Kernel Fisher Discriminants

Kernel Fisher Discriminants

vorgelegt von
Dipl. Inform. Sebastian Mika
aus Berlin

Von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften – Dr. rer. nat. –
genehmigte Dissertation

Promotionsaussschuß:

- Vorsitzender: Prof. Dr. Dieter Filbert
- Berichter: Prof. Dr. Stefan Jähnichen
- Berichter: Prof. Dr. Klaus–Robert Müller
- Berichter: Prof. Dr. Bernhard Schölkopf

Tag der wissenschaftlichen Aussprache: 19. Dezember 2002

Berlin
D83

Abstract

In this thesis we consider statistical learning problems and machines. A statistical learning machine tries to infer rules from a given set of examples such that it is able to make correct predictions on unseen examples. These predictions can for example be a classification or a regression. We consider the class of *kernel* based learning techniques. The main contributions of this work can be summarized as follows.

Building upon the theory of reproducing kernels we propose a number of new learning algorithms based on the maximization of a Rayleigh coefficient in a kernel feature space. We exemplify this for oriented (kernel) PCA, and especially for Fisher's discriminant, yielding kernel Fisher discriminants (KFD).

Furthermore, we show that KFD is intimately related to quadratic and linear optimization. Building upon this connection we propose several ways to deal with the optimization problems arising in kernel based methods and especially for KFD.

This mathematical programming formulation is the starting point to derive several important and interesting variants of KFD, namely robust KFD, sparse KFD and linear KFD. Several algorithms to solve the resulting optimization problems are discussed. As a last consequence of the mathematical programming formulation we are able to relate KFD to other techniques like support vector machines, relevance vector machines and Arc-GV. Through a structural comparison of the underlying optimization problems we illustrate that many modern learning techniques, including KFD, are highly similar.

In a separate chapter we present first results dealing with learning guarantees for eigenvalues and eigenvectors estimated from covariance matrices. We show that under some mild assumptions empirical eigenvalues are with high probability close to the expected eigenvalues when training on a specific, finite sample size. For eigenvectors we show that also with high probability an empirical eigenvector will be close to an eigenvector of the underlying distribution.

In a large collection of experiments we demonstrate that KFD and its variants proposed here are capable of producing state of the art results. We compare KFD to techniques like AdaBoost and support vector machines, carefully discussing its advantages and also its difficulties.

Keywords: Learning, Classification, Kernel Methods, Fisher's Discriminant, Regularization, Eigenproblems, Rayleigh coefficients, Mathematical Programming

Zusammenfassung

Diese Doktorarbeit beschäftigt sich mit statistischer Lerntheorie und statistischen Lernmaschinen. Eine statistische Lernmaschine versucht aus gegebenen Beispielen eine Regel zu inferieren mit der man in der Lage ist auf ungesehenen Beispielen korrekte Vorhersagen zu machen. Diese Vorhersagen sind reellwertig (Regression) oder diskret (Klassifikation). Wir betrachten insbesondere so genannte kernbasierte Lernverfahren. Die Hauptbeiträge dieser Arbeit lassen sich wie folgt zusammenfassen:

Aufbauend auf der Theorie der reproduzierenden Kerne schlagen wir neue Lernmaschinen vor, die auf der Maximierung eines Rayleigh Koeffizienten in einem Kernmerkmalsraum basieren. Wir machen dies beispielhaft für orientierte (Kern) Hauptkomponentenanalyse und insbesondere für Fishers Diskriminanten, was in Kern Fisher Diskriminanten (KFD) resultiert.

Dann wird gezeigt, daß KFD eng mit quadratischer und linearer Optimierung verbunden ist. Darauf aufbauend werden verschiedene Möglichkeiten diskutiert mit den Optimierungsproblemen umzugehen die bei kernbasierten Methoden und insbesondere KFD entstehen.

Die Formulierung als mathematisches Optimierungsproblem ist der Ausgangspunkt um verschiedene wichtige und interessante Varianten von KFD herzuleiten: Robuste KFD, sparse KFD und lineare KFD. Die mathematische Optimierung ermöglicht es darüber hinaus KFD mit anderen Techniken wie Support Vektor Maschinen, der Relevanzvektormethode und Boosting in Verbindung zu setzen. Durch einen strukturellen Vergleich der zugrundeliegenden Optimierungsprobleme wird illustriert, daß viele moderne Lernmethoden, auch KFD, sich sehr ähneln.

Außerdem präsentieren wir erste Ergebnisse über Lerngarantien für Eigenwerte und Eigenvektoren die aus Kovarianzmatrizen geschätzt werden. Wir zeigen, daß unter schwachen Annahmen, die empirischen Eigenwerte mit hoher Wahrscheinlichkeit nahe an den zu erwartenden Eigenwerten liegen. Für Eigenvektoren zeigen wir, daß mit hoher Wahrscheinlichkeit ein empirischer Eigenvektor nahe zu einem Eigenvektor der zugrundeliegenden Verteilung sein wird.

In einer großen Sammlung von Experimenten wird demonstriert, daß KFD und seine Varianten sehr gut in der Lage sind mit dem technischen Standard zu konkurrieren. Wir vergleichen KFD mit Boosting und Support Vektor Maschinen und diskutieren sorgfältig die Vor- und Nachteile der vorgeschlagenen Methoden.

Keywords: Lernen, Klassifikation, Kerne, Fishers Diskriminanten, Regularisierung, Eigenprobleme, Rayleigh Koeffizienten, Mathematische Programmierung

Acknowledgments

Writing this thesis would not have been possible with invaluable help and support from others, let it be financial, intellectual or personal. First of all I would like to thank my supervisors Prof. Dr. Klaus–Robert Müller, Prof. Dr. Bernhard Schölkopf, and Prof. Dr. Stefan Jähnichen. All gave their best in providing me with a stimulating and relaxed environment. Many ideas in this thesis are based on discussions I had with Klaus and Bernhard. However, their support was much more than just scientific! Similarly, I would like to thank Dr. Gunnar Rätsch with whom I shared the office for almost four years. Much of his energy, motivation and of the long discussions we had can be found in this work. I guess without him my thesis would not be half of what it is now.

Most of the work in this thesis has been done at Klaus-Robert's group at the Fraunhofer FIRST institute in Berlin. I deeply appreciate the support of all current and former members of this group for creating a pleasant atmosphere, including Benjamin Blankertz, Guido Dornhege, Stefan Harmeling, Motoaki Kawanabe, Jens Kohlmorgen, Roman Krepki, Julian Laub, Pavel Laskov, Frank Meinecke, Alex Smola, David Tax, Koji Tsuda, Ricardo Vigario, Thorsten Zander, and Andreas Ziehe. Thanks to you all for bearing me all the time! Special thanks also to Steven Lemm and Christin Schäfer for sharing the office, their thoughts and their chocolate with me and to Soeren Sonnenburg: keep it going!

Moreover I would like to thank Fraunhofer FIRST for supporting my studies by providing numerous hours of computing time and a like. Thanks also to Gabi Ambach at University of Technology for fixing up the legal stuff and her patience with me and the forms.

Other parts of this thesis were done during research visits at AT&T (Red Bank, New Jersey), Microsoft Research (Cambridge, UK), and the Australian National University (Canberra, Australia). Thanks for making these stays possible to Bernhard Schölkopf (Microsoft) and Alex Smola and Bob Williamson (ANU). To everybody I met there thanks for fruitful and encouraging discussions, thanks for sharing your knowledge and ideas with me.

There are some other people in the research community which helped to make it a pleasure being a part of it. Thanks to all the french guys: Oilvier Bousquet, Olivier Chapelle (hope you get well soon) and Andre Elisseeff. Special thanks to Jason Weston for showing me how much fun science can be and that it is important to take not everything too serious.

I gratefully acknowledge financial support from DFG grants JA 379/71, JA 379/91 and JA 379/92, from the EC NeuroCOLT project, and from the DAAD.

Last but not least there is number of people which have nothing to do with my research. But even then, they provided an evenly important share in making it possible. I would like to thank my parents for their love and care. My brother Christian, his girl-friend Katy and my sweet little Lilith helped me in giving a meaning to everything. Life would have been different without my flat mates Ingmar and Matthias - probably cleaner but very boring! Love to those of which I have the feeling you have been there all my life and will never be far (even though I know that this is an illusion): Lisa, Rinetta, Martin & Laura, Linda, Philipp, Malte & Nina, Steffi & Kristin, Justin, Christiane, and all the others I forgot - I did not have much time in making this up :-)

Sebastian Mika, October 2002.

Contents

Chapter 1

Introduction – Machines that Learn

Schwur: Müde vom Durchwandern öder Letternwüsten, voll leerer Hirngeburten, in anmaßendsten Wortnebeln; überdrüssig ästhetischer Süßler wie grammatischer Wässrer; entschloß ich mich: *Alles, was je schrieb, in Liebe und Haß, als immerfort mitlebend zu behandeln!* - - - 20.9.1958/Darmstadt i. d. Barberei

Arno Schmidt

In this introduction we will try to motivate the issues addressed in this work. The aim of this first chapter is to give the reader not familiar with the field a glimpse of the basic problems and ideas. However, this introduction is neither complete nor does it even approximately give a balanced overview of the complete field.

THIS thesis is about machine learning or, more generally, artificial intelligence. Machine learning, as the name suggests, tries to devise algorithms that solve complex tasks by learning a solution rather than by engineering it. Often the latter is very difficult for practical problems. A very recent but typical example is the analysis of the human genome. In principle, at least this is widely believed, there exists a fixed model how the genome codes for information and how this information is translated into something useful. It is known that only parts of the DNA are used, the genes. But it is not fully known yet where the genes are and how to (exactly) locate them. There exist some biological models but they are very complex and only partially explain what is going on. Machine learning approaches this type of problem differently. Instead of trying to find a model which explains the underlying processes correctly and can then be used to deduce an answer, one tries to find a rule which is able for each piece of DNA to answer the question:

Is this a gene or not? More generally, machine learning considers the problem to model relationships between objects. The art of machine learning is, however, to build machines that are able to do this without needing a lot of expert knowledge built-in.

Machine learning provides a framework to solve a large number of different problems, rather than trying to devise an algorithm that is able to solve one specific problem. Hence, the complexity of the engineering problem is shifted from solving a difficult, but specific problem, towards solving a still difficult but more general problem: How to make a machine learn? The advantage is clear: Once we solved the learning problem we can solve a host of other problems using the technology we developed.

1.1 Learning from Examples - Intuitively

There are many different areas in artificial intelligence and machine learning is one sub-area. The paradigm we follow here is called *Learning from Examples*. The idea is that often a few examples of relations between objects can suffice to figure out a general rule for relating them. The learning process is exactly this inductive inference, from the incomplete information provided by the example to the prediction rule describing certain aspects of the phenomenon. These rules are often conveniently modeled by functions which map input objects to output objects. Depending on the nature of especially the output objects we speak for example about classification or regression. The classification task is to assign each input object to one of finitely many classes. In regression we assign one or more usually continuous valued outputs to each input object. In both cases, the information provided for learning consists of finitely many example-label pairs.

Whilst it is somewhat disappointing that much in machine learning is largely concerned with function estimation, this way of approaching the learning problem bears some fundamental advantages. A rigorous mathematical formulation allows to use statistics and other tools to investigate under which conditions learning is possible. The field of statistical learning theory (e.g. Vapnik, 1998) investigates what the most important properties of learning algorithms and functions are and how they relate to the ability to successfully solve an estimation problem. This makes it possible to derive learning techniques that optimize these quantities and achieve improved results. Moreover, the theory is capable of giving guarantees for the number of mistakes that a certain method will commit. An important difference between modern machine learning research and the early AI research is that these results hold true for practical cases and not only in some asymptotical sense.

In reducing the learning problem to that of estimating functional dependencies, we are left with a complex task. Besides theoretical limits, there are many practical ramifications making learning difficult, where one is that we are only provided with a limited amount of data to learn from (the training data). Hence the problem becomes to estimate the true dependencies from this limited information. Often it is not possible to generate more data, either because gathering each example is very expensive (e.g. in DNA analysis) or because it is just impossible (monitoring malfunctions of a machine - if it does not malfunction we can not get our hands

on more examples). The converse can also be a problem. If the amount of information is too large many techniques ran into intractability problems (e.g. web search engines). Other problems include that the training examples or the data describing the problem are not complete, i.e. not in all cases all information is available, or the information provided might be inhomogeneous, i.e. it differs from trial to trial. A central problem in function estimation is, of course, that we have to expect that the information provided is inaccurate, e.g. corrupted be measurement noise.

Today machine learning research is very much concerned with solving these types of problems, i.e. how to make learning practical for small training sets, how to deal with noise and outliers (errors), or how to handle missing values. Especially in the past decade there have been big advances in solving these problems.

What we consider in this thesis are learning algorithms that are based upon Rayleigh coefficients, especially Fisher's discriminant, and that use kernel functions. Fisher's discriminant (Fisher, 1936) is a technique to find *linear* functions that are able to discriminate between two or more classes. Being a technique that is around for almost 70 years it is well known and widely used to build classifiers. However, many modern learning problems are not appropriately solvable using linear techniques. Therefore, we propose non-linear variants of Fisher's discriminant. This non-linearization is made possible through the use of kernel functions, a "trick" that is borrowed from support vector machines (Boser et al., 1992), in a way the successor of Rosenblatt's perceptron (Rosenblatt, 1956). Kernel functions represent a very principled and elegant way of formulating non–linear algorithms. The methods we derive have clear and intuitive interpretations. Moreover, although these techniques are able to describe highly non–linear relations we will be able to devise algorithms that find globally optimal solutions in polynomial time. We show, that the learning machines derived this way have state of the art performance.

1.1.1 How this thesis is organized

This thesis is divided into four major parts. We start in Chapter 1 by reviewing some basic tools and notations from statistical learning theory. We discuss the concept of consistency and introduce the structural risk minimization principle (Vapnik and Chervonenkis, 1974). We also review some theory of reproducing kernels and how they enable us to turn a linear algorithm into a non-linear one. The reader already familiar with these concepts may skip this part.

In Chapter 3 we derive the kernel Fisher discriminant (KFD) algorithm and several variations. We also discuss how KFD relates to mathematical optimization. From this connection we are able to propose several strategies to solve the arising optimization problems efficiently. We conclude this chapter by showing how KFD relates to other techniques like support vector machines, relevance vector machines (Tipping, 2000) and a variant of Boosting (Freund and Schapire, 1997).

Chapter 4 presents some first results on building a theory around KFD. There is an intimate connection between KFD and the estimation of eigenvectors and eigenvalues from covariance matrices. Building upon this connection, we give guarantees for estimates of these quantities.

Finally, Chapter 5 presents a large collection of illustrative and real–world applications of KFD. We demonstrate certain aspects of the proposed techniques and

show that they are readily usable in applications, yielding state of the art results.

1.1.2 A little bit of history

Machine learning and artificial intelligence are active research areas since over fifty years. However, most people only have a very vague idea what machine learning precisely is about, what is already possible and where the challenges are. Contrarily to what was proposed in the 60's by many researchers, that building an intelligent machine, will only be a question of a few years, there is no such thing yet. At least not in the sense proposed back then. What AI research in the beginning targeted at was to build a machine that were capable of general, human intelligent behavior. For example, it was supposed to stand the famous Turing test (Turing, 1950). He suggested to call a machine intelligent if a human that is communicating with the machine and another human through tele–typing (i.e. "chatting") can not decide which is which. Even today, over 50 years later this is far from being realistic. On the other hand, there are techniques which were invented before the 60's that are still up to date. Rosenblatt's perceptron (Rosenblatt, 1956) for example is almost equivalent to one of the most advanced learning techniques available today, the support vector machine (Boser et al., 1992). Another example is of course Fisher's discriminant (Fisher, 1936), laying the ground for this work. However, that old ideas are still in a way up to date does not imply that AI research has not made any progress over the past decades.

What has changed largely over the years is the way researchers approach the problem of learning. There are only very few people left which target at building a system that has a human like intelligence. One reason that most of these projects failed was that the problem is too complex to be modeled explicitly (and it still is). The big advances in AI research came up by resorting to simpler problems which can be considered in a mathematical framework. This mathematical framework does not try to simulate the world anymore but converts learning problems to problems of inferring a complex statistical relationship between several quantities.

However, it seems that AI research has come to a point where it seems to become realistic to ask the old question again: How can we build a machine that is truly intelligent? However, we conjecture that answering this question will require a joint effort between many different research areas, including but not limited to biology, artificial intelligence, mathematics and computer science. Whilst this thesis certainly does not answer this question we hope that it provides at least a small part to solve the big puzzle.

1.2 Notation

Symbol	Explanation
N, M	Dimensionality of data and number of samples.
$\mathbb{N}$, $\mathbb{N}^*$	The set of natural and strictly positive natural numbers.
$\mathbb{R}$, $\mathbb{R}^+$, $\mathbb{R}^*$	The set of real numbers, positive and strictly positive real numbers.
$\mathbb{C}$	Complex numbers.
$\mathbf{1}$	A vector of ones.
$\mathbf{a}, \ldots, \mathbf{z}$ or $\mathbf{ff}, \ldots$	Bold face letters denote vectors.
$\mathbf{x}$, y	Usually a training example and its label.
$\mathcal{X}$, $\mathcal{Y}$	Generic sample (input) and label (output) spaces, often $\mathbb{R}^N$ and $\{1, -1\}$ or $\mathbb{R}$, respectively.
$\mathcal{X}^R$	Shorthand for $\mathcal{X} \times \cdots \times \mathcal{X}$, R times.
$\mathcal{Z}$	A training sample $\mathcal{Z} \subseteq (\mathcal{X} \times \mathcal{Y})^M$.
$\mathcal{E}$	A feature space, usually associated with a kernel $\mathrm{k}(\mathbf{x}, \mathbf{x}\prime)$.
$\mathcal{F}, \mathcal{G}$	Function spaces.
$\mathcal{H}$	A Hilbert space.
$\mathrm{k}(\mathbf{x}, \mathbf{z})$	A kernel function, usually fulfilling Mercer's condition.
ℓ_p^N	N-dimensional sequence space with p-norm
L_p	Function space with p-norm
$(x_i)_i$	A sequence $(x_1, x_2, \ldots)$
$\mathcal{N}(\mathbf{m}, \Sigma)$	A Gaussian distribution with mean $\mathbf{m}$ and covariance Σ.
$\mathcal{I}, \mathcal{J}$	Index sets e.g. $\mathcal{I} \subseteq \{1, \ldots, M\}$.
$\mathcal{R}_M, \widehat{\mathcal{R}}_M, \mathcal{N}(\varepsilon, \mathcal{F}, \rho)$	Rademacher complexity, its empirical counterpart and covering numbers.
$\mathrm{diag}(\mathbf{v})$	A diagonal matrix with the elements of the vector $\mathbf{v}$ on the diagonal.
$\mathrm{tr}(A)$	The trace, i.e. $\sum A_{ii}$ of a matrix.
$\mathrm{P}(\mathbf{w})$	Some regularization operator.
$\mathbb{P}(\cdot)$	Probability of an event.
$\mathbb{E}_{\mathcal{Z}}[\cdot]$	Expectation over iid draw of the M–sample $\mathcal{Z}$.
μ	Some probability measure.
$\lambda_i(A)$	The i-th eigenvalue of A.
$\mathbf{A}_{i\bullet}$, $A_{\bullet i}$	the i-th row or column of a matrix A.
$\mathrm{dist}(A, B)$	The distance of two equal dimensional subspaces.
$\mathrm{sep}(A, B)$	The separation between the column span of A and B.
$\mathrm{span}(A)$	The span of the columns of A.
$\ell(y, y')$	A loss function.
R, $\mathrm{R}_{\mathrm{emp}}$	The generalization error and the empirical error.
$\mathrm{R}_{\mathrm{reg}}$, $\mathrm{R}_{\mathrm{loo}}$	A regularized error functional and the leave-one-out error.

Chapter 2

Statistical Learning Theory

> Our life is frittered away by detail. Simplify! Simplify!
>
> Henry D. Thoreau

In this chapter we introduce our basic notation. Furthermore we review basic concepts and ideas from (statistical) learning theory, starting with formulating the problem of learning from data as a statistical estimation problem. This chapter is a brief review of the existing literature in this field and focused around the ideas necessary for the developments proposed in the later chapters.

THE general problem considered here is that of inferring statistical relationships from a given, usually finite, set of examples. After the more informal motivation in the previous chapter we will now formulate these problems in a mathematical, statistical framework.

2.1 Learning from Examples

To start, we fix some notation. Denote the training data (the known observations from the world) we are given by the set $\mathcal{Z} = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} | i = 1, \ldots, M\}$. Depending on the structure of especially $\mathcal{Y}$ we want to solve a variety of problems. For example, if $\mathcal{Y} = \mathbb{R}$ we have to solve a regression problem, for $\mathcal{Y} = \{1, 2, 3\}$ a classification problem with three classes. The difficulty is, that whatever relationship between $\mathcal{X}$ and $\mathcal{Y}$ we want to model, the only knowledge we have about this relationship is given by the M pairs of examples. Since the world is not unambiguous many learning problems are not simple, deterministic one-to-one relationships. Hence, one considers the problem in a probabilistic framework. More specifically, in statistically learning theory one makes the assumption that there exists some unknown but fixed probability distribution $P(X, Y)$ over the space $\mathcal{X} \times \mathcal{Y}$ that fully describes the process generating our data. Furthermore, we assume that our training sample $\mathcal{Z}$ is drawn identically and independently from this distribution,

i.e. $\mathcal{Z}$ is i.i.d. sampled from P. It is worthwhile to note, that these assumptions of fixed, stationary distributions from which we can sample independently, albeit seeming quite general are already rather strong. In many practical applications these assumptions are violated or only true when viewing the problem in an extensively large framework (e.g. Garczarek, 2002). However, we are interested in modeling the probability of observing a specific $y \in \mathcal{Y}$ given an observation $\mathbf{x} \in \mathcal{X}$, i.e. we want to estimate $P(Y|X)$. By Bayes law this can be achieved through

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} = \frac{P(X,Y)}{P(X)}. \tag{2.1}$$

Assuming for a moment we could compute the quantity $P(Y|X)$ it is rather natural to decide each time we have to assign a $y \in \mathcal{Y}$ to a new $\mathbf{x} \in \mathcal{X}$ for

$$f^*(\mathbf{x}) = \text{argmax}_{y \in \mathcal{Y}} P(Y = y|X = \mathbf{x}), \tag{2.2}$$

i.e. we choose the y with the maximum a posteriori probability.[1] This is known as Bayes rule or the Bayes optimal decision. To see why or in which respect this strategy is optimal, we introduce the concept of a loss function:

$$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}. \tag{2.3}$$

Such a loss function should be non-negative and measure by the quantity $\ell(f(\mathbf{x}), y)$ how much we pay or lose if we predict $f(\mathbf{x}) \in \mathcal{Y}$ and the true result were $y \in \mathcal{Y}$. For a classification problem a natural choice would be $\ell(f(\mathbf{x}), y) = 0$ if the prediction is correct and $\ell(f(\mathbf{x}), y) = 1$ if it is wrong (see also Section 2.1.4). We define the expected error or risk, i.e. the overall loss we incur using a specific prediction rule f as

$$\mathrm{R}(f) = \mathbb{E}_P[\ell(f(\mathbf{x}), y)], \tag{2.4}$$

where $\mathbb{E}_P$ denotes the expectation with respect to the joint distribution $P(X, Y)$ of observations and outcomes. Then for the choice of $f = f^*$ made above R would be minimal among all possible choices for f (for a reasonably defined loss function ℓ). That is, if we knew (and could compute) the joint probability distribution $P(X, Y)$ we could solve any statistical dependence problem in an optimal way.

However, we do not know $P(X, Y)$. Now, one possible solution would be to estimate $P(X, Y)$ from the sample $\mathcal{Z}$ and many theoretical and practical approaches try exactly this in one or the other way (cf. Parzen, 1962; Bishop, 1995). But it is also well known that estimating a density without any assumptions is a hard problem. The number of examples one needs to get a reliable estimate of a density in N dimensions grows exponentially[2] with N. But what we are interested in, is learning when the amount of training data is small. That is, we need to find a way to choose a specific f from a class of candidate functions $\mathcal{F}$, such that the expected loss $\mathrm{R}(f)$ for our pick is as close as possible to the best we could do,

[1] This notation is slightly sloppy since it suggests that f^* defined this way were a function - but there could be more than just one optimal y. For convenience we assume that ties are broken by uniformly at random picking one of the possible outcomes.

[2] Consider for example binning an N dimensional space with D hypercubes in each dimension. One would need at least one observation for each of the D^N cubes to get a sensible estimate of the posterior probability.

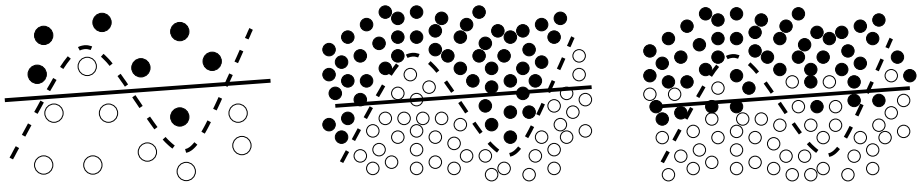

Figure 2.1: Illustration of the over–fitting dilemma: Given only a small sample (left) either, the solid or the dashed hypothesis might be true, the dashed one being more complex, but also having a smaller training error. Only with a large sample we are able to see which decision reflects the true distribution more closely. If the dashed hypothesis is correct the solid would under-fit (middle); if the solid were correct the dashed hypothesis would over-fit (right).

i.e. $\mathrm{R}(f^*)$[3]. Especially we would like to have that if the number of training examples increases (i.e. $M \to \infty$) our principle will select a $f \in \mathcal{F}$ with an expected error that converges to the minimal possible expected error over all $f \in \mathcal{F}$. Such a principle is called an induction principle and the requirements we just described make it *consistent* (cf. Section 2.1.1).

The most commonly used induction principle is the one of minimizing the empirical counter part of the expected error (2.4), i.e. the empirical error or empirical risk

$$\mathrm{R}_{\mathrm{emp}}(f) = \frac{1}{M} \sum_{i=1}^{M} \ell(f(\mathbf{x}_i), y_i). \tag{2.5}$$

Then learning consists in finding an algorithm that, given a training sample $\mathcal{Z}$, finds a function $f \in \mathcal{F}$ that minimizes (2.5). But is this principle consistent? Clearly not, if the class of functions $\mathcal{F}$ we can choose from is too large. For example, if $\mathcal{F}$ contains all possible functions from $\mathcal{X}$ to $\mathcal{Y}$, then there is an infinite number of functions for which the empirical error is zero (assuming a loss function with $\ell(y, y) = 0$), i.e. $f(\mathbf{x}_i) = y_i$ on the M training examples. But these functions f can take arbitrary values at all other points of $\mathcal{X}$ and hence make arbitrary predictions on unseen examples.

2.1.1 Over–Fitting and Consistency

This phenomenon is called over–fitting. An overly complex function f might describe the training data well but does not generalize to unseen examples. The converse could also happen. Assume the function class $\mathcal{F}$ we can choose from is very small, e.g. it contains only a single, fixed function. Then our learning machine would trivially be consistent, since $\mathrm{R}(f) = const$ for all $f \in \mathcal{F}$. But if this single $f \in \mathcal{F}$ is not by accident the rule that generates our data the decisions we make will have nothing to do with the concept generating our data. This phenomenon is called under–fitting (cf. Figure 2.1). Apparently we need some way of controlling how large the class of functions $\mathcal{F}$ is, such that we can avoid over– fitting and under–fitting. The questions of consistency, over– and under–fitting are closely related and will lead us to a concept known as regularization (e.g.

[3]We are slightly imprecise here. The truly optimal f^* given by (2.2) might not be an element of our function class. Then we replace f^* by a $f^\dagger \in \mathcal{F}$ which has the smallest possible expected error, i.e. $\mathrm{R}(f^\dagger) = \inf_{f \in \mathcal{F}} \mathrm{R}(f) \geq R(f^*)$.

Tikhonov and Arsenin, 1977; Morozov, 1984) and the structural risk minimization principle (Vapnik and Chervonenkis, 1974).

Consistency Let us define more closely what consistency means and how it can be characterized. By the law of large numbers we have that for any fixed f the empirical risk (2.5) will converge to the expected risk (2.4). Now, let us denote an empirical risk minimizer (i.e. a function $f \in \mathcal{F}$ that minimizes (2.5) for a given training sample $\mathcal{Z}$) by f^M, i.e. f^M is now dependent on the training sample. The question of consistency is, whether $\mathrm{R}_{\mathrm{emp}}(f^M)$ also converges to $\mathrm{R}(f^M)$, i.e.

$$|\,\mathrm{R}(f^M) - \mathrm{R}_{\mathrm{emp}}(f^M)| \to 0 \text{ as } M \to \infty,$$

in probability. We already gave an example above showing that this is not true in general, the reason being that f^M now depends on the sample $\mathcal{Z}$. We will not go into detail here but one can show that a sufficient and necessary condition for consistency is, that *uniformly* over all functions in $\mathcal{F}$ the difference between the expected and the empirical error converges to zero if the number of samples goes to infinity. This insight can be summarized in the following theorem:

Theorem 2.1 (Vapnik and Chervonenkis (1991)). *One-sided uniform convergence in probability, i.e.*

$$\lim_{M\to\infty} \mathbb{P}\left[\sup_{f\in\mathcal{F}} (\mathrm{R}(f) - \mathrm{R}_{\mathrm{emp}}(f)) > \epsilon\right] = 0, \tag{2.6}$$

for all $\epsilon > 0$, is a necessary and sufficient condition for (nontrivial) consistency of empirical risk minimization.

Since the condition in the theorem is not only sufficient but also necessary it seems reasonable that any "good" learning machine implementing a specific function class should fulfill (2.6).

2.1.2 Structural Risk Minimization

Having once accepted that consistency is desirable when doing empirical risk minimization the question arises if there are any principles that allow us to choose only such function classes for learning that fulfill Theorem 2.1? It will turn out that this is possible and crucially depends on the question how complex the functions in the class $\mathcal{F}$ are, a question we have already seen to be equally important when talking about over- and under–fitting. But what does complexity mean and how can one *control* the size of a function class? There are a number of different complexity measures for functions classes. Intuitively speaking, the complexity of a function class is determined by the number of different possible outcomes when choosing functions from this class. Popular measures are covering numbers, annealed entropy, VC entropy[4] and the VC dimension, or the Rademacher and Gaussian complexity. We will not go into much detail about these quantities here. The Rademacher complexity will be considered in more detail in Chapter 4.

A specific way of controlling the complexity of a function class is given by the VC theory and the structural risk minimization (SRM) principle (Vapnik and

[4] VC = Vapnik Chervonenkis.

Chervonenkis, 1974; Vapnik, 1995, 1998). Here the concept of complexity is captured by the VC dimension h of the function class $\mathcal{F}$ the estimate f is chosen from. Roughly speaking, the VC dimension measures how many (training) points can be shattered (i.e. separated) for all possible labellings using functions of the class. This quantity can be used to bound the probability that the expected error will deviate much from the empirical error for any function from the class, i.e. VC-style bounds usually take the form

$$\mathbb{P}\left[\sup_{f\in\mathcal{F}}\left(\mathrm{R}(f)-\mathrm{R}_{\mathrm{emp}}(f,\mathcal{Z})\right)>\epsilon\right]\leq H(\mathcal{F},M,\epsilon), \tag{2.7}$$

where H is some function that depends on properties of the function class $\mathcal{F}$, e.g. the VC-dimension, the size of the training set (usually in such bounds H decreases exponentially when M increases) and the desired closeness ϵ. By equating the right-hand side of (2.7) to $\delta > 0$ and solving $H = \delta$ for ϵ one can turn these bounds into expressions of the following form: With probability at least $1-\delta$ over the random draw of the training sample $\mathcal{Z}$[5]

$$\mathrm{R}(f)\leq \mathrm{R}_{\mathrm{emp}}(f,\mathcal{Z})+\widetilde{H}(\mathcal{F},M,\delta). \tag{2.8}$$

Now $\widetilde{H}$ is a penalty term that measures our degree of uncertainty. If the function class is simple/small (e.g. it contains only a single element) $\widetilde{H}$ is small. These penalty terms usually increase if we require a higher precision (e.g. with $\log(\frac{1}{\delta})$) and decrease if we observe more examples (e.g. with $\frac{1}{M}$ or $\frac{1}{\sqrt{M}}$). The practical implication of bounds like (2.8) is that our learning machine should be constructed such that

1. it finds a function with a small empirical error, and
2. at the same time keeps the penalty term $\widetilde{H}$ small.

Only if we achieve both we have a guarantee that the expected error of our estimate will be small (cf. Figure 2.2).[6]

One of the most famous learning bounds is due to Vapnik and Chervonenkis and lays the ground for the support vector algorithm that we will consider in more detail in Section 2.2:

Theorem 2.2 (Vapnik and Chervonenkis (1974)). *Let h denote the VC dimension of the function class $\mathcal{F}$ and let $\mathrm{R}_{\mathrm{emp}}$ be defined by* (2.5) *using the* 0/1*-loss (cf. Section 2.1.4). For all $\delta > 0$ and $f \in \mathcal{F}$ the inequality bounding the risk*

$$\mathrm{R}(f)\leq \mathrm{R}_{\mathrm{emp}}(f,\mathcal{Z})+\sqrt{\frac{h\left(\ln\frac{2M}{h}+1\right)-\ln(\delta/4)}{M}} \tag{2.9}$$

holds with probability of at least $1-\delta$ for $M > h$ over the random draw of the training sample $\mathcal{Z}$.

[5] Which is implicitly used to measure $\mathrm{R}_{\mathrm{emp}}$.

[6] Note that there are learning machines for which $\widetilde{H}$ is infinity which still work well in practice (e.g. k-nearest neighbors (Devroye et al., 1996)), i.e. these bounds are sufficient but not necessary conditions for successful learning.

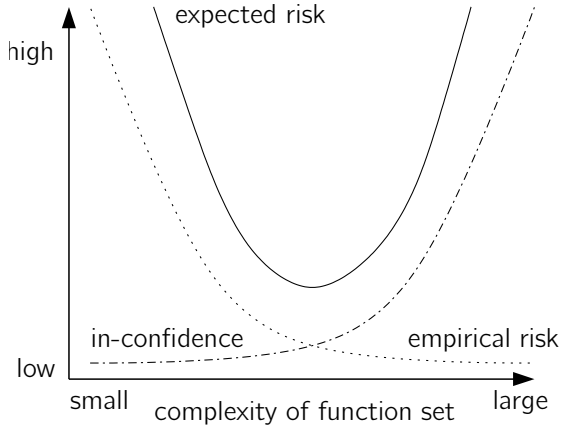

Figure 2.2: Schematic illustration of (2.8). The dotted line represents the training error (empirical risk), the dash-dotted line the upper bound on the complexity term (confidence). With higher complexity the empirical error decreases but the upper bound on the risk confidence becomes worse. For a certain complexity of the function class the best expected risk (solid line) is obtained. Thus, in practice the goal is to find the best trade-off between empirical error and complexity.

Vapnik and others (i.e. Cortes and Vapnik, 1995; Shawe-Taylor et al., 1996) also suggested the use of the so called structural risk minimization principle which now becomes a straight forward consequence of what we said before. Constructing a nested family of function classes $\mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_k$ with non-decreasing VC dimension $h_1 \leq \cdots \leq h_k$, the SRM principle proceeds as follows: Let $f_1, \ldots, f_k$ be the solutions of the empirical risk minimization (2.5) in the function classes $\mathcal{F}_i$. SRM chooses the function class $\mathcal{F}_i$ (and the function f_i) such that an upper bound on the generalization error like (2.9) is minimized. This bound is only an example and similar formulations are available for other loss functions (Vapnik, 1998) and other complexity measures (e.g. the fat-shattering dimension (Alon et al., 1997)).

2.1.3 Regularization

We have seen in the last section that for successful learning it is reasonable to find a minimum to a functional of the form (2.8)., i.e. we want to minimize the empirical error plus some penalty term. From a different point of view we want to minimize what is called a regularized risk, i.e. we define in analogy to (2.4) and (2.5)

$$R_{\text{reg}}(f) = R_{\text{emp}}(f, \mathcal{Z}) + \lambda\,\Omega(f). \tag{2.10}$$

Here $\Omega : \mathcal{F} \to \mathbb{R}^+$ is a regularization operator that measures in one or the other way properties of the function f. For example we will could try to design an operator such that $\Omega(f)$ is large when f belongs to a function class with large VC-dimension.[7] With the regularization constant $\lambda \geq 0$ we can trade-off what is more important: minimizing the empirical error or minimizing the regularizer. The hope

[7] When speaking about the VC-dimension of a function we actually mean the VC-dimension of the smallest class a learning machine can implement that contains this function. The VC-dimension of a single function is one.

is that for a suitably chosen λ and Ω the minimizer of R_{reg} will also be a minimizer of some bound like (2.8). Provided that the size of the regularization operator reflects the size of e.g. the VC-dimension of f (or some other complexity measure in another bound) we can implement the structural risk minimization principle by minimizing R_{reg} over a range of values for λ thereby getting the empirical risk minimizers from different, nested subsets. We then choose the function f (and the λ) which either minimizes some theoretical bound or e.g. the validation error. Either way, this poses the problem of model selection, a topic we will not cover here. In Section 2.2 we will see how support vector machines implement the minimization of a regularized risk functional to approximate the minimization of the bound in (2.9).

2.1.4 Loss Functions

While our discussion of the theoretical foundations of learning we have not yet talked about loss functions except for saying that they should be non-negative functions of the form (2.3). In the following we will discuss some of the most common choices of loss functions for classification and regression.

Classification When our goal is classification we are looking for functions $f : \mathcal{X} \to \{1, \ldots, D\}$, D being the number of classes. The natural choice for a loss function in this case would be the so called zero/one loss, i.e.

$$\ell(f(\mathbf{x}), y) = \begin{cases} 0 & \text{if } f(\mathbf{x}) = y, \\ 1 & \text{otherwise.} \end{cases} . \tag{2.11}$$

Then the expected loss (2.4) would exactly be the average number of errors we commit. However, there is one problem to a loss function like (2.11): minimizing the empirical risk becomes an NP-hard problem for most function classes $\mathcal{F}$. Instead, one often uses real valued (convex) loss functions and real valued functions f, the latter having the additional advantage that the absolute value of $f(\mathbf{x})$ can often be interpreted as a confidence measure for the decision we make. For two–class problems, i.e. the set of possible outcomes is $\mathcal{Y} = \{-1, +1\}$, classification is done by taking the sign, i.e. $\text{sgn}(f(\mathbf{x}))$ as a final decision. In this case so called soft margin (Bennett and Mangasarian, 1992) and logistic loss functions are common. The soft margin loss is defined as

$$\ell(f(\mathbf{x}), y) = \begin{cases} 0 & \text{if } yf(\mathbf{x}) \geq 1, \\ 1 - yf(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{2.12}$$

The product $yf(\mathbf{x})$ will be positive if the sign of $f(\mathbf{x})$ and y agree. The logistic loss is defined by

$$\ell(f(\mathbf{x}), y) = \log(1 + \exp(-yf(\mathbf{x}))), \tag{2.13}$$

and bears strong connections to the interpretability as probabilities of the outputs (cf. Section 3.2.2).

Regression If we do regression, we look for functions of the form $f : \mathcal{X} \to \mathbb{R}$ (or more generally $\mathbb{R}^D$). Here there are two loss functions commonly used, the simple squared loss

$$\ell(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2, \tag{2.14}$$

and the ϵ-insensitive loss

$$\ell(f(\mathbf{x}), y) = \begin{cases} |f(\mathbf{x}) - y| - \epsilon & \text{if } |f(\mathbf{x}) - y| > \epsilon \\ 0 & \text{otherwise.} \end{cases} \tag{2.15}$$

For $\epsilon = 0$ the ϵ-insensitive loss equals the ℓ_1-norm, otherwise it linearly penalizes deviations from the correct predictions by more than ϵ. Note, that the squared loss (2.14) is sometimes also used in classification.

Noise Model and Loss Functions There is a close connection between the loss function we choose and the noise model we assume over our predictions (e.g. Smola, 1998). In a probabilistic framework one would explicitly model this uncertainty assuming e.g. a Gaussian distribution over the (continuous) outputs. This would in our framework correspond to the squared loss function (2.14). An overview of different loss functions and their associated density models can be found in Table 2.1 along with an illustration in Figure 2.3. For more details see e.g. Schölkopf and Smola (2002).

Table 2.1: Loss functions for the slack variables $\xi = f(\mathrm{x}) - y$ and their corresponding density/noise models in a probabilistic framework (taken from Smola, 1998).

	loss function	density model
ε-insensitive	$\lvert\xi\rvert_\varepsilon$	$\frac{1}{2(1+\varepsilon)} \exp(-\lvert\xi\rvert_\varepsilon)$
Laplacian	$\lvert\xi\rvert$	$\frac{1}{2} \exp(-\lvert\xi\rvert)$
Gaussian	$\frac{1}{2}\xi^2$	$\frac{1}{\sqrt{2\pi}} \exp(-\frac{\xi^2}{2})$
Huber's robust loss	$\begin{cases} \frac{1}{2\sigma}\xi^2 \\ \lvert\xi\rvert - \frac{\sigma}{2} \end{cases}$	$\begin{cases} \exp(-\frac{\xi^2}{2\sigma}) & \text{if } \lvert\xi\rvert \le \sigma \\ \exp(\frac{\sigma}{2} - \lvert\xi\rvert) & \text{otherwise} \end{cases}$
Polynomial	$\frac{1}{p}\lvert\xi\rvert^p$	$\frac{p}{2\Gamma(1/p)} \exp(-\lvert\xi\rvert^p)$
Piecewise poly.	$\begin{cases} \frac{1}{p\sigma^{p-1}}\xi^p \\ \lvert\xi\rvert - \sigma\frac{p-1}{p} \end{cases}$	$\begin{cases} \exp(-\frac{\xi^p}{p\sigma^{p-1}}) & \text{if } \lvert\xi\rvert \le \sigma \\ \exp(\sigma\frac{p-1}{p} - \lvert\xi\rvert) & \text{otherwise} \end{cases}$

2.2 Learning Theory in Practice: SVM

Having collected theses prerequisites from statistical learning theory, we now give an example of a special learning machine that exemplary builds upon these insights. The Support Vector Machine algorithm (SVM) developed by Vapnik and others is one of the most successful techniques over the last decades, especially after being combined with the kernel trick which we shall discuss in Section 2.3 (e.g. Vapnik and Chervonenkis, 1974; Boser et al., 1992; Cortes and Vapnik, 1995; Shawe-Taylor et al., 1996; Cristianini and Shawe-Taylor, 2000; Müller et al., 2001; Schölkopf and Smola, 2002, and numerous others).

