Kernel Learning Approaches for Image Classification

Dissertation

zur Erlangung des akademischen Grades

Doctor rerum naturalium (Dr.rer.nat)

an der Naturwissenschaftlich-Technischen Fakultät I
der Universität des Saarlandes, Saarbrücken

vorgelegt von Dipl.-Inform.

Peter Vincent Gehler

10. Juni 2009

Tag des Kolloquiums:
20. November 2009

Dekan der Naturwissenschaftlich-Technischen Fakultät I:
Prof. Joachim Weickert

Prüfungsausschuss:
Prof. Joachim Weickert (Vorsitzender)
Prof. Matthias Hein (Berichterstattender)
Prof. Bernhard Schölkopf (Berichterstattender)
Dr. Matthias Seeger

Weiterer Berichterstattender:
Prof. Luc VanGool

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Tübingen, 10. Juni 2009

(Peter Vincent Gehler)

Summary

This thesis extends the use of kernel learning techniques to specific problems of image classification. Kernel learning is a paradigm in the field of machine learning that generalizes the use of inner products to compute similarities between arbitrary objects. In image classification one aims to separate images based on their visual content.

We address two important problems that arise in this context: learning with weak label information and combination of heterogeneous data sources. The contributions we report on are not unique to image classification, and apply to a more general class of problems.

We study the problem of learning with label ambiguity in the multiple instance learning framework. We discuss several different image classification scenarios that arise in this context and argue that the standard multiple instance learning requires a more detailed disambiguation. Finally we review kernel learning approaches proposed for this problem and derive a more efficient algorithm to solve them.

The multiple kernel learning framework is an approach to automatically select kernel parameters. We extend it to its infinite limit and present an algorithm to solve the resulting problem. This result is then applied in two directions. We show how to learn kernels that adapt to the special structure of images. Finally we compare different ways of combining image features for object classification and present significant improvements compared to previous methods.

Zusammenfassung

In dieser Dissertation verwenden wir Kernmethoden für spezielle Probleme der Bildklassifikation. Kernmethoden generalisieren die Verwendung von inneren Produkten zu Distanzen zwischen allgemeinen Objekten. Das Problem der Bildklassifikation ist es, Bilder anhand des visuellen Inhaltes zu unterscheiden.

Wir beschäftigen uns mit zwei wichtigen Aspekten, die in diesem Problem auftreten: lernen mit mehrdeutiger Annotation und die Kombination von verschiedenartigen Datenquellen. Unsere Ansätze sind nicht auf die Bildklassifikation beschränkt und für einen grösseren Problemkreis verwendbar.

Mehrdeutige Annotationen sind ein inhärentes Problem der Bildklassifikation. Wir diskutieren verschiedene Instanzen und schlagen eine neue Unterteilung in mehrere Szenarien vor. Danach stellen wir Kernmethoden für dieses Problem vor und entwickeln einen Algorithmus, der diese effizient löst.

Mit der Methode der Kernkombination werden Kernfunktionen anhand von Daten automatisch bestimmt. Wir generalisieren diesen Ansatz indem wir den Suchraum auf kontinuierlich parametrisierte Kernklassen ausgedehnen. Diese Methode wird in zwei verschiedenen Anwendungen eingesetzt. Wir betrachten spezifische Kerne für Bilddaten und lernen diese anhand von Beispielen. Schließlich vergleichen wir verschiedene Verfahren der Merkmalskombination und zeigen signifikante Verbesserungen im Bereich der Objekterkennung gegenüber bestehenden Methoden.

Contents

Acknowledgement

I was very fortunate to pursue my PhD studies in the Empirical Inference group at the Max Planck Institute for Biological Cybernetics in Tübingen. This place provides an excellent environment for curious driven research and great possibilities for a graduate student to learn and grow. I had a great time.

First I would like to thank Bernhard Schölkopf for providing such an challenging and thriving lab, and granting the freedom and time for pursuing my research. In the same respect I thank Sebastian Stark and Sabrina Nielebock for their excellent job during the past years.

Thanks a lot to Matthias Hein, Bernhard Schölkopf, and Luc Van Gool for agreeing to review this thesis.

Especially I thank Matthias Franz who took care of me in the very early days of my PhD, and Max Welling for his support not only during the two research stays at the University of Irvine. I am also deeply indebted to Andrew Blake, Olivier Chapelle, Geoff Hinton, and Carsten Rother who gave advise and inspiration during the last years. Furthermore I thank my colleagues and friends Jan Eichhorn, Sebastian Gerwinn, Wolf Kienzle, Malte Kuss, Christoph Lampert, Sebastian Nowozin, Cheng Soon Ong, Gunnar Rätsch, Florian Steinke and Christian Walder. It has been a pleasure and a lot of fun to work together at the MPI. I would also like to thank the co-authors of my publications Alex Holub, Tom Minka and Toby Sharp. I am very happy and grateful that I got the chance to work together and learn from so many inspiring researchers.

Special thanks to Sebastian Nowozin and Christoph Lampert for proof reading a draft of this thesis. The presentation benefited a lot from their valuable comments.

During my PhD studies I was supported financially through several programs which I gratefully acknowledge. Most of the time I was member in the EC project CLASS, IST 027978 and of the European Network of Excellence, Pascal and Pascal-2. The NSF Grant No. 0447903 supported two research stays at the UC Irvine and Microsoft Research an internship in Cambridge, UK. The Max Planck Society funded my PhD and several visits of different conferences.

My deepest thanks go to my family for their unconditional support: my parents, brother and grandparents. Together with my friends they always reminded me of the life beyond work. More than to anyone else I owe to the love, inspiration and encouragement of my wife Barbara. You are the most important part of my life.

1. Introduction

In this thesis we are concerned with the problem of visual image classification by means of kernel learning algorithms. The goal of image classification is to separate images according to their visual content into two or more disjoint classes.

We will focus on the problems of visual object and scene classification as specific instances from the general class of image classification problems. Our analysis and methods are however not unique to these instances and may be used in a more general context beyond image related problems. The topic of this thesis lies in the intersection of computer vision and machine learning research and we will report on advances in both directions.

Consider the images shown in Figure 1.1. All the images depict different physical objects from the real world. Nevertheless some of the images can be grouped together, since they show objects of the same type or in other words from the same category. For example there are several images showing different types of mountain bikes and all those images are considered to be of the same class. Our goal is to build a system that analyzes the visual content of an image and identifies the corresponding object category. The more general task of image classification is the problem of finding a mapping from images to a set of classes, not necessarily object categories.

The problem of single object classification we discuss in this thesis is a fundamental problem that many higher level tasks depend on. From a broader perspective we are interested in more complex tasks like scene understanding or object localization. Photographs of real world scenes usually contain many different objects from various categories like those shown in Figure 1.2. [1] To retrieve all possible object labels within such images generalizes the problem definition to the multi-label case. Image understanding even requires reasoning about the relations of the objects within the scene. Both scenarios crucially depend on the ability to be able to identify single objects and should benefit from advancements in this area.

The main challenge of object classification is the high variability of object appearances. Appearance changes due to transformations such as pose variations, change of lightning conditions and scale changes all affect the representation of the image. While this poses a problem already for recognizing single

[1] Taken from `labelme.csail.mit.edu`

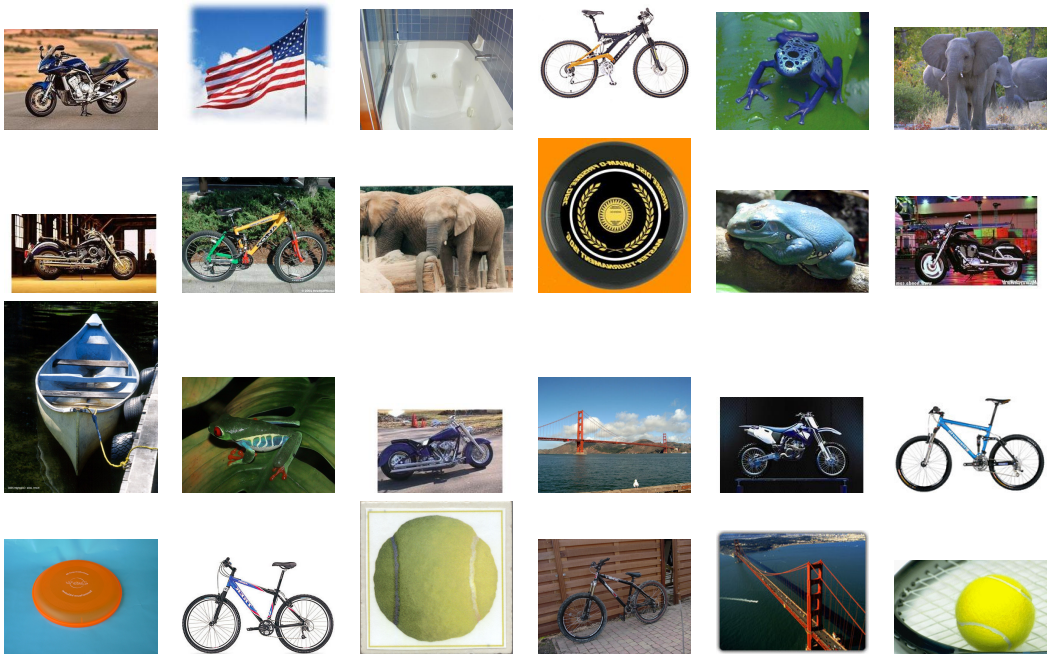


Figure 1.1.: Several example images from the Caltech-256 dataset. The images come with labels, the names of the categories that are shown are mountain bike, elephant, frog, American flag, canoe, bath tub, frisbee, motorbike, and tennis ball. In each single image there are parts that might be considered instances of another category, like the tires of the bikes. The object category we are interested is the category corresponding to the shown object as indicated by the category name.

specific instances, it is even harder when it comes to recognising entire classes of object categories. In the latter case appearance variations also stems from the variability of the instances within the classes. This intra-class variability depends on the specific object class. For example images of tennis balls as shown in Figure 1.1 show fewer appearance variations than images of non-rigid deformable objects such as the animal images of elephants and frogs. Some other classes like the mountain bikes and motor bikes are visually similar and also share appearance variations.

This problem is typically addressed by means of image descriptors that are designed to be invariant to the most dominant appearance changes. A simple representation as a collection of raw pixel values on the other hand is highly appearance dependent, and thus not robust enough for this task. There are

(a) (b)

Figure 1.2.: Two example images from the LabelMe dataset. Shown are two pairs of images, left the original image and right color coded an user annotation of objects within the images. Since there are many different object instances present in the scene this is a multi-label problem.

two criteria for the design of image descriptors. First, they should be general enough to capture similarities between instances of the same category. Second, they should be robust to aforementioned appearance changes of a single object. In the last years, several different image descriptors have been developed that are especially tailored to the task of object classification.

Image descriptors are typically limited in the sense that they make use of specific single statistics of the images like color, texture or shape information. This raises the problem that any single descriptor may not be sufficient to characterize entire object categories. In that case it needs the combination of several descriptors in order to derive discriminative decision rules. Different categories will require different combination of descriptors to distinguish them from others. Due to the large number of possible classes we need automatic ways of finding such combinations. Manually tuning every single one of them is infeasible.

As argued above, we are confronted with a scenario where manually designing classification functions is not possible. This is because the underlying principles of the data are far too complex to be codified. Therefore we adopt a data centric view of the problem and apply techniques to *learn* classification functions from an annotated set of example images.

A learning procedure typically involves the following two steps. First, a model class of admissible functions has to be chosen. In a second step, a function from this model class is searched by minimization of an appropriate error criterion. The latter step involves the incorporation of prior knowledge about suitable functions, as well as making use of available observations in order to estimate the quality of a function, for example its generalization abilities.

The utilization of learning techniques has been a recent trend in this subfield of computer vision. As a consequence, a variety of databases of labeled training examples have been collected for this particular purpose. These dataset

collections grew from a couple of images and a few object categories to impressive sizes. At the time of writing, the largest freely available image database for object classification[2] contains more than 3 million images of about 5000 different categories that are organized in a semantic hierarchy. Since even for this database the set of categories is by no means comprehensive, image collections are likely to grow in size in the future. This in turn demands for large scale machine learning techniques that are able to cope with this quantity of data.

A problem with attaching single class labels to images, as it has been done for those of Figure 1.1 is the following. Almost all images contain some background clutter. Although such context information might be helpful to classify the images in a specific scenario, an ideal classification system should be invariant to the environment an object is presented in. Since the label information does not convey information about the spatial extend of the object, it remains ambiguous. All that can be deduced is that some part of the image shows the object of interest. An approach to resolve this problem would be to manually provide segmentation masks as those in Figure 1.2. Since this requires substantially more effort in terms of manual annotation, we are interested in methods that are able to learn with weaker forms of annotations, like object class information.

We approach the image classification problem by extending kernel learning algorithms. As we will see, techniques from this class offer a versatile tool to approach all of the aforementioned problems: learning with ambiguity (chapter 3), learning discriminative image representations (chapter 6) and combination of different image descriptors (chapter 7).

1.1. Contribution of this Thesis and Outline

In the following we will outline the thesis and give an overview of the content of the individual chapters and their contributions. Most of the work described here has been published, and we include the citations to the corresponding publications.

This thesis is divided into two parts. The first part is on the topic of machine learning and more specifically about kernel learning algorithms. In particular it includes an introduction, two contributions in this field and an empirical evaluation of approaches that learn the kernel function from data. The second part of the thesis is about the specific problem of image classification. Here we demonstrate how the developments proposed in the first part can be put into practice for this particular scenario. Hence the first part is more focused

[2] `www.image-net.org`

on algorithmic and theoretical questions, while the second part is concerned with practical problems, and as such, contains a larger experimental part.

Each chapter ends with a conclusion, summarizing the presented material. The chapters relate to each other in the following way. All chapters build upon the material presented in the introductory chapter 2. The techniques introduced in chapter 4 are a pre-requisite for the chapters 5 to 7 that themselves may be read separately.

1.1.1. Part I: Kernel Classifiers

The machine learning part begins with an introductory chapter about kernel learning techniques in chapter 2. We introduce and review the main concepts of kernel learning and in particular the framework of regularized risk minimization. The most prominent algorithms for tasks like classification, regression and structured output prediction are summarized in a dedicated section. A version of this chapter appeared as a book chapter [Geh09b].

We then continue with chapter 3 with the topic of learning with in the presence of label ambiguity and weak label information. This is cast in the multiple instance learning (MIL) framework. We discuss several scenarios of label ambiguity and show how they relate to MIL. In particular we argue that multiple instance learning as it has been used throughout the literature, is itself not sufficiently well defined. We propose a disambiguation into several scenarios and discuss them in detail. Finally we turn our attention to kernel learning algorithm for this problem. These result in integer programming problems for which heuristic algorithms have been proposed. We derive a homotopy method to solve for better local minima of these non-convex problems and show empirically that those solutions translate to improved performance on standard datasets. The algorithmic part of this chapter appeared in [Geh07].

The topic of chapter 4 is on learning the kernel function itself from training data. We propose with *infinite kernel learning* (IKL) a framework, in which a kernel is estimated from a continuously parameterized set of kernel functions. This generalizes the concept of previous work on multiple kernel learning (MKL). MKL was limited to combine finitely many kernels which we will show how to overcome this restriction. We then derive and efficient algorithm to solve the resulting optimization problem for both the finite and infinite case.

In chapter 5 we present an empirical evaluation of kernel combination algorithms and compare them to the performance of a single kernel support vector machine. This is the most complete evaluation of the different techniques that has been published so far. While most recent developments in this field concentrated on algorithmic issues we will show that in many cases little or no performance gains can be expected. We present empirical evidence

that in cases where kernel combination yields a superior performance our new approach IKL significantly outperforms the finite version MKL. The content of the two chapters 4 and 5 have been published in [Geh08a, Geh08b].

1.1.2. Part II: Image Classification

Chapter 6 demonstrates how the approaches of learning the kernel function from data from chapter 4 can be used to improve the performances of image classification systems. Typical image classification systems involve a long pipeline of pre-processing steps and we show how this can be shorten in a principled way. Pre-processing choices such as the spatial layout or codebook representations of images are usually hand-tuned. We shift these design choices into the kernel function and thus make them available to the optimization problem. This work also appeared in [Geh09a].

In the last chapter 7 of this thesis we discuss the problem of combining several image descriptors for the purpose of image classification. This part has been published in [**?**]. As discussed earlier, an image can be represented through a variety of appearance descriptors. Combinations of several descriptors might be necessary to discriminate images from different classes. First we reuse the earlier approaches of kernel learning and augment them with a number of baseline comparisons. Inspired by the kernel combination approach we derive a boosting based algorithm for this problem. We present empirical results on challenging prominent benchmark datasets. Our approach consistently outperforms all previous attempts that have been proposed so far. Theses state-of-the-art results supports our claim that feature combination is an effective technique for the task of image classification and machine learning provides efficient tools to accomplish this task.

1.1.3. Related Contributions

Not all of the work that has been published during my time as a PhD student is contained in this thesis. In this section we briefly present the contributions that are relevant to the main theme of the thesis. These address research questions originating from the field of image classification, and computer vision or machine learning in general.

The Rate Adapting Poisson Model

The dominant paradigm for modelling histogram data is the extraction of latent semantic structure, often referred to as topics. Histogram representations of data arises naturally in the fields of text processing and image classification. We will see concrete examples of this type of representation for image data in section 6.1. Latent variable models determine a mapping from count

data to a compressed latent representation. This typically lower dimensional representation can subsequently be used for retrieval and classification tasks.

In [Geh06b] we propose a probabilistic undirected graphical model that yields a *distributed* latent representation. The undirected semantics of this model has interesting consequences. Most importantly, the latent variables are *conditionally independent* given the data, and vice versa. This is in stark contrast to the *marginal independence* of the latent variables in directed models. The implication is that the mapping from input space to latent space is given by a single matrix multiplication, possibly followed by a componentwise nonlinearity. This comes with a problem at learning time, due to the presence of an intractable normalization constant that depends on the parameters. We apply contrastive divergence to learn the parameters and show empirical results on problems from text and image retrieval.

Color Constancy

Color constancy is the tendency to perceive surface color consistently, despite variations in ambient illumination [Jam61]. Most generally, illumination variations occur both within scenes, and from scene to scene, and theories such as the "Retinex" [Lan71] have been devised to explain color constancy under such conditions.

In [Geh08c] we extend a Bayesian model for color constancy. We make the common assumption that illumination within a given scene is approximately uniform. In particular we report on three new results. Firstly we introduce a new dataset consisting of 568 images, captured using a high-quality digital SLR camera in RAW format, free of color correction. Secondly we examine the fusion algorithm [Gij07] which has appeared to give performance better than any individual greyworld algorithm. Using the new database we show that the fused algorithm is after all not significantly better than the best greyworld algorithm. Thirdly, we revisit the Bayesian approach of [Ros04] using the new data-set. Where in [Ros04] training was based on illumination labels that were only estimated, the new dataset provides more accurate illumination labels. This makes it possible to learn more precise priors for illumination and reflectance. Tests show that this leads to illuminant estimation for which the improvement in accuracy is statistically significant, at least for outdoor images. The newly trained Bayesian algorithm is shown also to perform significantly better than the greyworld algorithms, even when the greyworld algorithms are enhanced by inclusion of an illumination prior.

Gaussian Processes for Wiener Series Estimation

In its classical formulation, the estimation of the Wiener series assumes noise-free measurements of the system outputs during system identification. This assumption was also adopted in the kernel-based implicit estimation procedure

described in [**?**]. For real, noise-contaminated data, the estimated Wiener series will model both signal and noise of the training data which results in reduced prediction performance on independent test sets. But even in the ideal case of noise-free signals, Volterra and Wiener models typically show a reduced generalisation performance as compared to other types of nonlinear models. Roughly speaking, this is due to the "explosiveness" of polynomial models: for test inputs far outside the range of the training data, polynomial models usually assign outputs with extremely high absolute values. This results in a high sensitivity against outliers in the test data which in turn leads to a reduced prediction performance.

In [Fra06, Geh06a] we address the generalisation problems of Volterra and Wiener system models using two closely related frameworks: regularisation and Gaussian processes. Regularisation is the standard approach in machine learning to address the generalisation problem. The basic idea is to restrict the possible solutions in a suitable manner that reflects the prior knowledge of the experimenter about the different characteristics of the true signal and the corrupting noise. We show how this train of thought can be adapted to restrict our solutions not only to be smooth, but also to remain non-explosive in the input domains that are relevant to the problem at hand.

Part I.

Kernel Classifiers

2. An Introduction to Kernel Learning Algorithms

Algorithms that use positive definite kernels have considerably influenced the field of machine learning, pattern recognition and related fields over the last decade. For example the prominent Support Vector Machine has been applied with much success to a variety of tasks and nowadays belongs to the standard toolbox of every practitioner. Besides their empirical success kernel methods have a solid theoretical foundation and have also been studied in the mathematics and statistics communities. In this chapter we will review the basic mathematical concepts of kernel learning and introduce some prominent algorithms. In contrast to the majority of work on Support Vector Learning we will avoid using duality theory and will instead use the regularized risk formulation as the underlying basis for the derivation.

For almost all problems in image processing, including visual object classification and detection, one is confronted with a variety of different problems, some of which have already been described in the introduction. Image data is inherently high dimensional, which is the main reason one usually resorts to different feature representations rather than raw pixels for its representation. Furthermore, in almost all cases, there is only a limited amount of labeled training data available. Algorithms have to account for noise in the observations in a robust way. Image data may stem from different sources, for example hyperspectral image data taken from satellites, which requires methods of combining such information.

Kernel algorithms are suited to tackle such problems. With the design of a kernel function it is possible to combine different feature entities, as well as different feature dimensions and to account for high dimensional data. With the framework of regularized risk minimization, kernel methods are efficient even with small amount of training data and offer ways to incorporate unlabeled examples through semi-supervised models. Such algorithms have already proven to be a valuable tool for image processing applications such as image coding, image de-noising, image segmentation and image classification [Geh09a, Kim05, CV07, Gri07, Bla08]. In the later chapters of this thesis we will highlight the use of Support Vector Machines for the task of image classification.

In this chapter we will introduce the basic concepts of kernel learning and

some of its applications. This will lay the foundation for chapter 3 and 4 in which we will describe ways of how to learn a kernel function from data in a principled way and learn in the presence of ambiguity. The selection of topics for this chapter is by no means comprehensive and is intended to provide a self-contained exposition of the material. The main goal is to make the later chapters accessible but to also review some of the widely used kernel learning algorithms. More detailed introductions in the field which also covers the statistical background can be found in the articles [Sch02, Sha04, Hof08].

This chapter is divided into three sections. We start in Section 2.1 with a basic introduction to the notion of kernels and introduce the reproducing kernel Hilbert space. In Section 2.2 we state the representer theorem which serves as the foundation of all of the formulations which are presented in the following Section. Section 2.3 contains the most prominent kernel learning algorithms, and includes a paragraph about their applications to various problems.

2.1. Kernels

2.1.1. Measuring Similarity with Kernels

Suppose we are given empirical data

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathcal{X} \times \mathcal{Y}. \tag{2.1}$$

We will call $\mathbf{x}_i$ the *inputs* which are taken from the nonempty set $\mathcal{X}$ and $y_i \in \mathcal{Y}$ the *targets*. The problem of learning is to use this data in order to make statements about previously unseen elements $\mathbf{x} \in \mathcal{X}$. For example in binary classification where the training data stems from two classes with labels $\mathcal{Y} = \{-1, +1\}$ one aims to construct a function $f : \mathcal{X} \to \mathcal{Y}$ which assigns a class label to each element of $\mathcal{X}$. The function in which one is interested should not be arbitrary but one which *generalizes* well. Loosely speaking, this entails making few errors on unseen data from the same problem. In the classification example this corresponds to making as few mistakes as possible when inferring the class labels.

In order to enable generalization, we need to exploit the structure of the training examples and in order to impose a structure we will need to define a similarity between pairs of data points. The most general setting would be to define such a similarity between pairs of inputs $(\mathbf{x}, y)$ including the targets. For now we will restrict ourselves and only define similarities between inputs $\mathbf{x} \in \mathcal{X}$ and will refer to Section 2.3.4 for a generalization.

There is no other assumption about $\mathcal{X}$ other than it being a set and in particular nothing has been said about its inputs being similar to each other. Therefore we will first map the data into a space where we have a notion of

similarity, namely a dot product space $\mathcal{H}$, using a mapping

$$\phi : \mathcal{X} \to \mathcal{H}, \quad \mathbf{x} \mapsto \phi(\mathbf{x}). \tag{2.2}$$

The similarity between the elements in $\mathcal{H}$ can now be measured using its associated inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. For convenience we introduce the following function which does exactly that

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \ (\mathbf{x}, \mathbf{x}') \mapsto k(\mathbf{x}, \mathbf{x}'), \tag{2.3}$$

which we require to satisfy for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}} . \tag{2.4}$$

This function is called a *kernel*. The mapping ϕ is referred to as its *feature map* and the space $\mathcal{H}$ as its *feature space*. Usually the kernel itself is parameterized by some set of variables $\theta \in \Theta$. In these cases we will make the dependency of the kernel on the parameters explicit, using the notation $k(\cdot, \cdot; \theta)$. To simplify the notation we will omit the parameters θ whenever possible, i.e. in those cases where we refer to a general kernel function.

Although the construction of a kernel seems inconspicuous we will see that it has far reaching consequences. Sometimes we will drop the subscript specifying the origin of the dot product in these cases where it should be clear from the context.

2.1.2. Positive Definite Kernels

The construction of the similarity measure as the dot product in some space $\mathcal{H}$ is rather general. Different measures of similarity can be obtained by varying the feature map ϕ. A particular simple case is when $\mathcal{X}$ itself is a dot product space in which case one may choose ϕ to be the identity.

We will now show that the class of kernels that can be written in the form of (2.4) coincide with the class of positive definite kernels. This yields a very comfortable situation due to the following observation: algorithms which operate on the data only in terms of a dot product can be used with any positive definite kernel by simply replacing $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$ with kernel evaluations $k(\mathbf{x}, \mathbf{x}')$. This is a technique also known as the *kernel trick* [Sch98]. Another direct consequence is that for a positive definite kernel one does not need to know the explicit form of the feature map since it is implicitly defined through the kernel. We will even encounter examples where $\mathcal{H}$ is infinite dimensional and thus replacing the dot product with the kernel function evaluation is crucial in order to be able to numerically evaluate the dot product at all.

We need some definitions before we can state the equivalence between $k(\mathbf{x}, \mathbf{x}')$ and $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$.

Definition 2.1.1 (Gram matrix)**.** *Given a kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$. We call the $n \times n$ matrix $\mathbf{K}$ with entries*

$$\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) \tag{2.5}$$

the Gram matrix *or the* kernel matrix *of k with respect to $\mathbf{x}_1, \ldots, \mathbf{x}_n$.*

Definition 2.1.2 (Positive definite matrix)**.** *A real symmetric $n \times n$ matrix $\mathbf{K}$ is called* positive definite *if for all $c_1, \ldots, c_n \in \mathbb{R}$*

$$\sum_{i,j=1}^{n} c_i c_j \mathbf{K}_{ij} \geq 0. \tag{2.6}$$

If equality in (2.6) *only occurs for $c_1 = \ldots = c_n = 0$ then the matrix is called* strictly positive definite

A positive definite kernel is one which always produces a positive definite Gram matrix for elements in $\mathcal{X}$. More precisely:

Definition 2.1.3 (Positive definite kernel)**.** *If for all $n \in \mathbb{N}$ and for all $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ the Gram matrix $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is positive definite, then we call the kernel a* positive definite kernel*. Furthermore if for all $n \in \mathbb{N}$ and distinct $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ the kernel k gives rise to a strictly positive definite Gram matrix we will call it a* strictly positive definite kernel*.*

Now we are ready to state one of the most important observations for kernel methods. To this end we need to introduce the concept of a Hilbert space. Recall that a Hilbert space $\mathcal{H}$ is a real (or complex valued) inner product space which is completed by the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Some simple examples of Hilbert spaces are $\mathbb{R}^d$ and $\mathbb{C}^d$.

Proposition 2.1.4. *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel* if and only if *there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ we have $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$.*

Proof. "$\Leftarrow$" Assume the kernel can be written in the form (2.4). It being positive definite is a simple consequence of the bilinearity of the dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$

$$\sum_{i,j=1}^{n} c_i c_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^{n} c_i \phi(\mathbf{x}_i), \sum_{j=1}^{n} c_j \phi(\mathbf{x}_j) \right\rangle_{\mathcal{H}} = \left\| \sum_{i=1}^{n} c_i \phi(\mathbf{x}_i) \right\|_{\mathcal{H}}^2 \geq 0. \tag{2.7}$$

"$\Rightarrow$" In the next section 2.1.3 we will present how to construct, given a positive definite kernel, and a Hilbert space along with a feature map ϕ with the desired properties. This will conclude the proof. □

Due to this equivalence we will sometimes refer to a positive definite kernel simply as a *kernel*. Although kernels compute dot products in some space $\mathcal{H}$, they should not be mistaken to be themselves dot products in the input space. For example, they are not in general bilinear. However they share important properties such as the Cauchy-Schwarz inequality.

Proposition 2.1.5 (Cauchy-Schwarz)**.** *If k is a positive definite kernel, and $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$, then*

$$k(\mathbf{x}_1, \mathbf{x}_2)^2 \leq k(\mathbf{x}_1, \mathbf{x}_1) k(\mathbf{x}_2, \mathbf{x}_2). \tag{2.8}$$

Proof. Since k is positive definite, so is the 2×2 Gram matrix $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Therefore the eigenvalues of $\mathbf{K}$ are non-negative as is its determinant. Writing out the determinant completes the proof

$$0 \leq \det(\mathbf{K}) = k(\mathbf{x}_1, \mathbf{x}_1) k(\mathbf{x}_2, \mathbf{x}_2) - k(\mathbf{x}_1, \mathbf{x}_2)^2. \tag{2.9}$$

□

2.1.3. Constructing the Reproducing Kernel Hilbert Space

Using positive definite kernels as building blocks we will now consider functions which are linear combinations of kernel evaluations. This leads to the concept of a reproducing kernel Hilbert space (RKHS). In the following we will present a construction scheme for a fixed kernel k which will also conclude the proof of Proposition 2.1.4. The main idea is to construct a Hilbert space whose elements are functions. For a given kernel k we define the following set

$$F = \left\{ f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, \mathbf{x}_i);\ n \in \mathbb{N}, \alpha_i \in \mathbb{R}, \mathbf{x}_i \in \mathcal{X} \right\} \subseteq \mathbb{R}^{\mathcal{X}}, \tag{2.10}$$

where the elements $k(\cdot, \mathbf{x}) : \mathcal{X} \to \mathbb{R}$ are functions, and $k(\cdot, \mathbf{x})$ itself is an element in F. It is easy to see that this set can be turned into a vector space if we endow it with the two operations of addition $(f + g)(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})$ as well as multiplication with a scalar $(\lambda f)(\mathbf{x}) = \lambda f(\mathbf{x}), \lambda \in \mathbb{R}$. Now we define an inner product between two elements of this space

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, \mathbf{x}_i) \qquad \text{and} \qquad g(\cdot) = \sum_{j=1}^{n'} \beta_j k(\cdot, \mathbf{x}'_j) \tag{2.11}$$

with $n, n' \in \mathbb{N}, \alpha_i, \beta_j \in \mathbb{R}, \mathbf{x}_i, \mathbf{x}'_j \in \mathcal{X}$, as

$$\langle f, g \rangle_F := \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{x}'_j). \tag{2.12}$$

This is a well defined construction, i.e. it does not depend on the choice of the expansion coefficients of f or g. To see this, note that we can also write making use of the symmetry of the kernel

$$\sum_{j=1}^{n'} \beta_j f(\mathbf{x}'_j) = \langle f, g \rangle_F = \sum_{i=1}^{n} \alpha_i g(\mathbf{x}_i) \tag{2.13}$$

where the left term does not depend on the expansion of f and the right term does not depend on the expansion of g. From Eq. (2.13) we also see that $\langle \cdot, \cdot \rangle_F$ is bilinear. Furthermore it is symmetric and positive definite which follows from the positive definiteness of the kernel k, since

$$\langle f, f \rangle_F = \sum_{i,j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \tag{2.14}$$

implies that for any functions $f_1, \ldots, f_p \in F$ and any coefficients $c_1, \ldots, c_p \in \mathbb{R}$ we have

$$\sum_{i,j=1}^{p} c_i c_j \langle f_i, f_j \rangle_F = \left\langle \sum_{i=1}^{p} c_i f_i, \sum_{j=1}^{p} c_j f_j \right\rangle_F \geq 0. \tag{2.15}$$

From the last equation we see that $\langle \cdot, \cdot \rangle_F$ is a positive definite kernel defined on a vector space of functions.

If we choose $g(\cdot) = k(\cdot, \mathbf{x})$ it follows from the definition of the inner product that

$$\langle f, k(\cdot, \mathbf{x}) \rangle_F = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) = f(\mathbf{x}), \quad \forall x \in \mathcal{X} \tag{2.16}$$

and in particular

$$\langle k(\cdot, \mathbf{x}), k(\cdot, \mathbf{x}') \rangle_F = k(\mathbf{x}, \mathbf{x}'). \tag{2.17}$$

This property is known as the *reproducing property* of the kernel. A function f can thus be represented as a linear function defined by an inner product in the vector space F.

We still need to show definiteness of the inner product. Applying (2.16) and the Cauchy-Schwarz inequality we obtain

$$|f(\mathbf{x})|^2 = |\langle k(\cdot, \mathbf{x}), f \rangle_F|^2 \leq k(\mathbf{x}, \mathbf{x}) \cdot \langle f, f \rangle_F \tag{2.18}$$

which proves that $\langle f, f \rangle_F = 0 \Leftrightarrow f = 0$. The space we have constructed can be completed by adding all limit points of sequences that are convergent in the norm $\|f\|_F = \sqrt{\langle f, f \rangle}$ which yields the Hilbert space $\mathcal{H}$, see e.g. [Aro50] for details.

Due to the property (2.16) this space is called a *reproducing kernel Hilbert space* (RKHS) for k. The RKHS uniquely determines k and vice versa. This is the statement of the following theorem.

Theorem 2.1.6 (Moore-Aronszajn, see [Aro50]). *To every positive definite kernel k there exists a unique reproducing kernel Hilbert space $\mathcal{H}$ whose kernel is k and vice versa.*

Proof. Since we have already proved existence, uniqueness remains to be shown. Let $\mathcal{H}'$ be another Hilbert space for which k is the reproducing kernel. Then for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$

$$\langle k(\cdot, \mathbf{x}), k(\cdot, \mathbf{x}') \rangle_{\mathcal{H}} = k(\mathbf{x}, \mathbf{x}') = \langle k(\cdot, \mathbf{x}), k(\cdot, \mathbf{x}') \rangle_{\mathcal{H}'}. \tag{2.19}$$

Due to the linearity of the dot product and the uniqueness of the completion it must hold $\mathcal{H} = \mathcal{H}'$. Now assume $k, k' \in \mathcal{H}, K \neq K'$ are two different reproducing kernels of $\mathcal{H}$. Then there exists a $\mathbf{x} \in \mathcal{X}$ for which

$$\begin{aligned} 0 &< \|k(\cdot, \mathbf{x}) - k'(\cdot, \mathbf{x})\|_{\mathcal{H}}^2 = \langle k(\cdot, \mathbf{x}) - k'(\cdot, \mathbf{x}), k(\cdot, \mathbf{x}) - k'(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} & (2.20) \\ &= \langle k(\cdot, \mathbf{x}), k(\cdot, \mathbf{x}) - k'(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} - \langle k'(\cdot, \mathbf{x}), k(\cdot, \mathbf{x}) - k'(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} = 0, & (2.21) \end{aligned}$$

which is a contradiction. □

We have constructed a Hilbert space which can act as the feature space of our kernel. The corresponding feature map for this space is the so-called *Aronszajn map*

$$\phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}, \mathbf{x} \mapsto k(\cdot, \mathbf{x}). \tag{2.22}$$

For this ϕ it is easy to see that the kernel k is indeed of the form (2.4). We want to point out that although there exists a unique RKHS to each kernel k, it might well be that there are other feature maps that result in a inner product with the same value. If the two feature maps ϕ_1, ϕ_2 map into the feature spaces $\mathcal{H}_1$, resp. $\mathcal{H}_2$ then it might be the case that

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi_1(\mathbf{x}), \phi_1(\mathbf{x}') \rangle_{\mathcal{H}_1} = \langle \phi_2(\mathbf{x}), \phi_2(\mathbf{x}') \rangle_{\mathcal{H}_2}, \tag{2.23}$$

for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, and that does not necessarily imply that $\phi_1 = \phi_2$. For our purposes however we can consider the two spaces identical, as we are only interested in the kernel evaluation as the dot product and this remains identical.

2.1.4. Operations in RKHS

We will turn our attention to some basic operations in the reproducing kernel Hilbert space and show how they can be computed in terms of kernel function evaluations. Although the space $\mathcal{H}$ can be very high dimensional or even infinite dimensional, in some cases basic operations can still be computed. Essentially this is the case whenever we can express its elements in terms of kernel evaluations.

Translation

A translation in feature space can be written as the modified feature map $\tilde{\phi}(\mathbf{x}) = \phi(\mathbf{x}) + \Gamma$ with $\Gamma \in \mathcal{H}$. We expand the dot product for $\langle\tilde{\phi}(\mathbf{x}), \tilde{\phi}(\mathbf{x}')\rangle_{\mathcal{H}}$ to write

$$\langle\phi(\mathbf{x}) + \Gamma, \phi(\mathbf{x}') + \Gamma\rangle = \langle\phi(\mathbf{x}), \phi(\mathbf{x}')\rangle + \langle\phi(\mathbf{x}), \Gamma\rangle + \langle\Gamma, \phi(\mathbf{x}')\rangle + \langle\Gamma, \Gamma\rangle. \quad (2.24)$$

In general only the first term can be evaluated via the kernel function. But if we restrict Γ to have an expansion $\Gamma = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i)$, i.e. to lie in the span of the functions $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n) \in \mathcal{H}$, we can calculate the translated dot product

$$\langle\tilde{\phi}(\mathbf{x}), \tilde{\phi}(\mathbf{x}')\rangle = k(\mathbf{x}, \mathbf{x}') + \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + \sum_{i=1}^{n} \alpha_i k(\mathbf{x}', \mathbf{x}_i) + \sum_{i,j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j). \quad (2.25)$$

Centering

The translation operations allows us to *center* data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ *in the feature space*. The mean of the data is $\phi_\mu = \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{x}_i)$ and thus fulfills the requirements for the translation element Γ. Applying (2.25) with $\Gamma = -\phi_\mu$ thus yields a feature map for which

$$0 = \frac{1}{n} \sum_{i=1}^{n} \tilde{\phi}(\mathbf{x}_i). \quad (2.26)$$

Computing Distances

With the kernel corresponding to a dot product in a Hilbert Space $\mathcal{H}$ and thus inducing a norm it is natural to ask if one can also compute the distances of the images of the elements in $\mathcal{X}$. Such a distance can be evaluated entirely in terms of kernel evaluations as is evident from

$$d(\mathbf{x}_1, \mathbf{x}_2) = \|\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)\|_{\mathcal{H}} = \sqrt{k(\mathbf{x}_1, \mathbf{x}_1) + k(\mathbf{x}_2, \mathbf{x}_2) - 2k(\mathbf{x}_1, \mathbf{x}_2)} \quad (2.27)$$

for $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ and ϕ being the feature map for k. This is a very elegant way to measure the dissimilarity between arbitrary objects, for example between two graphs or two sentences.

