5)) and the best

phoneme	*aa*	*ae*	*ay*	*eh*	*ey*	*ih*	*ix*	*iy*	*n*	*s*	*z*	**avg.**
mfcc39	38.7	73.4	63.9	64.2	65.8	94.1	25.2	22.9	44.8	12.4	88.4	**53.98**
mfcc13, k_l	38.4	73.2	64.0	63.9	65.4	94.0	25.2	22.7	44.6	12.1	88.3	**53.80**
mfcc13, k_p	38.1	72.8	64.2	63.7	65.0	93.7	25.0	22.4	44.2	10.7	88.1	**53.45**
mfcc13, k_{rbf}	37.1	71.7	66.2	63.4	64.7	92.9	24.7	22.4	43.9	9.9	84.4	**52.85**
mfcc13,ac Σ	32.6	66.8	70.1	61.2	64.2	89.9	26.1	22.7	42.3	5.8	77.9	**50.87**
mfcc13,ac	32.4	67.7	66.2	61.4	64.7	88.0	24.4	25.3	41.3	7.0	76.7	**50.46**

Table 3.1: Results of classification error rates for two setups, from top (worse overall recognition rate) to bottom (best overall recognition rate).

by a single kernel for the same vector (item (1)). The relative improvement gain of this compared to the top table entry is 6.98%.

For class *ix*, classification abates a little when switching from ordinary MFCC vectors to the autocorrelation features. However, when using kernel combination, the error rate decreases noticeably again! The *ay* phoneme is recognized less accurately, whereas confusions between *s* and *z* improve remarkably when using MFCC-autocorrelation instead of MFCC$-\Delta-\Delta\Delta$ features. Overall recognition gain illustrated by the decreasing recognition rates for MFCC autocorrelation features is evident.

To mention some parameters: The best result following the grid search for the autocorrelation kernel combination setup was achieved for $\alpha = 0.92$ and $\gamma_1 = 0.001$ and $\gamma_2 = 0.015$. In all cases, the best α weight was between 0.96 and 0.99 with an average of 0.9625. This gives reasons for interpretation as follows: The second kernel most times seems to prevent a certain amount of overfitting, thus leading to a better generalization. Markable improvements seem to call for kernels designed for the specific task given, such as the autocorrelation features. In that case the weight becomes significantly smaller. All in all due to these results, the remaining experiments of this thesis were conducted with a single kernel only.

3.2 Crosscorrelation Features and *SME*-structure

The MFCC-correlation features extend over three frames, thereby rendering comparison to single-frame MFCC-features improper due to the difference in the amount of information. For this reason we consider 3-vector sequences of standard 13-dimensional MFCC-features (sMfcc) without Δ and $\Delta\Delta$, resulting in comparable feature vectors of the same dimension as standard MFCC vectors with 13 coefficients plus Δ and $\Delta\Delta$. The initial experiments, performed on the reduced set of 11 specific phonemes[1] (see table below), target at getting

[1]The subset was chosen due to several criteria: It should include phonemes similar to each other as well

an impression of the quality of the new features and the new structural approach. For the SVM we use a single exponential kernel, where the γ-parameter is again selected due to a rough grid search and set to 0.0001 for the sMfcc-SVMs and to 0.00001 for those based on *SME*-SVMs, setting *SME* weights to $w_{\mathcal{H}_T} = w_{\mathcal{H}} = 0.5$.

Table 3.2 shows the pooled results of the detailed, individual confusion tables given in appendix A.1. It illustrates the strong increase in the recognition rates and points out in particular, that different phases of phonemes are subject to partially large classification result differences. For phoneme *z* for instance, the confusion in the last four states drops remarkably compared to states SSS, SSM and SMM. The final state, EEE, is worst. This makes sense when considering that this is mostly noise or is 'swallowed' or shadowed by the subsequent phoneme/ sound already.

	sMfcc	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	**69.26**	73.68	87.72	91.58	94.41	96.01	96.05	91.54	**90.14**
ae	**59.79**	92.27	82.38	81.30	92.61	78.67	77.13	85.69	**84.29**
ay	**52.22**	80.83	83.48	91.08	96.35	90.05	75.56	79.22	**85.22**
eh	**44.22**	59.34	86.23	89.54	89.27	89.64	86.74	55.95	**79.53**
ey	**56.39**	74.42	78.52	78.40	88.70	81.21	82.22	87.56	**81.58**
ih	**37.82**	70.90	73.73	68.64	69.53	69.91	79.87	87.48	**74.29**
ix	**47.63**	41.58	80.14	92.15	82.63	94.99	87.80	7.37	**69.52**
iy	**77.44**	95.04	95.97	95.49	95.89	94.84	94.10	86.89	**94.03**
n	**88.63**	95.71	97.72	98.29	97.38	98.16	97.12	88.10	**96.07**
s	**88.43**	97.82	96.71	95.08	96.11	91.95	91.94	99.35	**95.57**
z	**42.50**	74.81	82.19	73.04	42.62	60.33	55.69	12.91	**57.37**
avg.	**60.39**								**82.50**

Table 3.2: Average recognition rates of *SME*-based classification compared to sMfcc features. Even phonemes like *ih* and *ix* that are hard to tell apart and often merged in experimental setups ([52], e.g.) are separated relatively well. While the state columns as well as the *avg.* column give an impression of the within class wise recognition gains, the **clean** column shows the real improvements after having tested all samples against all *SME* classes and applied majority voting. Following the latter, the overall relative recognition gain is approximately 36.61%, where each phoneme of the testset was classified by all pairwise SVMs of all states and the majority of votes was taken as the final result.

as completely different ones, vocals and consonants. Furthermore, both large and rather small amounts of training data must be in the subset, so that it as well as possible represents the complete set it was taken from.

3.3 *SME* probability classification

The feature extraction, svmlight and HTK setups equal those given in the previous section. For computing the probabilities, the training set is split into two distinct sets, where 70% are used for training of the SVMs while the remaining 30% serve as data for the estimation of the parameters a, b of the sigmoid function. Again, the *SME* weights are set to $w_{\mathcal{H}} = w_{\mathcal{H}_T} = 0.5$. Training of the pairwise support vector classifiers within svmlight is done using *leave-one-out* cross validation to additionally reduce any biases originating from the training data as much as possible.

We trained 55 models in a one-vs-one setup and illustrate the recognition rates in table 3.3. Comparing the results to GMM-classification on standard MFCC-vectors using diagonal covariance matrices and 16 Gaussian mixtures, one can already see the recognition boost when sequence features are considered. When using correlation features and applying the *SME*-concept, it can clearly be seen that the probability estimates are reasonable and even offer a noticeable recognition improvement.

	SVM	Prob	SVM	SVM	GMM 16
	70	70	100	sMFCC 100	MFCC
aa	90.03	88.80	90.40	69.26	65.89
ae	84.17	85.17	86.08	59.79	49.81
ay	85.11	87.46	87.60	52.22	49.94
eh	79.43	77.59	82.69	44.22	37.46
ey	81.47	83.02	85.37	56.39	48.61
ih	74.19	74.85	78.38	37.82	30.00
ix	69.31	70.55	73.38	47.63	37.59
iy	93.92	94.79	95.68	77.44	71.50
n	95.87	96.31	96.28	88.63	82.42
s	95.52	94.11	95.66	88.43	70.35
z	52.97	63.44	63.35	42.50	59.93
avg.	82.00	83.28	84.95	60.39	54.94

Table 3.3: The first two columns show the rates of *SME*-feature based SVM-classification and that of the second multiclass classification method described in [30], using the decision function 1.90. For reasons of easier comparison, columns three and four summarize the results form the experiments of the previous sections. sMFCC represents results for the *sequence*-MFCC vectors used for reasons of a fair evaluation as described in section 3.2. The last column shows the results of a GMM-training and -classification using 16 Gaussian mixtures and ordinary 39-dimensional MFCC-vectors including Δ and $\Delta\Delta$. All entries are recognition rates averaged over the seven states.

The detailed *SME*-confusion tables for this experiment are given in appendix A.2.

3.4 From *SME* features to continuous speech

Advancing from phoneme classification to speech recognition, we first of all have to consider the complete set of phonemes. Following [53], we reduced the amount of classes by merging similar phonemes of the complete TIMIT data set. Figure 3.4 shows the final set of classes. As one can see, in contrast to the cited work we did not include any stops in our experiments.

aa_ao	ae	ah_ax	aw	ay	b	
ch	d	dh	dx	eh	el_l	
em_m	en_n	eng_ng	er	ey	f	
g	hv	ih_ix	iy	jh	k	
ow	oy	p	r	s	sh_zh	t
th	uh	uw_ux	v	w	y	z

Table 3.4: Final set of phonemes used for the SVM-probability experiment. Stops are not considered.

Dealing with continuous speech, several decisions have to be made about both the process of training and the setup of the final recognizer. While focusing on important characteristics of the *SME*-concept, we also want to keep things comparable to standard methods. This is done by adopting a certain level of common consensus: For our inaugural experiments this amounts to being as close as possible to a standard tristate setup for individual monophones.

Let us recall that one starting point for progressing from results derived from deterministic classifications to posterior probabilities was, to use multiclass SVM probability estimates as direct values for a standard Viterby decoder. The individual steps based on this premise, given the *SME* setup, are now explained in detail.

(i) In a first step, the up to seven (or even eight, including the in this work left out case of the three-frame state **SME**, which occurs for short training or test samples) states are reduced to three. They are to serve as the usual tristate representations of phonemes within HMMs. The primal decision in this context is how to combine or reduce the *SME* states appropriately in a manner such that the emerging topology resembles the common and frequently used tristate acoustic model using a three state *start, middle* and *end* partitioning. Let us have a quick look at the origin of this major problem.

Within imbalanced phoneme class sizes across the *SME* states:
As already mentioned in section 2.5.3, statement (4), not all of the states are necessarily represented to an adequate degree by each class. Consider, for instance, a phoneme sample comprised of five speechframes. The partitioning presented in section 2.5.3 and figure 2.8 in this case has one *S* and *E*-subframe each and three *M*-subframes. It therefore only contributes to state classes *SMM, MMM* and *MME*. Phonemes *b, dx* and *g* are examples of such classes, where many states are underrepresented, see table 3.5. In general, from statements (1) and (4) in section 2.5.3 we deduce that due to the broad variance in the distribution of phoneme lengths across the classes, the amount of data for different states – for instance the overall amount of data for states *SSS, MMM, EEE* compared to that of states *SSM, SMM, MME, MEE* – varies severely by construction of the *SME*-partitioning described in section 2.5.3.

No matter how it is done, any approach reducing or merging partitions cancels out one of the most substantial foundations of the *SME*-concept and characteristics the underlying features to a certain degree: Both are motivated by the transitions within a phoneme and the correlation along with this, and some amount of information stemming upon this concept, most prominent especially in the associated *mixture*- or *transition*-states *SSM, ..., MEE* and computed by correlation with the center vector combination (see again section 2.5.3), is lost.

Summarized,

(a) the number of samples of different states within the classes themselves...

(b) the number of samples of the same state for different phonemes...

...are subject to strong variance.

(ii) The $3 \cdot \frac{M\cdot(M-1)}{2}$ pairwise classifiers for the new groups of states as well as their SVM-probabilities have to be computed for all phonemes.

(iii) As we are aiming at multiclass SVM probability estimations, the cross-correlation of an observation with all cumulative, observation specific class center vectors C_ω^m (section 2.5.3) have to computed via the linear transform matrices T_{m_s}, T_{m_m} and T_{m_e} comprised of the respective center vectors next. This delivers the sought cross-correlation features. Afterwards, we compute all $3 \cdot \frac{M(M-1)}{2}$ decision values of the latter, that is evaluate the feature for all SVMs of all classes and their three states chosen by step (i).

With continuous speech recognition being the final step of our experiments, at each step of its analysis the decoder needs probability information about all states and all phonemes. Given an observation ω and classes $m = 1, \dots, M$, this information is the probability for each class and state given the sample, where we have to keep in mind that the states representing the start-, middle and end period within the decoder are class dependent according to step (i).

(iv) Denoting the class dependent states with s_m^i, $i = 1, \dots, 3$, we use the results of steps (ii) and (iii) to compute the prior probabilities $\tilde{\mathbf{p}} = p(\omega|m, s_m^i)$ for each of the $3 \cdot M$ decision values and, using Bayes' Rule, to get the sought posterior probabilities $\mathbf{p} = p(m, s_m^i|\omega)$, where $\mathbf{p}$ is a vector of size M itself. However, it contains far more components than necessary: The posteriors were computed from the correlation features. Being determined themselves by the transform using the cumulative class center vectors, they already intrinsically include information on prior class decision emerging from the SVM classifiers.

As a consequence, the probability vector $\mathbf{p}$ is composed of $M - 1$ redundant probabilities derived from unsuitable class centers. Those are the $C_\omega^{\tilde{m}}$ where $m \neq \tilde{m}$. Omitting the above mentioned components and keeping only those where $m = \tilde{m}$ equates to using the specific components where the cumulative center vector coincides with the prior class. The latter comprise the final vector used for the decoding, which is composed of $3 \cdot M$ posterior probability values and is of the form

$$\begin{pmatrix} p(m = 1, s_{m=1}^1|\omega) \\ \vdots \\ p(m = M, s_{m=M}^1|\omega) \\ p(m = 1, s_{m=1}^2|\omega) \\ \vdots \\ p(m = M, s_{m=M}^2|\omega) \\ p(m = 1, s_{m=1}^3|\omega) \\ \vdots \\ p(m = M, s_{m=M}^3|\omega) \end{pmatrix}.$$

In the light of (i), table 3.5 shows the numbers of samples of each of the phonemes' original *SME* states.