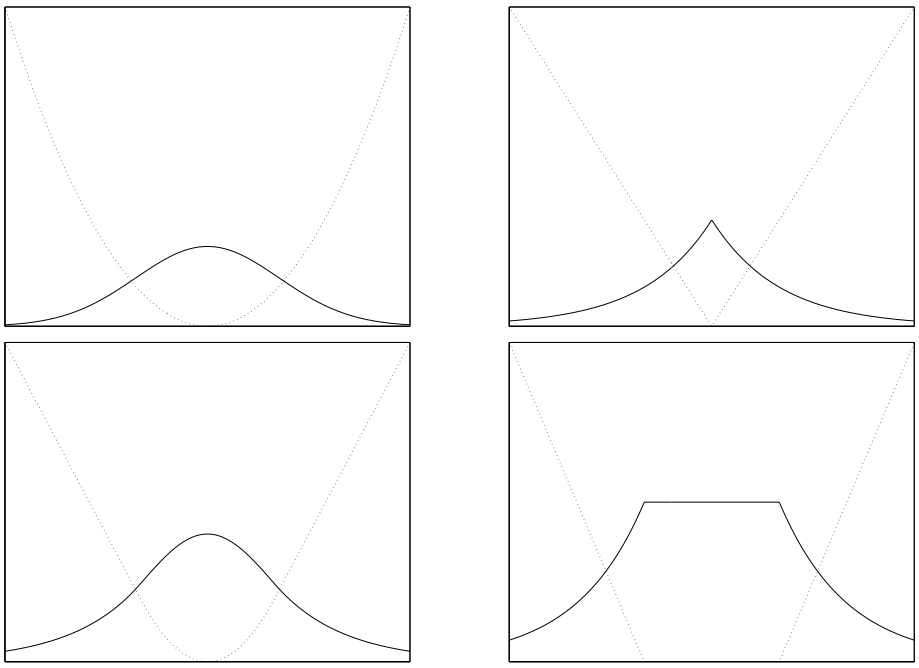

Figure 2.3: From left to right and top to bottom, an illustration of Gaussian, Laplacian, Huber's robust and ε–insensitive loss functions (dotted) and their corresponding densities (solid).

2.2.1 Margins and VC-dimension

Even if the idea of structural risk minimization seems very reasonable we have not yet discussed how one could possibly control the size of a function class nor have we discussed how we could select the empirical risk minimizer in this class.

In the following, let us assume that we are dealing with a two class classification problem (i.e. $\mathcal{Y} = \{-1, +1\}$) in a real valued vector space, e.g. $\mathcal{X} = \mathbb{R}^N$. Further, we assume that the distribution of this two classes is such, that they are linearly separable, i.e. one can find a linear function of the inputs $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) < 0$ whenever the label $y = -1$ and $f(\mathbf{x}) \geq 0$ otherwise. This can be conveniently expressed by a hyperplane in the space $\mathcal{X}$, i.e. we are looking for a function f of the form

$$f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b. \tag{2.16}$$

Now assume that the function class $\mathcal{F}$ we choose our solution from is the one containing all possible hyperplanes, i.e. $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R} | f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b\}$. To describe the complexity of this class we will use the (coarse) measure of its VC-dimension h. Recall, that the VC-dimension of a function class was defined as the maximal number M of examples $\mathbf{x}_i$ that can be separated (shattered) for an arbitrary labeling of the example using functions from $\mathcal{F}$, i.e. the maximal M such that there exists an $f \in \mathcal{F}$ with $f(\mathbf{x}_i) = y_i$ for arbitrary $y_i \in \{-1, +1\}$. For $\mathcal{X} = \mathbb{R}^N$ it is rather straight forward to show that the VC-dimension of this class of functions will be $h = N + 1$, i.e. in an N dimensional space the maximal number of points that can be separated for an arbitrary labeling using a hyperplane is $N+1$.

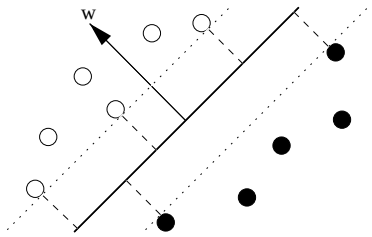

Figure 2.4: Linear classifier and margins: A linear classifier is defined by a hyperplane's normal vector w and an offset b, i.e. the decision boundary is $\{x|(w\cdot x)+b=0\}$ (solid line). Each of the two half spaces defined by this hyperplane corresponds to one class, i.e. $f(x)=\mathrm{sgn}((w\cdot x)+b)$. The margin of a linear classifier is the minimal distance of any training point to the hyperplane. In this case it is the distance between the dotted lines and the solid line.

Whilst it is satisfactory to see that the VC-dimension of the class of hyperplanes is finite and hence an application of (2.9) would make sense, we can not apply the structural risk minimization principle yet: there is no nested structure of function classes. But we can introduce this structure by limiting/regularizing the functions in $\mathcal{F}$. To this end define the function classes

$$\mathcal{F}_\Lambda = \{f : \mathbb{R}^N \to \mathbb{R} | f(x) = (\mathbf{w}\cdot\mathbf{x}) + b, \|\mathbf{w}\| \leq \Lambda\}. \tag{2.17}$$

Clearly $\mathcal{F}_{\Lambda_1} \subseteq \mathcal{F}_{\Lambda_2}$ whenever $\Lambda_1 \leq \Lambda_2$. But what effect does constraining the norm of the weight vector have on the corresponding VC-dimensions of $\mathcal{F}_\Lambda$? It turns out, that under certain assumptions outlined below we also get $h(\mathcal{F}_{\Lambda_1}) \leq h(\mathcal{F}_{\Lambda_2})$ for $\Lambda_1 \leq \Lambda_2$, i.e. we get the desired nested family of function classes with non-decreasing VC-dimension necessary for the application of the structural risk minimization principle.

The crucial ingredient in making the function classes $\mathcal{F}_\Lambda$ nested is to define a unique representation for each hyperplane. To this end, we introduce the concept of canonical hyperplanes and the notion of margins. Otherwise, if the data are separable by $(\mathbf{w}, b)$ then they are also separable by any (positive) multiple of $(\mathbf{w}, b)$ and hence there is an infinity number of representations for the same separating hyperplane. In particular, all function classes $\mathcal{F}_\Lambda$ would have the same VC-dimension as they would contain the same functions, just in different representations. A canonical hyperplane with respect to an M-sample $\mathcal{Z}$ is defined as a function $f(\mathbf{x}) = (\mathbf{w}\cdot\mathbf{x})+b$, where $\mathbf{w}$ is normalized, such that $\min_{i=1,\ldots,M} |f(\mathbf{x}_i)| = 1$, Then none of the training examples produces an absolute output that is smaller than one and the examples closest the hyperplane have exactly an output of one, i.e. $(\mathbf{w}\cdot\mathbf{x})+b = \pm 1$. Since we assumed the sample $\mathcal{Z}$ to be linearly separable, we can turn any f that separates the data into a canonical hyperplane by suitably normalizing the weight vector $\mathbf{w}$ and adjusting the threshold b correspondingly. The concept of a margin is closely related. We define the margin to be the minimal euclidean distance between any training example $\mathbf{x}_i$ and the separating hyperplane. Intuitively, the margin measures how good the separation between the two classes by a hyperplane is. Then, if this hyperplane is in canonical form, the margin can be measured by the length of the weight vector $\mathbf{w}$. Consider two examples $\mathbf{x}_1$ and $\mathbf{x}_2$ from different classes with $(\mathbf{w}\cdot\mathbf{x}_1)+b = 1$ and $(\mathbf{w}\cdot\mathbf{x}_2)+b = -1$, respectively. The margin is given by the distance of these two points, measured perpendicular

to the hyperplane, i.e. $\left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}_1 - \mathbf{x}_2)\right) = \frac{2}{\|\mathbf{w}\|}$ (e.g. Vapnik, 1995). Hence, the smaller the norm of the weight vector $\mathbf{w}$ in the canonical representation, the larger the margin. More general, it was shown, that the VC-dimension of the class $\mathcal{F}_\Lambda$ (restricted to canonical hyperplanes) can be bounded as

$$h \leq \min(\Lambda^2 R^2 + 1, N + 1) \quad \text{and} \quad \|\mathbf{w}\|_2 \leq \Lambda \tag{2.18}$$

where R is the radius of the smallest sphere around the data (e.g. Vapnik, 1995). Thus, if we bound the margin of a function class from below, say by $\frac{2}{\Lambda}$, we can control its VC-dimension and hence apply the SRM principle.[8] A particularly impor-

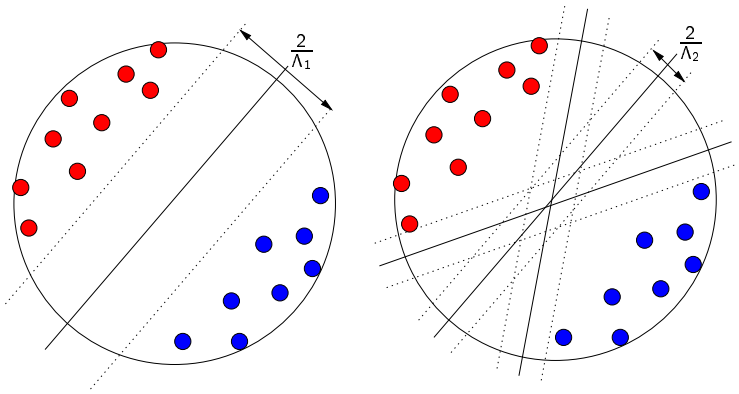

Figure 2.5: Illustration of why a large margin reduces the complexity of a linear hyperplane classifier. If we choose hyperplanes with a large margin, there is only a small number of possibilities to separate the data, i.e. the VC-dimension of $\mathcal{F}_{\Lambda_1}$ is small. (left panel). On the contrary, if we allow smaller margins there are more separating hyperplanes, i.e. the VC-dimension of $\mathcal{F}_{\Lambda_2}$ is large (right panel). Illustration inspired by Rätsch (2001).

tant insight is that the complexity only indirectly depends on the dimensionality of the data.[9] This is very much in contrast to e.g. density estimation, where the problems become more difficult as the dimensionality of the data increases (cf. Section 2.1). For SVM, if we can achieve a large margin the problem remains simple.

2.2.2 Support Vector Machines

Support Vector Machines are a practical implementation, merging the insights from VC-theory and the connection between margins and the VC-dimension of a linear function class into one algorithm. The central idea is to find a weight vector $\mathbf{w}$ such that the margin is as large as possible. Still assuming for the moment that the data are separable, we hence need to find the smallest possible

[8] There are some ramifications to this statement, that go beyond the scope of this presentation. Strictly speaking, VC theory requires the structure to be defined a priori, which has implications for the definition of the class of separating hyperplanes, cf. Shawe-Taylor et al. (1996).

[9] Indirectly means, (i) that the VC-dimension is the dimensionality if we can not achieve a large enough margin or small enough norm of w, respectively, and (ii) that the estimate for the sphere containing the data will often depend on the dimensionality as well. Nonetheless, achieving a large margin is often easier in high dimensions than in low dimensions.

w without committing any error. This can be conveniently expressed by a quadratic optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i\left((\mathbf{w}\cdot\mathbf{x})+b\right) \geq 1, \quad \forall i = 1,\ldots,M. \end{aligned} \tag{2.19}$$

The constraints in (2.19) assure that **w** and b will be chosen such that no example has a distance to the hyperplane smaller than one. For the optimal solution we will also achieve a canonical hyperplane. One can solve this problem directly using a quadratic optimizer (cf. Appendix A for more detail on linear and quadratic optimization). A particularly nice feature is that one can find a global minimum of this problem, this in contrast to many neural networks (e.g. Bishop, 1995). In fact, all minima of (2.19) are global minima, although they might not be unique as e.g. in the case when $M < N$, where N is the dimensionality of the data.

Another possibility to optimize (2.19) is to form its dual and optimize this instead. While this does not seem particularly useful at this point we shall see in the next section that this paves the way to derive powerful non-linear generalizations of SVM by using kernel functions. We will not go into much detail here about mathematical programming and how one derives dual formulations. A slightly more concise treatment can be found in Appendix A or in one of the many textbooks on this topic (e.g. Bertsekas, 1995; Luenberger, 1984; Mangasarian, 1997). The crucial point is, that for every quadratic (or linear) optimization problem (QP/LP) there exists a dual problem which is also a quadratic or linear problem, respectively. The original problem is called primal. Primal and dual are connected by the following facts:

- Either the primal is infeasible,[10] then the dual is unbounded (and vice versa),
- or both are feasible and then there exists an optimal (finite) solution.
- In this case, both, primal and dual reach the same objective function value at the optimal solution, and
- any primal and dual feasible solution for which the primal and dual objective functions coincide is already optimal.

The dual problem is usually stated as a maximization problem.

To derive the dual of (2.19), we introduce Lagrange multipliers $\alpha_i \geq 0$, $i = 1,\ldots,M$, one for each of the constraints in (2.19). We get the following Lagrangian:

$$L(\mathbf{w}, b, \mathbf{ff}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{M} \alpha_i (y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) - 1). \tag{2.20}$$

The task is to minimize (2.20) with respect to **w**, b and to maximize it with respect to α_i. At the optimal point, we have the following saddle point equations:

$$\frac{\partial L}{\partial b} = 0 \quad \text{and} \quad \frac{\partial L}{\partial \mathbf{w}} = 0,$$

[10] Infeasible means that there exist no solution to the problem which fulfills the constraints. A feasible solution is any variable assignment that fulfills the constraints.

which translate into

$$\sum_{i=1}^{M} \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^{M} \alpha_i y_i \mathbf{x}_i. \tag{2.21}$$

From the right equation of (2.21), we find that $\mathbf{w}$ is contained in the subspace spanned by the $\mathbf{x}_i$ in the training set. By substituting (2.21) into (2.20), we get the dual quadratic optimization problem:

$$\max_{\alpha} \quad \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \tag{2.22}$$

$$\text{subject to} \quad \alpha_i \geq 0, \ i = 1, \ldots, M, \tag{2.23}$$

$$\sum_{i=1}^{M} \alpha_i y_i = 0. \tag{2.24}$$

Thus, by solving the dual optimization problem, one obtains the coefficients α_i, $i = 1, \ldots, M$, which one needs to express the solution $\mathbf{w}$. This leads to the decision function

$$\begin{aligned} f(\mathbf{x}) &= \operatorname{sgn}\left((\mathbf{w} \cdot \mathbf{x}_i) + b\right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{M} y_i \alpha_i \left(\mathbf{x}_i \cdot \mathbf{x}\right) + b\right). \end{aligned} \tag{2.25}$$

Note once more, that this expression does not directly depend on the dimensionality N of the data but on the number of training examples M. As long as we are able to evaluate the scalar product $(\mathbf{x}_i \cdot \mathbf{x})$ the dimensionality could be arbitrary, even infinity. We will use this amazing fact in Section 2.3.

So far we have only considered the separable case which corresponds to an empirical error of zero (cf. Theorem 2.2). However for most practical applications, e.g. with noisy data this assumption will be violated. If the data is *not* linearly separable then problem (2.19) would not have any feasible solution. By allowing for some errors we might get better results and avoid over–fitting effects (cf. Figure 2.1).

Therefore a "good" trade-off between the empirical risk and the complexity term in (2.9) needs to be found. Using a technique which was first proposed in Bennett and Mangasarian (1992) and later used for SVMs in Cortes and Vapnik (1995), one introduces slack-variables to relax the hard-margin constraints:

$$y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, M, \tag{2.26}$$

additionally allowing for some classification errors. The SVM solution can then be found by (a) keeping the upper bound on the VC dimension small and (b) by minimizing an upper bound $\sum_{i=1}^{M} \xi_i$ on the empirical risk,[11] i.e. the number of training errors. Thus, one minimizes

$$\min_{\mathbf{w}, b, \xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{M} \xi_i.$$

[11] Other bounds on the empirical error, like $\sum_{i=1}^{M} \xi_i^2$ are also frequently used (e.g. Cortes and Vapnik, 1995; Mangasarian and Musicant, 2001).

where the regularization constant $C > 0$ determines the trade-off between the empirical error and the complexity term. This leads to the dual problem:

$$\max_{\text{ff}} \quad \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \tag{2.27}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, \; i = 1, \ldots, M, \tag{2.28}$$

$$\sum_{i=1}^{M} \alpha_i y_i = 0. \tag{2.29}$$

From introducing the slack-variables ξ_i, one gets the *box* constraints that limit the size of the Lagrange multipliers: $\alpha_i \leq C, \; i = 1, \ldots, M$.

Sparsity

Most optimization methods are based on the second order optimality conditions, so called Karush-Kuhn-Tucker conditions which state necessary and in some cases sufficient conditions for a set of variables to be optimal for an optimization problem. It comes in handy that these conditions are particularly simple for the dual SVM problem (2.27) (Vapnik, 1982):

$$\begin{array}{rcll} \alpha_i = 0 & \Rightarrow & y_i f(\mathbf{x}_i) \geq 1 \quad \text{and} & \xi_i = 0 \\ 0 < \alpha_i < C & \Rightarrow & y_i f(\mathbf{x}_i) = 1 \quad \text{and} & \xi_i = 0 \\ \alpha_i = C & \Rightarrow & y_i f(\mathbf{x}_i) \leq 1 \quad \text{and} & \xi_i \geq 0 \end{array} \tag{2.30}$$

They reveal one of the most important properties of SVMs: the solution is sparse in **ff**. For all examples that are outside the margin area the optimal α_i's are zero (cf. Figure 2.10). Specifically, the KKT conditions show that only such α_i connected to a training pattern $\mathbf{x}_i$, which is either on the edge of (i.e. $0 < \alpha_i < C$ and $y_i f(\mathbf{x}_i) = 1$) or inside the margin area (i.e. $\alpha_i = C$ and $y_i f(\mathbf{x}_i) < 1$) are non-zero. This sparsity property makes SVM learning practical for large data sets.

Computing the Threshold

The threshold b can be computed by exploiting the fact that for all support vectors $\mathbf{x}_i$ with $0 < \alpha_i < C$, the slack variable ξ_i is zero. This follows from the Karush-Kuhn-Tucker (KKT) conditions (cf. (2.30)). Thus, for any support vector $\mathbf{x}_i$ with $i \in I := \{i : 0 < \alpha_i < C\}$ holds:

$$y_i \left(b + \sum_{j=1}^{M} y_j \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)) \right) = 1.$$

Averaging over these patterns yields a numerically stable solution:

$$b = \frac{1}{|I|} \sum_{i \in I} \left(y_i - \sum_{j=1}^{M} y_j \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \right).$$

However, this way we only recover the threshold that was optimal according to the optimization problem (2.19) or its soft–margin version. It has been observed

in practice that there are cases when estimating a separate threshold does improve the generalization error (cf. Schölkopf, 1997). Furthermore, by increasing or decreasing the threshold we can change the number of false negative and false positive classified examples. This might be very important in e.g. medical applications when errors of one kind are much more severe than of the other kind (see also Lin et al. (2002)).

A Geometrical Explanation

Here, we will present an illustration of the SVM solution to enhance intuitive understandings. Let us normalize the weight vector to 1 (i.e. $\|\mathbf{w}\|_2 = 1$) and fix the threshold $b = 0$. Then, the set of all $\mathbf{w}$ which separate the training examples is completely described as

$$\mathcal{V} = \{\mathbf{w} | y_i f(\mathbf{x}_i) > 0; i = 1, \ldots, M, \|\mathbf{w}\|_2 = 1\}$$

The set $\mathcal{V}$ is called "version space" (Opper and Haussler, 1991). It can be shown that the SVM solution coincides with the Tchebycheff-center of the version space, which is the center of the largest sphere contained in $\mathcal{V}$ (cf. Shawe-Taylor and Williamson, 1997). However, the theoretical optimal point in version space yielding a Bayes-optimal decision boundary is the Bayes point, which is known to be closely approximated by the center of mass of the version space (Herbrich et al., 2001; Watkin, 1993). The version space is illustrated as a region on the sphere as shown in Figure 2.6. If the version space is shaped as in the left part of Figure 2.6, the SVM solution is near to the optimal point. However, if it has an elongated shape as in the right part of Figure 2.6, the SVM solution is far from the optimal one. To cope with this problem, several researchers (Ruján, 1996; Herbrich et al., 2001; Herbrich and Graepel, 2001) proposed a billiard sampling method for approximating the Bayes point. This method can achieve improved results, as shown on several benchmarks in comparison to SVMs.

2.3 Kernel Functions

In the last section we have seen that by restricting ourselves to linear functions we are well able to control the complexity of our learning machine. However, this may not come as a surprise: linear functions seem very simple anyway. For example, not even the easy XOR problem can be solved using hyperplanes (cf. Figure 2.7). We have avoided the problem of dealing with too complex functions at the price that the range of problems we can solve at this point is very limited. But there is a way to have both, linear models with controllable complexity *and* a very rich set of nonlinear decision functions, by using the tools that will be discussed in this section. Central to the success of support vector machines was the re-discovery of the so called Reproducing Kernel Hilbert Spaces (RKHS) and Mercer's Theorem (Boser et al., 1992). There is a large body of literature dealing with kernel functions, their theory and applicability (e.g. Kolmogorov (1941); Aronszajn (1950); Aizerman et al. (1964); Saitoh (1988); Boser et al. (1992) or Schölkopf and Smola (2002) for an overview). We only recall the basic definitions and properties necessary for turning our linear, hyperplane based learning technique into a very powerful

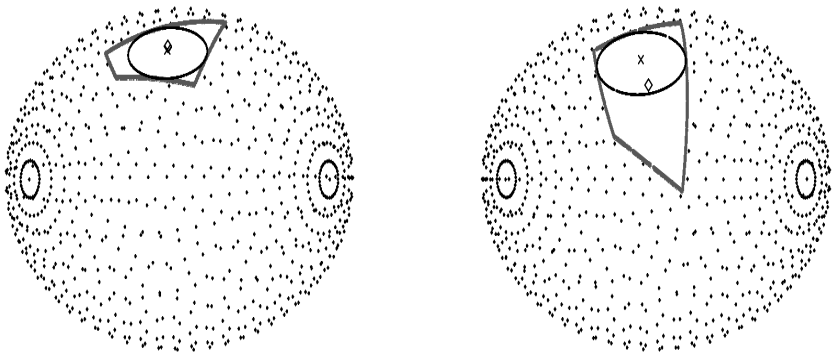

Figure 2.6: An example of the version space where the SVM works fine (left) and where the SVM works poorly (right) in approximating the center of mass. In the left example the center of mass ($\Diamond$) is close to the SVM solution ($\times$). In the right example the version space has an elongated shape and the center of mass is far from the SVM solution. Figures taken from Herbrich et al. (1999).

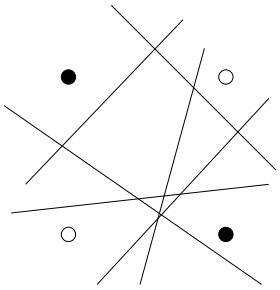

Figure 2.7: The XOR-problem: a classical example of a simple problem that can not be solved using linear functions. This generalizes to arbitrary dimensions. In N dimensions the maximal number of points that can be shattered (i.e. separated) by a linear function is $N+1$ (cf. Chapter 1), i.e. the VC-dimension of linear functions is $N+1$. Contrary, there are very simple non-linear models which can solve this type of problem.

algorithm well capable of finding non-linear decision functions with controllable complexity.

The basic idea of the so called *kernel-methods* is to first preprocess the data by some non-linear mapping Φ and then to apply the same linear algorithm as before, but in the image space of Φ. The hope is that for a sufficiently nonlinear and appropriate Φ a linear decision in the image space of Φ will be enough (cf. Figure 2.8 for an illustration). More formally we apply the mapping Φ,

$$\begin{aligned} \Phi : \mathbb{R}^N &\to \mathcal{E} \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}) \end{aligned}$$

to the data $\mathbf{x}_1, \ldots, \mathbf{x}_M \in \mathcal{X}$ and now consider our algorithm in $\mathcal{E}$ instead of $\mathcal{X}$, i.e. one works with the sample

$$\{(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_M), y_M)\} \subseteq (\mathcal{E} \times \mathcal{Y})^M.$$

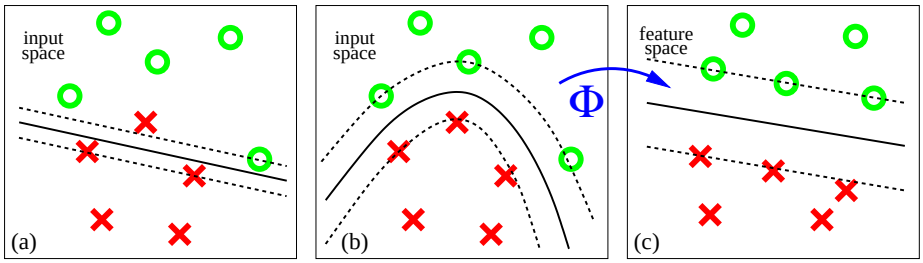

Figure 2.8: Three different views on the same dot versus cross separation problem. (a) In this example, a linear separation of the input points is not possible without errors. Even the misclassification of one data point permits only a small margin. The resulting linear classification function looks inappropriate for the data. (b) A better separation is permitted by nonlinear surfaces in input space. (c) These nonlinear surfaces correspond to linear surfaces in feature space. Data points are mapped from input space to feature space by the function Φ that is implied by the kernel function k.

In certain applications we might have sufficient knowledge about our problem such that we can design an appropriate Φ from hand (e.g. Zien et al., 2000; Blankertz et al., 2002). If this mapping is not too complex to compute and the space $\mathcal{E}$ is not too high dimensional, we might just explicitly apply this mapping to our data and are done. Something, similar is done for (one hidden layer) neural networks (Bishop, 1995), radial basis networks (e.g Moody and Darken, 1989) or Boosting algorithms (Freund and Schapire, 1997) where the input data is mapped to some representation given by the hidden layer, the RBF bumps or the hypotheses space respectively (Rätsch et al., 2002). The difference with kernel-methods is, as we shall see shortly, that for a suitably chosen Φ we get an algorithm that has powerful non-linearities but is still very intuitive and retains most of the nice properties of its linear input space version.

What can we do in cases when we do not have particular prior knowledge on how to linearize our problem or the mapping is intractable to compute, either in terms of computational complexity or in terms of storage requirements? To see that the latter might easily happen and as an example how one could solve these problems consider the following example in Figure 2.9: in two dimensions a rather complicated *nonlinear* decision surface is necessary to separate the classes, whereas in a feature space of second order monomials (e.g. Schürmann, 1996)

$$\begin{aligned} \Phi : \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2) \end{aligned} \tag{2.31}$$

all one needs for separation is a *linear* hyperplane. Clearly, in this example we could just carry out the mapping $\Phi(\mathbf{x}) = (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)^\top$ explicitly. But consider the case where our input space $\mathcal{X}$ consists of images of 16×16 pixels as patterns (i.e. 256 dimensional vectors) and we choose as nonlinearity all 5th order monomials – then one would map to a space that contains all 5^{th} order products of 256 pixels, i.e. to a

$$\binom{5 + 256 - 1}{5} \approx 10^{10}$$

dimensional space. Such a mapping would clearly be intractable to carry out explicitly.

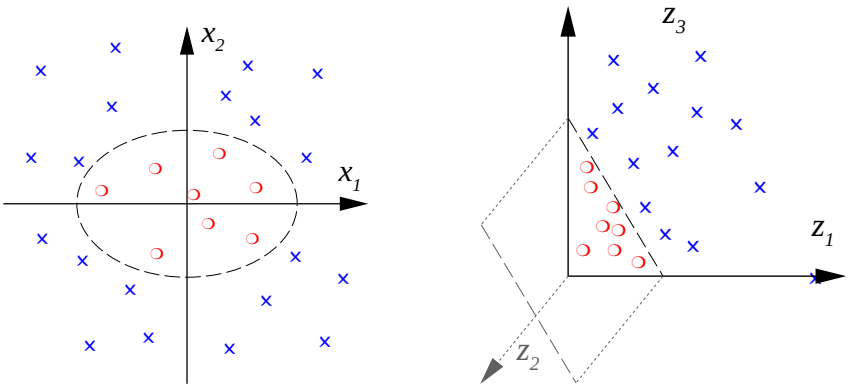

Figure 2.9: Two dimensional classification example. Using the second order monomials $x_1^2, \sqrt{2}x_1x_2$ and x_2^2 as features a separation in feature space can be found using a *linear* hyperplane (right). In input space this construction corresponds to a *non-linear* ellipsoidal decision boundary (left) (figure from Schölkopf and Smola (2002)).

However, what we can do is compute scalar products in this space. Let us come back to the example from (2.31). Here, the computation of a scalar product between two feature space vectors, can be readily reformulated in terms of a kernel function k

$$\begin{aligned}(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})) &= (x_1^2, \sqrt{2}\, x_1x_2, x_2^2)(z_1^2, \sqrt{2}\, z_1z_2, z_2^2)^\top \\ &= ((x_1, x_2)(z_1, z_2)^\top)^2 \\ &= (\mathbf{x} \cdot \mathbf{z})^2 \\ &=: \mathrm{k}(\mathbf{x}, \mathbf{z}).\end{aligned}$$

This finding generalizes:

- For $\mathbf{x}, \mathbf{z} \in \mathbb{R}^N$, and $d \in \mathbb{N}$ the kernel function

$$\mathrm{k}(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^d$$

  computes a scalar product in the space of all products of d vector entries (monomials) of $\mathbf{x}$ and $\mathbf{z}$ (Vapnik, 1995; Schölkopf et al., 1998b).

The *kernel trick* (Aizerman et al., 1964; Saitoh, 1988; Boser et al., 1992; Vapnik, 1995) is to take the original algorithm and formulate it such, that we only use $\Phi(\mathbf{x})$ in scalar products. Then, if we can efficiently evaluate these scalar products we do not need to carry out the mapping Φ explicitly and can still solve the problem in the huge feature space $\mathcal{E}$. Even better, it will turn out that we do not need to know the mapping Φ but only the kernel function. Below we will see under which conditions a mapping $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ corresponds to a scalar product.

Now it becomes clear why we formed the dual optimization problem for SVM (cf. (2.22) and (2.27)): the examples $\mathbf{x}$ only occur in scalar products and also in the resulting decision function (2.25) $\mathbf{x}$ only occurs in scalar products. Thus, we can replace any occurrence of $\mathbf{x} \cdot \mathbf{z}$ by $\mathrm{k}(\mathbf{x}, \mathbf{z})$ and solve the SVM in $\mathcal{E}$ instead of $\mathcal{X}$.

Now we can ask two questions

- For which mappings Φ is there a simple way to evaluate the scalar product and
- under which conditions does a function $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ correspond to a scalar product?

The first question is difficult to answer in general. But for the second question there exists an answer which we present in the following.

2.3.1 Feature Spaces

To see when a kernel function $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (or $\mathbb{C}$) is a dot-product let us first introduce some more notation and definitions. Given a training sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\} \subseteq \mathcal{X}$, the $M \times M$ matrix K whose elements are given by $K_{ij} = \mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)$ is called the kernel matrix or Gram matrix. An $M \times M$ matrix K (and any other symmetric matrix) is said to be positive definite if any quadratic form over K is positive, i.e. for all $r_i \in \mathbb{R}$ or $c_i \in \mathbb{C}$, $i = 1, \ldots, M$, we have

$$\sum_{i,j=1}^{M} r_i r_j K_{ij} \geq 0 \text{ or } \sum_{i,j=1}^{M} c_i \overline{c_j} K_{ij} \geq 0. \tag{2.32}$$

Positive definite kernels are exactly those giving rise to a positive definite kernel matrix K for all M and all sets $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\} \subseteq \mathcal{X}$. Note, that for a kernel (and a matrix) to be positive, it is necessary to be symmetric, i.e. $K_{ij} = \overline{K_{ji}}$ and $\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j) = \overline{\mathrm{k}(\mathbf{x}_j, \mathbf{x}_i)}$, respectively, and non-negative on the diagonal, i.e. $K_{ii} \geq 0$ and $\mathrm{k}(\mathbf{x}, \mathbf{x}) \geq 0$, respectively.

For any positive definite kernel k we can construct a mapping Φ into a feature space $\mathcal{E}$, such that k acts as a dot-product over Φ. As a matter of fact, it is possible to construct more than one of these spaces. We will omit many crucial details and only present the central results. For more details see e.g. Schölkopf and Smola (2002).

The Feature Map

Given a real-valued, positive definite kernel function k, defined over a non-empty set $\mathcal{X}$, we define the feature space $\mathcal{E}$ as the space of all functions mapping from $\mathcal{X}$ to $\mathbb{R}$, i.e. as $\mathcal{E} = \mathbb{R}^{\mathcal{X}} = \{f | f : \mathcal{X} \to \mathbb{R}\}$. We then define the mapping Φ as

$$\Phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}, \Phi(\mathbf{x}) = \mathrm{k}(\cdot, \mathbf{x}), \tag{2.33}$$

i.e. Φ maps each $\mathbf{x}$ to the function $\mathrm{k}(\cdot, \mathbf{x})$, i.e. the kernel k where the first argument is free and the second is fixed to $\mathbf{x}$. One can show that the set of all linear combinations of the form

$$f(\cdot) = \sum_{i=1}^{M} \alpha_i \, \mathrm{k}(\cdot, \mathbf{x}_i), \tag{2.34}$$

for arbitrary M, $\alpha_i \in \mathbb{R}$ and $\mathbf{x}_1, \ldots, \mathbf{x}_M$ forms a vector space. Especially, for all functions of the form (2.34) one gets

$$\langle \mathrm{k}(\cdot, \mathbf{x}), f \rangle_{\mathcal{H}} = f(\mathbf{x}),$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the scalar product in some Hilbert space that will be come clearer below. In particular we have

$$\begin{aligned} \langle \mathsf{k}(\cdot, \mathbf{x}), \mathsf{k}(\cdot, \mathbf{z}) \rangle_{\mathcal{H}} &= \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle_{\mathcal{E}} \\ &= \mathsf{k}(\mathbf{x}, \mathbf{z}). \end{aligned}$$

The last property is the reason why positive definite kernels are also called reproducing kernels: they reproduce the evaluation of f on $\mathbf{x}$. It also shows that k indeed computes, as desired, the dot–product in $\mathcal{E}$ for $\Phi(\mathbf{x})$ defined as in (2.33). Hence (2.33) is one possible realization of the mapping associated with a kernel and is called the feature map (for its empirical counterpart see e.g. Mika, 1998; Tsuda, 1998).

Before we move on to a second possible representation of the feature space $\mathcal{E}$ associated with a kernel k let us briefly state the definition of a reproducing kernel Hilbert space. The space $\mathcal{E}$ we defined before can be completed to be such a space by adding the limit points of all series that are convergent in the norm induced by the dot–product, i.e. $\|f\| = \sqrt{\langle f, f \rangle}$ (cf. Schölkopf and Smola, 2002).

Definition 2.1 (Reproducing Kernel Hilbert Space (RKHS)). Let $\mathcal{X}$ be a nonempty set (often called the index set) and $\mathcal{H}$ a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Then $\mathcal{H}$ is called a *reproducing kernel Hilbert space* endowed with the dot product $\langle \cdot, \cdot \rangle$ if there exists a function $\mathsf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the properties that

1. k has the reproducing property $\langle f, \mathsf{k}(\cdot, \mathbf{x}) \rangle = f(\mathbf{x})$ for all $f \in \mathcal{H}$, in particular $\langle \mathsf{k}(\cdot, \mathbf{x}), \mathsf{k}(\cdot, \mathbf{z}) \rangle = \mathsf{k}(\mathbf{x}, \mathbf{z})$, and
2. k spans $\mathcal{H}$, i.e. $\mathcal{H} = \overline{\text{span}\{\mathsf{k}(\cdot, \mathbf{x}) | \mathbf{x} \in \mathcal{X}\}}$, where $\overline{A}$ denotes the completion of the set A.

One can show, that the kernel k for such a RKHS is uniquely determined.

Mercer Kernels

As a second way to identify a feature space associated with a kernel k one can use a technique derived from Mercer's Theorem. Historically, when kernel functions were introduced in support vector machines, this was the reasoning used.

Mercer's Theorem, which we will reproduce in the following, states that if the function k (the kernel) gives rise to an positive integral operator, the evaluation of $\mathsf{k}(\mathbf{x}, \mathbf{z})$ can be expressed as a finite or infinite, absolute and uniformly convergent series, almost everywhere. This series then gives rise to another way to define a feature space and associated mapping connected to the kernel k. Especially, if the set $\mathcal{X}$, the kernel is defined on, is compact, a kernel will be a Mercer kernel if and only if it is a positive definite kernel (cf. Smola et al., 1998b). The following reasoning is taken once more from Schölkopf and Smola (2002).

Let $\mathcal{X}$ be a finite measure space, i.e. a space with a σ-algebra and a measure μ satisfying $\mu(\mathcal{X}) \leq \infty$.

Theorem 2.3 (Mercer's Theorem (Mercer, 1909)). *Suppose* $\mathsf{k} \in L_\infty(\mathcal{X}^2, \mu)$ *is a symmetric real-valued function such that the integral operator*

$$T_{\mathsf{k}} : L_2(\mathcal{X}, \mu) \to L_2(\mathcal{X}, \mu), \ (T_{\mathsf{k}} f)(\mathbf{x}) := \int_{\mathcal{X}} \mathsf{k}(\mathbf{x}, \mathbf{z}) f(\mathbf{z}) d\mu(\mathbf{z})$$

is positive definite, i.e. for all $f \in L_2(\mathcal{X}, \mu)$

$$\int_{\mathcal{X}^2} \mathsf{k}(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) d\mu(\mathbf{x}) d\mu(\mathbf{z}) \geq 0.$$

Let $\varphi_j \in L_2(\mathcal{X}, \mu)$ *be the normalized orthogonal eigenfunctions of* T_{k} *associated with the eigenvalues* $\lambda_j \geq 0$*, sorted in non-increasing order. Then*

1. $(\lambda_j)_j \in l_1$
2. $\mathsf{k}(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{N_\mathcal{E}} \lambda_j \varphi_j(\mathbf{x}) \varphi_j(\mathbf{z})$ *holds for almost all* $\mathbf{x}, \mathbf{z}$*. Either* $N_\mathcal{E} \in \mathbb{N}$ *or* $N_\mathcal{E} = \infty$*; in the latter case, the series converges absolutely and uniformly for almost all* $\mathbf{x}, \mathbf{z}$*.*

We call k *a Mercer kernel.*

Now, if we choose as feature space $\mathcal{E} = l_2^{N_\mathcal{E}}$ and the mapping Φ as

$$\Phi : \mathcal{X} \to l_2^{N_\mathcal{E}}, \ \Phi(\mathbf{x}) = (\sqrt{\lambda_j} \varphi_j(\mathbf{x}))_{j=1,\ldots,N_\mathcal{E}},$$

we see from the second statement in Theorem 2.3 that the Mercer kernel k corresponds to the dot product in $l_2^{N_\mathcal{E}}$, i.e. $\mathsf{k}(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle$.

2.3.2 Properties of Kernels

Besides being useful tools for the computation of dot-products in high or infinite dimensional spaces, kernels possess some additional properties that make them an interesting choice in algorithms. It was shown (Poggio and Girosi, 1990; Girosi et al., 1993) that using a particular positive definite kernel corresponds to an *implicit* choice of a regularization operator. For translation invariant kernels, the regularization properties can be expressed conveniently in Fourier space in terms of the frequencies (Smola et al., 1998b; Girosi, 1998). For example Gaussian kernels (cf. (2.35)) correspond to a general smoothness assumption in all k-th order derivatives (Smola et al., 1998b). Vice versa, using this correspondence, kernels matching a certain prior about the frequency content of the data can be constructed as to reflect our prior problem knowledge.

Another particularly nice feature of using kernel functions is that we are not restricted to kernels that operate on vectorial data, e.g. $\mathcal{X} = \mathbb{R}^N$. In principle it is possible to also define positive kernels for e.g. strings or graphs, i.e. making it possible to embed discrete objects into a metric space and apply metric-based algorithms (e.g. Haussler, 1999; Watkins, 2000; Zien et al., 2000).

Furthermore, many algorithms can be formulated using so called *conditionally positive definite* kernels (cf. Smola et al., 1998b; Schölkopf, 2001) which are a super class of the positive definite kernels considered so far. They can be interpreted as generalized nonlinear dissimilarity measures (opposed to just the scalar product) and are applicable e.g. in SVM and kernel PCA.

2.3.3 Examples of Kernels and Illustration

Table 2.2 lists some of the most widely used kernel functions. More sophisticated kernels (e.g. kernels generating splines or Fourier expansions) and kernels designed

Table 2.2: Common kernel functions: Gaussian RBF ($c \in \mathbb{R}$), polynomial ($d \in \mathbb{N}, \theta \in \mathbb{R}$), sigmoidal ($\kappa, \theta \in \mathbb{R}$) and inverse multi-quadric ($c \in \mathbb{R}_+$) kernel functions are among the most common ones. While RBF and polynomial are known to fulfill Mercers condition, this is not always the case for sigmoidal kernels (Smola et al., 1998b). Further valid kernels proposed in the context of regularization networks are e.g. multiquadric or spline kernels (Poggio and Girosi, 1990; Girosi et al., 1993; Smola et al., 1998b).

$$\text{Gaussian RBF} \qquad \mathrm{k}(\mathbf{x}, \mathbf{z}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{z}\|^2}{c}\right) \tag{2.35}$$

$$\text{Polynomial} \qquad \mathrm{k}(\mathbf{x}, \mathbf{z}) = ((\mathbf{x} \cdot \mathbf{z}) + \theta)^d \tag{2.36}$$

$$\text{Sigmoidal} \qquad \mathrm{k}(\mathbf{x}, \mathbf{z}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{z}) + \theta)$$

$$\text{inverse multi-quadric} \qquad \mathrm{k}(\mathbf{x}, \mathbf{z}) = \frac{1}{\sqrt{\|\mathbf{x} - \mathbf{z}\|^2 + c^2}}$$

for special applications like DNA analysis can be found in Vapnik (1998); Stitson et al. (1997); Smola et al. (1998b); Haussler (1999); Jaakkola et al. (2000); Zien et al. (2000); Tsuda et al. (2002) and numerous others. Finally, let us note, that we can recover the original linear algorithms by simply using linear kernels, i.e. the scalar product in the original input space: $\mathrm{k}(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})$. However, using linear kernels to solve the original (linear) problem will usually be not very efficient. For the kernel version, even if we use a linear kernel, we need to estimate as many parameters as we observe training examples, i.e. M parameters. Using the kernel formulation is only advisable, if the dimensionality N of the data is larger than the number of examples M.

In Figure 2.10 a toy example of a non-linear SVM with a RBF kernel can be found. We see that introducing non–linearity through the kernel function the dataset becomes *linearly* separable in the feature space $\mathcal{F}$. This results in a non-linear decision function in the input space which is shown here.