Subspace Projections

For two points $\Psi, \Gamma \in \mathcal{H}$ the projection of Ψ onto the subspace spanned by Γ is

$$\Psi' = \frac{\langle\Gamma, \Psi\rangle_{\mathcal{H}}}{\|\Gamma\|_{\mathcal{H}}^2}\Gamma. \quad (2.28)$$

If we have a kernel expansion of the points Ψ, Γ, then we can compute the projection Ψ' and also express it in terms of kernel evaluations.

2.1.5. Kernel Construction

The following proposition states some operations which preserve the positive definiteness of kernels. These operations can be used to create new, possibly complicated, kernels from existing ones.

Proposition 2.1.7. *Let $K_1, K_2, \ldots$ be arbitrary positive definite kernels on $\mathcal{X} \times \mathcal{X}$, where $\mathcal{X}$ is a nonempty set. Then*

(i) $\alpha_1 k_1 + \alpha_2 k_2$ *is positive definite for* $\alpha_1, \alpha_2 \geq 0$.

(ii) If $k(\mathbf{x}, \mathbf{x}') := \lim_{n\to\infty} K_n(\mathbf{x}, \mathbf{x}')$ *exists for all* $\mathbf{x}, \mathbf{x}'$*, then* kK *is positive definite.*

(iii) The pointwise product $k_1 k_2$ *is positive definite.*

(iv) Assume for $i = 1, 2, k_i$ *is a positive definite kernel on* $\mathcal{X}_i \times \mathcal{X}_i$*, where* $\mathcal{X}_i$ *is nonempty. The tensor product* $k_1 \otimes k_2$ *and the direct sum* $k_1 \oplus k_2$ *are positive definite kernels on* $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$.

(v) The function $k(\mathbf{x}, \mathbf{x}') := f(\mathbf{x}) f(\mathbf{x}')$ *is a positive definite kernel for any function* $f : \mathcal{X} \to \mathbb{R}$.

Proof. For proofs see [Ber84]. □

The first two statements of the last proposition state that the set of positive definite kernels is a closed convex cone. Loosely speaking the operations of (i)-(iv) are the only simple operations which preserve positive definiteness. Without stating the result we mention that it is possible to fully characterize the class of functions that preserve positive definiteness [Fit95, Hof08].

2.1.6. Examples of Kernels

With the closure properties of the last result we finally turn our view to some concrete examples of kernel functions. We concentrate on the most prominent ones and also introduce two kernels which have been used for several image processing tasks. For a more general overview including examples of other data structures such as graphs, trees, strings, etc. we refer to [Hof08, Sch02, Bak07, Sha04]. At this point we will present some general kernels which operate on real vector spaces. In the following chapters on image classification we will introduce some more kernel functions which are especially suited for problems in which the input elements themselves are either images or some kind of image features.

Linear Kernel

The most simple kernel is arguably the ordinary dot product in $\mathbb{R}^d$. Functions which are built upon this kernel are of the form

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle = \langle \mathbf{x}, \sum_{i=1}^{n} \alpha_i \mathbf{x}_i \rangle = \langle \mathbf{x}, \mathbf{w} \rangle, \tag{2.29}$$

where we defined $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$. Thus the RKHS of a linear kernel on $\mathbb{R}^d \times \mathbb{R}^d$ can be identified with the span of all hyperplanes in $\mathbb{R}^d$.

Polynomial Kernel

The *homogeneous polynomial* kernel $k(\mathbf{x}, \mathbf{x}'; p) = \langle \mathbf{x}, \mathbf{x}' \rangle^p$ is positive definite for all $p \in \mathbb{N}$ as a direct consequence from proposition 2.1.7 (iii). This again is an example where the input space $\mathcal{X}$ itself is a dot-product space. One can take the corresponding feature space: it is of finite dimension and consists of all monomials of degree p in the input coordinates. Another prominent polynomial kernel is the *inhomogeneous polynomial kernel* which computes the inner product of all monomials *up to* degree p: $k(\mathbf{x}, \mathbf{x}'; \{c, p\}) = (\langle \mathbf{x}, \mathbf{x}' \rangle + c)^p$

Gaussian Kernel

From the Taylor series expansion of the exponential function $e^z = \sum_{i=0}^{\infty} \frac{1}{i!} z^i$ and proposition 2.1.7-(ii) we see that

$$k(\mathbf{x}, \mathbf{x}'; \gamma) = e^{\gamma \langle \mathbf{x}, \mathbf{x}' \rangle} \tag{2.30}$$

is a positive definite kernel for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d, \gamma \in \mathbb{R}, \gamma \geq 0$. It follows immediately that the Gaussian function

$$e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2} = e^{-\gamma \langle \mathbf{x}, \mathbf{x} \rangle} e^{2\gamma \langle \mathbf{x}, \mathbf{x}' \rangle} e^{-\gamma \langle \mathbf{x}', \mathbf{x}' \rangle} \tag{2.31}$$

is a valid positive definite kernel. Furthermore we observe from the Taylor series expansion that it is a polynomial kernel with infinite degree. The corresponding Hilbert space is infinite dimensional and in fact it corresponds to a mapping into $\mathcal{C}^\infty$, the space of smooth functions.

χ^2 Histogram Kernel

A common type of instances occurring in text and image processing problems are aggregated representations such as histograms. We will discuss them in greater detail in the context of image classification in Section 6.1.1. A prominent kernel for comparing two histograms is the χ^2 *kernel*. For two histograms $\mathbf{h}, \mathbf{h}'$ of size d the χ^2-distance is defined as

$$\chi^2(\mathbf{h}, \mathbf{h}') = \sum_{i=1}^{d} \frac{(\mathbf{h}_i - \mathbf{h}'_i)^2}{\mathbf{h}_i + \mathbf{h}'_i}, \tag{2.32}$$

where we used the convention $x/0 := 0$. It was shown in [Hei05] that the χ^2 kernel of the form

$$k(\mathbf{h}, \mathbf{h}'; \gamma) = \exp\left(-\gamma \chi^2(\mathbf{h}, \mathbf{h}')\right) \tag{2.33}$$

is a positive definite kernel.

Set Kernels

So far we presented kernels for rather simple input spaces only. However the power of kernel methods also stems from the fact that we can measure similarity between possibly complex objects. Assume an input space being the power set of some other set, e.g. $\mathcal{X} = 2^{\mathcal{K}}$ for some finite dictionary $\mathcal{K}$. A similarity on this space could for example be defined by just counting the number of equal elements in the sets. The corresponding kernel for $X, X' \in \mathcal{X}$ reads

$$k(X, X') = \sum_{\mathbf{x} \in X} \sum_{\mathbf{x}' \in X} \delta(\mathbf{x} = \mathbf{x}'), \tag{2.34}$$

where we denote with

$$\delta(x, x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise} \end{cases} \tag{2.35}$$

the indicator function. We used upper case letters for the elements of $\mathcal{X}$ to make explicit that those elements are in fact sets. This kernel is widely used in text classification and is also referred to as the *sparse vector kernel*, see for example [Joa98]. Consider a text being represented as the bag of words which appear in the text. This kernel measures similarity between two texts by just counting the number of common words.

A more general kernel can be defined on the same input set with the use of a base kernel k_0. It sums up the similarities between all elements in the two sets

$$k(X, X'; k_0) = \sum_{\mathbf{x} \in X} \sum_{\mathbf{x}' \in X} k_0(\mathbf{x}, \mathbf{x}'). \tag{2.36}$$

This is a kernel if and only if k_0 itself is a kernel [Hau99].

2.2. The Representer Theorem

In the last section we introduced the RKHS $\mathcal{H}$ associated with a kernel k serving as its inner product. Functions $f \in \mathcal{H}$ can be represented as linear combinations of kernel expansions but have a possibly infinite number of expansion coefficients. The representer theorem [Kim71, Cox90] states that the solutions of a large class of optimization problems are expressible by only a finite number of kernel functions. We present a slightly more general version of the theorem [Sch01].

Theorem 2.2.1 (Representer Theorem)**.** *Let $\Omega : [0, \infty) \to \mathbb{R}$ be a strictly monotonic increasing function and $L : (\mathcal{X} \times \mathbb{R}^2)^n \to \mathbb{R} \cup \{\infty\}$ be an arbitrary loss function. Furthermore let $\mathcal{H}$ be a RKHS with reproducing kernel k. Each minimizer $f \in \mathcal{H}$ of the regularized functional*

$$\Omega\left(\|f\|_{\mathcal{H}}^2\right) + L\left((\mathbf{x}_1, y_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, y_n, f(\mathbf{x}_n))\right) \tag{2.37}$$

admits a representation of the form

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i), \tag{2.38}$$

with $\alpha_i \in \mathbb{R}$.

Proof. We decompose an element $f \in \mathcal{H}$ into the part $f^{\|}$ which is inside the span of kernel functions $k(\cdot, \mathbf{x}_1), \ldots, k(\cdot, \mathbf{x}_n)$ and its orthogonal complement $f^{\perp}$ and show that the latter is always zero. We write

$$f(\mathbf{x}) = f^{\|}(\mathbf{x}) + f^{\perp}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + f^{\perp}(\mathbf{x}) \tag{2.39}$$

with $\alpha_i \in \mathbb{R}$ and $\langle f^{\perp}, k(\cdot, \mathbf{x}_i)\rangle_{\mathcal{H}} = 0, \forall i \in \{1, \ldots, n\}$. Using the reproducing property of $\mathcal{H}$ we can write

$$f(\mathbf{x}) = \langle f, k(\cdot, \mathbf{x})\rangle_{\mathcal{H}} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + \langle f^{\perp}, k(\cdot, \mathbf{x})\rangle_{\mathcal{H}} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i). \tag{2.40}$$

We thus see that the term $f^{\perp}$ is irrelevant for the value of L in (2.37). Making use of the fact that Ω is monotonically decreasing we get the following inequality

$$\Omega\left(\|f\|_{\mathcal{H}}^2\right) = \Omega\left(\left\|\sum_{i=1}^{n} \alpha_i k(\cdot, \mathbf{x}_i)\right\|_{\mathcal{H}}^2 + \|f^{\perp}\|_{\mathcal{H}}^2\right) \geq \Omega\left(\left\|\sum_{i=1}^{n} \alpha_i k(\cdot, \mathbf{x}_i)\right\|_{\mathcal{H}}^2\right). \tag{2.41}$$

Thus for any fixed $\alpha_i \in \mathbb{R}$ the function Ω in (2.37) is minimized for $f^{\perp} = 0$. □

This theorem has widespread implications. It tells us that whenever we can formulate an objective function in form of (2.37) we can rest assured that the solution can be expressed in terms of finitely many kernel evaluations. Even if the function space is infinite dimensional, we only need to search for the n expansion coefficients.

Monotonicity of Ω does not ensure a unique minimizer of (2.37), we have to require convexity to prevent the possibility of several solutions. Indeed many algorithms make use convex loss functions. The strictness of the monotonicity of Ω can be discarded, but there might be minimizers of (2.37) which do not

admit the form (2.38). However it still follows that there is at least one other solution which is as good and which does admit the expansion form.

The minimizer of the regularized risk formulation should on one hand minimize the training error, as measured by the cost term L, and on the other hand have a low norm. Since the function spaces are usually extremely rich, for most problems there will be functions with incur no cost at all, for example by just memorizing the examples. However such solutions will be arbitrarily complex and therefore will not generalize well. The regularizer Ω can be understood as seeking to resolve this issue. Loosely speaking it can be seen as favoring smooth functions, where smoothness is measured by the RKHS norm $\|\cdot\|_{\mathcal{H}}$ (see [Sch02] for a detailed review of its regularization properties).

2.3. Learning with Kernels

With the ingredients of the last two sections we can now introduce some kernel based learning algorithms. Given a training set of observations

$$(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y} \tag{2.42}$$

we aim to find a function $f : \mathcal{X} \to \mathbb{R}$ which minimizes the empirical risk on this dataset. For binary classification problems where $\mathcal{Y} = \{-1, +1\}$, this can be posed as the search for a function $f : \mathcal{X} \to \mathbb{R}$ which maximizes the agreement between $\operatorname{sign} f(x)$ and the label of the pattern $y(x)$. We pose this search in the regularized risk framework and search over functions f in the space $\mathcal{H}$ which is implicitly defined by the kernel used to measure similarity between data points. From the representer theorem we know that the only parameters we have to search for are the coefficients of the kernel expansion. Since we search for linear functions in the high, or even infinite dimensional space we will write them also as $f(x) = \langle w, \phi(x) \rangle_{\mathcal{H}}$ or as an affine function $f(x) = \langle w, \phi(x) \rangle_{\mathcal{H}} + b$, with $w \in \mathcal{H}$ and $b \in \mathbb{R}$.

Minimizing the empirical risk with respect to the parameters (w, b) confronts us with several problems. First, the minimization is a NP hard problem [Min69]. Even approximately minimizing the empirical risk is NP hard not only for linear functions but also for other simple geometrical objects such as spheres [BD00]. The optimal function, that is the indicator function $\delta(f(x) \neq y)$, is discontinuous and even small changes in f may lead to large changes in both empirical and expected risk.

In order to overcome the difficulties arising from the exact minimization of the empirical risk, we will use the upper bound of the indicator function and minimize the upper bound. This is not only computationally effective but has also the benefit of yielding consistent estimators [Hof08].

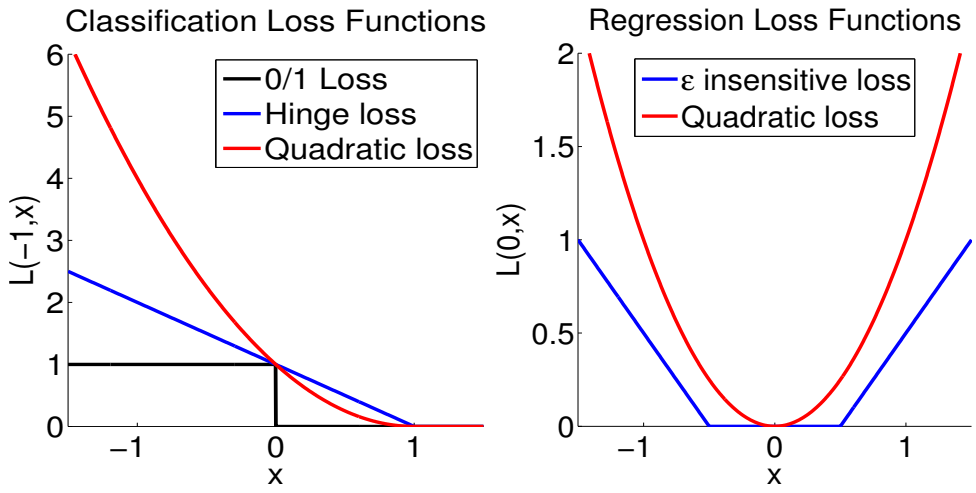


Figure 2.1.: Commonly used loss functions for classification (left) and regression (right) estimation. The 0-1 loss is positive for every element for which the sign does not equal its label. It is discontinuous and not convex. The Hinge and quadratic loss are convex approximations thereof. For regression estimation the ϵ insensitive and quadratic loss are commonly used. Both are symmetric and penalize the deviation from the true target value (in this case 0).

2.3.1. Support Vector Classification

Consider the problem of binary classification with input data

$$(x_1, y_n), \ldots, (x_n, y_n) \subset \mathcal{X} \times \{-1, +1\}. \tag{2.43}$$

We seek a function assigning to each point $x \in \mathcal{X}$ its corresponding label sign. In order to achieve this we implement the search for $f : \mathcal{X} \to \mathbb{R}$ over the function space $\mathcal{H}$ as the following regularized risk functional [Cor95, Vap95].

$$\min_{f \in \mathcal{H}} \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)). \tag{2.44}$$

The final classification function will then be obtained by taking $\text{sign}(f(x))$. Typical choices for the loss function are $L(y,t) = \max\{0, 1-yt\}^p$ with $p = 1, 2$. For $p = 1$ the loss is usually referred to as the *Hinge loss*, for $p = 2$ as the *quadratic loss*. Both are depicted in Figure 2.1 together with the indicator function $\delta(y \neq f(\mathbf{x}))$ (also known as the 0-1 loss function) of which they are both upper bounds. Note that the Hinge as well as the quadratic loss are convex with respect to their second argument. It is due to this convexity that the minimizer of (2.44) is unique. We have also introduced the regularization parameter $C \in \mathbb{R} \cup \{\infty\}$ to the optimization problem in order to control the trade-off between the smoothness of the function measured by $\|f\|_{\mathcal{H}}$ ans its ability to explain the data correctly.

We can set $C = \infty$, in which case (2.44) is also referred to as the *hard margin SVM*. With this choice we enforce the solution f to incur no loss at all [Vap63]. There might not be a solution to this problem because the function class $\mathcal{H}$ may not contain a function that perfectly separates the data.

The SVM formulation (2.44) is a quadratic program (QP) and can thus be solved efficiently [Boy04]. Several algorithmic strategies have been proposed for this particular problem, see for example the SMO algorithm [Pla99]. For a differentiable loss it is conceptually easiest to resort to simple gradient descent techniques. A detailed analysis of a Newton optimization scheme can be found in [Cha07].

2.3.2. Support Vector Regression

As the next example of kernel based learning algorithm we consider the task of regression with target space $\mathcal{Y} \subseteq \mathbb{R}$. Again we use a regularized risk functional and write

$$\min_{f \in \mathcal{H}} \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)). \tag{2.45}$$

Several loss functions can be used, for example the *ϵ-insensitive loss* $L_\epsilon(y,t) = \max\{0, |y-t|-\epsilon\}$ [Vap95, Vap97, Dru97] or the quadratic loss $L(y,t) = (y-t)^2$ yielding penalized least squares regression [Hoe70, Tik63, Mor84, Wah90]. In Figure 2.1 both loss functions are depicted. The representer theorem ensures a finite representation of the optimal solution of (2.45). Plugging this into the problem and for the special case of the quadratic loss, we obtain the following closed form solution for the optimal parameters α^*, where we assume $C > 0$

$$\begin{aligned} \alpha^* &= \operatorname*{argmin}_{\alpha \in \mathbb{R}^n} \frac{1}{2}\alpha^\top \mathbf{K}\alpha + C\|\mathbf{K}\alpha - \mathbf{y}\|^2 & (2.46)\\ &= \left(\mathbf{K} + \frac{1}{2C}\mathbf{I}\right)^{-1} \mathbf{y}, & (2.47) \end{aligned}$$

with $\mathbf{y} = (y_1, \ldots, y_n)^\top$. Due to the structure of the solution this is also referred to as *kernel ridge regression*, the "ridge" $1/2C$ added to the kernel matrix is a consequence of the regularizer. Using the shorthand $\mathbf{K}_\mathbf{x}(\cdot) = (k(\cdot, \mathbf{x}_1), \ldots, k(\cdot, \mathbf{x}_n))^\top$ the prediction function becomes

$$f(\mathbf{x}) = \mathbf{K}_\mathbf{x}(\mathbf{x})^\top \left(\mathbf{K} + \frac{1}{2C}\mathbf{I}\right)^{-1} \mathbf{y}. \tag{2.48}$$

2.3.3. Gaussian Processes

Another way of looking at the regression problem is from the viewpoint of Gaussian Processes (GPs). Gaussian processes provide a probabilistic ap-

proach for kernel learning and are not limited to regression. For a more general overview of the GP framework we refer to [Mac98, Ras06].

A GP defines a distribution over functions $f : \mathcal{X} \to \mathbb{R}$ and is fully described by a mean $m : \mathcal{X} \to \mathbb{R}$ and covariance function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \quad k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{2.49}$$

For notational simplicity we set m to be the zero function. Given a finite collection of data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ we first compute its covariance matrix $\mathbf{K_{xx}}$ in the same way as we did for the Gram matrix (2.5). The covariance matrix defines a distribution over the vector of output values $f_{\mathbf{x}} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^\top$

$$f_{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, \mathbf{K_{xx}}), \tag{2.50}$$

which is a multivariate Gaussian distribution. Therefore the specification of the covariance function implies the form of the distribution over the functions. The role of the covariance for GPs is the same as the role of kernels we used so far, both specify the notion of similarity. This is also the reason we choose kK to denote both quantities.

Let us consider again the task of real-valued regression. Given training data consisting of pairs of inputs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ the goal is to predict the output value y^* for a new test data point $\mathbf{x}^*$. We will assume that the output values we have access to are only noisy observations of the true underlying function $y = f(\mathbf{x}) + \epsilon$. Furthermore we assume the noise to be additive independently identically Gaussian distributed with zero mean and variance σ. For notational convenience we define the following vectors, using bold symbols for vectorial variables; the stacked output values $\mathbf{y} = (y_1, \ldots, y_n)^\top$, the covariance terms of the test point $\mathbf{K}_* = (k(\mathbf{x}^*, \mathbf{x}_1), \ldots, k(\mathbf{x}^*, \mathbf{x}_n))^\top$ as well as $\mathbf{K}_{**} = k(\mathbf{x}^*, \mathbf{x}^*)$. From the model assumption (2.50) we know that the output values are distributed according to

$$\begin{bmatrix} \mathbf{y} \\ f(\mathbf{x}^*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} (\mathbf{K} + \sigma^2\mathbf{I}) & \mathbf{K}_* \\ \mathbf{K}_*{}^\top & \mathbf{K}_{**} \end{pmatrix}\right). \tag{2.51}$$

The predictive equations for the Gaussian Process we are interested in are then obtained by computing the conditional distribution

$$f(\mathbf{x}^*)|\mathbf{y}, \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}^* \sim \mathcal{N}\left(\mathbf{K}_*^\top(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, \mathbf{K}_{**} - \mathbf{K}_*^\top(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_*\right). \tag{2.52}$$

Comparing with (2.48) we observe that the GP mean predictor is exactly the same solution we have obtained for Kernel Ridge Regression. The noise variance term σ^2 appeared as a regularization constant in the kernel ridge regression case. Also note that we have witnessed yet another manifestation of the representer theorem which we have not used explicitly. What differs to Kernel Ridge Regression is that not only a mean prediction is defined but we obtained a full distribution over the output values including an *uncertainty* of

the prediction. Note however that the expression of the predictive variance in (2.52) solely depends on the locations of the training points and not on any training labels. Since usually some parameter selection process is used to determine the kernel parameters the uncertainty term is not independent of the training labels.

2.3.4. Structured Prediction using Kernels

So far we have considered only very simple target spaces $\mathcal{Y}$, for example $\mathcal{Y} = \{-1, +1\}$ and $\mathcal{Y} = \mathbb{R}$. This is however a very limited scenario as in many tasks the objects of interest are more complex than being only binary class membership. For example ranking a set of web-pages according to their relevance for a given query is a task which is not easily expressible in the previously used framework. Making predictions about graph layouts, entire image patches or multi-label problems are a few additional examples which call for a more general framework.

We want to apply kernel methods to all problems of this type and the following simple modification of the kernel function allows us to reuse the results we have obtained so far [Alt04, Tso05, Cai04]. We extend the class of functions to be of the form

$$f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}, \quad (\mathbf{x}, y) \mapsto f(\mathbf{x}, y), \tag{2.53}$$

i.e. they depend on elements of *both* input and target space. Since the predictions we are interested in live in the space $\mathcal{Y}$ we will use the following prediction rule

$$y^*(\mathbf{x}) = \operatorname*{argmax}_{y \in \mathcal{Y}} f(\mathbf{x}, y). \tag{2.54}$$

Note that aside from this new prediction function little has changed from the setup we have developed so far. We just extend the input space and restricted the output space to always be $\mathbb{R}$. The feature map is of the form $\phi : \mathcal{X} \times \mathcal{Y} \to \mathcal{H}$, the corresponding kernel is the dot product in the RKHS $\mathcal{H}$

$$k(\mathbf{x}, y, \mathbf{x}', y') = \langle \phi(\mathbf{x}, y), \phi(\mathbf{x}', y') \rangle_{\mathcal{H}} \tag{2.55}$$

and the representer theorem ensures that the solutions of regularized risk functionals can be expressed in terms of expansions around training data points. The dependency of f on the target space provides the possibility to take its structure into account.

The loss function can also encode the structure in the target set. In binary classification there is no such structure beyond two elements being equal (belonging to the same class) or different (belonging to separate classes). Now consider the task of retrieving a ranked list of websites given some text query. Missing the most relevant website should incur a higher cost than missing the 10th most relevant website. This is encoded using a loss function of the type

$\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$. We will think of $\Delta(y, y')$ as the cost of predicting y' where it should have been y and therefore set $\Delta(y, y) = 0 \ \forall y \in \mathcal{Y}$. If the maximum of (2.54) is taken at the correct labeling, no cost is produced. However, predicting a different y incurs a cost which depends on the similarity of the true and the predicted output. This gives rise to the following regularized risk formulation [Tso05]

$$\min_{f \in \mathcal{H}} \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^{n} \max \left\{ 0, \max_{y \in \mathcal{Y}} \left(\Delta(y_i, y) - (f(\mathbf{x}_i, y_i) - f(\mathbf{x}_i, y)) \right) \right\}. \quad (2.56)$$

This is a convex problem which requires the solving of the inner maximization of (2.54). Since for many problems this can not be done efficiently, one can only hope for approximate solutions. A standard way of solving the problem is by dualizing the problem and using column-generation techniques [Het93, Ben00].

For the regularized risk formulation above the optimal function is of the form

$$f(\cdot, \cdot) = \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} \alpha_{iy} k((\cdot, \cdot), (\mathbf{x}_i, y)). \quad (2.57)$$

We will outline only a few applications of this model and refer to [Bak07] for a more detailed overview.

- The classical binary setup is recovered by simply setting $\phi(\mathbf{x}, y) = y\phi(\mathbf{x})$ and $\Delta(y, y') = \delta(y = y')$. The inner maximum reduces to $1 - 2y_i \sum_{j=1}^{n} \alpha_j y_j k(\mathbf{x}_i, \mathbf{x}_j)$ which (ignoring some offsets) yields exactly the SVM optimization problem (2.44) with the Hinge loss.

- Multiclass classification [Cra00, Col02, All00, Rät03] can be cast in this framework. Let $\mathcal{C}$ denote the number of classes. Then $y \in \{1, \ldots, \mathcal{C}\}$ and the loss function is a multicategory version of the $0-1$ loss, namely $\Delta(y, y') = 1 - \delta(y = y')$. Corresponding kernels are typically chosen to be $\delta(y = y')k(\mathbf{x}, \mathbf{x}')$.

- In the case of multi-label estimation one is interested to predict a set of labels $y \in 2^{\{1,\ldots,n\}}$ for each input point. This is described in [Eli02] where a ranking scheme is devised such that $f(\mathbf{x}, i) > f(\mathbf{x}, j)$ if the label $i \in y$ but $j \notin y$.

- In object localization one is interested in the extent of a visual object in a given image. Most approaches aim to find a *bounding box* around this object which is as tight as possible. The standard technique to do so is to derive a classifier for this object class which subsequently is applied to a number of regions in the test image, e.g. by means of a sliding window approach [Lam08b]. In [Bla08] this problem is posed as a structured regression problem. In their formulation, the extent of the bounding box (its upper left and lower right coordinate) are predicted directly without any detour of a specialized region classifier.

2.3.5. Kernel Principal Component Analysis

Principal Component Analysis (PCA) is a widely used algorithm with many applications such as feature extraction, dimensionality reduction and data visualization. Its benefits include that it is easy to compute and easy to interpret. Given some data $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the PCA is an orthogonal projection of these points onto their principal axes, which are those which minimize the average projection cost measured as the squared distance between the points and their projections. The PCA algorithm boils down to an eigenvalue decomposition of the empirical covariance matrix of the data $C_{\text{emp}} = \mathbf{E}_{\text{emp}}[(\mathbf{x} - \mathbf{E}_{\text{emp}}(\mathbf{x}))(\mathbf{x} - \mathbf{E}_{\text{emp}}(\mathbf{x}))^\top]$, i.e. solving the system of equations $\mathbf{C}_{\text{emp}}\mathbf{v}_k = \lambda_k \mathbf{v}_k$. For d-dimensional data $\mathbf{x}_i$ this problem can be solved in $O(d^3)$ time.

In [Sch98] this problem is posed in the feature space by simply replacing $\mathbf{x}$ with $\phi(\mathbf{x})$. Since the empirical covariance lies in the span of $\{\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)\}$ we can compute it in terms of kernel evaluations at the data points. For notational convenience we assume that we already centered the data in the feature space such that $\sum_{i=1}^n \phi(\mathbf{x}_i) = 0$. Then we can write the eigenvalue problem as

$$C_{\text{emp}}\mathbf{v}_k = \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top \mathbf{v}_k = \lambda_k \mathbf{v}_k \tag{2.58}$$

and thus we can see that the eigenvectors are of the form $\mathbf{v}_k = \sum_{i=1}^n \alpha_{ki}\phi(\mathbf{x}_i)$. Resubstituting this into (2.58) we find that the coefficients α are easily computed by the Eigenvalue problem

$$\mathbf{K}\alpha = \lambda\alpha \tag{2.59}$$

with $\mathbf{K}$ being the kernel matrix of the data. Having solved this eigenvalue problem we can compute the projection of any point $\mathbf{x}$ onto the k-th principal component of the data as $\langle \mathbf{v}_k, \phi(\mathbf{x})\rangle = \sum_{i=1}^n \alpha_{ki} k(\mathbf{x}, \mathbf{x}_i)$.

Kernel PCA can be used as a pre-processing step for algorithms which are not "kernelizable", that is algorithms not based entirely on dot products.

2.3.6. Applications of Support Vector Algorithms

Probably much of the success of kernel based algorithms and the SVM and SVR formulation in particular is due to the empirical success on diverse practical problems. Just to name a few we mention that SVMs were applied to handwritten recognition [DeC02] and achieved the best classification scores on the MNIST [LeC98] benchmark dataset. Visual object classification is another task, e.g. [Bla08, Bla96, Cha99] and Chapter 7 of this thesis. Other applications include Object Detection [Lam08a, Rom01], microarray processing tasks [Bro00], text categorization [Dum98], ranking [Her00], novelty detection [Hay01] and many more. Recently interdependent label problems have been approached by SVMs [McC05, Tso05].

Several authors applied kernel learning algorithms to image processing applications. In the later chapters of this thesis we will investigate in greater details their use for visual object classification in particular. A good source for an overview is [CV07] which includes applications for the classical problems in image processing, namely image coding, image de-noising and image segmentation. In [Kim05] image models based on Kernel PCA are proposed and it is shown that they perform well on image de-noising and super-resolution tasks.

2.4. Conclusion

In this chapter we have introduced the most basic concepts of positive definite kernels and presented some algorithms which build upon them. The main idea is that positive definite kernels provide a measure of similarity between possibly complex objects. With the regularized risk framework one can implement the search over rich classes of functions and still obtain functions which can be expressed in finitely many terms of kernel evaluations. Another benefit is that this search can be made convex and thus can yield problems which not only can be solved efficiently, but also guarantee global optimality.

3. Learning With Ambiguity

A typical assumption for building classification systems is the availability of training data as pairs of instances and corresponding labels. The label information is assumed to be precise in the sense that there is a one to one mapping between the label and the training points. In the Multiple Instance Learning (MIL) scenario this assumption is weakened and the correspondence between label information and training points is ambiguous.

Problems in which label information is ambiguous arise naturally in image classification. Consider the example image from the Caltech-101 dataset depicted in Figure 3.1 (input image). The label attached to this image is "Face". Although the face appears centered in the image it covers only a small fraction of the entire image. Therefore the label information should be interpreted as: "This image contains a face." rather than "The entire image is an example of a face image.". In other words, all we know is that there exists an unknown number of pixels in the image which belong to the shown face.

One might argue that with more label information provided from a user this problem could be solved. Suppose we would have additional information from a user who specified a bounding box around the face like the one in Figure 3.1(a). The problem of ambiguity remains. There is less clutter in this image but from the viewpoint of being a discriminative example for the category of faces, it is unclear whether or not choices (b) or (c) would have been better exemplars. One could standardize the position of the bounding box, e.g. specifying the location w.r.t. to the tip of the nose, but any convention is likely to be suboptimal. In the ideal setting the problem of selecting the most discriminative bounding boxes in the image is shifted to the learning algorithm for two reasons: i) a label for a bag is cheaper to obtain and ii) even if a user labeled all bounding boxes of an image it is unclear which of them to use for training. This is exactly the setting that can be cast as a multiple instance learning problem.

The MIL problem was first introduced in [Die97a] for the task of drug activity prediction. Since then a number of different applications emerged in the literature. Up to now the span of applications cover a variety of problems such as identification of proteins [Tao04], content based image retrieval [Zha02], object detection [Vio06], and prediction of failures in hard drives [Mur05].

In this chapter we report on two contributions. First we propose a disam-

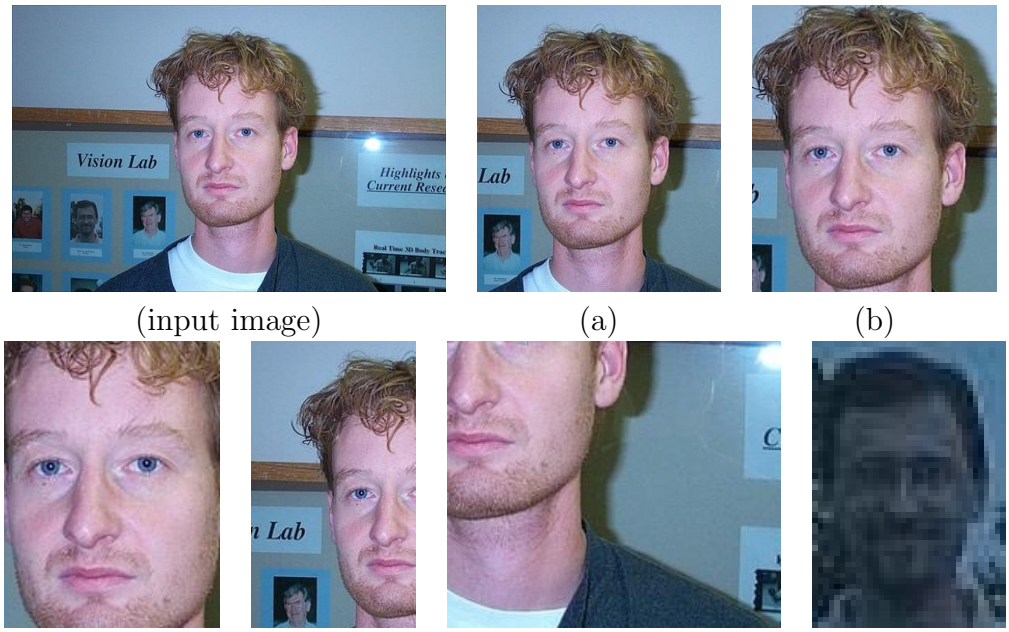


Figure 3.1.: An example image from the Caltech-101 dataset [FF04] (upper left) and several (re-scaled) subwindows thereof. The available label for this image is that it belongs to the category "Face". Although a face appears centered in the image it is unclear which of the image yields the best training example. Should the subwindows (d) and (e) be labeled "face", "non-face" or do they fall into a "void" category? There is more than one face in the image, (f) is a subwindow of (input image) found in the lower left of the image on the board behind the person.

biguation of different scenarios that can arise within the MIL setting. Although a large amount of prior work on the topic exists, most often they solve variations of the problem using the same name [Die97b, Wan02, Zha02, Gär02, Tao04, Vio06, Ray05, Che06b]. We start with a formal definition and argue that because of label ambiguity one has impose further assumptions to the problem. This leads to different problems each of which requires especially tailored methods to solve them. The second contribution is a new optimization technique for the kernel based MIL approaches of [And03, Man05] and a new formulation which overcomes shortcomings of those approaches, identified on a set of benchmark datasets.

3.1. The Multiple Instance Learning Problem

We begin with a formal definition of the multiple instance learning problem. From here on we will denote with upper case letters, bags[1] of elements and their labels and instances with their corresponding labels with lower case letters.

Definition 3.1.1 (Multiple Instance Learning)**.** *Given a training set of n elements $\{(X_i, Y_i)\}_{i=1,\dots,n}$, where each element consists of a bag of instances $X_i = \{x_i^1, \dots, x_i^{m_i}\}, x_i^j \in \mathcal{X}$ and a label $Y_i \in \{-1, 1\}$. With y_i^j we denote the label of the instance x_i^j. We can make the following statements about the instance labels y_i^j:*

$$Y_i = 1 \quad \Rightarrow \exists j_0 \in \{1, \dots, m_i\} \quad : \; y_i^{j_0} = 1, \tag{3.1}$$

$$Y_i = -1 \quad \Rightarrow \forall j \in \{1, \dots, m_i\} \quad : \; y_i^j = -1. \tag{3.2}$$

The bag label information in MIL induces constraints on instance labels in an asymmetric way. A negative labeled bag contains only instances to which a negative label can be assigned to. On the other hand a positive bag label only enforces that the bag contains *at least* one instance in the bag that can be assigned to the positive class. We will refer to this instance as the *witness* of the bag, since it is responsible for the positive label. There is no information about the remaining points, they might even belong to neither the positive nor the negative class. In general the instance labels are from the set $\{-1, 1, ?\}$, where the label ? can be though of as a void category.

We want to emphasize that one has to distinguish between the semantics of bag and instance label and bear in mind that they have a different meaning, e.g. "contains an positive example" versus "is the positive example". We will give some examples of this difference between bag and instance labels in the following section.

There are two possible goals, either a classification function for the bags

$$f : 2^{\mathcal{X}} \to \{-1, 1\}, \quad f(X) \mapsto Y \tag{3.3}$$

or for the instances therein

$$f : \mathcal{X} \to \{-1, 1\}, \quad f(x) \mapsto y \tag{3.4}$$

is sought. Which of the two cases apply depends on the actual problem which is cast as a MIL problem. One can always design a classification function for the instances and compute the bag label as the maximum label of all instances therein.

[1] In the MIL literature some authors confuse the term bag as being a name chosen for this particular problem. The bag which is referred to is the mathematical object, also known as a multiset.

3.2. Label Ambiguity

In order to select an appropriate method dealing with the ambiguity, further assumptions about the generating process of the bags need to be made. We differentiate between three different scenarios that arise due to a different interpretation about the ambiguous labels of the instances in the positively labeled bags.

3.2.1. Instance Scenario

The *Instance Scenario* states that to *all* of the instances $x \in \mathcal{X}$ one can associate either a positive or a negative class label. In other words $y_i^j \in \{-1, 1\}, \forall i, j$. While for all instances in negative bags the label information is already specified, ambiguity remains about instances in the bags with a positive label. Those can be elements of both negative and positive class, the only further information available is that at least one of them comes from the positive class.

Further information about the structure of the problem may be available. For example it could be known that only a certain number of instances in each positive labeled bag are from the positive class, say only one, or not more than three.

Example: Names to Faces

A practical example for this scenario is the problem of assigning names to faces in collection of user photographs. We aim to train a system that can identify a certain person in an image. Assume that for all photographs of a user a face detector retrieved all faces shown in the images. Providing the information whether or not a certain person is shown in a single photo or even a set of photos is of less effort than providing the exact one-to-one mapping of names to faces. Thus we are confronted with an assignment problem during training. If the user labeled a set of images as negative, that means he provided the information the person is not shown in any of them, we have full label information for all of the retrieved faces of these images. For positive labeled sets we simply know that some of the retrieved faces are of the person in question. Importantly, each single face can be assigned either to the positive or the negative class.