Remark 3.1
Taking into account the effects of (i), for the experiments on continuous speech recognition utilizing the *SME* structures, we decided on pooling *SSS, SSM*, *SMM, MMM, MME* and *MEE, EEE* and thereby trading better size balance for feature similarity across states. It is clear that the way of partitioning the original data and the way of merging or reducing states are directly connected and that the recognition rates will depend strongly on that choice. For this reason, section 3.6 evaluates the effect of one of these aspects, the pooling of the states and further degradation resulting from that specific information reduction.

To this effect, the SVMs are trained and the probabilities computed on the combined training data with three center vector equaling the component wise average of the former individual classes of each merged state, respectively.

	SME	SSS	SSM	SMM	MMM	MME	MEE	EEE
aa,ao	14	6801	3564	3664	10313	3664	3564	6801
ae	1	4502	1981	1995	6501	1995	1981	4502
ah,ax	537	1669	2701	4409	4705	4409	2701	1669
aw	0	2135	697	697	2851	697	697	2135
ay	0	4902	1740	1750	6656	1750	1740	4902
b	223	1	4	47	13	47	4	1
ch	4	472	668	734	1135	734	668	472
d	383	12	98	358	133	358	98	12
dh	496	51	338	989	512	989	338	51
dx	768	0	12	335	65	335	12	0
eh	18	2432	2638	2850	5073	2850	2638	2432
el,l	298	1339	2923	4289	4450	4289	2923	1339
em,m	200	735	2051	2859	2935	2859	2051	735
en,n	678	966	2840	5093	4235	5093	2840	966
eng,ng	103	192	600	950	838	950	600	192
er	1	2594	1520	1544	4111	1544	1520	2594
ey	0	3849	1975	1987	5784	1987	1975	3849
f	30	2386	1815	1950	4253	1950	1815	2386
g	312	1	14	210	60	210	14	1
hv	35	200	493	634	725	634	493	200
ih,ix	1017	2224	5388	8786	8105	8786	5388	2224
iy	36	3883	3681	4084	7592	4084	3681	3883
jh	76	193	463	793	717	793	463	193
k	476	394	1432	2332	1890	2332	1432	394
ow	0	2893	1481	1489	4342	1489	1481	2893
oy	0	994	304	304	1302	304	304	994
p	368	95	686	1318	876	1318	686	95
r	486	580	1870	3326	2720	3326	1870	580
s	9	8091	5171	5343	13264	5343	5171	8091
sh,zh	1	1971	1238	1255	3170	1255	1238	1971
t	581	315	1341	2304	1824	2304	1341	315
th	29	598	567	640	1146	640	567	598
uh	18	221	348	446	577	446	348	221
uw,ux	37	2186	1528	1756	3749	1756	1528	2186
v	147	264	995	1639	1393	1639	995	264
w	292	571	998	1530	1619	1530	998	571
y	132	130	370	640	543	640	370	130
z	35	1956	2792	3186	4806	3186	2792	1956

Table 3.5: The phoneme classes and the classes' numbers of training samples for each state. The vertical lines between the *SMM, SMM* and the *MME, MEE* states illustrate the grouping of the states for the reduction from seven to three states.

3.5 Speech recognition experiments

The quality of the new approach is determined by comparing continuous speech recognition performance on the TIMIT testset using the HTK toolset. The standard MFCC-Δ-$\Delta\Delta$ setup follows the tutorial, where we merge phonemes in the same manner as in the *SME* case, see table 3.4 and recognition was performed using monophonic HMMs in order to get a useful comparison.

For the *SME* based data, the first steps equal those of the phoneme classification procedure: Center vectors for each class serve as a foundation to compute correlation features, build SVMs for training and classify samples. Each utterance is handled piecewise, a new frame starting every 25 ms with 10 ms overlap. The SVM classified frames are transformed into posterior probabilities using the precomputed pairwise sigmoidal functions, see section 1.9, the results are written to a file afterwards. HTK was extended such that in the case of *SME* based recognition (selectable via a new command line parameter) the output probabilities are directly read from the above mentioned file, substituting the computation of HMMs in the ordinary MFCC-Δ-$\Delta\Delta$ setup.

For both variants, the recognizer parameters for HVite were set to no *word insertion penalty* (-p 0) applying a *grammar scale factor* of 5.0 (-s 5.0). Our new approach did not include any training data for silence, as extraction of (quasi) silence did not work reliably with tools such as HCopy. Silence was thus cut from the utterances as accurate as possible when computing the framewise *SME* features and their probabilities. As a consequence, in contrast to the standard setup, where an HMM is also created for silence, the *SME* recognition procedure had to perform evaluations without this improvement.
The training dictionary contains 4877 different words. Taking into account the pronunciation variations of words, the total number of words is 17913 including 59 out of vocabulary words (0.3%). The language model uses a trigram configuration with a perplexity of 234.844. The testset consisted of 1303 sentences and an overall amount of 10713 words. Tables 3.6 and 3.7 compare the sentence and word recognition results of both experiments.

	Sentences	Correct
MFCC-Δ-$\Delta\Delta$	1303	1136 (87.18%)
SME	1303	1104 (84.73%)

Table 3.6: MFCC-Δ-$\Delta\Delta$ vs. *SME* results for sentence recognition.

	Words	Correct	Substitutions	Deletions	Insertions
MFCC-Δ-$\Delta\Delta$	10713	9856 (92.00%)	497 (4.64%)	201 (1.88%)	159 (1.48%)
SME	10713	9760 (91.11%)	462 (4.31%)	209 (1.95%)	282 (2.63%)

Table 3.7: MFCC-Δ-$\Delta\Delta$ vs. *SME* results for word recognition.

3.6 Impact of the reduction

The previous section 3.5 revealed slightly worse results for the SVM-*SME* based speech recognition compared to the common MFCC-Δ-$\Delta\Delta$ feature vector setup: a drop of the recognition rate of about 2.81% for sentences and of 0.97% for words. Whereas a certain amount of the degradation clearly originates from the missing silence class/model in the *SME* setup, in this section we want to examine the additional impact of the two most severe steps accounting for this decline within the process of reducing the original seven state *SME* features to the three state variant used to compare the *SME* based recognition to a common MFCC-Δ-$\Delta\Delta$ based automatic speech recognition system. It becomes obvious that the information loss is rather strong and that we can expect far better speech recognition results when using the unmodified, original *SME*-structure and features. To keep things clear, we again conduct the experiments on the subset of eleven phonemes used for our initial classification evaluations.

3.6.1 Reduction of the number of states

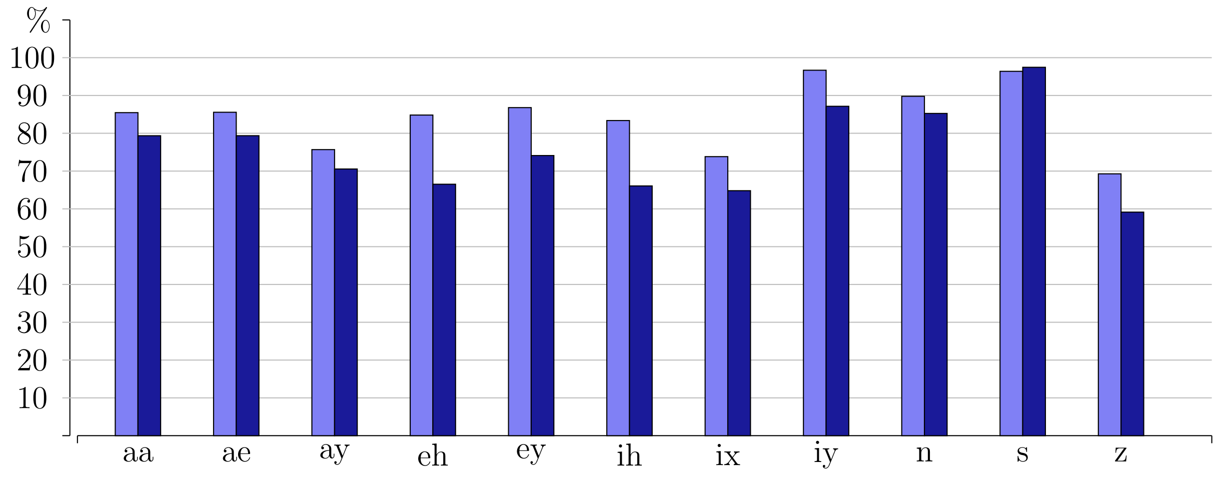
The first of the two major cuts occurs when reducing the seven states to three. We repeat some details for reasons of clarity: The reduction includes several steps, the first of which is to choose three instead of seven representative states and classes (*SSS, SSM,* $\cdots$*, MEE, EEE*) along with their representative, accumulated center vectors. For the following comparison the states were combined in the same way as described in remark (3.1). All SVMs were retrained accordingly. It becomes clear immediately that the larger amount of training data is not capable of absorbing the effects of the smaller amount of center vectors and the less accurate and rougher correlation resolution. Table 3.8 and its graphical pendant depict the results. Test samples were classified over all 55 binary SVMs of all states (three or seven for reduced and original *SME* data) and a simple histogram (= majority vote) served as the final classification decision. The detailed confusion tables are given in appendix B. Let us mention that we used a different parameter value for the SVMs than in section 3.2, setting $\gamma = 0.001$ for both cases. The parameters may differ from the experiments conducted for section 3.1 and appendix A, as we re-estimated them due to larger training sets.

Multiple kernel classifications are considered this time, as the effects readily become clear enough.

states	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
7	85.46	85.56	75.68	84.82	86.79	83.37	73.82	96.68	89.78	96.39	69.27
3	79.35	79.36	70.55	66.52	74.11	66.07	64.81	87.14	85.25	97.46	59.14

Table 3.8: Overall recognition rates for the eleven phonemes encompassing subset, comparing original and reduced *SME* structured classification.

Recognition rates: Seven (light blue) vs three (dark blue)*SME* states



3.6.2 Merging state probability vectors

During the process of preparing the reduced, three states encompassing *SME* posterior probability vectors necessary for the recognizer there are a total of three probability vectors for any observation, either of them associated with one of the three reduced states. The cumulative number of components of these three sums up to $3M$ components, where M is the number of classes. As this is a quite large pooled vector, the three state probability vectors are transformed into one single vector of size M by componentwise arithmetic averaging. Revisiting the detailed description of the computation of the *SME* features in section 2.5, where we used MFCC vectors of length 13 as a starting point, we deduce that for the full set of phonemes by construction the MFCC-Δ-$\Delta\Delta$ setup has probability vectors of length 39, the *SME* setup of length 38 after taking averages.

The averaging however results in unwanted and unfavourable deviations of the probability vectors which can lead to swaps of the largest components. The results in this section unveil these negative effects on the subset of 11 phonemes using the result vectors computed as described in the previous section. We start by illustrating the origin of the classification degradation by a simple but real example. Below, following the phonemes, the vectors to the left of the arrow show the three individual probability vectors for the states *S, M, E*, respectively, the one to the right the remaining one after component wise averaging.

$$\begin{array}{c} \vdots \\ ay \\ \vdots \\ ih \\ \vdots \end{array} \qquad \begin{pmatrix} \vdots \\ 0.77 \\ \vdots \\ 0.20 \\ \vdots \end{pmatrix} \begin{pmatrix} \vdots \\ 0.52 \\ \vdots \\ 0.30 \\ \vdots \end{pmatrix} \begin{pmatrix} \vdots \\ 0.08 \\ \vdots \\ 0.91 \\ \vdots \end{pmatrix} \qquad \Longrightarrow \qquad \begin{pmatrix} \vdots \\ 0.46 \\ \vdots \\ 0.47 \\ \vdots \end{pmatrix} \tag{3.1}$$

The observation in question is a phoneme from class *ay*, state *S*. The highest probabilities are in the corresponding correct components for two of the states, *S* and *M*, whereas for the *E*-state the probability in the component representing *ih* exceeds all others. Averaging in such a case leads to inadequate probabilities and henceforth incorrect recognitions. Instead, using a majority vote over the highest number of components for instance removes these misclassifications without any negative effect (that is misclassifying otherwise correctly classified phonemes). Tables and figures C.1 to C.11 in appendix C reveal the exact recognition rate differences between both variants.

It is also clear that not for all states and all phonemes the difference is statistically significant. The best examples for this is phoneme *s*. Table B.19 on page 123 already shows, that the recognition rates for this phoneme already are at a high level throughout the states. The impact of the averaging operation is clearly less than for other phonemes, especially when the performance on different states is not as homogeneous. To this effect, the improvements without averaging are in general in line with the results and texture of the reduced states confusion tables given in appendix C.