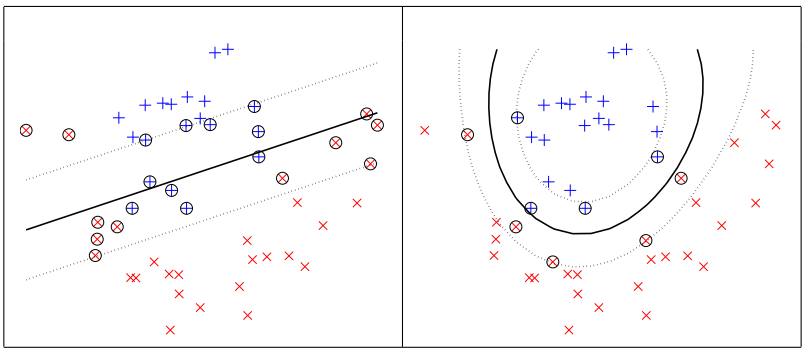

Figure 2.10: Illustration of a linear (left panel) and a nonlinear SVM with RBF kernel (2.35) (right panel) on a toy dataset. Training examples from the two classes are represented by red 'x' and blue '+', respectively. The solid curve shows the decision surface, i.e. $(\mathrm{w} \cdot \Phi(\mathrm{x})) + b = 0$, the dotted curves show the margin area, i.e. $(\mathrm{w} \cdot \Phi(\mathrm{x})) + b = \pm 1$. Support vectors are marked by small circles. It can be observed that only very few training patterns become support vectors, namely those that are inside the margin area or misclassified.

2.4 Summary

We have seen how the problem of learning from data can be cast formally into the problem of estimating functions from given observations. We reviewed some basic notations and concepts from statistics and especially from statistical learning theory. The latter provides us with two extremely important insights: (i) not the dimensionality of the data but the complexity of the functions class we choose our estimate from matters, (ii) for successful learning it is desirable to have consistency. Closely related to these two questions is the issue of regularization. Regularization allows us to *control* the complexity of our learning machine and often suffices to achieve consistency.

As an application of statistical learning theory we reviewed maximum margin hyperplanes. Whilst it is satisfactory to have a technique at hand that implements (at least partially) what the theory justifies, the algorithm is only capable of finding (linear) hyperplanes. To circumvent this restriction we introduced kernel functions yielding support vector machines. Kernel functions allow us to reformulate many algorithms in some kernel feature space that is nonlinearly related to the input space and yield powerful, non-linear techniques. This non-linearization using the kernel trick is possible whenever we are able to express an algorithm such that it only uses the data in the form of scalar products. However, since the algorithms are still linear in the feature space we can use the same theory and optimization strategies as before.

Kernel algorithms have seen a powerful development over the past years, starting with the support vector machine[12]. Among many theoretical (Shawe-Taylor et al., 1996; Williamson et al., 1998; Graepel et al., 2000; Bartlett et al., 2002) and algorithmic advances (Platt, 1999; Joachims, 1999; Keerthi et al., 1999) new algorithms using the kernel-trick have been proposed (e.g. Kernel PCA (Schölkopf et al., 1998b), one-class SVM (Schölkopf et al., 2000) or Bayes–Point machines (Herbrich et al., 2001)).[13] This development is still an ongoing and exciting field of study as we will see in the next chapters.

[12] Techniques based on kernel functions have been known in statistics for a very long time (e.g. Parzen, 1962; Hand, 1982). However, these techniques usually use kernels to estimate probability densities - they make no use of the interpretation as scalar products.

[13] See `http://www.kernel-machines.org/` for a collection of literature, data sets and implementations of kernel based learning techniques.

Chapter 3

Kernel Fisher Discriminants

> Der gesunde Menschenverstand (bon sens) ist die bestverteilte Sache der Welt, denn jedermann meint, damit so gut versehen zu sein, daß selbst diejenigen, die in allen übrigen Dingen sehr schwer zu befriedigen sind, doch gewöhnlich nicht mehr Verstand haben wollen, als sie wirklich haben.
>
> René Descartes

This chapter introduces a particular learning technique called Kernel Fisher Discriminants. Following the ideas outlined in the previous chapter Kernel Fisher Discriminants are a non–linear generalization of Fisher's Discriminant by using kernel functions. Also, we will derive several interesting variants of kernel Fisher discriminants together with algorithms to solve the resulting optimization problems. Finally we discuss connections to other techniques and the relation to other work.

THE goal of discriminant analysis can be summarized as that of finding a function returning scalar values which allow a good discrimination between different classes of the input data. These discriminants are subsequently used to train e.g. a classifier or to visualize certain aspects of the data. In this sense discriminant analysis can be understood as supervised preprocessing or feature extraction, supervised in the sense that we tell the learning algorithm which training examples are connected to which property.

3.1 Linear Discriminants

More formally one is looking for a function $f : \mathcal{X} \to \mathbb{R}^D$, such that $f(\mathbf{x})$ and $f(\mathbf{z})$ are similar whenever $\mathbf{x}$ and $\mathbf{z}$ are, and different otherwise. Similarity is usually

measured by class membership and Euclidean distance. In the special case of linear discriminant analysis one is seeking a linear function, i.e. a set of projections

$$f(\mathbf{x}) = W^\top \mathbf{x}, \quad W \in \mathbb{R}^{N \times D},$$

where the matrix W is chosen, such that a contrast criterion G is optimized, in some cases with respect to a set of constraints $\mathcal{S}$, i.e.

$$\max G(W) \text{ subject to } W \in \mathcal{S}. \tag{3.1}$$

This setup is absolutely equivalent to e.g. principal component analysis where the contrast criterion would be that of maximal variance (or least mean squared error) and the constraint set would be that of orthogonality of the matrix W. However, PCA is an unsupervised technique and does not use any labels. There is no guarantee that the directions found by PCA will be particularly discriminative.

To simplify the presentation we will in the following only consider one-dimensional discriminant functions, i.e. f is of the form $f = (\mathbf{w} \cdot \mathbf{x})$. However, most results can easily be generalized to the multidimensional case.

3.2 Fisher's Discriminant

Probably the most well known example of a linear discriminant is Fisher's discriminant (Fisher, 1936). Fisher's idea was to look for a direction $\mathbf{w}$ that separates the class means well (when projected onto the found direction) while achieving a small variance around these means. The hope is that it is easy to decide for either of the two classes from this projection with a small error. The quantity measuring the difference between the means is called *between class variance* and the quantity measuring the variance around these class means is called *within class variance*, respectively. Then the goal is to find a direction that maximizes the between class variance while minimizing the within class variance at the same time. This is illustrated in Figure 3.1. To describe this mathematically let $\mathcal{X}$ denote the space of observations (e.g. $\mathcal{X} \subseteq \mathbb{R}^N$) and $\mathcal{Y}$ the set of possible labels (here $\mathcal{Y} = \{+1, -1\}$). Furthermore, let $\mathcal{Z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)\} \subseteq \mathcal{X} \times \mathcal{Y}$ denote the training sample of size M and denote by $\mathcal{Z}_1 = \{(\mathbf{x}, y) \in \mathcal{Z} | y = 1\}$ and $\mathcal{Z}_2 = \{(\mathbf{x}, y) \in \mathcal{Z} | y = -1\}$ the split into the two classes of size $M_i = |\mathcal{Z}_i|$. Define $\mathbf{m}_1$ and $\mathbf{m}_2$ to be the empirical class means, i.e.[1]

$$\mathbf{m}_i = \frac{1}{M_i} \sum_{x \in \mathcal{Z}_i} \mathbf{x}.$$

Similarly, we can compute the means of the data projected onto some direction $\mathbf{w}$ by

$$\begin{aligned} \mu_i &= \frac{1}{M_i} \sum_{x \in \mathcal{Z}_i} \mathbf{w}^\top \mathbf{x} \\ &= \mathbf{w}^\top \mathbf{m}_i, \end{aligned} \tag{3.2}$$

[1] With a slight abuse of notation we denote the x in $(x, y) \in \mathcal{Z}$ by $x \in \mathcal{Z}$.

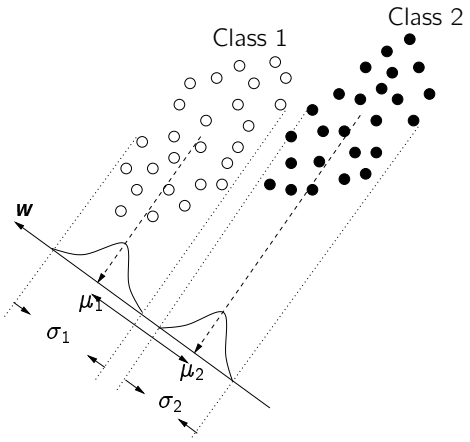

Figure 3.1: Illustration of Fisher's discriminant for two classes. We search for a direction w, such that the difference between the class means projected onto this directions (μ_1 and μ_2) is large and such that the variance around these means (σ_1 and σ_2) is small.

i.e. the means μ_i of the projections are the projected means $\mathbf{m}_i$. The variances[2] σ_1, σ_2 of the projected data can be expressed as

$$\sigma_i = \sum_{x \in \mathcal{Z}_i} (\mathbf{w}^\top \mathbf{x} - \mu_i)^2. \tag{3.3}$$

Then maximizing the between class variance and minimizing the within class variance can be achieved by maximizing

$$G(\mathbf{w}) = \frac{(\mu_1 - \mu_2)^2}{\sigma_1 + \sigma_2}, \tag{3.4}$$

which will yield a direction $\mathbf{w}$ such that the ratio of between-class variance (i.e. separation) and within-class variance (i.e. overlap) is maximal. Now, substituting the expression (3.2) for the means and (3.3) for the variances into (3.4) yields

$$G(\mathbf{w}) = \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w}}, \tag{3.5}$$

where we define the between and within class scatter matrices S_B and S_W as

$$S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top \qquad S_W = \sum_{i=1,2} \sum_{x \in \mathcal{Z}_i} (\mathbf{x} - \mathbf{m}_i)^2. \tag{3.6}$$

It is straight forward to check that (3.4) is absolutely equivalent to (3.5). This perfectly fits into the framework (3.1) with an empty constraint set $\mathcal{S}$. The quantity in Equation (3.5) is often referred to as a Rayleigh coefficient.

Finding w

One particularly nice property of Fisher's discriminant is that (i) (3.5) has a global solution (although not necessarily unique) and (ii) that a such a globally optimal

[2] Strictly speaking σ_i is the unnormalized variance which sometimes is also called scatter.

$\mathbf{w}$ maximizing (3.5) can be found by solving an eigenvalue problem. It is well known, that the $\mathbf{w}$ maximizing (3.5) is the leading eigenvector of the generalized eigenproblem

$$S_B\mathbf{w} = \lambda S_W\mathbf{w}. \tag{3.7}$$

This can be seen as follows: differentiating (3.5) with respect to $\mathbf{w}$ yields

$$\begin{aligned} & (\mathbf{w}^\top S_B\mathbf{w})S_W\mathbf{w} - (\mathbf{w}^\top S_W\mathbf{w})S_B\mathbf{w} = 0 \\ \Leftrightarrow \quad & S_B\mathbf{w} = \frac{\mathbf{w}^\top S_B\mathbf{w}}{\mathbf{w}^\top S_W\mathbf{w}} S_W\mathbf{w}. \end{aligned} \tag{3.8}$$

From the last equation it follows immediately that $\mathbf{w}$ must be a generalized eigenvector of (3.7). That $\mathbf{w}$ must also be the leading eigenvector (i.e. the quantity $\frac{\mathbf{w}^\top S_B\mathbf{w}}{\mathbf{w}^\top S_W\mathbf{w}}$ in (3.8) is the largest eigenvalue of (3.7)) can be seen as follows: Assume there is an eigenvector $\widetilde{\mathbf{w}}$ of (3.7) with corresponding eigenvalue $\widetilde{\lambda}$ such that for the optimal solution $\mathbf{w}_*$ to (3.5) $G(\mathbf{w}_*) < \widetilde{\lambda}$. Then evaluating (3.7) at $\widetilde{\mathbf{w}}$ and multiplying with $\widetilde{\mathbf{w}}^\top$ yields

$$\begin{aligned} S_B\widetilde{\mathbf{w}} = \widetilde{\lambda} S_W\widetilde{\mathbf{w}} \quad & \Rightarrow \quad \widetilde{\mathbf{w}}^\top S_B\widetilde{\mathbf{w}} = \widetilde{\lambda}\widetilde{\mathbf{w}}^\top S_W\widetilde{\mathbf{w}} \\ & \Rightarrow \quad \frac{\widetilde{\mathbf{w}}^\top S_B\widetilde{\mathbf{w}}}{\widetilde{\mathbf{w}}^\top S_W\widetilde{\mathbf{w}}} = \widetilde{\lambda} \\ & = \quad G(\widetilde{\mathbf{w}}) \\ & > \quad G(\mathbf{w}_*), \end{aligned}$$

where the last inequality follows from our assumption that $G(\mathbf{w}_*) < \widetilde{\lambda}$. This, however, is a contradiction to the assumption that $\mathbf{w}_*$ was the optimal solution to (3.5).

Examining the eigenproblem (3.8) closer one finds an even simpler way of obtaining the optimal $\mathbf{w}$. First, note that the way S_B is defined, $S_B\mathbf{w}$ will always point in the direction of $\mathbf{m}_2 - \mathbf{m}_1$. Secondly, from (3.5) we see that only the direction of $\mathbf{w}$ matters, not its length. Hence if we multiply both sides of (3.7) by S_W^{-1} and drop all scalar factors we get:

$$S_W^{-1}S_B\mathbf{w} = \lambda\mathbf{w} \qquad \equiv \qquad \mathbf{w} = S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1),$$

i.e. we can find an optimal direction $\mathbf{w}$ by inverting the within class scatter matrix S_W.

3.2.1 Connection to Least Squares

The Fisher discriminant problem described above bears strong connections to least squares approaches for classification[3]. Classically, one is looking for a linear discriminant function, now including a bias term, i.e.

$$f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b, \tag{3.9}$$

such that on the training sample the sum of squares error between the outputs $f(\mathbf{x}_i)$ and the known targets y_i is small, i.e. in a (linear) least squares approach

[3]The result presented here is adopted from Duda and Hart (1973) and slightly generalized.

one minimizes the sum of squares error

$$E(\mathbf{w}, b) = \sum_{(\mathrm{x},y)\in\mathcal{Z}} \left(f(\mathbf{x}) - y\right)^2 = \sum_{(\mathrm{x},y)\in\mathcal{Z}} \left(\mathbf{w}^\top\mathbf{x} + b - y\right)^2. \tag{3.10}$$

The least squares problem $\min_{\mathrm{w},b} E(\mathbf{w}, b)$ can be written in matrix notation as

$$\min_{\mathrm{w},b} \left\| \begin{bmatrix} X_1^\top & \mathbf{1}_1 \\ X_2^\top & \mathbf{1}_2 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} - \begin{bmatrix} -\mathbf{1}_1 \\ \mathbf{1}_2 \end{bmatrix} \right\|^2 \tag{3.11}$$

where $X = [X_1, X_2]$ is a matrix containing all training examples partitioned according to the labels ± 1, and $\mathbf{1}_i$ is a vector of ones of corresponding length. The solution to a least squares problem of the form $\|A\mathbf{x} - b\|^2$ can be computed by using the pseudo-inverse of A, i.e. $\mathbf{x}^* = A^\dagger b = (A^\top A)^{-1}A^\top b$, assuming that $A^\top A$ is not singular. Then $A^\dagger A = I$ and thus a necessary and sufficient condition for the solution $\mathbf{x}^*$ to the least squares problem is $(A^\top A)\mathbf{x}^* = A^\top b$. Applying this to (3.11) yields:

$$\begin{bmatrix} X_1 & X_2 \\ \mathbf{1}_1^\top & \mathbf{1}_2^\top \end{bmatrix} \begin{bmatrix} X_1^\top & \mathbf{1}_1 \\ X_2^\top & \mathbf{1}_2 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ \mathbf{1}_1^\top & \mathbf{1}_2^\top \end{bmatrix} \begin{bmatrix} -\mathbf{1}_1 \\ \mathbf{1}_2 \end{bmatrix}$$

Multiplying this matrices and using the definition of the sample means and within–class scatter for Fisher yields:

$$\begin{bmatrix} S_W + M_1\mathbf{m}_1\mathbf{m}_1^\top & M_1\mathbf{m}_1 + M_2\mathbf{m}_2 \\ (M_1\mathbf{m}_1 + M_2\mathbf{m}_2)^\top & M_1 + M_2 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} M_2\mathbf{m}_2 - M_1\mathbf{m}_1 \\ M_2 - M_1 \end{bmatrix} \tag{3.12}$$

Using the second equation in (3.12) to solve for b yields

$$b = \frac{M_2 - M_1 - (M_1\mathbf{m}_1 + M_2\mathbf{m}_2)^\top\mathbf{w}}{M_1 + M_2}. \tag{3.13}$$

Substituting this into the first equation of (3.12) and using a few algebraic manipulations, especially the relation $a - \frac{a^2}{a+b} = \frac{ab}{a+b}$ one obtains

$$\left(S_W + \frac{M_1 M_2}{M_1 + M_2} S_B\right)\mathbf{w} + \frac{M_1^2 + M_2^2}{M_1 + M_2}\left(\mathbf{m}_2 - \mathbf{m}_1\right) = 0 \tag{3.14}$$

Now, since still $S_B\mathbf{w}$ is in the direction of $(\mathbf{m}_2 - \mathbf{m}_1)$, there exists a scalar $\alpha \in \mathbb{R}$ such that

$$\frac{M_1 M_2}{M_1 + M_2} S_B\mathbf{w} = -\left(\frac{M_1^2 + M_2^2}{M_1 + M_2} - \alpha\right)\left(\mathbf{m}_2 - \mathbf{m}_1\right), \tag{3.15}$$

Then using (3.15) in (3.14) yields:

$$S_W\mathbf{w} = \alpha(\mathbf{m}_2 - \mathbf{m}_1) \Leftrightarrow \mathbf{w} = \alpha S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1). \tag{3.16}$$

This shows that the solution to the least squares problem is in the same direction as the solution of Fisher's discriminant, although it will have a different length. But as we already noticed, we are only interested in the *direction* of $\mathbf{w}$, not its length and hence the solutions are identical.

3.2.2 Bayes Optimality

Another well known fact about Fisher's discriminants is, that they are the Bayes optimal solution if the two classes are distributed according to a normal distribution, both with equal covariance matrix Σ (see Bishop, 1995; Duda and Hart, 1973)[4]. If the class conditional densities are given by a normal distribution they are of the form

$$p(\mathbf{x}|y=1) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_1)^\top \Sigma^{-1}(\mathbf{x}-\mathbf{m}_1)\right), \quad (3.17)$$

$$p(\mathbf{x}|y=-1) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_2)^\top \Sigma^{-1}(\mathbf{x}-\mathbf{m}_2)\right). \quad (3.18)$$

Computing the posterior probability using Bayes' theorem (cf. (2.1)), we get

$$P(y=1|\mathbf{x}) = \frac{p(\mathbf{x}|y=1)P(y=1)}{p(\mathbf{x}|y=1)P(y=1) + p(\mathbf{x}|y=-1)P(y=-1)},$$

what in turn can be written as

$$P(y=1|\mathbf{x}) = \frac{1}{1+\exp(-a)}, \quad (3.19)$$

with

$$a = \log\left(\frac{p(\mathbf{x}|y=1)P(y=1)}{p(\mathbf{x}|y=-1)P(y=-1)}\right).$$

Substituting (3.17) and (3.18) into (3.19) yields

$$a = \mathbf{w}^\top \mathbf{x} + b,$$

with

$$\mathbf{w} = \Sigma^{-1}(\mathbf{m}_1 - \mathbf{m}_2),$$

and

$$b = \frac{1}{2}\mathbf{m}_2^\top \Sigma^{-1}\mathbf{m}_2 - \frac{1}{2}\mathbf{m}_1^\top \Sigma^{-1}\mathbf{m}_1 + \log\left(\frac{P(y=1)}{P(y=-1)}\right).$$

This direction $\mathbf{w}$ is again, in an empirical version, equivalent up to a scaling factor to the direction found by Fisher's discriminant.

This also reveals one possibility how to make decisions when using Fisher's discriminant. Assuming that the class conditional distributions are indeed equal normal distributions, once $\mathbf{w}$ is known the class posterior probabilities can be written as

$$P(y=1|\mathbf{x}) = P(y=1|q(\mathbf{x})) = \frac{p(q(\mathbf{x})|y=1)P(y=1)}{p(q(\mathbf{x})|y=1)P(y=1) + p(q(\mathbf{x})|y=-1)P(y=-1)},$$

[4] If the covariance matrices for the classes are different but both class conditional distributions are still normal, the optimal decision boundary is given by quadratic discriminants (see Fukunaga, 1990).

and likewise for $P(y=-1|\mathbf{x})$ where $q \equiv q(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ and

$$p(\mathbf{x}|y=1) = \quad p(q(\mathbf{x})|y=1) \quad = \left(2\pi\sigma_1^2\right)^{-1/2} \exp\left(-\frac{(q(\mathbf{x})-\mu_1)^2}{2\sigma_1^2}\right),$$
$$p(\mathbf{x}|y=-1) = \quad p(q(\mathbf{x})|y=-1) \quad = \left(2\pi\sigma_2^2\right)^{-1/2} \exp\left(-\frac{(q(\mathbf{x})-\mu_2)^2}{2\sigma_2^2}\right),$$

μ_i and σ_i being defined in (3.2) and (3.3), respectively. The class priors can e.g. be estimated from the training sample $\mathcal{Z}$. We decide for class 1 whenever $P(y=1|\mathbf{x}) \geq \frac{1}{2}$, and class -1 otherwise.

3.2.3 Why Fisher can be really bad

Whilst it is satisfactory that there is a special setting in which Fishers discriminants are the best possible model and the way of fitting this model yields the theoretically optimal solution, one can construct a distribution of the training sample $\mathcal{Z}$ for which the directions found by Fisher can be arbitrarily bad (see Devroye et al., 1996, Problem 4.9). In particular, one can show that for every $\varepsilon > 0$ there exist a distribution with $\mathcal{X} = \mathbb{R}^2$, such that the two classes are linearly separable but even with the best possible threshold b on $\mathbf{w}^\top\mathbf{x} + b$ the expected error of $\mathbf{w}$ solving Fisher's discriminants is larger than $1/2 - \varepsilon$. Even worse, if one chooses b such as to minimize the least squares error between the labels and the output, the expected error will be larger than $1 - \varepsilon$. However, these constructed examples also exist for other techniques and are of little relevance in practical applications. Especially, having found a direction such that the error is $1 - \epsilon$ we can turn it into a solution with error only ϵ by using the negative weight vector $\mathbf{w}$.

3.3 Other Discriminant Techniques

Besides the within–class vs. between-class criterion optimized by Fisher's discriminant there are others (linear) techniques that are similar but use slightly different contrast criteria. Linear discriminants have strong connections to single-layer neural networks (see Duda and Hart, 1973; Bishop, 1995). Again considering only two classes, one generalizes the concept of linear discriminant functions such as to include a bias term, i.e. one is looking for functions of the form $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b$. As we have already shown in Section 3.2.1, fitting the parameters of such a model using the least squares error criterion yields again Fisher's discriminant. In a slightly more general setting called logistic discrimination (McCulloch and Pitts, 1943; Anderson, 1982), one adds a non-linear wrapper function to the outputs, i.e. uses $f(\mathbf{x}) = g(\mathbf{w}^\top\mathbf{x} + b)$ for a suitably chosen g. As we have seen in Section 3.2.2, $g(a) = \frac{1}{1+\exp(-a)}$ is the theoretically optimal choice if one assumes normal distributions with identical covariance structure for the class-conditional densities and, not surprising, also yields the direction found by Fisher's discriminants for the optimal $\mathbf{w}$.

Finally, if we introduce a set of D predefined nonlinear functions $\phi_i(\mathbf{x})$ acting

directly on the examples, i.e. search for a discriminant function of the form

$$f(\mathbf{x}) = \sum_{i=1}^{D} w_i \phi_i(\mathbf{x}) + b,$$

we arrive at what is called generalized linear discriminants. The resulting model will be a nonlinear function. However, since f is linear in the parameters $\mathbf{w}$ and b, optimization can be carried out by e.g. least squares fitting. A typical example would be radial-basis function (RBF) networks (e.g. Moody and Darken, 1989) in which one chooses $\phi_i(\mathbf{x}) = \exp(-\|\mathbf{x}_i - \mathbf{x}\|^2/\sigma)$ for some $\sigma \geq 0$. If we also optimize over e.g. the parameter σ of the basis function ϕ_i, we get a model which is nonlinear in its parameters and optimization has to be done using e.g. gradient descent.

There is a large body of work extending the classical linear discriminant framework by e.g. introducing special non-linearities, imposing certain types of regularization, and using special learning algorithms to find the optimal direction (Friedman, 1989; McLachlan, 1992; Hastie and Tibshirani, 1996; Hastie et al., 1995; Grove et al., 1997). But these techniques are often prone to local minima and lack an intuitive interpretation. However, this goes beyond the scope of this work and the reader is referred to the literature. Instead, we will consider relevant parts of the existing theory during the derivation of the special non-linear variant of Fisher's discriminant proposed in this thesis.

3.4 Introducing Kernels

As we have seen before, linear discriminants are not always optimal. Worse, even rather simple problems like the classical XOR-problem can not be solved using linear functions (cf. Figure 2.7). Hence, the question arises if one can modify linear discriminants such that one retains most of their appealing properties but gets the flexibility to solve problems which need non-linear decision functions. The most noticeable properties of e.g. Fishers discriminant are

1. The existence of a global solution and the absence of local minima.
2. This solution can be found in closed form.
3. A clear interpretation.

In the last paragraph we already mentioned non-linear variations of linear discriminants, most noticeably generalized discriminants passing the input examples through some non-linear functions. While for an appropriate choice of basis functions it is in principle possible to approximate any (decision) functions to an arbitrary precision, it is only seldom possible to find an satisfactory interpretation of the result. If the non-linearities include parameters which should be optimized as well (as e.g. in multilayer neural networks) we get a difficult optimization problem prone to local minima. Here, instead of hand-crafting a non-linear preprocessing we propose to use kernel functions as introduced in Chapter 2. They will allow for powerful, non-linear decision functions while still leaving us with a relatively simple

optimization problem, a global solution and a straight forward interpretation of what one is actually optimizing for.

As outlined in Chapter 1 the kernel trick amounts to performing the same algorithm as before, but implicitly in the kernel Hilbert space $\mathcal{E}$ connected to the kernel function used. Since for each kernel there exist a mapping $\Phi : \mathcal{X} \to \mathcal{E}$, such that $k(\mathbf{x}, \mathbf{z}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}))$, one is looking for a discriminant of the form

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}),$$

but now $\mathbf{w} \in \mathcal{E}$ is a vector in the feature space $\mathcal{E}$. As the only way for us to work in the feature space $\mathcal{E}$ is by using the kernel function, we need, as outlined for other techniques in Chapter 2, a formulation of Fisher's discriminants that only uses Φ in such dot-products.

3.4.1 Rayleigh Coefficients in Feature Space

Before deriving an explicit expression for Fisher's discriminant in a kernel feature space let us take a short detour. To find the optimal linear discriminant we need to maximize a Rayleigh coefficient (cf. Equation (3.5)). We arrived there by looking for a suitable contrast criterion to use in the setting of (3.1). As we already mentioned, also linear PCA can be interpreted in this framework, i.e. PCA can be expressed as the optimization of a Rayleigh coefficient. The largest principal component can be found by solving

$$G(\mathbf{w}) = \frac{\mathbf{w}^\top C \mathbf{w}}{\mathbf{w}^\top \mathbf{w}}.$$

Although PCA is an unsupervised technique whose assumption is that large variance in the data is an interesting feature, it is rather similar to Fisher's discriminant in spirit. Fisher's discriminants can also be interpreted as a feature extraction technique, only that now "interesting" is defined by the separability criterion (3.5). From this point of view, we can think of the Rayleigh coefficient as a general tool to find features which (i) cover much of what is considered to be interesting (e.g. variance in PCA) and at the same time avoid what is considered disturbing (e.g. within class variance in Fisher's discriminants). Of course there are more criteria one could be interested in to impose on the extracted features. Often one has prior information available that can be used to formulate improved feature quality criteria or, as in the case of Fisher, the features are extracted for a certain purpose, e.g. for subsequently training some classifier. For instance, we might know that the examples are corrupted by noise of a specific signature or that there are invariance transformations under which a classification should not change. These concepts, of known noise or transformation invariance, are closely related, i.e. they can both be interpreted as causing a change in the feature, which should be avoided: given two examples that are variations of the same observation, feature values extracted from these examples should be equal or at least very similar. Clearly, invariance alone is never a sufficient condition for a good feature, as we could simply take the constant feature. What one would like to obtain is a feature, which is as invariant as possible while still covering as much of the information necessary for describing the data's properties of interest.

Having this in mind, Rayleigh coefficients become an even more natural choice. Maximize

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top S_I \mathbf{w}}{\mathbf{w}^\top S_N \mathbf{w}}, \tag{3.20}$$

where S_I and S_N are symmetric matrices designed such that they measure the desired information and the undesired noise along the direction of $\mathbf{w}$. The ratio in (3.20) is maximized when one covers as much as possible of the desired information while avoiding the undesired. We have already shown in Section 3.2 that this problem can be solved via a generalized eigenproblem. By using the same technique, one can also compute second, third, etc., generalized eigenvectors from the generalized eigenproblem, for example in PCA where we are usually looking for more than just one feature. With the appropriate choices for S_I and S_N we recover many well known techniques (see also Figure 3.2).

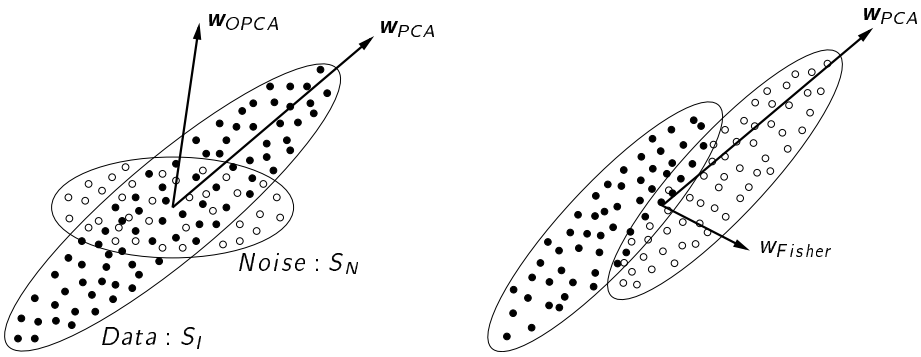

Figure 3.2: Illustrative comparison of principal component analysis (PCA), oriented PCA (OPCA), and Fisher's discriminant. Shown is the direction of the (first) feature extractor computed by these techniques.

3.4.2 Choices for Covariance Matrices

We will now discuss, how different choices for the covariance or scatter matrices S_I and S_N in (3.20) lead to different, well-known techniques. In the presentation we already assume that we want to carry out these algorithms in a kernel feature space $\mathcal{E}$ with associated mapping Φ.

PCA and oriented PCA We get the full covariance of a data set $\mathcal{Z}$ in the feature space $\mathcal{E}$ by

$$C = \frac{1}{M} \sum_{x \in \mathcal{Z}} (\Phi(\mathbf{x}) - \mathbf{m})(\Phi(\mathbf{x}) - \mathbf{m})^\top, \tag{3.21}$$

with

$$\mathbf{m} = \frac{1}{M} \sum_{x \in \mathcal{Z}} \Phi(\mathbf{x}),$$

which can be used as S_I in kernel PCA (Schölkopf et al., 1998b). For kernel PCA S_N would just be the identity matrix. A generalization of PCA, called oriented PCA

(Diamantaras and Kung, 1996), tries to improve the directions of PCA by taking into account known noise variance. The idea is to find directions with high (or low) variance that are different from the known directions of high variance. This is achieved by setting S_N to an estimate of this unwanted variance. In practice one can compute S_N the same way as the covariance in (3.21) but over (mapped) examples sampled from the assumed noise distribution. In summary, choosing the matrices this way we obtain nonlinear versions of PCA and oriented PCA.

Fisher's discriminant The standard formulation of the Fisher discriminant in $\mathcal{E}$, yielding the *kernel Fisher discriminant* (KFD) proposed in Mika et al. (1999a, 2000) (see also Roth and Steinhage (2000); Baudat and Anouar (2000)) is given by

$$\begin{aligned} S_N &= \sum_{i=1,2} \sum_{x \in \mathcal{Z}_i} (\Phi(\mathbf{x}) - \mathbf{m}_i)(\Phi(\mathbf{x}) - \mathbf{m}_i)^\top \\ \text{and } S_I &= (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top, \end{aligned} \tag{3.22}$$

with the within-class scatter as S_N, and the between class scatter as S_I. Here $\mathbf{m}_i$ is the sample mean for examples from class i. This is in complete analogy to the linear algorithm discussed earlier.

Transformation Invariance To incorporate a known invariance e.g. in oriented kernel PCA, one can use a matrix similar to the tangent covariance matrix (Schölkopf, 1997; Simard et al., 1998),

$$T = \sum_{x \in \mathcal{Z}} (\Phi(\mathbf{x}) - \Phi(\mathcal{L}_t \mathbf{x}))(\Phi(\mathbf{x}) - \Phi(\mathcal{L}_t \mathbf{x}))^\top, \tag{3.23}$$

for some small $t > 0$. Here $\mathcal{L}_t$ is a 1-parameter transformation. T can be seen as a finite difference approximation of the covariance of the tangent of $\mathcal{L}_t$ at point $\Phi(\mathbf{x})$ (details e.g. in Schölkopf (1997)). Using $S_I = C$ and $S_N = T$ in oriented kernel PCA, we impose invariance under the local transformation $\mathcal{L}_t$ by penalizing $\sum_{x \in \mathcal{Z}} (\mathbf{w} \cdot (\Phi(\mathbf{x}) - \Phi(\mathcal{L}_t(\mathbf{x}))))^2$. It is important to note that this matrix is not only constructed from the training examples in $\mathcal{Z}$, but also from the transformations generated by $\mathcal{L}_t(\mathbf{x})$.[5]

Combination Of course, we are free to combine any of these matrices to get both, say invariance and discrimination, by setting S_N to a weighted sum of the tangent covariance matrix and the within class scatter. Finally, we can also add several tangent covariance matrices to impose invariances under more than just one transformation or add different quality criteria to compute S_I. In the general case we obtain

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \left(\sum_i S_{Ii}\right) \mathbf{w}}{\mathbf{w}^\top \left(\sum_i S_{Ni}\right) \mathbf{w}}, \tag{3.24}$$

where for simplicity the appropriate weighting is subsumed in the matrices.

[5] The impact of this observation will become apparent in the next paragraph when the expansion (3.25) for the solution w is derived.

3.4.3 Formulation using kernel functions

Following the reasoning in Chapter 2, to optimize (3.20) or (3.24) in some kernel feature space $\mathcal{E}$ we need to find a formulation, which uses only dot products of Φ-images. As numerator and denominator are both scalars, this can be done independently. Furthermore, under some mild assumptions on S_I and S_N outlined below (mild in the sense, that all choices above but (3.23) fulfill them), every solution $\mathbf{w} \in \mathcal{E}$ can be written as an expansion in terms of mapped training data, i.e.

$$\mathbf{w} = \sum_{x \in \mathcal{Z}}^{M} \alpha_x \Phi(\mathbf{x}), \quad \alpha_x \in \mathbb{R}. \tag{3.25}$$

This is an important issue in any kernel based learning technique: Since we can only access the feature space $\mathcal{E}$ by means of the kernel function, either because it is too high or infinite dimensional, or because we do not even know the actual feature space connected to the kernel k, it would be impossible and quite useless to obtain an explicit solution $\mathbf{w} \in \mathcal{E}$. Contrary, the use of expansion (3.25) makes things tractable. To see that this expansion is valid, consider symmetric operators S on the finite-dimensional subspace spanned by the $\Phi(\mathbf{x}_i)$ in a possibly infinite space $\mathcal{E}$, e.g. any matrix S, which is exclusively constructed from the $\Phi(\mathbf{x}_i)$ by means of linear operations. Furthermore, let $\mathbf{w} = \mathbf{v}_1 + \mathbf{v}_2$ be a decomposition of $\mathbf{w}$ into a part $\mathbf{v}_1$ lying in the span of the training data, i.e. $\mathbf{v}_1 \in \text{span}\{\Phi(\mathbf{x}_i) : i = 1, \ldots, M\}$ and a part $\mathbf{v}_2$ in its orthogonal complement, i.e. $\mathbf{v}_2 \perp \text{span}\{\Phi(\mathbf{x}_i) : i = 1, \ldots, M\}$. Then for any such S,

$$\begin{aligned} \langle \mathbf{w}, S\mathbf{w} \rangle &= \langle (\mathbf{v}_1 + \mathbf{v}_2), S(\mathbf{v}_1 + \mathbf{v}_2) \rangle \\ &= \langle \mathbf{v}_1, S\mathbf{v}_1 \rangle, \end{aligned}$$

using $S\mathbf{v}_2 = 0$ and $\langle \mathbf{v}, S\mathbf{v} \rangle = \langle S\mathbf{v}, \mathbf{v} \rangle$ for symmetric matrices. As $\mathbf{v}_1$ lies in the span of the $\Phi(\mathbf{x}_i)$ and S, by construction, only operates on this subspace, $S\mathbf{v}_1$ lies in $\text{span}\{\Phi(\mathbf{x}_i)\}$ as well. Thus, for any such quadratic form $\mathbf{w}^\top S\mathbf{w}$ it is sufficient to consider that part of $\mathbf{w}$, which lies in the span of the examples: There already exist an expansion of the form (3.25) for $\mathbf{w}$, which maximizes (3.20) or (3.24) (in $\mathcal{E}$), respectively (see also Schölkopf et al. (2001)).

Multiplying either of the matrices proposed for S_I and S_N or a sum thereof from the left and right with the expansion (3.25), we can find a formulation that uses only dot products. In the following we exemplify this procedure by giving the explicit derivation of (3.20) in the feature space $\mathcal{E}$ for Fisher's discriminant (yielding kernel Fisher discriminants (KFD)).

3.4.4 Derivation of Kernel Fisher Discriminants

Let Φ be the non–linear mapping to the feature space $\mathcal{E}$. To find the linear discriminant in $\mathcal{E}$ (which is then nonlinear in the input space) we need to maximize

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_I \mathbf{w}}{\mathbf{w}^T S_N \mathbf{w}},$$

where now $\mathbf{w} \in \mathcal{E}$ and S_I and S_N are as in (3.22), where $m_i := \frac{1}{M_i} \sum_{x \in \mathcal{Z}_i} \Phi(\mathbf{x})$; M_i denotes the number of training examples in class i. To solve this using only

kernel functions

$$\mathrm{k}(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})), \tag{3.26}$$

we first need a formulation of (3.20) with the matrices (3.22) in terms of *dot products* only, which we then replace by some kernel function. We define K_x as the column corresponding to the element $\mathbf{x}$ in the kernel matrix $(K)_{\mathrm{xz}} = \mathrm{k}(\mathbf{x}, \mathbf{z})$, $\mathbf{x}, \mathbf{z} \in \mathcal{Z}$. Using the expansion (3.25) and the definition of $\mathbf{m}_i$ we write

$$\begin{aligned} \mathbf{w}^\top \mathbf{m}_i &= \frac{1}{M_i} \sum_{\mathrm{x} \in \mathcal{Z}} \sum_{\mathrm{z} \in \mathcal{Z}_i} \alpha_\mathrm{x} (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})) \\ &= \frac{1}{M_i} \sum_{\mathrm{x} \in \mathcal{Z}} \sum_{\mathrm{z} \in \mathcal{Z}_i} \alpha_\mathrm{x} \, \mathrm{k}(\mathbf{x}, \mathbf{z}) \\ &= \mathbf{f}\mathbf{f}^\top \bar{}_i \end{aligned} \tag{3.27}$$

where we define $(\bar{}_i)_j := \frac{1}{M_i} \sum_{\mathrm{x} \in \mathcal{Z}_i} K_\mathrm{x}$ and replaced the dot products by the kernel function (cf. (3.26)). Now consider the numerator of our Rayleigh coefficient. By using the definition of S_I and (3.27) it can be rewritten as

$$\begin{aligned} \mathbf{w}^\top S_I \mathbf{w} &= \mathbf{w}^\top (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top \mathbf{w} \\ &= \mathbf{f}\mathbf{f}^\top (\bar{}_2 - \bar{}_1)(\bar{}_2 - \bar{}_1)^\top \mathbf{f}\mathbf{f} \\ &= \mathbf{f}\mathbf{f}^\top M \mathbf{f}\mathbf{f}, \qquad (3.28) \\ &\quad \text{with} \quad M := (\bar{}_2 - \bar{}_1)(\bar{}_2 - \bar{}_1)^\top. \qquad (3.29) \end{aligned}$$

Considering the denominator, using (3.25), the definition of $\bar{}_i$ and a similar transformation as in (3.28) we find:

$$\begin{aligned} \mathbf{w}^\top S_N \mathbf{w} &= \mathbf{w}^\top \left[\sum_{i=1,2} \sum_{\mathrm{x} \in \mathcal{Z}_i} (\Phi(\mathbf{x}) - \mathbf{m}_i) (\Phi(\mathbf{x}) - \mathbf{m}_i)^\top \right] \mathbf{w} \\ &= \mathbf{f}\mathbf{f}^\top \left[\sum_{i=1,2} \sum_{\mathrm{x} \in \mathcal{Z}_i} (K_\mathrm{x} - \bar{}_i) (K_\mathrm{x} - \bar{}_i)^\top \right] \mathbf{f}\mathbf{f} \\ &= \mathbf{f}\mathbf{f}^\top (K(I - \mathbf{v}_1 \mathbf{v}_1^\top - \mathbf{v}_2 \mathbf{v}_2^\top) K^\top) \mathbf{f}\mathbf{f} \\ &= \mathbf{f}\mathbf{f}^\top N \mathbf{f}\mathbf{f} \qquad (3.30) \\ &\quad \text{with} \quad N := KDK^\top. \\ &\quad \text{and} \quad D := I - \mathbf{v}_1 \mathbf{v}_1^\top - \mathbf{v}_2 \mathbf{v}_2^\top, \end{aligned}$$

where I is the identity matrix and $\mathbf{v}_j$ is the vector with element $(\mathbf{v}_j)_i = 1/\sqrt{M_j}$ if the example i belongs to class j and zero otherwise.

Combining (3.28) and (3.30) we can find Fisher's discriminant in the feature space $\mathcal{E}$ by maximizing

$$J(\mathbf{f}\mathbf{f}) = \frac{\mathbf{f}\mathbf{f}^T M \mathbf{f}\mathbf{f}}{\mathbf{f}\mathbf{f}^T N \mathbf{f}\mathbf{f}}. \tag{3.31}$$

We still need to maximize a Rayleigh coefficient and everything we said before on how to find the optimal solution still holds true. However, now it is a quotient in terms of finitely many expansion coefficients $\mathbf{f}\mathbf{f}$, and not in terms of $\mathbf{w} \in \mathcal{E}$, which

would have been a potentially infinite-dimensional space. The projection of a new example $\mathbf{x}$ onto $\mathbf{w}$ in $\mathcal{E}$ can be computed by

$$(\mathbf{w} \cdot \Phi(\mathbf{z})) = \sum_{x \in \mathcal{Z}}^{M} \alpha_x \, k(\mathbf{x}, \mathbf{z}), \tag{3.32}$$

without explicit reference to the mapping Φ. If we are looking for more than one feature (e.g. in oriented kernel PCA) it is straightforward to check that their expansions are given by the subsequent generalized eigenvectors of (3.31), again in complete analogy to the input space algorithms.