3.2.2. Witness Scenario

A second scenario is the following. Only some instances of a positive labeled bag can be assigned a positive label. All other points in that bag are neither from the positive nor the negative class. Here the instance labels take on values in the set $\{-1, 1, ?\}$. The problem becomes the identification of a witness that

is the cause for the bag label. Once this instance has been identified, all other points should be ignored. The crucial difference to the instance assumption is that not all instances can be assigned to a class.

Example: Visual Object Classification

The image classification problem shown in Figure 3.1 illustrates this scenario. Suppose we try to use such image data to construct a object classifier. As argued earlier we should interpret the label "Face" as the information of a face being present somewhere in the image. It is reasonable to assume that subwindows like (a),(b) or (c) are useful examples for this task, but it is unclear which of them to use for training or whether it is better to use all of them. Although the designer of the system has some idea about the object of interest, it is a-priori unclear how much context around the object is needed for a robust classification.

For subwindows like (d) and (e), those which show a part of the face, it is not sensible to make a decision whether a face is shown, these examples are neither in the background nor face class. The witness scenario deliberately avoids assigning labels to *all* subwindows in the image. However there may be more than a single witness per bag. For example in (input image) more than one face is shown. The image patch (f) that is taken from the lower left of (input image) also depicts a face.

For several image classification tasks it has observed that the context, e.g. the region around the object of interest, can provide discriminative information. In [Vio06] the problem of face detection in a video conferencing scenario is described. They face the problem of having only low resolution images where a head is less than 10 pixels wide and thus context is not only an additional source of information, but indeed crucial to detect faces reliably. We want to emphasize that the problem of choosing the correct amount of context is an inherent problem for any visual object classification system. Any user provided bounding box or segmentation is likely to be suboptimal. The ideal system should be capable of selecting the most discriminative bounding box itself, and also has the benefit of requiring an easier image labeling task.

3.2.3. (Sub)Set Scenario

The third scenario is that it is the collection of instances in a bag that constitutes a positive bag label. One can think of the bag as a collection of parts, where each part alone may be necessary but not sufficient for a positive bag label. Additionally, bags may include parts that are not relevant to the bag label at all. We refer to this specific problem as the *Subset Scenario*.

We make the following two observations. The subset scenario can be reduced to the standard witness scenario of form 3.2.2 in the following way. For each bag with a positive label an new bag is formed from the power set $X_i' = 2^{X_i}$ of

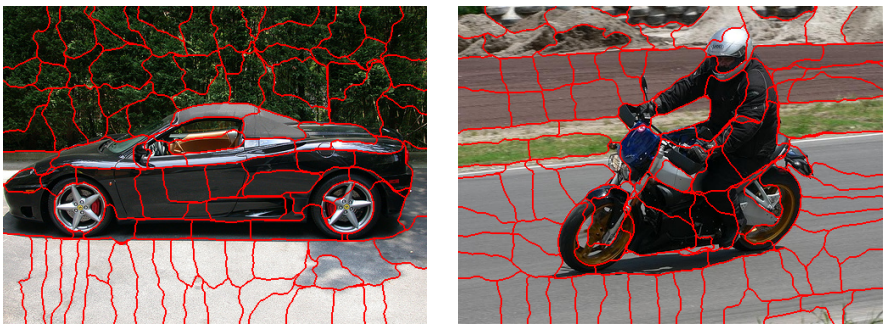


Figure 3.2.: Two images from the VOC-2007 dataset with a segmentation super-imposed. Each segment is regarded as a "superpixel" and described separately. It takes several superpixels to described the entire object. Note that for both images a subset of superpixels exists that is almost perfectly aligned with the object (plus its shadow).

all instances. With this transformation, one element of X_i' will be the witness of the positive label. However, since the superset of X_i is of size 2^{m_i}, this problem renders to be infeasible due to its high dimensionality.

Alternatively this problem could be regarded as a supervised classification task, but on "bag-level". Standard classification techniques can be reused to classify a bags directly, e.g. by deriving kernels for bags instead of comparing instances. Such an approach is proposed in [Gär02], where specific kernels for the MIL problem are developed. Other approaches of this category are the approaches of [Tao04, Che06a].

Example: Visual Object Classification

As argued earlier visual object classification is inherently a multiple instance learning problem. Instead of searching for a single bounding box describing the entirety of an object we could also aim to describe it as its collection of parts. For example [Bur98] and later [Web00, Fer03] considered an object to be composed of parts and shape, where parts correspond to image patches and shape encodes geometry information. This representation is illustrated in Figure 3.2, where two images are shown with super-imposed segmentation obtained by normalized cuts [Shi00]. We refer to each segment as a superpixel of the image and regard a superpixel as the instance, while the image is the bag of superpixels. For both examples there exists a set of superpixel which almost perfectly covers the object. The label of the image (car and motorbike) only provides the information that a set of superpixels exists within the image whose union shows the object. No information is available which of the superpixels contains parts of the object, which yields a multiple instance problem with the

set scenario.

3.3. SVM for Multiple Instance Learning

In the following we will show a way to adapt support vector learning to the multiple instance learning problem. This requires the incorporation of the label ambiguity into the algorithm, and we present ways to do so for the instance and the witness scenario.

3.3.1. SVM for the Instance Scenario

In the instance scenario one is confronted with an assignment problem for instances in the positive labeled bags. The first one to introduce SVM classifiers to the MIL problem was [And03] who introduced the *mi-SVM* that we will review in this section.

For each instance $x_i^j \in X_i$ in every training bag X_i we introduce a discrete variable $y_i^j \in \{-1, 1\}$ to serve as its label. The objective is to optimize over this new set of discrete variables jointly with the SVM parameters. The objective function of mi-SVM is

$$\min_{f \in \mathcal{H}, \{y_i^j\}} \quad \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^{n} \sum_{j=1}^{m_i} L\left(y_i^j, f(x_i^j)\right) \tag{3.5}$$

$$\text{sb.t.} \quad y_i^j \in \{-1, 1\}, \quad \forall i = 1, \ldots, n, j = 1 \ldots, m_i, \tag{3.6}$$

$$y_i^j = -1, \quad \forall i : Y_i = -1, j = 1, \ldots, m_i, \tag{3.7}$$

$$\sum_{j=1}^{m_i} \frac{y_i^j + 1}{2} \geq 1, \quad \forall i : Y_i = 1. \tag{3.8}$$

Note that two constraints (3.8) and (3.7) ensure label consistency with the bag labels. Due to the presence of discrete variables y_i^j this problem is an integer program. We will defer optimization strategies for this formulation to Section 3.4.

3.3.2. SVM for the Witness Scenario

Two SVM variants that are build for the witness scenario were originally proposed in [And03] and [Man05]. We begin with the *MI-SVM* formulation of [And03]. For positively labeled bag X_i we introduce a binary vector $\mathbf{s}_i = (s_i^1, \ldots, s_i^{m_i}) \in \{0, 1\}^{m_i}$ that encodes the selection of the witness in this bag. For bags with a negative label we fix $s_i^j = 1, j = 1, \ldots, m_i$ and require for positive labeled bags that only one witness is selected $\sum_{j=1}^{m_i} s_i^j = 1$. The

optimization problem is

$$\min_{f\in\mathcal{H},\mathbf{s}} \quad \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{n}\sum_{j=1}^{m_i} s_i^j L\left(Y_i, f(x_i^j)\right) \tag{3.9}$$

$$\text{sb.t.} \quad s_i^j \in \{0,1\}, \quad \forall i : Y_i = 1, j = 1,\ldots,m_i, \tag{3.10}$$

$$s_i^j = 1, \quad \forall i : Y_i = -1, j = 1,\ldots,m_i, \tag{3.11}$$

$$\sum_{j=1}^{m_i} s_i^j = 1 \quad \forall i : Y_i = 1. \tag{3.12}$$

This problem is an integer program due to discrete variables $\mathbf{s}$. A slightly different formulation was proposed in [Man05] under the name *MICA*,

$$\min_{f\in\mathcal{H},\nu} \quad \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{n}\left[\delta(Y_i = 1)L\left(1, \sum_{j=1}^{m_i}\nu_i^j f(x_i^j)\right)\right. \tag{3.13}$$

$$\left. +\delta(Y_i = -1)\sum_{j=1}^{m_i} L\left(-1, f(x_i^j)\right)\right] \tag{3.14}$$

$$\text{sb.t.} \quad \nu_i \in \Delta_{m_i} \quad i = 1,\ldots,n, \tag{3.15}$$

where we denote by Δ_m the m dimensional simplex. MICA is not directly identifying a single witness in the bag but a convex combination of all points in a bag that acts as a witness. This removes the integer representation involved in the MI-SVM at the expense of adding bilinear constraints to the program.

The solutions of MICA and MI-SVM differ only in the following case. At the optimal solution consider the set W_i of indices from instances in bag X_i that incur minimum loss

$$W_i = \left\{\operatorname*{argmin}_{i=1,\ldots,m_i} L(1, f(x_i^j))\right\}. \tag{3.16}$$

For all bags X_i where there is only one such instance, i.e. $|W_i| = 1$ the solutions to both formulations agree: $s_i^j = \nu_i^j, \forall j = 1,\ldots,m_i$. Otherwise the objective value of (3.9), resp. (3.13) could be lowered by changing s_i^j, resp. ν_i^j to be 1 for $j \in W_i$. For the bags with $|W_i| > 1$ multiple equally good solutions exist. Each of the elements W_i can be chosen as a witness s_i^j and likewise all ν_i with $\sum_{j\in W_i}\nu_i^j = 1$ have the same objective value. Hence it is only for those bags that the solution of the two formulations differ and it depends on the algorithm used which among the possible solutions is found.

3.4. Optimization Strategies

Most formulations proposed for multiple instance learning share the problem of being combinatorial problems of the instance labels, including mi-SVM, MI-SVM and MICA. In this section we will describe the strategies proposed in the

original works which introduced the SVM formulations [And03, Man05] and devise a deterministic annealing algorithm to solve for better local optima of the problems.

3.4.1. Alternation Algorithm

Both the authors of the mi-SVM and MI-SVM [And03] and of MICA [Man05] propose iterative algorithms to solve their formulations.

Solving the Instance SVM

The algorithm for solving the mi-SVM proposed in [And03] is summarized in Algorithm 1. First, the instance labels are initialized with their bag label. Now the following two steps are alternated until convergence. Using the current assignment an SVM is trained and the resulting function is used to classify all instances, thus determining their new values. If necessary, constraint (3.8) is enforced by switching labels of the least negative instances in a bag for which the constraint is violated.

This alternating algorithm was shown in [Che06c] to be an instance of a Convex-Concave-Procedure [Yui02, An05] and thus is guaranteed to monotonically decrease the objective function and converge to a local minima.

Algorithm 1 Alternation for mi-SVM

Input: Training data $\{(X_i, Y_i)\}_{i=1,\dots,n}$, Regularization parameter $C > 0$
Output: Local optimum $(f, \mathbf{y})$

1: $y_i^j \leftarrow Y_i \; \forall i = 1, \dots, n, j = 1, \dots, m_i$.
2: **while** The assignments $\{y_i^j\}$ changed **do**
3: $\quad f \leftarrow$ SVM solution using labels $\{y_i^j\}$.
4: $\quad y_i^j \leftarrow \text{sign} f(x_i^j), \; \forall i : Y_i = 1, j = 1, \dots, m_i$
5: $\quad$ **if** constraint (3.8) violated **then**
6: $\quad\quad j_i' \leftarrow \text{argmax}_{j=1,\dots,m_i} f(x_i^j), \; i : Y_i = 1$
7: $\quad\quad y_i^{j_i'} \leftarrow 1$
8: $\quad$ **end if**
9: **end while**
10: **return** $f, \mathbf{y}$

This optimization procedure initializes instance labels to be identical to the bag label. We observed that for each iteration of this algorithm only few labels are changed and the algorithm is biased toward solutions with a large number of positive labeled points.

Solving the Witness SVM

The suggested procedure to solve the MICA and MI-SVM is analogue to the algorithm described above. For both MICA and MI-SVM the algorithm is initialized using the mean of the instances in positive labeled bags as witnesses of the bags. Subsequently the training of an SVM is alternated with the search for the new witness. In the case of MI-SVM the instance with maximal function value $f(x_i^j)$ is chosen and for MICA a linear program has to be solved to obtain the new values of $\{\nu_i^j\}$.

3.4.2. Deterministic Annealing

Deterministic annealing, e.g. [Ros98], is a special case of a homotopy method and may be applied in a more general context than introduced here. Suppose one is confronted with a non-convex optimization problem of the form

$$\mathbf{y}^* = \underset{\mathbf{y}\in\{0,1\}^n}{\operatorname{argmin}} F(\mathbf{y}). \tag{3.17}$$

Deterministic annealing finds a local minimum of this function as follows. The discrete variables $\mathbf{y}$ are regarded as random binary variables with a probability distributions from a space $\mathcal{P}$. Instead of solving the optimization problem directly one searches for a distribution $p \in \mathcal{P}$ which minimizes the expected value of F. By doing so, the optimization problem becomes continuous but is not easier to solve. Therefore, an additional convex term is added to the objective function: the entropy $\mathcal{S}$ of the distribution. The new problem becomes

$$p^* = \underset{p\in\mathcal{P}}{\operatorname{argmin}} E_p\left(F\left(\mathbf{y}\right)\right) - T\mathcal{S}(p). \tag{3.18}$$

The parameter T controls the trade-off between the expectation and the entropy and is called the *temperature* of the problem.

As a first observation, note that the function F always has a (not necessarily unique) global minimum due to the finiteness of the possible parameters $\mathbf{y} \in \{0,1\}^n$. For $T = 0$ and $\mathcal{P}$ including all point-mass distributions over $\{0,1\}^n$ the global minimizer p^* of the problem above will put all of its probability mass on the global minimizers of F. Thus the new formulation preserves the optimality of the original problem. If on the other hand $T \gg 0$ the entropy term in (3.18) dominates the objective function and the problem will be solved easily thanks to convexity.

A solution to (3.18) is found by solving a sequence of problems for values of $T_0 > T_1 > \ldots > T_\infty = 0$, each of which is initialized at the solution obtained by the previous one. This sequence of temperatures is referred to as the *annealing schedule*. As T approaches zero, the influence of the entropy term vanishes and the distribution will become more concentrated on a local minimum of $E_p[F]$. In this case we can identify the discrete variables $\mathbf{y}$ from p^*.

However, there is no guarantee for global optimality because there might not be a path connecting the local minimizers for the chosen annealing schedule to the global optimum of F.

3.4.3. Deterministic Annealing for Instance-SVM

The goal of the instance scenario based mi-SVM is to infer all missing labels of the instances in positive labeled bags. Following the deterministic annealing principle, we will regard the label y_i^j of a instance $x_i^j \in X_i$ from a positive labeled bag $Y_i = 1$ as a binary random variable.

In principle our space of distributions $\mathcal{P}$ could consist of all possible distributions over y_i^j. For simplicity we use as search space $\mathcal{P}$ the space of all factorial distributions that can be written as

$$P(\mathbf{y}) = \prod_{i=1}^{n} \prod_{j=1}^{m_i} P(y_i^j). \tag{3.19}$$

This class does include distributions which assign a non-zero probability to configurations of instance labels that violate constraint (3.8). However this fact does not pose a problem, if we can assure that at the last step of the deterministic annealing algorithm a distribution is reached which ensures consistency with the constraint. In other words, the final distribution $\mathbf{p}$ must concentrate its mass on a valid distribution $\mathbf{y}$ of label assignments but the path taken to this optimum does not need to satisfy consistency.

With p_i^j we denote the probability $P(y_i^j = 1)$ and consequently we have $P(y_i^j = -1) = (1 - p_i^j)$. One can think of the value of p_i^j as the belief that the instance x_i^j belongs to the positive (instance) class. To simplify the notation we will introduce but fix $p_i^j = 0$ for all instances from negative labeled bags. The entropy of the distribution (3.19) is

$$\mathcal{S}(\mathbf{p}) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(p_i^j \log p_i^j + \left(1 - p_i^j\right) \log \left(1 - p_i^j\right) \right). \tag{3.20}$$

In summary, we use all factorial distribution (3.19), apply the substitution (3.18) to (3.5), and add the entropy term (3.20) to arrive at the minimization

program

$$\begin{aligned}
\min_{p,f\in\mathcal{H}} \quad & \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{n}\sum_{j=1}^{m_i}\left[p_i^j L\left(1, f(x_i^j)\right)\right. \\
& \left. + \left(1-p_i^j\right) L\left(-1, f(x_i^j)\right)\right] && (3.21)\\
& + T\mathcal{S}(\mathbf{p}) && (3.22)\\
\text{sb.t.} \quad & \sum_{j=1}^{m_i} p_i^j \geq 1, \quad i = 1,\ldots,n, && (3.23)\\
& 0 \leq p_i^j \leq 1, \quad i = 1,\ldots,n, j = 1,\ldots,m_i && (3.24)\\
& && (3.25)
\end{aligned}$$

Iteratively solving (3.21)

The problem (3.21) needs to be solved for a sequence of decreasing temperatures T. First we need to check whether the optimum point of this program is a valid assignment of the instance labels. To see that this is the case, we note that in the case of zero temperature $T = 0$ the entropy term vanishes. For each instance x_i^j the objective function is minimized by setting

$$p_i^j = \begin{cases} 1 & L(-1, f(x_i^j)) > L(1, f(x_i^j)) \\ 0 & L(-1, f(x_i^j)) < L(1, f(x_i^j)) \end{cases}. \tag{3.26}$$

If with this choice constraint (3.23) is violated, the minimal objective is achieved by setting $p_i^j = 1$ for the instance with minimal loss $L(1, f(x_i^j))$. Thus an optimal point is achieved for a distribution $\mathbf{p}$ that takes on only values 0 or 1.

For each new temperature T we start a search for the optimum solution using the previously found point $(f, \mathbf{p})$ and alternate between the updates of f and $\mathbf{p}$ until we converge to the next local minimum. For fixed $\mathbf{p}$ the classification function f can be found using any SVM solver. A simple way to do so is to duplicate the instances from the positive labeled bags (one with a positive label and one with a negative) and use two different costs for each instance, namely Cp_i^j and $C(1-p_i^j)$.

The optimal value of $\mathbf{p}$ can be found analytically for each new f. This can be seen as follows. Let d_i^j denote the difference of positive and negative loss for instance x_i^j

$$d_i^j = L\left(1, f(x_i^j)\right) - L\left(-1, f(x_i^j)\right). \tag{3.27}$$

Since only parameters within one bag are coupled, one can optimize for each bag separately. We denote the parameters in bag i with $\mathbf{p}_i$ and write the

Lagrangian with respect to this parameter

$$\begin{aligned}\mathcal{L}(\mathbf{p}_i, \lambda_i) &= C \sum_{j=1}^{m_i} p_i^j d_i^j && (3.28)\\ &+T \sum_{j=1}^{m_i} \left(p_i^j \log p_i^j + \left(1 - p_i^j\right) \log \left(1 - p_i^j\right)\right) && (3.29)\\ &+\lambda_i \left(\sum_{j=1}^{m_i} p_i^j - 1\right). && (3.30)\end{aligned}$$

Taking the derivative w.r.t. p_i^j and equating to zero yields the following expression for the optimal value

$$p_i^j(\lambda_i) = \sigma\left(\frac{-Cd_i^j + \lambda_i}{T}\right), \tag{3.31}$$

where $\sigma(t) = (1 + \exp(-t))^{-1}$ denotes the sigmoid function. We wrote $p(\lambda_i)$ to make explicit the dependency of the solution on the Lagrange multiplier. Since $\sigma(t) \in (0, 1)$, the solution always satisfies $0 \leq p_i^j \leq 1$.

To solve for p_i^j one checks whether

$$\sum_{j=1}^{m_i} p_i^j(0) \geq 1, \tag{3.32}$$

in which case the constraint is satisfied and thus the Lagrange multiplier $\lambda_i = 0$. Otherwise we know that the constraint will be tight, that is $\sum_{j=1}^{m_i} p_i^j(\lambda_i) = 1$ which implies

$$p_i^j = \frac{\sigma\left(-\frac{C}{T} d_i^j\right)}{\sum_{j'=1}^{m_i} \sigma\left(-\frac{C}{T} d_i^{j'}\right)}. \tag{3.33}$$

The computation of $\mathbf{p}$ given the SVM parameters can thus be done very efficiently and only incurs marginal costs compared to the quadratic programs one has to solve at each iteration.

The quadratic program can be initialized with the solution from the previous iteration to speed up convergence. In our implementation we use Newton optimization [Cha07] which most often only requires one additional step to converge to a new solution after updating $\mathbf{p}$.

In order to start with an easy convex program we have to choose T_0 to ensure that we start with high entropy distributions. We found that choosing $T_0 = 10C$ is sufficient to ensure a high entropy and thus $p_i^j \approx 0.5$ for all experiments. The final algorithm AL-SVM for solving (3.21) is summarized in Algorithm 2.

Algorithm 2 Deterministic Annealing for identifying all Labels (AL-SVM)

Input: Training data $\{(X_i, Y_i)\}_{i=1,\dots,n}$, Regularization parameter $C > 0$
Output: Local optimum $(f, \mathbf{y})$

Set $p_i^j \leftarrow \begin{cases} \frac{1}{2} & \text{if } Y_i = 1 \\ 0 & \text{otherwise} \end{cases}$.
Set $T \leftarrow 10C$ {relatively high temperature}
while $\mathcal{S}(p) > \epsilon$ **do**
  while $\mathbf{p}$ changed **do**
    $f \leftarrow$ SVM solution using instances with cost Cp_i^j and $C(1 - p_i^j)$
    $\mathbf{p} \leftarrow$ solve using Eq.(3.31)+(3.33)
  end while
  $T \leftarrow T/1.5$ {annealing schedule}
end while
$\mathbf{y} \leftarrow \mathbf{p}$
return $f, \mathbf{y}$

The alternating algorithm of Section 3.4.1 is equivalent to the deterministic annealing algorithm if we initialize $p_i^j = (Y_i + 1)/2, j = 1, \dots, m_i$, and start the annealing schedule with $T_0 \approx 0$. In the experiments we use $T_0 = 10^{-8}$ to emulate his case.

3.4.4. Deterministic Annealing for Witness-SVM

We will now show how to use deterministic annealing for the SVMs implementing the witness scenario. Again only variables within one bag are coupled and we introduce a multinomial distribution for bag X_i: $\mathbf{p}_i = (p_i^1, \dots, p_i^{m_i})$ with $\sum_{j=1}^{m_i} p_i^j = 1$. The value p_i^j is the probability that the instance j in bag i is the witness of the bag label. For negative labeled bags we set $p_i^j = 1$, effectively treating them as bags with a single instance, and keep these values fixed. The entropy of this distribution is

$$\mathcal{S}(\mathbf{p}_i) = \sum_{j=1}^{m_i} p_i^j \log p_i^j. \tag{3.34}$$

Applying the deterministic annealing principle to the objective function (3.9) we arrive at the following problem

$$\min_{\mathbf{p}, f \in \mathcal{H}} \quad \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^{n} \sum_{j=1}^{m_i} p_i^j L\left(Y_i, f(x_i^j)\right) + T \sum_{i=1}^{n} \mathcal{S}(\mathbf{p}_i) \tag{3.35}$$

$$\text{sb.t.} \quad \sum_{i=1}^{m_i} p_i^j = 1, \quad \forall i : Y_i = 1, \tag{3.36}$$

$$0 \le p_i^j \le 1 \quad i = 1, \dots, n, j = 1, \dots, m_i. \tag{3.37}$$

For each temperature T we solve iteratively by updating f and $\mathbf{p}$ to find the next local minimum. The search for f reduces to a SVM problem with cost Cp_i^j for instance x_i^j.

For a given f we find the update rule for $\mathbf{p}$ again by writing the Lagrangian and equating to zero (see Eq. (3.28))

$$p_i^j = \frac{\exp\left(-\frac{C}{T}L(Y_i, f(x_i^j)\right)}{\sum_{j'=1}^{m_i}\exp\left(-\frac{C}{T}L(Y_i, f(x_i^{j'})\right)}, \tag{3.38}$$

Again the updates of p_i^j can be computed with only little additional cost.

A high value of T enforces high entropy distributions and in this regime $p_i^j \approx 1/m_i$. The lower the value of T the more concentrated the distributions will become and thus p_i^j will be concentrated on instances which incur low cost. In the extreme $T \to 0$ only points with $L(Y_i, f(x_i^j)) = 0$ will have $p_i^j > 0$, or if all points in a bag have a positive cost the one with the least positive output will be chosen as the witness of the bag label.

The latter case for $T = 0$ is exactly the same solution found by the alternation algorithm proposed for MICA. This is seen by identifying $p_i^j = \nu_i^j$. We are optimizing the same objective using deterministic annealing rather than the alternating algorithm proposed in [Man05] and described in Section 3.4.1. In the case of several equally good solutions for MICA the entropy term determines the choice of the convex combinations, namely those concentrated on one point only. Therefore the set of optimal points of MICA and MI-SVM are identical.

Again $T_0 = 10C$ was used as a starting value for the temperature. The complete algorithm is summarized in Algorithm 3.

3.5. Experiments: Two dimensional toy dataset

To compare the differences between the algorithm from [And03, Man05] and the annealing algorithm, we conducted an experiment using synthetic 2D data. In this way we control the fraction of instances with a positive label in the positive labeled bags and therefore test the results for instance as well as for bag accuracy. In the typical MIL scenario the assessment of the accuracy obtained on individual instances is impossible due to the ambiguity of the label information.

Experimental Setup

Ten different datasets were created by varying the fraction of positive labeled points per bag over $f = 0.1, 0.2, \ldots, 1$. A bag was generated in the following way. The label Y_i and the size m_i were sampled uniform at random from

Algorithm 3 Deterministic Annealing for identifying the Witness (AW-SVM)

Input: Training data $\{(X_i, Y_i)\}_{i=1,\ldots,n}$, Regularization parameter $C > 0$
Output: Local optimum $f, \mathbf{y}$

Set $p_i^j \leftarrow \begin{cases} \frac{1}{m_i} & \text{if } Y_i = 1 \\ 1 & \text{if } Y_i = -1 \end{cases}$
Initialize $T \leftarrow 10C$
while $\mathbf{p}$ changed in the inner loop **do**
  while $\mathbf{p}$ changed **do**
    $f \leftarrow$ SVM solution using cost Cp_i^j for instance x_i^j
    $\mathbf{p} \leftarrow$ update according to Eq. (3.38)
  end while
  $T \leftarrow T/1.5$ {annealing schedule}
end while
$y_i^j \leftarrow 1$ for all $p_i^j > \epsilon$ and $Y_i = 1$
$f \leftarrow$ SVM solution using $\mathbf{y}$
return $f, \mathbf{y}$

$\{-1, 1\}$ and $\{1, 2, \ldots, 10\}$ respectively. If the bag label is negative we sampled m_i instances uniformly from the black region (negative class) in the leftmost picture in Figure 3.3(a). A positive labeled bag consists of $\lceil f m_i \rceil$ points sampled uniformly from the white region (positive class) and the remaining $\lfloor (1-f) m_i \rfloor$ points from the negative class. For each fraction f we sampled 30 training and 100 test bags.

The hyperparameters were fixed to $C = 100$ and we used a Gaussian kernel with bandwidth $\sigma = 1$. With this data we trained the AL-SVM and AW-SVM with two different starting temperatures, $T = 10^{-8}$ and $T = 10C$. [2]

Results and Discussion

The averaged results over 50 independent runs are shown in the Figure 3.3(b)+(c).

The experiments reveal an important property of the algorithms. The first observation is, that the formulation implementing the witness scenario performs equally well whether or not the deterministic annealing algorithm is used. For this dataset both methods seem not to be prone to local minima. Due to the simple structure of the toy dataset it is more or less irrelevant which instances are identified as the witness of positive labeled bags. As the ambiguity decreases ($f \to 1$), the bag accuracy increases while the instance accuracy decreases. A decision surface is shown in Figure 3.3(d). The number of positive instances is underestimated but all bags are correctly identified.

For the instance SVMs we observe the following. With only a low number

[2] Matlab code used for the experiments is available online at `http://www.kyb.mpg.de/bs/people/pgehler/mil/mil.html`

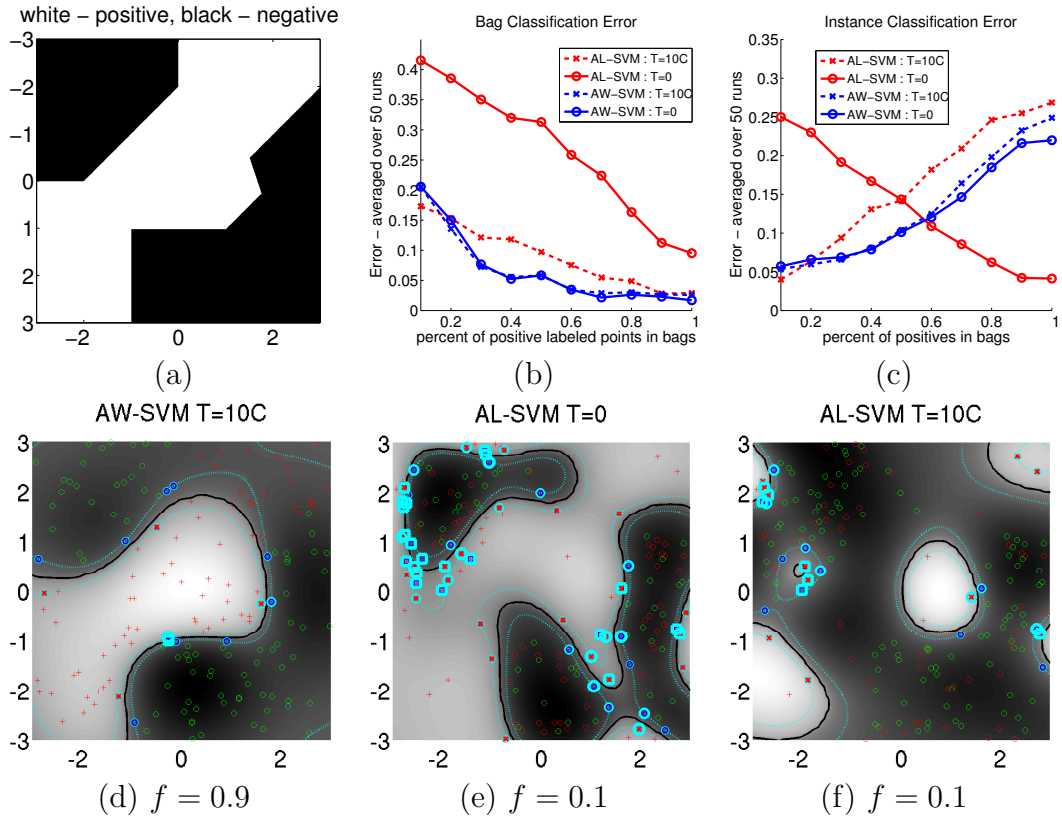


Figure 3.3.: A 2D toy dataset. (a) distribution of instance labels, instances in the white region are labeled positive, instances of the black region negative. (b)+(c) error rates for a varying number of positive labeled instances per positive bag. In (d)-(f) three different decision functions of AW-SVM (d) and AL-SVM (e)+(f) are shown. The dark area corresponds to the support of the negative class.

of true positive instances ($f \to 0$) the mi-SVM yields very high error rates on both bag- as well as instance-level. This is due to the fact that the initialization (setting all instances of positive labeled bags to the positive class) creates a bias towards solutions of a high number of instance labels. A decision function is plotted in (e) and it can be seen that too many instances are classified to be positive (white region). Using the annealing algorithm the opposite effect appears, too few instances are labeled positive. This is illustrated in Figure 3.3 (f) where the dark region is dominant.

We conclude that both algorithms to solve the SVM implementing the instance scenario come with a problem: the annealing scheme underestimates the number of positive points, whereas the simple alternating steps overestimate this number.

3.6. A new objective function - ALP-SVM

The findings in the previous section raise the question of whether the objective function of the mi-SVM Eq.(3.5) is suited for the MIL problem at all. The problem of underestimating the number of positive labeled instances motivates the following extension of the objective function. We include a term in the objective function which penalizes a deviation from a pre-specified number of positive points

$$\min_{f\in\mathcal{H},\{y_i^j\}} \quad \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{n}\sum_{j=1}^{m_i} L\left(y_i^j, f(x_i^j)\right) \tag{3.39}$$

$$+D\sum_{i=1}^{n}\left(\sum_{j=1}^{m_i}\frac{y_i^j+1}{2} - m_i p_i^*\right)^2 \tag{3.40}$$

$$\text{sb.t.} \quad y_i^j \in \{-1,1\}, \tag{3.41}$$

$$y_i^j = -1, \quad \forall i : Y_i = -1, \tag{3.42}$$

$$\sum_{j=1}^{m_i}\frac{y_i^j+1}{2} \geq 1, \quad \forall i : Y_i = 1. \tag{3.43}$$

The difference to the mi-SVM formulation is the additional term (3.40). A new hyperparameter p_i^* can be used to control the expected number of positive labeled points per bag. Assignments $\{y_i^1, \ldots, y_i^{m_i}\}$ that deviate from this fraction are penalized. For bags with a negative bag label we set $p_i^* = 0$ because we know that there are no positive labeled points in these bags. This way an over– and underestimation of the fraction of positive labeled points per bag can be avoided, similar to a *balancing constraint* in semi-supervised learning. A balancing constraint ensures that the fraction of positive to negative labeled points estimated on the unlabeled point set is the same as that from the labeled training examples. In semi-supervised learning this quantity can be estimated from the training set. This is not possible in a MIL scenario, where there is ambiguity of the data and therefore no obvious way of how to choose this value.

The value for p_i^* can either be prefixed due to prior knowledge or be left open as a hyperparameter estimated via cross validation. As the number of parameters to be estimated scales with the number of positive bags we will simplify by setting $p_i^* = p_j^* \ \forall i,j : Y_i = Y_j$.

The objective function (3.39) can be optimized using the deterministic annealing algorithm with the only difference to (3.21) being the addition of the term

$$D\sum_{i=1}^{n}\left(\sum_{j=1}^{m_i} p_i^j - m_i p_i^*\right)^2. \tag{3.44}$$

Name	#Bags+	#Bags-	#Inst+	#Inst-	dim
Corel	100	100	651	660	143
Musk1	47	45	207	269	166
Musk2	39	63	1017	5581	166

Table 3.1.: Statistics of the benchmark datasets: number of positive labeled bags #Bags+, negative labeled bags #Bags-, points in positive bags (average) #Inst+ and points in negative bags #Inst- and the dimension of the dataset

This changes the update rule for $\mathbf{p}$. Again the parameters can be optimized for each bag independently. For a fixed function f we solve (3.39) using the conjugate gradient method[3] while ignoring constraint (3.23). If a solution does not satisfy (3.23), i.e. is outside the feasible region of ALP-SVM we know that a solution of the constraint problem will lie on the simplex $\sum_{j=1}^{m_i} p_i^j = 1$. In this case (3.44) is simply a constant and thus the solution is the same as for the AL-SVM.

To solve problem (3.39) one can use Algorithm (2) with the additional parameters D, p^* and the new update rule of $\mathbf{p}$ in step 6.

3.7. Experiments: Benchmark MIL datasets

For a comparison of the proposed algorithms to those published in the literature and especially the SVM programs described above we conducted experiments on some benchmark datasets for the MIL problem. We used the MUSK and the COREL datasets (Elephant, Fox, Tiger) introduced in [And03].[4] Statistics of these datasets are shown in Table 3.1.

Experimental Setup

Again in a first set of experiments we compare deterministic annealing to the alternation algorithm. We used $T = 10^{-8}$ and initialized all instance labels to their corresponding bag label to emulate the alternating algorithm.

The results published in [And03, Man05] were obtained using different regularization functions Ω. To unify the presentation and to enable a comparison between algorithms not design choices we use the formulation (2.44), i.e. $\Omega(x) = x$ and the quadratic loss for the annealing algorithm.

A Gaussian kernel with the bandwidth set to the median of the pairwise instance distances denoted by σ_{emp} is used for the first set of experiments.

[3] `http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/`
[4] `http://www.cs.columbia.edu/~andrews/mil/datasets.html`

	AL-SVM, T=0		AL-SVM, T=10C		ALP-SVM, T=10C	
	err	$\hat{p}$	err	$\hat{p}$	err	$\hat{p}$
Musk1	**14.3**	100%	20.6	38%	**14.3**	99%
Elephant	24.0	91%	30.5	14%	**16.5**	58%
Fox	43.5	60%	38.5	16%	**35.0**	72%
Tiger	25.0	79%	30.5	19%	**14.0**	60%

Table 3.2.: Results on several benchmark datasets. Standard deviation of the 10x fold cross validation error is usually around 3.5%. Also shown is the averaged error (err) and the fraction of instances per bag $\hat{p}$ that are assigned positive labels.

The remaining hyperparameters were optimized using 10 fold cross validation. We searched over the grid $C \in \{1, 10\}$, $D \in \{1, 10\}$ and $p^* \in \{0.1, .0.2, \ldots, 1\}$.

Results and Discussion

Results are shown in Table 3.2. In addition to the cross validation error we also report the average fraction of estimated positive instances in a positive bag $\hat{p}$. For this quantity there is no ground-truth value available, since labels are only given for bags. On all dataset we observe the same behavior as in the 2D toy example. Setting $T = 0$ leads to high values of $\hat{p}$ while using the annealing algorithm $T = 10C$ yields to a low value of positive labeled instances $\hat{p}$. The ALP-SVM penalizes deviation from the pre-specified fraction p^* and therefore overcomes this problem by finding solutions which lie "in between". Using the new objective function we obtain better results on all COREL datasets. For the MUSK1 dataset there was no better solution found than setting all instance labels positive, a solution also found by the ALP-SVM.

A final set of experiments was done on all the datasets described above as well as on MUSK2. We used the same grid of hyperparameters as in the initial experiments but now also varied the width of the kernel bandwidth in the interval $\sigma \in \{\sigma_{emp}, 2\sigma_{emp}, 0.5\sigma_{emp}\}$. Best performance was almost always attained using σ_{emp}.

The final results together with those reported in [Zha02, Man05, And03] are shown in Table 3.3. Note that the results obtained using MICA,MI-SVM and AW-SVM on the one and mi-SVM and AL-SVM on the other hand despite their similarity vary quite a bit. We therefore suspect that the datasets are very sensitive to model selection.

The experiments show that deterministic annealing does not help for the formulations identifying the witness. For the MUSK datasets it even worsens the performance. However we have to emphasize that using deterministic annealing one always achieves a lower value of the objective function (numbers

	EM-DD	MI-SVM	MICA	AW-SVM		mi-SVM	AL-SVM		ALP-SVM	
MUSK1	15.2	22.1	15.6	14.3	20.6	**12.6**	14.3	20.6	13.7	$\hat{p} = 1$
MUSK2	15.1	15.7	**9.5**	16.2	20.8	16.4	17.4	13.8	13.8	$\hat{p} = 0.28$
Elephant	21.7	18.6	17.5	18.0	19.0	17.8	20.5	29.0	**16.5**	$\hat{p} = 0.58$
Fox	43.9	42.2	38.0	36.5	37.0	41.8	36.5	37.0	**34.0**	$\hat{p} = 0.71$
Tiger	27.9	16.0	18.0	17.0	17.0	21.6	21.5	28.0	**14.0**	$\hat{p} = 0.6$

Table 3.3.: Results on several benchmark datasets. Left column in AW-SVM and AL-SVM are results obtained with $T_0 = 10^{-8} \approx 0$, whereas the right column states the result for $T = 10C$. The standard deviation of the 10x fold error is usually around 3.5% for our experiments. The published result for MICA was obtained using a ℓ_1 penalization of the SVM parameters.

not reported here). The fact that a better local minimum does not translate into better performance, indicates that the objective function is inadequate or that the correct identification of the witness is not important to classify the bag correctly.