3.7 Conclusions and prospects

In this thesis we have introduced new MFCC correlation features for phoneme classification and automatic speech recognition. In their light we proposed a new family of support vector classifiers associated directly with the specific structure of the underlying features. Both the classifier topology and the reproducing kernel utilized for training and evaluation within those were deduced directly from the correlation operation, which itself was motivated by two main characteristics of speech: quasi stationarity and transitions between phonemes. The new approach yielded promising recognition improvements especially for phoneme classification tasks. Reasons for deficits in the case of continuous speech recognition were traced back to potential causes. In this final chapter we want to analyze the results further and show additional directions.

The foundations for the partitioning of phonemes into the *SME*-subclasses *SME, SSS, . . ., EEE* are on one hand the partitioning theme itself, on the other hand the center vectors c^l_σ, $1 \le l \le L$, $\sigma \in \{s, m, e\}$. Both are illustrated in details in section 2.5.3. The first one, the partitioning scheme, was chosen to be rather simple and leads to certain imbalances across the sizes of the *SME*-subclasses. For longer training samples, the outmost subclasses will include a comparatively small numbers of samples, and the class sizes strongly increase symmetrically towards the middle class *MMM*. For short phonemes such as *b* or *g*, this partitioning scheme leads to an even more severe situation The margin classes are extremely underrepresented or are even empty. The problem of class imbalances already smoldering in the original dataset is carried over and potentially amplified in the process of partitioning both within the same and across the different subclasses. Table 3.5 gives an overview of the strongly varying *SME*-class sizes resulting from the initial dataset.

Second, the center vectors are supposed to be representative for their classes with respect to their states *s, m, e*. For classes with low within-class variance, they surely are good representatives. In general however, considering several such centers for instance by pre-clustering each class and state into two ore more clusters and using the multiple achieved centers of the different clusters will presumably reward us with a more suitable and accurate representation. Another potentially big gain can be achieved by taking the cross correlation of more than just three adjacent vectors – see section 2.5.2 – as a root for computing our features. For both approaches, to keep the resulting, (pooled subclass) feature vectors reasonably small when using several center vectors and correlation features, data reduction methods such as (kernel) PCA or LDA can be considered as a subsequent step, while this is non trivial when dealing with multiple classes – see for instance [54]. Table 3.9 illustrates this idea.

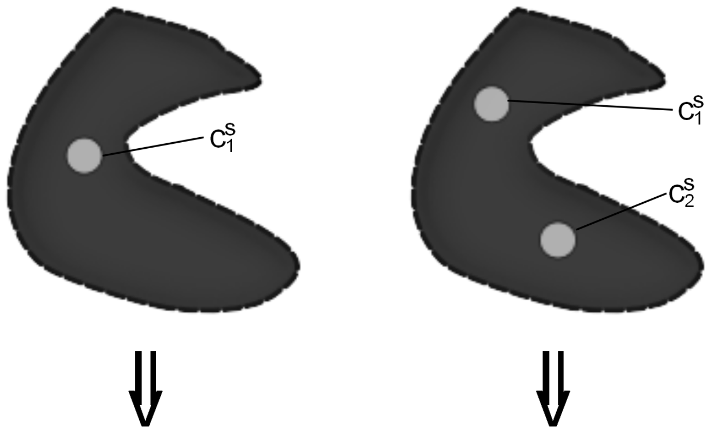
All in all, the importance and efficiency of the partitioning and thus of using time position specific data becomes obvious when looking at the results of section 3.6.1. As a consequence, finding an appropriate balance between this and the above mentioned design aspects – comparable to the weighting of acoustic and language model of a recognizer – will probably improve recognition further to quite some extend.

Using several centers also emphasizes the main idea of using correlation, or better autocorrelation, the information of which depends on time lags separating features and observations. Offering one averaged vector only for this information surely blurs and reduces such data to quite some degree, especially for data deviating heavily from this representative, the strategy of using several center is recommendable. The problem of imbalanced classes becomes especially crucial in the context of learning the classifiers. While strong underrepresentation such as mentioned before will affect any learner, alternate methods instead of SVMs for large but manageable differences are worth to be considered. In this context and under the premise of sufficient amounts of training data, ensemble classifier such as **random forests** are reasonable candidates.

A **Random Forests**$^{(\mathrm{tm})}$ classification system is an ensemble classifier comprised of a set of individual and (as well as possible) decorrelated decision tree classifiers, where each such tree is generated during training by randomized sample choices iid but with one underlying distribution for all trees, see [55] for details. Fan ([56]) introduced a method utilizing reproducing kernels and aims at combining the advantages of modeling non linear (local) decisions and tree based classification systems. After recursively creating a data partitioning, the kernels are applied as decision makers for splitting the data for each tree in the forest. This is especially appealing, as the specific kernel deduced in our work can be integrated into the kernel random forest in the same manner as done within the SVMs.

Being able to address multiclass problems by nature in contrast to the binary SVMs, training will probably be faster and classification more reliable. For instance, unclassifiable regions as common for multiclass SVMs (see for instance [16], chapters one and two) will not exist due to the decision tree based concept. More important, classification for large datasets and a large number of classes (in the light of the *SME* topology) is far more efficient. Last, random forests offer a better understanding of relations between different classes, which in the context of similar phonemes can be exploited. The interested reader is referred to the above mentioned references as well as [57] for a detailed description of ordinary random forests and those utilizing kernel based strategies.

Pre-Clustering



training samples $\longrightarrow$ correlation correlation

$\downarrow$ **P C A, L D A, . . .** $\downarrow$

features(c_1^s) features(c_1^s, c_2^s)

Table 3.9: Replacing single center vectors by multiple. Class indices are omitted and state s is assumed in this example comparing the single center vector approach to one with two center vectors. Given incoming training samples, the left part of the figure illustrates the current single center based path, the right one an alternative two center approach. For the first, correlation of samples with the center vector is performed as usual. For the latter, two centers are determined for instance by a preceding clustering method. After correlation is computed with both center vectors, methods such as for instance (Kernel) PCA or LDA can be applied subsequently to reduce dimensionality. The vectors are pooled afterwards to get the final, cumulative feature vector which depends on either one or several – here two – representative centers.

Appendix A – Confusion tables

Due to precision restriction and rounding, columns might not add up to 100%.

A.1 SME-structured SVMs

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	73.68	87.72	91.58	94.41	96.01	96.05	91.54	**90.141**
ae	0.50	0.44	0.14	0.19	0.29	0.58	2.70	**0.691**
ay	24.61	9.21	6.42	4.47	2.71	1.61	1.64	**7.239**
eh	0.85	1.90	1.14	0.42	0.43	1.32	2.84	**1.271**
ey	0.14	0.00	0.00	0.05	0.00	0.00	0.57	**0.109**
ih	0.14	0.15	0.00	0.14	0.14	0.00	0.21	**0.111**
ix	0.07	0.29	0.14	0.19	0.29	0.15	0.00	**0.161**
iy	0.00	0.00	0.14	0.09	0.00	0.00	0.07	**0.043**
n	0.00	0.00	0.14	0.00	0.00	0.15	0.00	**0.041**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.292	0.285	0.047	0.143	0.146	0.427	**0.191**

Table A.1: Confusion table of phoneme *aa*, using *SME*-SVM classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ae	92.27	82.38	81.30	92.61	78.67	77.13	85.69	**84.293**
ay	0.51	0.46	1.24	1.49	6.34	8.19	9.10	**3.904**
eh	4.55	16.23	15.92	5.56	13.29	13.60	4.91	**10.580**
ey	2.60	0.15	0.00	0.00	0.00	0.00	0.00	**0.393**
ih	0.00	0.15	0.31	0.15	0.31	0.46	0.14	**0.217**
ix	0.00	0.46	1.08	0.15	0.93	0.31	0.00	**0.419**
iy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
n	0.07	0.15	0.15	0.05	0.46	0.31	0.07	**0.180**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.072	**0.010**

Table A.2: Confusion table of phoneme *ae*, using *SME*-SVM classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	17.45	14.46	5.83	1.97	1.54	0.00	0.00	**5.893**
ae	1.38	1.03	0.69	1.12	4.46	12.22	15.79	**5.241**
ay	80.83	83.48	91.08	96.35	90.05	75.56	79.22	**85.224**
eh	0.17	0.69	1.72	0.56	2.92	10.84	3.85	**2.964**
ey	0.11	0.17	0.00	0.00	0.00	0.00	0.00	**0.040**
ih	0.00	0.00	0.00	0.00	0.00	0.52	1.03	**0.221**
ix	0.00	0.17	0.51	0.00	0.69	0.69	0.00	**0.294**
iy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
n	0.00	0.00	0.17	0.00	0.34	0.17	0.11	**0.113**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.057	0.000	0.000	0.000	0.000	0.000	0.000	**0.008**

Table A.3: Confusion table of phoneme *ay*, using *SME*-SVM classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.12	0.10	0.09	0.05	0.00	0.00	0.00	**0.051**
ae	20.51	6.07	4.11	5.50	3.36	5.47	19.78	**9.257**
ay	0.49	0.10	0.37	1.46	1.96	3.24	21.84	**4.209**
eh	59.34	86.23	89.54	89.27	89.64	86.74	55.95	**79.530**
ey	18.08	1.52	0.28	0.38	0.00	0.00	0.12	**2.911**
ih	0.97	2.73	1.68	1.40	1.87	2.73	2.31	**1.956**
ix	0.36	3.04	3.55	1.83	2.80	1.62	0.00	**1.886**
iy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
n	0.12	0.20	0.37	0.11	0.37	0.20	0.00	**0.196**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.4: Confusion table of phoneme *eh*, using *SME*-SVM classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.14	0.14	**0.040**
ae	12.65	1.42	0.56	0.23	0.00	0.00	0.00	**2.123**
ay	0.28	0.85	0.98	1.19	0.14	0.00	1.82	**0.751**
eh	10.55	11.24	7.99	3.39	1.12	0.57	0.00	**4.980**
ey	74.42	78.52	78.40	88.70	81.21	82.22	87.56	**81.576**
ih	1.54	4.98	8.13	4.03	11.78	9.10	2.73	**6.041**
ix	0.35	2.13	2.38	1.56	1.26	1.14	0.07	**1.270**
iy	0.14	0.85	1.40	0.87	4.49	6.83	7.62	**3.171**
n	0.07	0.00	0.14	0.05	0.00	0.00	0.00	**0.037**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.07	**0.010**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.5: Confusion table of phoneme *ey*, using *SME*-SVM classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.08	0.00	0.00	0.10	0.34	**0.074**
ae	3.21	0.67	0.64	1.15	0.16	0.19	1.35	**1.053**
ay	0.00	0.10	0.08	0.36	0.16	0.10	3.72	**0.646**
eh	2.54	2.49	2.63	2.60	2.23	1.92	1.52	**2.276**
ey	12.18	4.41	3.99	10.54	3.43	2.59	5.08	**6.031**
ih	70.90	73.73	68.64	69.53	69.91	79.87	87.48	**74.294**
ix	4.23	16.59	22.43	14.54	23.70	14.96	0.17	**13.803**
iy	6.77	2.01	1.52	1.27	0.40	0.29	0.34	**1.800**
n	0.17	0.00	0.00	0.00	0.00	0.00	0.00	**0.024**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.6: Confusion table of phoneme *ih*, using *SME*-SVM classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.53	0.00	0.06	0.17	0.06	0.00	0.53	**0.193**
ae	2.11	0.36	0.06	0.76	0.00	0.00	0.53	**0.546**
ay	0.00	0.00	0.00	0.17	0.00	0.00	1.05	**0.174**
eh	3.68	1.20	0.22	0.25	0.06	0.24	1.05	**0.957**
ey	5.26	0.60	0.22	0.93	0.06	0.24	1.05	**1.194**
ih	45.26	17.58	7.18	14.75	4.79	11.72	88.42	**27.100**
ix	41.58	80.14	92.15	82.63	94.99	87.80	7.37	**69.523**
iy	1.05	0.00	0.11	0.34	0.06	0.00	0.00	**0.223**
n	0.53	0.12	0.00	0.00	0.00	0.00	0.00	**0.093**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.7: Confusion table of phoneme *ix*, using *SME*-SVM classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ae	0.15	0.00	0.00	0.00	0.00	0.00	0.00	**0.021**
ay	0.00	0.00	0.00	0.04	0.06	0.07	0.44	**0.087**
eh	0.22	0.00	0.00	0.00	0.06	0.14	0.37	**0.113**
ey	0.67	0.65	1.10	2.48	2.97	4.25	11.48	**3.371**
ih	3.85	2.30	2.00	1.02	1.29	1.08	0.67	**1.744**
ix	0.07	1.08	1.29	0.55	0.71	0.29	0.07	**0.580**
iy	95.04	95.97	95.49	95.89	94.84	94.10	86.89	**94.031**
n	0.00	0.00	0.13	0.04	0.06	0.07	0.07	**0.053**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.8: Confusion table of phoneme *iy*, using *SME*-SVM classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ae	1.90	0.12	0.00	0.08	0.06	0.24	7.14	**1.363**
ay	0.00	0.00	0.00	0.25	0.06	0.00	0.48	**0.113**
eh	1.90	1.08	0.70	1.06	0.70	1.56	2.38	**1.340**
ey	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ih	0.48	0.00	0.06	0.25	0.06	0.36	0.00	**0.173**
ix	0.00	0.72	0.95	0.98	0.95	0.72	0.48	**0.686**
iy	0.00	0.36	0.00	0.00	0.00	0.00	0.48	**0.120**
n	95.71	97.72	98.29	97.38	98.16	97.12	88.10	**96.069**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.95	**0.136**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.9: Confusion table of phoneme *n*, using *SME*-SVM classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ae	0.00	0.00	0.00	0.00	0.05	0.05	0.00	**0.014**
ay	0.00	0.00	0.00	0.00	0.00	0.00	0.07	**0.010**
eh	0.00	0.00	0.00	0.00	0.00	0.00	0.03	**0.004**
ey	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ih	0.00	0.00	0.00	0.00	0.05	0.00	0.00	**0.007**
ix	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
iy	0.00	0.00	0.00	0.00	0.00	0.05	0.00	**0.007**
n	0.14	0.00	0.00	0.00	0.00	0.05	0.00	**0.027**
s	97.82	96.71	95.08	96.11	91.95	91.94	99.35	**95.566**
z	2.05	3.29	4.92	3.89	7.94	7.90	0.55	**4.363**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.10: Confusion table of phoneme *s*, using *SME*-SVM classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ae	0.00	0.00	0.00	0.00	0.00	0.00	0.31	**0.044**
ay	0.00	0.00	0.00	0.00	0.00	0.00	0.62	**0.089**
eh	0.00	0.00	0.00	0.00	0.00	0.00	0.16	**0.023**
ey	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ih	0.00	0.00	0.00	0.00	0.00	0.21	0.00	**0.030**
ix	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
iy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
n	0.16	0.00	0.10	0.00	0.00	0.00	0.00	**0.037**
s	25.04	17.81	26.86	57.38	39.67	44.10	86.00	**42.409**
z	74.81	82.19	73.04	42.62	60.33	55.69	12.91	**57.370**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.11: Confusion table of phoneme *z*, using *SME*-SVM classification.