But there is one problem: Even if the kernel matrix K has full rank (what is a reasonably practical assumption and always true e.g. for the RBF kernel), the matrix N does not, i.e. there exist a vector $\mathbf{ff}$, such that $\mathbf{ff}^\top N \mathbf{ff} = 0$ and the expression (3.31) is not well defined anymore. In fact, N has at most rank $M-2$. To see this, note that $D = I - \mathbf{v}_1\mathbf{v}_1^\top - \mathbf{v}_2\mathbf{v}_2^\top$ has rank $M-2$, where $\mathbf{v}_1$ and $\mathbf{v}_2$ span the null space of D. As the null space of N is at least as large as the one of D, N has at most rank $M-2$. In practice, one could resolve this problem by minimizing $1/J(\mathbf{ff})$ instead of maximizing $J(\mathbf{ff})$. Another possibility is to use the fact, that the optimal solution to (3.31) is in the direction of $N^{-1}(\bar{}_1 - \bar{}_2)$, and use the pseudo-inverse instead of the inverse (and thus getting the vector $\mathbf{ff}$ with the smallest ℓ_2–norm). This approach would also yield the correct solution for the case of a linear kernel, i.e. the solution of linear Fisher discriminants. However, this shows one problem with Fisher's discriminants: If the number of dimensions is large compared to the number of examples, the problem becomes ill-posed. Especially, in the case of kernel algorithms we work effectively in the space spanned by all M $\Phi(\mathbf{x})$ vectors which – as noted before – are in practice often linearly independent. Thus, we are estimating covariances in an M–dimensional space from M examples, which is ill-posed as well. Along the lines of e.g. Friedman (1989) we will treat this difficulty by using regularization in Section 3.4.6.

3.4.5 Other Rayleigh Coefficients

Algorithms using different matrices for S_I or S_N can easily be obtained along the same lines. For example, if we wanted to do kernel PCA, we would choose S_I as the covariance matrix (3.21) and also end up with:

$$\mathbf{w}^\top S_I \mathbf{w} = \mathbf{ff}^\top N_{PCA} \mathbf{ff},$$

but now N_{PCA} simplifies to $N_{PCA} = K(I - \mathbf{v}\mathbf{v}^\top)K^\top$ with $\mathbf{v}$ being the $1/\sqrt{M}$ vector. Since in the PCA case $S_N = I$ we trivially get

$$\begin{aligned}
\mathbf{w}^\top S_N \mathbf{w} &= \mathbf{w}^\top \mathbf{w} \\
&= \left(\sum_{x \in \mathcal{Z}} \alpha_x \Phi(\mathbf{x})^\top \right) \left(\sum_{z \in \mathcal{Z}} \alpha_z \Phi(\mathbf{z}) \right) \\
&= \sum_{x,z \in \mathcal{Z}} \alpha_x \alpha_z \left(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) \right) \\
&= \mathbf{ff}^\top K \mathbf{ff},
\end{aligned}$$

K being the kernel matrix.

If we do e.g. oriented kernel PCA or incorporate invariances into KFD using the tangent covariance matrix (3.23) things are slightly more complicated. Since this matrix is not only constructed from the training examples but also from the transformed examples we can not readily use (3.25). This leaves us with two options:

- One can show that the expansion for $\mathbf{w}$ is still correct but it would have to range over all training patterns *and* the transformed examples. However, this would result in matrices that are twice as large, i.e. $2M \times 2M$ for one transformation. Each additional transformation would increase the number of coefficients that have to be estimated by M.

- The second option is to just ignore this fact and still use the shorter expansion (3.25) over the training examples only. If the training set is large enough and the transformations are only small this should still give a reasonably good approximation. Then the size of the final problem stays $M \times M$, regardless of the number of transformations we incorporate.

Here we adopt the second strategy and get, using the expansion (3.25) for $\mathbf{w}$, that

$$\mathbf{w}^\top T \mathbf{w} = \mathbf{f}\mathbf{f}^\top N_T \mathbf{f}\mathbf{f},$$

where $N_T = KK^\top + \bar{K}\bar{K}^\top - K\bar{K}^\top - \bar{K}K^\top$, where $\bar{K}$ is the kernel matrix between the original examples and the transformed examples $\Phi(\mathcal{L}_t \mathbf{x})$.

3.4.6 Regularization

As we have already mentioned in Section 3.4.4 optimizing the Rayleigh coefficient for Fisher's discriminant in a feature space poses some problems. The matrix N is not strictly positive and numerical problems can cause the matrix N not even to be positive semi-definite. Furthermore, from Chapter 2 we know that for successful learning it is absolutely mandatory to control the size and complexity of the function class we choose our estimates from. This issue was not particularly problematic for linear discriminants since they already present a rather simple hypothesis class. Now, using the kernel-trick, we can represent an extremely rich class of possible non-linear solutions. For example, using the Gaussian RBF kernel (cf. (2.35)) we can always achieve a solution with zero within class variance (i.e. $\mathbf{f}\mathbf{f}^\top N \mathbf{f}\mathbf{f} = 0$). Such a solution will, except for pathological cases, be over–fitting.

To impose a certain regularity, the simplest possible solution is to add a multiple of the identity matrix to N, i.e. replace N by N_μ where

$$N_\mu := N + \mu I \quad (\mu \geq 0). \tag{3.33}$$

This can be viewed in different ways:

- If μ is sufficiently large this makes the problem feasible and numerically more stable as N_μ becomes positive definite.

- Increasing μ decreases the variance inherent to the estimate N; for $\mu \to \infty$ the estimate becomes less and less sensitive to the covariance structure. In fact, for $\mu = \infty$ the solution will lie in the direction of $\bar{}_2 - \bar{}_1$. The estimate of this "means" however, converges very fast and is very stable (cf. Bousquet and Elisseeff, 2002).

- For a suitable choice of μ, this can be seen as decreasing the bias in sample based estimation of eigenvalues (cf. Friedman, 1989). The point here is, that the empirical eigenvalue of a covariance matrix is not an unbiased estimator of the corresponding eigenvalue of the true covariance matrix, i.e. as we see more and more examples, the largest eigenvalue does *not* converge to the largest eigenvalue of the true covariance matrix. One can show that the largest eigenvalues are over–estimated and that the smallest eigenvalues are under–estimated (see also Chapter 4). However, the sum of all eigenvalues (i.e. the trace) does converge since the estimation of the covariance matrix itself (when done properly) is unbiased.

- It also imposes a regularization on $\|\mathbf{ff}\|^2$, favoring solutions with small expansion coefficients. This can be seen in analogy to e.g. weight decay as used with neural networks (Bishop, 1995; Ripley, 1996).

Another possible regularization strategy would be to add a multiple of the kernel matrix K to N, i.e. replace N with

$$N_\mu := N + \mu K \quad (\mu \geq 0). \tag{3.34}$$

This would, in analogy to SVM penalize $\|\mathbf{w}\|^2$, since $\|\mathbf{w}\|^2 = \mathbf{ff}^\top K \mathbf{ff}$. In practice it does not seem to make a difference which way of regularization one chooses as long as we do careful model selection. However, as we will see in the next section the $\|\mathbf{w}\|^2$ variant has some advantages.

3.4.7 KFD, Least Squares, and Quadratic Optimization

As we have already shown in Section 3.2.1 Fisher's discriminants are equivalent to a least squares regression to the labels (e.g. Duda and Hart, 1973). It is straight forward to show that the same holds true in the feature space $\mathcal{E}$. Thus, to solve the two class kernel Fisher discriminant (KFD) problem one can solve the following, quadratic optimization problem:

$$\min_{\mathrm{w},b,\xi} \quad \frac{1}{2}\|\xi\|^2 + C\,\mathrm{P}(\mathbf{w}), \tag{3.35}$$

subject to:

$$(\mathbf{w} \cdot \Phi(\mathbf{x})) + b = y - \xi_{\mathrm{x}}, \ \forall (\mathbf{x}, y) \in \mathcal{Z}. \tag{3.36}$$

Since we have already argued that some form of regularization will be necessary, we have added the additional term $C\,\mathrm{P}(\mathbf{w})$ to the objective function. Here P denotes some regularization operator and $C \in \mathbb{R}$, $C \geq 0$, is an user-defined constant that controls the amount of regularization imposed. For $C = 0$ and a linear kernel (i.e. $\mathrm{k}(\mathbf{x}, \mathbf{z}) = \mathbf{x} \cdot \mathbf{z}$) we recover Fisher's linear discriminant.

Lets us analyze this mathematical program a little closer to gain some insight on the nature of this problem. The first option is to replace $\mathbf{w}$ with the expansion (3.25). Let $\mathbf{1}$ be a vector of all ones of appropriate length and $\mathbf{y}$ the (± 1) vector of labels. Then we obtain the following problem:

$$\min_{\mathbf{ff},b,\,} \quad \frac{1}{2}\|\,\|^2 + C\,\mathrm{P}(\mathbf{ff}), \tag{3.37}$$

subject to:

$$K\mathbf{ff} + \mathbf{1}b = \mathbf{y} - \,. \tag{3.38}$$

Note that we changed the regularization operator to $\mathrm{P}(\mathbf{ff})$. This way, whatever we choose for P the problem (3.37) has no explicit reference to the mapping Φ anymore and can in principle be solved. However, only linear or quadratic forms of P will lead to linear or quadratic optimization problems.

The second option to deal with (3.35) would be to assume some quadratic form for $\mathrm{P}(\mathbf{w})$. If we choose $\mathrm{P}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ as in SVM we can form the dual optimization problem as follows. First we introduce Lagrange multipliers α_x, $\mathbf{x} \in \mathcal{Z}$, for each of the constraints (3.36). Since we are dealing with equality constraints these α_x are, in contrast to the inequality constraints (2.26) of SVM, unbounded variables. The Lagrangian then reads:

$$L(\mathbf{w}, b, \xi, \mathbf{ff}) = \frac{1}{2}\|\,\|^2 + \frac{C}{2}\|\mathbf{w}\|^2 - \sum_{(x,y)\in\mathcal{Z}} \alpha_x \left((\mathbf{w}\cdot\Phi(\mathbf{x})) + b - y + \xi_x\right). \tag{3.39}$$

Since for optimality we must be at a saddle point of L we take the derivatives with respect to the primal variables $\mathbf{w}$, b and ξ to obtain the following optimality conditions:

$$\frac{\partial L}{\partial \mathbf{w}} = C\mathbf{w} - \sum_{x\in\mathcal{Z}} \alpha_x \Phi(\mathbf{x}) \qquad \stackrel{!}{=} 0, \tag{3.40}$$

$$\frac{\partial L}{\partial b} = \sum_{x\in\mathcal{Z}} \alpha_x \qquad \stackrel{!}{=} 0, \tag{3.41}$$

$$\frac{\partial L}{\partial \,} = \, - \mathbf{ff} \qquad \stackrel{!}{=} 0. \tag{3.42}$$

It follows once more from (3.40) that $\mathbf{w}$ can be expressed as a linear combination of the (mapped) training examples, i.e.

$$\mathbf{w} = \frac{1}{C}\sum_{x\in\mathcal{Z}} \alpha_x \Phi(\mathbf{x}). \tag{3.43}$$

Using the optimality conditions (3.40)-(3.42) we can form the following dual optimization problem by replacing in (3.39) $\mathbf{w}$ by (3.43) and using $\, = \mathbf{ff}$:

$$\max_{\mathbf{ff}} \quad -\frac{1}{2}\|\mathbf{ff}\|^2 - \frac{1}{2C}\mathbf{ff}^\top K\mathbf{ff} + \mathbf{y}^\top\mathbf{ff}, \tag{3.44}$$

subject to:

$$\sum_{x\in\mathcal{Z}} \alpha_x = 0. \tag{3.45}$$

The question arises whether the solution obtained by the dual problem (3.44) is any different from the solution of the problem (3.37) when setting the regularization operator to $P(\mathbf{ff}) = \frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{ff}^\top K\mathbf{ff}$? The answer is: no. To see this, we also form the dual of (3.37): Forming again the Lagrangian with multipliers β_x, $\mathbf{x} \in \mathcal{Z}$, we get

$$L(\mathbf{ff}, b, \xi, \mathbf{fi}) = \frac{1}{2}\|\xi\|^2 + \frac{C}{2}\mathbf{ff}^\top K\mathbf{ff} - \mathbf{fi}^\top (K\mathbf{ff} + \mathbf{1}b - \mathbf{y} + \xi).$$

The conditions for optimality now read

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{ff}} &= CK\mathbf{ff} - K\mathbf{fi} && \stackrel{!}{=} 0, \\ \frac{\partial L}{\partial b} &= \mathbf{1}^\top \mathbf{fi} && \stackrel{!}{=} 0, \\ \frac{\partial L}{\partial \xi} &= \xi - \mathbf{fi} && \stackrel{!}{=} 0. \end{aligned}$$

Using these conditions, especially that $K\mathbf{ff} = \frac{1}{C}K\mathbf{fi}$, we get the following dual problem:

$$\max_{\mathbf{fi}} \quad -\frac{1}{2}\|\mathbf{fi}\|^2 - \frac{1}{2C}\mathbf{fi}^\top K\mathbf{fi} + \mathbf{y}^\top \mathbf{fi}, \tag{3.46}$$

subject to:

$$\mathbf{1}^\top \mathbf{fi} = 0, \tag{3.47}$$

which is clearly equivalent to (3.44). It remains the question whether the solutions for $\mathbf{w}$ will be identical. This is straight forward to see if K has full rank since then for the second dual problem $\mathbf{ff} = \frac{1}{C}\mathbf{fi}$ which coincides with the solution for $\mathbf{w}$ obtained in the first dual. Otherwise, if K does not have full rank, the solutions will at least agree on the space spanned by the training examples $\Phi(\mathbf{x})$ which is all we are interested in.

Finally, let us also consider the dual optimization problem of (3.37) if we choose, in analogy to (3.33), the regularizer $P = I$, i.e. penalize $\|\mathbf{ff}\|^2$. The Lagrangian becomes:

$$L(\mathbf{ff}, b, \xi, \mathbf{fi}) = \frac{1}{2}\|\xi\|^2 + \frac{C}{2}\|\mathbf{ff}\|^2 - \mathbf{fi}^\top (K\mathbf{ff} + \mathbf{1}b - \mathbf{y} + \xi),$$

and the KKT conditions are

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{ff}} &= C\mathbf{ff} - K\mathbf{fi} && \stackrel{!}{=} 0, \\ \frac{\partial L}{\partial b} &= \mathbf{1}^\top \mathbf{fi} && \stackrel{!}{=} 0, \\ \frac{\partial L}{\partial \xi} &= \xi - \mathbf{fi} && \stackrel{!}{=} 0. \end{aligned}$$

Using this time that $\mathbf{ff} = \frac{1}{C}K\mathbf{fi}$, we get the following dual problem:

$$\max_{\mathbf{fi}} \quad -\frac{1}{2}\|\mathbf{fi}\|^2 - \frac{1}{2C}\mathbf{fi}^\top K^\top K\mathbf{fi} + \mathbf{y}^\top \mathbf{fi}, \tag{3.48}$$

subject to:

$$\mathbf{1}^\top \mathbf{fi} = 0. \tag{3.49}$$

Casting the kernel Fisher discriminant problem into a quadratic program reveals some interesting properties. This formulation might be taken as another motivation for adding a regularizer to linear or kernel Fisher discriminants. If there are more examples than dimensions then the least squares approach induces a form of regularization by the fact that the system of equalities we are trying to solve is over-determined, i.e. there are more constraints than there are variables (in the linear case). Otherwise, if the number of examples is equal or even smaller than the dimension of the space one is working in (as in the kernel case), the system is under-determined, e.g. there are one or more solutions, which are a perfect fit to the labels, and as such almost certainly over–fitting.

From a classification point of view the quadratic program has an appealing interpretation. The constraints ensures, that the average class distance, projected onto the direction of discrimination, is two (for ± 1 labels), while the intra class variance is minimized, i.e. we maximize the *average* margin. Contrarily, the SVM approach (Boser et al., 1992) optimizes for a large *minimal* margin (see also Figure 5.7).

3.4.8 Sparse and Linear KFD

We will now use the quadratic optimization formulations of KFD presented in the last section to derive some modified kernel Fisher variants with interesting properties. First, since the expansion coefficients in KFD are free variables and since there is no term in the QP objectives which would enforce zero expansion coefficients the solutions **ff** will be non-sparse, i.e. all terms α_x will be nonzero. This is undesirable for two reasons: (i) for large M it will make the evaluation of $\mathbf{w} \cdot \Phi(\mathbf{x})$ slow and (ii) the optimization of e.g. (3.44) will be more difficult. Clever tricks like chunking (Saunders et al., 1998b; Osuna et al., 1997) or sequential minimal optimization (SMO) (Platt, 1999) will not be applicable or only at a much higher computational cost. Second, penalizing the squared deviation from the label via $\|\xi\|^2$ is often considered to be non-robust in the sense, that a single outlier can have an inappropriately strong influence on the solution (Huber, 1981).

The three KFD variants proposed in the following will circumvent the problems just discussed.

Robust KFD

To prevent a too big influence of a single training example a common rule of thumb suggests to penalize errors only linearly or to use the so called *robust loss*. The first option would amount to replace the squared error term $\|\xi\|^2$ in (3.37) by the ℓ_1-norm of ξ, i.e. $\|\xi\|_1$. In a mathematical programming formulation this reads:

$$\min_{w,b,\xi,\xi^-} \quad \mathbf{1}^\top(\xi + \xi^-) + \frac{C}{2}\|\mathbf{w}\|^2, \tag{3.50}$$

subject to:

$$(\mathbf{w} \cdot \Phi(\mathbf{x})) + b = y - (\xi - \xi^-)_x, \ \forall(\mathbf{x}, y) \in \mathcal{Z},$$

and

$$\xi, \xi^- \geq 0,$$

where we have used a standard trick from mathematical programming and replaced the variable ξ by its positive and negative part respectively, i.e. $\xi = \xi_+ - \xi_-$. This way the ℓ_1-norm of ξ can expressed as a linear function. For the dual optimization problem this means that instead of penalizing the squared norm of $\mathbf{ff}$ we will limit the size of $\mathbf{ff}$. Still, formulating the dual shows that $\mathbf{w}$ is given by the expansion (3.43) (and we could also have written $\frac{1}{2C}\mathbf{ff}^\top K\mathbf{ff}$ in the objective of (3.50)):

$$\max_{\mathbf{ff}} \quad -\frac{1}{2C}\mathbf{ff}^\top K\mathbf{ff} + \mathbf{y}^\top\mathbf{ff}, \tag{3.51}$$

subject to:

$$\begin{array}{rcl} \mathbf{1}^\top\mathbf{ff} & = & 0, \\ \mathbf{1} \leq \quad \mathbf{ff} & \leq & \mathbf{1}. \end{array}$$

Another way to achieve more robustness is to use what is known as Huber's robust loss function (cf. Section 2.1.4 and Table 2.1):

$$\ell(\xi_\mathsf{x}) = \begin{cases} \frac{1}{2\sigma}\xi_\mathsf{x}^2 & \text{if } |\xi_\mathsf{x}| \leq \sigma, \\ |\xi_\mathsf{x}| - \frac{\sigma}{2} & \text{otherwise,} \end{cases} \tag{3.52}$$

i.e. we penalize small errors by the squared norm but errors which are too large (larger than σ) are only penalized linearly. This can also be formulated in a mathematical program by splitting the slack variables ξ appropriately:

$$\min_{\mathsf{w},b,\xi,\xi_+,\xi_-} \quad \frac{1}{2\sigma}\|\xi\|^2 + \mathbf{1}^\top(\xi_+ + \xi_- - \mathbf{1}\frac{\sigma}{2}) + \frac{C}{2}\|\mathbf{w}\|^2,$$

subject to:

$$\begin{array}{rcll} (\mathbf{w}\cdot\Phi(\mathbf{x})) + b & = & y - \xi_\mathsf{x} - (\xi_+ - \xi_-)_\mathsf{x} & \forall(\mathbf{x},y) \in \mathcal{Z}, \\ -\mathbf{1}\sigma \leq & \xi & \leq \mathbf{1}\sigma, & \\ & \xi_+, \xi_- & \geq 0. & \end{array}$$

Since we will not consider this problem further we skip deriving its corresponding dual.

Sparse KFD

Next we will investigate how to modify (3.37) or (3.44) to yield sparse expansion coefficients. The cleanest way to achieve sparse expansions $\mathbf{ff}$ would be to add a constraint on the ℓ_0-norm of $\mathbf{ff}$ counting the number of non-zero coefficients p, i.e. $\|\mathbf{ff}\|_0 \leq p$ for some a priori chosen maximal number of non-zero coefficients p. However, such an approach leads to a non-convex optimization problem that can only be solved exactly by searching the space of all possible solutions with p non-zero α_x. This is clearly prohibitive for more than a few training examples since there are $\binom{M}{p}$ possible solutions.

A more practical approach is the following: In mathematical programming sparsity can often be achieved by constraining or penalizing the ℓ_1-norm of the variables. In our case that would amount to put a $\|\mathbf{ff}\|_1$ in the objective. Roughly speaking, a reason for the induced sparseness is the fact that vectors far from the coordinate axes are "larger" with respect to the ℓ_1–norm than with respect to

ℓ_p-norms with $p > 1$. For example, consider the vectors $(1, 0)$ and $(1/\sqrt{2}, 1/\sqrt{2})$. For the two norm, $\|(1, 0)\|_2 = \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = 1$, but for the ℓ_1–norm, $1 = \|(1, 0)\|_1 < \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = \sqrt{2}$. Note that using the ℓ_1–norm as regularizer the optimal solution is always a vertex solution (or can be expressed as such) and tends to be very sparse. The resulting optimization problem is still convex and can be solved using standard quadratic or linear optimization routines.

The question is where to put the $\|\mathbf{ff}\|_1$? While it seems tempting to change the ℓ_2-norm in the standard KFD dual problem (3.44) to be a ℓ_1-norm, this does not work out since we could only put $\|\mathbf{ff}\|^2$ into the objective since we knew from the KKT condition (3.42) that $\mathbf{ff} = \xi$ at the optimal solution. Forming the dual in this case would reveal that now the optimal solution is $\mathbf{ff} \equiv 0$.

Since enforcing sparsity can also be seen as some form of regularization we will proceed differently and put the ℓ_1-norm as regularizer in the primal problem (3.37):

$$\min_{\mathbf{ff}_+, \mathbf{ff}_-, b, \xi} \quad \frac{1}{2}\|\xi\|^2 + C\mathbf{1}^\top(\mathbf{ff}_+ + \mathbf{ff}_-), \tag{3.53}$$

subject to:

$$K(\mathbf{ff}_+ - \mathbf{ff}_-) + \mathbf{1}b = \mathbf{y} - \xi,$$

and

$$\mathbf{ff}_+, \mathbf{ff}_- \geq 0.$$

We call this particular variant sparse KFD (SKFD). It is interesting to see what happens to the dual of the KFD problem when using the ℓ_1-norm regularizer instead of $\|\mathbf{w}\|^2$:

$$\max_{\mathbf{fi}} \quad -\frac{1}{2}\|\mathbf{fi}\|^2 + \mathbf{y}^\top\mathbf{fi}, \tag{3.54}$$

subject to:

$$\begin{array}{rcccl} \mathbf{1}^\top\mathbf{fi} & = & 0 & & , \\ -C\mathbf{1} & \leq & K\mathbf{fi} & \leq & C\mathbf{1}. \end{array}$$

Note that also for this problem the (primal) slack variables ξ will be equal to the dual variables (i.e. the Lagrange multipliers) $\mathbf{fi}$ (cf. (3.42)). But here the dual variables are not the ones anymore that we need to express the solution. However, most quadratic and linear optimization packages also return the Lagrange multipliers at the optimal solution. It turns out that the (dual) Lagrange multipliers for the constraint $|K\mathbf{fi}| \leq C\mathbf{1}$ are exactly the $\mathbf{ff}$'s we are looking for.

Linear Sparse KFD

The final algorithm we propose here is the straight forward combination of robust KFD and sparse KFD, i.e. we use the linear loss for ξ and the ℓ_1-norm regularizer. We call this setting linear sparse KFD (LSKFD). The optimization problem becomes:

$$\min_{\mathbf{ff}_+, \mathbf{ff}_-, b, \xi_+, \xi_-} \quad \mathbf{1}^\top(\xi_+ + \xi_-) + C\mathbf{1}^\top(\mathbf{ff}_+ + \mathbf{ff}_-), \tag{3.55}$$

subject to:

$$K(\mathbf{ff}_+ - \mathbf{ff}_-) + \mathbf{1}b = \mathbf{y} - (\xi_+ - \xi_-),$$

and

$$\mathbf{ff}_+, \mathbf{ff}_-, \text{,}_+, \text{,}_- \geq 0.$$

The dual of this problem is

$$\max_{\mathbf{fi}} \quad \mathbf{y}^\top \mathbf{fi}, \tag{3.56}$$

subject to:

$$\begin{array}{rcccl} & & \mathbf{1}^\top \mathbf{fi} & = & 0, \\ -C\mathbf{1} & \leq & K\mathbf{fi} & \leq & C\mathbf{1}, \\ -\mathbf{1} & \leq & \mathbf{fi} & \leq & \mathbf{1}. \end{array}$$

We see how the dual constraints of linear KFD and sparse KFD merge together here to achieve both, a robust linear loss and sparse solutions. However, also here it is not possible to obtain the expansion coefficients $\mathbf{ff}$ directly from the dual variables $\mathbf{fi}$. Instead we have $\mathbf{ff} = K^{-1}\mathbf{fi}$. But as for sparse KFD, the primal variables $\mathbf{ff}$ can be recovered from the Lagrange multipliers of the $|K\mathbf{fi}| \leq C\mathbf{1}$ constraints.

3.5 Algorithms

Having successfully formulated a Rayleigh coefficient in feature space there occurs one major problem: While we could avoid working explicitly in the extremely high or infinite dimensional space $\mathcal{E}$ we are now facing a problem in M variables, a number which in many practical applications would not allow to store or manipulate $M \times M$ matrices on a computer anymore. Furthermore, solving e.g. an eigenproblem of size $M \times M$ is rather time consuming ($\mathcal{O}(M^3)$). In the following we will first propose several ways to deal with this general problem when optimizing the Rayleigh coefficient in feature space. Second, we will derive some numerical methods explicitly adopted to the special structure of the kernel Fisher discriminant problem.

3.5.1 Training on a subset

To maximize (3.31) in the case of KFD or the equivalent for other choices of matrices (cf. Section 3.4.5), we need to solve an $M \times M$ eigenproblem, which might be intractable for large M. As the solutions are not sparse, one can not directly use efficient algorithms like chunking for support vector machines (cf. Schölkopf et al., 1999a). One possible solution, which is applicable to any choice of matrices S_I and S_N, is to restrict the feature extractor $\mathbf{w}$ to lie in a subspace, i.e. instead of expanding $\mathbf{w}$ by (3.25) we write

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \Phi(\mathbf{z}_i), \tag{3.57}$$

with $m \ll M$ (analogous to the reduced set method of SVM (cf. Schölkopf et al., 1999b)). The examples $\mathbf{z}_i$, $i = 1, \ldots, m$, could either be a subset of the training sample $\mathcal{Z}$ or e.g. be estimated by some clustering algorithm. The derivation of

(3.31) does not change, only K is now $m \times M$ and we end up with $m \times m$ matrices for N and M. Another advantage of this approach is that it makes the matrix N non-singular by increasing the rank of N (relative to its size). However, since it is hard to determine the most important points $\mathbf{z}_i$ to describe a good solution such an approach can not be seen as much more than a computationally cheap heuristic. However, if m is not too small and the data are reasonably well behaved (i.e. not extremely noisy), experience shows that using this simple and straight forward "trick", we do not loose much in terms of generalization ability.

3.5.2 A Sparse, Greedy Approach

Another way to find a solution to (3.31) is given by the following idea: Instead of trying to find a full solution $\mathbf{w}$ described by M non-zero α's, one can try to approximate $\mathbf{w}$ by a shorter expansion (i.e. by using a vector of α's with many zero entries). But instead of choosing the non-zero coefficients a priori as proposed in the last paragraph one iteratively selects new examples to add to the expansion in a greedy fashion, e.g. such that the actual objective function is minimized among all possible choices (see also Golub and van Loan (1996)). But, contrary to the approach which selects the expansion examples a priori, this algorithm will only be feasible, if there is an *efficient* way of evaluating the optimality criterion in each step, an assumption that is not true for all choices of matrices S_I and S_N and all selection criteria. But for e.g. KFD this is possible and has been described in Mika et al. (2001b). Such an approach is straight forward to implement and much faster than solving a quadratic program or eigenproblem, provided that the number of non–zero α's necessary to get a good approximation to the full solution is small. However, this approach will only be an approximation to the optimal solution, which has the same number of non–zero coefficients. There is some theoretical evidence that such an approach gets close to the optimal sparse solution (Natarajan, 1995). But it is also possible to construct situations in which such a heuristic will fail (Miller, 1990). In general, finding the optimal m out of M coefficients to adjust is a NP–hard problem. The approach presented here is very similar in spirit to what has been discussed in the context of SVM in Smola and Schölkopf (2000) and for Gaussian Process Regression in Smola and Bartlett (2001).

To proceed, let us rewrite (3.37). Solving the equality constraint for γ and substituting it into the objective we get

$$\min_{\mathbf{f},b} \frac{1}{2} \begin{bmatrix} \mathbf{f} \\ b \end{bmatrix}^\top \begin{bmatrix} K^\top K + C\,\mathrm{P} & K^\top \mathbf{1} \\ \mathbf{1}^\top K & M \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ b \end{bmatrix} - \begin{bmatrix} K^\top \mathbf{y} \\ M_2 - M_1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{f} \\ b \end{bmatrix} + \frac{M}{2}, \tag{3.58}$$

where the regularization is either done via $P = I$ or $P = K$. Since now we do not have any constraints left, we can compute the optimal solution by equating the derivative to zero and get

$$\begin{bmatrix} \mathbf{f}^* \\ b^* \end{bmatrix} = \begin{bmatrix} K^\top K + C\,\mathrm{P} & K^\top \mathbf{1} \\ \mathbf{1}^\top K & M \end{bmatrix}^{-1} \begin{bmatrix} K^\top \mathbf{y} \\ M_2 - M_1 \end{bmatrix}. \tag{3.59}$$

Of course, this problem is not easier to solve than the original one nor does it yield a sparse solution: there is an $(M+1) \times (M+1)$ matrix involved and for large data sets its inversion is not feasible, neither in terms of time nor memory cost.

Now, the idea is to use the following, greedy approximation scheme (cf. Smola and Bartlett, 2001): Instead of trying to find a full set of M α_i's for the solution (3.25), we approximate the optimal solution by a shorter expansion containing only $m \ll M$ terms.

Starting with an empty expansion $m = 0$, one selects in each iteration a new sample $\mathbf{x}_i$ (or an index i) and resolves the problem for the expansion (3.25) containing this new index and all previously picked indices; we stop as soon as a suitable criterion is satisfied. This approach would still be infeasible in terms of computational cost if we had to solve the system (3.59) anew in each iteration. But with the derivation made before it is possible to find a close approximation to the optimal solution in each iteration at a cost of $\mathcal{O}(\kappa M m^2)$ where κ is a user defined value (see below).

Writing down the quadratic program (3.37) for KFD when the expansion for the solution is restricted to an m element subset $\mathcal{I} \subset [M]$

$$\mathbf{w}_{\mathcal{I}} = \sum_{i \in \mathcal{I}} \alpha_i \Phi(\mathbf{x}_i) \tag{3.60}$$

of the training patterns amounts to replacing the $M \times M$ matrix K by the $M \times m$ matrix K_m, where $(K_m)_{ij} = \mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)$, $i = 1, \ldots, M$ and $j \in \mathcal{I}$. Analogously, we can derive the formulation (3.58) and (3.59) using the matrix K_m. The problem is of order $m \times m$ now.

Rank One Update Of Inverse

Lets introduce the shorthand H for the matrix involved in (3.59) using the expansion restricted to m functions, i.e.

$$H = \begin{bmatrix} K_m^\top K_m + C\,\mathrm{P} & K_m^\top \mathbf{1} \\ \mathbf{1}^\top K_m & M \end{bmatrix}. \tag{3.61}$$

Assume we already know the optimal solution (and inverse of H) using m kernel-functions. Then the inverse matrix H^{-1} for $m+1$ samples can be obtained by a rank one update of the previous H^{-1} using only m basis functions: The following Lemma (e.g. Golub and van Loan, 1996) tells us how to obtain the new H^{-1}.

Lemma 3.1 (Sherman–Woodbury–Formula). *The inverse of a symmetric, positive matrix can be computed as:*

$$\begin{bmatrix} H & B \\ B^\top & C \end{bmatrix}^{-1} = \begin{bmatrix} H^{-1} + (H^{-1}B)\gamma(H^{-1}B)^\top & -\gamma(H^{-1}B) \\ -\gamma(H^{-1}B)^\top & \gamma \end{bmatrix},$$

where $\gamma = (C - B^\top H^{-1} B)^{-1}$.

Note that for our case B is a vector and C a scalar. This is an operation of cost $\mathcal{O}(m^2)$ as we already know the inverse of the smaller system.

Selection Rules

The last major problem is to pick an index i in each iteration. Choosing one index at a time can be considered as a coordinate wise descent method with the difference

that we update all coordinates which were already chosen in each iteration. As we are dealing with a convex, quadratic optimization problem any selection rule will finally lead to the (optimal) solution of the full problem. And adding only kernel functions to the expansion, it is easy to see that this convergence is monotonic with respect to the primal objective. The interesting question is how one has to choose the sequence of indices such that this greedy approach will get close to the full solution as soon as possible. Here we propose to choose the i for which we get the biggest decrease in the primal-objective (or equivalently as they are identical for the optimal coefficients $\mathbf{ff}$, the dual objective (3.46) or (3.48), respectively). This corresponds to a steepest descent method in a restricted space of the gradients.

A probabilistic speed-up

Testing the selection rule for all $M - m$ indices which are unused so far is again too expensive. This would require to compute all unused columns of the kernel matrix and to update the inverse matrix $M - m$ times. One possible solution lies in a second approximation. Instead of choosing the *best* possible index it is usually sufficient to find an index for which with high probability we achieve something close to the optimum. It turns out (cf. Smola and Schölkopf, 2000) that it can be enough to consider 59 randomly chosen indices from the remaining ones:

Lemma 3.2 (Maximum of Random Variables). *Denote by $\rho_1, \ldots, \rho_m$ identically distributed independent random variables with a common cumulative distribution function F. Then the cumulative distribution function of $\rho = \max_{i \in [m]} \rho_i$ is $(F)^m$.*

This means that e.g. for the uniform distribution on $[0, 1]$ $\max_{i \in [m]} \rho_i$ is distributed according to $(\rho)^m$. Thus, to obtain an estimate that is with probability 0.95 among the best 0.05 of all estimates, a random sample of size $\kappa := (\ln 0.05 / \ln 0.95) = 59$ is enough.

Termination

Still open is the question when to stop. If one wanted to compute the full solution this approach would not be very efficient as it would take $\mathcal{O}(\kappa M^3)$ which is worse than the original problem. A principled stopping rule would be to measure the distance of $\mathbf{w}_{\mathcal{I}}$ to the solution of the full problem and to stop when this falls below a certain threshold. Unfortunately the full solution is, for obvious reasons, not available. Instead one could try to bound the difference of the objective (3.37) for the current solution to the optimal value obtained for the full problem as done in Smola and Bartlett (2001). But in our case an efficient way to bound this difference is, again, not available. Instead we have decided for a very simple heuristic which turned out to work well in the experiments: stop when the average improvement in the dual objective (3.46) over the last p iterations is less than some threshold θ. The longer the averaging process, the more confident we are that the current solution is not at a plateau. The smaller the threshold, the closer we are to the original solution (indeed, setting the threshold to zero forces the algorithm to take all training samples into account). The complete algorithm for a sparse greedy solution to the KFD problem is sketched in Figure 3.3. It is easy

arguments: Sample $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}, \mathbf{y} = \{y_1, \ldots, y_M\}$
Maximum number of coefficients
or parameters of other stopping criterion: OPTS
Regularization constant C, κ and kernel k
returns: Set of indices I and corresponding **ff**'s.
Threshold b.
function SGKFD($X, \mathbf{y}, C, \kappa$, k, OPTS)

```
m ← 0
I ← ∅
while termination criterion not satisfied do
  S ← (κ elements from [M]\I)
  obj_max ← ∞
  for i ∈ S do
    Compute column i of kernel matrix
    Update inverse adding the i-th kernel
    Compute optimal ff and b
    Compute new objective
    if objective < obj_max do
      i_opt ← i
      obj_max ← objective
    endif
  endfor
  Update inverse H and solution a with kernel i_opt
  I ← I∪{i_opt}
  Check termination criterion
endwhile
```

Figure 3.3: The Sparse Greedy Kernel Fisher Algorithm (for the selection criterion which minimizes the objective).

to implement using a linear algebra package like BLAS and has the potential to be easily parallelized (the matrix update) and distributed. Furthermore, in multi-class problems (if one is using an one against the rest scheme) it is possible to consider all two-class problems simultaneously. Testing the same subset of indices for each classifier would result in a reasonable speedup, as the computation of the columns of the kernel matrix for each picked index is among the most expensive parts of this algorithm and must now only be done once.

3.5.3 Coordinate Descent & SMO

In this section we propose an optimization strategy for the (dual) quadratic optimization problem (3.44), i.e. KFD with a $\|\mathbf{w}\|^2$ regularizer, that is especially appealing. It can be seen as a coordinate wise descent method and is very similar in spirit to the Sequential Minimal Optimization (SMO) proposed by Platt (1999) for Support Vector Machines. A similar technique has been proposed for least square support vector machines by Keerthi and Shevade (2002). Its most impressive feature is that it does not require storage of the $M \times M$ matrix K at any time.

It will, however, be computationally more demanding than a SMO algorithm for SVM as the solution is not sparse. In SMO for SVM, we, in principle, only need to iteratively optimize those α_i that will be support vectors, i.e. $\alpha_i \neq 0$. Since we have here $\alpha_i \neq 0$ for all $i = 1, \ldots, M$ there is more work to be done.

The idea is to iteratively minimize the objective for the smallest number of coefficients possible, keeping the other coefficients fixed. The hope is, that each step will be computationally cheap. Assume that we are given an feasible, but not yet optimal, intermediate solution $\mathbf{f}\mathbf{f}^t$. Since the QP (3.44) has the constraint that the sum of all coefficients should be zero, i.e.

$$\mathbf{1}^\top \mathbf{f}\mathbf{f} = 0,$$

the minimal number of coefficients one has to change in order to keep the constraint satisfied is two (increasing one by the same amount the other is decreased). Assume that we have decided to optimize the coefficients α_i and α_j, $i \neq j$. Then the new solution $\mathbf{f}\mathbf{f}^{t+1}$ in the next iteration can be written as

$$\alpha_i^{t+1} = \alpha_i^t + r, \text{ and } \alpha_j^{t+1} = \alpha_j^t - r,$$

for some $r \in \mathbb{R}$. Substituting this expression into the dual objective (3.44) and differentiating with respect to r yields that the optimal step is given by

$$r = -\frac{\frac{1}{C}(K\mathbf{f}\mathbf{f}^t)_i + \alpha_i^t - y_i - \frac{1}{C}(K\mathbf{f}\mathbf{f}^t)_j - \alpha_j^t + y_j}{\frac{1}{C}K_{ii} + \frac{1}{C}K_{jj} - \frac{2}{C}K_{ij} + 2}.$$

It can be seen from the last equation that in order to compute the step r we only need access to the old $K\mathbf{f}\mathbf{f}^t$ product and the new kernel elements K_{ii}, K_{jj} and K_{ij}. This implies, that in an iterative optimization scheme we only need to track $K\mathbf{f}\mathbf{f}$. Since $K\mathbf{f}\mathbf{f}^{t+1} = K\mathbf{f}\mathbf{f}^t + rK_{\bullet i} - rK_{\bullet j}$ this can be done by computing the two columns of the kernel matrix corresponding to the two indices chosen in each step.

Choosing Indices

The question is however which indices to choose at each iteration. Since the solution will not be sparse anyway we could in principle choose all $(M-1)M$ combinations in turn. But then we might pick indices for which the increase in the objective is very small, i.e. convergence would be very slow. Since we have to increase one variable and to decrease the other we propose to choose the pair of indices with the largest and smallest gradient. The gradient can also be computed without explicitly accessing the kernel matrix K. Only using the product $K\mathbf{f}\mathbf{f}^t$ it is given by

$$\nabla_{\mathbf{f}\mathbf{f}} = \frac{1}{C}K\mathbf{f}\mathbf{f}^t + \mathbf{f}\mathbf{f}^t - \mathbf{y}.$$

and hence finding these indices is a simple search through all elements in $\nabla_{\mathbf{f}\mathbf{f}}$.

In order to see that such an approach converges it is important to notice that the gradient at the optimal solution will be a multiple of the unit vector. The proof is trivial and follows from the fact that at an optimal point $(\mathbf{f}\mathbf{f}, b)$ the primal

constraints can be written in terms of the gradient of the dual objective as follows:

$$\begin{aligned} \frac{1}{C}K\mathbf{ff} + \mathbf{1}b = \mathbf{y} - \, , \quad & \overset{\text{at optimum}}{\Rightarrow} \quad \frac{1}{C}K\mathbf{ff} + \mathbf{1}b = \mathbf{y} - \mathbf{ff} \\ & \Leftrightarrow \quad \frac{1}{C}K\mathbf{ff} + \mathbf{ff} - y = -\mathbf{1}b \\ & \Leftrightarrow \quad \nabla_{\mathsf{ff}} = -\mathbf{1}b. \end{aligned}$$

Vice versa one can show that we can not yet be at an optimal point if this equation does not hold true. This also implies that the dual objective will increase in each step.

Monitoring and Stopping

Since we are doing an iterative optimization it is important to know when the current solution $\mathbf{ff}^t$ is sufficiently close to the true solution. This can be done by monitoring the duality gap and the primal infeasibility. From the KKT conditions of this problem (cf. (3.42)) we know, that at the optimal solution $\, = \mathbf{ff}$. All we need to find in order to compute the primal infeasibility and objective is the threshold b. Keeping all other variables in the primal fixed but b, it is straight forward to show that the currently optimal value is given by:

$$b^t = -\frac{\mathbf{1}^\top \left(\frac{1}{C}K\mathbf{ff}^t + \mathbf{ff}^t - \mathbf{y}\right)}{M}.$$

All this as well as evaluating the primal objective can also be done just using $K\mathbf{ff}^t$ without the need to evaluate the complete kernel matrix.