The results of the ALP-SVM are promising. Using this formulation the under/overestimation of $\hat{p}$ is overcome and a better local minima of the objective function also yields a better classification performance. In the direct comparison with an annealed and non-annealed AL-SVM the cross validation error is lower on all datasets. As the results using the ALP-SVM are better than those from AW-SVM,MI-SVM and MICA (except MUSK2) it seems that the latter methods make inefficient use of the available information by using only one point per bag for building the decision function.

3.8. Conclusion

In this chapter we presented the multiple instance learning problem as a way to learn from data with label ambiguity. We presented a number of image classification problems that are naturally cast as a MIL problem. We argued that the term multiple instance learning does not refer to a single well defined scenario, but according to additional assumptions on the problem at hand can result to three different settings.

The use of support vector machines for MIL has been proposed before [And03, Man05] and in this chapter we derived a deterministic annealing algorithm for these formulations. This algorithm consistently finds better local minima of the objective functions. Furthermore we reported results which led to the conclusion that the algorithm of the mi-SVM as presented in [And03] is only applicable for problems with very little ambiguity. The deterministic annealing

algorithm for the instance SVMs comes with the problem of under-estimating the number of positive labeled points. These behaviors render both algorithms inapplicable for a general class of datasets.

The formulation ALP-SVM introduced in this chapter overcomes this problem. This extension of the mi-SVM objective function opens up the possibility to encode prior knowledge about the dataset in a principled way. The deterministic annealing algorithm can be used to optimize the new objective function with similar computational cost as the AL-SVM. This new formulation achieved the best published results on the COREL datasets.

4. Learning the Kernel Function

In the previous chapters we introduced the concept of learning with kernels and presented some of the most prominent kernel learning algorithms. One of the main advantages of these algorithms is that they can be used with any positive definite kernel function. The choice of the kernel largely influences the performance of the algorithm, and it is imperative to choose a suitable kernel for a given learning task. Therefore the following question arises naturally and is central to the use of kernel learning algorithms *For a given task, what is the best kernel function to choose?* This is exactly the question we will discuss in this chapter.

The question about an appropriate kernel is amongst the most important ones for kernel learning algorithms and has consequently received much attention ever since these algorithms have been applied, e.g. [Cha02, Bou03, Cra03, Cri02, Lan04, Gra03]. In this chapter we will review some of the approaches which have been taken and in particular introduce the multiple kernel learning (MKL) framework for supervised learning in which the kernel function is optimized over during the training phase of the algorithm. During training for MKL a linear combination of so-called proposal kernels is sought which yields the final kernel of the SVM classifier.

The main contribution of this chapter is the generalization of MKL to the infinite case. This will allow us to design algorithms which can choose a kernel function automatically from very general classes of kernels, e.g. all Gaussian kernels with positive definite covariance matrices and all convex combinations thereof. There are several advantages of our approach, namely ease-of-use, interpretability, and the possibility to discover structure in the data which may otherwise be unknown to the user. A similar generalization was independently derived in [Arg06, Özö08]. An empirical evaluation of the methods introduced here is presented in Chapter 5.

4.1. Parameter Selection for SVM

There are two ingredients that have to be specified prior to SVM training: the strength of regularization and the function space to search over. In the formulation we have presented in Section 2.3.1 the amount of regularization is controlled by the scalar variable $C \in \mathbb{R}_+$. The second ingredient is the space of

admissible functions $\mathcal{H}$, and as already discussed in Chapter 2, it is implicitly defined through the choice of a positive definite kernel $k(\cdot, \cdot; \theta)$. The problem of parameter selection for support vector machines thus amounts to finding appropriate choices C and θ. Since with θ we define both the type and the parameters of the kernel, e.g. Gaussian and its width, we will speak of the problem of kernel selection and parameter selection interchangeably.

Ideally the parameters θ and C are chosen such that the resulting classification function yields a low generalization error. Since in training we usually are given a finite set of samples, the generalization error has to be estimated and parameters have to be set according to these estimates.

4.1.1. Cross Validation

The technique of *cross validation* (CV) is arguably the most common way to select hyper-parameters in model selection [Koh95]. In N-fold cross validation the available data $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is partitioned into N disjoint sets $\mathcal{S}_1, \ldots, \mathcal{S}_N$ of approximately equal size, also called validation sets. The model is then trained N times, each time using the remaining data $\mathcal{S} \setminus \mathcal{S}_k$ as training data and measuring the performance on the corresponding validation set $\mathcal{S}_k$. The cross validation estimate is the mean of the performances on all validation sets. If each set $\mathcal{S}_k$ is of size n', then the cross validation estimate is an unbiased estimator of the expected performance of the model trained with $n - n'$ training examples. In the limit one can choose $N = n$, in which case one also speaks of the *leave-one-out-estimate*. The collection of predictions on the validation sets are referred to as the *cross validation scores*.

The main problem of the cross validation technique is the necessity of retraining the model N times for each possible parameter. For models which are expensive to train this might already pose a serious problem. This situation becomes even worse if models are parameterized by a number of parameters since exploring combinations is often impractical. The number of training procedures involved scales exponentially with the numbers of parameters to be set. Consider the case of a model with three free parameters. If for each parameter there are ten possible values, one already has to compare 10^3 models, each of which needs to be trained N times in order to estimate its cross validation error.

4.2. Multiple Kernel Learning

In multiple kernel learning one follows a different route to select kernel parameters for SVM classifiers. Two main ideas have lead to this development [Lan04]. The first idea is to parameterize the kernel function as a linear combination of positive definite kernels. The kernel parameters to be chosen are the mixing

coefficients of this linear combination rather than the kernel parameters itself. Secondly, the coefficients of this linear combination are optimized over during the training phase of the algorithm using an extended version of the SVM objective function.

Assume for a given problem one already has available a finite set of kernels

$$k(\cdot, \cdot; \theta), \quad \theta \in \Theta \tag{4.1}$$

to choose from. Instead of selecting exactly one kernel among them using some scoring function like the cross validation estimate, one rather introduces a more general kernel, parameterized in the following way

$$k(\cdot, \cdot; \{d_\theta; \theta \in \Theta\}) = \sum_{\theta \in \Theta} d_\theta k(\cdot, \cdot; \theta), \; d_\theta \in \mathbb{R}. \tag{4.2}$$

It is important to note that not all possible choices of the kernel parameters $\{d_\theta; \theta \in \Theta\}$ yield a positive definite kernel. Restricting the parameter range of d_θ to the positive orthant, $d_\theta \in \mathbb{R}_+$ ensures, according to property 2.1.7, a positive definite kernel. In the remainder of this chapter we will make use of this constraint and therefore only consider positive linear combinations of kernel functions.

One can regard the new kernel of the form (4.2) simply as a differently parameterized one, with the "original" kernel parameters θ being replaced with the mixing coefficients d_θ. The kernel class is also enlarged since all linear combinations of proposal kernels are now included. However one still is confronted with the problem of parameter selection for which one could resort to estimators as discussed in [Cha02] or cross validation over a set of mixing coefficients.

The approach of combining multiple kernels offers the flexibility of combining several heterogeneous data sources in a single framework. In Chapter 6 and Chapter 7 we will show how this property can be exploited for the task of image classification. In image classification each image is represented by a variety of different feature descriptors, each of them characterizing different aspects of the image, like color or texture. With MKL it is possible to combine these aspects in a joint model. Another property accounted to MKL is its *interpretability* [Son06, Var07, Kum07]. After MKL training the final mixing coefficients can be inspected and may reveal properties of the problem at hand.

MKL Objective Function

In the framework of multiple kernel learning the coefficients d_θ are optimized jointly with the other parameters function, namely α and b. This is implemented in the following way. Each kernel in the set Θ corresponds to one

feature space $\mathcal{H}_\theta$ and one feature mapping $\phi_\theta : \mathcal{X} \to \mathcal{H}_\theta$. Thus the hyperplanes in this space are of the form $\langle w_\theta, \phi_\theta(\cdot)\rangle_{\mathcal{H}_\theta}$. The function which is seeked in multiple kernel learning is a linear combination thereof and thus of the form

$$f(x) = \sum_{\theta\in\Theta} d_\theta \langle w_\theta, \phi_\theta(x)\rangle_{\mathcal{H}_\theta} + b, \tag{4.3}$$

where we have added an offset b to the function. According to the representer theorem we know that for minima of a regularized risk formulation there is an equivalent formulation in terms of kernel expansions of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i \sum_{\theta\in\Theta} d_\theta k(x, x_i; \theta) + b. \tag{4.4}$$

So far there is no difference to the ordinary SVM problem, all we did was to re-parameterize the kernel function. The question of how to select the mixing weights d_θ is still to be addressed.

For notational convenience we use the shorthand notation

$$\mathbf{d} = \left(d_{\theta_1}, \ldots, d_{\theta_{|\Theta|}}\right) \tag{4.5}$$

and define the simplex

$$\Delta(\Theta) = \left\{ \mathbf{d} \middle| \sum_{\theta\in\Theta} d_\theta = 1, d_\theta \geq 0, \theta \in \Theta \right\}. \tag{4.6}$$

We pose the multiple kernel learning objective function in the regularized risk framework (2.37). The kernel parameters $\mathbf{d}$ enter the objective function both in the loss and regularization term

$$\min_{\{w_\theta;\theta\in\Theta\},b,\mathbf{d}} \quad \frac{1}{2}\sum_{\theta\in\Theta} d_\theta \|w_\theta\|^2 + C\sum_{i=1}^{n} L\left(y_i, \sum_{\theta\in\Theta} d_\theta \langle w_\theta, \phi_\theta(x_i)\rangle_{\mathcal{H}_\theta} + b\right) \tag{4.7}$$

$$\text{sb.t.} \quad \mathbf{d} \in \Delta(\Theta) \tag{4.8}$$

The only difference to the standard SVM problem is that we choose a different regularization function $\Omega(f) = \frac{1}{2}\sum_{\theta\in\Theta} d_\theta \|w_\theta\|^2$ which weights the influence of the individual functions with their mixing coefficients. This choice of regularization was used in [Zie07] and was also shown to be equivalent with

$$\Omega(f) = \frac{1}{2}\left(\sum_{\theta\in\Theta} d_\theta \|w_\theta\|\right)^2 \tag{4.9}$$

used in [Bac04, Son06].

Due to the product terms between the primal variables d_θ and w_θ this problem is not convex. In [Zie07] it is noted that the simple substitution $v_\theta = d_\theta w_\theta$ (see [Boy04, Section 4.1.3]) yields an equivalent convex minimization problem, provided the loss function L is convex. Therefore efficient convex minimization techniques can be applied to solve it, and global optimality is guaranteed. The resulting problem in primal form reads

$$\min_{\{v_\theta;\theta\in\Theta\},b,\mathbf{d}} \quad \frac{1}{2}\sum_{\theta\in\Theta}\frac{1}{d_\theta}\|v_\theta\|^2 + C\sum_{i=1}^{n} L\left(y_i, \sum_{\theta\in\Theta}\langle v_\theta, \phi_\theta(x_i)\rangle_{\mathcal{H}_\theta} + b\right) \tag{4.10}$$

$$\text{sb.t.} \quad \mathbf{d} \in \Delta(\Theta). \tag{4.11}$$

4.2.1. Non-sparse Multiple Kernel Learning

The formulation introduced so far promotes sparsity of the selected kernels as a consequence of the ℓ_1 constraint of the mixing coefficients. This choice is not crucial to the method and in fact can be replaced using other regularizers for the variables d_θ or just restricting them to yield a positive definite kernel. The latter choice was the original proposal in [Lan04] but has the downside that it turns the problem into a semi-definite program (SDP) for which the best known algorithms scale in the order of $O(n^6)$.

In [Klo08] the simplex constraint is replaced with the unit ball

$$B(\Theta) = \left\{ \mathbf{d} \middle| \sum_{\theta\in\Theta} d_\theta^2 \leq 1, d_\theta \geq 0, \theta \in \Theta \right\} \tag{4.12}$$

which results in a program of the form

$$\min_{\{v_\theta;\theta\in\Theta\},b,\mathbf{d}} \quad \frac{1}{2}\sum_{\theta\in\Theta}\frac{1}{d_\theta}\|v_\theta\|^2 + C\sum_{i=1}^{n} L(y_i, f(x_i)) \tag{4.13}$$

$$\text{sb.t.} \quad \mathbf{d} \in B(\Theta), \tag{4.14}$$

with f as in (4.10). This modification has the following effect. Instead of a sparse solution at the optimum *all* weights d_θ will be non-zero. It can be shown that the program is equivalent if constraint (4.14) is replaced by

$$\mathbf{d} \in \partial B(\Theta), \tag{4.15}$$

that is $\mathbf{d}$ is an element of the boundary of the unit ball only. The latter constraint would render the problem not convex.

Since we constrain the coefficients with the ℓ_2 norm we refer to this formulation as the ℓ_2-MKL and the simplex MKL of the last section as the ℓ_1-MKL. The difference between the ℓ_1-MKL and the ℓ_2-MKL formulation can be understood as two different ways to express prior knowledge about the set of kernels Θ. If we believe that only a few of them are suitable, i.e. the set of kernels

Θ contains a large number of uninformative kernels, the ℓ_1-MKL should be chosen. If on the other hand all of the possible choices of kernel parameters are reasonable one can use the ℓ_2 formulation to include them all but let the algorithm adjust the ratios between them.

4.3. Multiple Kernel Learning Algorithms

In this section we will review two algorithms devised to solve the multiple kernel learning formulation. Both make use of standard SVM solvers as a subroutine.

4.3.1. SimpleMKL

Both the problem (4.10) as well as (4.13) are jointly convex in both $\mathbf{d}$ and the SVM parameters for convex loss functions L. For every possible fixed choice of $\mathbf{d}$ the problem reduces to the standard SVM problem with the kernel defined by this particular mixing coefficients. This suggest a very simple optimization procedure using gradient descent methods [Cha02, Rak08]. Minimization is split into two parts, an inner part which corresponds to the SVM optimization given a fixed choice of $\mathbf{d}$ and an outer minimization of the mixing coefficients $\mathbf{d}$ fixing the SVM parameters. To this end we introduce a function g which corresponds to the optimal SVM objective value for a given parameterization $\mathbf{d}$ of the kernel

$$g(\mathbf{d}) = \min_{\{v_\theta;\theta\in\Theta\},b} \left\{ \frac{1}{2}\sum_{\theta\in\Theta}\frac{1}{d_\theta}\|v_\theta\|^2 + C\sum_{i=1}^{n} L\left(y_i, \sum_{\theta\in\Theta}\langle v_\theta, \phi_\theta(x_i)\rangle_{\mathcal{H}_\theta} + b\right)\right\}. \tag{4.16}$$

One can evaluate g using any SVM solver and the progress in the last years has lead to the development of several efficient SVM solvers [Joa99, Pla99, Cha01]. The remaining step is the minimization of g with respect to the admissible range of $\mathbf{d}$. Thus problem (4.10) (or (4.13))can be equivalently written as

$$\min_{\mathbf{d}} \quad g(\mathbf{d}) \tag{4.17}$$

$$\text{sb.t.} \quad \mathbf{d} \in \Delta(\Theta) \quad (\text{or} \quad \mathbf{d} \in B(\Theta)). \tag{4.18}$$

This can be solved using a gradient descent scheme. Although this includes derivatives of the SVM parameters with respect to the mixing coefficients, it can be shown that these derivatives vanish at the maximal points (α^*, b^*) of g, namely $\frac{\partial\alpha}{\partial d_\theta}(\alpha^*) = 0$ and $\frac{\partial b}{\partial d_\theta}(b^*) = 0$. Therefore the derivative is easily computed, using the expression for v_θ in (B-9), which becomes

$$\frac{\partial g}{\partial d_\theta} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i^*\alpha_j^* y_i y_j k(x_i, x_j; \theta). \tag{4.19}$$

This can be efficiently computed once the variables α, b are known. This gradient descent algorithm is also known as *SimpleMKL* [Rak08]. It is summarized in Algorithm 4.

Algorithm 4 SimpleMKL

Input: Regularization parameter $C > 0$, Kernel set Θ, Training set S
Output: Parameters $\alpha, b, \mathbf{d}$

1: $d_\theta \leftarrow 1/|\Theta|, \forall \theta \in \Theta$
2: **while** Stopping criterion not met **do**
3: $(\alpha, b) \leftarrow$ SVM solution using $k(\cdot,\cdot) = \sum_{\theta\in\Theta} d_\theta k(\cdot,\cdot;\theta)$ (SVM solver)
4: Update $\mathbf{d}$, e.g. using a gradient step
5: **end while**

A variant of SimpleMKL is to use second order information in the gradient descent step [Cha08]. This involves the inversion of a matrix of the size $n_{\text{SV}} \times n_{\text{SV}}$ with n_{SV} being the number of support vectors at the current optimal solution of $g(\mathbf{d})$. This might already be available as a by-product of the SVM algorithm, e.g. [Cha07] but in any case it is not more expensive to compute than the SVM solution itself.

4.3.2. SILP

Another algorithm specifically designed for MKL optimization is the *Semi-Infinite-Linear-Programming* (SILP) approach of [Son06]. It is possible to reformulate the problem (4.10) into

$$\begin{aligned}
\max_{\gamma,\mathbf{d}} \quad & \gamma && (4.20)\\
\text{sb.t.} \quad & \gamma \in \mathbb{R}, && (4.21)\\
& \mathbf{d} \in \Delta(\Theta), && (4.22)\\
& \sum_{\theta\in\Theta} d_\theta S_\theta(\alpha) \geq \gamma, \quad \forall \alpha \in Z, && (4.23)\\
& Z = \left\{ \alpha \in \mathbb{R}^n \middle| 0 \leq \alpha_i \leq C, \sum_{i=1}^n \alpha_i y_i = 0 \right\}, && (4.24)
\end{aligned}$$

where the function S_θ is defined as

$$S_\theta(\alpha) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j k(x_i, x_j; \theta) - \sum_{i=1}^n \alpha_i. \tag{4.25}$$

Note that this is a *linear program* (LP) for γ and $\mathbf{d}$ that are only linearly constrained. There are an infinite number of constraints (4.23) which have to be satisfied at the optimal solution. Each $\alpha \in Z$ corresponds to one such

constraint. This problem can be solved by delayed constraint generation using the following simple alternating scheme. For any finite number of constraints $\alpha^0, \alpha^1, \ldots, \alpha^t$, the problem can be solved for $(\gamma, \mathbf{d})$ using a linear program solver. This finite version is also referred to as the *restricted master problem*. Now the *subproblem* has to be solved: Given the current estimate of $\gamma, \mathbf{d}$ find the maximum violating constraint, namely

$$\alpha^* = \operatorname*{argmax}_{\alpha \in Z} \gamma - \sum_{\theta \in \Theta} d_\theta S_\theta(\alpha). \tag{4.26}$$

Recalling the definition of S this is recognized as solving a SVM with a kernel as in (4.2). In [Son06] a variant of this algorithm is proposed that avoids the optimization of the SVM up to a high precision at intermediate steps and also removes inactive constraints during the course of the computation. In Algorithm 5 we state the simpler algorithm we presented in this section.

Algorithm 5 SILP for MKL

Input: Regularization parameter $C > 0$, Kernel set Θ, Training set S
Output: Parameters $\alpha, b, \mathbf{d}$

1: $d_\theta \leftarrow 1/|\Theta|, \forall \theta \in \Theta$
2: $(\alpha^0, b^0) \leftarrow$ SVM solution with $\mathbf{d}$
3: $t \leftarrow 0$
4: **while** Stopping criterion not met **do**
5: $\quad(\mathbf{d}^t, \gamma^t) \leftarrow$ solution of (4.20) using the constraint set $\{\alpha^0, \ldots, \alpha^t\}$
6: $\quad\alpha^{t+1} \leftarrow$ solution of (4.26)
7: $\quad$**if** $\sum_{\theta \in \Theta} d_\theta^t S_\theta(\alpha^{t+1}) \geq \gamma^t$ **then**
8: $\quad\quad$break
9: $\quad$**end if**
10: $\quad t \leftarrow t + 1$
11: **end while**

The algorithm iterates between solving an SVM (line 5) and a linear program (line 6) until the stopping criterion is met, e.g. decrease of the objective value.

4.4. Infinite Kernel Learning

So far we have considered MKL as the problem of learning the linear combination of a finite number of proposal kernels. In the following we will show that the limitation to optimize only over a *finite* number of proposal kernels is not necessary and that it is possible to optimize over a possibly infinite set of kernels. We will call this formulation *infinite kernel learning* (IKL). The problem remains convex, but the algorithm we devise to solve it involves the maximization of a non-convex problem. We will argue that this should not

be considered a disadvantage compared to MKL. There, the non-convexity of the problem was concealed using a pre-selection of finitely many parameters. Optimizing over a continuously parameterized set of kernels opens up several possibilities and benefits which we will discuss in more details in chapter 5 and 6.

4.4.1. Derivation

Instead of only finite kernel parameter sets Θ, we allow sets of infinite size, e.g. the class of Gaussian kernels that is continuously parameterized by the covariance matrix. To distinguish between these two cases we will use from now on the notation Θ_f whenever the set of parameters is finite and denote with Θ sets of arbitrary size, including finite and infinite sets.

The primal formulation of the MKL problem (4.10) serves as the starting point of our derivation. Given the set Θ of proposal kernels we seek the finite subset $\Theta_f \subset \Theta$ thereof that yields the lowest objective value. We write the following optimization problem

$$\text{(IKL-primal)} \inf_{\Theta_f \subset \Theta} \min_{\mathbf{d}, \{v_\theta; \theta \in \Theta_f\}, b} \sum_{\theta \in \Theta_f} \frac{1}{d_\theta} \|v_\theta\|^2 + C \sum_{i=1}^{n} L\left(y_i, f(x_i)\right) \quad (4.27)$$

$$\text{sb.t.} \quad \mathbf{d} \in \Delta(\Theta_f). \quad (4.28)$$

The infimum can be replaced with a minimum operation, or in other words, the optimal solution of (IKL-primal) is always achieved with a finite number of nonzero d_θ. To see this we dualize the problem.[1] The final problem in dual form is

$$\text{(IKL-dual-1)} \sup_{\Theta_f \subset \Theta} \max_{\alpha, \lambda} \sum_{i=1}^{n} \alpha_i - \lambda \quad (4.29)$$

$$\text{sb.t.} \quad \alpha \in \mathbb{R}^n, \ \lambda \in \mathbb{R}, \quad (4.30)$$

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n, \quad (4.31)$$

$$T(\theta; \alpha) \leq \lambda, \quad \forall \theta \in \Theta_f, \quad (4.32)$$

where we defined

$$T(\theta; \alpha) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j; \theta) \quad (4.33)$$

for easier reference. The variable λ is the Lagrange multiplier of the equality constraint $\sum_{\theta \in \Theta_f} d_\theta = 1$.

We note that if some point (α^*, λ^*) satisfies condition (4.32) for all $\theta \in \Theta$ then it also satisfies the condition for all finite subsets Θ_f thereof. Thus we

[1] The derivation can be found in the Appendix B.

omit the supremum of (IKL-dual-1) and extend the program to the following semi-infinite program

$$\begin{aligned}
\text{(IKL-dual)} \quad & \max_{\alpha,\lambda} \quad \sum_{i=1}^{n} \alpha_i - \lambda && (4.34)\\
& \text{sb.t.} \quad \alpha \in \mathbb{R}^n, && (4.35)\\
& \qquad \lambda \in \mathbb{R}, && (4.36)\\
& \qquad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n, && (4.37)\\
& \qquad T(\theta; \alpha) \leq \lambda, \quad \forall \theta \in \Theta. && (4.38)
\end{aligned}$$

Note that there are as many constraints (4.38) as there are kernel parameters in the set Θ, which might be infinitely many.

To show equivalence between both programs it remains to prove that any optimal point of (IKL-dual) is also optimal for (IKL-dual-1). In other words one can construct a finite set Θ_f for the solution of (IKL-dual) which is also optimal for (IKL-dual-1). This is exactly the statement of the following theorem.

Theorem 4.4.1. ***[Het93, Theorem 4.2]*** *Let $v(P)$ denote the value of program P. If for all $\theta \in \Theta$, Θ compact, and for all $\alpha \in [0, C]^n$ we have $T(\theta; \alpha) < \infty$, then there exists a finite set $\Theta_f \subset \Theta$ for which the optimum of (IKL-dual-1) is taken and v(IKL-dual-1) $= v$(IKL-dual).*

A proof of this theorem is given in [**?**, Theorem 2.1]. In particular the previous theorem states that at the optimum of (4.34) only a finite number of d_θ are non-zero. This ensures that the optimal kernel can be represented with finitely many elements. The final function is thus always of the following form

$$f(x) = b + \sum_{i=1}^{n} y_i \alpha_i \sum_{\theta \in \Theta : d_\theta > 0} d_\theta k(x, x_i; \theta). \tag{4.39}$$

The derivation of the IKL problem including the statement about finiteness of the global solution relies on the constraint $\mathbf{d} \in \Delta(\Theta)$ which enforces sparsity of the mixing coefficients. We presented a non-sparse version of MKL in Section 4.2.1 replacing this constraint with $\mathbf{d} \in B(\Theta)$. Although exchanging the constraints removes the desirable property of a finite global solution it would still be possible to *approximately* solve the resulting problem.

4.5. Infinite Kernel Learning Algorithm

The dual form (4.34) of the problem is a semi-infinite program and suggests a delayed constraint generation approach to solve it. Each kernel parameter of the set Θ defines one linear constraint (4.38) of the problem and thus the

constraint generation turns into a search for the next kernel to include. Starting with a finite constraint set $\Theta_0 \subset \Theta$ ones reiterates between the restricted master problem, that is the search for optimal α, b, λ and the subproblem, a search for violated constraints indexed by θ which are subsequently included in $\Theta_t \subset \Theta_{t+1} \subset \Theta$. This should not be confused with the SILP algorithm, where the constraints are defined by the dual variables α and not by the kernel parameters.

The final algorithm for IKL is summarized in Algorithm 6. We will first discuss the two main ingredients of the algorithm and postpone a statement about its convergence to Section 4.5.3.

Algorithm 6 Infinite Kernel Learning

Input: Regularization parameter $C > 0$, Kernel set Θ, Training set S
Output: Parameters α, b, d_θ

1: Select any $\theta^0 \in \Theta$ and set $\Theta_0 = \{\theta^0\}$
2: $t \leftarrow 0$
3: **loop**
4: $(\alpha, b, d_\theta, \lambda) \leftarrow$ MKL solution with Θ_t {Solve restricted master problem}
5: $\theta^{t+1} \leftarrow \operatorname{argmax}_{\theta \in \Theta} T(\theta; \alpha)$ {Solve subproblem}
6: **if** $T(\theta^{t+1}; \alpha) \leq \lambda$ **then**
7: **break**
8: **end if**
9: $\Theta_{t+1} = \Theta_t \cup \{\theta^{t+1}\}$
10: $t \leftarrow t + 1$
11: **end loop**

4.5.1. The Restricted Master Problem

The restricted master problem in line 4 reduces to a standard MKL problem with the set of constraints as possible kernel parameters. Therefore any MKL algorithm like SILP or SimpleMKL can be used to solve it. Since the parameter λ is the Lagrange multiplier of the equality constraint $\sum_{\theta \in \Theta_t} d_\theta = 1$ (or the equivalent condition of the non-sparse formulation) this parameters is already available as a by-product of the MKL solution.

4.5.2. The Subproblem

Essential to the approach of infinite kernel learning is to solve the subproblem of the problem, which corresponds to finding violated constraints. Therefore we state the problem explicitly.

Problem 1 (IKL-Subproblem)**.** *Given the parameters* $0 \leq \alpha_i \leq C$ *and training points* $\{x_i, y_i\}$, $i = 1, \ldots, n$, *solve*

$$\theta' = \operatorname*{argmax}_{\theta \in \Theta} T(\theta; \alpha) = \operatorname*{argmax}_{\theta \in \Theta} \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j; \theta). \tag{4.40}$$

With $T(\theta; \alpha)$ we have an analytic score which guides the selection of kernels. Note that it is equivalent to the negative gradient of g (4.19).

The subproblem function T can be interpreted as follows. On the one hand there is the outer product of the labels $\mathbf{y}\mathbf{y}^\top$ which we will refer to as the *label matrix*. On the other hand the α re-weighted kernel matrix $\alpha\alpha^\top * K_\theta$, where $*$ denotes elementwise multiplication. We use the following inner product between Gram matrices

$$\langle K_1, K_2 \rangle_F = \sum_{i,j=1}^{m} K_1(x_i, x_j) K_2(x_i, x_j)$$

so we can recognize the subproblem as an *alignment problem* between two matrices

$$T(\theta; \alpha) = \frac{1}{2} \left\langle \mathbf{y}\mathbf{y}^\top, \alpha\alpha^\top * \mathbf{K}_\theta \right\rangle_F . \tag{4.41}$$

There is a high alignment if the α re-weighted Gram matrix is of a similar structure as the label matrix. Since the label matrix can be understood as the optimal Gram matrix, achieving a high alignment with it is desirable. The re-weighting ensures that the alignment is only measured for support vectors, since for all non-support vectors the coefficient $\alpha_i = 0$. This makes intuitive sense: all points which are no support vectors are already classified correctly, adapting the kernel for them will not change the loss function for those points.

There is no general recipe to solve the subproblem since it depends on the kernel class Θ. In particular the problem is not convex and therefore in general a global optimum of the function T is hard to find. Some general approaches on how to solve the subproblem or how to find local maxima are discussed in greater detail in Section 4.5.4.

For the case of a finite set of kernel parameters Θ the subproblem is easy to solve since one can visit each single parameter in turn and check whether the constraint (4.38) is satisfied. Due to this observation it is evident that the IKL algorithm can also be considered as a new algorithm to solve MKL. Especially for the case of very many kernels it is beneficial since only a small working set of kernel matrices needs to be considered at each step, i.e. the restricted master problem is easier to solve due to a fewer number of variables to solve for.

Comparison of MKL and IKL

Let us reconsider how MKL can be understood in this framework. In MKL a finite number of kernels have to be preselected prior to training. Since the set of kernels remains fixed during the course of the algorithm, the MKL performance depends on whether or not discriminative kernels have been chosen by the user. The pre-selection step circumvents the non-convex subproblem and renders the whole MKL formulation to be a convex problem. Although the IKL problem is also convex problem, it requires to solve a non-convex subproblem to solve it.

There are two possible scenarios for kernel combination methods. If the approach of learning a kernel during training is chosen because one is unsure about a good kernel, or one has reason to believe that a combination of kernels is beneficial, the subproblem can be used to exploit structure of the kernel class in a principled way. It provides an analytic measure which is amendable to optimization. For some problems at hand a finite set of kernels could be chosen on purpose in order to express prior knowledge. In this case the selection of kernels acts as a regularization of the function space.

We will demonstrate in the experimental validation presented in Chapter 5 that in cases where linear combination of kernels are beneficial for classification, the MKL solution is always outperformed by IKL.

Related Work

Maximizing the alignment between the Gram and the label matrix has been proposed before, usually as a pre-processing step for kernel machines. In [Cri02] a normalized version of the above alignment was introduced as *Kernel Target Alignment* (KTA). In our notation it reads

$$\text{(KTA)} \quad A_{\text{KTA}}(K_\theta, \mathbf{y}\mathbf{y}^\top) = \frac{\langle K_\theta, \mathbf{y}\mathbf{y}^\top \rangle_F}{n\sqrt{\langle K_\theta, K_\theta \rangle_F}}. \tag{4.42}$$

In [Cri02] a concentration inequality for this sample-based alignment is presented, stating that it is not too dependent on the training set S. Furthermore it is possible to upper bound the generalization error of a Parzen window classifier in terms of KTA. It is concluded that for high KTA one may expect good generalizations and some empirical evidence for this is presented.

The KTA measures the cosine of the angle between the two bi-dimensional vectors $\mathbf{y}\mathbf{y}^\top$ and K_θ and is thus invariant to a rescaling of the kernel function. This suggests to normalize the kernels by restricting the class to

$$k'(\cdot, \cdot, ; \theta) = \frac{k(\cdot, \cdot; \theta)}{\sqrt{\langle K_\theta, K_\theta \rangle_F}}. \tag{4.43}$$

With this substitution, the subproblem turns into KTA possibly at the expense of a more difficult subproblem. Note that $\sqrt{\langle K_\theta, K_\theta \rangle_F}$ is simply a scalar

depending on θ and the training data, and thus the normalized kernel is again a kernel.

Other work on kernel alignments include [Bar05] where *Kernel Polarization* (KP) is proposed which simply omits the normalization term of KTA $A_{\text{KP}}(S, K_\theta, \mathbf{y}\mathbf{y}^\top) = n^{-2}\langle K_\theta, \mathbf{y}\mathbf{y}^\top\rangle_F$. This is equivalent to the subproblem for constant $\alpha = 1/n$.

An unnormalized version of the alignment was also used in [Cra03] to adapt a kernel matrix by explicitly maximizing the alignment in a boosting scheme. The kernels are chosen using the inner product as above, the $\alpha_i\alpha_j$ terms turn into a weight corresponding to the difficulty predicting whether the points x_i and x_j have the same label or not. They restrict the kernel set to outer products and show that solving the weighted alignment turns into a generalized eigenvector problem.

4.5.3. Convergence

We can make the following statement, on the convergence of Algorithm 6 which depends on whether the subproblem can be solved for global optima.

Theorem 4.5.1. ***[Het93, Theorem 7.2]*** *If the subproblem can be solved, Algorithm 6 either stops after a finite number of iterations, or has at least one point of accumulation and each one of these points solve (IKL-dual).*

Proof. See Appendix C. □

Although Theorem 4.4.1 guarantees the existence of a finite subset of constraints Θ_f, binding at the optimum of (IKL-dual), there is no guarantee that Algorithm 6 will identify this set. But if the algorithm terminates after a finite number of steps (and the subproblem can be solved) it reached *global* optimality. Each strictly violated constraint with $\theta \in \Theta$ corresponds to a primal descend direction in d_θ. Therefore, if θ is added to the finite set of kernels and the problem is resolved with this new constraint included, the objective function is guaranteed to decrease. To ensure global optimality at some solution one has to guarantee that no further descend direction exists. This is equivalent to be able to solve the subproblem.

An important observation is that all intermediate solutions of the algorithm are primal feasible. Therefore even if the subproblem can not be solved, i.e. only local maxima of the function T can be found with no guarantee of global optimality, the algorithm always produces a valid classification function.

For the ℓ_2-MKL variant no finite optimal solution exists. However the same argument as mentioned above still holds. At each iteration the inclusion of kernels $\theta \in \Theta$ will decrease the objective function. Instead of having to pre-select a finite subset of kernels beforehand one can instead search for the steepest descent direction using the subproblem and impose a termination criterion, e.g. decrease of the objective function. This will guarantee a better or equal

solution in terms of the objective value, than if the kernel set is restricted to a finite subset of kernels $\Theta_f \subset \Theta$.

4.5.4. Solving the subproblem

The IKL algorithm is suited for any class of parameterized kernel families for which a way to solve the subproblem or at least local minima can be found. In the following we discuss several optimization strategies which could be pursued to tackle this problem. In Section 5.1.1 the following optimization techniques are applied to the class of Gaussian kernels.

Gradient Ascent

For cases of continuously parameterized kernel functions which are differentiable with respect to their parameter a simple method to solve for local maxima are gradient ascent techniques. In each iteration of the IKL algorithm a search may be initialized in multiple points, and a gradient ascent search is started. This will result in a number of local maxima from which the highest is returned. There is no guarantee for finding the global optimum.

Branch-And-Bound

The Branch-And-Bound algorithm is a general technique to find optimal solutions of optimization problems. It involves the following two steps:

1. partition the space of parameters recursively (branching step);

2. make statements that hold for an entire partition (bounding step).

We want to find maxima of the function $T(\cdot;\alpha)$ on the set Θ. Therefore we need a partition operation that splits a given set $\Theta_0 \subseteq \Theta$ into a number of partitions $\Theta_0^1 \cup \ldots \cup \Theta_0^m = \Theta_0$. This depends on the parameterization of the problem, e.g. splitting intervals of parameters.

The second ingredient is a bounding function that upper bounds the value of the true function T on a partition $\Theta_0 \subseteq \Theta$. This upper bound $\widehat{T}$ must fulfill

$$\max_{\theta \in \Theta_0} T(\theta) \leq \widehat{T}(\Theta_0), \quad \forall \Theta_0 \subseteq \Theta. \tag{4.44}$$

If for some $\theta^* \in \Theta \setminus \Theta_0$ we have $\widehat{T}(\Theta_0) \leq T(\theta^*)$, the whole partition Θ_0 can be discarded from the search space. Otherwise Θ_0 is partitioned again and recursively searched over.

The efficiency of Branch-And-Bound depends on the tightness of the bound $\widehat{T}$. Loose bounds incur many splits and thus require many iterations of the algorithm.

Global Optimization

In general one can borrow techniques from the field of Global Optimization [Hor96]. Besides the aforementioned strategies methods like homotopy methods [Ros98], Difference of Convex functions (DC) [Yui02, An05] or Lipschitz optimization are possible candidates. The latter requires Lipschitz-continuous kernels, those which satisfy

$$|k(x,x';\theta_1) - k(x,x';\theta_2)| \leq L|\theta_1 - \theta_2|, \quad \forall x,x' \in \mathcal{X} \tag{4.45}$$

for some constant $L > 0$. Due to efficiency reasons current Lipschitz solvers can be applied to problems up to approximately 20 variables only.

4.5.5. Implementation Details

In practice we make several modifications to Algorithm 6.

Multiple pricing: At each iteration of the algorithm the subproblem solver returns a set of violating constraints instead only the most active to speed up convergence. In the experiments up to five violating constraints are seeked.

Working set: Coefficients d_θ which became zero are pruned out. For the final solution the corresponding constraints are inserted again in the problem to check whether they remain inactive. Other working set methods could be employed.

Warm starts: Both the restricted master problem as well as the subproblem calls are initialized at the previously solution which speeds up their convergence.
An implementation of the algorithm using a combination of Coin-IPopt-3.3.5 [Wäc06] and the libSVM package [Cha01] is available under the GPL licence at `http://www.mloss.org/software/view/174/`. This also includes a MKL algorithm and the implementation of several subproblem solvers for differentiable kernel classes.

4.6. Conclusion

Learning or estimating a suitable kernel from empirical data is one of the biggest challenges in the field of kernel learning algorithms. It is among those which have considerable impact on real world problems and enhance the usability of kernel learning algorithms especially to users which are not experts in this field. In this chapter we have reported on some progress in this direction.

In particular we focused on the approach to linear combination of kernels. This problem was originally posed as multiple kernel learning problem [Lan04]

although earlier approaches with similar scope exists [Cha02]. The main contribution reported on here is the generalization of the MKL approach to its infinite limit. This generalization is direct in the sense that there is no disadvantage to MKL.

With the infinite kernel learning framework it is possible to optimize over large classes of kernels without need for pre-selection. In particular the structure of kernel classes can be exploited to efficiently traverse the space of kernels which enables a principled way for kernel selection. This can be exploited in two ways. On the one hand it may lead to algorithms which are more easy to use. For practitioners the choice of the kernel function is sometimes regarded as a burden rather than a degree of freedom. On the other hand it allows interpretability of the resulting classification function. For MKL one can only discover structure within the data one has already encoded in the pre-selected kernels. Using IKL with much enriched kernel classes it is possible to discover structure beyond this selection.