A.2 SME-structure based Probabilities

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	70.84	85.23	90.01	93.18	95.01	95.32	92.03	**88.803**
ae	0.50	0.29	0.14	0.19	0.29	0.58	2.28	**0.610**
ay	27.45	12.87	8.27	5.74	3.71	2.63	2.63	**9.043**
eh	0.78	1.17	1.14	0.38	0.57	1.17	2.06	**1.039**
ey	0.28	0.00	0.00	0.05	0.00	0.00	0.64	**0.139**
ih	0.07	0.15	0.00	0.05	0.14	0.00	0.00	**0.059**
ix	0.07	0.15	0.14	0.19	0.14	0.00	0.21	**0.129**
iy	0.00	0.00	0.14	0.24	0.00	0.00	0.07	**0.064**
n	0.00	0.15	0.14	0.00	0.14	0.29	0.07	**0.113**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.12: Confusion table of phoneme *aa*, using *SME*-SVM probability classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.00	0.22	**0.031**
ae	89.60	85.32	84.23	94.24	82.38	76.82	83.60	**85.170**
ay	0.87	0.46	1.08	1.24	5.72	10.66	11.27	**4.471**
eh	4.84	12.67	13.29	3.97	10.36	11.44	4.48	**8.721**
ey	4.62	0.62	0.00	0.25	0.00	0.00	0.00	**0.784**
ih	0.00	0.31	0.46	0.10	0.31	0.62	0.14	**0.277**
ix	0.00	0.15	0.77	0.15	0.77	0.00	0.00	**0.263**
iy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
n	0.07	0.46	0.15	0.05	0.46	0.46	0.22	**0.267**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.07	**0.010**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.13: Confusion table of phoneme *ae*, using *SME*-SVM probability classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	14.70	11.19	4.80	1.37	0.86	0.00	0.00	**4.703**
ae	0.57	1.20	1.20	1.63	5.66	10.84	13.49	**4.941**
ay	84.16	86.92	91.94	96.48	90.74	80.21	81.80	**87.464**
eh	0.11	0.52	1.03	0.52	1.89	7.92	3.21	**2.171**
ey	0.46	0.17	0.34	0.00	0.00	0.00	0.23	**0.171**
ih	0.00	0.00	0.00	0.00	0.00	0.52	0.63	**0.164**
ix	0.00	0.00	0.34	0.00	0.34	0.52	0.46	**0.237**
iy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
n	0.00	0.00	0.34	0.00	0.51	0.00	0.17	**0.146**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.14: Confusion table of phoneme *ay*, using *SME*-SVM probability classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.24	0.10	0.19	0.11	0.00	0.00	0.00	**0.091**
ae	16.02	8.20	6.16	8.90	5.32	6.68	18.33	**9.944**
ay	0.36	0.20	0.56	1.46	2.33	4.76	24.88	**4.936**
eh	58.98	82.59	86.93	86.19	87.77	85.02	55.10	**77.511**
ey	21.97	3.64	1.03	0.76	0.09	0.00	0.24	**3.961**
ih	1.58	2.73	2.05	1.46	1.96	2.13	1.46	**1.910**
ix	0.73	2.33	2.52	1.02	1.96	1.11	0.00	**1.381**
iy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
n	0.12	0.20	0.56	0.11	0.56	0.30	0.00	**0.264**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.15: Confusion table of phoneme *eh*, using *SME*-SVM probability classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.14	0.07	**0.030**
ae	8.74	1.56	0.42	0.18	0.00	0.00	0.00	**1.557**
ay	0.21	0.28	0.28	0.87	0.14	0.00	1.12	**0.414**
eh	10.90	9.10	6.73	2.65	0.98	0.43	0.00	**4.399**
ey	76.94	79.80	80.08	89.06	83.59	84.35	87.35	**83.024**
ih	2.38	6.12	8.56	4.53	9.26	6.97	2.17	**5.713**
ix	0.63	2.13	2.24	1.42	1.12	1.00	0.35	**1.270**
iy	0.14	1.00	1.54	1.24	4.91	7.11	8.81	**3.536**
n	0.07	0.00	0.14	0.05	0.00	0.00	0.00	**0.037**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.14	**0.020**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.16: Confusion table of phoneme *ey*, using *SME*-SVM probability classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.10	0.16	0.00	0.00	0.10	0.68	**0.149**
ae	2.37	0.67	0.40	0.85	0.24	0.19	1.86	**0.940**
ay	0.00	0.10	0.08	0.24	0.16	0.19	3.55	**0.617**
eh	2.88	1.82	2.31	2.54	2.55	2.21	2.88	**2.456**
ey	9.31	4.03	3.83	10.36	4.71	3.55	5.08	**5.839**
ih	74.28	76.03	70.71	70.93	70.79	79.96	81.22	**74.846**
ix	4.23	15.05	20.51	13.63	20.91	13.42	4.23	**13.140**
iy	6.77	2.21	2.00	1.45	0.64	0.38	0.51	**1.994**
n	0.17	0.00	0.00	0.00	0.00	0.00	0.00	**0.024**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.17: Confusion table of phoneme *ih*, using *SME*-SVM probability classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.53	0.00	0.06	0.17	0.06	0.00	0.53	**0.193**
ae	1.58	0.36	0.06	0.85	0.06	0.00	0.53	**0.491**
ay	0.00	0.00	0.00	0.17	0.00	0.00	0.53	**0.100**
eh	3.68	1.20	0.39	0.59	0.22	0.36	2.11	**1.221**
ey	3.16	0.36	0.22	0.76	0.11	0.24	1.05	**0.843**
ih	47.37	20.69	8.46	16.69	6.35	14.11	69.47	**26.163**
ix	41.58	77.03	90.53	80.34	93.04	85.29	25.79	**70.514**
iy	1.58	0.12	0.17	0.42	0.11	0.00	0.00	**0.343**
n	0.53	0.24	0.06	0.00	0.00	0.00	0.00	**0.119**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.06	0.00	0.06	0.00	0.00	**0.017**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.18: Confusion table of phoneme *ix*, using *SME*-SVM probability classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ae	0.07	0.00	0.00	0.00	0.00	0.00	0.00	**0.010**
ay	0.00	0.00	0.00	0.00	0.06	0.07	0.30	**0.061**
eh	0.30	0.00	0.00	0.00	0.06	0.14	0.37	**0.124**
ey	0.44	0.36	0.77	2.04	3.03	4.32	10.00	**2.994**
ih	3.78	2.09	1.68	0.95	1.16	0.65	0.52	**1.547**
ix	0.07	0.72	0.97	0.44	0.39	0.29	0.00	**0.411**
iy	95.33	96.83	96.39	96.54	95.23	94.46	88.74	**94.789**
n	0.00	0.00	0.19	0.04	0.06	0.07	0.07	**0.061**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.19: Confusion table of phoneme *iy*, using *SME*-SVM probability classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.06	0.08	0.00	0.00	0.00	**0.020**
ae	1.43	0.12	0.06	0.33	0.06	0.24	6.67	**1.273**
ay	0.00	0.00	0.00	0.25	0.06	0.00	0.00	**0.044**
eh	1.90	0.84	0.44	0.74	0.51	1.32	1.90	**1.093**
ey	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ih	0.48	0.00	0.06	0.25	0.06	0.36	0.00	**0.173**
ix	0.00	0.36	0.89	0.82	0.76	0.24	1.43	**0.643**
iy	0.00	0.48	0.25	0.00	0.00	0.00	0.95	**0.240**
n	96.19	98.20	98.23	97.55	98.54	97.84	87.62	**96.310**
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
z	0.00	0.00	0.00	0.00	0.00	0.00	1.43	**0.204**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.20: Confusion table of phoneme *n*, using *SME*-SVM probability classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ae	0.00	0.00	0.00	0.00	0.05	0.05	0.00	**0.014**
ay	0.00	0.00	0.00	0.00	0.00	0.00	0.07	**0.010**
eh	0.00	0.00	0.00	0.00	0.00	0.00	0.03	**0.004**
ey	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ih	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ix	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
iy	0.00	0.00	0.00	0.00	0.05	0.00	0.00	**0.007**
n	0.14	0.00	0.00	0.00	0.00	0.00	0.00	**0.020**
s	97.34	95.86	94.05	94.11	89.49	90.72	97.17	**94.106**
z	2.52	4.14	5.95	5.89	10.40	9.23	2.73	**5.837**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.21: Confusion table of phoneme *s*, using *SME*-SVM probability classification.

	SSS	SSM	SMM	MMM	MME	MEE	EEE	**avg.**
aa	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ae	0.00	0.00	0.00	0.00	0.00	0.00	0.31	**0.044**
ay	0.00	0.00	0.00	0.00	0.00	0.00	0.62	**0.089**
eh	0.00	0.00	0.00	0.00	0.00	0.00	0.16	**0.023**
ey	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
ih	0.00	0.00	0.00	0.00	0.00	0.21	0.00	**0.030**
ix	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
iy	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.000**
n	0.16	0.00	0.10	0.00	0.00	0.00	0.00	**0.037**
s	20.22	15.13	22.18	48.73	35.85	41.20	71.07	**36.340**
z	79.63	84.87	77.72	51.27	64.15	58.58	27.84	**63.437**
uncl.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.000**

Table A.22: Confusion table of phoneme *z*, using *SME*-SVM probability classification.

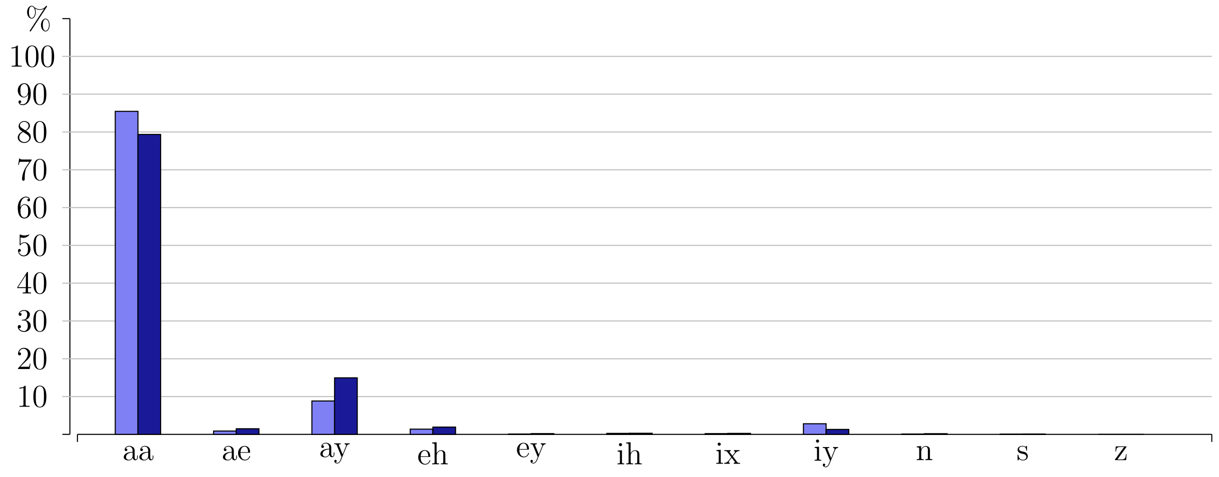
Appendix B – SME recognition rates, unaltered vs. reduced