From the theory of mathematical programming (cf. Appendix A) we know that a necessary and sufficient condition for optimality of the current solution $\mathbf{ff}^t$ is given by primal and dual feasibility and through the duality gap. Since by construction we are always dual feasible we are left with primal infeasibility and the duality gap, i.e. the difference between primal and dual objective. Primal infeasibility is measured relatively to the right hand side of the equality, i.e. in our case by

$$\frac{\|\frac{1}{C}K\mathbf{ff}^t + \mathbf{ff}^t - \mathbf{y}\|}{\|\mathbf{y}\|}$$

From the differences of the objectives we can compute the number of significant figures of the current objective value as (Vanderbei, 1997)

$$\max\left(-\log_{10}\left(\frac{|o_{\text{primal}} - o_{\text{dual}}|}{|o_{\text{primal}}| + 1}\right), 0\right). \tag{3.62}$$

As soon as this number is bigger than some predefined threshold and the primal infeasibility is small enough we know that the current solution is close to the true solution and stop. Since we keep track of the current $K\mathbf{ff}^t$ the cost for the monitoring process is only $\mathcal{O}(M)$. In Chapter 5 we will show that these thresholds can be chosen relatively small (e.g. only one significant figure and a primal infeasibility smaller than 0.1) to get good approximate solutions.

Greedy SMO and Other Extensions

Of course we are free to combine the greedy approximation scheme with this sequential optimization scheme to get both, sparse solutions (greedy approximation) and an algorithm with only linear memory needs (the SMO-type technique). We could use the greedy scheme to iteratively select new indices to add to the solution. But instead of maintaining the inverse of the ever growing matrix H (cf. (3.61)), we could use the SMO scheme to resolve our problem. The old solution could be used as a starting point and as long as the number of samples necessary to express a sufficiently good solution is small the iteration should converge quickly.

Another extension in the SMO scheme would be to make use of the past columns of the kernel matrix that have been computed already. First, one could store them in a cache to speed up the computation. However, in practice it seems that all indices are chosen with approximately equal probability, rendering a cache not very efficient. What one could do, however, is to use the cached columns of the kernel matrix to also update all α_i corresponding to them. This would be very similar to chunking techniques used in SVM and could be done by solving a small QP in each iteration whose size depends on the number of columns in the cache. In each iteration, we remove those columns from the cache with the smallest absolute difference in their gradients and add those two columns that we had chosen before anyway.

3.5.4 Solving a Linear System

The last method to solve the KFD problem with the $\|\mathbf{w}\|^2$ regularizer, i.e. (3.35), is through a linear system of equations. From the derivation of the dual (3.44) we know that the slacks ξ in the primal must be equal to the expansion coefficients $\mathbf{ff}$ at the optimal solution. We already used that thus the optimal solution must satisfy the primal equality constraint with ξ replaced by $\mathbf{ff}$, i.e.

$$\left(\frac{1}{C}K + I\right)\mathbf{ff} + \mathbf{1}b = \mathbf{y}.$$

This system is still under determined. But we also know, again from the dual, that the sum of all α_i must be zero, i.e. $\mathbf{1}^\top\mathbf{ff} = 0$. Combining the last two equalities we get that at the optimal solution

$$\begin{bmatrix} \frac{1}{C}K + I & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix} \begin{bmatrix} \mathbf{ff} \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}. \tag{3.63}$$

This is a system of $M+1$ equalities in $M+1$ variables. Since the kernel matrix K is at least positive semi-definite, for sufficiently large C this system will have full rank and hence possess a unique solution. Also here it would be possible to use the greedy approximation scheme discussed in Section 3.5.2 to approximate the inverse of the system.

By a similar argumentation one can show that we can find the optimal solution $\mathbf{fi}$ for the KFD problem with the $\|\mathbf{ff}\|^2$ regularizer by solving for

$$\begin{bmatrix} \frac{1}{C}K^\top K + I & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix} \begin{bmatrix} \mathbf{fi} \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}, \tag{3.64}$$

where, following the KKT conditions of (3.37) and (3.48) the solution **ff** is given by

$$\mathbf{ff} = \frac{1}{C} K \mathbf{fi}.$$

3.5.5 Interior Point and Incomplete Cholesky

The last possibility to solve any of the KFD problems we will discuss here is to resort to standard mathematical programming techniques. For solving linear problems the most well known technique is the so called simplex method (Dantzig, 1962). Whilst it is provably not polynomial in the number of variables it usually converges quite well. Alternative techniques for linear optimization like the ellipsoid method (Khachiyan, 1979) possess a polynomial convergence rate but they are impossible to use in practice. Here we want to concentrate on another technique called interior point optimization (Karmarkar, 1984; Mehrotra, 1992; Doljansky and Teboulle, 1998), suitable for linear and quadratic optimization.[6] The name *interior point* stems from the fact that these techniques try to find a point that satisfies both, the primal and dual constraints and the KKT complimentary conditions, i.e. is in the interior of the feasible set.

A more formal treatment of how interior point methods work and how to derive an interior point code in the spirit of Vanderbei (1994) can be found in the Appendix A. There we also outline how an interior point code can be adopted to the special structure of the KFD problems.

It turns out, that deriving a specialized interior point code for KFD has some advantages. The dominating cost in using an interior point optimizer is the solution of a reduced KKT system twice in each iteration. The special adoption carried out in the appendix allows to substantially reduce the size of this system using the special problem structure. However, this reduced system will always, in one or the other way, contain a copy of the dense matrix K. There are several ways in which one could continue the work presented here to cope with this situation:

- It has been suggested by Achlioptas et al. (2002) that the kernel matrix K can be approximated by a sparse matrix. This would greatly reduce the cost of solving the reduced KKT; especially, it would be helpful when solving the system by an iterative method.
- Low rank approximations of K might also be helpful since then the reduced KKT might become substantially smaller.
- Finally, the incomplete Cholesky factorization proposed e.g. in Bach and Jordan (2002) or the Nyström approximation (Smola and Schölkopf, 2000; Williams and Seeger, 2001) could be used to making solving the reduced KKT easier.

3.5.6 Linear Optimization

The last algorithm we want to mention is a very elegant way to solve the linear sparse KFD problem (3.55) (LSKFD). It can be understood in analogy to linear

[6] Indeed, interior points methods can also applied to non–convex, non–linear problems (Vanderbei and Shanno, 1997); here we restrict ourselves to what we actually need.

programming machines as they emerged in the context of SVM (cf. Graepel et al., 1999). Such a linear program will be very sparse in $\mathbf{ff}$ and can be optimized using a technique called column generation (e.g. Bennett et al., 2000; Rätsch, 2001; Rätsch et al., 2002, and references therein).

The idea behind column generation can be summarized as follows: Each primal variable (i.e. each α_i) yields a constraint in the dual problem. From the Karush-Kuhn-Tucker complimentary condition we know that only those variables in the primal will be non-zero for which the corresponding dual constraint is active, i.e. the inequality is fulfilled with equality. For the LSKFD problem this translates into the following: At the optimal solution, for each i we have $\alpha_i \neq 0$ if and only if $K_{i\bullet}\mathbf{fi} = C$ or $K_{i\bullet}\mathbf{fi} = -C$. Assume for a moment we would know before we start the optimization which α_i are non-zero. Then the problem would be much easier to solve since we could eliminate many primal variables and most of the dual constraints. Especially important is that this allows us to get rid of large parts of the kernel matrix K. Now, column generation proceeds as follows (see

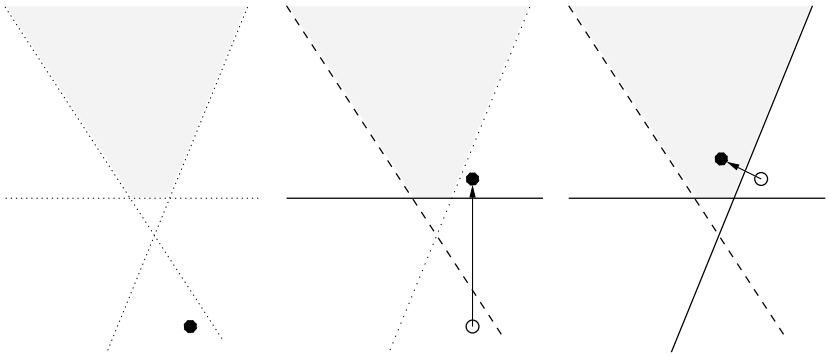

Figure 3.4: Illustration of column generation (without objective). Shown are three linear constraints and the feasible region (shaded). The current solution is depicted by the black dot. In the beginning all three constraints are violated. The next solution is found by including the most violated constraint. This also satisfies another constraint although it was not explicitly evaluated. In the final step the last violated constraint is added to the problem and a feasible solution is found.

also Figure 3.4): We start with a dual problem containing no constraints of the form $-C\mathbf{1} \leq K\mathbf{fi} \leq C\mathbf{1}$. Then in each iteration we try to find a constraint in the dual (a row of the kernel matrix) which is violated. The name column generation stems from the fact that this way we add a column, i.e. a variable, in the primal. We then add this constraint to the problem and resolve. Luckily, solving a linear program that only differs from another program we know the optimal solution for in a few (here two) constraints is faster than solving the complete problem. Even nicer, removing an inactive constraint from our dual problem can be done at no cost at all, i.e. in each iteration we might as well remove constraints which have been inactive for a number of past iterations. They correspond to primal variables α_i that would be zero for the current (intermediate) solution. If there are no more violated constraints left to add to the dual we are at the optimal (dual) solution. As mentioned before, since most linear optimizers also compute the Lagrange multipliers for the problem they solve, we can directly recover the primal solutions $\mathbf{ff}_+$ and $\mathbf{ff}_-$ from the multipliers associated with the constraints

$K\boldsymbol{\alpha} \leq C\mathbf{1}$ and $-C\mathbf{1} \leq K\boldsymbol{\alpha}$, respectively.

The major difficulty with such an approach is, however, to find those columns of K which violate the dual constraints. Since we want to avoid storing or computing the complete matrix K in each iteration we resort to the random sampling technique proposed already in the context of the sparse, greedy approximation scheme (cf. Section 3.5.2). This way it will be very unlikely that we miss a violated constraint and this is achieved at a reasonable cost. If we do not find violated constraints anymore way we might either terminate or run the standard checks for the duality-gap and primal feasibility. If they also indicate that we are at an optimal solution it becomes even more unlikely that there are any (dual) violated constraints left. As a last resort we might scan before termination through the complete matrix K: If we find violated constraints we add them and continue, otherwise we know that we definitely have found the optimum.

The advantage of such an approach is that for a reasonable degree of sparsity we never need to store the complete kernel matrix.

3.6 Comparison to other Techniques

Having formulated KFD and its variants as a mathematical program is it striking how similar it looks to e.g. an SVM. But it is also possible to show close connections to other techniques, either directly or by interpreting the underlying mathematical programs.

3.6.1 Least Squares Support Vector Machines

As we have already pointed out, the technique proposed in Suykens and Vanderwalle (1999) called least squares support vector machines (LQSVM) is exactly the same as Kernel Fisher Discriminants. This can be shown via some complex mathematics as in Gestel et al. (2001) or by using the QP formulation (3.35). However, we believe that the motivation for KFD as a non–linear version of Fisher's discriminant is more appealing than as that of least square SVMs.

3.6.2 Relevance Vector Machines

The following connection is particularly interesting. We have already shown that Fisher's discriminants are the Bayes optimal classifier for two Gaussian distributions with equal covariance and hence KFD is Bayes optimal for two Gaussians in feature space. We have also shown that Fisher's discriminant (and KFD) can be understood as a regression to the labels.

Here we will make yet another ansatz leading to a similar conclusion and showing the strong connection between the Relevance Vector Machine for regression (RVM) (Tipping, 2000) and KFD. The RVM is a Bayesian technique for regression and also classification. Its derivation is as follows: Consider for a moment we were solving a regression problem in $\mathbf{x}$ and y. Then it is standard to assume that the likelihood $p(y|\mathbf{x})$ is given by a Gaussian distribution. Assume furthermore, that for a given $\mathbf{x}$ the mean of this Gaussian is given by $\mathbf{w}^\top\mathbf{x} + b$ where $\mathbf{w}$ is defined as

in (3.25), i.e.

$$\mathbf{w}^\top \mathbf{x} + b = \sum_{i=1}^{M} \alpha_i \Phi(\mathbf{x}_i) + b,$$

and that this Gaussian's variance is given by σ^2. Then the likelihood of the complete dataset can be written as

$$\begin{aligned} p(\mathbf{y}|\mathbf{ff}, b, \sigma^2) &= \frac{1}{(2\pi\sigma^2)^{M/2}} \exp\left(-\frac{1}{2\sigma^2}\|K\mathbf{ff} + \mathbf{1}b - \mathbf{y}\|^2\right) \\ &= \frac{1}{(2\pi\sigma^2)^{M/2}} \exp\left(-\frac{1}{2\sigma^2}\|\,,\|^2\right), \end{aligned}$$

where , is defined as in (3.38). Now, in a Bayesian reasoning one does not simply solve this problem for a pre–specified value of σ but introduces a prior on the parameters $\mathbf{ff}$, b and σ with additional hyper-parameters. This prior can also be seen as imposing some regularization. Especially, choosing a Gaussian prior with zero mean favors small "weights". Specifically, for the RVM one chooses a prior of the form

$$p(\mathbf{ff}, b|\mathbf{C}) = \mathcal{N}(b|0, C_0) \prod_{i=1}^{M} \mathcal{N}(\mathbf{ff}_i|0, C_i),$$

where one assumes a single hyper-parameter for each α_i and the offset b. Leaving out the specification of a prior for σ^2 for now, one can compute the posterior over the parameters α_i and b as

$$p(\mathbf{ff}, b|\mathbf{y}, \mathbf{C}, \sigma^2) = p(\mathbf{y}|\mathbf{ff}, b, \sigma^2) p(\mathbf{ff}, b|\mathbf{C}).$$

In a fully Bayesian framework one would have to introduce another level of priors over the hyper-parameters and also for σ^2 until ultimately arriving at a parameter free prior. However, in RVMs one stops here and computes the maximum likelihood solution without explicit priors on C_i and σ^2. Implicitly this corresponds to assuming a uniform prior over these parameters.

An advantage of the RVM approach is that all hyper-parameters σ and $\mathbf{C}$ are estimated automatically. The drawback, however, is that for RVM one has to solve a hard, computationally expensive optimization problem. The following simplifications show how KFD can be seen as an approximation to this probabilistic approach. Assuming the noise variance σ is known (i.e. dropping all terms depending solely on σ) and taking the logarithm of the posterior $p(\mathbf{y}|\mathbf{ff}, b, \sigma^2)p(\mathbf{ff}, b|\mathbf{C})$, yields the following optimization problem

$$\min_{\mathbf{ff}, b, ,} \|\,,\|^2 + \log(p(\mathbf{ff}, b|\mathbf{C})), \tag{3.65}$$

subject to the constraint (3.38). If we now interpret the prior as a regularization operator P, replace the vector of hyper-parameters $\mathbf{C} = (C_i)$ by a single parameter C this yields the KFD problem (3.37).

The disadvantage of this is twofold: The parameters σ^2 and C have now to be chosen by model selection instead of being estimated automatically and using a single C we loose the appealing high degree of sparsity present in RVM. The advantage however is that we get a tractable optimization problem with a global minimum. This probabilistic interpretation of KFD also has some appealing properties, which we outline in the following.

Interpretation of Outputs

The above probabilistic framework reflects the fact, that the outputs produced by KFD can be interpreted as probabilities, thus making it possible to assign a confidence to the final classification. This is in contrast to SVMs whose outputs can not directly be seen as probabilities (see also Platt (2001) and Section 3.2.2).

RVM Noise Models and KFD Loss

In the above discussion we assumed a Gaussian noise model and some yet unspecified prior that corresponds to a regularizer (Smola et al., 1998b). Of course, one is not limited to Gaussian models. E.g. assuming a Laplacian noise model we would get $\|\cdot\|_1$ instead of $\|\cdot\|_2^2$ in the objective (3.65) or (3.37), respectively, yielding what was introduced as robust KFD (Section 3.4.8). We have already discussed the connection between noise models and loss functions in Section 2.1.4 (cf. Table 2.1 giving a selection of different noise models and their corresponding loss functions, which could be used and Figure 2.3 for an illustration). All of them still lead to convex linear or quadratic programming problems in the KFD framework.

RVM Priors and KFD Regularizers

Still open in this probabilistic interpretation of KFD was the choice of the prior $p(\mathbf{ff}, b|\mathbf{C})$. One choice would be a zero–mean Gaussian as for the RVM. Assuming again that this Gaussians' variance $\mathbf{C}$ is known and a multiple of the identity, this would lead to a regularizer of the form $P(\mathbf{ff}) = \|\mathbf{ff}\|^2$ as proposed earlier in Section 3.4.7. As noted before the disadvantage is that choosing a single, fixed variance parameter for all $\mathbf{ff}$ we do not achieve sparsity as in RVM anymore. But of course any other choice, again using the connection between densities and loss functions (or regularizers, respectively) illustrated in Table 2.1 is possible. Especially interesting here is the choice of a Laplacian prior, which in the optimization procedure would correspond to a l_1–loss on the $\mathbf{ff}$'s, i.e. $P(\mathbf{ff}) = \|\mathbf{ff}\|_1$ and yields the sparse KFD problem or linear sparse KFD problem, respectively, depending on the noise model.

3.6.3 Support Vector Machines

Considering the fact that the least squares (kernel–regression) problem solved by (3.37) is equivalent to KFD, it is straight forward to derive a corresponding regression technique. Instead of ± 1 outputs y we now have real–valued y's. In fact, such an approach has been proposed in Saunders et al. (1998a), as a "kernelized" version of ridge regression.

The connection to SVM regression (e.g. Vapnik, 1995) is obvious: it is equivalent to problem (3.37) with the ε–insensitive loss for $\cdot$ (cf. Table 2.1) and a K–regularizer, i.e. $P(\mathbf{ff}) = \mathbf{ff}^\top K \mathbf{ff} = \|\mathbf{w}\|^2$.

We can as well draw the connection to a SVM classifier. In SVM classification one is maximizing the (smallest) margin, traded off against the complexity controlled by $\|\mathbf{w}\|^2$ (cf. Section 2.2). Contrary, besides parallels in the algorithmic formulation, there is no explicit concept of a margin in KFD. Instead, implicitly,

the *average margin*, i.e. the average distance of the examples to the boundary, is maximized (see also Figure 5.7).

Of course, since KFD is equivalent, we can also use the motivation that lead to the least squares SVM approach to link up SVM and KFD. The difference between LQSVM/KFD and SVM is that the inequality constraints generating the margin are replaced by equality constraints. This way we loose the large margin but achieve a small with-in class variance.

3.6.4 Arc-GV

The last connection we want to make here is to an algorithm called Arc-GV (Breiman, 1997). Arc-GV is a Boosting-type algorithm (Freund and Schapire, 1997; Friedman et al., 1998; Breiman, 1997). In Boosting (or more precisely but less intuitively, Leveraging) we are given a set of potential base hypotheses $\{h_i : \mathcal{X} \to \mathcal{Y}\}$ which is implemented through a so called weak learner. Now the idea is that even if each single hypothesis is not much better than chance we can build a good classifier if we combine many weak hypotheses into a "strong" one, i.e. we seek some function $f : \mathcal{X} \to \mathcal{Y}$ that is a linear (actually convex) combination of the base hypotheses,

$$f(\mathbf{x}) = \sum_{j=1}^{J} w_j h_j(\mathbf{x}), w_j \geq 0, \tag{3.66}$$

which we call combined hypothesis. This clearly is similar to the kernel expansion (3.25) found in kernel-based methods and there has been some work showing that these techniques are very much related (cf. Rätsch, 2001). Indeed, one can show that most Boosting techniques can be understood as also maximizing some margin. However, this margin is not measured in the kernel space (which might be infinite dimensional) but in the space spanned by the used hypotheses. Also, this margin is not measured using the ℓ_2-norm but the ℓ_1-norm. For details see e.g. Rätsch (2001) and references there in. More information about Boosting and its most important extensions can be found in Mason et al. (1998); Rätsch et al. (2001, 2000c). Boosting and Leveraging techniques usually differ in how they choose the next hypothesis that is added to (3.66). We will not discuss how the original Arc-GV algorithm does this. However, in Rätsch et al. (2000c) it was shown that Arc-GV iteratively constructs a solution that converges to the optimum of the following mathematical programming problem:

$$\begin{array}{ll} \max & \rho \\ \text{s.t.} & y_i \sum_{j=1}^{J} w_j h_j(\mathbf{x}_i) \geq \rho \quad \forall i \ , \\ & \|\mathbf{w}\|_1 = 1, \end{array}$$

where the quantity ρ is the margin which should be maximized. If we now recall that an SVM tries to find a hyperplane such that the margin is maximized and assume that we drop the threshold b, the dual optimization problem solved by an SVM (cf. (2.22)) can be stated as

$$\begin{array}{ll} \max & \rho \\ \text{s.t.} & y_i \sum_{j=1}^{N} w_j \Phi_j(\mathbf{x}_i) \geq \rho \quad \forall i \\ & \|\mathbf{w}\|_2 = 1. \end{array}$$

Here N denotes the dimensionality of the feature space. We see that this is identical to the Arc-GV problem except for the ℓ_2-norm constraint on the weight vector $\mathbf{w}$ and that the hypothesis we can choose from are the single dimensions of the feature space mapping Φ. For SVM we needed to use the two norm since using the kernel we do not have explicit access to the single dimensions j of the feature space and can hence not compute the ℓ_1-norm.

3.6.5 Mathematical Programming and Learning

Collecting the ideas from the previous sections, we can build up a table where we have a smooth transition from KFD via SVM to Arc-GV on the level of their mathematical programming formulations (cf. Table 3.1). This is not meant to be

Table 3.1: Structural comparison of different techniques on the level of mathematical programming formulations.

	primal	**dual**
KFD/LQSVM	$\min \frac{1}{2}\|\mathbf{w}\|_2^2 + C\|\xi\|_2^2$ s.t. $y_i((\mathbf{w}\cdot\Phi(\mathbf{x}_i))+b) = 1-\xi_i \quad \forall i$	
SKFD	$\min \frac{1}{2}\|\mathbf{w}\|_2^2 + C\|\xi\|_1$ s.t. $y_i((\mathbf{w}\cdot\Phi(\mathbf{x}_i))+b) = 1-\xi_i \quad \forall i$	
SVM	$\min \frac{1}{2}\|\mathbf{w}\|_2^2 + C\|\xi\|_1$ s.t. $y_i((\mathbf{w}\cdot\Phi(\mathbf{x}_i))+b) \geq 1-\xi_i \quad \forall i$	$\max \rho$ s.t. $y_i \sum_{j=1}^{N} w_j \Phi_j(\mathbf{x}_i) \geq \rho \quad \forall i$ $\|\mathbf{w}\|_2 = 1$
Arc-GV		$\max \rho$ s.t. $y_i \sum_{j=1}^{J} w_j h_j(\mathbf{x}_i) \geq \rho \quad \forall i$ $\|\mathbf{w}\|_1 = 1$

a rigorous mathematical equivalence but rather a structural analogy. The primal sparse KFD (SKFD) problem and the primal SVM problem differ in the way ξ is computed, the dual SVM problem and the (dual) Arc-GV problem in the way the length of $\mathbf{w}$ is penalized. We tried to highlight the parts that are essentially different in Table 3.1. It can be seen, that in the primal perspective the way of penalizing and computing the slacks is important. In the dual domain the way we measure the length of $\mathbf{w}$ is of importance. This naturally suggest a complete host of other algorithms one could derive by playing around with these quantities. However, having already proposed more than enough KFD variants we shall refrain from this.

Interestingly enough, we see that many (and there are more than just the algorithms listed in Figure 3.1; see e.g. Rätsch (2001)) state of the art learning techniques can be cast as mathematical optimization problems. Naturally these problems are structurally very similar. An appealing property of the mathematical programming formulation is that we might use all the tools from optimization theory to analyze these methods. Furthermore, such a structural equivalence might be taken as one explanation why all these methods perform similarly good in practice. It might also be taken as a basis to apply algorithms in new domains, e.g. to derive Boosting algorithms for one-class classification (Rätsch et al., 2002) or regression (Rätsch et al., 2002). In fact, as we outlined in Section 3.2.1 one

could see KFD as regression technique that is applied to classification.

3.7 Relation to Other Work

Since we published our first paper on kernel Fisher discriminants (Mika et al., 1999a) there has been a fairly large amount of work targeting in a similar direction. Indeed, KFD can be derived from a variety of different concepts, and it only recently became clear that there are many algorithms with different names that are actually (more or less) the same.

Independently of us Baudat and Anouar (2000) and Roth and Steinhage (2000) published papers with exactly the same idea, i.e. perform Fisher's discriminant in a kernel feature space. The main difference can be seen in that they both carried out the calculations for the multi-class case which we purposefully avoided here. There has up to now not been any empirical (or theoretical) evidence that shows any advantage of finding multiple features for discrimination at once.

As we already mentioned, the least square support vector machine proposed in Suykens and Vanderwalle (1999), whilst being motivated completely differently is also exactly the same as KFD. This has been noted e.g. in Gestel et al. (2001). The same connection was shown in Xu et al. (2001). Finally, Billings and Lee (2002) yet again proposed to do Fisher's discriminant using kernel functions. Building upon the connection between KFD and a least squares regression to the labels (cf. Section 3.2.1) we have also a connection to the kernel-ridge regression approach proposed in Saunders et al. (1998a).

A variant of Fisher's discriminant for cases where the classes have extremely unbalanced priors is proposed in Elad et al. (2002). They consider the case of discriminating target images (faces) from clutter (i.e. non-faces). The resulting method is, as the authors note, up to a re–weighting equivalent to Fisher. They finally introduce some pre-processing on the images which amounts to a kernel function (although the authors do this explicitly instead of resorting to kernel functions).

On the theoretical side Shashua (1999) showed some straight forward equivalence between support vector machines and KFD.

3.8 Summary

In the task of learning from data it is, to a certain extent, equivalent to have prior knowledge about e.g. invariances or about specific sources of noise. In the case of feature extraction, we seek features, which are all sufficiently (noise-) invariant while still *describing* interesting structure. For classification we compute *discriminating* features that are – at the same time – invariant with respect to certain invariance transformations. Oriented PCA on one side and Fisher's discriminant on the other side, use particularly simple features, since they only consider first and second order statistics for maximizing a Rayleigh coefficient (cf. (3.20)). Since linear methods are often too restricted in real-world applications, we used kernel functions to obtain *nonlinear* versions of these *linear* algorithms, namely oriented

kernel PCA and kernel Fisher discriminant analysis, following the ideas outlined in Schölkopf et al. (1998b) for PCA.

Having once formulated kernel Fisher discriminants we have shown how they can be cast into a quadratic optimization problem allowing us to derive a variety of other algorithms such as robust, sparse and linear sparse KFD. Important for a successful learning technique is that the underlying optimization problem can be solved efficiently. The central problem in KFD, as with all kernel based learning techniques, is that the kernel matrix K can only be stored for small datasets. To circumvent this problem, we proposed several methods to optimize the different algorithms more efficiently using either greedy techniques or exploiting the special structure of the problem. As we will see in Chapter 5, KFD and its variants are very capable of achieving state of the art results. Finally, we have shown how KFD relates to other techniques.

Comparing KFD to other techniques it certainly has it advantages but also it drawbacks. Appealing about KFD is that the underlying idea is very intuitive and well motivated. Furthermore, there exists a global solution and this solution can be found within a reasonable computational complexity. We conjecture that future research on KFD will produce comparably efficient techniques as they are now available for e.g. SVM. However, for the time being the KFD problem is still more demanding than the SVM problem from an algorithmic point of view. A feature that makes KFD an interesting choice in many applications is its strong connection to probabilistic approaches. Often it is not only important to get a small generalization error but also to be able to assign a confidence to the final classification. Unlike for SVM, the outputs of KFD can directly be interpreted as probabilities. A drawback is that there is up to now no satisfying theoretical framework that explains the good performance, this very much in contrast to e.g. SVM. Whilst maximizing the average margin instead of the smallest margin does not seem to be a big difference most up to date theoretical guarantees are not applicable. In Chapter 4 we will try to do a first step in building up a theory capable of explaining the good performance of KFD.

Chapter 4

Bounds

> In nichts zeigt sich der Mangel an mathematischer Bildung mehr, als in einer übertrieben genauen Rechnung.
>
> Carl Friedrich Gauß

In this chapter we present some theoretical results targeting at giving learning guarantees for techniques based on covariance structures like e.g. kernel Fisher discriminants. However, the present material is yet only concerned with principal component analysis. But we conjecture that this could form a basis for deriving generalization error bounds for KFD.

IN the last chapter we have introduced a kernelized version of Fisher's discriminant and some variations. However, we have not yet investigated in how far these algorithms fulfill the requirements of a "good" statistical learning machine as discussed in Chapter 2, e.g. if they are consistent. The question arises if it is possible to give similar generalization guarantees for KFD and its variants as e.g. for SVM (cf. Theorem (2.9) and (2.18))?

The present chapter does not answer this question but we hope it lays the ground for further investigations. Instead of considering the KFD problem we start with investigating if it is possible to give guarantees on the eigenvalues and eigenvectors computed from covariance matrices (i.e. we consider principal component analysis (PCA)). The rational behind this is twofold:

- The PCA problem is structurally simpler than the KFD problem. However, it relies on the same quantities, i.e. we know that the KFD problem can be solved by computing eigenvalues and eigenvectors of matrices based on first and second order statistics of the data.
- If it is not possible to show for PCA that the results possess some "nice" properties it seems unlikely that this would be possible for KFD. The other way around, having developed a framework to analyze the behavior of PCA it should be possible to extend this to KFD.

The techniques used here are very recent in the field of machine learning at the time of writing this thesis. The first part considers eigenvalues of covariance matrices and how fast the empirical eigenvalues converge to their expected value. It uses the concept of *stability* as introduced e.g. in Bousquet and Elisseeff (2002) and concentration inequalities (McDiarmid, 1989; Devroye et al., 1996). The second part tries to bound how much an empirically computed eigenvector deviates from a "true" eigenvector using the concept of *Rademacher* complexity as e.g. introduced in Koltchinskii and Panchenko (2000); Bartlett and Mendelson (2002).

It arises following the question: in how far can bounding the closeness of the eigenvectors or eigenvalues to their true value be useful to give learning guarantees for KFD? Very recently a new paradigm of deriving generalization bounds for learning machines was introduced. Recalling what was presented in Chapter 2, we notice that e.g. VC-theory targets at giving bounds that hold *uniformly* over a complete class of functions. For learning machines this means that we get guarantees over all functions this machine implements, including the worst case. However, since we believe that our learning machines are sensibly designed to pick functions which are related to the specific problem we try to solve, i.e. cover some of the properties of the underlying distribution, such bounds will be overly pessimistic. VC-bounds do not take into account the special way an algorithms searches a class of functions nor do they take into account the known observations.[1] For instance, an algorithm might be able to implement a function class with infinite VC-dimension but for a given problem it will only search in a very limited part of this class. It seems likely that it is possible to derive better bounds by also considering algorithmic details and the training data more closely. Recent work targets exactly at this and seems very promising for algorithms like KFD. We consider especially the concept of stability and its relation to the generalization error as introduced in e.g. Bousquet and Elisseeff (2002) and the concept of algorithmic luckiness as proposed in Herbrich and Williamson (2002).

Stability Bounds Bousquet and Elisseeff (2002) derive bounds for the generalization error that are based on the stability of a specific algorithm. Roughly, stability measures how much the outcome (the decision function) changes upon small changes in the training data. Intuitively, an algorithm that is very stable, i.e. whose solution does not depend much on a single example, tends to have a generalization error that is close to the empirical error (or, as considered in Bousquet and Elisseeff (2002) as well, the leave-one-out error; cf. Section 4.4). This intuition is beautifully summarized in the following citation due to Talagrand (1996) which we take from Bousquet and Elisseeff (2002):

> A random variable that depends (in a "smooth way") on the influence of many independent variables (but not too much on any of them) is essentially constant.

For the results (or similar ones) presented in Bousquet and Elisseeff (2002) to be applicable for KFD one would need to show that KFD is stable, e.g. that the largest change in the output of our classifier when removing one training sample

[1] At least they only do this indirectly by using e.g. the achieved margin, and the training error.

(denoted by $\mathcal{Z} \setminus (\mathbf{x}_i, y_i)$) is smaller than some constant β. More precisely, it is very likely that we can derive good generalization bounds for KFD if it has uniform stability (Bousquet and Elisseeff, 2002):

Definition 4.1 (Uniform Stability). An algorithm A has uniform stability β_M with respect to the loss function ℓ and a sample of size M if

$$\forall \mathcal{Z} \in (\mathcal{X} \times \mathcal{Y})^M, \forall i \in \{1, \ldots, M\} : \| \ell(A_{\mathcal{Z}}, \cdot) - \ell(A_{\mathcal{Z}\setminus(\mathrm{x}_i, y_i)}, \cdot)\|_\infty \leq \beta_M.$$

For the results in Bousquet and Elisseeff (2002) to be nontrivial the constant β_M, seen as function of the number of examples, must scale with $\frac{1}{M}$. Now the directions found by KFD are generalized eigenvectors from covariance structures. If we can show that eigenvectors estimated from covariance structures converge quickly and are stable might enable us to use stability bounds for KFD.

Algorithmic Luckiness An approach in a sense similar to the one just described but motivated differently, is taken by the algorithmic luckiness framework of Herbrich and Williamson (2002) which can be seen as an extension of the pure luckiness (Shawe-Taylor et al., 1998). The idea in both approaches is to reduce the size of the function space whose complexity is measured through covering numbers (e.g. Smola, 1998). The covering number $\mathcal{N}(\varepsilon, \mathcal{G}, \rho)$ at scale $\varepsilon > 0$ of a set $\mathcal{G} \subseteq \mathcal{H}$ (of functions in our case) with respect to the metric $\rho : \mathcal{H} \times \mathcal{H} \to \mathbb{R}^+$ is defined as the size of the smallest subset $\mathcal{C} \subseteq \mathcal{H}$, such that for every $g \in \mathcal{G}$ there exists a $c \in \mathcal{C}$ with $\rho(g, c) \leq \varepsilon$. In other words, it measures how many spheres of radius ε, centered at elements of $\mathcal{H}$ are necessary to *cover* the set $\mathcal{G}$. The smaller the covering number the smaller the set $\mathcal{G}$ is. In fact, what we need to bound is not the covering number of the set of hypothesis but, once the loss function we use is fixed (cf. Chapter 2.1.4) only the covering number of the set of functions induced by it, i.e.

$$\mathcal{L}_\ell(\mathcal{H}) = \{(\mathbf{x}, y) \to \ell(h(\mathbf{x}), y) | h \in \mathcal{H}\},$$

i.e. we form equivalence classes over hypothesis $h \in \mathcal{H}$ which induce an identical loss. In the algorithmic luckiness framework one then introduces a so called algorithmic luckiness function and a lucky set. The algorithmic luckiness function L maps an algorithm A and a training sample $\mathcal{Z}$ to a real value. The lucky set $\mathcal{H}_A(L, \mathcal{Z})$ is a subset of all possible hypothesis the algorithm can choose from. This subset is restricted to those hypothesis that achieve a higher luckiness on an *arbitrary* subset of size $\frac{M}{2}$ of the complete training sample (assuming M is even)[2] compared to the luckiness achieved on the *first* $\frac{M}{2}$ training examples in $\mathcal{Z}$. There are many details which go far beyond the scope of this work. However, what we ultimately need to bound is that with high probability the covering number of the induced loss functions $\mathcal{L}_\ell(\mathcal{H}_A(L, \mathcal{Z}))$ of the lucky set is smaller than some function ω of the luckiness L of our algorithm on an $\frac{M}{2}$ sample (we need an ω-small algorithmic luckiness functions). If this is possible one can derive good generalization bounds depending on the algorithm and the training data likewise. Our hope is that the ratio of between class variance and within class variance optimized for by

[2] Said differently, which achieve a higher luckiness on the *first* $\frac{M}{2}$ examples in any possible permutation of M examples.

KFD could serve as an algorithmic luckiness functions. Since the value of this ratio is given by the largest eigenvalue of a generalized eigenproblem we conjecture that the study of the stability of these empirical eigenvalues can be useful in applying the algorithmic luckiness framework to KFD.

Existing Bounds Since the class of functions implemented by KFD is exactly the class of hyperplanes in the feature space $\mathcal{E}$ connected to the kernel any bound which holds uniformly over this class of functions would be applicable to KFD. However, the practical value of most generalization error bounds is restricted to their shape, rather than their actual prediction: More often than not, their predictions are trivial, i.e. larger than one. What makes generalization error bounds interesting is to see which quantities are involved in these bounds and how they relate to the error. Since KFD does not maximize the margin we can not expect very good predictions from bounds that are tailored to an algorithm that maximizes the margin. Likewise, since the results of KFD are not sparse it would be pointless to apply a bound that depends on the degree of sparsity in the solution (cf. *compression schemes* (Littlestone and Warmuth, 1986)). But drawing the conclusion that KFD can not be an useful algorithm because it does not optimize what is considered important in existing bounds would be too easy: as we will see in Chapter 5 its performance is on par with e.g. SVM and other state of the art techniques and we strongly believe that it is possible to underpin this observation theoretically.

There exists some other work dealing with sensitivity and perturbation analysis of eigenvalues and eigenvectors of (mostly symmetric) matrices. However, standard matrix analysis is concerned with the uncertainty due to numerical errors, rather than with the randomness introduced through the sampling of the data underlying a covariance matrix (e.g. Eisenstat and Ipsen, 1998; van Dorsselaer et al., 2000). In Johnstone (2000) an analysis is carried out for the distribution of the largest principal component. But this analysis is not valid for covariance matrices. Finally, Beran and Srivastava (1985) analyzed the use of bootstrap methods to give confidence regions for functions of covariance matrices.

4.1 Notation

To start, let us review and adopt our notation. Since we consider PCA now, an unsupervised learning technique, let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\} \subseteq \mathbb{R}^N$ be our training sample which we consider to be a multiset, i.e. an unordered collection of objects with possible repetitions. As mentioned above, we want to use the concept of stability. Stability is based on the question what happens to our result if we delete, add or alter one of the training samples. Hence, we define the multiset operations $\mathcal{X} \cup \mathbf{x}$ to be the addition of $\mathbf{x}$ to $\mathcal{X}$ (regardless if there is already a $\mathbf{z} \in \mathcal{X}$ with $\mathbf{x} = \mathbf{z}$) and $\mathcal{X} \setminus \mathbf{x}$ to be the deletion of *one* arbitrary occurrence of $\mathbf{x}$ from $\mathcal{X}$. Hence for our training sample $\mathcal{X} \setminus \mathbf{x}_i$ denotes $\mathcal{X}$ with the i-th training example (or an identical element) removed and $\mathcal{X}(\setminus \mathbf{x}_i) \cup \mathbf{x}$ the sample $\mathcal{X}$ with the i-th (or an identical) element replaced by $\mathbf{x}$. Let $C_{\mathcal{X}}$ and $\mathbf{m}_{\mathcal{X}}$ denote the covariance and

mean estimated from $\mathcal{X}$ respectively, i.e.

$$C_{\mathcal{X}} := \frac{1}{M-1}\sum_{x\in\mathcal{X}}(\mathbf{x}-\mathbf{m}_{\mathcal{X}})(\mathbf{x}-\mathbf{m}_{\mathcal{X}})^{\top}, \tag{4.1}$$

$$\mathbf{m}_{\mathcal{X}} := \frac{1}{M}\sum_{x\in\mathcal{X}}\mathbf{x}, \tag{4.2}$$

where we have chosen an unbiased estimate for the covariance matrix $C_{\mathcal{X}}$, i.e. one can show that

$$\begin{aligned}\mathbb{E}[C_{\mathcal{X}}] &= \lim_{M\to\infty} C_{\mathcal{X}} \\ &=: C,\end{aligned}$$

where C denotes the true covariance matrix of the data given by

$$C := \mathbb{E}\left[(X-\mathbb{E}[X])(X-\mathbb{E}[X])^{\top}\right]. \tag{4.3}$$

If we assume that the data have zero mean we can use the following definition for the covariance matrix:

$$C_{\mathcal{X}} := \frac{1}{M}\sum_{x\in\mathcal{X}}\mathbf{x}\mathbf{x}^{\top}, \tag{4.4}$$

which, under the assumption $\mathbb{E}[X]=0$ is again unbiased.

In the whole chapter we make one very important assumption: We assume that the data all lie within a hyper-sphere of fixed, finite radius, i.e. $\|\mathbf{x}-\mathbb{E}[X]\|\leq R$, $R\in\mathbb{R}_{+}$, for all $\mathbf{x}$ drawn from the true distribution. In many applications this is easily verified. However, there are others when such an assumption does hardly hold. Its importance stems from the fact that it makes us somewhat independent from the dimensionality N of the data by limiting the maximal value x_i in each dimension to R. A trivial consequence is that

$$\max_{x,z}\|\mathbf{x}-\mathbf{z}\|^2 \leq 4R^2,$$

for all $\mathbf{x}$ and $\mathbf{z}$ from the distribution.

4.2 Stability of Eigenvalues

The first result we will establish is concerned with the eigenvalues of an empirical covariance matrix (4.1). The same result has been derived independently in Shawe-Taylor et al. (2002). We show that by changing one element in the training set $\mathcal{X}$ the change in the estimated eigenvalues will only be of order $\frac{1}{M}$.

In how far is such a result interesting in the context of PCA? What we would like to do is to use concentration inequalities to answer the following questions:

1. How much will the true reconstruction error (or retained energy) differ from what we observed empirically?

2. How far away are the empirical eigenvalues from the true eigenvalues?

To understand how these questions are connected to the eigenvalues of the empirical covariance matrix we need to review some more prerequisites. In the following we assume that the reader is fairly familiar with principal component analysis and, more generally, eigenproblems. For more information about PCA see e.g. Duda and Hart (1973); Bishop (1995); Diamantaras and Kung (1996) and references there in. For eigenproblems the reader is referred e.g. to Wilkinson (1965) and Golub and van Loan (1996).