5. Empirical Evaluation of Kernel Combination Methods

In the last chapter we presented the multiple kernel learning formulation and generalized it to the case of linearly combining an infinite number of kernels. In this chapter we concentrate on the experimental comparison of these approaches. The goal is twofold. With a first set of experiments on artificial toy data we will provide an intuition about the IKL algorithm. A second set of experiments is conducted using a variety of standard machine learning benchmark datasets that have been used to compare different classification methods [Rät01]. In particular there are two questions we try to answer:

- Does the increased flexibility of linearly combining kernels instead of choosing only a single one lead to overfitting?
- Both Cross Validation as well as MKL/IKL can be understood as estimators for the kernel hyper-parameters. How do they compare?

In Section 5.2 we discuss the IKL algorithm in more detail by means of simple toy datasets. The main experimental part is then presented in section 5.3 after which we conclude with a Discussion. Before we start we will introduce the kernel classes which are used throughout the experiments presented in this Chapter.

5.1. Kernel Classes: Gaussian Kernels

We build on the Gaussian kernel described in Section 2.1.6. In total we use three different variants, which differ in the number of parameters to set. Let us write the kernel function as

$$k(x, x'; \{\sigma_d\}_{d=1}^{D}) = \exp\left(-\sum_{d=1}^{D} \sigma_d (x_d - x'_d)^2\right), \tag{5.1}$$

where $\sigma_d \geq 0$ is the bandwidth for the d'th dimension of the data. For the first class we restrict all bandwidths to be identical, i.e. $\sigma_1 = \sigma_2 = \ldots = \sigma_D$ and refer to this class as (single). This is a standard choice for building SVM classifiers which treats each feature dimension as being equally important. The second kernel class consists of those Gaussian kernels which ignore all but one

feature dimension of the data. In this case we set $\sigma_d > 0$ and $\sigma_{d' \neq d} = 0$. We use the name (separate) to refer to this class of kernels. The last class is the most general one and includes the two previous classes as special cases. The class (product) consists of all kernels with non-negative diagonal covariance matrix, i.e. $\sigma_d \geq 0, d = 1, \ldots, D$.

There is a varying number of free parameters for the three classes. For (single) only one free parameter has to be set, for (separate) one has two parameters (index d and value of σ_d) while for (product) D parameters need to be specified. This turns the subproblem of the IKL algorithm into a D dimensional non-convex optimization problem. However the class (product) is too large to compute Cross Validation estimates for a reasonable number of parameter choices and thus it is crucial to exploit the structure of this dataset in order to efficiently search over it.

5.1.1. Solving the subproblem

To solve the subproblem we use i) for one dimensional subproblem functions the branch-and-bound algorithm and ii) for all other subproblems a Newton method. The concepts of both algorithms are introduced in 4.5.4, here we will described their application to the special case of Gaussian kernels.

Gradient Ascent

We employ the Newton method for maximizing the subproblem function. The derivative of the subproblem w.r.t. to the kernel parameters is easily calculated to be

$$\frac{\partial T}{\partial \sigma_d} = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (x_{id} - x_{jd})^2 \alpha_i \alpha_j y_i y_j k(x_i, x_j; \theta), \tag{5.2}$$

where x_{id} denotes the d'the component of the instance x_i. The calculation of the second derivative is analog.

We initialize the search in multiple starting points and perform a gradient ascent. During the course of the algorithm all previously identified violating constraints are added to the set of starting points. This worked very well in practice and is used the IKL-experiments with a more than one dimensional subproblem.

Branch-And-Bound

For the Branch-And-Bound algorithm 4.5.4 we need to the specify an upper bound $\widehat{T}$ on the subproblem function together with a strategy for partitioning the sets of parameters.

For the case of Gaussian kernels the set of kernel parameters to search over is some interval $\sigma \in [\sigma_{\min}, \sigma_{\max}]$. At each step of the algorithm we partition a

set $[\underline{\theta}, \bar{\theta}]$ into two sets $[\underline{\theta}, \theta']$ and $[\theta', \bar{\theta}]$, where $\theta' \in [\underline{\theta}, \bar{\theta}]$. The multi-dimensional search is analogous, there a partitioning is achieved by splitting only one dimension.

We define $\underline{\sigma_d}(\Theta), \overline{\sigma_d}(\Theta)$ to be the minimum (resp. maximum) value for the d'th component in the set Θ

$$\underline{\sigma_d}(\Theta) = \operatorname*{argmin}_{\sigma \in \Theta} \sigma_d \tag{5.3}$$

$$\overline{\sigma_d}(\Theta) = \operatorname*{argmax}_{\sigma \in \Theta} \sigma_d, \tag{5.4}$$

and the upper bound $\widehat{T}$

$$\widehat{T}(\theta; \alpha) \stackrel{\text{def}}{=} \sum_{y_i = y_j} \alpha_i \alpha_j \exp\left(- \sum_{d=1}^{D} \underline{\sigma_d}(\Theta)(x_{id} - x_{jd})^2 \right) \tag{5.5}$$

$$- \sum_{y_i \neq y_j} \alpha_i \alpha_j \exp\left(- \sum_{d=1}^{D} \overline{\sigma_d}(\Theta)(x_{id} - x_{jd})^2 \right) \tag{5.6}$$

The subproblem function can be bounded from above fullfilling the requirement (4.44)

$$T(\theta; \alpha) \leq \widehat{T}(\Theta, \alpha). \tag{5.7}$$

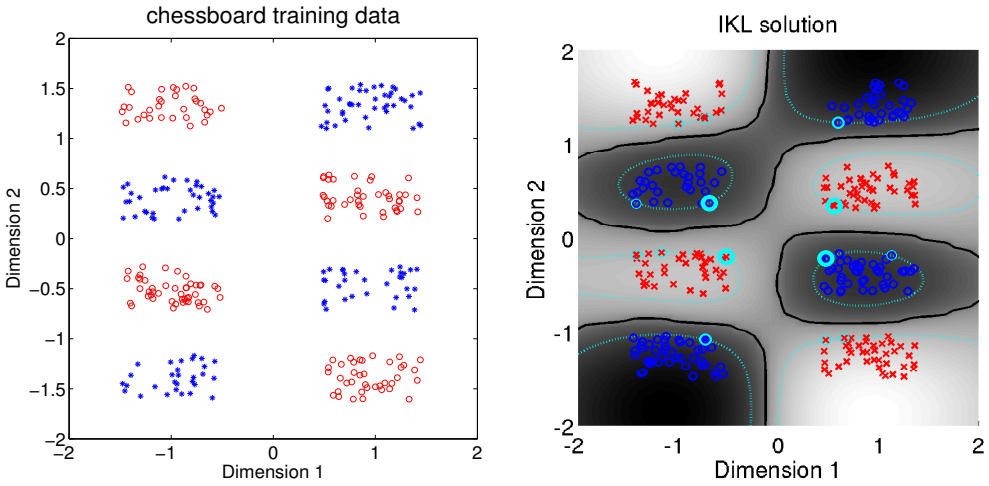
This bound turns out to be very loose resulting in a time consuming search and hence branch-and-bound could only be used for one dimensional subproblems only. The advantage is that it guarantees global optimality of the solution.

5.2. Toy Examples

All experiments in this section are conducted on artificially created data with the purpose of: 1. highlighting the benefit of learning with continuously parameterized classes of kernels and 2. illustrate the execution of the IKL algorithm.

5.2.1. Chessboard Data

We begin with a binary classification example. The kernel class used for this experiment are the Gaussian kernels of class (product) described in Section 5.1. A total of 300 training points are sampled from a two dimensional chessboard pattern as shown in Figure 5.1(a). Eighteen noise dimensions sampled from a normal distribution are added to form a 20 dimensional feature vector. Both classes are equally likely and are indicated by different color and marker in the figure. This data is normalized to zero mean and unit variance.



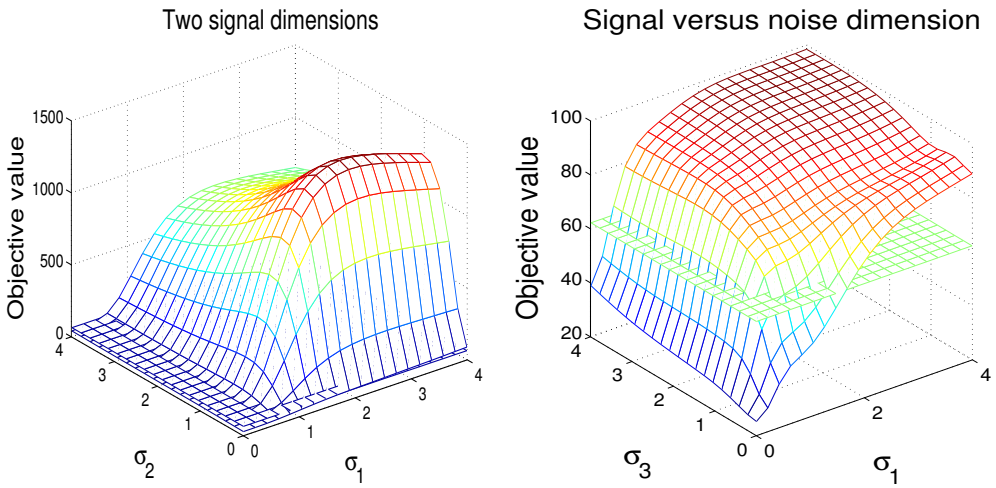
(a) (b)

Figure 5.1.: Chessboard toy example, (a) distribution of training data in the first two dimensions, (b) resulting classification function of the IKL classifier. The IKL algorithm chooses chooses only two kernels (5.8) and (5.9) which correctly model the data while ignoring the noise dimensions.

Note that the chessboard pattern was created with two columns and four rows such that the optimal bandwidths of a Gaussian kernel are different for these two dimension. For the remaining 18 noise dimensions the ideal bandwidths are zero since there is no information contained in them. The class (product) is flexible enough to model all feature dimensions separately.

Now the IKL algorithm is started using some initial kernel which is not the "correct" one. In Figure 5.2 two slices of the subproblem function T are shown. In (a) the scaling parameters of the two signal dimensions σ_1 and σ_2 are plotted against the objective value and in (b) σ_1 is plotted against a scaling factor of a noise dimension σ_3. In both cases we set all other scaling factors to zero. The hyperplane associated with the Lagrange multiplier $\lambda \approx 62$ is also shown in the plot. All parameters with function values above this hyperplane are violating constraints and thus their inclusion will decrease the objective. The plot of Figure 5.2(a) includes the "ideal" set of parameters for the problem and indeed the objective function takes a maximum at this point. Note that in Figure 5.2(b) the scale on the z-axis is different from the one in plot (a), the subproblem function T takes much lower values and thus the decrease of the IKL objective function is also lower if those constraints are included. It can also be seen in (b) that the objective function is much more dependent on the value of σ_1 than it is on the scaling factor σ_3.

In the course of the IKL algorithm the signal dimensions are identified automatically and the following two kernels are selected which achieve a perfect



(a) (b)

Figure 5.2.: Two projections of the 20 dimensional IKL subproblem function for the chessboard toy example. (a) IKL subproblem in the first iteration for the two signal dimensions (σ_1 and σ_2), (b) for one signal σ_1 and one noise σ_3 dimension. The hyperplane corresponding to the Lagrange multiplier $\lambda \approx 62$ is also shown (note a different scaling of the two plots). Inclusion of kernels with parameters with an objective value higher than this value will decrease the IKL objective function. Any subproblem solver should find local maxima of this function.

accuracy on a held out test set

$$d_{\theta_1} = 0.98, \quad \theta_1 = (1.1, 3.2, 0, 0, 0, 0, 0, 0, 0, 0, 0, \ldots, 0) \tag{5.8}$$

$$d_{\theta_2} = 0.02, \quad \theta_2 = (0, 0, 0, 0, 0, 0, 0, 2.1, 0, 0.2, 0, \ldots, 0). \tag{5.9}$$

The resulting decision boundary for the first two dimensions is plotted in Figure 5.1 (b). Support vectors are marked with cyan rings in this plot. The first kernel is modeling only the two signal dimensions while ignoring all noise dimensions. Scaling factors are such that they are as large as possible since those correspond to a smooth function while not making training errors.

To achieve the same result with an SVM or MKL learner one would have to cross-validate over different scaling factors for each dimension or guess the correct scalings for the proposal kernels. Cross validation over the same kernel class (product) is prohibitive due to the large number of kernels parameters.

5.3. Standard ML benchmark datasets

Up to now and to the best of our knowledge there is no extensive comparison between properly tuned SVM classifiers and those which linearly combine multiple kernels. In this section we will fill this gap. Furthermore we want to investigate whether IKL as the limit of MKL improves the performance, leads to overfitting or gives qualitatively the same results. In total we compare three different methods

1. SVM using kernel class (single),
2. MKL using kernel classes (single) and (separate),
3. IKL using kernel classes (single) and (product).

5.3.1. Experimental Setup

Thirteen different binary datasets with up to 100 independent splits (the datasets "image" and "splice" are split only 20 times) and the very same experimental setup from [Rät01] are used. With five fold CV on the first five splits the hyper-parameters of the methods are determined and subsequently applied to obtain the results on all splits. Parameters to be set with are the regularizer C for all methods and the kernel $k(\cdot,\cdot;\theta)$ for SVM. The following parameters are considered for all thirteen datasets $\sigma^{-1} \in \{1,2,3,5,10,20,30,40,50,75,100,125,150\}$. The regularization constant is selected from the values $C \in \{10^{-2}, 10^{-1}, \ldots, 10^{3}\}$.

Five more multiclass datasets were taken from [Dua05] which are split 20 times into training and test data. On each split we use five fold CV to determine the hyper-parameters which are the used to learn a final classifier. Multiclass decisions are resolved in a one-versus-rest scheme. The tested kernel parameters are $\sigma^{-1} \in \{0.5,1,2,5,7,10,12,15,17,20\}$ and the regularization constant is chosen among $C \in \{10^{-2}, 10^{-1}, \ldots, 10^{4}\}$. All data was scaled to have zero mean and unit variance.

The amount of free parameters was varied using the three different classes as described in Section 5.1. For (single) we compared all three methods: SVM, MKL and IKL. For SVM one single kernel is selected using the Cross Validation estimate. For MKL we constructed as proposal kernels all kernel matrices with the kernel parameter in the range as described above. For IKL we allowed for all kernels with $\sigma \in [0,30]$. This range includes the finite choices of kernel parameters but we assured that restricting the admissible range to $\sigma^{-1} \in [1,150]$ (resp. $\sigma^{-1} \in [0.5,20]$) does not change the results.

For the class (separate) all (single) kernels are used and additionally we applied the same range of kernel parameters to every dimension separately.

This yields a total of $13(D+1)$ proposal kernels for a D dimensional dataset. Finally for (products) we allow all kernels with parameters in $\sigma_d \in [0, 30]$. The corresponding subproblem is D dimensional.

As a subproblem solver for the IKL algorithm we used the Branch-And-Bound technique for class (single) to assure a global optima (up to a precision of 10^{-3}). For the other two classes we used the MKL parameters as starting points for a gradient ascent search in each iteration. For these two classes a global optimum can not be guaranteed. Therefore we used the following two stopping criteria. Either no violating constraint was found during the subproblem phase or the change of improvement in terms of the relative change of the objective function falls below 0.01%.

5.3.2. Results

he results are shown in Table 5.2 and Table 5.3. The four missing values in Table 5.3 correspond to experiments which were computationally too expensive to compute. Even for the results reported here several millions of SVM trainings were performed.

5.3.3. Discussion

We can draw several conclusions from the results. Comparing only the three results obtained with the class (single) we see that the SVM almost always yields to better results than MKL or IKL. We test the difference of the methods with a paired t-test and find that only on the datasets Ringnorm, Titanic and Waveform the SVM method is outperformed by IKL.The added flexibility of MKL and IKL to combine more than one kernel seems to be of little use with the possible exception of ABE (Table 5.3). For this kernel class the parameter space is sampled densely enough such that the best kernel parameter for the SVM is well enough approximated. The results indicate that the Cross Validation estimate is a more reliable estimator to select kernel parameter than the objective problems of MKL or IKL.

Adding the flexibility to model each dimension separately using class (separate) yields better results only on the datasets Splice, SEG and ABE.

The most general setting (products) results in some impressive gains in performance for Image, Splice and SEG, even improving over the MKL-(separate) results. In these cases the possibility to model correlations between different dimensions explicitly yields more discriminative kernels. For the remaining datasets there either is no discriminative structure in subsets of the feature dimensions or the IKL subproblem solver was unable to identify those.

For some datasets we observe worse results which are possibly due to overfitting behavior: Heart, Ringnorm, Twonorm and WAV.

For the practitioner there are thus two methods to choose from: SVM because it is much faster to train than the other two methods and shows good performance, or IKL because the enlarged kernel class might lead to a significant performance increase. For all those cases where linear combinations of kernels improved the performance IKL outperformed the MKL results using the enlarged kernel class (product).

Dataset	#dim	#train	#test
Banana	2	400	4900
Breast-cancer	9	200	77
Diabetes	8	468	300
Flare-Solar	9	666	400
German	20	700	300
Heart	13	170	100
Image	18	130	1010
Ringnorm	20	400	7000
Splice	60	1000	2175
Thyroid	5	140	75
Titanic	3	150	2051
Twonorm	20	400	7000
Waveform	21	400	4600

Table 5.1.: Statistics of the benchmark classification datasets used for the experiments. Given are the name of the datasets, the dimensionality (#dim) of the instances and the number of training (#train) as well as test (#test) examples.

5.4. Kernel Classes

In this section we briefly mention some other interesting kernel classes which can be used in the IKL framework.

Polynomial kernels (c.f. Section 2.1.6) are commonly used in kernel classifiers and can be equipped with a weighting factor $1/\sigma_d$ for each feature dimension of the data

$$k(x, x'; \{\sigma_d\}_{d=1}^{D}, p) = \left(1 + \sum_{d=1}^{D} \frac{x_d x'_d}{\sigma_d^2}\right)^p . \tag{5.10}$$

As for the Gaussian kernels a gradient ascent method can be used to solve the corresponding D dimensional subproblem for σ. The degree parameter would have to be searched over in a separate step.

Dimensionality reduction or whitening vectorial data using a linear transformation A such as the commonly used Principal Component Analysis (PCA) can be turned into a kernel parameter for any kernel k by using proposal kernels of the form $k(x, x'; A) = k(Ax, Ax')$. In the case the kernel k is differentiable with respect to the parameter A we can use gradient based methods to solve the subproblem. To choose a number of sensible kernel parameters from this set might pose a difficult problem to the designer of the kernels. In those cases it is imperative to have an analytic score to search over this space efficiently.

Product Kernels provide a very flexible class of kernels. We consider those of the form

$$k(x, x'; \gamma_1, \dots \gamma_K) = \prod_{k=1}^{K} \exp\left(-\gamma_k d_k(x, x')\right), \tag{5.11}$$

with some distance functions $d_k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$. The Gaussian kernel used for the previous experiments is a member of this class with d_k being the Euclidean distance. By setting $\gamma_k = 0$ the distance or features d_k can be ignored altogether. A kernel of this type recently won the PASCAL VOC 2007 classification challenge [Eve07].

A blend between the χ^2 and Gauss kernel which are commonly used kernels for image classification methods. With the following parameterization we can "blend" between them

$$k(x, x'; q) = \exp\left(-\sum_{i=1}^{d} (x_i + x_i')^q (x_i - x_i')^2\right). \tag{5.12}$$

The choice $q = -1$ corresponds to the χ^2 kernel and $q = 0$ to the Gaussian kernel. Note that the search for q is only 1-dimensional and the kernel function is differentiable with respect to q.

5.5. Conclusion

The experiments show that only limited performance gains can be expected using a linear combination of different kernel functions on standard benchmark datasets. In total we found that for four out of 19 different datasets the classification performance improves if the kernel classes is extended and a linear combination is sought using the IKL/MKL framework.

For the four datasets where a linear combination improves performance the IKL algorithm found better solutions than the MKL approach. This is not surprising since the kernel classes available to the IKL algorithm are much larger and thus more likely to contain well adapted kernels for the problem. Because in the IKL algorithms the kernels are chosen during the training phase

of the algorithm and not prior to training, the structure of the kernel class can be exploited.

For most datasets we found with both the multiple and infinite kernel learning framework it is possible to automatically select among a large class of kernel functions those which achieve a good performance. We believe that such approaches can be used to assist designers of classification functions in selecting appropriate kernel parameters. This can be used to reduce the amount of prior knowledge needed to design good classification functions and especially inexperienced users may benefit from this feature.

In the next chapter we turn our attention to the special case of classifying image data and propose kernel classes which model the special structure of images. The kernel learning algorithms will be used to select appropriate kernels among these classes.

	(single)					(separate)		(products)	
	SVM	MKL		IKL		MKL		IKL	
Dataset	err	err	#k	err	#k	err	#k	err	#k
Banana	**10.5 ± 0.5**	**10.5 ± 0.5**	1.0	**10.6 ± 0.5**	6.6	**10.5 ± 0.5**	1.0	10.7 ± 0.5	3.7
Breast-cancer	**25.9 ± 4.3**	27.9 ± 4.0	2.3	26.9 ± 4.7	2.2	26.7 ± 4.2	4.5	**25.7 ± 4.1**	16.1
Diabetis	**23.2 ± 1.6**	24.2 ± 1.9	2.8	23.9 ± 1.6	3.1	24.5 ± 1.6	4.0	24.3 ± 1.8	22.3
Flare-Solar	**32.4 ± 1.7**	35.1 ± 1.7	1.9	34.9 ± 1.8	3.3	34.3 ± 2.1	2.9	**32.8 ± 1.9**	2.6
German	**23.7 ± 2.1**	25.3 ± 2.3	2.0	25.1 ± 2.5	3.2	25.1 ± 2.2	8.3	24.6 ± 2.4	46.1
Heart	**15.2 ± 3.1**	16.4 ± 3.3	1.0	17.0 ± 3.2	2.7	16.7 ± 4.1	9.0	20.1 ± 3.6	28.2
Image	3.0 ± 0.6	3.3 ± 0.7	1.0	3.5 ± 0.6	6.2	3.0 ± 0.6	1.6	**1.4 ± 0.3**	27.1
Ringnorm	1.6 ± 0.1	1.6 ± 0.1	1.0	**1.5 ± 0.1**	5.3	1.7 ± 0.1	2.6	2.1 ± 0.2	16.3
Splice	10.6 ± 0.7	11.1 ± 0.7	2.0	11.0 ± 0.8	2.4	6.0 ± 0.4	24.1	**3.1 ± 0.3**	72.8
Thyroid	**4.0 ± 2.2**	4.7 ± 2.1	1.0	**3.5 ± 2.1**	3.2	4.7 ± 2.1	1.0	4.1 ± 2.0	12.7
Titanic	22.9 ± 1.2	**22.4 ± 1.0**	1.1	**22.5 ± 1.1**	3.8	22.4 ± 1.0	1.9	**22.4 ± 1.1**	5.2
Twonorm	**2.5 ± 0.1**	**2.5 ± 0.1**	2.0	**2.5 ± 0.2**	1.3	**2.5 ± 0.1**	3.8	3.8 ± 0.4	36.2
Waveform	10.1 ± 0.5	**9.9 ± 0.4**	2.9	**9.8 ± 0.4**	3.0	10.2 ± 0.4	9.7	11.4 ± 0.6	33.7

Table 5.2.: Test error (err) and number of selected kernels (#k) on several two class datasets averaged over 100 (20) runs. In bold face are those results which are not statistically indistinguishable from the best result using a paired t-test.

	statistics				**(single)**					**(separate)**		**(products)**	
					SVM	MKL		IKL		MKL		IKL	
Dataset	#dim	#train	#test	#cl	err	err	#k	err	#k	err	#k	err	#k
WAV	21	300	4700	3	15.6 ± 1.2	15.5 ± 0.6	2.7	15.8 ± 0.7	2.1	16.4 ± 1.7	13.6	18.0 ± 1.0	35.1
SEG	17	500	1810	7	6.5 ± 1.0	6.8 ± 0.9	2.8	6.9 ± 0.9	3.7	5.0 ± 0.7	8.4	3.0 ± 0.5	18.0
ABE	16	560	1763	3	1.1 ± 0.3	0.8 ± 0.3	2.5	0.8 ± 0.3	3.0	0.7 ± 0.3	11.3	0.7 ± 0.2	33.8
SAT	36	1500	4935	6	10.4 ± 0.4	10.2 ± 0.3	3.6	10.1 ± 0.4	4.0	n/a		n/a	
DNA	181	500	2686	3	7.7 ± 0.7	7.8 ± 0.7	1.4	7.7 ± 0.8	2.0	n/a		n/a	

Table 5.3.: Statistics and error rates for the five datasets from [Dua05]. The results are averaged over 20 independent splits of the available data. Also plotted are the average number of selected kernels (#k). In the fifth column the number of classes (#cl) of the dataset is given. Multiclass decisions are resolved in a one-versus-rest scheme.

Part II.

Image Classification

6. Optimizing Pre-processing Steps via Kernel Learning

In the previous chapters we have introduced kernel classifiers and developed ways to perform automatic kernel selection. In the remainder of this thesis we will discuss how kernel methods can be used for classifying images. We consider image classification as a standard multiclass classification problem, for which the training data are pairs of images and corresponding labels.

The possibility to formalize a notion of similarity between complicated objects, such as images, through the construction of a kernel function has received much attention in the field of image classification and consequently led to the design of specifically tailored kernel functions, e.g. [Rub00, Laz06, Gra07, Ved08]. In this chapter we show how kernel design for standard computer vision problems can be done with the kernel learning approach introduced in chapter 4.

The typical procedure of building a classification function for computer vision systems from data can be split into two parts: the pre-processing pipeline and the search for the classification function itself.

A pre-processing pipeline may include many different steps and each single one influences the overall classification function. For visual object classification there are a number of typical pre-processing steps. These include the choice of type and number of features to be extracted from each image, its transformation to aggregated representations such as histograms and possible normalizations of the latter. The main problem is that the effect of a single choice somewhere along the pipeline on the final classification function is a priori unclear. Most often the only way to quantify its effect is to compare all possible choices.

This is sub-optimal however. Ideally one would like to place a suitable prior over parameter choices and allowing the learning algorithm to freely optimize the quality of the classification function over *all* parameter settings.

To get closer to this goal, we propose to apply the following simple paradigm when building classification systems. Instead of choosing parameters of pre-processing steps beforehand we turn them into parameters of the kernel function and thus make them available to the kernel learning algorithm. In particular we follow the same line of thinking which was proposed in the previous chapters. Instead of using only one parameterizable kernel, we use general

classes of kernels from which the best kernel is chosen. This is implemented using the technique of MKL with sparsity enforcing priors over the weights (Section 4.2) as well as its non-sparse counterpart (Section 4.2.1).

Pre-processing choices are included into the kernel design and thus are available to the objective function. Optimizing kernel parameters corresponds to optimizing the pre-processing parameters. This allows a principled selection of the various parameters involved in the system.

We will focus on two important problems reoccurring in many computer vision systems and demonstrate how to apply the approach:

1. How to compare various kinds of local features extracted from image patches?

2. How to take into account spatial relationship of image features?

Experiments are presented for both scenarios, learning the optimal codebook for a bag-of-visual-words representation in Section 6.1 and learning the spatial configuration of a spatial kernel function in Section 6.2. We show that the learning-based system consistently outperforms previous approaches on a number of benchmark datasets.

6.1. Learning a Codebook of Visual Words

Finding a discriminative image representation is the key challenge for most image classification tasks. A common approach is to represent an image as a collection of local feature descriptors [Mik05a], evaluated at some keypoints of the image [Now06, Mik05b]. This is motivated through the intuition that images depicting the same object or scene do share certain localized parts with similar statistics. Such a collection of image descriptors could be used either by itself or be further transformed into a fixed length representation such as a histogram. Using such a "bag-of-visual-words" representation has become standard practice for many computer vision tasks. The benefit of using a histogram representation is that it converts sets of arbitrary many elements to a fixed length feature descriptor and does not require methods to compare bags of instances.

6.1.1. Bag-of-visual-words

The usual procedure to obtain the bag-of-visual-words representation is as follows. In a first step some keypoints in the image plane are generated at which local image descriptors are subsequently computed. This can be realized by so called *interest point detectors* [Mik05b] that identify stable patterns. That

are those points in the image which are detected even after image transformations like rotation or viewpoint change. A conceptually simpler method is to generate keypoints using a regular grid placed on the image which is also referred to as *dense sampling*. At each such keypoint an image descriptor is extracted, e.g. a SIFT descriptor [Low99]. A detailed comparison of several different image descriptors can be found in [Mik05a], however the design of efficient and discriminative image descriptors remains an open research topic.

Now a codebook of size K with elements of the same feature space is created, e.g. using a k-means clustering algorithm. All feature descriptors of an image are vector-quantized into the codebook, which yield a histogram representation. This non-linear pre-processing step involves different choices to be made: Which is the best codebook for a given task? How should quantization be performed? Should the resulting representation be normalized and if so, how? Many recent works have addressed these problems and proposed different solutions [Now06, Laz07, vG08].

6.1.2. Codebook Kernels

The common part of all approaches making use of bag-of-visual-words representations is the use of a classification function (most often an SVM) *after* application of the pre-processing pipeline. The approach we advocate here is to start with this function directly and design kernels which correspond to the different choices one could have made beforehand. It is hard to a priori distinguish between discriminative and not-so-discriminative codebooks and therefore we explicitly avoid to do so. We regard the codebooks themselves as free parameters *that are available* to the learning algorithm to optimize.

We design proposal kernels in the following way. Let X, X' be bags of D-dimensional image features, e.g. 128 dimensional SIFT features. Each bag might contain a different number of feature descriptors. They are quantized to a histogram representation using a codebook $\mathcal{C}$ with K codewords by means of a nearest-neighbor quantization function

$$q_{\mathcal{C}} : 2^{\mathbb{R}^D} \to \mathbb{R}_+^K \tag{6.1}$$

mapping sets of features into a histogram representation using $\mathcal{C}$. We regard each possible choice of a codebook as a parameter of the kernel. The proposal kernels are χ^2-kernels of the form

$$k(X, X'; \{\gamma^2, \mathcal{C}\}) = \exp\left(-\gamma^2 \chi^2(q_{\mathcal{C}}(X), q_{\mathcal{C}}(X'))\right), \tag{6.2}$$

where $\chi^2 : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$ denotes the χ^2 distance function introduced in Section 2.1.6. We refer to this kernel as a *codebook kernel* due to its dependency of a codebook. We could jointly optimize over γ^2 but for simplicity fix γ^2 to be the reciprocal of the median of all pairwise training distances. The only parameter left in the problem is the codebook $\mathcal{C}$.

A linear combination of kernels which correspond to different codebooks can also be understood as a soft quantization step. In a soft quantization each image descriptor is mapped into a vector with continuous values rather than a binary one. The value in each bin depends on the distance of the image descriptor to the prototype of the bin. The final histogram for an image is obtained by summing the vectors of all the image descriptors as done for hard quantization. Therefore the position of an image in the feature space is described more precisely since it is measured with respect to multiple points.

Learning with Codebook Kernels

Learning a linear combination of the codebook kernels using IKL involves the maximization of the subproblem $T(\mathcal{C}; \alpha)$ over the set of all possible codebooks. Since this is difficult we aim for an approximation and optimize $T(\mathcal{C}; \alpha)$ by evaluating it at many different codebooks. A codebook is generated by randomly sampling feature vectors from the training set that are subsequently used as prototypes used for the quantization step.

6.1.3. Benchmark Datasets

We test our approach of learning the codebook on four standard datasets. Some sample images from these datasets are shown in Figure 6.1. We continue with a more detailed description of the datasets. The first three contain images of textures, whereas the last one is a benchmark dataset for object detection.

Brodatz

The Brodatz [Cen] dataset[1] contains 112 textures images, one per class. These images were subdivided into thirds horizontally and vertically to produce 9 images per class.

KTH-TIPS

The KTH-TIPS [Fri04] dataset[2] of textures contains 810 images from ten different categories: aluminum, brown bread, corduroy, cotton, cracker, linen, orange peel, sandpaper, sponge and styrofoam. For each category 81 images are available that are recorded at nine different scales.

UIUCTex

The UIUCTex dataset [Laz05][3], consists of 40 images per class of 25 textures distorted by significant viewpoint changes and some non-rigid deformations.

[1] http://www.cipr.rpi.edu/resource/stills/brodatz.html
[2] http://www.nada.kth.se/cvap/databases/kth-tips/documentation.html
[3] http://www-cvr.ai.uiuc.edu/ponce_grp/data/index.html

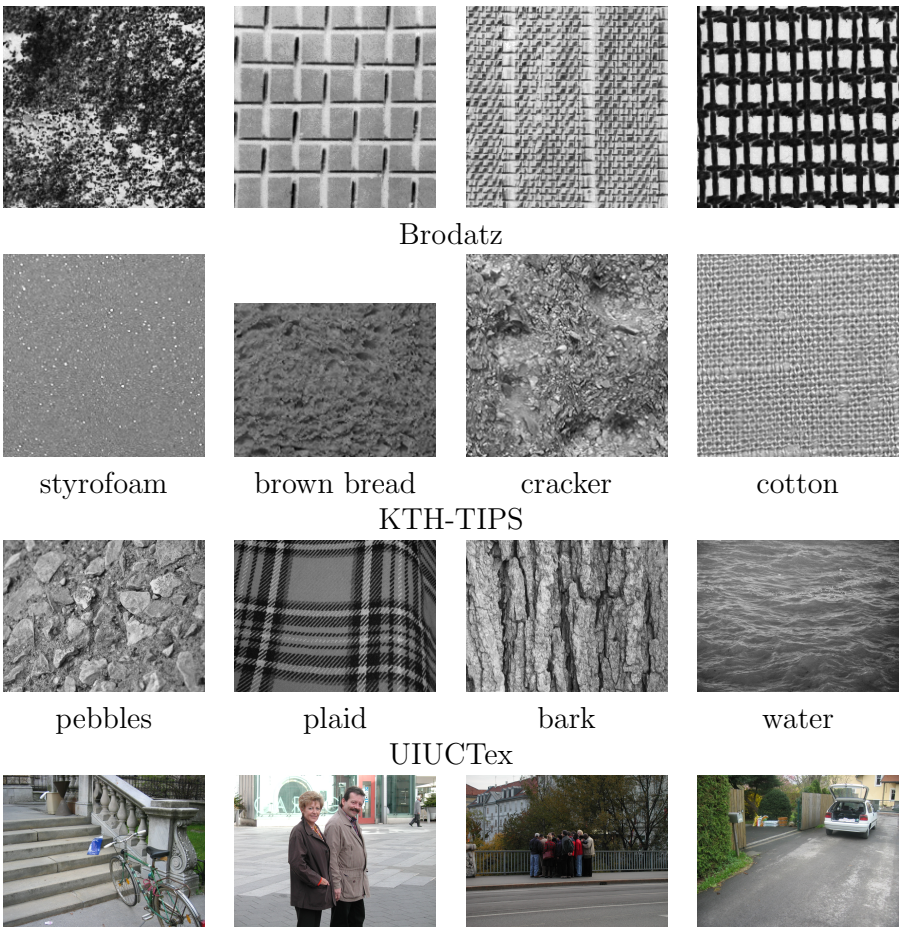


Figure 6.1.: Four example images from each of the datasets used for codebook learning experiments. All datasets are multiclass, see text for details.

Graz-02

The Graz dataset [Ope06][4] was designed for benchmarking object classification algorithms. It contains three different object classes, bikes, persons, cars, and one background class. The background images are taken at similar locations but do not depict any of the aforementioned objects. Without the background class the dataset consists of 1096 images that are equally divided into the three categories.

[4] http://www.emt.tugraz.at/~pinz/data/GRAZ_02/

6.1.4. Experimental Setup

We extract PCA-SIFT [Ke04] feature descriptors from the images using a dense grid of points. The radius of the patches we use for feature extraction is varied over 4 different scales (8 to 50 pixels for the UIUCTex images and 4 to 16 pixels for the others). This resulted in the following numbers of features per image for the datasets: Brodatz 1764 features, KTH-TIPS 1600, Graz-02 3072 and UIUCTex 1976 features. The PCA-SIFT features are 36 dimensional and all images within one dataset yield the same number of feature descriptors since they are all of the same size.

Each dataset is split 25 times into half, where one half is used for training and the other half for testing. The first five splits are used for model selection: we choose the regularization constant C from the range $10^{-2}, 10^{-1}, \ldots, 10^{3}$ which gives the best performance on those five splits. This choice for the hyperparameter is subsequently applied to all 25 splits of the data to obtain the reported result. We resolve multiclass decisions by using one-versus-rest classifiers. We use three different codebook sizes $K \in \{100, 300, 1000\}$.

As a baseline we implemented a SVM classifier using a single codebook only. Empirically we found that it is the classification performance does not depend on whether the codebook used for this experiment is created using a k-means clustering of training features or created by sampling the prototypes at random from the pool of all training features. This finding is consistent with a result reported in [Now06].

All other methods have access to multiple codebooks. In summary we compare the following four different models

1. SVM using a single codebook kernel;
2. SVM using an average of multiple codebook kernels with uniform weights ($d_\theta = 1/|\Theta|$);
3. ℓ_1 MKL learning of the weights;
4. ℓ_2 MKL learning of the weights.

6.1.5. Results

The results of the experiments are shown in Figure 6.2 and Table 6.1. For all plots in the figure the size of the codebook is color coded. A solid line represents the results obtained using a single codebook only. The dashed lines are the results of averaging a different number of codebook kernels but without applying any learning procedure. The results using learning are depicted as a circle for the ℓ_2 results and as a star for the ℓ_1 results. They are plotted at the position on the x-axis which corresponds to the number of positive mixing weights d_θ. This corresponds to the number of selected codebook kernels.

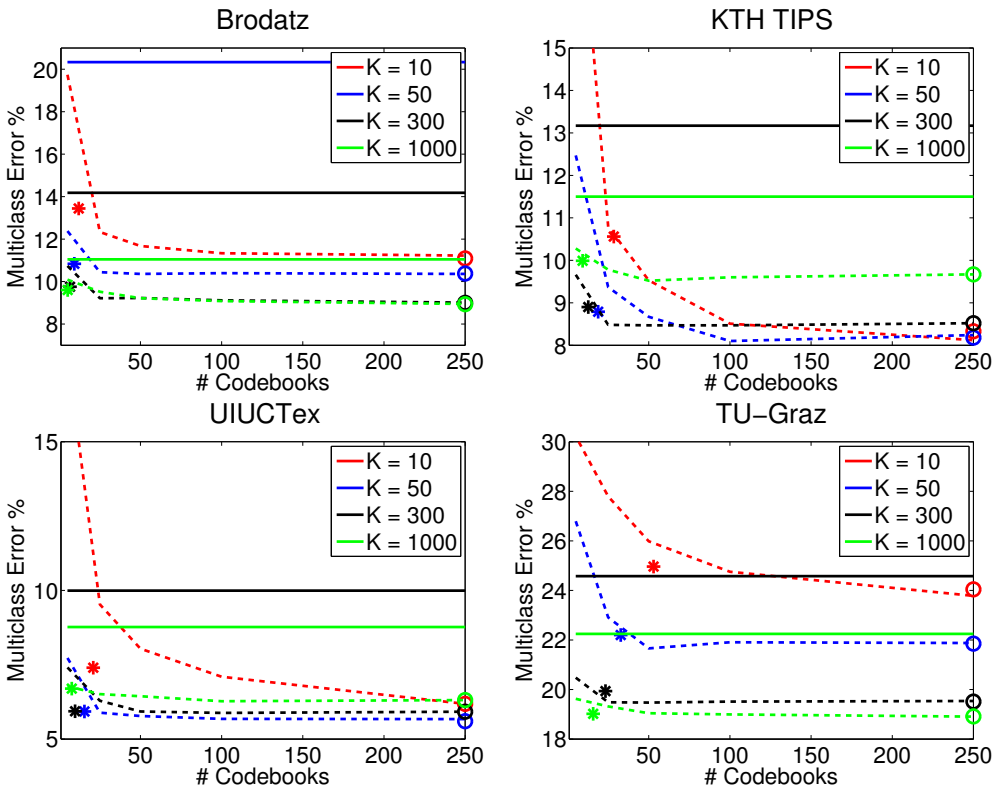


Figure 6.2.: Error rates as function of the number of codebooks on four datasets. The solid line corresponds the result obtained using a single codebook, dashed are results averaging over multiple codebooks. The circle (and stars) correspond to the ℓ_2-MKL (and ℓ_1-MKL) results using all kernels. The MKL results are plotted at the position of the x-axis which corresponds to the number of selected kernels.