Phoneme aa

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	**56.56**	0.85	40.76	0.71	0.00	0.14	0.00	0.99	0.00	0.00	0.00
SSM	**84.80**	0.15	7.60	1.32	0.00	0.00	0.00	6.14	0.00	0.00	0.00
SMM	**83.02**	0.00	5.42	1.00	0.00	0.00	0.00	10.13	0.43	0.00	0.00
MMM	**95.08**	0.32	2.57	0.64	0.00	0.00	0.00	1.39	0.00	0.00	0.00
MME	**97.00**	0.43	1.00	0.57	0.00	0.00	0.14	0.86	0.00	0.00	0.00
MEE	**95.47**	1.02	1.90	1.17	0.00	0.15	0.15	0.00	0.15	0.00	0.00
EEE	**86.32**	3.38	2.54	4.23	0.56	1.55	1.13	0.14	0.00	0.14	0.00
avg.	**85.46**	0.88	8.83	1.38	0.08	0.26	0.20	2.81	0.08	0.02	0.00

Table B.1: Recognition rates for phones *aa*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	**59.30**	0.73	39.16	0.00	0.14	0.03	0.02	0.61	0.01	0.00	0.00
MMM-3	**92.81**	0.32	2.39	0.93	0.00	0.00	0.02	3.19	0.34	0.00	0.00
EEE-3	**85.93**	3.41	3.32	4.83	0.47	0.84	0.74	0.14	0.18	0.14	0.00
avg.	**79.35**	1.49	14.96	1.92	0.20	0.29	0.26	1.31	0.18	0.05	0.00



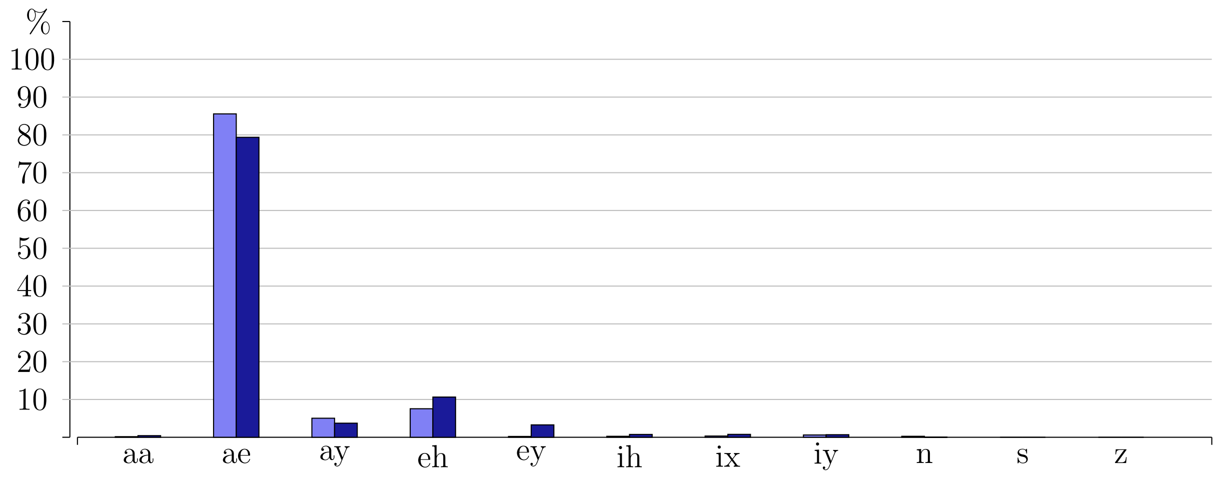
Phoneme ae

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	0.00	**93.12**	0.77	4.19	1.23	0.00	0.00	0.64	0.05	0.00	0.00
SSM	0.00	**89.05**	0.45	8.88	0.00	0.15	0.15	1.16	0.15	0.00	0.00
SMM	0.00	**87.82**	0.55	8.87	0.00	0.20	0.25	1.20	1.10	0.00	0.00
MMM	0.00	**92.60**	0.89	5.55	0.00	0.14	0.11	0.68	0.04	0.00	0.00
MME	0.05	**84.06**	6.42	8.22	0.00	0.40	0.50	0.15	0.20	0.00	0.00
MEE	0.20	**77.44**	8.28	12.62	0.00	0.40	1.01	0.00	0.05	0.00	0.00
EEE	0.87	**74.86**	17.94	4.51	0.32	0.50	0.32	0.46	0.23	0.00	0.00
avg.	0.16	**85.56**	5.04	7.55	0.22	0.26	0.33	0.61	0.26	0.00	0.00

Table B.2: Recognition rates for phones *ae*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	0.00	**85.40**	0.51	8.63	4.32	0.00	0.00	1.13	0.01	0.00	0.00
MMM-3	0.02	**86.02**	3.88	6.53	1.39	0.65	1.11	0.34	0.05	0.01	0.00
EEE-3	2.26	**66.65**	6.83	16.77	4.12	1.61	1.19	0.57	0.00	0.00	0.00
avg.	0.43	**79.36**	3.73	10.64	3.28	0.75	0.77	0.68	0.02	0.00	0.00

Table B.3: Recognition rates for phones *ae*, using reduced 3-state like *SME*-classification.



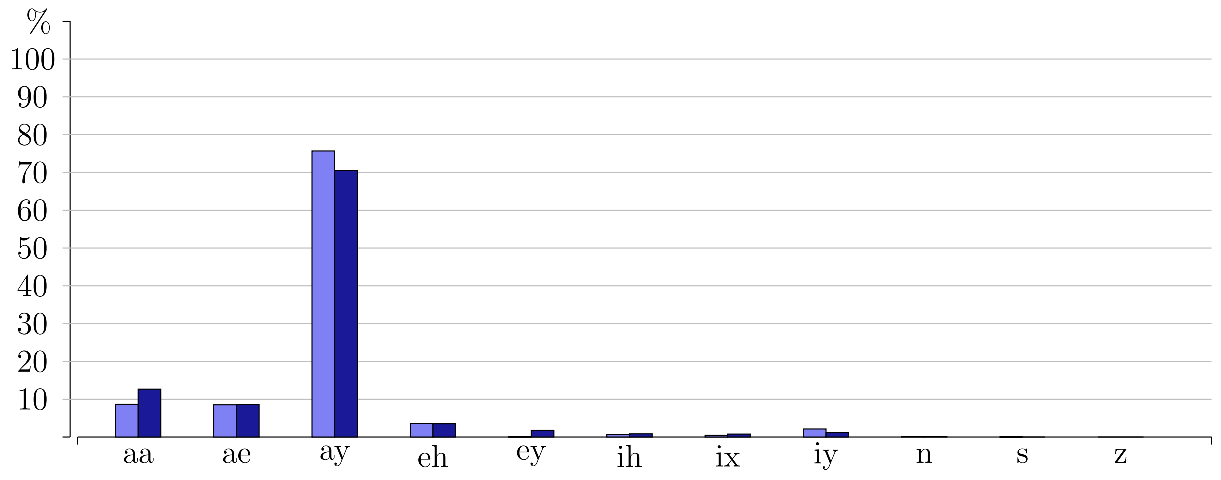
Phoneme ay

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	12.29	1.02	**86.04**	0.38	0.00	0.00	0.00	0.26	0.00	0.00	0.00
SSM	21.52	0.17	**70.40**	0.52	0.00	0.00	0.00	7.40	0.00	0.00	0.00
SMM	17.32	0.34	**74.61**	1.20	0.00	0.00	0.00	5.66	0.86	0.00	0.00
MMM	6.50	1.34	**90.51**	0.52	0.00	0.00	0.00	0.93	0.21	0.00	0.00
MME	1.37	14.41	**80.79**	2.74	0.00	0.00	0.34	0.17	0.17	0.00	0.00
MEE	0.00	22.89	**61.62**	13.43	0.00	0.17	1.89	0.00	0.00	0.00	0.00
EEE	1.66	19.46	**65.81**	6.53	0.26	4.61	1.02	0.51	0.00	0.13	0.00
avg.	8.67	8.52	**75.68**	3.62	0.04	0.68	0.47	2.13	0.18	0.02	0.00

Table B.4: Recognition rates for phones *ay*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	19.76	3.84	**71.81**	1.02	0.00	0.00	1.16	2.23	0.18	0.00	0.00
MMM-3	11.63	4.21	**78.97**	0.84	2.07	1.45	0.00	0.65	0.18	0.00	0.00
EEE-3	6.62	17.88	**60.87**	8.65	3.30	0.99	1.17	0.52	0.00	0.00	0.00
avg.	12.67	8.64	**70.55**	3.50	1.79	0.84	0.78	1.13	0.12	0.00	0.00

Table B.5: Recognition rates for phones *ay*, using reduced 3-state like *SME*-classification.



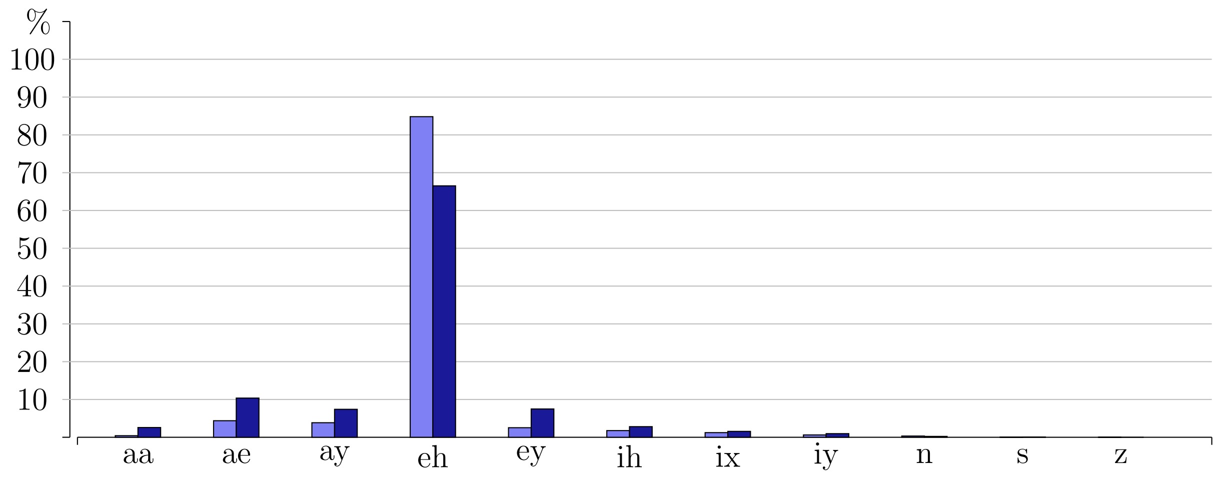
Phoneme eh

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	0.12	10.25	0.43	**73.05**	14.30	0.68	0.06	1.11	0.00	0.00	0.00
SSM	0.04	2.84	0.08	**92.53**	1.29	0.83	1.52	0.64	0.23	0.00	0.00
SMM	0.00	2.04	0.14	**93.75**	0.21	0.95	1.23	0.53	1.16	0.00	0.00
MMM	0.22	1.95	0.50	**94.96**	0.29	0.72	0.47	0.50	0.40	0.00	0.00
MME	0.00	1.75	1.19	**93.51**	0.00	1.44	1.65	0.04	0.42	0.00	0.00
MEE	0.00	3.87	4.89	**86.28**	0.00	2.08	2.88	0.00	0.00	0.00	0.00
EEE	2.58	7.98	19.71	**59.67**	1.60	5.71	0.80	1.53	0.12	0.25	0.06
avg.	0.42	4.38	3.85	**84.82**	2.53	1.77	1.23	0.62	0.33	0.04	0.01

Table B.6: Recognition rates for phones *eh*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	0.14	9.43	1.63	**70.71**	15.64	0.84	0.97	0.62	0.02	0.00	0.00
MMM-3	3.92	11.68	0.94	**74.23**	4.71	1.93	0.89	1.61	0.09	0.00	0.00
EEE-3	3.68	9.99	19.64	**54.63**	2.13	5.66	2.82	0.64	0.59	0.22	0.00
avg.	2.58	10.37	7.40	**66.52**	7.49	2.81	1.56	0.96	0.23	0.07	0.00

Table B.7: Recognition rates for phones *eh*, using reduced 3-state like *SME*-classification.