Let us define A as our learning machine, i.e. here PCA, and $A_{\mathcal{X}}$ as the function chosen by our learning machinery when trained on the M-sample $\mathcal{X}$, i.e. $A_{\mathcal{X}}$ is described by a selected number of eigenvectors $\mathbf{v}_i$, $i \in \mathcal{I} \subseteq \{1, \ldots, N\}$, and corresponds to a function $A_{\mathcal{X}} : \mathbb{R}^N \to \mathbb{R}^N$, $A_{\mathcal{X}}(\mathbf{x}) = \sum_{i \in \mathcal{I}} (\mathbf{v}_i \cdot \mathbf{x}) \mathbf{v}_i$. Furthermore, let $\lambda_i(C)$ denote the i-th (not necessarily singular) eigenvalue of the matrix C where $\lambda_1(C) \geq \lambda_2(C) \geq \ldots \geq \lambda_N(C)$. The first question is motivated by the fact that if we define the loss function $\ell(A_{\mathcal{X}}, \mathbf{x})$ as the squared reconstruction error

$$\ell(A_{\mathcal{X}}(\mathbf{x}), \mathbf{x}) = \|\mathbf{x} - A_{\mathcal{X}}(\mathbf{x})\|^2 = \|\mathbf{x} - \sum_{i \in \mathcal{I}} (\mathbf{v}_i \cdot \mathbf{x}) \mathbf{v}_i\|^2,$$

and chose as the empirical error the mean squared error, i.e.

$$\mathrm{R}_{\mathrm{emp}}(A, \mathcal{X}) = \frac{1}{M} \sum_{i=1}^{M} \ell(A_{\mathcal{X}}(\mathbf{x}_i), \mathbf{x}_i),$$

it is well known, that the sum of dropped eigenvalue is equal to the empirical error, i.e.

$$\mathrm{R}_{\mathrm{emp}}(A, \mathcal{X}) = \sum_{i \in \mathcal{J}} \lambda_i(C_{\mathcal{X}}), \tag{4.5}$$

where $\mathcal{J} = \{1, \ldots, N\} \setminus \mathcal{I}$ (e.g. Diamantaras and Kung, 1996). The risk or generalization error is denoted by

$$\mathrm{R}(A, \mathcal{X}) = \int \ell(A_{\mathcal{X}}(\mathbf{x}), \mathbf{x}) dP(\mathbf{x}).$$

Since we want to minimize the reconstruction error the most natural (and probably the only sensible) choice for $\mathcal{I}$ is $\{1, \ldots, k\}$, $1 \leq k \leq N$, i.e. choosing the first k eigenvectors. This way the empirical error (4.5), i.e. the sum of the dropped eigenvalues, will be as small as possible when choosing k directions to project on. Using this notations we reformulate the first question as

$$\mathbb{P}_{\mathcal{X}}[|\, \mathrm{R}(A, \mathcal{X}) - \mathrm{R}_{\mathrm{emp}}(A, \mathcal{X})| \geq t], \tag{4.6}$$

i.e. we want to know how probable it is that the empirically observed reconstruction error if equal to the expected reconstruction error. Likewise the second questions now reads

$$\mathbb{P}_{\mathcal{X}}[|\sum_{\mathcal{I}} (\lambda_i(C) - \lambda_i(C_{\mathcal{X}}))\,| \geq t], \tag{4.7}$$

where $C = \mathbb{E}_{\mathcal{X}}[C_{\mathcal{X}}]$ (cf. (4.3)).

Concentration inequalities provide us with bounds on how much an empirical process deviates from its expected value. In particular, we will use McDiarmid's inequality (McDiarmid, 1989). To state this theorem we first need to define what we understand by stability:

Definition 4.2 (Stability). Let $X_1, \ldots, X_M$ be independent random variables taking values in a set $\mathcal{Z}$ and let $f : \mathcal{Z}^M \to \mathbb{R}$ be a function. If there exist absolute constants c_i, $i = 1, \ldots, M$, such that

$$\sup_{\substack{x_1,\ldots,x_M,\\ x_i' \in \mathcal{Z}}} |f(\mathbf{x}_1, \ldots, \mathbf{x}_M) - f(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_i', \mathbf{x}_{i+1}, \ldots, \mathbf{x}_M)| \leq c_i, \quad 1 \leq i \leq M, \tag{4.8}$$

we call f stable. Analogously, let $\mathcal{F}$ be a class of functions. If for all $f \in \mathcal{F}$ condition (4.8) holds, we call $\mathcal{F}$ stable.

McDiarmid showed in particular for each stable function f the following theorem:

Theorem 4.1 (McDiarmid's Inequality). *Under the conditions of Definition 4.2, for all* $\varepsilon \geq 0$

$$\mathbb{P}\left[f(X_1, \ldots, X_M) - \mathbb{E}\, f(X_1, \ldots, X_M) \geq \varepsilon\right] \leq \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^M c_i^2}\right)$$

and

$$\mathbb{P}\left[\mathbb{E}\, f(X_1, \ldots, X_M) - f(X_1, \ldots, X_M) \geq \varepsilon\right] \leq \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^M c_i^2}\right)$$

These expressions closely resemble the questions we raised, i.e. (4.6) and (4.7): They both involve an empirical quantity ($\mathrm{R}_{\mathrm{emp}}$ and $\lambda_i(C_{\mathcal{X}})$, respectively) and another quantity which is its expected counterpart (R and $\lambda_i(C)$). However, there is one caveat here which is intimately connected to the problem of estimating eigenvalues from empirical covariance matrices: Even if we have chosen $C_{\mathcal{X}}$ to be unbiased, i.e. $\mathbb{E}[C_{\mathcal{X}}] = C$, the relation $\mathbb{E}[\lambda_i(C_{\mathcal{X}})] = \lambda_i(\mathbb{E}[C_{\mathcal{X}}])$ does *not* hold true! Instead, it is well known, that the leading eigenvalues of a covariance matrix are over–estimated and that the trailing eigenvalues are under–estimated. Especially, $\lambda_i(C_{\mathcal{X}})$ is *not* an unbiased estimator of $\lambda_i(C)$.[3]

Remark 4.1. One can show (Wilkinson, 1965) that the function computing the largest eigenvalue λ_1 is a convex function of the matrix (interpreting the symmetric matrix as a $\binom{N+1}{2}$ vector) and that the function λ_N is a concave function[4]

We have not yet resolved this problem in answering our questions (and neither did Shawe-Taylor et al. (2002)). Instead, we prove a slightly weaker statement which is summarized in the following theorem:

Theorem 4.2 (Probabilistic bound on eigenvalues). *Let* $\mathcal{X} \subseteq \mathbb{R}^N$ *be an* M *sample distributed iid. according to some fixed but unknown probability distribution* P *with* $\|\mathbf{x}\| \leq R$ *for all* $\mathbf{x} \sim P$. *Then for any* $\mathcal{I} \subseteq \{1, \ldots, N\}$ *and all* $t > 0$*:*

$$\begin{aligned} \mathbb{P}_{\mathcal{X}}\left(\sum_{j\in\mathcal{I}} \lambda_j(C_{\mathcal{X}}) - \mathbb{E}_{\bar{\mathcal{X}}}\left[\sum_{j\in\mathcal{I}} \lambda_j(C_{\bar{\mathcal{X}}})\right] \geq t\right) &\leq e^{-\frac{t^2(M-2)^2}{32MR^4}} \\ &\leq e^{-\frac{t^2(M-4)}{32R^4}} \end{aligned} \tag{4.9}$$

[3] However, the sum of all empirical eigenvalues is an unbiased estimator of the sum of all true eigenvalues. This simply is true since $\sum \lambda_i(C_{\mathcal{X}}) = \mathrm{tr}(C_{\mathcal{X}})$, the trace is a linear function and hence $\mathbb{E}[\mathrm{tr}(C_{\mathcal{X}})] = \mathrm{tr}(\mathbb{E}[C_{\mathcal{X}}]) = \mathrm{tr}(C)$.

[4] Let Z^{ij} denote a matrix all zero except for $Z^{ij}_{ij} = Z^{ij}_{ji} = 1$. Then by convexity/concavity of the leading/trailing eigenvalue we mean $\lambda_1((1-\gamma)A + \gamma Z^{ij}) \leq (1-\gamma)\lambda_1(A) + \gamma\lambda_1(Z^{ij})$ and $\lambda_N((1-\gamma)A + \gamma Z^{ij}) \geq (1-\gamma)\lambda_N(A) + \gamma\lambda_N(Z^{ij})$.

and

$$\mathbb{P}_{\mathcal{X}}\left(\mathbb{E}_{\bar{\mathcal{X}}}\left[\sum_{j\in\mathcal{I}}\lambda_j(C_{\bar{\mathcal{X}}})\right]-\sum_{j\in\mathcal{I}}\lambda_j(C_{\mathcal{X}})\geq t\right) \leq e^{-\frac{t^2(M-2)^2}{32MR^4}} \leq e^{-\frac{t^2(M-4)}{32R^4}}. \quad (4.10)$$

The proof can be found below. We see that the answer we can give to the second question is not how much the empirical eigenvalues deviate from the true eigenvalues but rather how much they will deviate from the expected eigenvalues when performing PCA on an M sample. This also has consequences for the first question we raised. Using Theorem 4.2 we will show the following weaker statement:

Lemma 4.1 (Probabilistic bound on reconstruction error). *Under the assumptions of Theorem 4.2:*

$$\mathbb{P}_{\mathcal{X}}\left(\mathrm{R}_{\mathrm{emp}}(A,\mathcal{X})-\mathbb{E}_{\bar{\mathcal{X}}}\left[\mathrm{R}_{\mathrm{emp}}(A,\bar{\mathcal{X}})\right]\geq t\right) \leq e^{-\frac{t^2(M-2)^2}{32MR^4}} \leq e^{-\frac{t^2(M-4)}{32R^4}}$$

and

$$\mathbb{P}_{\mathcal{X}}\left(\mathbb{E}_{\bar{\mathcal{X}}}\left[\mathrm{R}_{\mathrm{emp}}(A,\bar{\mathcal{X}})\right]-\mathrm{R}_{\mathrm{emp}}(A,\mathcal{X})\geq t\right) \leq e^{-\frac{t^2(M-2)^2}{32MR^4}} \leq e^{-\frac{t^2(M-4)}{32MR^4}}.$$

Proof. This follows trivially by substituting (4.5) into (4.9) and (4.10), respectively. □

It remains an open question if it is possible to bound the original questions, i.e. (4.6) and (4.7). John Shawe-Taylor and Andre Elisseeff (private communication) suggested that this might be possible by using covering numbers. In principle what one would need to do is to bound the probability that the expected value of the eigenvalues or the expectation of the empirical error is not very different from the true eigenvalue or expected risk, respectively, i.e. $\mathbb{P}[\mathrm{R}(A,\mathcal{X})-\mathbb{E}_{\bar{\mathcal{X}}}[\mathrm{R}_{\mathrm{emp}}(A,\bar{\mathcal{X}})]$ and $\mathbb{P}[\sum_{\mathcal{I}}(\lambda_i(C)-\mathbb{E}_{\bar{\mathcal{X}}}[\lambda_i(C_{\bar{\mathcal{X}}})])]$. We conjecture that bounding these probabilities should be possible with a high confidence.

4.2.1 Proof of Theorem 4.2

To prove Theorem 4.2 we will show that the eigenvalues of an empirical covariance matrix are stable in the sense of Definition 4.2. First note that exchanging one element in the sample we estimate the covariance matrix from does only cause a small change. The following relation can be shown by a straight forward calculation:

Lemma 4.2 (Update of Covariance Matrices). *Consider the definition* (4.1) *of the empirical covariance matrix. Addition and deletion of one element from the training set $\mathcal{X}$ results in rank one update and a scaling of order one of the empirical*

covariance matrix, i.e.

$$\begin{aligned} C_{\mathcal{X}\cup\mathbf{x}} &= \frac{M-1}{M}\left[C_{\mathcal{X}} + \frac{M}{M^2-1}\left(\mathbf{m}_{\mathcal{X}}-\mathbf{x}\right)\left(\mathbf{m}_{\mathcal{X}}-\mathbf{x}\right)^\top\right] \\ C_{\mathcal{X}\setminus\mathbf{x}_i} &= \frac{M-1}{M-2}\left[C_{\mathcal{X}} - \frac{M}{(M-1)^2}\left(\mathbf{m}_{\mathcal{X}}-\mathbf{x}_i\right)\left(\mathbf{m}_{\mathcal{X}}-\mathbf{x}_i\right)^\top\right]. \end{aligned}$$

Exchanging one element results in a rank two update, i.e.

$$C_{(\mathcal{X}\setminus\mathbf{x}_i)\cup\mathbf{x}} = C_{\mathcal{X}} - \frac{M}{(M-1)^2}\left(\mathbf{m}_{\mathcal{X}}-\mathbf{x}_i\right)\left(\mathbf{m}_{\mathcal{X}}-\mathbf{x}_i\right)^\top + \frac{1}{M}\left(\mathbf{m}_{\mathcal{X}\setminus\mathbf{x}_i}-\mathbf{x}\right)\left(\mathbf{m}_{\mathcal{X}\setminus\mathbf{x}_i}-\mathbf{x}\right)^\top. \tag{4.11}$$

For the definition (4.4) *a similar relation trivially holds true as well.*

Proof. See Appendix B.1. □

Now the question is how much the i-th eigenvalue changes under a rank one or rank two update, respectively. Luckily, there is the following Theorem due to Wilkinson (1965) (see also Golub and van Loan (1996)):

Theorem 4.3 (Eigenvalues under rank 1-updates (Wilkinson, 1965)). *Suppose $B = A + \tau\mathbf{c}\mathbf{c}^\top$ where $A \in \mathbb{R}^{N\times N}$ is symmetric, $\mathbf{c} \in \mathbb{R}^N$ has unit ℓ_2-norm and $\tau \in \mathbb{R}$. If $\tau \geq 0$, then*

$$\lambda_i(B) \in [\lambda_i(A), \lambda_{i-1}(A)], \text{ for } i = 2, \ldots, N,$$

while if $\tau \leq 0$, then

$$\lambda_i(B) \in [\lambda_{i+1}(A), \lambda_i(A)], \text{ for } i = 1, \ldots, N-1.$$

In either case, there exists $m_1, \ldots, m_N \geq 0$, $\sum m_i = 1$, such that

$$\lambda_i(B) = \lambda_i(A) + m_i\tau, i = 1, \ldots, N.$$

The theorem says that whatever the rank one update is, all eigenvalues will just be shifted up or down, respectively, and the total amount of this shift will be exactly τ. As a straight forward consequence of Theorem 4.3 we find:

Lemma 4.3 (Eigenvalues under symmetric rank k update). *Suppose $B = A + \sum_{j=1}^k \tau_j\mathbf{c}_j\mathbf{c}_j^\top$, where A is as in Theorem 4.3, $\mathbf{c}_j \in \mathbb{R}^N$ has unit ℓ_2-norm and $\tau_j \in \mathbb{R}$. Then there exists $m_{ij} \geq 0$, $i = 1, \ldots, N$, $j = 1, \ldots, k$ with $\sum_i m_{ij} = 1$ for all j, such that*

$$\lambda_i(B) = \lambda_i(A) + \sum_{j=1}^k m_{ij}\tau_j, \quad \forall i = 1, \ldots, N.$$

Proof. Define $B_j := B_{j-1} + \tau_j\mathbf{c}_j\mathbf{c}_j^\top$ where $B_0 := A$ (hence $B_k = B$). Then by Theorem 4.3 for each $j = 1, \ldots, k$ there exists $m_{ij} \geq 0$ with $\sum_i m_{ij} = 1$ such that $\lambda_i(B_j) = \lambda_i(B_{j-1}) + m_{ij}\tau_j$ for any i. It follows that

$$\begin{aligned} \lambda_i(B) &= \lambda_i(B_k) \\ &= \lambda_i(B_{k-1}) + m_{ik}\tau_k \\ &= \ldots \\ &= \lambda_i(A) + \sum_{j=1}^k m_{ij}\tau_j \quad \forall i. \end{aligned}$$

□

We will now use this lemma for the special case $k = 2$ to bound the change in the eigenvalues of the empirical covariance matrices under consideration. During the proof we will use the following trivial relation:

Lemma 4.4. *For a symmetric matrix $A \in \mathbb{R}^{N\times N}$ and $\rho \neq 0$ the relation $\lambda_i(A) = \lambda_i(\rho A)/\rho$ holds true.*

We are now ready to state the first main result in proving Theorem 4.2:

Proposition 4.1. *Let $\mathbf{x} \in \mathbb{R}^N$. Then for $j = 1, \ldots, N$ there exists $m_{j1}, m_{j2} \geq 0$, $\sum_j m_{jk} = 1$, $k = 1, 2$, such that*

$$\lambda_j(C_{(\mathcal{X}\setminus\mathbf{x}_i)\cup\mathbf{x}}) = \lambda_j(C_{\mathcal{X}}) - m_{j1}\frac{M\|\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i\|^2}{(M-1)^2} + m_{j2}\frac{\|\mathbf{m}_{\mathcal{X}\setminus\mathbf{x}_i} - \mathbf{x}\|^2}{M},$$

for all $i = 1, \ldots, M$.

Proof. Let

$$\begin{aligned}\Delta_{i1} &= \frac{\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i}{\|\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i\|}\\ \Delta_{i2} &= \frac{\mathbf{m}_{\mathcal{X}\setminus\mathbf{x}_i} - \mathbf{x}}{\|\mathbf{m}_{\mathcal{X}\setminus\mathbf{x}_i} - \mathbf{x}\|},\end{aligned}$$

where we assume that $\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i \neq 0$ and $\mathbf{m}_{\mathcal{X}\setminus\mathbf{x}_i} - \mathbf{x} \neq 0$; otherwise these terms are just omitted in the following. Then we have from equation 4.11 in Lemma 4.2

$$C_{(\mathcal{X}\setminus\mathbf{x}_i)\cup\mathbf{x}} = C_{\mathcal{X}} - \frac{M\|\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i\|^2}{(M-1)^2}\Delta_{i1}\Delta_{i1}^\top + \frac{\|\mathbf{m}_{\mathcal{X}\setminus\mathbf{x}_i} - \mathbf{x}\|^2}{M}\Delta_{i2}\Delta_{i2}^\top.$$

Since $\|\Delta_{ij}\| = 1$ and $C_{\mathcal{X}}$ is symmetric we can apply Lemma 4.3, yielding the desired result. □

Now we have all necessary prerequisites to state that the eigenvalues of the empirical covariance matrix are stable in the sense of (4.8):

Theorem 4.4. *Assume that $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\} \subset \mathbb{R}^N$ is generated according to some unknown but fixed distribution P and that $\|\mathbf{x} - \mathbb{E}[X]\| \leq R < \infty$ for all $\mathbf{x} \sim P$. Let $C_{\mathcal{X}}$ and $\mathbf{m}_{\mathcal{X}}$ be the covariance matrix and mean estimates from $\mathcal{X}$ (cf. (4.1), (4.2)) and let $C_{(\mathcal{X}\setminus\mathbf{x}_i)\cup\mathbf{x}}$, $\mathbf{m}_{(\mathcal{X}\setminus\mathbf{x}_i)\cup\mathbf{x}}$ be the covariance and mean estimated from $\mathcal{X}$ with the i-th element replaced by $\mathbf{x}$. Then for any $\mathcal{I} \subseteq \{1, \ldots, N\}$ and $i = 1, \ldots, M$:*

$$\left|\sum_{j\in\mathcal{I}}\left[\lambda_j(C_{\mathcal{X}}) - \lambda_j(C_{(\mathcal{X}\setminus\mathbf{x}_i)\cup\mathbf{x}})\right]\right| \leq \frac{8R^2}{M-2}.$$

Proof. Using Proposition 4.1 we know that there exists $m_{j1}, m_{j2} \geq 0$, $\sum_j m_{jk} = 1$ $(k = 1, 2)$, such that

$$\left|\sum_{j\in\mathcal{I}}\left[\lambda_j(C_{\mathcal{X}}) - \lambda_j(C_{(\mathcal{X}\setminus\mathbf{x}_i)\cup\mathbf{x}})\right]\right| = \left|\sum_{j\in\mathcal{I}}\left[m_{j2}\frac{\|\mathbf{m}_{\mathcal{X}\setminus\mathbf{x}_i} - \mathbf{x}\|^2}{M} - m_{j1}\frac{M\|\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i\|^2}{(M-1)^2}\right]\right| \quad (4.12)$$

Now, using the triangle inequality and that the m_{jk} sum to one over j (with $m_{jk} \geq 0$) we get:

$$\left| \sum_{j \in \mathcal{I}} \left[\lambda_j(C_{\mathcal{X}}) - \lambda_j(C_{(\mathcal{X} \setminus \mathbf{x}_i) \cup \mathbf{x}}) \right] \right| \leq \frac{\|\mathbf{m}_{\mathcal{X} \setminus \mathbf{x}_i} - \mathbf{x}\|^2}{M} + \frac{M \|\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i\|^2}{(M-1)^2}. \tag{4.13}$$

Since we assumed that all data lie in a ball with radius R we know that $\|\mathbf{m}_{\mathcal{X} \setminus \mathbf{x}_i} - \mathbf{x}\|^2 \leq 4R^2$ and $\|\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i\|^2 \leq 4R^2$. Hence we end up with:

$$\begin{aligned} \left| \sum_{j \in \mathcal{I}} \left[\lambda_j(C_{\mathcal{X}}) - \lambda_j(C_{(\mathcal{X} \setminus \mathbf{x}_i) \cup \mathbf{x}}) \right] \right| &\leq \frac{4R^2}{M} + \frac{4R^2 M}{(M-1)^2} \\ &\leq \frac{8R^2}{M-2}. \end{aligned}$$

□

Theorem 4.2 now follows by a straight forward application of McDiarmid's inequality (cf. Theorem 4.1). Setting $c_i = \frac{8R^2}{M-2}$ yields the desired result.

4.3 Bounding the Change in Eigenvectors

We will now change our focus from eigenvalues to eigenvectors. In particular we will ask how reliable the estimate of an eigenvector from an empirical covariance matrix is. This problem is closely related to what is known as perturbation in matrix theory (Stewart and Sun, 1990). It will turn out, that this problem is more complex than bounding the change in the eigenvalues. While for the eigenvalues the multiplicity of the eigenvalue did not matter (i.e. the number of times λ_i is equal to λ_j) it will for eigenvectors. This can easily be demonstrated: Assume that there are two eigenvectors $\mathbf{v}_{i^*}$ and $\mathbf{v}_{j^*}$ and that the associated eigenvalues are equal, i.e. $\lambda_{i^*} = \lambda_{j^*}$. Then these two eigenvectors span an orthogonal subspace and any other choice of two orthogonal vectors in this subspace can be used instead to represent the eigenvectors.[5] I.e. if we empirically observe two eigenvectors with identical eigenvalues it is impossible to bound how close any of these eigenvectors is to the "true" eigenvector, simply because there is a complete (invariant) subspace the "true" eigenvector can be chosen from. In other words, the question would not be well defined. However, what can be done is to bound how close the subspace spanned by $\mathbf{v}_{i^*}$ and $\mathbf{v}_{j^*}$ is to a true invariant subspace of the distribution. Furthermore, it will turn out that the essential quantity to obtain bounds on the reliability of eigenvectors or subspaces spanned by eigenvectors is how much the corresponding eigenvalues are separated from the other eigenvalues. The larger this gap is, the more reliable the estimate is.

4.3.1 The Bound

To begin with, let us state the result. The first variant, for which we will give a detailed proof in the next section, is concerned with only one eigenvector:

[5] More formally, we have for any vector $\mathsf{z} = \alpha \mathsf{v}_{i^*} + \beta \mathsf{v}_{j^*}$, $\alpha, \beta \in \mathbb{R}$, that $A\mathsf{z} = \alpha A \mathsf{v}_{i^*} + \beta A \mathsf{v}_{j^*} = \lambda_{i^*}(\alpha \mathsf{v}_{i^*} + \beta \mathsf{v}_{j^*}) = \lambda_{i^*} \mathsf{z}$, i.e. each such z is also an eigenvector of A with eigenvalue λ_{i^*}.

Theorem 4.5. *Let $\mathcal{X} \subseteq \mathbb{R}^N$ be a random iid. sample of size M with $\|\mathbf{x}\| \leq R < \infty$, $\mathbb{E}[X] = 0$, and let $C_{\mathcal{X}}$ be as in (4.4). Let $C = \mathbb{E}[C_{\mathcal{X}}]$ and let $\mathbf{v}_i$ be the i-th eigenvector of $C_{\mathcal{X}}$ and V a normal matrix for the subspace orthogonal to $\mathbf{v}_i$. Finally let $E = C - C_{\mathcal{X}}$ and*

$$[\mathbf{v}_i V]^\top C_{\mathcal{X}} [\mathbf{v}_i V] = \begin{bmatrix} \lambda & 0 \\ 0 & D \end{bmatrix}, [\mathbf{v}_i V]^\top E [\mathbf{v}_i V] = \begin{bmatrix} \epsilon & \mathbf{e}^\top \\ \mathbf{e} & E_{22} \end{bmatrix}.$$

Then with probability $(1 - \delta)$ over the random draw of the sample $\mathcal{X}$, if

$$d := \min_{j \neq i} |\lambda_i(C_{\mathcal{X}}) - \lambda_j(C_{\mathcal{X}})| \geq \rho$$

where

$$\rho = \frac{64}{M} \sqrt{\sum_{i=1}^{M} \langle \mathbf{x}_i, \mathbf{x}_i \rangle} + 68R \sqrt{\frac{2}{M} \log\left(\frac{2}{\delta}\right)}$$

there exists $\mathbf{p} \in \mathbb{R}^{N-1}$, $\|\mathbf{p}\|_2 \leq \frac{4}{d} \|\mathbf{e}\|_2$, such that $\tilde{\mathbf{v}} = (\mathbf{v}_i + V\mathbf{p})/\sqrt{1 + \mathbf{p}^\top \mathbf{p}}$ is a unit ℓ_2-norm eigenvector of C. Moreover:

$$\text{dist}(\text{span}(\mathbf{v}_i), \text{span}(\tilde{\mathbf{v}})) \leq \frac{\rho}{d}.$$

A few remarks are in place:

Remark 4.2. The distance $\text{dist}(A, B)$ between two equally dimensional subspaces (here the span of the empirical and true eigenvector) is defined as the norm of the difference between the orthogonal projection operators onto these spaces. In our case these are exactly the eigenvectors itself and hence

$$\text{dist}(\text{span}(\mathbf{v}_i), \text{span}(\tilde{\mathbf{v}})) = \|\mathbf{v}_i - \tilde{\mathbf{v}}\|_2.$$

Remark 4.3. The central statement of the theorem is that if the empirical eigenvalue λ_i corresponding to $\mathbf{v}_i$ is separated from all other eigenvalues (i.e. is a *singular* eigenvalue) at a rate which depends upon the average norm of the observations and decreases like $\frac{1}{\sqrt{M}}$, then with high probability the empirical eigenvector will be close to a true eigenvector.

Remark 4.4. Even if the bound is non–trivial the theorem does not guarantee that the vector $\tilde{\mathbf{v}}$ is actually the i-th eigenvector of C. In principle it could be any eigenvector. But we conjecture that this is very unlikely.

Theorem 4.5 can be generalized to the case when we consider more than one eigenvector at once, i.e. a complete subspace spanned by several eigenvectors.

Lemma 4.5. *Let $\mathcal{X}$, $C_{\mathcal{X}}$, C and E be as in Theorem 4.5. Let $\mathcal{I} = \{i_1, \ldots, i_D\} \subseteq \{1, \ldots, N\}$ and let $V_{\mathcal{I}} = [\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_D}]$, $1 \leq D \leq N$, be a matrix containing the eigenvectors of $C_{\mathcal{X}}$ indexed by $\mathcal{I}$. Finally, let V a normal matrix for the subspace orthogonal to $V_{\mathcal{I}}$. Define*

$$[V_{\mathcal{I}} V]^\top C_{\mathcal{X}} [V_{\mathcal{I}} V] = \begin{bmatrix} \Lambda & 0 \\ 0 & Q \end{bmatrix}, [V_{\mathcal{I}} V]^\top E [V_{\mathcal{I}} V] = \begin{bmatrix} E_{11} & E_{21}^\top \\ E_{21} & E_{22} \end{bmatrix}.$$

Then with probability $(1 - \delta)$ over the random draw of the sample $\mathcal{X}$, if

$$d := \text{sep}(\Lambda, Q) \geq \rho,$$

where

$$\rho = \frac{80}{M}\sqrt{\sum_{i=1}^{M}\langle \mathbf{x}_i, \mathbf{x}_i\rangle} + 85R\sqrt{\frac{2}{M}\log\left(\frac{2}{\delta}\right)}$$

there exists a matrix $P \in \mathbb{R}^{(N-1)\times D}$ *with* $\|P\|_2 \leq \frac{4}{d}\|E_{21}\|_2$*, such that the columns of* $\tilde{V} = (V_{\mathcal{I}} + VP)(I + P^\top P)^{-\frac{1}{2}}$ *define an orthogonal basis for a subspace spanned by* D *eigenvectors of* C*. Moreover:*

$$\text{dist}(\text{span}(V_{\mathcal{I}}), \text{span}(\tilde{V})) \leq \frac{\rho}{d}.$$

Again we need a few remarks:

Remark 4.5. The separation $\text{sep}(A, B)$ of two equally dimensional symmetric matrices is, in analogy to what occurred in Theorem 4.5, defined as

$$\text{sep}(A, B) = \min_{\lambda\in\lambda(A),\mu\in\lambda(B)} |\lambda - \mu|,$$

i.e. in our case it measures the smallest distance between eigenvalues associated with eigenvectors indexed by $\mathcal{I}$ and the remaining eigenvalues indexed by $\{1, \ldots, N\} \setminus \mathcal{I}$.

Remark 4.6. The span (or range) $\text{span}(A)$ of a matrix A is given by the subspace spanned by its columns. The distance of these subspaces is again measured by the distance of the orthogonal projection operators.

Remark 4.7. As said before Lemma 4.5 is the straight forward generalization of Theorem 4.5. If the subspace spanned by the eigenvectors under consideration is sufficiently separated from the remaining subspace spanned by the other eigenvectors, where the sufficiency depends on the average norm of the observations and $\frac{1}{\sqrt{M}}$, then it is very likely that there is a similar subspace spanned by eigenvectors of the true covariance matrix. Again, the Lemma does not imply that this subspace is connected to the same eigenvalues.

4.3.2 Proof of Theorem 4.5

The proofs to Theorem 4.5 and Lemma 4.5 will both be based on the following theorems which are due to Stewart (1973) and can be found e.g. in Golub and van Loan (1996).

Theorem 4.6 (Perturbation of Invariant Subspaces (Stewart, 1973)). *Let* A *and* $A + E$ *be* $N \times N$ *symmetric matrices and let*

$$V = [V_1, V_2], V_1 \in \mathbb{R}^{D\times N}, V_2 \in \mathbb{R}^{(N-D)\times N}$$

be an orthogonal matrix such that $\text{span}(V_1)$ *is an invariant subspace for* A*. Partition the matrices* $V^\top AV$ *and* $V^\top EV$ *as follows:*

$$V^\top AV = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}, V^\top EV = \begin{bmatrix} E_{11} & E_{21}^\top \\ E_{21} & E_{22} \end{bmatrix}.$$

If $d = \text{sep}(Q_1, Q_2) > 0$ *and*

$$\|E\|_2 \leq \frac{d}{5},$$

then there exists a matrix $P \in \mathbb{R}^{(N-D)\times D}$ *with*

$$\|P\|_2 \leq \frac{4}{d}\|E_{21}\|_2$$

such that the columns of $\tilde{V}_1 = (V_1 + V_2P)(I + P^\top P)^{-\frac{1}{2}}$ *define a orthonormal basis for a subspace that is invariant for* $A + E$*. Moreover, then*

$$\text{dist}(\text{span}(V_1), \text{span}(\tilde{V}_1)) \leq \frac{4}{d}\|E_{21}\|_2.$$

An invariant subspaces is any subspace spanned by an arbitrary subset of D eigenvectors of A. Especially, if $\text{span}(V_1)$ is an invariant subspace of A, then $\lambda(A) = \lambda(Q_1) \cup \lambda(Q_2)$. In a slightly less general form, as we will do later in the proofs, we could require the columns of V_1 to be eigenvectors of A. However, what we will show is in principle independent of the representation, i.e. the columns of V_1, chosen for that subspace. In the special case $D = 1$ (then V_1 *must* be an eigenvector of A) there is a slightly refined version of the same theorem:

Theorem 4.7 (Perturbation of Eigenvector (Stewart, 1973)). *Under the assumptions of Theorem 4.6 and the additional assumption that* $D = 1$*, i.e.*

$$V = [\mathbf{v}_1, V_2],$$

where $\mathbf{v}_1$ *is an eigenvector of* A*, partition the matrices* $V^\top AV$ *and* $V^\top EV$ *as follows:*

$$V^\top AV = \begin{bmatrix} \lambda & 0 \\ 0 & Q \end{bmatrix}, V^\top EV = \begin{bmatrix} \epsilon & \mathbf{e}^\top \\ \mathbf{e} & E_{22} \end{bmatrix}.$$

If $d = \min_{\mu\in\lambda(Q)} |\lambda - \mu| > 0$ *and*

$$\|E\|_2 \leq \frac{d}{4},$$

then there exists a vector $\mathbf{p} \in \mathbb{R}^{N-1}$ *with*

$$\|p\|_2 \leq \frac{4}{d}\|\mathbf{e}\|_2$$

such that $\tilde{\mathbf{v}}_1 = (\mathbf{v}_1 + V_2\mathbf{p})/\sqrt{1 + \mathbf{p}^\top\mathbf{p}}$ *is a unit* ℓ_2*-norm eigenvector for* $A + E$*. Moreover,*

$$\text{dist}(\text{span}(\mathbf{v}_1), \text{span}(\tilde{\mathbf{v}}_1)) \leq \frac{4}{d}\|\mathbf{e}\|_2.$$

Note the slightly improved constant in the condition on $\|E\|_2$.

What we will do now is use Theorem 4.7 to prove Theorem 4.5. The proof of Lemma 4.5 follows in complete analogy using Theorem 4.6.

Proof of Theorem 4.5. We will make the proof in several steps.

1. **Applying Theorem 4.7** In Theorem 4.7 set $A = C_\mathcal{X}$ and let $\mathbf{v}_1$ be the i-th eigenvector of $C_\mathcal{X}$. Then it follows from 4.7 that if

$$d := \min_{i\neq j} |\lambda_i(C_\mathcal{X}) - \lambda_j(C_\mathcal{X})| > 0,$$

and

$$\|E\|_2 = \|C - C_{\mathcal{X}}\|_2 \leq \frac{d}{4},$$

there exist the vector $\mathbf{p} \in \mathbb{R}^{N-1}$ with $\|\mathbf{p}\|_2 \leq \frac{4}{d}\|\mathbf{e}\|_2$ such that we can construct $\mathbf{v}$ as being a unit ℓ_2-norm eigenvector of C.

2. **Bounding the Approximation Error** Suppose for a moment that we can apply Theorem 4.7 and that $\|E\|_2 \leq \tilde{\rho} \leq \frac{d}{4}$. Then from Theorem 4.7 it follow that

$$\text{dist}(\text{span}(\mathbf{v}_i), \text{span}(\tilde{\mathbf{v}})) \leq \frac{4}{d}\|\mathbf{e}\|_2.$$

Now, since the ℓ_2-norm is mutually consistent (cf. Golub and van Loan, 1996), i.e. $\|PQ\|_2 \leq \|P\|_2\|Q\|_2$ for matrices P and Q, the following inequality holds true

$$\|\mathbf{e}\|_2 = \|\mathbf{v}_i^\top EV\|_2 \leq \|\mathbf{v}_i\|_2\|E\|_2\|V\|_2 = \|E\|_2,$$

where we used $\|\mathbf{v}_i\|_2 = \|V\|_2 = 1$ since $\mathbf{v}_i$ and V are normal by assumption. Since we assumed that $\|E\| \leq \tilde{\rho}$ it follows that

$$\text{dist}(\text{span}(\mathbf{v}_i), \text{span}(\tilde{\mathbf{v}})) \leq \frac{4\tilde{\rho}}{d}.$$

Since also $d \geq 4\tilde{\rho}$ by assumption, the bound will be non–trivial, i.e. ≤ 1, if it is applicable.

3. **Bounding** $\|E\|_2$ The crucial assumption above was that we could bound $\|E\|_2$ by some constant $\tilde{\rho}$ that is smaller than $\frac{d}{4}$. However, since we can not compute $E = C - C_{\mathcal{X}}$ simply because we do not know C we need to find a bound which does not depend on C. Furthermore, since we do not want to make any unnecessary assumptions about the distribution P generating our data $\mathcal{X}$ such a bound can only be of a probabilistic nature.

What we will do in the following is to derive a bound of the form: with probability $1 - \delta$ over the random draw of $\mathcal{X}$

$$\|E\|_2 \leq \tilde{\rho}(M, R, \mathcal{X}, \delta).$$

Then, setting $\rho = 4\tilde{\rho}$ the proof of Theorem 4.5 follows.

Bounding $\|E\|_2$ We want to bound $\|E\|$ from above. We will do this by casting this problem as the problem of bounding the deviation between an average and its expectation. First, we define the function classes $\mathcal{F}$ and $\mathcal{G}$ as:

$$\mathcal{F} = \{f : f(\mathbf{x}) = (\mathbf{u}^\top\mathbf{x})^2, \|\mathbf{u}\| \leq 1\}, \tag{4.14}$$

$$\mathcal{G} = \{g : g(\mathbf{x}) = \mathbf{u}^\top\mathbf{x}, \|\mathbf{u}\| \leq \frac{1}{R}\}, \tag{4.15}$$

which we will later use to express the covariances involved in E.

Lemma 4.6. *Under the assumption* $\|\mathbf{x}\| \leq R$ *and* $\|\mathbf{u}\| \leq 1$ *we have*

$$\mathbf{u}^\top \mathbf{x} \leq \frac{\mathbf{x}^\top \mathbf{x}}{\|\mathbf{x}\|} = \|\mathbf{x}\| \leq R,$$

and hence

$$\forall f \in \mathcal{F} : |f(\mathbf{x})| = (\mathbf{u}^\top \mathbf{x})^2 \leq R^2.$$

Similarly, with $\|\mathbf{u}\| \leq \frac{1}{R}$ *we get*

$$\forall g \in \mathcal{G} : |g(\mathbf{x})| = |\mathbf{u}^\top \mathbf{x}| \leq 1.$$

Next we need to introduce the concept of Rademacher complexities. We follow Bartlett and Mendelson (2002) and define

Definition 4.3 (Rademacher complexity). Let μ be a probability measure on the space $\mathcal{X}$ and let $\mathcal{F}$ be a class of uniformly bounded functions on $\mathcal{X}$. Then for every integer M the Rademacher complexity $\mathcal{R}_M(\mathcal{F})$ of $\mathcal{F}$ is defined as

$$\mathcal{R}_M(\mathcal{F}) = \mathbb{E}_\mu \, \mathbb{E}_\epsilon \, \frac{2}{M} \sup_{f \in \mathcal{F}} \left| \sum_{i=1}^{M} \epsilon_i f(X_i) \right|, \tag{4.16}$$

where X_i, $i = 1, \ldots, M$ are independent, random variables distributed according to μ and ϵ_i, $i = 1, \ldots, M$ are independent Rademacher random variables, i.e. $\mathbb{P}(\epsilon_i = 1) = \mathbb{P}(\epsilon_i = -1) = \frac{1}{2}$. Likewise, the empirical counterpart $\widehat{\mathcal{R}}_M(\mathcal{F})$ is defined by

$$\widehat{\mathcal{R}}_M(\mathcal{F}) = \mathbb{E}_\epsilon \, \frac{2}{M} \sup_{f \in \mathcal{F}} \left| \sum_{i=1}^{M} \epsilon_i f(X_i) \right|, \tag{4.17}$$

and hence $\mathcal{R}_M(\mathcal{F}) = \mathbb{E}_\mu \, \widehat{\mathcal{R}}_M(\mathcal{F})$ (cf. Bartlett and Mendelson, 2002).

We will have to bound the Rademacher complexity of $\mathcal{F}$ and will do so, by using an existing bound on the Rademacher complexity of $\mathcal{G}$. Luckily, there is a theorem which helps us to relate these two complexities:

Theorem 4.8 (Bartlett and Mendelson (2002), Theorem 12(4)). *Let* $\mathcal{G}$ *be a class of real functions. If* $\varphi : \mathbb{R} \to \mathbb{R}$ *is Lipschitz with constant* L *and satisfies* $\varphi(0) = 0$*, then* $\mathcal{R}_M(\varphi \circ \mathcal{G}) \leq 2L\mathcal{R}_M(\mathcal{G})$.