6.1.6. Discussion

From the results we learn the following.

1. Using many codebooks and simply averaging the resulting Gram matrices consistently improves the performance with respect to using only a single codebook (dashed versus solid lines). This holds for all codebook sizes.

2. Using ℓ_2-MKL learning does not significantly improve the result over a simple averaging step but leads to similar results (circles versus dashed lines).

3. The ℓ_1-MKL yields competitive results for the datasets UIUCTex and TU-Graz (stars). This is achieved with only a small number of codebooks which is of advantage at test time.

		Single	Average		ℓ_1-MKL		ℓ_2-MKL	
Dataset	K	err	err	#k	err	#k	err	#k
Brodatz	10	46.3 ± 1.8	11.4 ± 1.6	250	13.4 ± 1.8	11.9	11.1 ± 1.4	250
Brodatz	300	14.4 ± 1.1	9.2 ± 1.6	250	9.8 ± 1.2	6.6	9.0 ± 1.8	250
Brodatz	1000	11.1 ± 1.5	9.1 ± 1.3	250	9.7 ± 1.5	5.1	**8.9 ± 1.2**	250
KTH-TIPS	10	33.4 ± 4.5	**8.0 ± 2.5**	250	10.4 ± 2.3	28.2	8.3 ± 1.6	250
KTH-TIPS	300	13.0 ± 2.2	8.5 ± 2.0	250	8.8 ± 2.6	13.0	8.5 ± 2.4	250
KTH-TIPS	1000	11.3 ± 2.2	9.6 ± 2.0	250	9.8 ± 2.1	9.3	9.7 ± 2.2	250
UIUCTex	10	36.4 ± 2.4	6.1 ± 1.0	250	7.3 ± 1.2	20.9	6.2 ± 1.1	250
UIUCTex	300	10.1 ± 1.1	**5.8 ± 0.8**	250	5.9 ± 0.8	9.7	5.9 ± 0.9	250
UIUCTex	1000	8.7 ± 1.0	6.3 ± 0.8	250	6.6 ± 1.0	7.7	6.3 ± 1.1	250
TU-Graz	10	39.8 ± 3.1	24.2 ± 2.2	250	25.0 ± 2.2	47.3	24.0 ± 2.3	250
TU-Graz	300	24.7 ± 1.8	19.8 ± 2.0	250	20.2 ± 1.8	23.2	19.5 ± 2.0	250
TU-Graz	1000	22.7 ± 2.5	19.2 ± 1.8	250	19.3 ± 1.8	15.4	**18.9 ± 1.6**	250

Table 6.1.: Classification error (err) and number of selected kernels (#k) for the datasets Brodatz, KTH-TIPS, UIUCTex and Graz-02. We compare four methods: single codebook with SVM, averaging of different codebook kernels, and learning the weights using ℓ_1 and ℓ_2-MKL. All results are averaged over 25 runs for each of which the data is split into 50% training and 50% test images. The best result for each dataset is printed in boldface.

4. The ℓ_1-MKL solution is much better than just averaging a comparative random number of codebooks (stars versus dashed lines). Thus ℓ_1-MKL learning is able to find a more discriminative codebook and is advantageous over random sampling the same number of codebooks.

A notable result is that even using a small codebook with only 10 elements yields competitive performance on KTH-TIPS and UIUCTex. We checked that there is no further performance gain in averaging over even more codebooks, the results level out at 250 codebooks.

The hard quantization step using the function $q_{\mathcal{C}}$ in Equation (6.1) is not the only applicable choice. Different quantization schemes are applicable, like the soft quantization mentioned above. For example in [vG08] different types of soft quantization with different normalization based on uncertainty or plausibility have been proposed and were compared to each other. In our framework we do not pick one soft quantization and normalization beforehand but offer all of them to the learning algorithm.

$L = 0$ $L = 1$ $L = 2$

Figure 6.3.: Schematic illustration of the spatial pyramid scheme. The original image (left) from the Graz-02 dataset and the decomposition into the first two levels of the spatial pyramid (middle and right). For each cell of the pyramid a separate histogram is computed.

6.2. Learning the Optimal Spatial Layout

Kernels making use of spatial information of the image are an instructive example of how special structure of the data can be used for kernel design. The idea behind a spatial kernel is very simple and dates back to local receptive fields of neural networks. Instead of comparing two images in its entirety only those features that fall into a certain subwindow are compared, e.g. all those of the upper half. To use this idea for kernel functions was first suggested for handwritten digit classification in [Sch97] in combination with polynomial kernels.

In this section we will present how the spatial kernel design can be guided by using a kernel learning approach.

6.2.1. Spatial Kernels

In the following we will discuss the main ideas behind spatial kernels and then show how our paradigm can be applied to this class of kernels.

Spatial Pyramid Kernel

The *spatial pyramid match kernel* was developed in [Laz06] as a spatial variant of the so called *pyramid match kernel* [Gra07]. It gained some attention after excellent performance on the standard Caltech datasets were reported making use of this kernel [Laz06, Gri07][5]. Both kernels fall into the larger class of kernels comparing probability distributions [Hei05].

The spatial pyramid scheme is implemented as follows. We subdivide the image into L different levels of a pyramid, enumerated by $l = 0, 1, \ldots, L-1$.

[5] In [Var07, Bos07b] even better results were reported but were found to be wrong due to errors in the experimental setup, cf. Section 7.1.

The level l corresponds to a $2^l \times 2^l$ equally spaced grid on the image plane, and thus consists of 4^l non overlapping *cells* of equal size. Level 0 of the pyramid is the image itself. This procedure is also illustrated in Figure 6.3. In this construction higher levels of the pyramid correspond to a finer griding of the image. We build a histogram for each cell in a level l individually which results in 4^l histograms for this level. All histograms of one level are concatenated for a final representation yielding a feature vector of dimension $4^l K$.

The spatial pyramid kernel compares two images by measuring the similarity between the concatenated histograms for each level individually and linearly combining them as

$$k(x, x'; \{\theta_l, d_l\}_{l=0}^{L-1}) = \sum_{l=0}^{L-1} d_l k_l(x, x'; \theta_l), \tag{6.3}$$

with k_l being the kernel comparing the images x and x' only through level l. The authors of [Laz06] propose weighting coefficients $d_l = 2^{-(L-l)}$ with L being the maximum level, usually around 3. This choice gives more importance to the finer levels of the pyramid. Although good results are reported using the spatial pyramid kernel, the natural question arises whether or not the pyramidal representation of the image is the best for a given task and if the intuition behind the choice of d_k is justified. In the following we will address these two questions.

A General Class of Spatial Kernels

Instead of pre-choosing the spatial layout and its combination weights we will move these design choices into the learning problem. To this end we propose the following class of spatial kernels. By B we denote a box in the image plane defined in relative coordinates, such that its specification is independent of the actual size of the image. With $\chi^2_B(x, x')$ we denote the χ^2-distance comparing two images represented by a collection of local features, taking into account only features which fall into the box B. Our proposal kernels are of the following form

$$k(x, x'; \{\gamma^2, B\}) = \exp\left(-\gamma^2 \chi^2_B(x, x')\right). \tag{6.4}$$

For the experiments the parameter γ^2 is again chosen as the reciprocal of the median of the pairwise distances.

During ℓ_1-MKL learning we have to maximize the function $T(B; \alpha)$ over all possible boxes. This could be done by the efficient subwindow search method [Lam08b] using a branch-and-bound algorithm ensuring global optimality. For simplicity we resort to approximate optimization by evaluating T at many samples. In order to do so, a box is sampled uniformly from the set of all boxes which fall entirely in the image. This distribution favors small boxes as in the pyramid construction. In the upper left picture of Figure 6.7

Figure 6.4.: 18 example images from the Caltech-101 dataset. In total there are 102 different classes of object categories.

we plotted 1000 randomly sampled boxes on top of each other. Points in the middle of an image appear in more boxes than points at the boundary. In the pyramidal representation every pixel coordinate falls into the same number of boxes, thus the corresponding picture would be all-white.

6.2.2. Benchmark Datasets

To test the effectiveness of our paradigm, we re-implemented and augmented the experiments of [Laz06]. Experiments were carried out on two datasets, the Scene-13 dataset [Li05][6] consisting of thirteen different scene types like kitchen, landscape, urban, etc. shown in Table 6.5 and the prominent Caltech-101 [FF04][7] dataset. The latter consists of 101 classes, with the number of training images per class ranging from 31 to 800. Some sample images are shown in Figure 6.4.

We seek to demonstrate how the spatial layout can be optimized over in our framework and are not aiming to compete with the state-of-the-art methods for this dataset. In order to obtain competitive results with the best reported Caltech-101 results, e.g. [Gri07, Pin08] one would at least need to integrate

[6] `http://visionlab.ece.uiuc.edu/datasets.html`
[7] `http://www.vision.caltech.edu/Image_Datasets/Caltech101/Caltech101.html`

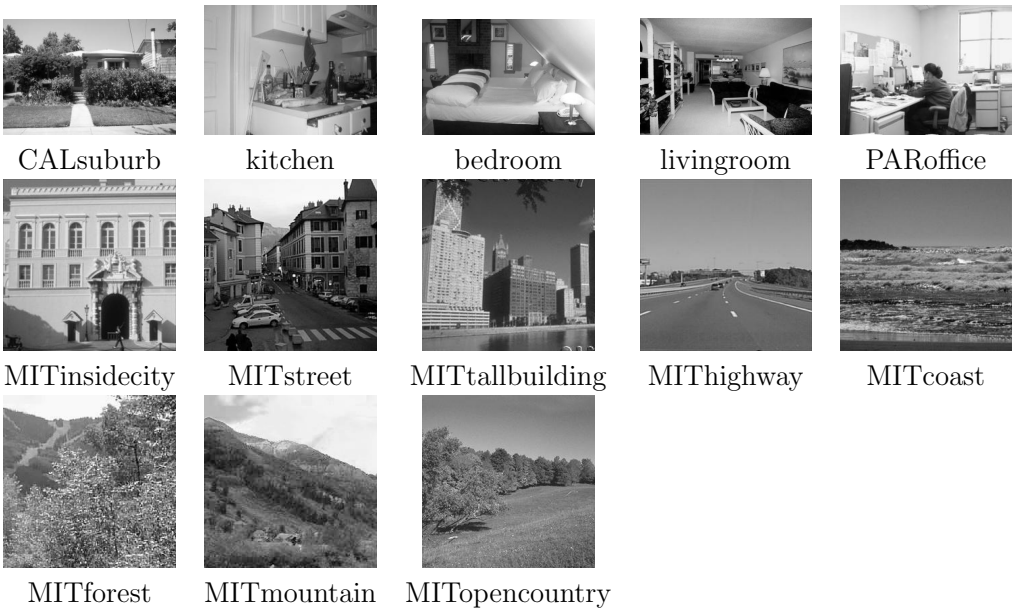


Figure 6.5.: Example images from the Scene-13 dataset, one image per category

more feature cues such as color information. This will be the topic of the next chapter where we will show how to combine different features to increase the classification performance for this particular dataset.

The Scene-13 dataset is split 10 times using 100 images of each category as training and the rest as testing images. Following common practice for the Caltech-101 dataset we used conducted two experiments using 15 and 30 training images per category and the remaining ones for testing. We report the multiclass error that is obtained by averaging the per class recall rates to not over-emphasize categories with a large number of images. The multiclass error err_{mc} is the average

$$\mathrm{err}_{mc} = \frac{1}{\mathcal{C}} \sum_{c=1}^{\mathcal{C}} \mathrm{err}(c), \tag{6.5}$$

where

$$\mathrm{err}(c) = \frac{\text{true positives in class } c}{\text{elements in class } c}. \tag{6.6}$$

The results are averaged over five independent splits of the data.

6.2.3. Experimental Setup

SIFT features of 128 dimensions are extracted densely from the image using every 10th pixel at four different scales with the radii 4, 8, 12 and 16.

Subsequently the features are quantized to a codebook of size $K = 300$ (and additionally $K = 1000$ for the experiments on Scene-13). The codebook size is kept fix in order to eliminate the relative influence of this choice in the results. For the Caltech-101 experiments the regularizer was set to $C = 1000$.[8] For the Scene-13 dataset we perform model selection to choose $C \in \{0.1, 1, 10, 100\}$ using five-fold cross validation on the training set only. As before, a one-versus-rest scheme is used to resolve multiclass decisions.

Three different spatial layouts were used for the experiments

1. a pyramidal representation using concatenated histograms within the levels (marked "Levels" in Table 6.2 and 6.3);

2. using all cells of the pyramid as bounding boxes B for the spatial kernel (6.4) ("Cells");

3. randomly sampling bounding boxes as described above.

6.2.4. Results

In Figure 6.6 the performance of the different approaches is plotted as a function of the number of available proposal kernels. Solid lines correspond to results making use of randomly sampled boxes to generate the proposal kernels and dashed lines to those using the pyramidal representation. The red line depicts the ℓ_1-MKL results, but additionally we plotted the result obtained by using all subwindow kernels (125 for Scene-13 and 1000 for Caltech-101) at the position of the x-axis corresponding to thee number of selected kernels.

The raw numbers of the experiments are shown in the Tables 6.2 and 6.3.

6.2.5. Discussion

From the results we can draw the following conclusions.

1. On both datasets, the use of randomly sampled boxes outperforms the pyramid layout using both fixed and learned mixing coefficients. This improvement is substantial for the Caltech-101 indicating that the pyramidal layout that has been used so far is not well suited for this kind of image data.

2. For the Scene-13 dataset the ℓ_1-MKL performs better, on the Caltech dataset the ℓ_2-MKL gives better results. This is presumably due to the fact that the Caltech images depict the objects in the center of the image and thus the average of all bounding boxes (Figure 6.7, upper left) is a more suitable prior of the discriminative regions in the image.

[8] The authors of [Laz06] did not report this parameter, but since our results are in accordance with theirs, this seems to be a fair choice and enables comparison.

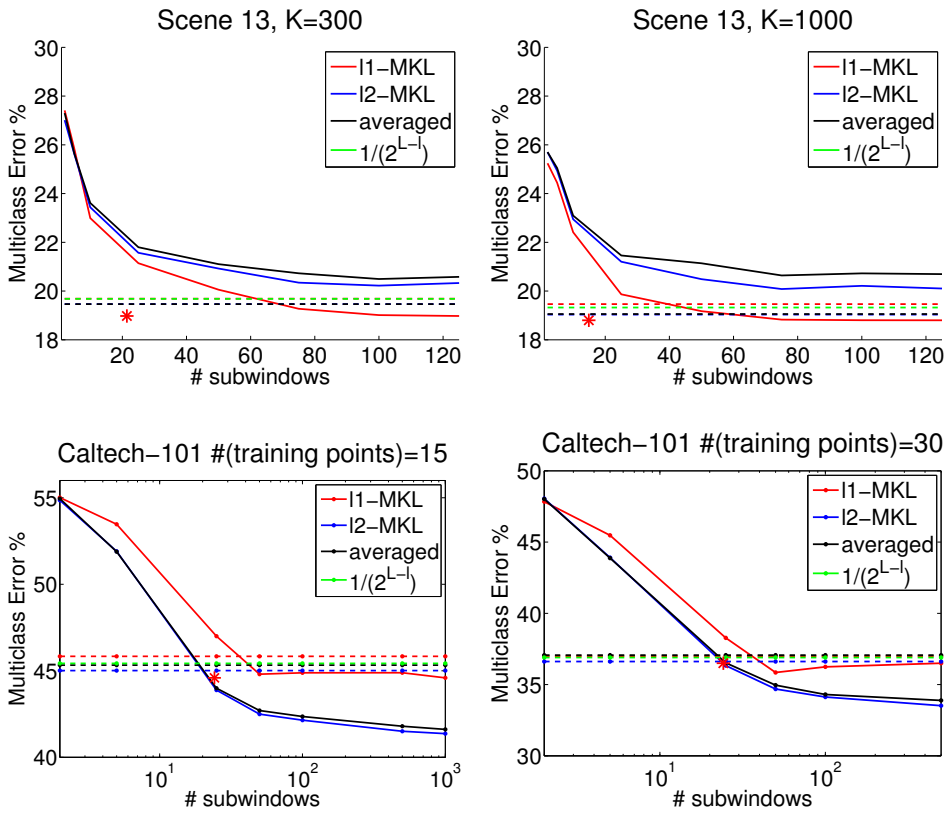


Figure 6.6.: Results of the classification task on the Scene-13 dataset (upper) and Caltech-101 (lower) using 15/30 training images. The dashed lines correspond to the best result obtained using a pyramidal representation. The solid lines are results from averaging and learning using randomly sampled boxes. Additionally we plot the ℓ_1-MKL result at the position of the x-axis corresponding to the number of selected kernels. (Thus the rightmost value of the solid red line coincides with the red star marker.)

3. Learning using ℓ_2-MKL improves over simple averaging, but averaging is a competitive baseline.

4. The choice $d_l = 2^{-(L-l)}$ yields good performance but an averaging is equally good.

For the Caltech-101 dataset the best stated result in [Laz06] is 35.4% misclassification error using 30 training points (picking the best obtained test error a-posteriori). We record 36.9% for our re-implementation of their method using a pyramidal layout with the choice of $d_l = 2^{-(L-l)}$ but only 33.5% for randomly sampled subwindows and ℓ_2-MKL optimization.

Five spatial layouts found by the ℓ_1-MKL algorithm for the Scene-13 dataset are shown in Figure 6.7(b)-(f) again by plotting the participating bounding

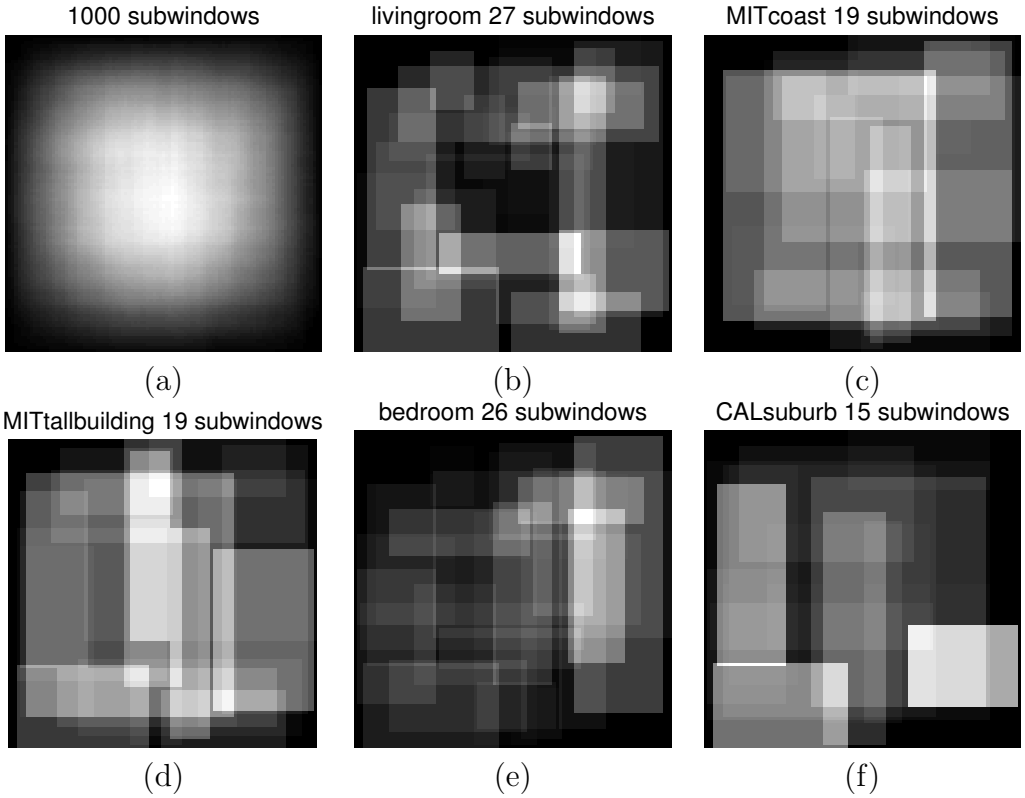


Figure 6.7.: The upper left image (a) depicts 1000 randomly sampled boxes plotted on top of each other with equal weights. The other five images (b)-(f) show the learned spatial configuration for different classes of the Scene-13 dataset. Each one outperforms the pyramidal representation for both levels and cells.

boxes on top of each other. All of the configurations shown in this plot outperform the pyramidal representation, no matter if the weights are set a priori or learnt. For the livingroom and bedroom classes many small subwindows which are approximately uniformly distributed over the image are identified to be discriminative. Conversely for the classes MITcoast, MITtallbuilding and CALsuburb, very large boxes are selected for the final kernel. Here, large regions are to be compared for good results whereas images of bedrooms and livingrooms consist of many small details. We are cautious to not over-interpret the resulting spatial layout but it is evident from the plot that for different classes different configurations are identified.

In summary, the experiments demonstrate that we can learn a mixture of kernels, each of which has access to only a limited part of the image. This overcomes the fixed-grid limitation of the standard spatial pyramid kernel and improves classification performance on both datasets. Although the choice of a pyramidal scheme for image kernels is intuitively appealing it is nevertheless

arbitrary.

6.3. Conclusion

The goal of this chapter was to demonstrate that arbitrary design choices should be avoided when it is possible to optimize over them.

We proposed to view parameters of the pre-processing pipeline as kernel parameters and offer them to a learning algorithm. In particular we focused on two important problems of image classification systems. The demonstrated benefit is two-fold,

- we are relieved from making manual parameter choices;
- the resulting classification functions perform better.

For the practitioner the experimental results give some general insights on what can be expected to work: if many different but equally plausible pre-processing choices exist, then a simple averaging gives competitive results; we have seen this behavior for the codebook learning experiments. If on the other hand it is expected that only few of many possible pre-processing choices are effective, then MKL/IKL can identify the best ones and there is no need for manual pre-selection; this has been observed on the spatial layout learning experiment and the Scene-13 dataset.

We believe the proposed methodology has applications in all high-level computer vision tasks where machine learning methods are used successfully.

Level	Single SVM	Average		$2^{-(L-l)}$		ℓ_1-MKL		ℓ_2-MKL	
	Levels	Levels	Cells	Levels	Cells	Levels	Cells	Levels	Cells
0 (1×1)	26.1 ± 1.0	-	-	-	-	-	-	-	-
1 (2×2)	22.0 ± 0.5	21.3 ± 0.6	21.2 ± 0.9	21.1 ± 0.7	21.6 ± 0.9	21.3 ± 0.6	21.0 ± 0.5	20.9 ± 0.7	21.2 ± 0.9
2 (4×4)	20.9 ± 0.6	19.8 ± 0.5	19.8 ± 0.5	19.7 ± 0.6	20.3 ± 0.6	19.7 ± 0.5	20.3 ± 0.7	19.7 ± 0.6	19.6 ± 0.6
3 (8×8)	22.6 ± 1.0	19.5 ± 0.5	20.7 ± 1.0	19.8 ± 0.6	21.4 ± 0.7	19.7 ± 0.4	20.3 ± 0.3	19.5 ± 0.4	20.4 ± 0.8
125 random win.	-	20.6 ± 0.9		-		**19.0 ± 0.8 (21 kernels)**		20.3 ± 0.9	

Table 6.2.: Error rates on the Scene-13 dataset averaged over 10 splits. We used different pyramidal setups, either (Levels) concatenating all cell histograms of one level or (Cells) using each single pyramid cell as one kernel. The last row gives the result when using random subwindows for the layout.

	Level	Single SVM	Average		$2^{-(L-l)}$		ℓ_1-MKL		ℓ_2-MKL	
		Levels	Levels	Cells	Levels	Cells	Levels	Cells	Levels	Cells
15 tr.pts.	0 (1×1)	59.0 ± 0.5	-	-	-	-	51.1 ± 0.4	50.7 ± 0.6	51.0 ± 0.6	50.4 ± 0.5
	1 (2×2)	51.2 ± 0.5	51.5 ± 0.6	50.4 ± 0.4	50.8 ± 0.6	50.5 ± 0.4	46.0 ± 0.6	45.8 ± 0.5	47.0 ± 0.7	45.0 ± 0.4
	2 (4×4)	45.9 ± 0.6	47.8 ± 0.6	45.3 ± 0.4	46.1 ± 0.8	45.4 ± 0.4	47.8 ± 0.7	48.4 ± 0.7	46.2 ± 1.0	46.1 ± 0.5
	3 (8×8)	47.8 ± 0.7	46.7 ± 0.7	46.8 ± 0.5	45.6 ± 0.8	47.8 ± 0.7	44.6 ± 0.7 (24.2 kernels)		**41.4±0.7**	
1000 random win.		-	**41.6±0.3**		-					
30 tr.pts.	0 (1×1)	52.9 ± 1.0	-	-	-	-	43.7 ± 0.7	43.2 ± 0.4	43.1 ± 0.6	42.7 ± 0.7
	1 (2×2)	43.8 ± 0.6	43.8 ± 0.6	42.6 ± 0.5	43.0 ± 0.7	42.9 ± 0.5	37.8 ± 0.8	37.0 ± 0.8	38.6 ± 0.5	36.6 ± 0.7
	2 (4×4)	37.7 ± 1.0	39.5 ± 0.5	37.1 ± 0.7	37.5 ± 0.6	37.0 ± 0.8	38.7 ± 1.1	38.2 ± 0.8	37.4 ± 0.5	37.1 ± 1.1
	3 (8×8)	39.2 ± 1.7	37.9 ± 0.4	37.8 ± 0.9	36.9 ± 1.3	38.8 ± 1.2	36.2 ± 0.6 (21.3 kernels)		**34.1 ± 0.7**	
100 random win.		-	**34.3 ± 0.7**		-					

Table 6.3.: As in Table 6.2 but for the Caltech-101 dataset using 15 and 30 training points. All results are averaged over 5 different splits of the data.

7. Image Feature Combination for Multiclass Object Classification

7.1. Introduction

In this chapter we address the problem of object category classification by combining multiple diverse feature types. For a given test image the classifier has to decide which class the image belongs to. This problem is challenging because the instances belonging to the same class usually have high intra-class variability.

To overcome the problem of variability, one strategy is to design feature descriptors which are highly invariant to the variations present within the classes. Invariance is an improvement, but not all of the feature descriptors will have the same discriminative power for all classes. For example, features based on color information might perform well when classifying flowers of different types, whereas a classifier for cars should be invariant to the actual color of the car. Therefore it is widely accepted that, instead of using a single feature type for all classes it is better to adaptively *combine* a set of diverse and complementary features – such as features based on color, shape and texture information – in order to discriminate each class best from all other classes.

Finding these feature combinations is a recent trend in class-level object recognition and image classification. One popular method in computer vision to tackle this problem is MKL. In the application of MKL to object classification, the approach can be seen to linearly combine similarity functions between images such that the combined similarity function yields improved classification performance [Kum07, Lin07, Var07]. [1]

In Section 7.2 we give a general overview of the addressed problem. The

[1] In the recent years the MKL based method of combining different features was believed to be a very powerful technique after excellent results (up to 98.1% for Caltech-101) have been reported using this method [Var07, Bos07a, Bos07b, Bos08]. It was also seen as the winner of the Caltech-256 object classification challenge [cal07]. In the course of our experimentation we failed to reproduce those results and subsequently found severe errors in the experimental setup used to obtain the results claimed in those works. Test label information was available already during the training time of the algorithms. The authors acknowledged the fact that errors occurred which positively benefited the reported classification accuracy. In this work we deliberately will not compare against these published results since the errors render the them incorrect.

Sections 7.3-7.5 describes several combination approaches. Experiments are presented in Section 7.6 and 7.7. We conclude in Section 7.8.

7.2. Feature Combination Methods

We begin with a formal definition of the problem we address in this chapter.

Definition 7.2.1 (Feature Combination Problem)**.** *Given a training set* $\{(x_i, y_i)\}_{i=1,\ldots,n}$ *of* n *instances consisting of an image* $x_i \in \mathcal{X}$ *and a class label* $y_i \in \{1, \ldots, \mathcal{C}\}$*, and given a set of* F *image features* $f_m : \mathcal{X} \to \mathbb{R}^{d_m}$*,* $m = 1, \ldots, F$ *where* d_m *denotes the dimensionality of the* m*'th feature descriptor, the problem of learning a classification function* $y : \mathcal{X} \to \{1, \ldots, \mathcal{C}\}$ *from the features and training set is called* feature combination problem*.*

A typical example of such a feature f_m would be a bag-of-visual-words histogram of the image as introduced in Section 6.1.1. Then, the corresponding dimensionality d_m would be the codebook size used for the vector quantization step.

In the following, we will use the name *feature combination method* for all methods which address the feature combination problem.

The feature combination problem is a special case of multiclass classification. Since the main topic of this thesis are kernel methods we will address the problem of learning a multiclass classifier from training data by means of kernel classifiers. As described earlier, kernel methods make use of kernel functions defining a measure of similarity between pairs of instances. In the context of feature combination it is useful to associate a kernel to each image feature as follows. For a kernel function k between real vectors we define the short-hand notation

$$k_m(x, x') = k(f_m(x), f_m(x')), \tag{7.1}$$

such that the image kernel

$$k_m : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \tag{7.2}$$

only considers similarity with respect to image feature f_m. If the image feature is specific to a certain aspect, say, it only considers texture information, then the kernel measures similarity only with regard to this aspect. The subscript m of the kernel can be understood as indexing into the set of features.

In the following, for notational convenience, we will denote the kernel response of the m'th feature for a given sample $x \in \mathcal{X}$ to all training samples x_i, $i = 1, \ldots, n$ as $K_m(x) \in \mathbb{R}^n$ with

$$K_m(x) = [k_m(x, x_1), k_m(x, x_2), \ldots, k_m(x, x_n)]^T.$$

In case x is the i'th training sample, i.e. $x = x_i$, then $K_m(x)$ is simply the i'th column of the m'th kernel matrix.

Name	Test-time function	Coefficients	Training	Parameters	References
Averaging	$y(x) = \underset{c=1,\ldots,\mathcal{C}}{\operatorname{argmax}}[\alpha_c^\top \left(\frac{1}{F}\sum_{m=1}^F K_m(x)\right) + b_c]$	$\alpha \in \mathbb{R}^{\mathcal{C}\times n}$ $b \in \mathbb{R}^{\mathcal{C}}$	$(\alpha, b)_c$, ind.	C	
Product	$y(x) = \underset{c=1,\ldots,\mathcal{C}}{\operatorname{argmax}}[\alpha_c^\top \left(\prod_{m=1}^F K_m(x)\right)^{1/F} + b_c]$	$\alpha \in \mathbb{R}^{\mathcal{C}\times n}$ $b \in \mathbb{R}^{\mathcal{C}}$	$(\alpha, b)_c$, ind.	C	
MKL	$y(x) = \underset{c=1,\ldots,\mathcal{C}}{\operatorname{argmax}} \sum_{m=1}^F \beta_m^c \left(\alpha_c^\top K_m(x) + b_c\right)$	$\beta \in \mathbb{R}^{\mathcal{C}\times F}$ $\alpha \in \mathbb{R}^{\mathcal{C}\times n}$ $b \in \mathbb{R}^{\mathcal{C}}$	$(\alpha_c, b_c, \beta^c)_c$ ind.	C	[Bac04, Son06, Var07]
MC-MKL	$y(x) = \underset{c=1,\ldots,\mathcal{C}}{\operatorname{argmax}} \sum_{m=1}^F \beta_m^c \left(\alpha_c^\top K_m(x) + b_c\right)$	$\beta \in \mathbb{R}^{\mathcal{C}\times F}$ $\alpha \in \mathbb{R}^{\mathcal{C}\times n}$ $b \in \mathbb{R}^{\mathcal{C}}$	$((\alpha, b, \beta)_c)$ jointly	C	[Zie07]
CG-Boost	$y(x) = \underset{c=1,\ldots,\mathcal{C}}{\operatorname{argmax}}[\sum_{m=1}^F \alpha_{c,m}^\top K_m(x) + b_c]$	$\alpha \in \mathbb{R}^{\mathcal{C}\times F\times n}$ $b \in \mathbb{R}^{\mathcal{C}}$	$(\alpha, b)_c$, ind.	C	[Bi04]
LP-β	$y(x) = \underset{c=1,\ldots,\mathcal{C}}{\operatorname{argmax}} \sum_{m=1}^F \beta_m \left(\alpha_{c,m}^\top K_m(x) + b_{c,m}\right)$	$\beta \in \mathbb{R}^F$ $\alpha \in \mathbb{R}^{\mathcal{C}\times F\times n}$ $b \in \mathbb{R}^{\mathcal{C}\times F}$	1. $(\alpha, b)_c$, ind 2. β, jointly	1. C_m 2. $\nu \in (0, 1)$	[Dem02]
LP-B	$y(x) = \underset{c=1,\ldots,\mathcal{C}}{\operatorname{argmax}} \sum_{m=1}^F B_m^c \left(\alpha_{c,m}^\top K_m(x) + b_{c,m}\right)$	$B \in \mathbb{R}^{F\times\mathcal{C}}$ $\alpha \in \mathbb{R}^{\mathcal{C}\times F\times n}$ $b \in \mathbb{R}^{\mathcal{C}\times F}$	1. $(\alpha, b)_c$, ind 2. B, jointly	1. C_m, 2. $\nu \in (0, 1)$	[Dem02]

Table 7.1.: Comparison of multiclass learning approaches to the feature combination problem in image and object classification. In the column "Training" it is also noted which parameters are trained independently (ind.) w.r.t. to classes c and which are trained jointly. The methods LP-β and LP-B are trained in two separate stages, which is denoted by 1. and 2. In principle one can select a regularization constant C for each class separately. We used only one value of C for all classes jointly.

Feature selection as kernel selection In this chapter we study kernel classifiers that aim to combine several kernels into a single model. Since we associate image features with kernel functions, kernel combination/selection translates naturally into feature combination/selection.

In the following we will present several methods in a unified setting along with their training procedures. An overview of the different methods in their multiclass variant can also be found in the Table 7.1.

7.3. Methods: Baselines

We include three simple baseline methods, both of which combine kernels in a pre-defined deterministic way to form a new kernel that is subsequently used for SVM training.

7.3.1. Best Single Feature

A conceptually simple approach is the use of Cross Validation to select a single kernel from the set $\{k_1, \ldots, k_F\}$. Every feature combination method should be able to outperform this baseline method or at least match its performance if a single feature is sufficient for good classification.

7.3.2. Averaging Kernels

Arguably the simplest method to combine several kernels is to average them. We define the kernel function

$$k^*(x, x') = \frac{1}{F} \sum_{m=1}^{F} k_m(x, x'), \tag{7.3}$$

which is subsequently used in a support vector machine (SVM).

Training The only free parameters are the SVM parameters. We use CV to estimate the best regularization constant. A multiclass variant is built using a one-versus-rest scheme.

7.3.3. Product Kernels

The next baseline method we consider is to combine several kernel by multiplication. All F kernels we use are of the form $k_m(x, x') = \exp(-\gamma_m d_m(x, x'))$ and we use

$$k^*(x, x') = \prod_{m=1}^{F} \exp\left(-\frac{\gamma_m}{F} d_m(x, x')\right) \tag{7.4}$$

as the single kernel in a SVM.

Training Same as for averaging.

7.4. Methods: Multiple Kernel Learning

In MKL the kernel combination is a part of the optimization problem. This method was already introduced in depth in Chapter 4 and we just recall the that the final decision function of MKL is of the following form

$$F_{\mathrm{MKL}}(x) = \mathrm{sign}\left(\sum_{m=1}^{F} \beta_m \left(K_m(x)^T \alpha + b\right)\right). \tag{7.5}$$

A slightly different variant of the multiple kernel learning objective (4.10) has been proposed in [Var07] specifically for the task of feature combination. We also implemented this variant but found that for all experiments the results using either variant do not differ. Therefore we omitted the presentation of these results, they can be regarded as equal to the ones obtained using the previously used MKL solution.

Training The only free parameter in the MKL approaches provided that the proposal kernels are pre-selected is the regularization constant C, which is chosen using CV. A multiclass decision is resolved with a one-versus-rest scheme. The final decision function is also shown in Table 7.1. All one-versus-rest classifiers can be trained in parallel.

Multiclass MKL For strongly unbalanced datasets a MKL classifier trained as a multiclass classifier might be preferable over the one-versus-rest setup. The authors of [Zie07] derive such a MC-MKL formulation in which all parameters for all classes are trained jointly. Due to performance issues this approach renders infeasible for the large scale experiments presented here. [2].

7.5. Methods: Boosting Approaches

As last class of feature combination methods we look at boosting approaches and in particular propose two more methods which are inspired by the MKL decision function. All methods in this section are, as MKL, based on mixtures of kernels. We start with the presentation of the binary classifier in 7.5.1 and generalize it to the multiclass case in Section 7.5.2.

[2]Personal communications with the authors of [Zie07].

7.5.1. LPBoost - Binary Classification

With the mixing coefficients β_m summing to one, the binary MKL decision function is a convex linear combination of the real valued outputs of F SVMs $f_m(x) = K_m(x)^T\alpha + b$. Furthermore we observe that all of the SVMs included in the sum share the same parameter set $\{\alpha, b\}$. Having noted this, MKL can be understood as a restricted version of the following more general form

$$F(x) = \text{sign}\left(\sum_{m=1}^{F} \beta_m f_m(x)\right), \tag{7.6}$$

where $f_m(x)$ are some real valued functions, not necessarily support vector machines and not necessarily trained jointly. In Boosting terminology the f_m are known as *weak learners*.

This observation leads naturally to the following model. We use the kernels $k_m, m = 1, \ldots, F$ to train separate SVMs f_m, resulting in different parameter sets $\{\alpha_m, b_m\}$. Subsequently we optimize over $\beta = \{\beta_1, \ldots, \beta_F\}$ in a second step. Each individual function f_m is not restricted to share the parameters but can be trained independently to yield maximal generalization. The details of this two-step learning procedure are given in Section 7.5.2.

The possibility of unconstraining the SVM parameters α does not enhance the space of possible decision functions. A resulting decision function of form (7.6) is a convex combination of several hyperplanes and thus itself again a hyperplane. If it were the optimal one for the MKL problem it could, due to the representer theorem, be represented as a combination of the kernel evaluations using only n training points.

A benefit of training the parameters in a two stage process is that with only F mixing coefficients to optimize over in the second step, a true multiclass formulation becomes feasible. The downside of the approach is that it is more demanding in terms of training data.