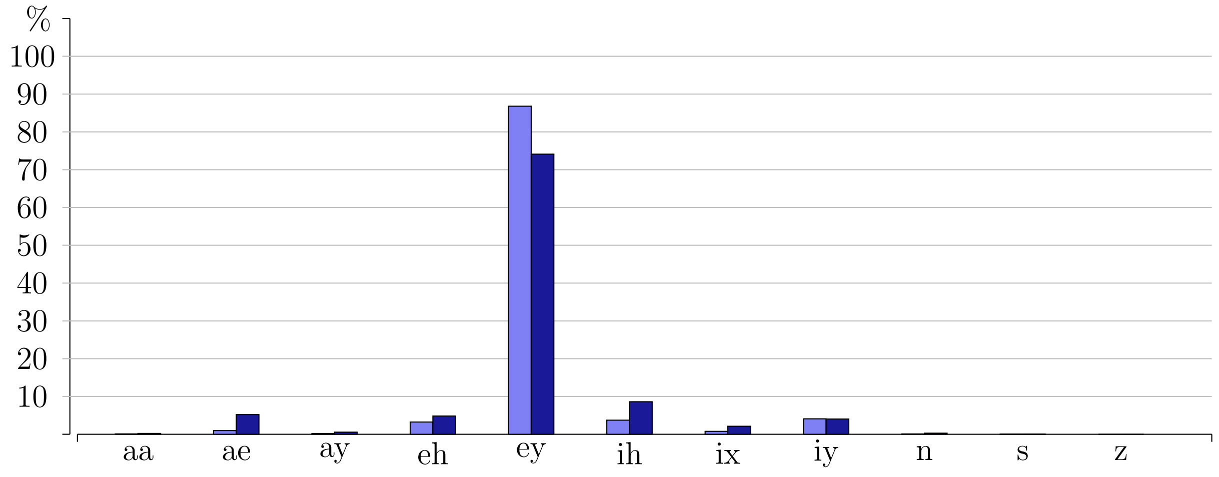
Phoneme ey

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	0.15	6.30	0.89	7.48	**83.22**	0.98	0.10	0.89	0.00	0.00	0.00
SSM	0.10	0.41	0.15	7.49	**83.54**	4.51	1.62	2.18	0.00	0.00	0.00
SMM	0.05	0.05	0.15	5.13	**84.30**	5.74	1.41	2.72	0.45	0.00	0.00
MMM	0.12	0.08	0.19	1.78	**90.85**	3.76	0.81	2.40	0.00	0.00	0.00
MME	0.05	0.00	0.00	0.70	**85.41**	6.14	0.86	6.79	0.05	0.00	0.00
MEE	0.10	0.05	0.10	0.10	**85.82**	3.24	0.46	10.03	0.10	0.00	0.00
EEE	0.10	0.00	0.00	0.00	**94.39**	1.72	0.15	3.59	0.00	0.05	0.00
avg.	0.10	0.98	0.21	3.24	**86.79**	3.73	0.77	4.09	0.09	0.01	0.00

Table B.8: Recognition rates for phones *ey*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	0.10	8.80	0.12	9.11	**76.22**	1.26	1.22	3.17	0.00	0.00	0.00
MMM-3	0.05	5.96	0.40	3.57	**79.33**	7.52	0.68	1.86	0.63	0.00	0.00
EEE-3	0.52	0.87	1.16	1.82	**66.78**	17.04	4.45	7.09	0.23	0.04	0.00
avg.	0.22	5.21	0.56	4.83	**74.11**	8.61	2.12	4.04	0.29	0.01	0.00

Table B.9: Recognition rates for phones *ey*, using reduced 3-state like *SME*-classification.



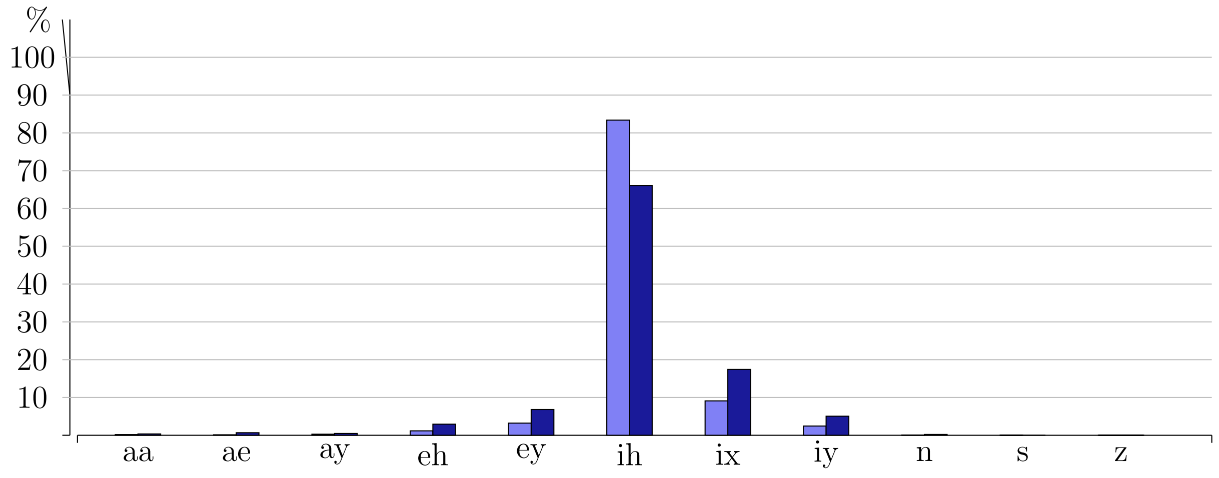
Phoneme ih

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	0.00	0.32	0.48	1.19	6.53	**80.97**	0.96	9.55	0.00	0.00	0.00
SSM	0.03	0.16	0.07	0.82	2.57	**84.30**	9.53	2.47	0.03	0.00	0.00
SMM	0.03	0.14	0.00	0.94	1.76	**83.46**	11.78	1.62	0.28	0.00	0.00
MMM	0.07	0.24	0.03	1.29	4.44	**82.89**	9.39	1.59	0.07	0.00	0.00
MME	0.03	0.06	0.08	1.76	1.18	**80.63**	16.07	0.17	0.03	0.00	0.00
MEE	0.03	0.00	0.20	1.42	0.43	**83.14**	14.65	0.10	0.03	0.00	0.00
EEE	1.11	0.00	1.11	0.80	5.57	**88.22**	1.35	1.59	0.00	0.16	0.08
avg.	0.19	0.13	0.28	1.17	3.21	**83.37**	9.10	2.44	0.06	0.02	0.01

Table B.10: Recognition rates for phones *ih*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	0.05	0.96	0.20	3.43	7.47	**70.03**	12.66	5.11	0.09	0.00	0.00
MMM-3	0.02	1.03	0.06	2.02	6.46	**65.14**	21.21	3.60	0.44	0.00	0.02
EEE-3	0.98	0.04	1.17	3.35	6.52	**63.03**	18.38	6.40	0.12	0.00	0.01
avg.	0.35	0.68	0.48	2.93	6.82	**66.07**	17.42	5.04	0.22	0.00	0.01

Table B.11: Recognition rates for phones *ih*, using reduced 3-state like *SME*-classification.



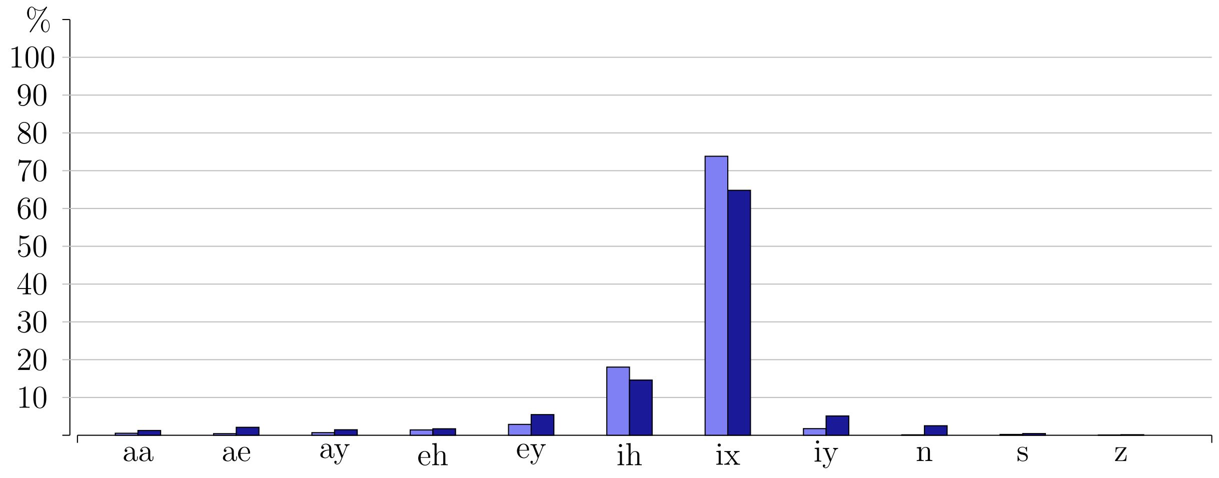
Phoneme ix

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	0.25	1.77	1.26	6.06	6.06	38.38	**38.13**	7.83	0.25	0.00	0.00
SSM	0.00	0.13	0.00	0.72	0.17	16.21	**81.33**	1.40	0.04	0.00	0.00
SMM	0.00	0.06	0.00	0.00	0.04	3.47	**95.77**	0.50	0.16	0.00	0.00
MMM	0.15	0.04	0.07	0.19	0.22	8.49	**89.36**	1.37	0.11	0.00	0.00
MME	0.00	0.00	0.00	0.04	0.00	2.93	**96.88**	0.12	0.02	0.00	0.02
MEE	0.13	0.25	0.30	2.16	0.00	22.70	**74.33**	0.08	0.00	0.00	0.04
EEE	3.28	0.76	3.28	0.76	13.64	34.09	**40.91**	1.01	0.25	1.52	0.51
avg.	0.54	0.43	0.70	1.42	2.88	18.04	**73.82**	1.76	0.12	0.22	0.08

Table B.12: Recognition rates for phones *ix*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	0.40	0.86	1.94	1.01	2.01	10.74	**78.75**	4.14	0.15	0.00	0.00
MMM-3	2.30	2.26	0.05	2.89	3.69	7.90	**66.20**	7.81	6.36	0.50	0.04
EEE-3	1.11	3.22	2.38	2.27	10.70	25.20	**49.47**	3.37	1.01	0.86	0.41
avg.	1.27	2.11	1.46	1.72	5.47	14.61	**64.81**	5.11	2.51	0.45	0.15

Table B.13: Recognition rates for phones *ix*, using reduced 3-state like *SME*-classification.



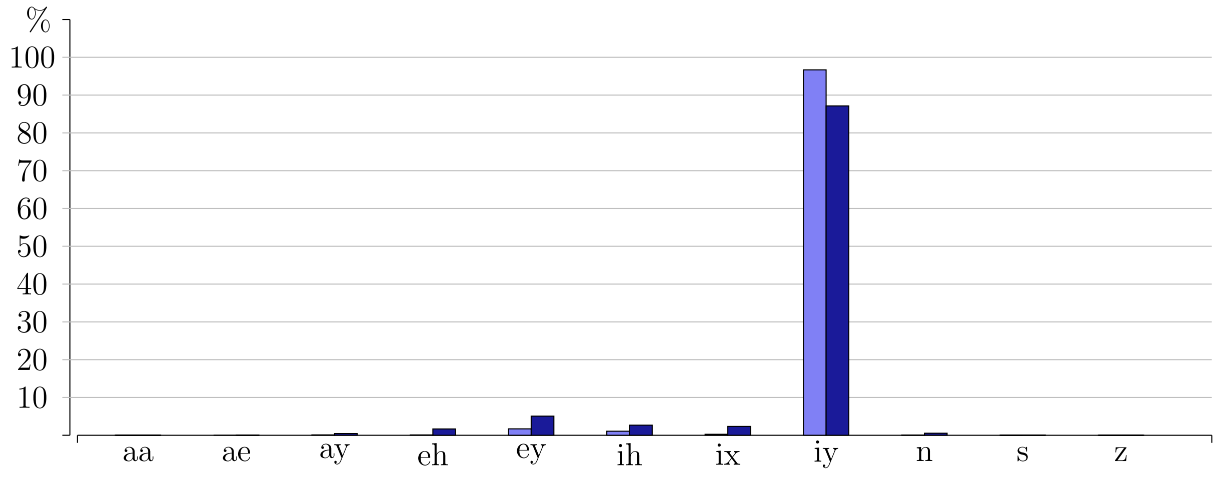
Phoneme iy

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	0.00	0.00	0.13	0.00	0.33	1.75	0.04	**97.75**	0.00	0.00	0.00
SSM	0.00	0.00	0.00	0.00	0.03	0.98	0.30	**98.64**	0.05	0.00	0.00
SMM	0.00	0.00	0.00	0.00	0.00	0.64	0.64	**98.68**	0.05	0.00	0.00
MMM	0.00	0.00	0.00	0.00	0.40	0.25	0.20	**99.08**	0.07	0.00	0.00
MME	0.00	0.00	0.00	0.02	1.62	0.44	0.34	**97.58**	0.00	0.00	0.00
MEE	0.03	0.00	0.00	0.00	2.83	0.68	0.24	**96.17**	0.05	0.00	0.00
EEE	0.08	0.00	0.58	0.63	6.68	2.76	0.04	**88.85**	0.21	0.13	0.04
avg.	0.02	0.00	0.10	0.09	1.70	1.07	0.26	**96.68**	0.06	0.02	0.01

Table B.14: Recognition rates for phones *iy*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	0.00	0.06	0.11	3.72	0.98	1.21	2.82	**90.95**	0.15	0.00	0.00
MMM-3	0.05	0.00	0.15	0.42	6.89	2.70	3.02	**86.48**	0.27	0.00	0.02
EEE-3	0.04	0.06	1.13	0.87	7.32	4.09	1.18	**84.00**	1.20	0.11	0.00
avg.	0.03	0.04	0.46	1.67	5.06	2.67	2.34	**87.14**	0.54	0.04	0.01

Table B.15: Recognition rates for phones *iy*, using reduced 3-state like *SME*-classification.