Here $\varphi \circ \mathcal{G}$ denotes the element wise application of φ to elements $g \in \mathcal{G}$. Now, if we choose $\varphi(x) = \frac{1}{R}x^2$ we have $\varphi \circ \mathcal{G} = \mathcal{F}$ and could hence bound the Rademacher complexity of $\mathcal{F}$ by the one of $\mathcal{G}$, provided the conditions of the theorem are fulfilled. The second condition $\varphi(0) = 0^2 = 0$ if obviously true. The following Lemma shows that φ is also Lipschitz:

Lemma 4.7. *For* $a, b \in \mathbb{R}$, $|a|, |b| \leq D \leq \infty$, *the function* $\varphi : \mathbb{R} \to \mathbb{R}$, $\varphi(a) = \frac{1}{r}a^2$, $r > 0$ *is Lipschitz with constant* $\frac{2D}{r}$*, i.e.*

$$|\varphi(a) - \varphi(b)| \leq \frac{2D}{r}|a - b|.$$

Proof of Lemma 4.7.

$$\begin{aligned} |\varphi(a) - \varphi(b)| &= |\frac{1}{r}a^2 - \frac{1}{r}b^2| \\ &= \frac{1}{r}|(a+b)(a-b)| \\ &= \frac{1}{r}|a+b||a-b| \\ &\leq \frac{1}{r}(|a|+|b|)|a-b| \\ &\leq \frac{2D}{r}|a-b|. \end{aligned}$$

□

Since we assumed $\|\mathbf{x}\| \leq R$ and have already shown $f(\mathbf{x}) \leq R^2$ we can apply Lemma 4.7 with $D = R^2$, $r = R$, leading to the conclusion:

Lemma 4.8 (Bounding $\mathcal{R}_M(\mathcal{F})$). *For the classes $\mathcal{F}$ and $\mathcal{G}$ defined above and under the assumption $\|\mathbf{x}\| \leq R$ for all $\mathbf{x}$ sampled from the assumed distribution:*

$$\mathcal{R}_M(\mathcal{F}) \leq 4R\mathcal{R}_M(\mathcal{G}).$$

What remains now, is to bound $\mathcal{R}_M(\mathcal{G})$ which we will do by using the empirical Rademacher complexity and the following theorem:

Theorem 4.9 (Bartlett and Mendelson (2002), Theorem 11). *Let $\mathcal{G}$ be a class of functions mapping to $[-1, 1]$. For any integer M*

$$\mathbb{P}\left[\left|\mathcal{R}_M(\mathcal{G}) - \frac{2}{M}\sup_{g\in\mathcal{G}}\left|\sum_{i=1}^{M}\sigma_i g(X_i)\right|\right| \geq \epsilon\right] \leq 2\exp\left(\frac{-\epsilon^2 M}{8}\right)$$

and

$$\mathbb{P}\left[\left|\mathcal{R}_M(\mathcal{G}) - \widehat{\mathcal{R}}_M(\mathcal{G})\right| \geq \epsilon\right] \leq 2\exp\left(\frac{-\epsilon^2 M}{8}\right).$$

The proof of this inequalities is a straight forward application of McDiarmid's inequality. One can show that the involved, empirical quantities are stable with $\frac{4}{M}$. We will use the second inequality and since we only need a one–sided bound on $\mathcal{R}_M(\mathcal{G})$ we can drop the factor of two in front. We get, that with probability $1 - \delta$

$$\mathcal{R}_M(\mathcal{G}) \leq \widehat{\mathcal{R}}_M(\mathcal{G}) + \sqrt{\frac{8}{M}\log(\frac{1}{\delta})}. \tag{4.18}$$

This theorem is directly applicable to our class $\mathcal{G}$ under consideration: as shown before, $g \in \mathcal{G}$ maps to $[-1, 1]$ (cf. Lemma 4.6). Finally, we need to compute $\widehat{\mathcal{R}}_M(\mathcal{G})$. This was done by Bartlett and Mendelson (2002) and we reproduce their proof in the appendix:

Lemma 4.9 (Bartlett and Mendelson (2002)). *Let $X_1, \ldots, X_M$ be random elements of $\mathcal{X}$ with $\|X\| \leq R$ as usual and let the class $\mathcal{G}$ of function be defined as above. Then*

$$\widehat{\mathcal{R}}_M(\mathcal{G}) \leq \frac{2}{RM}\sqrt{\sum_{i=1}^{M}\langle X_i, X_i\rangle}.$$

Having collected these prerequisites, let us start to actually bound $\|E\|_2$. By definition

$$\|E\|_2 = \sup_{x \neq 0} \frac{\|E\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \max_{\|x\|=1} \|E\mathbf{x}\|_2$$

From the last equality it follows that

$$\|E\|_2 = \sqrt{\lambda_1(E^\top E)},$$

since

$$\lambda_1(A) := \max_{\|x\|=1} \mathbf{x}^\top A\mathbf{x} \quad \text{and} \quad \|E\mathbf{x}\|_2 = \sqrt{\mathbf{x}^\top E^\top E\mathbf{x}}.$$

Now, since E is real and symmetric there exists U, $V \in \mathbb{R}^{N\times N}$ such that $E = U^\top V U$, U is normal (i.e. $U^\top U = I$) and V a diagonal matrix containing the eigenvalues of E. Then $E^\top E = (U^\top V U)(U^\top V U) = U^\top V^\top V U$ and therefore the eigenvalues of $E^\top E$ are those of E squared. In particular it follows that

$$\sqrt{\lambda_1(E^\top E)} = \max(\lambda_1(E), -\lambda_N(E))$$

First we rewrite $\|E\|$ using the definition of $E = C - C_{\mathcal{X}}$ and the definition of the largest/smallest eigenvalue, yielding the following:

$$\begin{aligned}
\|E\|_2 &= \sqrt{\lambda_1(E^\top E)} \\
&= \max(\lambda_1(E), -\lambda_N(E)) \\
&= \max(\max_{\|u\|=1}(\mathbf{u}^\top E\mathbf{u}), -\min_{\|u\|=1}(\mathbf{u}^\top E\mathbf{u})) \\
&= \max_{\|u\|=1} \left|\mathbf{u}^\top (C - C_{\mathcal{X}})\mathbf{u}\right| \\
&= \max_{\|u\|=1} \left|\mathbf{u}^\top C\mathbf{u} - \mathbf{u}^\top C_{\mathcal{X}}\mathbf{u}\right|.
\end{aligned}$$

Now, since $C_{\mathcal{X}'}$ for arbitrary iid. draw of the sample $\mathcal{X}'$ is an unbiased estimator of C, we have $C = \mathbb{E}\, C_{\mathcal{X}'}$ and

$$\begin{aligned}
\mathbf{u}^\top C\mathbf{u} &= \mathbf{u}^\top (\mathbb{E}_{\mathcal{X}'}\, C_{\mathcal{X}'})\mathbf{u} \\
&= \mathbb{E}_{\mathcal{X}'}[\mathbf{u}^\top C_{\mathcal{X}'}\mathbf{u}],
\end{aligned}$$

for any (fixed) $\mathbf{u}$. Therefore:

$$\|E\|_2 = \max_{\|u\|=1} \left|\mathbb{E}_{\mathcal{X}'}[\mathbf{u}^\top C_{\mathcal{X}'}\mathbf{u}] - \mathbf{u}^\top C_{\mathcal{X}}\mathbf{u})\right|.$$

Now consider again the function class $\mathcal{F}$ defined in (4.14). Since we assumed $\mathbb{E}[X] = 0$, we can write:

$$\begin{aligned}
\mathbf{u}^\top C_{\mathcal{X}}\mathbf{u} &= \frac{1}{M}\sum_{x\in\mathcal{X}} \underbrace{(\mathbf{u}^\top \mathbf{x})^2}_{f(x)\in\mathcal{F}} \\
&= \frac{1}{M}\sum_{x\in\mathcal{X}} f(\mathbf{x}),
\end{aligned}$$

and likewise for $C_{\mathcal{X}'}$. Substituting this expression in (4.19) yields:

$$\begin{aligned}\|E\|_2 &= \max_{\|u\|\leq 1} \left|\mathbb{E}_{\mathcal{X}'}[\mathbf{u}^\top C_{\mathcal{X}'}\mathbf{u}] - \mathbf{u}^\top C_{\mathcal{X}}\mathbf{u}\right| \\ &= \max_{f\in\mathcal{F}} \left|\mathbb{E}_{\mathcal{X}'}\left[\frac{1}{|\mathcal{X}'|}\sum_{y\in\mathcal{X}'} f(\mathbf{y})\right] - \frac{1}{M}\sum_{x\in\mathcal{X}} f(\mathbf{x})\right| \\ &= \max_{f\in\mathcal{F}} \left|\mathbb{E}_{\mathcal{X}'}[f(X')] - \frac{1}{M}\sum_{x\in\mathcal{X}} f(\mathbf{x})\right|. \end{aligned} \tag{4.19}$$

But now we have cast the problem of bounding $\|E\|_2$ as one of bounding the maximal difference between an expectation and the empirical mean.
We start by using that expressions like (4.19) are very stable. The proof is elementary and can be found in the appendix. We get the following Lemma:

Lemma 4.10. *With probability* $1-\delta$

$$\|E\|_2 \leq \mathbb{E}\,\|E\|_2 + \sqrt{\frac{2R^2}{M}\log\left(\frac{1}{\delta}\right)}. \tag{4.20}$$

We continue the proof following Bartlett et al. (2002); van der Vaart and Wellner (1996) where we find a bound on the expectation of (4.19) in terms of the Rademacher complexity of $\mathcal{F}$:

Lemma 4.11 (Bartlett et al. (2002), Lemma 4). *For any class of functions* $\mathcal{F}$*:*

$$\begin{aligned}\mathbb{E}_{\mathcal{X}}\left[\sup_{f\in\mathcal{F}} \mathbb{E}_{\mathcal{X}'}[f(X')] - \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{x}_i)\right] &\leq \mathbb{E}_{\mathcal{X}}\left[\sup_{f\in\mathcal{F}}\left|\mathbb{E}_{\mathcal{X}'}[f(X')] - \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{x}_i)\right|\right] \\ &\leq 2\mathcal{R}_M(\mathcal{F})\end{aligned}$$

We use Lemma 4.11 to conclude from (4.20) that still with probability $1-\delta$:

$$\|E\|_2 \leq 2\mathcal{R}_M(\mathcal{F}) + \sqrt{\frac{2R^2}{M}\log\left(\frac{1}{\delta}\right)}. \tag{4.21}$$

Next, we apply Lemma 4.8 to change the Rademacher complexity of $\mathcal{F}$ into the one of $\mathcal{G}$ yielding

$$\|E\|_2 \leq 8R\mathcal{R}_M(\mathcal{G}) + \sqrt{\frac{2R^2}{M}\log\left(\frac{1}{\delta}\right)}. \tag{4.22}$$

Now we can apply Theorem 4.9 to replace the (true) Rademacher complexity of $\mathcal{G}$ with the empirical complexity and get: With probability $1-\delta$

$$\|E\|_2 \leq 8R\widehat{\mathcal{R}}_M(\mathcal{G}) + 8R\sqrt{\frac{8}{M}\log\left(\frac{2}{\delta}\right)} + \sqrt{\frac{2R^2}{M}\log\left(\frac{2}{\delta}\right)}. \tag{4.23}$$

Finally we bound $\widehat{\mathcal{R}}_M(\mathcal{G})$ using Lemma 4.9 and end get that with probability at least $1-\delta$

$$\|E\|_2 \leq \frac{16}{M}\sqrt{\sum_{i=1}^{M}\langle \mathbf{x}_i, \mathbf{x}_i\rangle} + 8R\sqrt{\frac{8}{M}\log\left(\frac{2}{\delta}\right)} + \sqrt{\frac{2R^2}{M}\log\left(\frac{2}{\delta}\right)} \quad (4.24)$$

$$= \frac{16}{M}\sqrt{\sum_{i=1}^{M}\langle \mathbf{x}_i, \mathbf{x}_i\rangle} + 17R\sqrt{\frac{2}{M}\log\left(\frac{2}{\delta}\right)}. \quad (4.25)$$

Now defining

$$\tilde{\rho} := \frac{16}{M}\sqrt{\sum_{i=1}^{M}\langle \mathbf{x}_i, \mathbf{x}_i\rangle} + 17R\sqrt{\frac{2}{M}\log\left(\frac{2}{\delta}\right)},$$

we find that if

$$\begin{aligned}\rho &= 4\tilde{\rho} \\ &= \frac{64}{M}\sqrt{\sum_{i=1}^{M}\langle \mathbf{x}_i, \mathbf{x}_i\rangle} + 68R\sqrt{\frac{2}{M}\log\left(\frac{2}{\delta}\right)},\end{aligned}$$

is smaller than $d = \min_{i\neq j}|\lambda_i(C_{\mathcal{X}}) - \lambda_j(C_{\mathcal{X}})|$ the conditions of Theorem 4.7 are fulfilled with probability $1-\delta$. This completes the proof of Theorem 4.5.
The proof for Lemma 4.5 is almost identical. In step 2 we use that V_1 and V_2 are normal. The only difference is that in the final step we have the condition

$$\begin{aligned}\rho &= 5\tilde{\rho} \\ &= \frac{80}{M}\sqrt{\sum_{i=1}^{M}\langle \mathbf{x}_i, \mathbf{x}_i\rangle} + 85R\sqrt{\frac{2}{M}\log\left(\frac{2}{\delta}\right)}.\end{aligned}$$

□

4.4 Leave-One-Out Error Calculation for KFD

In this last section we will discuss how to compute the leave–one–out error for KFD. The leave-one-out error (Luntz and Brailowsky, 1969), sometimes called delete estimate, of an algorithm A on a training set $\mathcal{Z} = (\mathcal{X}\times\mathcal{Y})^M$ is defined as

$$\mathrm{R}_{\mathrm{loo}}(A, \mathcal{Z}) = \frac{1}{M}\sum_{i=1}^{M}\ell(A_{\mathcal{Z}\setminus(\mathrm{x}_i, y_i)}, y_i),$$

i.e. it measures the average loss over examples in the training set that we incur if they are left out during training. The leave-one-out error can be seen as an extreme form of cross validation (e.g. Efron and Tibshirani, 1997): Instead of partitioning the training set into, say, five folds and train on four subsets to test on the fifth, the training set is split into M folds, i.e. we do M-fold cross validation. The most remarkable property of the leave-one-out error is that it provides us with

an (almost) unbiased estimate of the generalization error of our learning algorithm A trained on the data $\mathcal{Z}$. One can show (Luntz and Brailowsky, 1969) that

$$\mathrm{R}(A, \mathcal{Z}') = \mathbb{E}_{\mathcal{Z}}[\mathrm{R}_{\mathrm{loo}}(A, \mathcal{Z})],$$

where $\mathcal{Z}'$ denotes a training set of size $M-1$. However, there are also some drawbacks:

- The leave-one-out error is known to have a high variance, i.e. a particular estimate might well be overly optimistic or pessimistic.
- In the absence of any algorithmic advantages one can use (as we shall do below) computation of the leave-one-out error can be very expensive since we have to train the learning machine M times.

Here we deal with the second issue: the computational complexity. Clearly, if we had to solve the KFD problem M times (and M is large) then, since KFD scales roughly with $\mathcal{O}(M^3)$, computation of the leave-one-out error would have a cost of $\mathcal{O}(M^4)$. In many practical applications that would be prohibitive. Here we present a way to compute the leave-one-out error for KFD at a cost of $\mathcal{O}(M^3)$, i.e. the same as running the algorithm itself.

As we have seen in Section 3.5.4, the KFD problem with either the $\|\mathbf{w}\|^2$ or the $\|\mathbf{ff}\|^2$ regularizer can be solved by a system of linear equations. We first consider the case $\|\mathbf{w}\|^2$ which can be solved by

$$\begin{bmatrix} \frac{1}{C}K + I & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix} \begin{bmatrix} \mathbf{ff} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}, \tag{4.26}$$

Now, the idea is that if we leave out one example in the training set this linear system remain largely unchanged. Assume that this left out element is the p-th example. Then we have to solve the same system as before but with the p-th row and column removed. However, equivalently we can solve system (4.26) but setting the p-th row and column to zero and requiring $\alpha_p = 0$ instead. This would amount to the following problem:

$$\begin{bmatrix} \dots & \frac{1}{C}K_{1,p-1} & 0 & \frac{1}{C}K_{1,p+1} & \dots \\ \ddots & \vdots & \vdots & \vdots & \ddots \\ \dots & \frac{1}{C}K_{p-1,p-1}+1 & 0 & \frac{1}{C}K_{p-1,p+1} & \dots \\ \dots & 0 & \frac{1}{C}K_{p,p}+1 & 0 & \dots \\ \dots & \frac{1}{C}K_{p+1,p-1} & 0 & \frac{1}{C}K_{p+1,p+1}+1 & \dots \\ \ddots & \vdots & \vdots & \vdots & \ddots \\ \dots & \frac{1}{C}K_{M,p-1} & 0 & \frac{1}{C}K_{M,p+1} & \dots \\ \dots & 1 & 0 & 1 & \dots \end{bmatrix} \begin{bmatrix} \mathbf{ff}_1 \\ \vdots \\ \mathbf{ff}_{p-1} \\ \mathbf{ff}_p \\ \mathbf{ff}_{p+1} \\ \vdots \\ \mathbf{ff}_M \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_{p-1} \\ 0 \\ y_{p+1} \\ \vdots \\ y_M \\ 0 \end{bmatrix}. \tag{4.27}$$

In fact, requiring $\alpha_p = 0$ is not strictly necessary but keeps the system at full rank. A straight forward calculation reveals that the matrix in (4.27) is just a rank

two update of the matrix in (4.26). Let

$$U = \begin{bmatrix} 0 & \frac{1}{C}K_{p,1} \\ \dots & \dots \\ 0 & \frac{1}{C}K_{p,p-1} \\ 1 & 0 \\ 0 & \frac{1}{C}K_{p,p+1} \\ \dots & \dots \\ 0 & \frac{1}{C}K_{p,M} \end{bmatrix}, V = \begin{bmatrix} \frac{1}{C}K_{p,1} & 0 \\ \dots & \dots \\ \frac{1}{C}K_{p,p-1} & 0 \\ 0 & 1 \\ \frac{1}{C}K_{p,p+1} & 0 \\ \dots & \dots \\ \frac{1}{C}K_{p,M} & 0 \end{bmatrix}.$$

Then we can compute the matrix in the linear system we need to solve if leaving out the p-th example by

$$\begin{bmatrix} \frac{1}{C}K + I & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix} - UV^\top =: A - UV^\top.$$

However, to solve the system what we need is not the matrix itself but its inverse. Here we can use the same trick as we already did for the sparse greedy algorithm: A rank k update of a matrix results in a rank k update of its inverse. More specifically we will use the following formula (Golub and van Loan, 1996):

$$(A - UV^\top)^{-1} = A^{-1} + A^{-1}U(I - V^\top A^{-1}U)^{-1}V^\top A^{-1}.$$

In our case this is an operation of order $\mathcal{O}(M^2)$[6]. Since we have to repeat this procedure M times we can compute the leave-one-out error in $\mathcal{O}(M^3)$.

For the case we use the $\|\mathbf{ff}\|^2$ regularizer the system we have to solve is given by

$$\begin{bmatrix} \frac{1}{C}K^\top K + I & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix} \begin{bmatrix} \mathbf{fi} \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}, \tag{4.28}$$

where here $\mathbf{ff} = \frac{1}{C}K\mathbf{fi}$. We can apply the same trick as before. The only difference is that the rank two update is now given through

$$U = \begin{bmatrix} 0 & \frac{1}{C}K_{p\bullet}K_{\bullet 1} \\ \dots & \dots \\ 0 & \frac{1}{C}K_{p\bullet}K_{\bullet p-1} \\ 1 & 0 \\ 0 & \frac{1}{C}K_{p\bullet}K_{\bullet p+1} \\ \dots & \dots \\ 0 & \frac{1}{C}K_{p\bullet}K_{\bullet M} \end{bmatrix}, V = \begin{bmatrix} \frac{1}{C}K_{p\bullet}K_{\bullet 1} & 0 \\ \dots & \dots \\ \frac{1}{C}K_{p\bullet}K_{\bullet p-1} & 0 \\ 0 & 1 \\ \frac{1}{C}K_{p\bullet}K_{\bullet p+1} & 0 \\ \dots & \dots \\ \frac{1}{C}K_{p\bullet}K_{\bullet M} & 0 \end{bmatrix}.$$

4.5 Summary

We discussed in this chapter two possible ways to derive generalization error bounds for KFD based on stability and algorithmic luckiness. We motivated that bounds of the first kind could be build upon an analysis of the stability of the eigenvector solution computed by KFD while bounds of the second kind could be build upon the eigenvalue estimated in KFD. While it is still ongoing research to actually derive

[6] The matrix $I - V^\top A^{-1}U$ is of size 2×2 and its inverse can be found analytically.

such bounds, we demonstrated that it is at least possible to derive guarantees on the eigenvalues and eigenvectors as computed by simple PCA. Specifically, we have shown that the empirical eigenvalues of PCA are with high probability close to the expected empirical eigenvalues computed on an M sample. We have also shown that an empirical eigenvector is with high probability close to an eigenvector of the underlying distribution. Future work will either try to generalize what has been done here to the generalized eigenproblem solved for by KFD or to use the tools presented in this section to directly derive bound for KFD. But the results presented here might be interesting in their own right. PCA is a widely used technique and the bounds derived here are the first distribution free guarantees for PCA we are aware of. Finally, we showed that it is possible to compute the leave-one-out error of KFD at a reasonable cost which is of the same order as the algorithm itself.

Chapter 5

Applications

> Wer A sagt, muß nicht B sagen. Er kann auch erkennen, daß A falsch war.
>
> Bertold Brecht

This chapter summarizes our experimental results. We illustrate certain aspects of the proposed techniques and give an experimental comparison of the distributions of the output generated by KFD. Also, we evaluate the performance of the different algorithms proposed and present experiments on real world data to demonstrate that KFD is capable of competing with other techniques like SVM.

IN this chapter we first present some experiments on toy data to illustrate the nature of the results produced by the proposed kernel Fisher discriminants. All experiments with KFD, if not stated explicitly, use the $\|\mathbf{ff}\|^2$ regularization, i.e. optimize (3.37), and use RBF kernels (2.35).

In Figure 5.1 we show the results of KFD (cf. (3.35)) and linear sparse KFD (LSKFD, cf. (3.55)) on the same toy data set that was used to illustrate SVMs (cf. Figure 2.10). It can be observed, that introducing the non–linearity through the RBF kernel yields a sensible non-linear decision function in the input space. Also, we see that LSKFD produces solutions which are sparse in the expansion coefficients (cf. (3.25)), i.e. only a few of the training examples are used to describe the solution (in the linear case only two). For SVMs the set of support vectors is structured in the sense that every example that is not outside the margin area becomes a support vector. For LSKFD we do not observe such a behavior: the set of non-zero coefficients does not have a specific structure. However, being allowed to choose any example for the expansion we can get sparser solutions (cf. Table 5.5).

Figure 5.2 shows an illustrative comparison of the features found by KFD and the first and second (non–linear) feature found by kernel PCA (Schölkopf et al., 1998b) on a toy data set. For both we used a polynomial kernel of degree two (cf. (2.36)). Depicted are the two classes (crosses and dots), the feature value

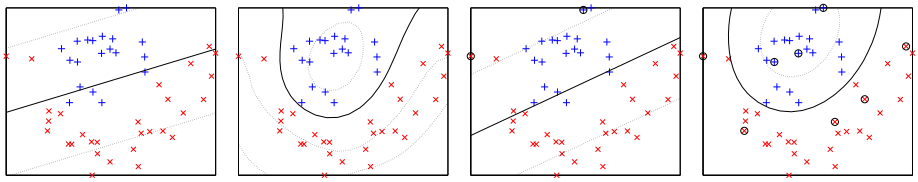

Figure 5.1: Illustration of linear and non–linear KFD and LSKFD. For the non–linear case a RBF kernel (2.35) was used. The left two panels show the results for KFD, the right two panels for LSKFD (see also Figure 2.10 for the result of an SVM on the same toy data). Training examples from the two classes are represented by red 'x' and blue '+', respectively. The solid curve shows the decision surface, i.e. $(\mathbf{w} \cdot \Phi(\mathbf{x})) + b = 0$, the dotted curves show the area where $(\mathbf{w} \cdot \Phi(\mathbf{x})) + b = \pm 1$. For LSKFD "support vectors" are marked by small circles, (for KFD all examples are support vectors).

(indicated by gray level) and contour lines of identical feature value. Each class consists of two noisy parabolic shapes mirrored at the x and y axis respectively. We see, that the KFD feature discriminates the two classes in a nearly optimal way, whereas the kernel PCA features, albeit describing interesting properties of the data set, do not separate the two classes well (although higher order kernel PCA features might be discriminating, too).

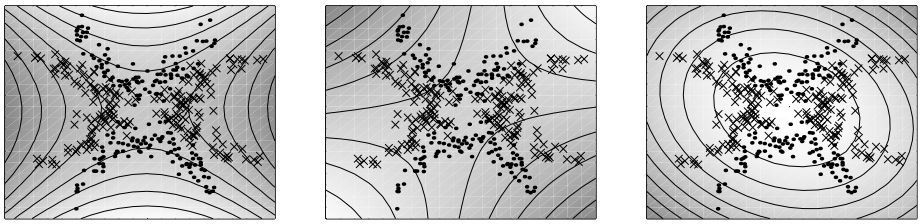

Figure 5.2: Comparison of feature found by KFD (left) and those found by kernel PCA: first (middle) and second (right); details see text.

We repeated a similar experiment on the USPS data set. This data set consists of 7291 scanned and segmented 16×16 gray-value pixel images of handwritten digits (cf. Figure 5.10). We trained linear PCA, kernel PCA, Fisher's discriminant and KFD on the first 500 training examples (where the task was to discriminate digit three against the rest). We used a RBF kernel and, where necessary, set parameters to values that we found to be optimal in other experiments (see Section 5.3.1). Fisher's discriminant as well as KFD solve an eigenproblem and hence it is possible to compute more than just one (the leading) eigenvector. In fact, the next eigenvectors computed from the within-class and between-class scatter are those directions that are most discriminating after the directions with larger eigenvalue have been projected out. What we show in Figure 5.3 is the embedding into a two dimensional space which we get if we project the (complete) data set onto the first two directions found by each method. We see that for both linear PCA and kernel PCA the directions with largest variance are not discriminating. For Fisher's discriminant the data are at least a little separated but there is still a large overlap between the classes. For KFD the separation is evidently much better.

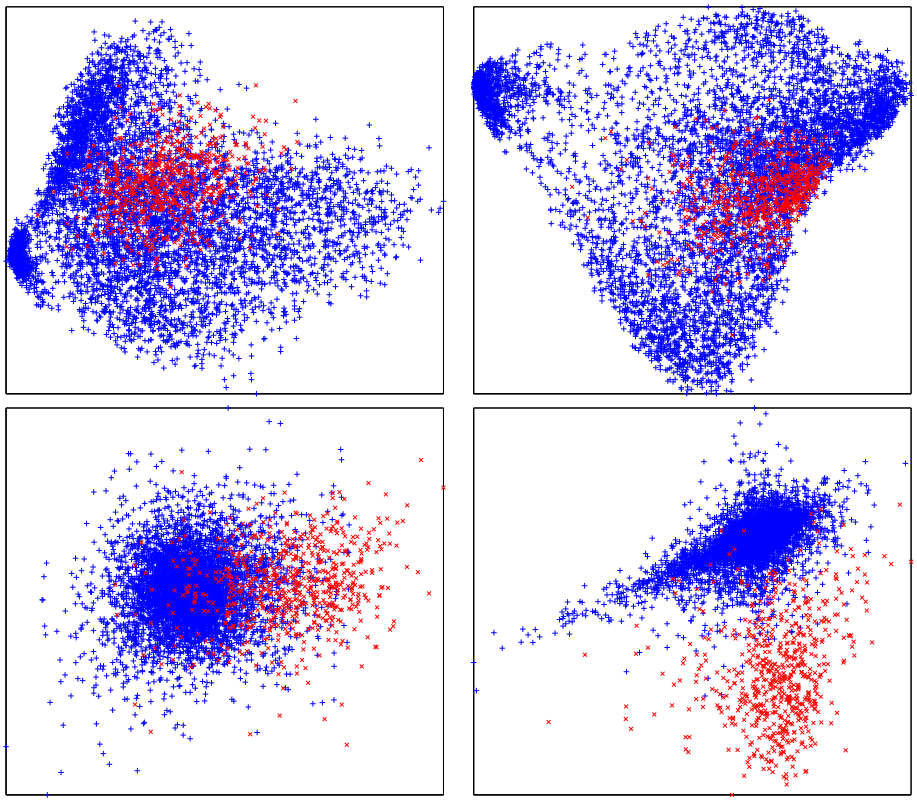

Figure 5.3: Comparison of the first two features found by linear PCA (upper left), kernel PCA (upper right), linear Fisher's discriminant (lower left) and KFD (lower right) on the USPS dataset in discriminating digit 4 (red x) against the remaining nine digits (blue +). Depicted is the feature value extracted by projecting the data onto the first (generalized) eigenvector versus the second (generalized) eigenvector found by the methods (see text for details). PCA, since it is an unsupervised technique does not succeed in discriminating the two classes. Linear Fisher and KFD use the labels. However, the overlap for linear Fisher is substantial and it is much smaller for KFD.

Oriented Kernel PCA A toy example (Figure 5.4) shows a comparison of kernel PCA and oriented kernel PCA (cf. Section 3.4.2). Here we use oriented kernel PCA with the full covariance matrix (4.1) to measure the information in the features (like (kernel) PCA does) and the tangent covariance matrix (3.23) to measure the noise. We used (i) rotated patterns and (ii) along the x-axis translated patterns for the tangent covariance matrix. The toy example shows how imposing the desired invariance yields meaningful invariant features.

Regression Since we have discussed that KFD (and linear Fisher as well) is equivalent to a least squares regression to the labels we can use it as well for regression (cf. Section 3.2.1). To illustrate this fact, we conducted a toy experiment on the "sinc" function (cf. Figure 5.5) using the sparse KFD variant, i.e. (3.53). In terms of the number of support vectors we obtain similarly sparse results as with RVMs Tipping (2000), and a much smaller number of non–zero coefficients than in SVM regression.

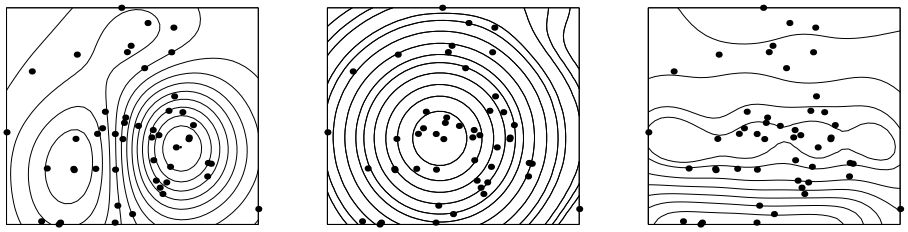

Figure 5.4: Comparison of first features found by kernel PCA and oriented kernel PCA (see text); from left to right: KPCA, oriented KPCA with rotation and translation invariance; all with Gaussian kernel.

5.1 The Output Distribution

Next we compare the output distributions generated by a SVM and KFD as well as some of its variants (cf. Figure 5.6 and Figure 5.7). By maximizing the smallest margin and using linear slack variables for examples which do not achieve a reasonable margin, the SVM produces a training output sharply peaked around ± 1 with Laplacian-like tails inside the margin area (the *inside* margin area is the interval $[-1, 1]$, the *outside* area its complement). Contrarily, plain KFD tends to produce normal distributions which have a small variance along the discriminating direction. Comparing the distributions on the training set to those on the test set, there is almost no difference for KFD with the right amount of regularization. In this sense the direction found on the training data is consistent with the test data. For SVM the output distribution on the test set is significantly different. In the example given in Figure 5.7 KFD and its variants performed slightly better than SVM (1.5%/1.6% vs. 1.7% for the best parameters found by 5-fold cross validation; cf. Section 5.3.2), a fact that is surprising looking only on the training distribution (which is perfectly separated for SVM but has some overlap for the KFD algorithms). We get a similar picture for robust KFD (Section 3.4.8) and linear sparse KFD (Section 3.4.8). In this case the distributions are more Laplacian due to the density model connected to the ℓ_1-penalization (cf. Section 2.1.4).

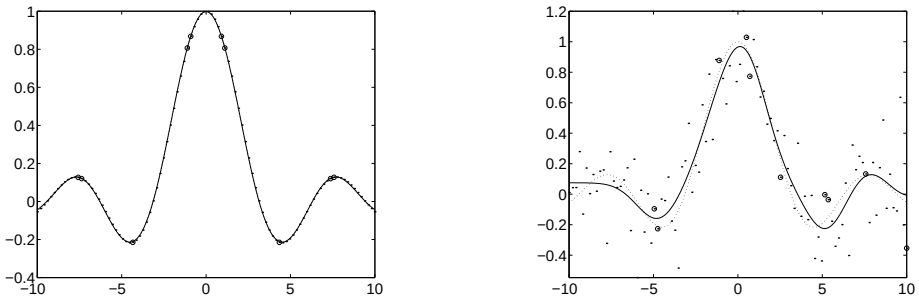

Figure 5.5: Illustration of KFD regression. The left panel shows a fit to the noise–free "sinc" function sampled on 100 equally spaced points, the right panel with Gaussian noise of std. dev. 0.2 added. In both cases we used RBF–kernel $\exp(-\|x - y\|^2/c)$ of width $c = 4.0$ and $c = 3.0$, respectively. The regularization was $C = 0.01$ and $C = 0.1$ (small dots training samples, circled dots support vectors).

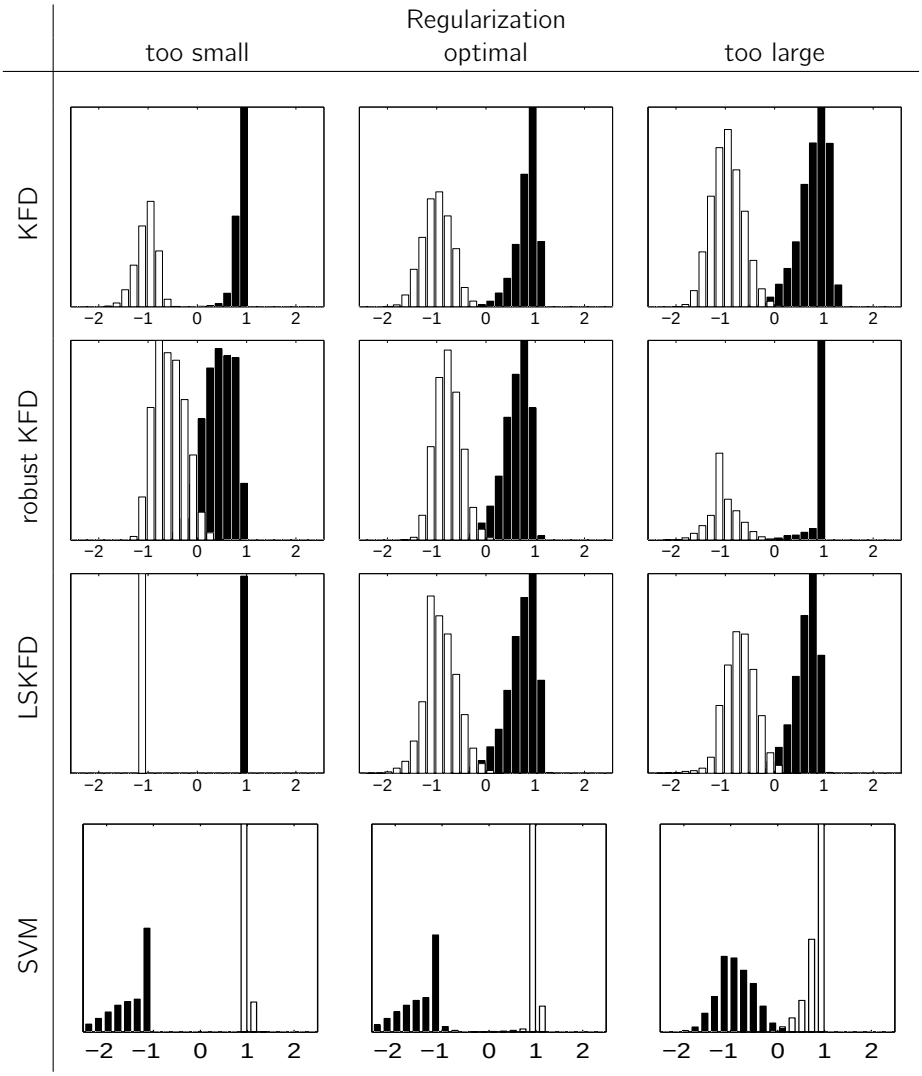

Figure 5.6: Comparison of the output distribution on the training data of the `ringnorm` data set produced by KFD, robust KFD, linear sparse KFD, and a SVM using RBF kernels. Each row shows the results for different choices of the regularization (too small, optimal, too large). The kernel width in each row is constant, set to the optimal value found in the model selection process (see Section 5.3.2). Results are averaged over 100 runs.

Also, we observe peaks at ± 1 in the distribution of the training outputs. But still, the distribution of the projections of the training and test data are much more consistent than for SVM.

5.2 Evaluation of Different Algorithms

We now report experiments to illustrate the run-time behavior of some algorithms proposed in Section 3.5. Specifically we consider the sparse greedy approach (Section 3.5.2), the coordinate descent/SMO approach (Section 3.5.3) and the

column generation approach (Section 3.5.6). The first two are used to optimize the standard KFD problem, the last one is used for the linear sparse KFD. We investigate how fast the methods converge compared to solving the complete optimization problem at once using e.g. a mathematical optimization software. Also, we analyze how much we pay for the approximative character of these methods in terms of performance.

5.2.1 Evaluation of the Sparse Greedy Approach

We will show that the sparse greedy approach from Section 3.5.2 improves significantly over the full quadratic optimization of (3.37). Furthermore, we show that the approximation does not significantly degrade the quality of the solutions. All timing experiments were carried out on the same machine using an optimized linear algebra library (BLAS). We compared this to solving the quadratic program given by (3.37) using as quadratic optimizer `loqo` (Vanderbei, 1997).

Runtime First we compare the runtime of the sparse greedy algorithm to the quadratic optimizer. We used a one-against-the-rest task constructed from the USPS handwritten digit data set (Figure 5.10). All experiments were done with a RBF kernel $\exp(\|\mathbf{x}-\mathbf{y}\|^2/c)$, $c = 0.3 \cdot N$ (N being the dimensionality of the data, i.e. $N = 256$), and with a fixed regularization constant $C = 1$. The results are summarized in Figure 5.8. It can be seen that the new training algorithm halves the scaling exponent of the training time for large sample sizes. In addition, it is important to keep in mind that the sparse greedy approach needs to store at most an $m \times m$ matrix, where m is the maximal number of kernel functions chosen before termination. In contrast, solving the full problem we need to store $M \times M$ matrices.

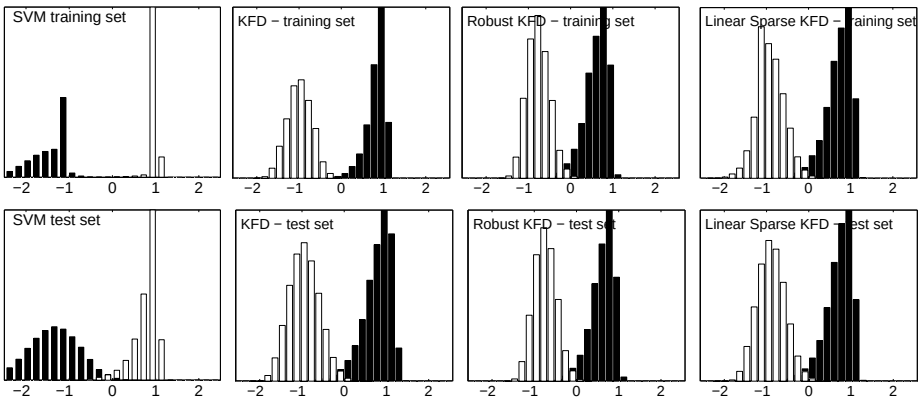

Figure 5.7: Comparison of output distributions on training and test set for SVM and KFD, robust KFD, and linear sparse KFD for optimal parameters on the `ringnorm` dataset (averaged over 100 different partitions). It is clearly observable, that the training and test set distributions for KFD and its variants are quite consistent while they are considerable different for SVM. This might be one explanation for the good performance of KFD.

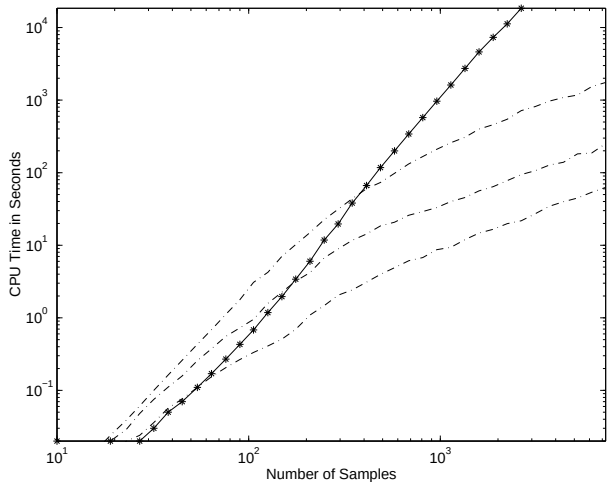

Figure 5.8: Runtime of sparse greedy KFD training. Depicted is the number of samples in the training set versus the CPU time of the sparse greedy algorithm (dash dotted lines) and the QP formulation (3.37) (solid line). The estimates are averages over ten trials, one for each of the ten one–against–the–rest problems in the USPS database. The three lines for sparse greedy KFD are generated by requiring different accuracies on the dual error function in the stopping criterion, namely $10^{-a}, a = 1, \ldots, 3$ relative to the function value (in that order from bottom to top). There is a speed-accuracy trade-off in that for large a, the algorithm converges more slowly. In the log-log plot it can be seen that the QP algorithm roughly scales cubic in the number of samples while the sparse greedy algorithm scales with an exponent of about $\frac{3}{2}$ for large sample sizes .

Performance To test how the performance of this approach relates to the "true" solution we repeated the above experiment on the USPS database for different regularization constants $C = 10^{-3}, 10^{-4}, 10^{-5}$ and different kernel widths $c = 0.3 \cdot N, 0.4 \cdot N, 0.5 \cdot N$. The algorithm was terminated when the average achievement in the dual objective over the last five iterations was less than $10^{-1}, 10^{-2}, 5 \cdot 10^{-3}, 10^{-3}$, respectively, relative to the objective or when a maximum of 500 coefficients was found. As the purpose of this experiment is to show that the sparse greedy approach is capable of producing results comparable to the full system, no model selection was performed and just the best results on the *test set* are reported (cf. Table 5.1). A small improvement in the test error can be achieved by using an optimized threshold b rather than the one given by the algorithm itself. This optimized b is found by training a linear support vector machine on the one dimensional outputs of the training data, i.e. we try to find a threshold which maximizes the smallest distance of the projections to the decision boundary (details e.g. in Mika et al. (1999a)). So far the best result for KFD on the USPS dataset (without using prior knowledge) was 3.7% (Mika et al., 2000), using an expansion restricted to the first 3000 training patterns and the optimized threshold. From Table 5.1 it can be seen that the sparse greedy approach produces results close to the QP solution, however, using a significantly smaller number of kernel functions (less than 500 vs. 3000). It can be observed that the chosen precision for the termination criterion is an important parameter. Although the high precision of 10^{-3} takes longer to train than for smaller precisions, the runtime

Table 5.1: Minimal 10–class test error on the USPS dataset using the parameters described in the text. Shown is the threshold on the improvement in the dual objective used to terminate the algorithm (tolerance), the test error using the threshold given by the algorithm itself, and the test error using an extra, optimized threshold b (see text). The best result of 3.8% is almost identical to the result of 3.7% obtained on the same dataset using an expansion fixed to the first 3000 training examples (Mika et al., 2000). Note, moreover, that for the sparse greedy algorithm the number of examples in the expansion (3.25) is less than 500 in each single classifier.

tolerance	10^{-1}	10^{-2}	$5 \cdot 10^{-3}$	10^{-3}
test error with QP threshold	10.4%	6.4%	5.3%	3.9%
test error with optimized threshold	10.3%	6.3%	5.3%	3.8%

of the approach is more than ten times smaller than solving the QP with 3000 patterns.