Problem formulation We propose to learn all parameters of the model in two separate steps. In the first step the functions f_m are trained individually. Subsequently we optimize over β using the following linear program (LP), which is equivalent to ν-LPBoost [Dem02]

$$\min_{\beta,\xi} \quad -\rho + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i \tag{7.7}$$

$$\text{sb.t.} \quad y_i \sum_{m=1}^{F} \beta_m f_m(x_i) + \xi_i \geq \rho, \quad i = 1, \ldots, n, \tag{7.8}$$

$$\sum_{m=1}^{F} \beta_m = 1, \quad \beta_m \geq 0, \quad m = 1, \ldots, F, \tag{7.9}$$

$$\xi_i \geq 0, \quad i = 1, \ldots, n, \tag{7.10}$$

with $\{\xi_1, \ldots, \xi_n\}$ being slack variables. The equivalence to LPBoost can be seen by considering the hypothesis space to be the finite set of functions $\{f_1, f_2, \ldots, f_F\}$. The problem can easily be solved using standard linear programming solvers. Due to the special coupled structure in the dense constraint matrix, we found interior-point based solvers to be consistently faster than simplex based method.[3] There is only one hyper-parameter ν in the problem which trades the smoothness of the resulting function with the hinge loss on the points, analogously to the SVM regularization parameter C.

7.5.2. LPboost - Multiclass Variant: LP-β and LP-B

It is straightforward to derive a multiclass version of Problem (7.7). In the multiclass case with $\mathcal{C}$ classes, the functions f_m are no longer real-valued but map into a $\mathcal{C}$ dimensional space $f_m(x) \to \mathbb{R}^{\mathcal{C}}$. The c'th output of $f_m(x)$ will be denoted by $f_{m,c}(x)$.

We consider two possible variations of learning mixing weights. The first, termed *LP-β* uses a single vector β for all classes. This β defines a combination that works well for all classes *jointly*. Alternatively each class can have its own weight vector over the features, in which case there is a weight matrix $B \in \mathbb{R}^{F \times \mathcal{C}}$, we name this method *LP-B*.

LP-β.

The decision rule of LP-β can be found in Table 7.1. All parameters of the decision functions f_m, e.g. (α_m, b_m) for all classes and features are determined in a first step. The mixing coefficients β are learned by the following multiclass extension of LPBoost.

$$\min_{\beta,\xi,\rho} \quad -\rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \tag{7.11}$$

$$\text{sb.t.} \quad \sum_{m=1}^{F}\beta_m f_{m,y_i}(x_i) - \operatorname*{argmax}_{y_j \neq y_i}\sum_{m=1}^{F}\beta_m f_{m,y_j}(x_i)$$

$$+\xi_i \geq \rho, \quad i = 1, \ldots, n, \tag{7.12}$$

$$\sum_{m=1}^{F}\beta_m = 1, \quad \beta_m \geq 0,\ m = 1, \ldots, F, \tag{7.13}$$

$$\xi_i \geq 0, \quad i = 1, \ldots, n. \tag{7.14}$$

Since we only optimize over $\mathcal{C}$ parameters, learning them in such a true multiclass formulation is feasible.

A benefit of this method is that β is sparse on the level of the features. This means features for which $\beta_m = 0$ need not be computed for the final decision

[3] We use the MOSEK interior-point solver, see `http://www.mosek.com/`.

function, which is of advantage at test time. Although the MKL solution is sparse for every class separately, it may not be sparse *jointly* in a one-versus-rest MKL setup. In the experiments we observed that almost always every feature is selected at least once, making its computation necessary also during test time.

LP-B

In this second variant each class is assigned its own weight vector resulting in a $F \times \mathcal{C}$ weighting matrix B. The coefficient B_m^c is the mixing coefficient for the m'th feature response for class c. The decision rule is shown in Table 7.1. The corresponding learning problem extends the Hinge loss in (7.12) to the multiclass Hinge loss originally proposed by [Wes99] for support vector machines. This formulation reads

$$\min_{B,\xi,\rho} \quad -\rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \tag{7.15}$$

$$\text{sb.t.} \quad \sum_{m=1}^{F} B_m^{y_i} f_{m,y_i}(x_i) - \sum_{m=1}^{F} B_m^{y_j} f_{m,y_j}(x_i)$$

$$+\xi_i \geq \rho, \quad i = 1,\ldots,n,\ y_j \neq y_i, \tag{7.16}$$

$$\sum_{m=1}^{F} B_m^c = 1, \quad m = 1,\ldots,F, \tag{7.17}$$

$$B_m^c \geq 0, \quad m = 1,\ldots,F,\ c = 1,\ldots,\mathcal{C}, \tag{7.18}$$

$$\xi_i \geq 0, \quad i = 1,\ldots,n. \tag{7.19}$$

Note that this is still a linear programming problem, but more expensive to solve than LP-β due to the increased number of parameters.

Training

The training procedures for LP-β and LP-B are analogous. Ideally we have enough data to adjust f_m and β on independent sets. If this is not the case, we use the following two-stage scheme to avoid biased estimates.

1. First we perform model selection using 5 fold CV to select the best hyper-parameters for each f_m individually (in our case f_m are SVMs and we need to select C). With this choice of hyperparameters we train for each of the five cross validation split a function $f_m^i, i = 1,\ldots,5$ on the training set solely for the purpose of computing the outputs on the validation set of that split. The cross-validation scores is the union all those outputs. This results in a prediction for each training point using a classifier which was *not* trained using that point (but on 80% of the training data). For each feature we train a final function f_m using the entire training data. At this point the only hyper-parameter left is ν.

2. We estimate the hyperparameter ν as well as the mixing coefficients using *only* the cross-validation scores. We perform CV to select the best parameter ν and subsequently train the final combination β. Afterwards, the cross-validation scores can be discarded.

The main concern using this training scheme, is that the generated training data for LP-β training stems not from the classifiers $f_m, m = 1, \ldots, F$ that are later used in the combination but are the cross-validation scores. Therefore we have to make the assumption that the functions f_m^i and f_m are not too different. This is reasonable since the functions f_m^i used to produce the training data for LP-β are trained on 80% of the training data. The experiments validate this assumption as we do not observe overfitting for the LP-β model. We want to emphasize that the entire procedure uses training data only.

The LP-B results tend to be worse compared to LP-β (see Results section), so fitting its $\mathcal{C} \times F$ instead of F parameters seems to demand for more training data or its objective function is not suited for this problem.

7.5.3. Column Generation Boosting for Mixtures of Kernels

A different variant of the Boosting technique that is based on the same observation from 7.5.1 was proposed in [Bi04]. Instead of maintaining the separation between the SVM parameters (α, b) and mixing coefficients β the authors propose to solve

$$\min_{\alpha_m} \quad \frac{1}{2} \sum_{m=1}^{F} \alpha_m^T \alpha_m + C \sum_{i=1}^{n} L \left(y_i, b + \sum_{m=1}^{F} K_m(x_i)^T \alpha_m \right). \tag{7.20}$$

This formulation can be understood as training a SVM with a linear kernel with the kernel evaluations at the training points as features.[4] This method is referred to as CG-Boost in the experiments.

Training Since the formulation reduces to a linear SVM the only free parameter is again the regularization parameter C, which is selected using CV. We experimented with different loss functions L and found that logistic loss

$$L(y, x) = \log\left(1 + \exp(-y f(x))\right) \tag{7.21}$$

yields best results while assuring good convergence of the algorithm.

7.6. The Oxford Flowers dataset

In this section we present results on the Oxford flowers dataset [Nil06]. This dataset consists of images depicting flowers. In total there are 17 different

[4]Implemented using liblinear-1.33, a standard solver for linear SVM.

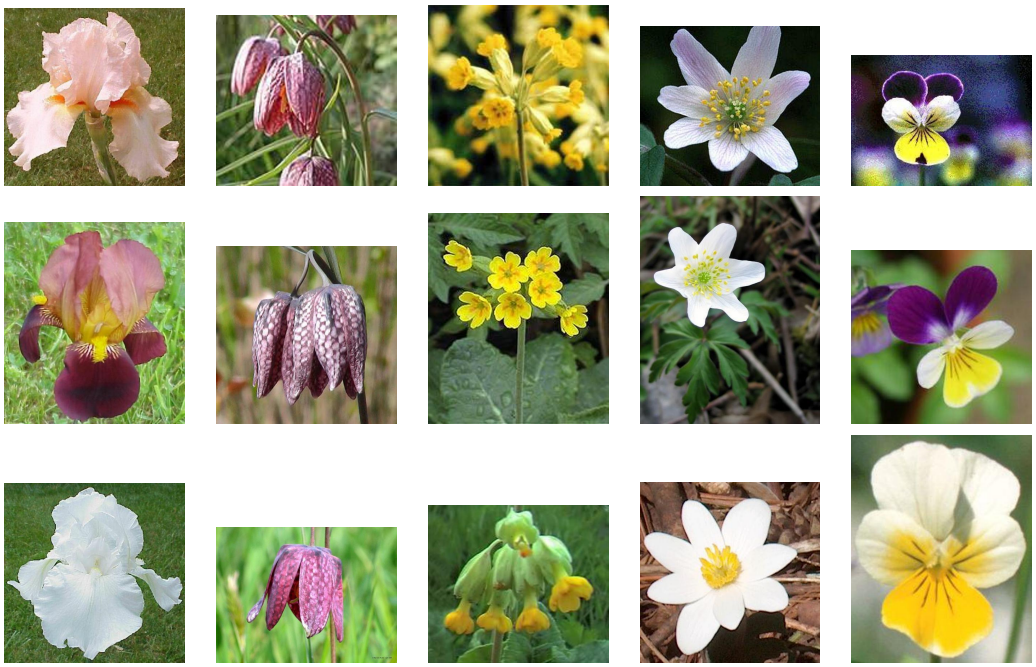


Figure 7.1.: 15 example images from five categories of the Oxford Flowers dataset. Images in the same column are from the same class.

types of flowers with 80 images per category. Example images are shown in Figure 7.1. The task is to classify the images into the different flower types.

The dataset comes with three predefined splits into test (17×20 images), train (17×40 images) and validation set (17×20 images). Furthermore the authors of [Nil08] provide seven precomputed distance matrices online on their website[5] and those matrices are used for the experiments presented here. Each matrix is computed using a different feature type, namely clustered HSV values (HSV), SIFT features on the foreground region (siftint), SIFT features on the foreground boundary (siftbdy) and three matrices derived from colour, shape and texture vocabularies. Since we are interested in the combination of different features rather than feature design we refer to [Nil06, Nil08] for details on the image features.

7.6.1. Experimental Setup

We first compare the overall performance of all models presented in this chapter. To this end we use the predefined splits for training and model selection. The regularization parameter is selected from the range $C \in \{0.01,0.1,1,10,100,1000\}$. For LP-$\beta$ and LP-B the regularization parameter $\nu \in \{0.05, 0.1, \ldots, 0.95\}$ of the second stage is also selected on the valida-

[5] `http://www.robots.ox.ac.uk/~vgg/research/flowers/index.html`

Single features			Combination methods		
Feature	Accuracy	Time	Method	Accuracy	Time
Colour	60.9 ± 2.1	3	product	85.5 ± 1.2	2
Shape	70.2 ± 1.3	4	averaging	84.9 ± 1.9	10
Texture	63.7 ± 2.7	3	CG-Boost	84.8 ± 2.2	1225
HOG	58.5 ± 4.5	4	MKL (SILP)	85.2 ± 1.5	97
HSV	61.3 ± 0.7	3	MKL (Simple)	85.2 ± 1.5	152
siftint	70.6 ± 1.6	4	LP-β	85.5 ± 3.0	80
siftbdy	59.4 ± 3.3	5	LP-B	85.4 ± 2.4	98

Table 7.2.: Mean accuracy for all methods on the Oxford Flowers dataset using the predefined splits [Nil08]. Also plotted is the total time for model selection, training and testing in seconds. The simple product and averaging combination methods are orders of magnitude faster than learning based methods.

tion set, using the procedure described in Section 7.5.2. Kernel functions are computed as

$$k_m(x, x'; \sigma_m) = \exp\left(-\frac{1}{\sigma_m} d_m(x, x')\right) \tag{7.22}$$

with d_m being one of the pre-computed distances and σ fixed to the median of the pairwise distances. This leads to a total of seven different kernel functions.

7.6.2. Results

The results are shown in Table 7.2, with results using a SVM with a single kernel only are shown in the left, and combination methods in the right column. Since all classes have the same number of test images we report the accuracy as the number of correctly classified examples divided by the number of all examples. The accuracies for different features vary from 58.5 to 70.6, while the combination results are approximately the same. Also shown is the total time in seconds, needed for model selection, training and testing the methods. The regular SVM and baseline methods are orders of magnitude faster than the other combination methods.

7.6.3. Discussion

From the results we can draw several conclusions. All feature combination methods dramatically improve the classification performance, a finding consistent with [Nil08]. These results clearly indicate that a combination of image features is advantageous over using single specialized features only. The best single feature achieves an accuracy of about 71%, while every combination methods yields to about 85%.

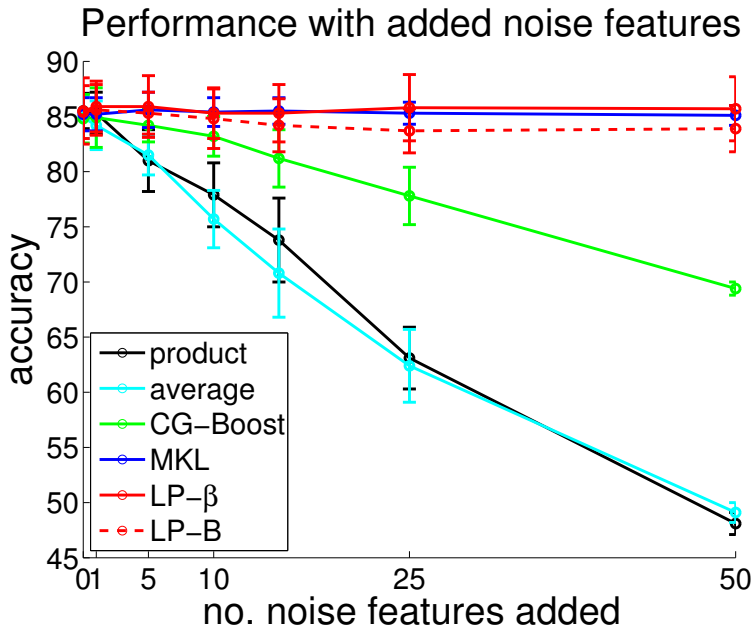


Table 7.3.: Accuracy of the different methods while adding more noise features to the proposal set. All learning based combination methods are robust to the inclusion of noise features, while baseline techniques are not.

Having said this, we note that the baselines (single) and (product) that are almost always left out in comparisons (e.g. [Nil08, Var07, Rak08, Son06, Kum07]) yield equally good results but are magnitudes faster than any other combination method. For example the entire time needed to perform model selection plus final training of the classifier using a SVM with an averaged kernel takes only 10 seconds compared to approximately 100 seconds needed for MKL and LP-β training.

Many authors have argued that the mixing coefficients of the MKL solution are interpretable [Nil08, Nil06, Var07, Son06]. They are usually interpreted as the influence of the features on a particular class. While this may be intuitive for the case of binary classification, in a multiclass setting this reasoning is misleading. One could equally plausible argue, that, due to the same result of the averaging baseline with the MKL classifier, all features are equally important for the multiclass decision.

7.6.4. Learning With Uninformative Features

In a second experiment we will highlight the benefit of using a learning approach for feature combination over the baseline methods. Additional to the seven discriminative features we add non-discriminative features by generat-

ing random vectors from a three dimensional normal distribution. A Gaussian kernel is computed on these noise features and included into the set of kernels. Now the same experiment as before is repeated. In Figure 7.3 we plot the performance of the models against the number of added noise kernels.

The baselines incorporate all features with the same importance and subsequently their performance drops severely. Among the learning methods MKL and LP-β turn out to be very robust to uninformative features, while the CG-Boosting approach slowly decays in performance.

This experiment highlights a feature of MKL, namely that it is able to select kernels out of a large class of potentially un-informative ones, e.g. wrong kernel parameters. However this is exactly *not* the scenario typically encountered in object classification, where each feature on its own is designed to be discriminative.

7.7. The Caltech Object Classification Datasets

For the second set of experiments we use the well known Caltech datasets [FF04, Gri07] which have become the standard benchmark for visual object classification besides the Pascal VOC dataset [Eve07]. The Caltech-101 dataset was already introduced in the last chapter and some example images are shown in Figure 6.4.

7.7.1. Experimental Setup

We follow the experimental setup proposed by the designers of the datasets. The performance is measured as the mean prediction rate over all classes. We compute the per class error $\mathrm{err}(c)$ as in (6.5) and the multiclass accuracy acc_{mc} with

$$\mathrm{acc}_{mc} = 1 - \frac{1}{\mathcal{C}} \sum_{c=1}^{\mathcal{C}} \mathrm{err}(c). \tag{7.23}$$

This balances the influence of categories with a large number of test examples. The total number of images available in Caltech-101 is 9144 with 102 categories while Caltech-256 consists of 30607 images in 257 categories. The 30 largest categories of Caltech-101 are also part of the Caltech-256 dataset.

While the minimum number of images in one class of the Caltech-101 dataset is 31, making it possible to use only 30 training images per class while retaining images to test on, the smallest category of Caltech-256 consists of 80 images. We report results using all 102 classes of the Caltech-101 dataset averaged over five splits and for 256 classes of the Caltech-256 dataset, excluding its clutter category, on a single split.

The number of training images for the Caltech-101 experiments are 5, 10, 15, 20, 25, 30 images per category for training and up to 50 images per category

for testing. For the Caltech-256 dataset 25 images per class are used for testing testing and 5, 10, 15, 20, 25, 30, 40, 50 images per class for training.

All participating kernels are of the form as in (7.22) with σ again fixed to the reciprocal of the median of all pairwise distances. The features used to compute the distances are described in more detail in Section 7.7.2.

The LP methods require training in two stages. In a first step separate SVMs are trained using each feature type individually. To this end we cross validate on the training set the hyper-parameter C from the set $\{0.1, 1, 10, 50, 100, 500, 1000\}$. For the LP ν is selected in the range of 0.05 to 0.95 in steps of 0.05. Values around 0.8 are typically selected. For MKL we tried a number of regularization constants and chose $C = 1000$ which yields best results. This choice is in accordance to many results published in the literature [Var07, Laz06].

7.7.2. Image Descriptors

In the following we briefly describe the features that were used for the experiments. For all but the V1S+ features we also compute spatial pyramid variants in the same manner as outlined in Section 6.2.1.

PHOG Shape Descriptor

Shape is modeled using the PHOG descriptor proposed in [Bos07a]. The descriptor is a histogram of oriented (Shp_{360}) or unoriented (Shp_{180}) gradients computed on the output of a Canny edge detector. The oriented histogram Shp_{360} contains 40 bins, the unoriented Shp_{180} 20 bins yielding a total of 2×4 kernels (L=3). The χ^2 distance is used to compute histogram similarity.

Appearance Descriptor

Appearance information is modeled using SIFT descriptors [Low99] which are computed on a regular grid on the image with a spacing of 10 pixels and for the four different radii $r = 4, 8, 12, 16$. The descriptors are subsequently quantized into a vocabulary of visual words that is generated by k-means clustering. We use four variants of this feature type: two codebook sizes (300 and 1000 prototypes) and grey image descriptors (128 dims) as well as HSV-SIFT (3*128=384dims)). HSV-SIFT are computed by concatenation of SIFT descriptors computed separately on the hue, saturation and value channels of the color images. RGB color images were converted to gray-scale using the transformation $I = 0.3R + 0.59G + 0.11B$, where I denotes the intensity of the gray level image.

With a pyramid representation ($L = 3$) this adds to a total of 4×4 kernels again using the χ^2 distance.

In the last chapter, Section 6.2 we identified a kernel particular well suited for the Caltech datasets. This kernel was constructed by averaging over many kernels of the form (6.4) with bounding boxes B drawn at random. [6] Here we will use an average over 100 randomly sampled subwindows and refer to the final averaged kernel as the *subwindow kernel*. Although learning the mixing weights with ℓ_2 MKL instead of taking the average was shown to be slightly advantageous we chose the conceptually simpler averaging step as a compromise of performance and speed.

Region Covariance

In [Tuz07] it is proposed to use the covariances of simple per-pixel features as robust and discriminative statistics for human detection. We use the tangent-space projected features. We use the Euclidean distance and a pyramid representation which yields 3 kernels (L=2).

Local Binary Patterns

The authors of [Oja02] argue to use *locally binary pattern* (LBP) features, retaining the classification performance of textons for texture classification while being much faster and simpler to extract. We use histograms of uniform rotation-invariant $LBP_{8,1}$-features and create 3 kernels (L=2) with the χ^2-distance.

V1S+

In [Pin08] a population of locally normalized, thresholded Gabor functions spanning a range of orientations and spatial frequencies are derived and advocated as particular simple features. Image features are compared using the Euclidean distance of features on the entire image ($L = 0$). This results in one kernel.

7.7.3. Results and Discussion

We distinguish two different settings for the experiments: combining kernels based on very similar features, i.e. different levels of a pyramid and combining diverse features e.g. different types of image features. Additionally we will compare our results to the best published results in the literature.

Combining Similar Features

In Figure 7.2 results are shown for combining the four kernels of the spatial pyramid using SIFT (a) or PHOG (b) features. We plot the performance

[6] The distribution is shown in 6.7(a).

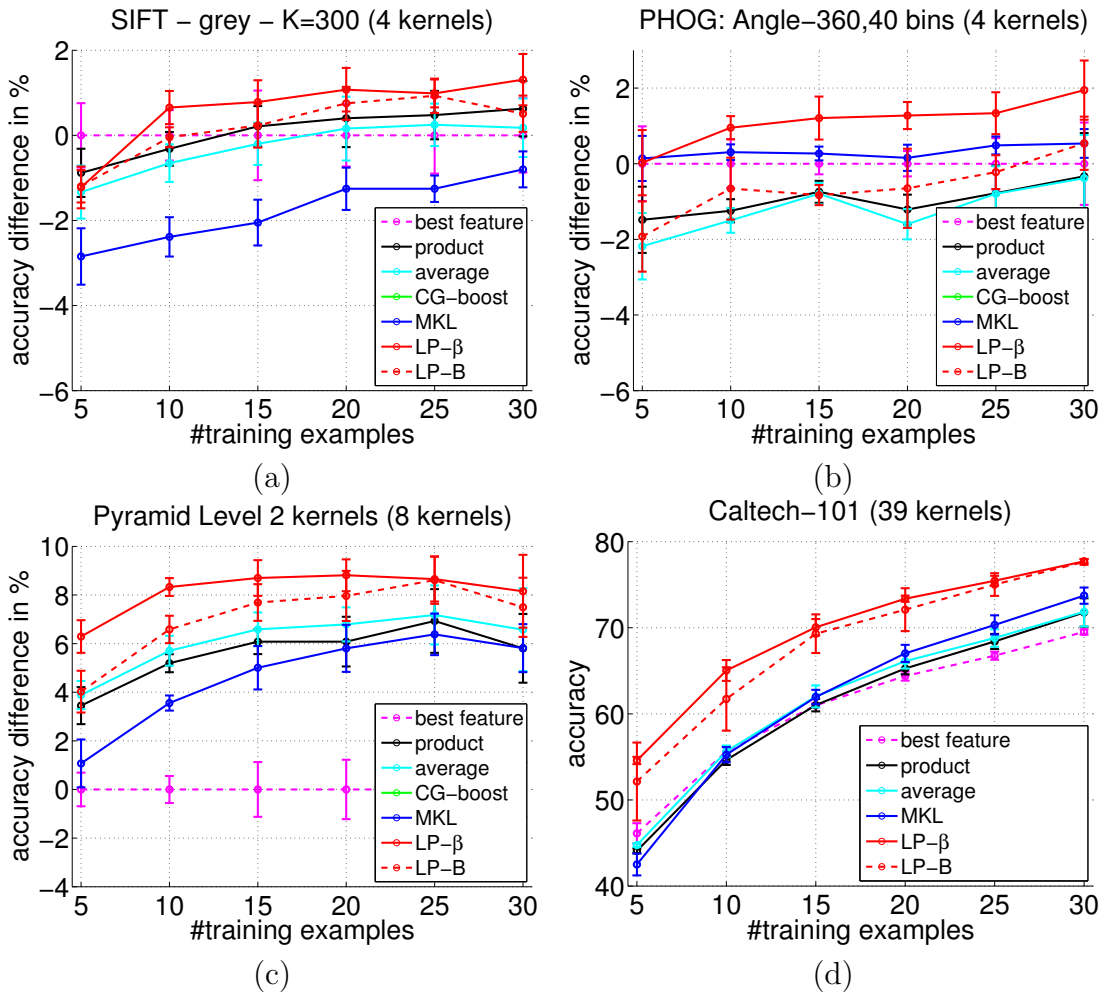
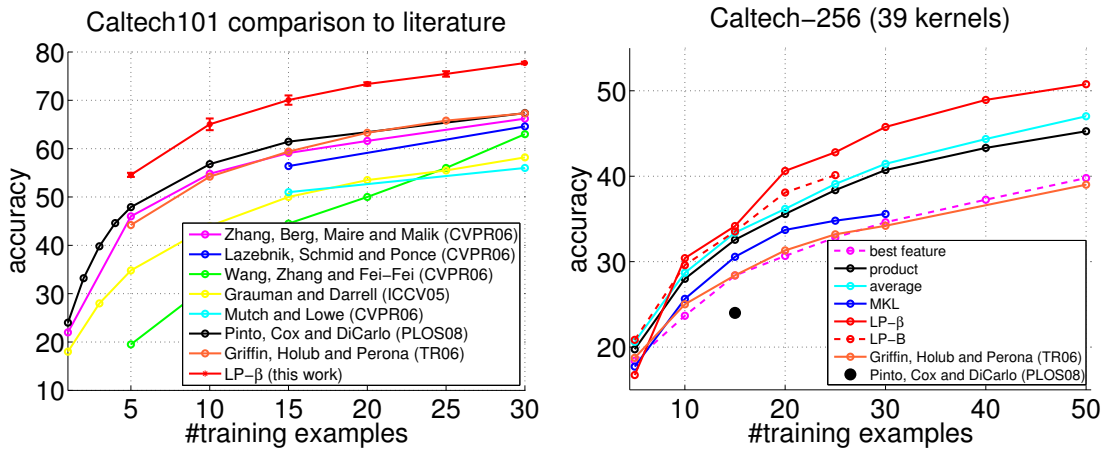


Figure 7.2.: Multiclass accuracy of different combination methods on the Caltech-101 dataset. In (a),(b) and (c) the performance difference with respect to the best single feature identified via CV is plotted. Shown are: in (a)+(b) results combining four kernels of a spatial pyramid using the same image feature type (SIFT or PHOG). (c) combining eight kernels of different image features. (d) comparison of methods using a total of 39 kernels.

difference with respect to the best single kernel which is selected using Cross Validation.

All models yield similar results but several observations can be made.

- When combining pyramid kernels of SIFT features, MKL and CGBoost are outperformed by the baseline methods average and product and even yield worse results than using a single kernel alone (Figure 7.2(a)). For combination of PHOG pyramid kernels the baseline results are worse than CGBoost or MKL.



(a) (b)

Figure 7.3.: Comparing the results to the best ones reported in the literature for Caltech-101 (a) and Caltech-256 (b). LP-β is found to outperform all previously published methods by about 10%. Some numbers have been estimated from plots, see Table 7.5 for the exact numbers.

- LP-β yields the best results for both combinations if trained with more than five examples per category.

Combining Diverse Features

Figure 7.2(c) shows the result of combining 8 kernels corresponding to different image features. Of every feature descriptor described previously we use the kernel of Level 2. These are more diverse than combining kernels derived from levels of the same pyramid. We make the following observations.

- All feature combinations yield a significant improvement over the result using the single best kernel (best feature).
- Among the learning based method LP-β performs best.
- CG-Boost and MKL are outperformed by the baseline methods.

In the next experiment we aim for maximal performance on the Caltech datasets using a total of 39 different kernels. In addition to the kernels from Section 7.7.2 we include the products of the pyramid levels for each feature resulting in an additional 7 kernels. Furthermore we use two subwindow kernels with SIFT and HSV-SIFTs (codebooksize $K = 1000$). The result is shown in Figure 7.2(d).

The best single feature, estimated using Cross Validation, among the 39 different kernels are the V1S+ features. We find the same qualitative result as

before. The LPBoosting techniques yield best performance while the baselines and MKL are comparable. As mentioned already, LP-β identifies a sparse solution on the level of the features. For Caltech-101 with 30 training examples per class 7 out of 39 features are selected:. The LP-B approach consistently gives worse results than its restricted version LP-β.

Comparison to Literature

In Figure 7.3(a) we compare LP-β as the best method to state-of-the-art results for the Caltech datasets published in the literature [Gra05, Laz06, Zha06, Wan06, Mut06, Gri07, Pin08]. Compared to these results, the methods in this chapter like MKL and the baselines, yield a better performance. This is due to the use of very discriminative features, whereas some results from the literature are obtained using a single image feature only. This is a clear indication that the features employed for the experiments are very compatible and feature combination methods are powerful tools for this dataset.

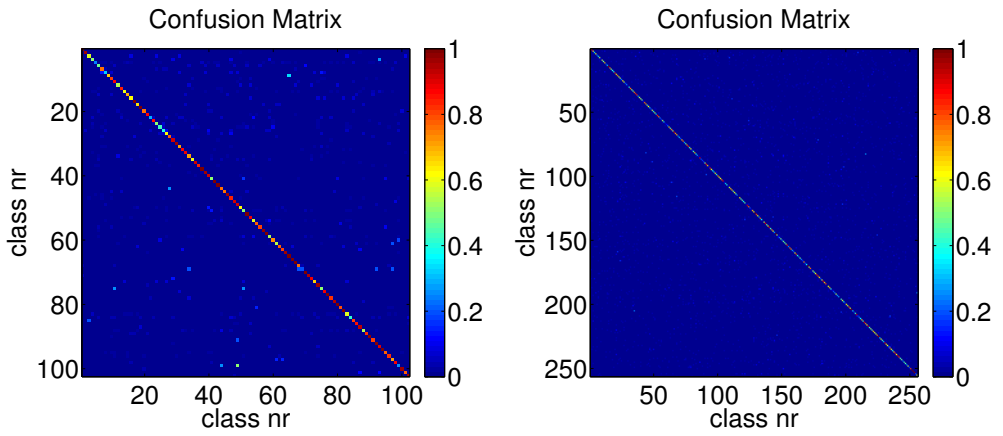
The results for Caltech-256 are shown in Figure 7.3(b) with LP-β achieving a more than 10% improvement over the best published results [Gri07, Pin08]. Using 30 training images the final classifier is a linear combination of 15 individual one-versus-rest SVMs, thus a sparse number of feature is selected among the proposed ones.

For the best result using the LP-β method we plot confusion matrices in Figure 7.4 and a list of the accuracies attained on the tested object categories in Figure 7.5 and Figure 7.6. A trend which can be observed from the latter two plots is that rigid categories like motorbikes, scissors, minaret are among the easier categories while object categories consisting of animal images like crocodile, octopus, beaver, ant are the ones with the lowest accuracy. This is most probably because of a higher intra-class variability of the animal pictures.

7.7.4. Training Time Comparison

We compare the runtime performance of the different methods in terms of a single run to train the model. The numbers are based on using 15 training examples per class, which adds to a total of 1530 for Caltech-101 (and 3840 for Caltech-256).

The required training time for an entire one-versus-rest SVM multiclass classifier using a single kernel is about 5s (50s for Caltech-256). Estimating the 39 coefficients of β takes 60s (8.5m) and for B 935s (4.9h). Thus a single run of LP-β requires about $6 \times 39 \times 5s + 60s \approx 21$m ($6 \times 39 \times 50$s+8.5m$\approx$ 3.4h) which includes the computation of the CV output (5×39 SVMs) and the final weak classifiers (39 SVMs). These numbers are comparable to MKL training which takes about 23m (5h) for all classes. All implementations can most likely be optimized.



Caltech-101 Caltech-256

Figure 7.4.: Confusion matrices for the Caltech-101 (left) and Caltech-256 (right) dataset. Shown are the results obtained using 30 (resp.50) training images per class.

The combined time for training LP-β are higher since we perform model selection, whereas for MKL we fixed the hyper-parameter beforehand. The training time for the baseline methods are orders of magnitudes faster. For a good performance at test time one has to reduce the amount of kernel evaluations and more importantly the amount of image features which have to be computed for each single image. LP-β is advantageous over the other methods in the last respect since it selects a sparse number of image features which work well for all classes simultaneously.

7.8. Conclusion

In this chapter we studied several methods for combining multiple image features for visual object classification systems. We interpreted the MKL decision function as a convex combination of SVMs and inspired by this proposed formulations based on the Linear Programming Boosting. These are different to CG-Boost [Dem02], in that they maintain two distinct sets of parameters which are optimized separately in two separate steps.

In order to enable efficient training we derived a two step training procedure that works well in practice. This two step training procedure arguably is less principled than a joint optimization. However in practice this seems not to be a problem and works well even in the case of few training examples. Due to the two training stages most of the training, e.g. training the SVMs, can be parallelized.

We found that the LP-β approach consistently outperforms all other con-

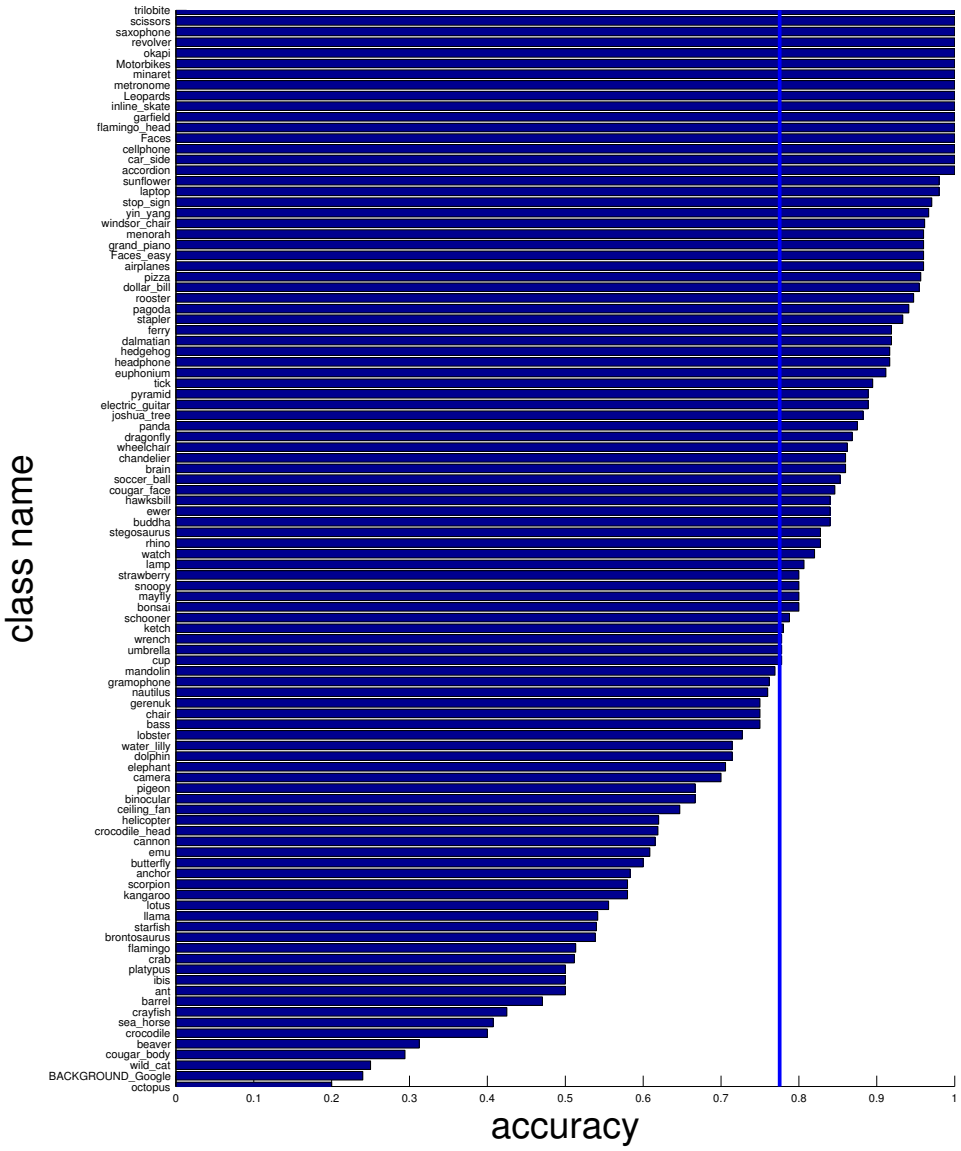


Figure 7.5.: Accuracy of LP-β using 30 training images on the Caltech-101 dataset. Shown is the per class accuracy (diagonal of the confusion matrix, $\mathrm{acc}(c) = 1 - \mathrm{err}(c)$) as bars, sorted in descending order. The final multiclass accuracy is the average and depicted by a solid blue line.

sidered methods and obtain the best results published up-to-date. On both Caltech datasets we observe an more than 10% improved performance over the best published result.

An interesting fact is that LP-β was found to yield better results than the best participating weak learner. This is in general not true for the other combination methods. Therefore we expect even better performance if we train with

more image features or include other classification functions. Adding more learners comes with a reasonable additional cost since it only scales linearly in F, while any trained weak learners can be reused.

Most results in this chapter turn out to be disadvantageous for MKL. The baseline methods yield competitive results and outperform MKL on several setups. This is due to the fact that the available kernels on their own are already discriminative. In the presence of uninformative kernels MKL and the LPBoosting techniques are able to identify a discriminative set of kernels and maintain good performance. However we want to stress that this is exactly not the typical scenario in multiclass object classification where designers of image features strive to make their features as expressible as possible and tune for optimal performance.

We conclude with the observation that MKL might have been overestimated in the past, due to non reproducible results reported in [Var07]. The baseline methods "average" and "product" should be considered as its canonical competitors and included in any study using MKL. With LP-β we derived a method that yields better performance, is equally fast and leads to sparse multiclass object classification systems.

Method	1	3	5	10	15	20	25	30
Zhang, Berg, Maire and Malik (CVPR06)[Zha06]	22*	-	46*	54.8*	59.1	61.6*	-	66.2
Lazebnik, Schmid and Ponce (CVPR06)[Laz06]	-	-	-	-	56.4	-	-	64.6±0.8
Wang, Zhang and Fei-Fei (CVPR06)[Wan06]	-	-	19.5*	-	44.5*	50*	56*	63
Grauman and Darrell (ICCV05)[Gra05]	18*	28*	34.8*	44*	50	53.5*	55.5*	58.2
Mutch and Lowe (CVPR06)[Mut06]	-	-	-	-	51	-	-	56
Pinto, Cox and DiCarlo (PLOS08)[Pin08]	**24**	**39.8**	47.9	56.8	61.44	-	-	67.36
Griffin, Holub and Perona (TR07)[Gri07]	-	-	44.2*	54.2*	59.4*	63.3*	65.8*	67.6±1.4
LP-β	-	-	**54.6 ± 0.4**	**65.0 ± 1.2**	**70.0 ± 1.0**	**73.4 ± 0.4**	**75.5 ± 0.6**	**77.7 ± 0.3**

Table 7.4.: Classification performance on the Caltech-101 dataset for different state-of-the art methods from the literature. These numbers have been used to create the Figure 7.3 (a) (and part of (b)). Some results found in the literature had to be estimated from plots and are marked with a star.