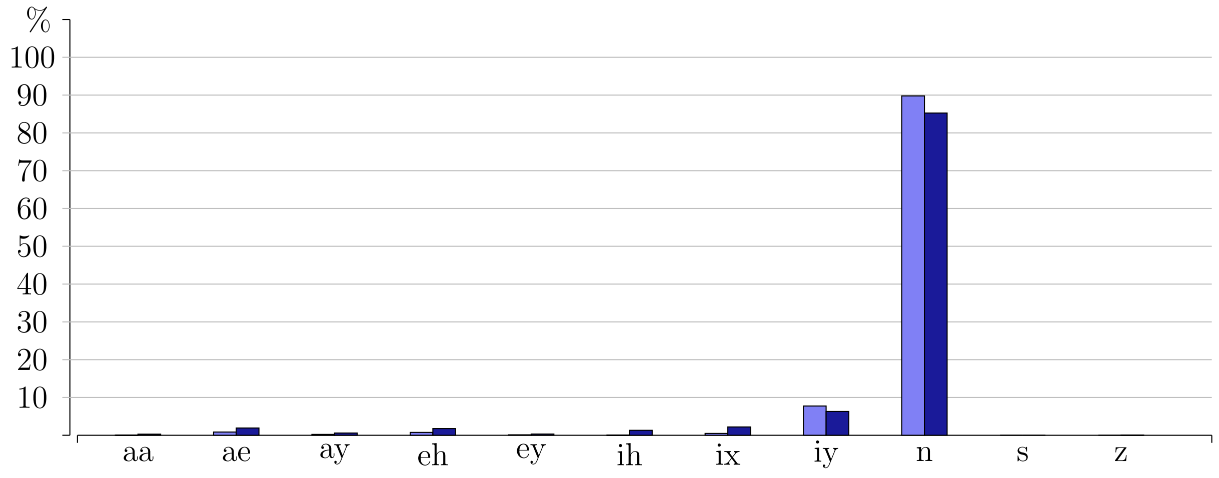
Phoneme n

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	0.00	0.58	0.00	2.34	0.00	0.00	0.00	26.32	**70.76**	0.00	0.00
SSM	0.00	0.00	0.00	0.48	0.00	0.00	0.36	6.96	**92.20**	0.00	0.00
SMM	0.00	0.00	0.00	0.25	0.00	0.06	0.44	0.00	**99.24**	0.00	0.00
MMM	0.11	0.00	0.00	0.32	0.00	0.11	0.54	4.09	**94.84**	0.00	0.00
MME	0.00	0.00	0.06	0.38	0.00	0.00	0.89	0.32	**98.35**	0.00	0.00
MEE	0.00	0.12	0.36	0.36	0.24	0.00	0.60	0.12	**98.20**	0.00	0.00
EEE	0.00	5.26	1.17	1.17	0.58	0.00	0.58	16.37	**74.85**	0.00	0.00
avg.	0.02	0.85	0.23	0.76	0.12	0.02	0.49	7.74	**89.78**	0.00	0.00

Table B.16: Recognition rates for phones *n*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	0.26	0.76	0.64	1.98	0.00	0.05	0.34	9.12	**86.85**	0.00	0.00
MMM-3	0.48	0.54	0.02	1.14	0.00	3.30	3.12	1.27	**90.02**	0.00	0.11
EEE-3	0.12	4.46	1.11	2.22	0.96	0.58	3.15	8.52	**78.88**	0.00	0.00
avg.	0.29	1.92	0.59	1.78	0.32	1.31	2.20	6.30	**85.25**	0.00	0.04

Table B.17: Recognition rates for phones *n*, using reduced 3-state like *SME*-classification.



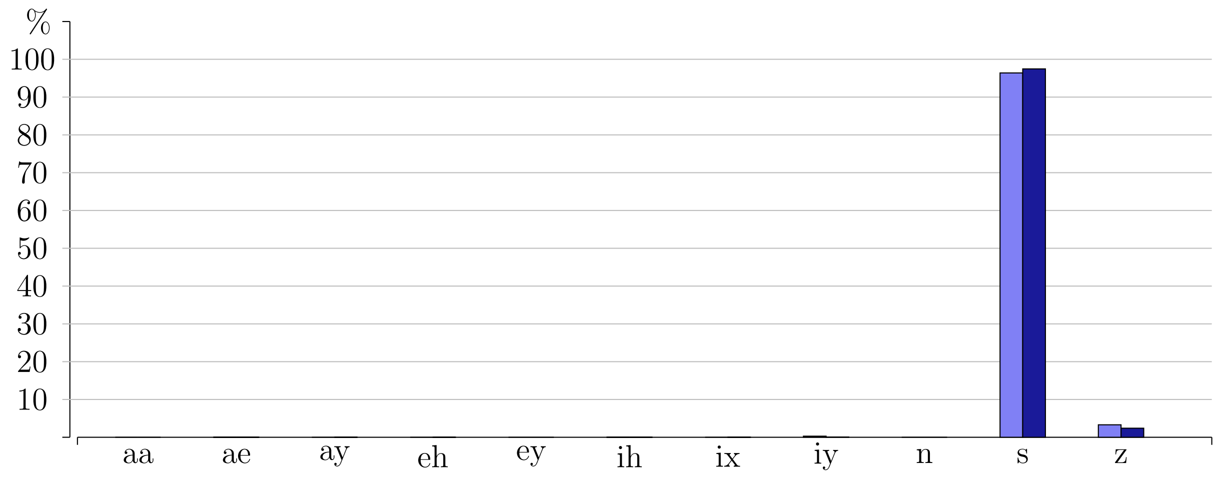
Phoneme s

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	0.00	0.04	0.02	0.00	0.00	0.00	0.00	0.07	0.00	**98.84**	1.03
SSM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.48	0.00	**98.34**	1.18
SMM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.00	**96.52**	2.56
MMM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.00	**96.95**	2.64
MME	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	**92.85**	7.02
MEE	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.02	**91.32**	8.64
EEE	0.00	0.00	0.00	0.02	0.00	0.07	0.00	0.02	0.00	**99.89**	0.00
avg.	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.29	0.00	**96.39**	3.30

Table B.18: Recognition rates for phones *s*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	0.00	0.02	0.02	0.00	0.00	0.00	0.03	0.14	0.00	**97.72**	2.07
MMM-3	0.00	0.01	0.00	0.00	0.00	0.00	0.04	0.04	0.00	**98.61**	1.30
EEE-3	0.00	0.00	0.00	0.04	0.00	0.04	0.00	0.03	0.01	**96.04**	3.84
avg.	0.00	0.01	0.01	0.01	0.00	0.01	0.02	0.07	0.00	**97.46**	2.40

Table B.19: Recognition rates for phones *s*, using reduced 3-state like *SME*-classification.



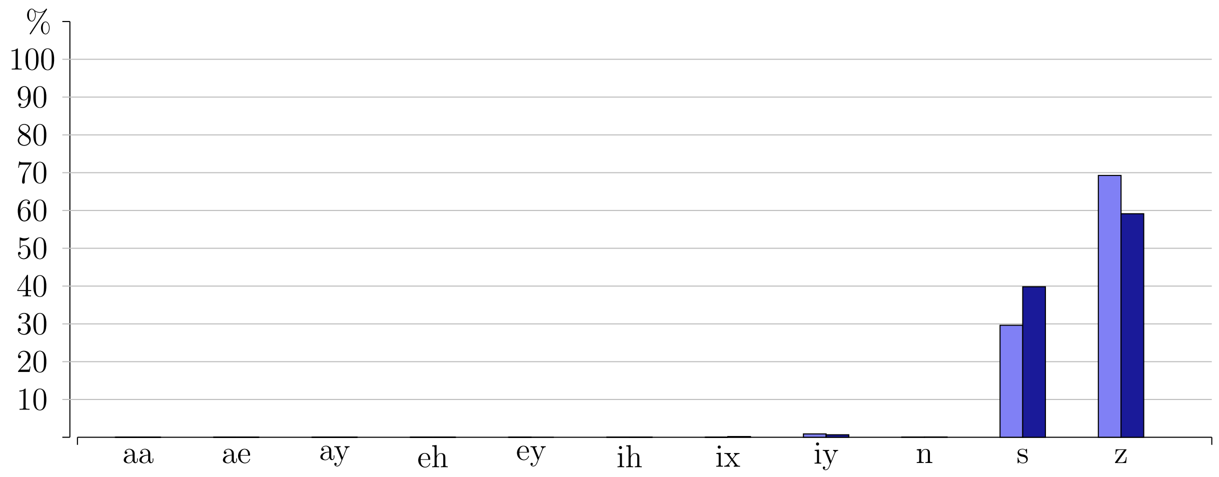
Phoneme z

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS	0.00	0.00	0.08	0.00	0.00	0.00	0.00	1.43	0.00	28.38	**70.11**
SSM	0.00	0.00	0.00	0.04	0.00	0.00	0.00	1.00	0.07	8.13	**90.76**
SMM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.85	0.25	12.49	**85.40**
MMM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.32	0.00	35.84	**62.84**
MME	0.00	0.03	0.00	0.00	0.00	0.00	0.03	0.35	0.03	23.73	**75.83**
MEE	0.00	0.07	0.00	0.04	0.00	0.04	0.04	0.00	0.11	0.07	**99.64**
EEE	0.08	0.08	0.00	0.00	0.15	0.30	0.00	0.30	0.00	98.79	**0.30**
avg.	0.01	0.03	0.01	0.01	0.02	0.05	0.01	0.89	0.07	29.63	**69.27**

Table B.20: Recognition rates for phones *z*, using 7-state *SME*-classification.

	aa	ae	ay	eh	ey	ih	ix	iy	n	s	z
SSS-3	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.11	0.00	23.67	**76.20**
MMM-3	0.00	0.04	0.03	0.02	0.00	0.07	0.43	1.23	0.00	29.84	**68.34**
EEE-3	0.04	0.00	0.07	0.00	0.00	0.00	0.15	0.64	0.24	65.97	**32.89**
avg.	0.01	0.01	0.03	0.01	0.00	0.04	0.19	0.66	0.08	39.82	**59.14**

Table B.21: Recognition rates for phones *z*, using reduced 3-state like *SME*-classification.



Appendix C – Averaged probability vectors vs. individual

The following tables show the improvements when classifying the, in the previous step, reduced states *SSS, MMM, EEE* individually versus the averaged, component wise vectors, where the latter where the ones used during the continuous speech recognition. For more details the reader can review section 3.6.2.

The first row of each table shows the number of samples of the averaged vectors classified correctly, the second one that numbers of all samples classified. The last row holds the number of samples classified correctly in addition when considering the three states without summing and averaging the components. The figures to the right of the table display the relative improvements of the respective phoneme.

Phonemes aa, ae

aa	SSS	MMM	EEE
merged	826	2168	1197
all	1393	2336	1393
additional	+47	+46	+29

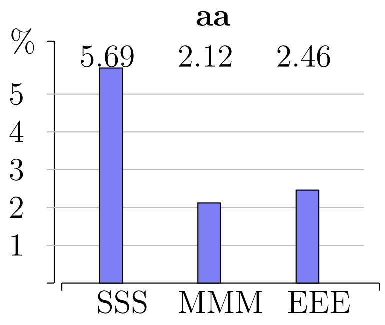


Table C.1: Confusion table of phoneme *aa*, using *SME*-SVM classification.

ae	SSS	MMM	EEE
merged	3567	5850	2784
all	4177	6801	4177
additional	+39	+77	+53

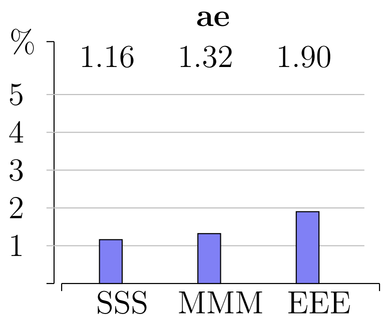


Table C.2: Confusion table of phoneme *ae*, using *SME*-SVM classification.

Phonemes ay, eh, ey

ay	SSS	MMM	EEE
merged	978	1686	829
all	1362	2135	1362
additional	+23	+30	+27

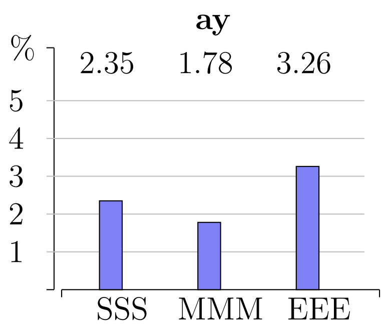


Table C.3: Confusion table of phoneme *ay*, using *SME*-SVM classification.

eh	SSS	MMM	EEE
merged	3017	6292	2331
all	4267	8476	4267
additional	+71	+119	+84

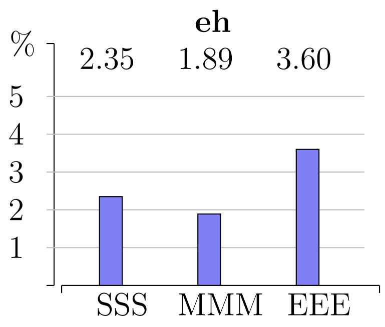


Table C.4: Confusion table of phoneme *eh*, using *SME*-SVM classification.

ey	SSS	MMM	EEE
merged	3054	5199	2676
all	4007	6554	4007
additional	+94	+143	+120

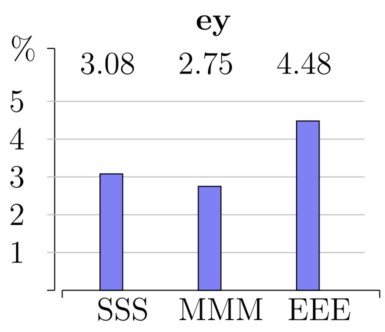


Table C.5: Confusion table of phoneme *ey*, using *SME*-SVM classification.