Method	5	10	15	20	25	30	40	50
Best Feature	18.4	23.7	28.4	30.7	32.8	34.6	37.3	39.8
Product	19.7	28.0	32.6	35.6	38.4	40.7	43.3	45.3
Average	20.6	28.7	33.4	36.2	39.1	41.5	44.4	47.0
MKL	17.7	25.6	30.6	33.7	34.8	35.6		
LP-β	16.7	**30.4**	**34.2**	**40.6**	**42.8**	**45.8**	**48.9**	**50.8**
LP-B	**20.8**	29.6	33.5	38.1	40.1			
Pinto, Cox and DiCarlo [Pin08]	-	-	24	-	-	-	-	-
Griffin, Holub and Perona [Gri07]	18.74± 0.5	25.01±0.5	28.4*	31.31±0.7	33.2*	34.2±0.2	36.8*	39.0±0.5

Table 7.5.: Results on Caltech-256. The best selected single kernel is the subwindow kernel on SIFT (K=1000) features. All results are obtained using all 256 categories leaving out only 257.clutter. Numbers estimated from plots are marked with a star. Results of [Pin08] are obtained on the first 250 categories only and should be slightly higher when tested on all 256 categories since the last 6 classes are the "easiest" in the dataset.

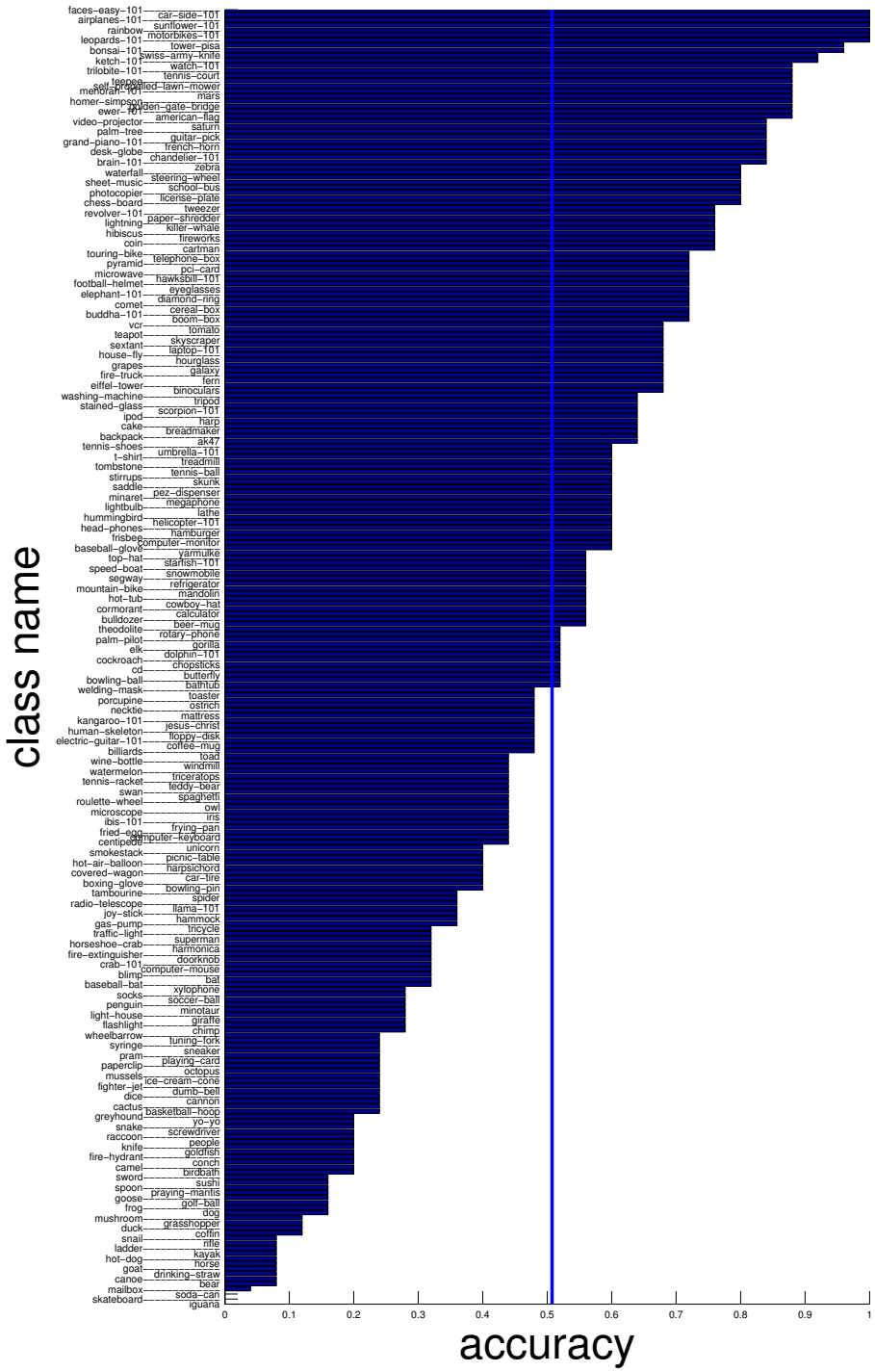


Figure 7.6.: Same as Figure 7.5 but for Caltech-256 using a LP-β classifier trained with 50 images per class. The solid blue line denotes the multiclass error. Categories containing rigid objects yield higher accuracy, especially the categories also used in Caltech-101 are amongst the "easiest".

Appendix

A. Notation

Symbol	Description
SVM	support vector machine
MKL	multiple kernel learning
IKL	infinite kernel learning
MIL	multiple instance learning
$\mathcal{X}, \mathcal{Y}$	input/output space
(x, y)	pair of training example and label
$k(\cdot, \cdot'; \theta)$	kernel function parameterized by θ
$\mathbf{K}, \mathbf{x}$	matrices and vectors
$f \in \mathcal{H}$	function in reproducing kernel Hilbert space
$\langle \cdot, \cdot \rangle_{\mathcal{H}}$	inner product in $\mathcal{H}$
$K_{\mathbf{x}}$ or K_m	row of the Kernel Matrix
$i = 1, \ldots, n$	number of instances
$j = 1, \ldots, m_i$	number of instances in Bag X_i
$j = 1, \ldots, d$	number of input dimensions
$m = 1, \ldots, M$	number of kernels
$m = 1, \ldots, F$	number of features
$c = 1, \ldots, \mathcal{C}$	number of classes
Θ	set of kernel parameters
Θ_f	set of kernel parameters with finite size
L	loss function
$\mathcal{L}$	Lagrangian
Ω	regularization operator

B. Multiple Kernel Learning Dual

For easier reference we restate the primal formulation of the multiple kernel learning formulation (4.27) using the Hinge loss and the regularizer $\Omega(f) = \sum_{\theta\in\Theta} d_\theta \|w_\theta\|^2$. The convex problem (using the substitution $v_\theta = d_\theta w_\theta$) is

$$\begin{aligned}
\min_{v_\theta, b, d_\theta} \quad & \frac{1}{2}\sum_{\theta\in\Theta}\frac{1}{d_\theta}\|v_\theta\|^2 + C\sum_{i=1}^{n}\xi_i && \text{(B-1)}\\
\text{sb.t.} \quad & \sum_{\theta\in\Theta}\langle v_\theta, x_i\rangle + b \geq 1 - \xi_i, \quad i = 1,\ldots,n, && \text{(B-2)}\\
& \xi_i \geq 0, \quad i = 1,\ldots,n, && \text{(B-3)}\\
& d_\theta \geq 0, \quad \theta \in \Theta, && \text{(B-4)}\\
& \sum_{\theta\in\Theta} d_\theta = 1. && \text{(B-5)}
\end{aligned}$$

We write the Lagrangian of this problem

$$\begin{aligned}
\mathcal{L}(v_\theta, b, d_\theta, \xi, \alpha, \sigma) \;=\; & \frac{1}{2}\sum_{\theta\in\Theta}\frac{1}{d_\theta}\|v_\theta\|^2 + C\sum_{i=1}^{n}\xi_i + \lambda\left(\sum_{\theta\in\Theta} d_\theta - 1\right) && \text{(B-6)}\\
& - \sum_{\theta\in\Theta}\delta_\theta d_\theta - \sum_{i=1}^{n}\eta_i\xi_i && \text{(B-7)}\\
& - \sum_{i=1}^{n}\alpha_i\left(y_i\left(\sum_{\theta\in\Theta}\langle v_\theta, \phi_\theta(x_i)\rangle + b\right) + \xi_i - 1\right) && \text{(B-8)}
\end{aligned}$$

and take the derivatives with respect to the primal variables,

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial d_\theta} \;&=\; -\frac{1}{2d_\theta^2}\|v_\theta\|^2 + \lambda - \delta_\theta, && \text{(B-9)}\\
\frac{\partial\mathcal{L}}{\partial v_\theta} \;&=\; \frac{1}{d_\theta}v_\theta - \sum_{i=1}^{n}\alpha_i y_i \phi_\theta(x_i), && \text{(B-10)}\\
\frac{\partial\mathcal{L}}{\partial b} \;&=\; -\sum_{i=1}^{n}\alpha_i y_i, && \text{(B-11)}\\
\frac{\partial\mathcal{L}}{\partial \xi_i} \;&=\; C - \alpha_i - \eta_i. && \text{(B-12)}
\end{aligned}$$

Resubstituting and imposing non-negativity on the Lagrange multipliers corresponding to an inequality constraint, we arrive at the dual problem with a

particularly simple form

$$\max_{\alpha,\lambda} \quad \sum_{i=1}^{n} \alpha_i - \lambda \tag{B-13}$$

$$\text{sb.t.} \quad \alpha \in \mathbb{R}^n, \tag{B-14}$$

$$\lambda \in \mathbb{R}, \tag{B-15}$$

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n, \tag{B-16}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \tag{B-17}$$

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j; \theta) \leq \lambda, \quad \forall \theta \in \Theta. \tag{B-18}$$

C. Proof of Theorem 6

In this Section we present the proof of Theorem 4.5.1.

Theorem C.1. ***[Het93, Theorem 7.2]*** *If the subproblem can be solved, Algorithm 6 either stops after a finite number of iterations, or has at least one point of accumulation and each one of these points solve (IKL-dual).*

Proof. First we note that since $\alpha_i, i = 1, \ldots, n$ can take on values only in the compact set $\alpha_i \in [0, C]$, a point of accumulation always exists. Assume w.l.o.g. that $\alpha^t \overset{t\to\infty}{\to} \bar{\alpha}$. It remains to show that $\bar{\alpha}$ solves the problem. For notational convenience we use denote by

$$v(\alpha) = \max_{\theta\in\Theta} T(\theta; \alpha) - \lambda, \tag{C-1}$$

the maximum of the subproblem function. Assume that $\bar{\alpha}$ is not a feasible points, i.e. it does violate a constraint

$$v(\bar{\alpha}) = \max_{\theta\in\Theta} T(\theta; \bar{\alpha}) - \lambda > 0. \tag{C-2}$$

Suppose at iteration t of the algorithm we have identified θ^t as the maximal violating constraint, which implies

$$T(\theta^t; \alpha^t) - \lambda > 0 \quad \text{and} \quad T(\theta^t; \alpha^t) \geq T(\theta; \alpha^t)\ \forall \theta \in \Theta. \tag{C-3}$$

Now we have

$$\begin{aligned} v(\bar{\alpha}) &= v(\alpha^t) + v(\bar{\alpha}) - v(\alpha^t) && \text{(C-4)}\\ &= T(\theta^t; \alpha^t) - \lambda + v(\bar{\alpha}) - v(\alpha^t) && \text{(C-5)}\\ &\leq \left(T(\theta^t; \alpha^t) - T(\theta^t; \bar{\alpha})\right) + \left(v(\bar{\alpha}) - v(\alpha^t)\right), && \text{(C-6)} \end{aligned}$$

where we used the fact that $T(\theta^t; \bar{\alpha}) - \lambda \leq 0$, since θ^t is a constraint that is satisfied for $\bar{\alpha}$. Both T and v are continuous and therefore the right hand side can be made arbitrarily small. This contradicts the assumption $v(\bar{\alpha}) > 0$. □

Bibliography

[All00] Allwein, E. L., Schapire, R. E., and Singer, Y. Reducing multiclass to binary: a unifying approach for margin classifiers. In P. Langley, ed., *Proceedings of the 16th International Conference on Machine Learning (ICML)*, vol. 17, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.

[Alt04] Altun, Y., Smola, A. J., and Hofmann, T. Exponential Families for Conditional Random Fields. In *Proceedings of the 20th Conference Uncertainty in AI (UAI)*. AUAI Press, Arlington, Virginia, 2004.

[An05] An, L. T. H. and Tao, P. D. The DC (Difference of Convex Functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems. *Annals of Operations Research*, vol. 133:23–46, 2005.

[And03] Andrews, S., Tsochantaridis, I., and Hofmann, T. Support Vector Machines for Multiple-Instance Learning. In S. Becker, S. Thrun, and K. Obermayer, eds., *Advances in Neural Information Processing Systems 15*, pages 561–568. MIT Press, Cambridge, MA, 2003.

[Arg06] Argyriou, A., Hauser, R., Micchelli, C. A., and Pontil, M. A DC-programming algorithm for kernel selection. In W. W. Cohen and A. Moore, eds., *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, vol. 148 of *ACM International Conference Proceeding Series*, pages 41–48. ACM, New York, NY, USA, 6 2006.

[Aro50] Aronszajn, N. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, vol. 68:337–404, 1950.

[Bac04] Bach, F. R., Lanckriet, G. R. G., and Jordan, M. I. Multiple kernel learning, conic duality, and the SMO algorithm. In C. E. Brodley, ed., *Proceedings of the 21st International Conference on Machine Learning (ICML)*, vol. 69, page 6. ACM, 7 2004.

[Bak07] Bakir, G., Hofmann, T., Schölkopf, B., Smola, A., Taskar, B., et al. *Predicting Structured Data*. MIT Press, Cambridge, Massachusetts, 2007.

[Bar05] Baram, Y. Learning by Kernel Polarization. *Neural Computation*, vol. 17(6):1264–1275, 2005.

[BD00] Ben-David, S., Eiron, N., and Long, P. M. On the Difficulty of Approximately Maximizing Agreements. In *Proc. 13th Annu. Conference on Comput. Learning Theory*, pages 266–274. Morgan Kaufmann, San Francisco, 2000.

[Ben00] Bennett, K. P., Demiriz, A., and Shawe-Taylor, J. A Column Generation Algorithm for Boosting. In P. Langley, ed., *Proceedings of the 16th International Conference on Machine Learning (ICML)*, vol. 17, pages 65–72. Morgan Kaufmann, San Francisco, CA, 2000.

[Ber84] Berg, C., Christensen, J. P. R., and Ressel, P. *Harmonic Analysis on Semigroups*. Springer, New York, 1984.

[Bi04] Bi, J., Zhang, T., and Bennett, K. P. Column-generation boosting methods for mixture of kernels. In *KDD*. 2004.

[Bla96] Blanz, V., Schölkopf, B., Bülthoff, H., Burges, C., Vapnik, V., et al. Comparison of view-based object recognition algorithms using realistic 3D models. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, eds., *Artificial Neural Networks ICANN'96*, vol. 1112 of *Lecture Notes in Comput. Sci.*, pages 251–256. Springer-Verlag, Berlin, 1996.

[Bla08] Blaschko, M. B. and Lampert, C. H. Learning to Localize Objects with Structured Output Regression. In D. A. Forsyth, P. H. Torr, and A. Zisserman, eds., *Proceedings of the 10th European Conference on Computer Vision (ECCV)*, pages 2–15. Springer, Berlin, Germany, 10 2008.

[Bos07a] Bosch, A., Zisserman, A., and Muñoz, X. Image Classification using Random Forests and Ferns. In *Proceedings of the 11th International Conference on Computer Vision, (ICCV)*. IEEE Computer Society, 10 2007.

[Bos07b] Bosch, A., Zisserman, A., and Muñoz, X. Representing shape with a spatial pyramid kernel. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408. 2007.

[Bos08] Bosch, A., Zisserman, A., and Muñoz, X. Image Classification Using ROIs and Multiple Kernel Learning. *International Journal of Computer Vision*, 2008. (submitted June 2008), `http://eia.udg.es/~aboschr/Publicacions/bosch08a_preliminary.pdf`.

[Bou03] Bousquet, O. and Herrmann, D. On the Complexity of Learning the Kernel Matrix. In S. Becker, S. Thrun, and K. Obermayer, eds.,

Advances in Neural Information Processing Systems 15. MIT Press, Cambridge, MA, 2003.

[Boy04] Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

[Bro00] Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., et al. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci.*, vol. 97:262–267, 2000.

[Bur98] Burl, M. C., Weber, M., and Perona, P. A Probabilistic Approach to Object Recognition Using Local Photometry and Global Geometry. In *Proceedings of the 5th European Conference on Computer Vision (ECCV)*, vol. 2 of *Lecture Notes in Computer Science*, pages 628–641. Springer, London, UK, 1998.

[Cai04] Cai, L. and Hofmann, T. Hierarchical Document Categorization with Support Vector Machines. In *Proceedings of the Thirteenth ACM conference on Information and knowledge management*, pages 78–87. ACM Press, New York, NY, USA, 2004.

[cal07] Caltech-256 classification challenge, 2007. Challenge website: `http://www.vision.caltech.edu/CaltechChallenge2007/`.

[Cen] Center for Image Processing Research, Rensselaer Polytechnic Institute. Brodatz dataset of textures. `http://www.cipr.rpi.edu/resource/stills/brodatz.html`.

[Cha99] Chapelle, O., Haffner, P., and Vapnik, V. SVMs for histogram-based image classification. *IEEE Transactions on Neural Networks*, vol. 10(5):1055–1064, 1999.

[Cha01] Chang, C.-C. and Lin, C.-J. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Cha02] Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. Choosing Multiple Parameters for Support Vector Machines. *Machine Learning*, vol. 46(1-3):131–159, 2002.

[Cha07] Chapelle, O. Training a Support Vector Machine in the Primal. *Neural Computation*, vol. 19(5):1155–1178, 03 2007.

[Cha08] Chapelle, O. and Rakotomamonjy, A. Second order optimization of kernel parameters. In *Proceedings of the NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels (LK ASOK 2008)*. 12 2008.

[Che06a] Chen, A. Fast Kernel Density Independent Component Analysis. In *Proceedings of 6th international conference on ICA and BSS*, vol. 3889 of *Lecture Notes in Computer Science*, pages 24– 31. Springer, Heidelberg, 2006.

[Che06b] Chen, Y., Bi, J., and Wang, J. Z. MILES: Multiple-Instance Learning via Embedded Instance Selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28(12), 2006.

[Che06c] Cheung, P.-M. and Kwok, J. T. A regularization framework for multiple-instance learning. In W. W. Cohen and A. Moore, eds., *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, vol. 148 of *ACM International Conference Proceeding Series*, pages 193–200. ACM, New York, NY, USA, 6 2006.

[Col02] Collins, M. Discriminative training methods for Hidden Markov Models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, vol. 10, pages 1–8. Association for Computational Linguistics, Morristown, NJ, USA, 2002.

[Cor95] Cortes, C. and Vapnik, V. Support Vector Networks. *Machine Learning*, vol. 20(3):273–297, 1995.

[Cox90] Cox, D. D. and O'Sullivan, F. Asymptotic Analysis of Penalized Likelihood and Related Estimators. *The Annals of Statistics*, vol. 18(4):1676–1695, 1990.

[Cra00] Crammer, K. and Singer, Y. On the Learnability and Design of Output Codes for Multiclass Problems. In N. Cesa-Bianchi and S. Goldman, eds., *Proc. Annual Conf. Computational Learning Theory*, pages 35–46. Morgan Kaufmann Publishers, San Francisco, CA, 2000.

[Cra03] Crammer, K., Keshet, J., and Singer, Y. Kernel Design Using Boosting. In S. Becker, S. Thrun, and K. Obermayer, eds., *Advances in Neural Information Processing Systems 15*, pages 537–544. MIT Press, Cambridge, MA, 2003.

[Cri02] Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. On Kernel-Target Alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, eds., *Advances in Neural Information Processing Systems 14*, pages 367–373. MIT Press, Cambridge, MA, 2002.

[CV07] Camps-Valls, G., Rojo-Álvarez, J. L., and Martínez-Ramón, M. *Kernel Methods in Bioengineering, Signal and Image Processing.* Idea Group, 2007.

[DeC02] DeCoste, D. and Schölkopf, B. Training invariant support vector machines. *Machine Learning*, vol. 46:161–190, 2002.

[Dem02] Demiriz, A., Bennett, K. P., and Shawe-Taylor, J. Linear Programming Boosting via Column Generation. *Journal of Machine Learning Research*, vol. 46:225–254, 2002.

[Die97a] Dietterich, T. G. Machine Learning Research: Four Current Directions. *AI Magazine*, vol. 18(4):97–136, 1997.

[Die97b] Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. Solving the Multiple Instance Problem with Axis-Parallel Rectangles. *Artif. Intell.*, vol. 89(1-2):31–71, 1997.

[Dru97] Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J., and Vapnik, V. Support vector regression machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, eds., *Advances in Neural Information Processing Systems 9*, pages 155–161. MIT Press, Cambridge, MA, 1997.

[Dua05] Duan, K.-B. and Keerthi, S. S. Which Is the Best Multiclass SVM Method? An Empirical Study. In *Multiple Classifier Systems*, vol. 3541 of *Lecture Notes in Computer Science*, pages 278–285. 2005.

[Dum98] Dumais, S. Using SVMs for Text Categorization. *IEEE Intelligent Systems*, vol. 13(4):18–28, 1998. In: M. A. Hearst, B. Schölkopf, S. Dumais, E. Osuna, and J. Platt: Trends and Controversies - Support Vector Machines.

[Eli02] Elisseeff, A. and Weston, J. A kernel method for multi-labeled classification. In T. G. Dietterich, S. Becker, and Z. Ghahramani, eds., *Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press, Cambridge, MA, 2002.

[Eve07] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007). `http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html`, 2007.

[Fer03] Fergus, R., Perona, P., and Zisserman, A. Object Class Recognition by Unsupervised Scale-Invariant Learning. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 264–271. IEEE Computer Society, 6 2003.

[FF04] Fei-Fei, L., Fergus, R., and Perona, P. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition*

Workshop (CVPRW'04) Volume 12, page 178. IEEE Computer Society, Washington, DC, USA, 2004.

[Fit95] FitzGerald, C. H., Micchelli, C. A., and Pinkus, A. Functions that preserve families of positive semidefinite matrices. *Linear Algebra Appl.*, vol. 221:83–102, 1995.

[Fra06] Franz, M. O. and Gehler, P. V. How to choose the covariance for Gaussian process regression independently of the basis. In *Workshop Gaussian Processes in Practice 2006*. videolectures.net, Scottsdale, AZ, USA, 06 2006.

[Fri04] Fritz, M., Hayman, E., Caputo, B., and Eklundh, J.-O. `http://www.nada.kth.se/cvap/databases/kth-tips/documentation.html`, 2004.

[Gär02] Gärtner, T., Flach, P. A., Kowalczyk, A., and Smola, A. J. Multi-Instance Kernels. In C. Sammut and A. G. Hoffmann, eds., *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 179–186. Morgan Kaufmann, 7 2002.

[Geh06a] Gehler, P. V. and Franz, M. Implicit Wiener Series, Part II: Regularised estimation. Tech. Rep. 148, Max Planck Institute for Biological Cybernetics, 11 2006.

[Geh06b] Gehler, P. V., Holub, A. D., and Welling, M. The Rate Adapting Poisson Model for Information Retrieval and Object Recognition. In W. W. Cohen and A. Moore, eds., *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, vol. 148 of *ACM International Conference Proceeding Series*. ACM, New York, NY, USA, 6 2006.

[Geh07] Gehler, P. V. and Chapelle, O. Deterministic Annealing for Multiple Instance Learning. In M. Meila and X. Shen, eds., *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Microtome, Brookline, MA, USA, 03 2007.

[Geh08a] Gehler, P. V. and Nowozin, S. Infinite Kernel Learning. Tech. Rep. 178, Max-Planck Institute for Biological Cybernetics, 10 2008.

[Geh08b] Gehler, P. V. and Nowozin, S. Infinite Kernel Learning. In *Proceedings of the NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels (LK ASOK 2008)*. 12 2008.

[Geh08c] Gehler, P. V., Rother, C., Blake, A., Minka, T., and Sharp, T. Bayesian Color Constancy Revisited. In *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 06 2008.

[Geh09a] Gehler, P. V. and Nowozin, S. Let the Kernel Figure it Out: Principled Learning of Pre-processing for Kernel Classifiers. In *Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 06 2009.

[Geh09b] Gehler, P. V. and Schölkopf, B. *An Introduction to Kernel Learning Algorithms*, chap. 2, pages 39–60. John Wiley and Sons, 2009.

[Gij07] Gijsenij, A. and Gevers, T. Color Constancy using Natural Image Statistics. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Minneapolis, USA, 6 2007.

[Gra03] Grandvalet, Y. and Canu, S. Adaptive Scaling for Feature Selection in SVMs. In S. Becker, S. Thrun, and K. Obermayer, eds., *Advances in Neural Information Processing Systems 15*. MIT Press, Cambridge, MA, 2003.

[Gra05] Grauman, K. and Darrell, T. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *Proceedings of the 10th International Conference on Computer Vision, (ICCV)*. IEEE Computer Society, 2005.

[Gra07] Grauman, K. and Darrell, T. The Pyramid Match Kernel: Efficient Learning with Sets of Features. *Journal of Machine Learning Research*, pages 725–760, 2007.

[Gri07] Griffin, G., Holub, A., and Perona, P. Caltech-256 Object Category Dataset. Tech. Rep. 7694, California Institute of Technology, 2007.

[Hau99] Haussler, D. Convolutional Kernels on Discrete Structures. Tech. Rep. UCSC-CRL-99-10, Computer Science Department, UC Santa Cruz, 1999.

[Hay01] Hayton, P., Schölkopf, B., Tarassenko, L., and Anuzis, P. Support vector novelty detection applied to jet engine vibration spectra. In T. K. Leen, T. G. Dietterich, and V. Tresp, eds., *Advances in Neural Information Processing Systems 13*, pages 946–952. MIT Press, Cambridge, MA, 2001.

[Hei05] Hein, M. and Bousquet, O. Hilbertian metrics and positive definite kernels on probability measures. In Z. Ghahramani and R. Cowell, eds., *Proceedings of the 10th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 10. 2005.

[Her00] Herbrich, R., Graepel, T., and Obermayer, K. Large margin rank boundaries for ordinal regression. In A. J. Smola, P. L. Bartlett,

B. Schölkopf, and D. Schuurmans, eds., *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, Cambridge, MA, 2000.

[Het93] Hettich, R. and Kortanek, K. O. Semi-infinite programming: theory, methods, and applications. *SIAM Rev.*, vol. 35(3):380–429, 1993.

[Hoe70] Hoerl, A. E. and Kennard, R. W. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, vol. 12:55–67, 1970.

[Hof08] Hofmann, T., Schölkopf, B., and Smola, A. J. Kernel Methods in Machine Learning. *Annals of Statistics*, vol. 36:1171–1220, 6 2008.

[Hor96] Horst, R. and Tuy, H. *Global Optimization*. Springer-Verlag, Berlin, 3 edn., 1996.

[Jam61] Jameson, D. and Hurvich, L. Complexities of perceived brightness. *Science*, vol. 133:174–179, 1961.

[Joa98] Joachims, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142. Springer, Berlin, 1998.

[Joa99] Joachims, T. Making Large-Scale SVM Learning Practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds., *Advances in Kernel Methods - -Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA, 1999.

[Ke04] Ke, Y. and Sukthankar, R. PCA-SIFT: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pages 506–513. IEEE Computer Society, 2004.

[Kim71] Kimeldorf, G. S. and Wahba, G. Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, vol. 33:82–95, 1971.

[Kim05] Kim, K. I., Franz, M. O., and Schölkopf, B. Iterative Kernel Principal Component Analysis for Image Modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27(9):1351–1366, 09 2005.

[Klo08] Kloft, M., Brefeld, U., Laskov, P., and Sonnenburg, S. Non-sparse Multiple Kernel Learning. In *Proceedings of the NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels (LK ASOK 2008)*. 12 2008.

[Koh95] Kohavi, R. A study of Cross validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Neural Networks*. 1995.

[Kum07] Kumar, A. and Sminchisescu, C. Support Kernel Machines for Object Recognition. In *Proceedings of the 11th International Conference on Computer Vision, (ICCV)*. IEEE Computer Society, 10 2007.

[Lam08a] Lampert, C. and Blaschko, M. B. A Multiple Kernel Learning Approach to Joint Multi-Class Object Detection. In G. Rigoll, ed., *30th Annual Symposium of the German Association for Pattern Recognition*, pages 31–40. Springer, Berlin, Germany, 2008.

[Lam08b] Lampert, C. H., Blaschko, M. B., and Hofmann, T. Beyond Sliding Windows: Object Localization by Efficient Subwindow Search. In *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE Computer Society, 06 2008.

[Lan71] Land, E. and McCann, J. Lightness and retinex theory. *Journal of the Optical Society of America*, vol. 61:1–11, 1971.

[Lan04] Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. Learning the Kernel Matrix with Semidefinite Programming. *Journal of Machine Learning Research*, vol. 5:27–72, 2004.

[Laz05] Lazebnik, S., Schmid, C., and Ponce, J. A Sparse Texture Representation Using Local Affine Regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27(8):1265–1278, 2005.

[Laz06] Lazebnik, S., Schmid, C., and Ponce, J. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2169–2178. IEEE Computer Society, 2006.

[Laz07] Lazebnik, S. and Raginsky, M. Learning Nearest-Neighbor Quantizers from Labeled Data by Information Loss Minimization. In M. Meila and X. Shen, eds., *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Microtome, Brookline, MA, USA, 03 2007.

[LeC98] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, vol. 86(11):2278–2324, November 1998.

[Li05] Li, F.-F. and Perona, P. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 524–531. IEEE Computer Society, 2005.

[Lin07] Lin, Y.-Y., Liu, T.-L., and Fuh, C.-S. Local Ensemble Kernel Learning for Object Category Recognition. In *Proceedings of the 11th International Conference on Computer Vision, (ICCV)*. IEEE Computer Society, 10 2007.

[Low99] Lowe, D. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision, (ICCV)*, pages 1150–1157. IEEE Computer Society, 1999.

[Mac98] MacKay, D. J. C. Introduction to Gaussian Processes. In C. M. Bishop, ed., *Neural Networks and Machine Learning*, pages 133–165. Springer, Berlin, 1998.

[Man05] Mangasarian, O. L. and Wild, E. W. Multiple Instance Classification via Successive Linear Programming. Tech. Rep. 05-02, Data Mining Institute, University of Wisconsin, 2005.

[McC05] McCallum, A., Bellare, K., and Pereira, F. A Conditional Random Field for Discriminatively-trained Finite-state String Edit Distance. In *Proceedings of the 21st Conference Uncertainty in AI (UAI)*, page 388. AUAI Press, Arlington, Virginia, 2005.

[Mik05a] Mikolajczyk, K. and Schmid, C. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27(10):1615–1630, 2005.

[Mik05b] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., et al. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, vol. 65(1-2):43–72, 2005.

[Min69] Minsky, M. and Papert, S. *Perceptrons: An Introduction To Computational Geometry*. MIT Press, Cambridge, MA, 1969.

[Mor84] Morozov, V. A. *Methods for Solving Incorrectly Posed Problems*. Springer-Verlag, New York, 1984.

[Mur05] Murray, J. F., Hughes, G. F., and Kreutz-Delgado, K. Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application. *Journal of Machine Learning Research*, vol. 6:783–816, 2005.

[Mut06] Mutch, J. and Lowe, D. G. Multiclass Object Recognition with Sparse, Localized Features. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11–18. IEEE Computer Society, 2006.

[Nil06] Nilsback, M.-E. and Zisserman, A. A Visual Vocabulary for Flower Classification. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1447–1454. IEEE Computer Society, 2006.

[Nil08] Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*. Dec 2008.

[Now06] Nowak, E., Jurie, F., and Triggs, B. Sampling strategies for bag-of-features image classification. In A. Leonardis, H. Bischof, and A. Pinz, eds., *Proceedings of the 9th European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science. Springer, 2006.

[Oja02] Ojala, T., Pietikäinen, M., and Mäenpää, T. Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24(7):971–987, Jul. 2002.

[Ope06] Opelt, A., Fussenegger, M., and Auer, P. Generic Object Recognition with Boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28(3):416–431, 2006.

[Özö08] Özöğür-Akyüz, S. and Weber, G. W. Learning with Infinitely Many Kernels via Semi-Infinite Programming. In *Proceedings of Euro mini conference on "Continuous Optimization and Knowledge Based Technologies"*. 2008.

[Pin08] Pinto, N., Cox, D. D., and Dicarlo, J. J. Why is Real-World Visual Object Recognition Hard? *PLoS Computational Biology*, vol. 4(1):e27+, January 2008.

[Pla99] Platt, J. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds., *Advances in Kernel Methods - -Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999.

[Rak08] Rakotomamonjy, A., Bach, F., Grandvalet, Y., and Canu, S. SimpleMKL. *Journal of Machine Learning Research*, vol. 9:2491–2521, November 2008.

[Ras06] Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[Rät01] Rätsch, G. *Robust Boosting via Convex Optimization: Theory and Applications*. Ph.D. thesis, University of Potsdam, 2001.

[Rät03] Rätsch, G., Mika, S., and Smola, A. J. Adapting Codes and Embeddings for Polychotomies. In S. T. S. Becker and K. Obermayer, eds., *Advances in Neural Information Processing Systems 15*, pages 513–520. MIT Press, Cambridge, MA, 2003.

[Ray05] Ray, S. and Craven, M. Supervised versus multiple instance learning: an empirical comparison. In L. D. Raedt and S. Wrobel, eds., *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, vol. 119, pages 697–704. ACM, 8 2005.

[Rom01] Romdhani, S., Torr, P., Schölkopf, B., and Blake, A. Fast face detection, using a sequential reduced support vector evaluation. In *Proceedings of the 8th International Conference on Computer Vision, (ICCV)*. IEEE Computer Society, Los Alamitos, CA, 2001.

[Ros98] Rose, K. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In *Proceedings of IEEE*, vol. 86, pages 2210–2239. 1998.

[Ros04] Rosenberg, C., Minka, T., and Ladsariya, A. Bayesian Color Constancy with Non-Gaussian Models. In S. Thrun, L. Saul, and B. Schölkopf, eds., *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[Rub00] Rubner, Y., Tomasi, C., and Guibas, L. J. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, vol. 40(2):99–121, 2000.

[Sch97] Schölkopf, B. *Support Vector Learning*. R. Oldenbourg Verlag, Munich, 1997. Download: http://www.kernel-machines.org.

[Sch98] Schölkopf, B., Smola, A. J., and Müller, K.-R. Nonlinear component analysis as a kernel Eigenvalue problem. *Neural Comput.*, vol. 10:1299–1319, 1998.

[Sch01] Schölkopf, B., Herbrich, R., and Smola, A. J. A Generalized Representer Theorem. In D. P. Helmbold and B. Williamson, eds., *Proc. Annual Conf. Computational Learning Theory*, no. 2111 in Lecture Notes in Comput. Sci., pages 416–426. Springer-Verlag, London, UK, 2001.

[Sch02] Schölkopf, B. and Smola, A. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[Sha04] Shawe-Taylor, J. and Cristianini, N. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.

[Shi00] Shi, J. and Malik, J. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22(8):888–905, 2000.

[Son06] Sonnenburg, S., Rätsch, G., Schäfer, C., and Schölkopf, B. Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research*, 2006.

[Tao04] Tao, Q., Scott, S., Vinodchandran, N. V., and Osugi, T. T. SVM-based generalized multiple-instance learning via approximate box counting. In C. E. Brodley, ed., *Proceedings of the 21st International Conference on Machine Learning (ICML)*, vol. 69, page 101. ACM, 7 2004.

[Tik63] Tikhonov, A. N. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, vol. 4:1035–1038, 1963.

[Tso05] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research*, vol. 6:1453–1484, 2005.

[Tuz07] Tuzel, O., Porikli, F., and Meer, P. Human Detection via Classification on Riemannian Manifolds. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Minneapolis, USA, 6 2007.

[Vap63] Vapnik, V. and Lerner, A. Pattern Recognition using Generalized Portrait Method. *Autom. Remote Control*, vol. 24:774–780, 1963.

[Vap95] Vapnik, V. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[Vap97] Vapnik, V., Golowich, S., and Smola, A. J. Support vector method for function approximation, regression estimation, and signal processing. In M. C. Mozer, M. I. Jordan, and T. Petsche, eds., *Advances in Neural Information Processing Systems 9*, pages 281–287. MIT Press, Cambridge, MA, 1997.

[Var07] Varma, M. and Ray, D. Learning The Discriminative Power-Invariance Trade-Off. In *Proceedings of the 11th International Conference on Computer Vision, (ICCV)*. IEEE Computer Society, 10 2007.

[Ved08] Vedaldi, A. and Soatto, S. Relaxed Matching Kernels for Object Recognition. In *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 06 2008.

[vG08] van Gemert, J., Geusebroek, J.-M., Veenman, C. J., and Smeulders, A. W. M. Kernel Codebooks for Scene Categorization. In D. A. Forsyth, P. H. Torr, and A. Zisserman, eds., *Proceedings of the 10th European Conference on Computer Vision (ECCV)*, pages 696–709. Springer, Berlin, Germany, 10 2008.

[Vio06] Viola, P., Platt, J., and Zhang, C. Multiple Instance Boosting for Object Detection. In Y. Weiss, B. Schölkopf, and J. Platt, eds., *Advances in Neural Information Processing Systems 18*, pages 1417–1424. MIT Press, Cambridge, MA, 2006.

[Wäc06] Wächter, A. and Biegler, L. T. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, vol. 106(1):25–57, 2006.

[Wah90] Wahba, G. *Spline Models for Observational Data*, vol. 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

[Wan02] Wang, J. and Zucker, J.-D. Solving the Multiple-Instance Problem: A Lazy Learning Approach. In C. Sammut and A. G. Hoffmann, eds., *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 1119–1126. Morgan Kaufmann, 7 2002.

[Wan06] Wang, G., Zhang, Y., and Fei-Fei, L. Using dependent regions for object categorization in a generative framework. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2006.

[Web00] Weber, M., Welling, M., and Perona, P. Towards automatic discovery of object categories. In *Proceedings of the 2000 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 101–108. IEEE Computer Society, 2000.

[Wes99] Weston, J. and Watkins, C. Support vector machines for multi-class pattern recognition. In *ESANN*, pages 219–224. 1999.

[Yui02] Yuille, A. L. and Rangarajan, A. The Concave-Convex Procedure. In T. G. Dietterich, S. Becker, and Z. Ghahramani, eds., *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA, 2002.

[Zha02] Zhang, Q., Goldman, S. A., Yu, W., and Fritts, J. Content-Based Image Retrieval Using Multiple-Instance Learning. In C. Sammut and A. G. Hoffmann, eds., *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 682–689. Morgan Kaufmann, 7 2002.

[Zha06] Zhang, H., Berg, A. C., Maire, M., and Malik, J. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2126–2136. IEEE Computer Society, 2006.

[Zie07] Zien, A. and Ong, C. S. Multiclass Multiple Kernel Learning. In Z. Ghahramani, ed., *Proceedings of the 24th International Conference on Machine Learning (ICML)*, vol. 227 of *ACM International Conference Proceeding Series*. ACM, 6 2007.