Phonemes ih, ix, iy

ih	SSS	MMM	EEE
merged	3002	6657	2702
all	4287	10219	4287
additional	+49	+141	+84

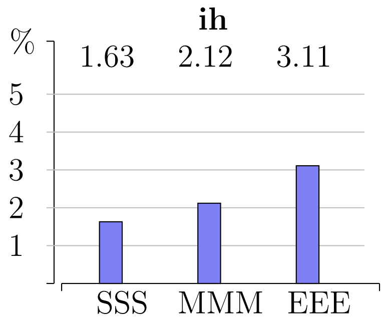


Table C.6: Confusion table of phoneme *ih*, using *SME*-SVM classification.

ix	SSS	MMM	EEE
merged	2168	8607	1362
all	2753	13002	2753
additional	+57	+102	+69

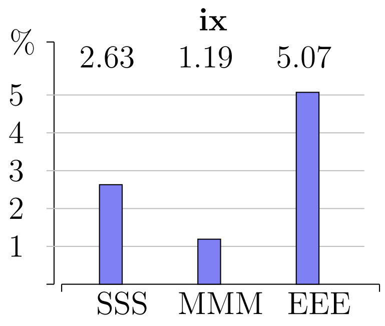


Table C.7: Confusion table of phoneme *ix*, using *SME*-SVM classification.

iy	SSS	MMM	EEE
merged	5526	10524	5104
all	6076	12169	6076
additional	+59	+131	+72

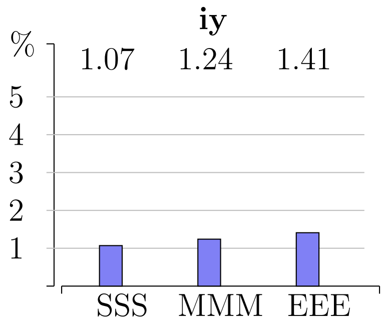


Table C.8: Confusion table of phoneme *iy*, using *SME*-SVM classification.

Phonemes n, s, z

n	SSS	MMM	EEE
merged	872	3680	792
all	1004	4088	1004
additional	+5	+9	+29

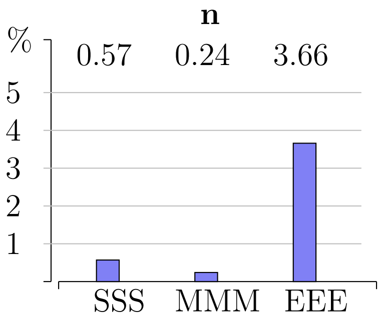


Table C.9: Confusion table of phoneme *n*, using *SME*-SVM classification.

s	SSS	MMM	EEE
merged	9509	16657	9346
all	9731	16892	9731
additional	+11	+4	+7

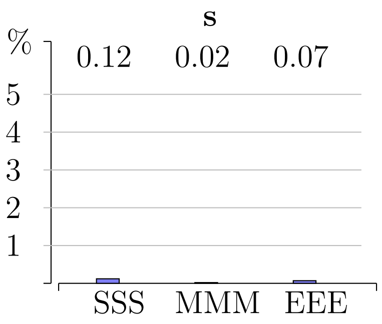


Table C.10: Confusion table of phoneme *s*, using *SME*-SVM classification.

z	SSS	MMM	EEE
merged	3137	6273	1354
all	4117	9179	4117
additional	+11	+40	+60

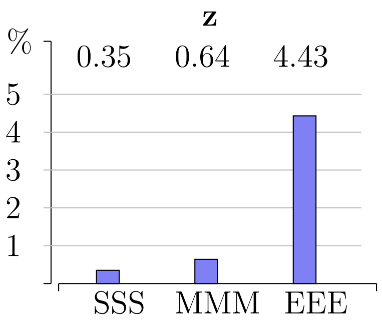


Table C.11: Confusion table of phoneme *z*, using *SME*-SVM classification.

Bibliography

[1] Baum, L.E. and Petrie, T., “Statistical Inference for Probabilistic Functions of Finite State Markov Chains,” *The Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[2] Baum, L. E. and Eagon, J.A., “An Inequality with Applications to Statistical Estimation for Probabilistic Functions of a Markov Process and to a Model for Ecology,” *Bulletin of the American Mathematical Society*, vol. 73, pp. 360–363, 1967.

[3] Baum, L.E. and Petrie, T. and Soules, G. and Weiss, N., “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains,” *Ann. Math. Statist.*, vol. 41, pp. 164–171, 1970.

[4] Baum, L.E., “An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes,” *Oved SHISHA, ed. Inequalities III: Proceedings of the Third Symposium on Inequalities*, pp. 1–8, 1972.

[5] Ellstrodt, Jürgen, *Maß- und Integrationstheorie*, Springer, 3., erweiterte Auflage edition, 2002.

[6] Aronszajn, N., “Theory of reproducing kernels,” *Transactions of the American Mathematical Society*, vol. 68, pp. 337–404, 1950.

[7] Agler, J. and McCarthy, K., *Pick Interpolation and Hilbert Function Spaces*, vol. 44 of *Graduate Studies in Mathematics*, American Mathematical Society, 2002.

[8] Alt, Hans Wilhelm, *Lineare Funktionalanalysis*, Springer, 5 edition, 2006.

[9] Shawe-Taylor, J. and Chrstianini, N., *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.

[10] Saitoh, S., *Integral transforms, reproducing kernels and their applications*, vol. 369 of *Pitman Research Notes in Mathematics Series*, Addison Wesley Longman Ltd., 1997.

[11] Schölkopf, B. and Smola, A., *Learning with Kernels*, MIT Press, Cambridge, 2002.

[12] Bishop, Christopher M., *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 1 edition, 2007.

[13] Werner, D., *Funktionalanalysis*, Springer Berlin, 6., korrigierte Auflage edition, 2007.

[14] Kreyszig, Erwin, *Introductory Functional Analysis with Application*, Wiley Classics Library. Wiley and Sons, 1978.

[15] Boid, S. and Vandenberghe, L., *Convex Optimization*, Cambridge University Press, 2008.

[16] Shigeo, A., *Support Vector Machines for Pattern Classification*, Advances in Pattern Recognition. Springer, 2005.

[17] Cortes, C. and Vapnik, V., "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[18] John C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," 1998.

[19] Edgar Osuna and Robert Freund and Federico Girosi, "An Improved Training Algorithm for Support Vector Machines," pp. 276–285, 1997.

[20] Bregman, L. M., "The Relaxation Method of Finding the Common Point of Convex Sets and Its Application to the Solution of Problems in Convex Programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, pp. 200–217, 1967.

[21] Yair Censor, "Row-Action Methods for Huge and Sparse Systems and Their Applications," *SIAM Review*, vol. 23, no. 4, pp. 444–466, 1981.

[22] S.S. Keerthi and S. K. Shevade and C. Bhattacharyya and K. R. K. Murthy, "Improvements to Platt's SMO Algorithm for SVM Classifier Design," 1999.

[23] Lanckriet, Gert R. G. and others, "Learning the Kernel Matrix with Semidefinite Programming," *Journal of Machine Learning*, vol. 5, pp. 27 – 72, 2004.

[24] Lanckriet, Gert R. G. and others, "A statistical framework for genomic data fusion," *Bioinformatics*, vol. 20, pp. 2626 – 2635, 2004.

[25] Bach, Friedrich R. and Lanckriet, Gert R. G. and Jordan, Michael I., "Multiple Kernel Learning, Conic Duality, and the SVM Algortihm," *Preprint, Proccedings of 21st International Conference on Machine Learning*, 2004.

[26] Diego, Isaac Martin de, "Combining Kernel Information for Support Vector Classification," *LNCS*, 2004.

[27] Hirokai, T. and others, "Simple but effective methods for combining kernels in computational biology," *RIVF*, pp. 71 – 78, 2008.

[28] Platt, John C., "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods," *Advances in Large Margin Classifiers*, 1999.

[29] Lin, Hsuan-Tien and Lin, Chih-Jen and Weng, Ruby C., "A Note Platt's paper on Probabilistic Outputs for Support Vector Machines," *Technical Report, Department of Computer Science*, 2003.

[30] Wu, Ting-Fan and Lin, Chih-Jen and Weng, Ruby C., "Probability Estimates for Multiclass Classification by Pairwise Coding," *Journal of Machine Learning Research*, vol. Vol. 5, 2004.

[31] Hastie, L. and Tibshirani, R., "Classification by pairwise coupling," *The Annals of Statistics*, vol. 26(1), pp. 451–471, 1998.

[32] Hunter, David R., "MM algorithms for generalized Bradley-Terry models," *The Annals of Statistics*, vol. 32, pp. 386–408, 2004.

[33] Levinson, Stephen E., *Mathematical models for Speech Technology*, John Wiley and Sons Ltd, 2005.

[34] Sondhi, M. M. Sondhi, "A Model for wave propagation in a lossy vocal tract," *J. Acoust.*, vol. 55, pp. 1070–1075, 1974.

[35] Portnoff, M. R., *A quasi-one-dimensional digital simulation for the time varying vocal tract*, Masters thesis. MIT, 1973.

[36] Ahmed, N. and Natarajan T. and Rao, K.R., "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. C-23, pp. 90–93, January 1974.

[37] Huang, Xuedong and Acero, Alex and Hon, Hsiao-Wuen, *Spoken Language Processing*, Prentice Hall, 2001.

[38] Jäger, H., "Discrete-Time, Discrete-Value Observable Operator Models: a Tutorial," *Technical Report, University of Freiburg, Germany*, 2000.

[39] Spanćzer, I., "Observable Operator Models," *Austrian Journal Of Statistics*, vol. 36, pp. 41–52, 2007.

[40] Phillip Koehn, *Statistical Machine Translation*, Cambridge Univserity Press, 2009.

[41] Bahl, L.r. and Jelinek, F. and Mercer, R.L., "A Maximum Likelihood Approach to Continuous Speech Recognition," *Readings in Speech Recognition*, 1990.

[42] Raj, B. and Seltzer, M. L. and Stern, R. M., "Reconstruction of missing features for robust speech recognition," *Speech Communications*, vol. 43, no. 4, pp. 275 – 296, 2004.

[43] Hamilton, J. D., *Time Series Analysis*, Princeton University Press, 1994.

[44] Kreiß, J. P. and Neuhaus, G., *Einführung in die Zeitreihenanalyse*, Springer, 2006.

[45] Meintrup, D. and Schäffler, S., *Stochastik – Theorie und Anwendungen*, Springer, 2004.

[46] mpiktas, "Does correlation assume stationarity of data," *http://stats.stackexchange.com/questions/7376/does-correlation-assume-stationarity-of-data.*

[47] Yuchun Tang and Yan-Qing Zhang and Nitesh V. Chawla and Sven Krasser, "SVMs Modeling for Highly Imbalanced Classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 39, no. 1, pp. 281–288, 2009.

[48] Lin, W.-J and Chane, J. J., "Class-imbalanced classifiers for high-dimensional data," *Briefings in Bioinformatics*, vol. 14, pp. 13–26, 2013/01.

[49] Japkowicz, N. and Stephen, S., "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[50] Sahare, M. and Gupta, H., "A Review of Multi-Class Classification for Imbalanced Data," *International Journal of Advanced Computer Research*, vol. 2, no. 3, pp. 160–164, 09/2912.

[51] T. Joachims, "Making large-Scale SVM Learning Practical," *Advances in Kernel Methods - Support Vector Learning*, 1999.

[52] Smith, N.D. and Gales, M. J. F., "Using SVMs and discriminative models for speech recognition," *IEEE International conference on accoustic speech and signal processing*, vol. 1, pp. I– 77 – 80, 2002.

[53] Lee, Kai-Fu and Hon, Hsiao-Wuen, "Speaker-Independent Phone Recognition Using Hidden Markov Models," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, 1989.

[54] Duin, Robert - P.W. and Loog, M. and Haeb-Umbach, R., "Multi-class Linear Feature Extraction by Nonlinear PCA," *In Proc. Int. Conference on Pattern Recognition*, 2000.

[55] Breiman, L., "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[56] Fan, G., "*Kernel*-Induced Classification Trees and Random Forests," *Technical Report*, 06/ 2009.

[57] Guangzhe, F. and Cao, J. and Wang, J., "Functional Data Classification for Temporal Gene Expression Data with Kernel-Induced Random Forests," *CIBC (Proceedings of the 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology)*, vol. 2010, pp. 1–5.

Index