5.2.2 Evaluation of Coordinate Descent

Similarly as described above for the sparse greedy approach we evaluated the coordinate descent method described in Section 3.5.3. We used again the USPS dataset and evaluated the time necessary for convergence. Here we varied the minimal number of significant figures (3.62) and the maximal size of the primal infeasibility we tolerate. Since these two quantities are related we coupled them in choosing the minimal number of significant figures as a and requiring a maximal primal infeasibility of 10^{-a}, for $a = 1, 2, 3, 4$. Using again a RBF kernel, the results of our findings are shown in Figure 5.9. The kernel width and the regularization constant were fixed to near optimal values. The direct optimization of the complete mathematical program was done using CPLEX (CPL, 1994), a commercial package. The runtime behavior of the coordinate descent approach is,

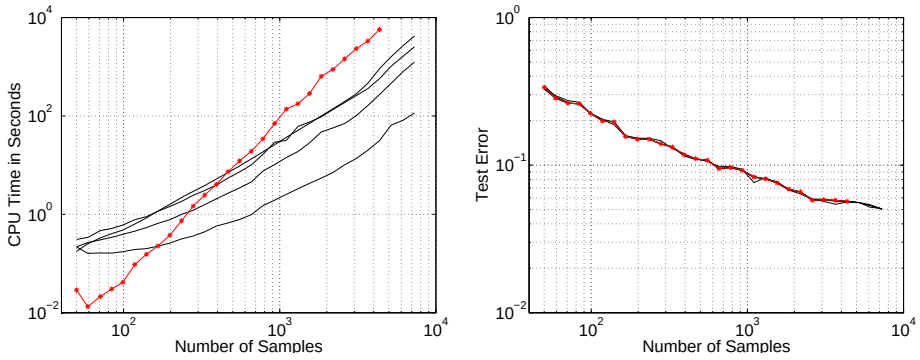

Figure 5.9: Runtime of Coordinate Descent/SMO. Left panel: Shown is the number of examples in the training set versus the CPU time in seconds of the SMO-type algorithm proposed in Section 3.5.3 (black lines) and the QP formulation (3.37) (red line with dots). The estimates are averages over ten trials, one for each of the ten one–against–the–rest problems in the USPS database. The four lines for the coordinate descent method are generated by requiring a different number of significant figures and different accuracies on the primal infeasibility. From top to bottom they were chosen as a and 10^{-a}, respectively, for $a = 1, 2, 3, 4$. The right panel shows the number of examples vs. the 10-class test error. In contrast to the sparse greedy approach, already the lowest precision gives results that are identical to higher precisions or the full solution.

for sample sizes larger than about 500 examples not worse than the optimization of the QP. However, note that this coordinate descent techniques only needs a very small amount of memory which remains constant during the optimization. This in contrast to the full optimization but also compared to the sparse greedy approach which is growing in each iteration. Also, the time spend for each iteration stays constant. In the right panel of Figure 5.9 we plotted the number of training samples versus the ten-class generalization error which was computed using a winner takes all scheme. Particularly amazing is the fact that already the smallest precision of $a = 1$ significant figures yields the same result as solving the complete QP. Interpolating the graph on the left (solving the complete QP for more than 4000 training samples was not feasible anymore), we see that we can get a speed up of a factor 100 and more using the coordinate descent approach. And this with only very little memory needs and without loss in generalization ability. This is also confirmed by the following experiment where we performed a model selection for the coordinate descent approach on the USPS data. The results for one significant figure and a maximal primal infeasibility of 10^{-1} were not worse than for precisions up to four significant figures and an infeasibility of at most 10^{-4}. However, the training time is much smaller for the low precisions. The best 10-class test error over all tested parameter combinations with one significant figure was 3.5% for a kernel width of $c = 0.3 \cdot 256$ and a regularization constant of $C = 10^{-2}$. This improves the best KFD result of 3.7% on the USPS data set so far which was obtained by expanding into the first 3000 examples only. It is considerably better than the result of a SVM on the same data (without prior knowledge) which achieves an error of 4.2%.

5.2.3 Evaluation of Column Generation

The last algorithm which we want to evaluate here is the column generation approach proposed for linear sparse KFD (cf. Section 3.5.6). We have yet only performed a small benchmark to verify that the solutions are not worse than those of the complete linear optimization problem. To this end, we ran a five fold model selection on ten datasets in 5 different realizations for 25 combinations of the parameters C and c using RBF kernel. In fact, we repeated what we will present in Section 5.3.2. To get more reliable estimated we ran this complete model selection 5 times, i.e. we had to do $5 \cdot 10 \cdot 5 \cdot 25 \cdot 5 = 31250$ training runs. Averaged over these 31250 repetitions, solving the full linear program using CPLEX (CPL, 1994) took 3.1 seconds per run, using the column generation approach (also based on CPLEX) took 2.6 seconds per run. The difference in the results was negligible.

However, we also noticed, that using larger data set (for instance the USPS data) the column generation approach ran into difficulties. It is unclear yet whether these problems are related to the technique itself or rather to the implementation in the CPLEX software. Also, it was observed e.g. for linear programming machines (Demiriz et al., 2001) that the speed of column generation crucially depends on the problem formulation. Small changes which leave the problem identical from a mathematical point of view can have a big influence on the convergence.

5.3 Benchmark Performance

We will now evaluate in more detail how KFD and its variants compare to other techniques on real world and benchmark data sets. First we present an experiment where we incorporated invariances into an optical character recognition task using KFD (Section 5.3.1). In another set of experiments (Section 5.3.2) we compare KFD, sparse KFD and linear sparse KFD to support vector machines, radial basis function networks, AdaBoost and regularized AdaBoost.

5.3.1 USPS Invariance Experiment

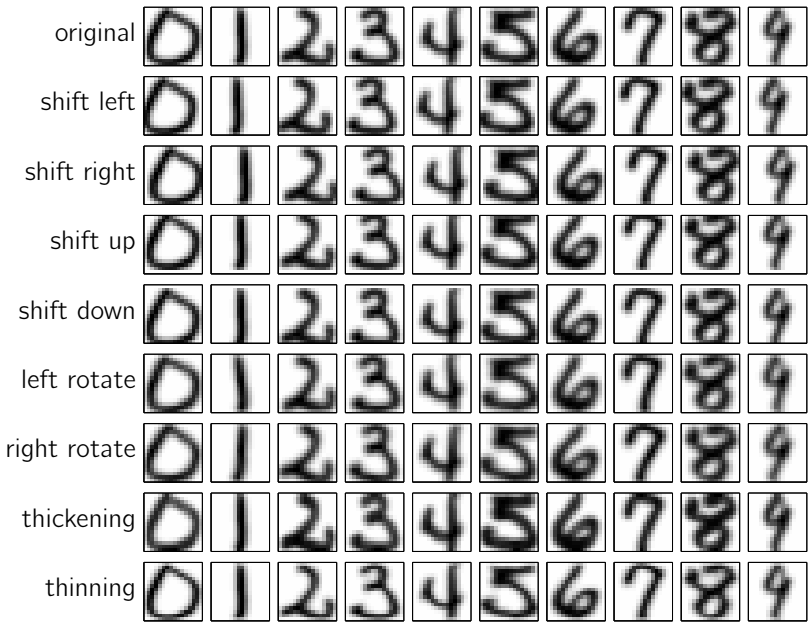

Figure 5.10: Examples of the USPS handwritten digit data used in some of the experiments. Each column shows the first occurrence of a digit in the training set. The top row shows the original data. Each pair of subsequent rows shows the horizontally translated, vertically translated, rotated and thickened/thinned example, respectively, used in the invariance experiment. The data set consists of 7291 training examples and 2007 test examples, each 16×16 gray value pixels.

In a first experiment we incorporate prior knowledge in KFD for the USPS database (cf. Figure 5.10). We used the regularized within-class scatter (3.33) ($\mu = 10^{-3}$) as S_N and added to it an multiple λ of the tangent covariance (3.23), i.e. $S_N = N_\mu + \lambda T$. As invariance transformations we have chosen horizontal and vertical translation, rotation, and thickening (cf. Schölkopf, 1997), where we simply averaged the matrices corresponding to each transformation. Figure 5.10 shows some examples of these data and the applied transformations. The feature was extracted by using the restricted expansion (3.57), where the patterns $\mathbf{z}_i$ were the first 3000 training examples. As kernel we have chosen a Gaussian with width $c = 0.3 \cdot N$, which is optimal for SVMs (Schölkopf, 1997). For each class we trained a KFD which classified that class against the rest and computed the 10-class error

by the winner-takes-all scheme. The threshold was estimated by minimizing the empirical risk on the normalized outputs of KFD and then choosing the solution with a maximal margin (which is very simple in this case since the projected data are one dimensional).
Without invariances, i.e. $\lambda = 0$, we achieved a test error of 3.7%, slightly better than a plain SVM with the same kernel (4.2%) (Schölkopf, 1997). For $\lambda = 10^{-3}$, using the tangent covariance matrix led to a very slight improvement to 3.6%. That the result was not significantly better than the corresponding one for KFD (3.7%) can be attributed to the fact that we used the same expansion coefficients in both cases. The tangent covariance matrix, however, lives in a slightly different subspace (cf. discussion in Section 3.4.5). And indeed, a subsequent experiment where we used vectors which were obtained by k–means clustering a larger dataset, including virtual examples generated by appropriate invariance transformation, led to 3.1%, comparable to an SVM using prior knowledge (e.g. Schölkopf (1997); best SVM result 2.9% with local kernel and virtual support vectors).

5.3.2 IDA Benchmarks

In this final set of experiments we perform an extensive comparison of KFD, sparse KFD and linear sparse KFD to other techniques on a collection of 10 different data sets. These data sets come from a variety of sources, among them the statlog repository[1] and the UCI repository[2]. Those of them which are originally multi-class problems have been partitioned into two class problems by building two super-classes from the original classes. If there originally were separate test and training/validation sets they have been merged to form one larger data set. Then these data sets of now only two class problems have randomly been split into 100 partitions of training, and test data, keeping the class ratio at the level of the complete data set. The data and the 100 splits can be found online at the Fraunhofer FIRST.IDA benchmark repository[3]. More details on how the data were generated can be found in Rätsch et al. (2001).

We compared kernel Fisher discriminants and support vector machines, both with Gaussian kernel, to RBF-networks (e.g. Moody and Darken, 1989), AdaBoost (Freund and Schapire, 1997) and regularized AdaBoost (Rätsch et al., 2001) (cf. Table 5.3). For KFD we used the regularized within-class scatter (3.33) and computed projections onto the optimal direction $\mathbf{w} \in \mathcal{F}$ by means of (3.32). To use $\mathbf{w}$ for classification we have to estimate a threshold. Although being seemingly a trivial problem there is a host of possibilities to do this and which threshold we choose can have a large influence on the solution. For various reasons we did not use the threshold that the quadratic programming formulation of KFD suggests, i.e. the one which minimizes the mean squared error between labels and output (cf. (3.35)). In practice this threshold shows a rather poor performance and the example in Section 3.2.3 also suggests that this is not a good choice. Again we decided to compute the threshold such that we maximize the margin on the outputs in analogy to a support vector machine, where we deal with errors on the training set by using the SVM soft margin approach. The computational cost

[1] `ftp://ftp.ncc.up.pt/pub/statlog`
[2] `http://www.ics.uci.edu/~mlearn/MLRepository.html`; (Blake and Merz, 1998)
[3] `http://ida.first.fhg.de/~raetsch/data/benchmarks.htm`

Table 5.2: Statistics of the data sets used in the comparison.

	dimensionality	Size of training set	test set
Banana	2	400	4900
B.Cancer	9	200	77
Diabetes	8	468	300
German	20	700	300
Heart	13	170	100
Ringnorm	20	400	7000
F.Sonar	9	666	400
Thyroid	5	140	75
Titanic	3	150	2051
Waveform	21	400	4600

is negligible and this one–dimensional problem can easily be solved using gradient descent. The results in Table 5.3 show the average test error and the standard deviation of the averages' estimation, over 100 runs with the different realizations of the datasets. To estimate the necessary parameters, we ran 5-fold cross validation on the first five realizations of the training sets and took the model parameters to be the median over the five estimates. The same setting has been used in Rätsch et al. (2001). Usually we did more than just one model selection run by first coarsely scanning the range of candidate values for the parameters and then refining this range in further runs. It has to be noted, that such this model selection procedure is not quite clean. The way the data were generated, there will be examples which are in the test set of one partition that are in the training set of another partition. Then by averaging over the estimated parameters and repeating this procedure the algorithm potentially gains some knowledge about the test set of a partition. But since we applied this procedure consistently for all algorithms we do not expect a big influence in the comparison, except that the results are slightly biased.

Comparing only plain KFD to the other techniques KFD yields the best result in 5 out of 10 cases and is among the best two methods in 9 out of 10 cases. Comparing SVM and the three KFD variants (cf. Table 5.4) we get a similar picture.

However, comparing the average performance over the ten data sets suggest that SVM, KFD, SKFD and LSKFD perform equally well, closely followed by regularized AdaBoost. Only RBF networks and plain AdaBoost are slightly worse although this difference is hardly significant.

In Table 5.5 we compare the sparsity of the optimal solutions produced by SVM, sparse KFD and linear sparse KFD (for plain KFD all coefficients are non-zero). It is noteworthy that the solutions of SKFD and LSKFD are much sparser than those of an SVM. As we have seen in the previous tables this high sparsity can be achieved without a loss in terms of the generalization error.[4]

[4]However, one should note that it is possible to post-process the SVM solution to get a sparser approximation using so called reduced set methods (e.g. Burges and Schölkopf, 1997; Schölkopf et al., 1999b). On the other hand, these methods are applicable to KFD as well.

Table 5.3: Comparison between KFD, a single RBF classifier, AdaBoost (AB), regularized AdaBoost (AB_R) and support vector machine (SVM) (see text). Best method in bold face, second best emphasized. Shown is the generalization error for the ten data sets averaged over 100 partitions and the averages' standard deviation.

	RBF	AB	AB_R	SVM	KFD
Banana	**10.8±0.06**	12.3±0.07	*10.9±0.04*	11.5±0.07	**10.8±0.05**
B.Cancer	27.6±0.47	30.4±0.47	26.5±0.45	*26.0±0.47*	**25.8±0.46**
Diabetes	24.3±0.19	26.5±0.23	23.8±0.18	*23.5±0.17*	**23.2±0.16**
German	24.7±0.24	27.5±0.25	24.3±0.21	**23.6±0.21**	*23.7±0.22*
Heart	17.6±0.33	20.3±0.34	16.5±0.35	**16.0±0.33**	*16.1±0.34*
Ringnorm	1.7±0.02	1.9±0.03	*1.6±0.01*	1.7±0.01	**1.5±0.01**
F.Sonar	34.4±0.20	35.7±0.18	34.2±0.22	**32.4±0.18**	*33.2±0.17*
Thyroid	4.5±0.21	*4.4±0.22*	4.6±0.22	4.8±0.22	**4.2±0.21**
Titanic	23.3±0.13	*22.6±0.12*	*22.6±0.12*	**22.4±0.10**	23.2±0.20
Waveform	10.7±0.11	10.8±0.06	**9.8±0.08**	*9.9±0.04*	*9.9±0.04*
Average	18.0%	20.2%	17.5%	17.2%	17.2%

Table 5.4: Comparison between SVM, KFD, sparse KFD (SKFD) and sparse KFD with linear loss on , (LSKFD) (see text). Best result in bold face, second best in italics (which is by coincidence also consistent with Table 5.3). Also shown is the average performance over all data sets which suggest that KFD and its variants are well competitive with SVM.

	SVM	KFD	SKFD	LSKFD
Banana	11.5±0.07	*10.8±0.05*	11.2±0.48	**10.6±0.04**
B.Cancer	26.0±0.47	*25.8±0.46*	**25.2±0.44**	*25.8±0.47*
Diabetes	23.5±0.17	*23.2±0.16*	**23.1±0.18**	23.6±0.18
German	**23.6±0.21**	*23.7±0.22*	**23.6±0.23**	24.1±0.23
Heart	**16.0±0.33**	*16.1±0.34*	16.4±0.31	**16.0±0.36**
Ringnorm	1.7±0.01	**1.5±0.01**	*1.6±0.01*	**1.5±0.01**
F.Sonar	**32.4±0.18**	*33.2±0.17*	33.4±0.17	34.4±0.23
Thyroid	4.8±0.22	**4.2±0.21**	*4.3±0.18*	4.7±0.22
Titanic	**22.4±0.10**	23.2±0.20	22.6±0.17	*22.5±0.20*
Waveform	**9.9±0.04**	**9.9±0.04**	*10.1±0.04*	10.2±0.04
Average	17.2%	17.2%	17.2%	17.3%

5.3.3 Other Work

There have been some successful applications of KFD, especially in the area of computer vision, in particular in the field of face detection. This might have historical reasons as Fisher's (linear) discriminant has been very popular in this field. But it might also stem from the fact that people in these areas are specifically interested to obtain outputs which are interpretable as probabilities, this for instance in contrast to SVM. Yang et al. (2000) generalized what is known as eigenfaces to the kernel case building upon KFD. A series of papers published at the *5th IEEE conference on automatic face and gesture recognition* also used KFD for face recognition: Kurita and Taguchi (2002) modify KFD to be make it easier applicable to face data, and Liu et al. (2002) applied KFD in practical face recognition tasks. Additionally Yang (2002) compared kernel eigenfaces (kernel PCA eigenvectors) against what they call kernel fisher-eigenfaces (i.e. the eigen-

Table 5.5: Comparison of the degree of sparsity achieved with SVM, sparse KFD and linear sparse KFD. Shown is the fraction of expansion coefficient which are zero for the optimal solution and the average over all data sets. Note the much higher degree of sparsity in SKFD and LSKFD than for SVM at a comparable performance (cf. Table 5.4).

	SVM	SKFD	LSKFD
Banana	78%	86%	92%
B.Cancer	42%	88%	88%
Diabetes	57%	97%	97%
German	58%	96%	98%
Heart	51%	88%	96%
Ringnorm	62%	85%	94%
F.Sonar	9%	67%	99%
Thyroid	79%	88%	89%
Titanic	10%	8%	95%
Waveform	60%	81%	96%
Average	44.8%	78.8%	94.4%

vectors found by KFD). In a more general vision setting Bradshaw et al. (2000) applied KFD to learn image semantics, e.g. to discriminate target objects from background clutter or to classify parts on an image as sky, trees and persons.

5.4 Summary

In this chapter we analyzed the proposed KFD techniques experimentally. We have shown that KFD and its variants yield solutions that are as good as the ones obtained using other state of the art techniques, among them SVMs and regularized AdaBoost. We illustrated on real world data sets that sparse and linear sparse KFD yield solutions that only need a small number of examples to describe the decision function. The set of examples chosen to express the solution is very different from the support vectors selected by a SVM. But the freedom to choose any example instead of only those which "support" the separating hyperplane as in SVM might be one explanation for the much higher degree of sparsity in sparse and linear sparse KFD. On the USPS data set of handwritten digits, we demonstrated that the proposed algorithms for KFD work in practice. The sparse greedy approximation, the coordinate descent (SMO) techniques or the column generation method achieve the desired speed up in computation time. When comparing the solutions of the proposed algorithms, we observe no loss in terms of the generalization error. However, training of KFD is so far still more expensive than for example training a SVM. But we conjecture that further research on optimization strategies will help circumvent this shortcoming, especially for the sparse KFD variants. For practical applications with only a small number of examples (less than 1000) solving the corresponding eigenproblem is a good choice: The implementation is straight forward, and fast and reliable packages to find eigenvectors or invert matrices are available for most computer platforms. For larger datasets the coordinate descent method poses an interesting alternative. Especially since, as observed in the experiments, already a low precision (and hence quickly con-

verging) solution yields good results. If computational cost in evaluating the final decision is an issue (e.g. in real time applications) the linear sparse KFD technique is an interesting option because of its high sparsity.

In another set of experiments we demonstrated on toy and real world data that incorporating prior knowledge into oriented kernel PCA and KFD using the tangent covariance matrix yields reasonable results. Using the USPS data set with invariances we were able to achieve a test error that is close to the best results reported so far for this data.

Usually the good performance of SVMs is explained by the large margin and the sparsity of the solution. However, KFD does not maximize the margin (only the average margin) and only for SKFD and LSKFD the solutions are sparse. We conjecture that one explanation for the good performance of KFD might be the strong consistency of the output distributions on training and test set. Under the assumption that the training and test data are identically sampled from the same distribution this similarity appears desirable. Comparing these distributions for a SVM they do differ significantly.

Finally, we have shown on toy data and the USPS set that directions found by KFD might be interesting for visualization of high dimensional data. Using the fact that with KFD we can compute more than just one discriminating direction, embeddings into two and more dimensions are easily realizable. Since KFD is able to use the prior knowledge coded for by the labels we expect that such embeddings will more closely reflect the class structure. In particular in comparison to unsupervised techniques like (kernel) PCA or for example locally linear embedding (Roweis and Saul, 2000). However, a thorough evaluation of KFD as an embedding technique still has to be carried out.

Chapter 6

Conclusion

> Ein Mathematiker weiß nie, wovon er spricht, noch ob das, was er sagt, wahr ist.
>
> Bertrand Russell

IN this thesis we have considered learning methods based on the maximization of a Rayleigh coefficient. We proposed non–linear generalizations of methods like oriented PCA, and especially Fisher's discriminant. We started by reviewing some background material of statistical learning theory and discussed under which theoretical conditions learning is possible. It turned out that it is particularly important to have some mechanism that allows to control the complexity of the resulting machine. We then demonstrated how to turn a linear, scalar-product based algorithm into a non–linear algorithm using kernel functions.

Then, as the central part of this thesis, we applied this "kernelization" idea to Rayleigh coefficients. We have shown how introducing kernel functions results in powerful, state of the art learning techniques, namely oriented kernel PCA and kernel Fisher discriminants (KFD). We then focused primarily on KFD and discussed how to formulate KFD as a mathematical optimization problem. Within this framework we proposed several ways to introduce an appropriate regularization. Also, we were able to derive several interesting variations of KFD within the mathematical programming framework: robust KFD, sparse KFD, and linear sparse KFD. Additionally we discussed how to efficiently solve the optimization problems arising in KFD and its variants. Among the proposed techniques, the sparse greedy optimization (cf. Section 3.5.2) and the coordinate descent approach (cf. Section 3.5.3) are especially interesting for solving the KFD problem. For linear sparse KFD especially the column generation approach proposed in Section 3.5.6 is very well suited. Finally, we showed how KFD relates to techniques like support vector machines, relevance vector machines and Arc-GV. Seen as mathematical optimization problems these techniques are structurally extremely similar, their main difference lying in the way how training errors and model complexity are measured. We conjecture that on the basis of mathematical programming

formulations it is possible to explain at least partially why seemingly so different techniques perform similarly well in practice.

Having successfully derived KFD we discussed first results and directions that target at giving generalization error bounds for KFD. We motivate why recent techniques based on the notion of stability and algorithmic luckiness are particularly interesting for KFD. Two possible directions to derive error bounds for KFD would be (i) to show that resulting eigen-directions are very stable and concentrated and (ii) that the achieved ratio of between class variance and within class variance can serve as an algorithmic luckiness measure. The first way would require the stability of the eigenvectors found by KFD, the second way the stability of the eigenvalues. As a first step we subsequently derived bounds for the derivation of empirically estimated eigenvalues and eigenvectors from their expectations. These bounds are interesting in their own right since they establish the first distribution free learning guarantees for PCA. Finally, we showed that it is possible for KFD to compute the leave one out error at a cost which is just a constant multiple of the computational cost for solving a single KFD problem.

To illustrate interesting properties and to evaluate the performance of KFD and its variants we presented a large collection of experiments using artificial and real world data. Comparing the distributions of the training and test data when projected onto the direction of discrimination between KFD variants and support vector machines showed an important difference in both approaches. For all KFD variants the distribution of the training data closely resembles the one of the test data. The distribution of the data in the direction of discrimination is consistent between training and test set. For support vector machines, however, we observe the converse: The training set outputs have a distribution which strongly differs from the distribution on the projected test data. We conjecture that this consistence of KFD might be one reason for its good generalization ability.

In a second set of experiments we evaluated the performance of the proposed algorithms and showed that they reduce the computational cost for solving the KFD problems considerably (compared to using a full quadratic optimization or an eigensolver) without a loss in terms of generalization ability.

Finally we evaluated the performance of KFD, sparse KFD and linear sparse KFD on a large collection of data sets. Comparing these results to support vector machines, AdaBoost and radial basis function networks showed, that the performance of the KFD algorithms is on par or better than the other approaches. In particular, KFD generates results that are equally good as those produced by a support vector machine, today's de facto standard in learning. Here we also showed that sparse KFD and linear sparse KFD produce results that are noticeably sparser than those of a support vector machine at a comparable performance.

In summary, we demonstrated that the kernel based versions of Fisher's discriminant belong to the best learning techniques available today. Their intuitive interpretation, the capability to produce results that can be interpreted as probabilities and their simple implementation make them particularly interesting for many applications. However, we also showed that most state of the art learning techniques, besides being based on similar optimization problems, have a performance that

is close to each other. It would certainly be wrong to draw from this work the conclusion the KFD is better than other techniques, but we have demonstrated that KFD yields good results. And as for most algorithms, there are particular situations in which the special way of solving the learning problem imposed by KFD has its advantages.

Appendix A

Mathematical Programming

In this appendix we collected some material of general interest dealing with mathematical optimization problems and techniques. We review some basic concepts such as duality and the Karush-Kuhn-Tucker conditions. We conclude by reviewing a special technique for solving linear and quadratic programs, the so called interior point methods and how they can be applied to KFD.

A.1 Linear and Quadratic Optimization

Mathematical programming is concerned with the question of how to find solutions to problems of the form

$$\min_{x \in \mathcal{X}} f(\mathbf{x}), \text{ subject to } \mathbf{x} \in \mathcal{S}, \tag{A.1}$$

i.e. how to find solutions $\mathbf{x}$ that (globally) minimize the *objective* function f and are in the set of feasible solutions $\mathcal{S}$ (*constraints*). Depending on the structure of f, $\mathcal{X}$ and $\mathcal{S}$ such a problem can be relatively easy or very difficult to solve. Here we will only discuss the cases of convex optimization, i.e. the objective f is a convex function and the constraint set $\mathcal{S}$ can be written as

$$\mathbf{x} \in \mathcal{S} \Leftrightarrow c_i(\mathbf{x}) \leq 0, \forall i = 1, \ldots, N,$$

for some (finite of infinite) number N of convex constraint functions $c_i : \mathcal{X} \to \mathbb{R}$. The most prominent examples of this type of problems are known as linear (LP) and quadratic programming (QP), respectively. The generic LP or QP can be written in the form

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top H\mathbf{x} + \mathbf{c}^\top\mathbf{x}, \text{ subject to } A\mathbf{x} \leq \mathbf{b}, \tag{A.2}$$

where for linear problems H is all zero (i.e. missing) and a positive matrix otherwise. A is a $N \times M$ matrix of constrains where each *row* $A_{i\bullet}$ of A codes for the constraint $A_{i\bullet}\mathbf{x} \leq b_i$, and the vector $\mathbf{b}$ summarizes the right hand sides. Note that every possible LP/QP can be written in this generic form by suitably transforming

the constraints[1]. Both, LPs and QPs have some nice properties making them especially interesting:

- If there is a feasible solution at all, the problem is either unbounded (i.e. the optimal objective value is $-\infty$ or there exists a global (finite) minimum. However, this minimum needs not to be unique.
- There are polynomial time algorithms (in the number of variables, i.e. the dimensionality of $\mathbf{x}$) that are capable of either finding such a global minimum or determining the infeasibility of the problem.

We will not go into much detail about convex optimization and mathematical programming in general but only review some basic facts used in this thesis. More information can be found in one of the many textbooks about optimization (e.g. Luenberger, 1984; Bertsekas, 1995; Nash and Sofer, 1996).

In the following we assume that the domain $\mathcal{X}$ of our optimization problem is $\mathbb{R}^N$. Central to the issue of linear and quadratic programming are the famous Kuhn-Tucker and Karush-Kuhn-Tucker (KKT) conditions (Karush, 1939; Kuhn and Tucker, 1951). Kuhn & Tucker proved the following Theorem:[2]

Theorem A.1 (Kuhn-Tucker Condition). *Consider the Lagrangian*

$$L(\mathbf{x}, \mathbf{a}) = f(\mathbf{x}) + \mathbf{a}^\top A\mathbf{x}, \tag{A.3}$$

where $\mathbf{a} \geq 0$. *If there exists* $(\bar{\mathbf{x}}, \bar{\mathbf{a}}) \in \mathbb{R}^M \times \mathbb{R}^N_+$ *such that for all* $\mathbf{x} \in \mathbb{R}^M$ *and* $\mathbf{a} \in \mathbb{R}^N_+$

$$L(\bar{\mathbf{x}}, \mathbf{a}) \leq L(\bar{\mathbf{x}}, \bar{\mathbf{a}}) \leq L(\mathbf{x}, \bar{\mathbf{a}}),$$

i.e. $(\bar{\mathbf{x}}, \bar{\mathbf{a}})$ *is a saddle point of the Lagrangian* (A.3)*, then* $\bar{\mathbf{x}}$ *is a solution to* (A.2).

The coefficients a_i summarized in the vector $\mathbf{a}$ are called Lagrange multipliers. From the proof of Theorem A.1 one also gets the well-known KKT conditions: For each of the constraints $A_{i\bullet}\mathbf{x} \leq b_i$ and corresponding Lagrange multiplier a_i either of the following two conditions holds:

$$A_{i\bullet}\mathbf{x} - \mathbf{b}_i = 0 \text{ or } a_i = 0 \text{ or equivalently } (A_{i\bullet}\mathbf{x} - b_i)a_i = 0.$$

Motivated by the last equality, sometimes the KKT conditions are also called complimentary conditions. From Theorem A.1 we see that one possibility to find a solution to our MP problem is to find a saddle point of the Lagrangian. However, the questions arises when the condition of Theorem A.1 is not only sufficient but also necessary. Luckily, for our special setting of quadratic or linear problems, respectively they already are.

Differentiating $L(\mathbf{x}, \mathbf{a})$ with respect to $\mathbf{x}$ and $\mathbf{a}$ and using the saddle point condition, the complementary condition and the fact that $\mathbf{a} \geq 0$, one can show

[1] However, all of the following can be done for an arbitrary mixture of equality and inequality constraints, leading to much more efficient formulations.

[2] Most Theorems and alike here and in the following can be stated in a slightly more general form, e.g. only assuming convexity of the objective and the constraints on some et $\mathcal{X}$. However, we do state them in a way sufficient for our purposes, i.e. linear and quadratic optimization.

that for problems of the form (A.2) we get the following necessary and sufficient optimality conditions:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{x}} L(\mathbf{x}, \mathbf{a}) &\stackrel{!}{=} 0.\\ \frac{\partial}{\partial \mathbf{a}} L(\mathbf{x}, \mathbf{a}) &\stackrel{!}{\leq} 0,\\ \mathbf{a}^\top (A\mathbf{x} - b) &\stackrel{!}{=} 0,\\ \mathbf{a} &\geq 0.\end{aligned}$$

Lets us inspect more closely what this means for an LP and QP respectively:

Linear Optimization The Lagrangian for the linear optimization problem (A.2) (i.e. the matrix H is empty) can be written as:

$$L(\mathbf{x}, \mathbf{a}) = \mathbf{c}^\top \mathbf{x} + \mathbf{a}^\top (A\mathbf{x} - \mathbf{b}).$$

Then the optimality conditions read

$$\begin{aligned}\frac{\partial}{\partial \mathbf{x}} L(\mathbf{x}, \mathbf{a}) =& \mathbf{c} + A\mathbf{a} && = 0.\\ \frac{\partial}{\partial \mathbf{a}} L(\mathbf{x}, \mathbf{a}) =& A\mathbf{x} - b && \leq 0,\\ & \mathbf{a}^\top (A\mathbf{x} - b) && = 0,\\ & \mathbf{a} && \geq 0.\end{aligned}$$

If we now use the first equality in the Lagrangian we can eliminate the primal variables $\mathbf{x}$. If we further recall, that for optimality we had to find a maximal (saddle) point in $\mathbf{a}$ we see that the following dual problem computes exactly the optimal $\bar{\mathbf{a}}$:

$$\max_{\mathbf{a}} -\mathbf{b}^\top \mathbf{a}, \text{ subject to } A\mathbf{a} + \mathbf{c} = 0, \mathbf{a} \geq 0,$$

The complimentary condition will be satisfied since it was just a consequence of the theorem. This dual problem has some interesting properties. Most noticeable, one can show, that for a given pair $(\mathbf{x}, \mathbf{a})$ that is both, primal and dual feasible, the dual objective will always be *smaller* than the primal and only at an optimal solution $(\bar{\mathbf{x}}, \bar{\mathbf{a}})$ they will coincide. The difference between primal and dual objective is called duality gap and can be taken as a measure on how well an intermediate solution approximates the true solution. The second important property in linear optimization is that if we form the dual of the dual, we retain again the primal problem.

Quadratic Optimization If we now consider quadratic optimization problems (i.e. now H is a positive matrix) we get analogous to the linear problem the following Lagrangian

$$L(\mathbf{x}, \mathbf{a}) = \frac{1}{2}\mathbf{x}^\top H\mathbf{x} + \mathbf{c}^\top \mathbf{x} + \mathbf{a}^\top (A\mathbf{x} - \mathbf{b}),$$

together with the optimality conditions

$$\begin{aligned}\frac{\partial}{\partial \mathbf{x}} L(\mathbf{x}, \mathbf{a}) =& H\mathbf{x} + \mathbf{c} + A\mathbf{a} && = 0.\\ \frac{\partial}{\partial \mathbf{a}} L(\mathbf{x}, \mathbf{a}) =& A\mathbf{x} - b && \leq 0,\\ & \mathbf{a}^\top (A\mathbf{x} - b) && = 0,\\ & \mathbf{a} && \geq 0.\end{aligned}$$

We can again use the first equality with the Lagrangian and obtain, in analogy to the linear case, the following dual problem:

$$\max_{\mathbf{a},\mathbf{x}} -\frac{1}{2}\mathbf{x}^\top H\mathbf{x} - \mathbf{b}^\top \mathbf{a} \text{ subject to } H\mathbf{x} + A\mathbf{a} + c = 0, \mathbf{a} \geq 0.$$

If we additionally use the equality constraint to solve for $\mathbf{x}$ by

$$\mathbf{x} = -H^{-1}(c + A\mathbf{a}),$$

we can eliminate the primal variables and obtain, dropping constant factors:

$$\max_{\mathbf{a}} -\frac{1}{2}\mathbf{a}^\top A^\top H^{-1} A\mathbf{a} - (c^\top H^{-1} A^\top + \mathbf{b})\mathbf{a} \text{ subject to } \mathbf{a} \geq 0.$$

Whilst this dual looks less nice than the linear case we have seen, that e.g. for SVM or KFD it can have a pretty simple structure as well.

Finally, lets us summarize some other important facts about the primal-dual relation of linear and quadratic programs. The first fact was already mentioned:

- If a primal (dual) solution exists, there also exists a dual (primal) solution and the optimal objective functions values are equal.

Another important relation is the possibility to identify the case when *no* optimal solution exists via the dual:

- There exists no primal (dual) solution, iff the dual (primal) problem is unbounded from above (below) or infeasible.

Often it will be much easier to determine whether a problem is infeasible than to figure out that there is no optimal solution (e.g. there might be a sequence of feasible solutions that converges to some optimal value; but there is no feasible solution taking this objective value).

A.2 Interior Point Optimization

There are many different techniques to find the optimal solution of a linear or quadratic program, among them the well known simplex methods (Dantzig, 1962), the ellipsoid method (Khachiyan, 1979) or barrier methods (Bertsekas, 1995). In this section we will consider another technique called *interior point* methods (Karmarkar, 1984; Mehrotra, 1992). The idea with interior point methods is to find a point which in primal and dual feasible and fulfills the the Karush-Kuhn-Tucker conditions. As we have seen above such a point, if it exists, will be an

optimal solution. The particular approach to interior point methods presented here follows Vanderbei (1997) and uses an predictor-corrector technique. The name interior point methods stems from the fact that we iteratively estimate approximate solutions to the original problem that lie inside a cone defining the feasible region of our problem, i.e. in the interior of this cone. To achieve this it will turn out that we have to solve a system of linear and non-linear equations. Since solving general non-linear equations in closed form is difficult we resort to an iterative approximation scheme called predictor-corrector approach. The idea is to avoid solving a non-linear problem by solving a sequence of linearized problems. In each iteration we first make a predictor step, solving a linearized version of our problem. Using these predicted values, we resubstitute them into the linearized problem and resolve it to account for higher order effect, yielding the corrector step which is then taken. The exposition will only be made for quadratic optimization problems.

We will now derive a general purpose interior point formulation. Let $M = M_1 + M_2 + M_3$ denote the number of variables, $N = N_1 + N_2$ the number of constraints, and let $H \in \mathbb{R}^{M\times M}$, $A \in \mathbb{R}^{N\times M}$, $b \in \mathbb{R}^N$, and $\mathbf{c}, \mathbf{x} \in \mathbb{R}^M$ with

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \tag{A.4}$$

$$H = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{pmatrix} \qquad \mathbf{c} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{pmatrix} \tag{A.5}$$

be a partitioning which will become clear in the following. Finally, let $A_{\bullet i}$ or $A_{i\bullet}$ denote all blocks in the i-th columns or rows of the partitioned matrix A, respectively. Now we formulate the most general primal problem, containing all sorts of equality and inequality constraints:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top H \mathbf{x} \tag{A.6}$$

subject to:

$$\begin{array}{rcccl} \mathbf{b}_1 & \leq & A_{1\bullet}\mathbf{x} & \leq & \mathbf{b}_1 + \mathbf{r} \\ \mathbf{b}_2 & \leq & A_{2\bullet}\mathbf{x} & & \\ \mathbf{l}_1 & \leq & \mathbf{x}_1 & \leq & \mathbf{u} \\ \mathbf{l}_2 & \leq & \mathbf{x}_2 & & \\ & & \mathbf{x}_3 \text{ free} & & \end{array} \tag{A.7}$$

To derive an interior point formulation the first step is to replace all inequality constraints by simple non-negativity constraints using slack variables. Defining $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2)^\top$, $\mathbf{p}$, $\mathbf{t}$ and $\mathbf{g} = (\mathbf{g}_1, \mathbf{g}_2)^\top$ we eliminate all inequality constraints and get the modified primal problem:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top H \mathbf{x} \tag{A.8}$$

subject to:

$$\begin{array}{rcll} A\mathbf{x} - \mathbf{w} & = & \mathbf{b} & \\ \mathbf{w}_1 + \mathbf{p} & = & \mathbf{r} & \\ (\mathbf{x}_1, \mathbf{x}_2)^\top - \mathbf{g} & = & \mathbf{l} & \\ \mathbf{x}_1 + \mathbf{t} & = & \mathbf{u} & \\ \mathbf{w}, \mathbf{p}, \mathbf{g}, \mathbf{t} & \geq & 0, \quad \mathbf{x} \text{ free} & \end{array} \tag{A.9}$$

where $\mathbf{l} = (\mathbf{l}_1, \mathbf{l}_2)^\top$. After some longish calculations the corresponding dual can be derived as:

$$\max_{\mathbf{y}} \quad \mathbf{b}^\top \mathbf{y} - \frac{1}{2}\mathbf{x}^\top H \mathbf{x} + \mathbf{l}^\top \mathbf{z} - \mathbf{u}^\top \mathbf{s} - \mathbf{r}^\top \mathbf{q} \tag{A.10}$$

subject to:

$$\begin{array}{rcrcrcrcl} A_{\bullet 1}^\top \mathbf{y} & + & \mathbf{z}_1 & - & \mathbf{s} & - & H_{1\bullet}\mathbf{x} & = & \mathbf{c}_1 \\ A_{\bullet 2}^\top \mathbf{y} & + & \mathbf{z}_2 & & & - & H_{2\bullet}\mathbf{x} & = & \mathbf{c}_2 \\ A_{\bullet 3}^\top \mathbf{y} & & & & & - & H_{3\bullet}\mathbf{x} & = & \mathbf{c}_3 \\ \mathbf{y}_1 & + & \mathbf{q} & - & \mathbf{v}_1 & & & = & 0 \\ \mathbf{y}_2 & & & - & \mathbf{v}_2 & & & = & 0 \\ & & & & & & \mathbf{v}, \mathbf{q}, \mathbf{z}, \mathbf{s} & \geq & 0, \quad \mathbf{y} \text{ free} \end{array} \tag{A.11}$$

where $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)^\top$, $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2)^\top$ and $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2)^\top$.

Now we know from the theory of mathematical programming that the optimal solution, if it exists, must be

- primal feasible,
- dual feasible,
- and fulfill the Karush-Kuhn Tucker complimentary condition.

The first two conditions are given by the primal and dual constraints. The third condition can be expressed as $\mathbf{g}_i^\top \mathbf{z}_i = 0$, $\mathbf{w}_i^\top \mathbf{v}_i = 0$, $\mathbf{p}^\top \mathbf{q} = 0$ and $\mathbf{s}^\top \mathbf{t} = 0$, for $i = 1, 2$. Introducing the notation

$$G = \begin{pmatrix} g_1 & 0 & \dots & 0 \\ 0 & g_2 & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & \dots & g_{N-1} & 0 \\ 0 & \dots & 0 & g_N \end{pmatrix}$$

i.e. the uppercase symbol denotes a diagonal matrix with elements specified by the

corresponding lowercase vector, these condition can be summarize as:

$$
\begin{aligned}
A_{11}\mathbf{x}_1 + A_{12}\mathbf{x}_2 + A_{13}\mathbf{x}_3 - \mathbf{w}_1 &= \mathbf{b}_1 \\
A_{21}\mathbf{x}_1 + A_{22}\mathbf{x}_2 + A_{23}\mathbf{x}_3 - \mathbf{w}_2 &= \mathbf{b}_2 \\
\mathbf{w}_1 + \mathbf{p} &= \mathbf{r} \\
\mathbf{x}_1 - \mathbf{g}_1 &= \mathbf{l}_1 \\
\mathbf{x}_2 - \mathbf{g}_2 &= \mathbf{l}_2 \\
\mathbf{x}_1 + \mathbf{t} &= \mathbf{u}
\end{aligned}
$$

$$
\begin{aligned}
A_{11}^\top\mathbf{y}_1 + A_{21}^\top\mathbf{y}_2 + \mathbf{z}_1 - \mathbf{s} - H_{11}\mathbf{x}_1 - H_{12}\mathbf{x}_2 - H_{13}\mathbf{x}_3 &= \mathbf{c}_1 \\
A_{12}^\top\mathbf{y}_1 + A_{22}^\top\mathbf{y}_2 + \mathbf{z}_2 - H_{21}\mathbf{x}_1 - H_{22}\mathbf{x}_2 - H_{23}\mathbf{x}_3 &= \mathbf{c}_2 \\
A_{13}^\top\mathbf{y}_1 + A_{23}^\top\mathbf{y}_2 - H_{31}\mathbf{x}_1 - H_{32}\mathbf{x}_2 - H_{33}\mathbf{x}_3 &= \mathbf{c}_3 \\
\mathbf{y}_1 + \mathbf{q} - \mathbf{v}_1 &= 0 \\
\mathbf{y}_2 - \mathbf{v}_2 &= 0
\end{aligned}
$$

$$
\begin{aligned}
G_1 Z_1 \mathbf{e} &= \mu\mathbf{e} \\
G_2 Z_2 \mathbf{e} &= \mu\mathbf{e} \\
PQ\mathbf{e} &= \mu\mathbf{e} \\
ST\mathbf{e} &= \mu\mathbf{e} \\
W_1 V_1 \mathbf{e} &= \mu\mathbf{e} \\
W_2 V_2 \mathbf{e} &= \mu\mathbf{e}
\end{aligned}
$$

Here we have already replaced the condition that the non-linear equations are zero by the relaxed condition that they are equal to some value $\mu \geq 0$. The rational behind this is, that using the predictor-corrector approach we can only make small steps without leaving the interior. However, we need some starting point for our iteration and requiring $\mu = 0$ from the beginning would hardly be possible and result in very small steps in the other variables (and also make the intermediate problems to be solved extremely ill-posed). Hence we will start with some positive value $\mu = \mu_0 > 0$ which is then suitably reduced while the iteration continues (see below).

To derive the predictor-corrector steps, we now introduce Δ-variables for the steps, i.e. we rewrite our system such, that we replace all variables (e.g. $\mathbf{x}_1$) by a the variable plus a step (e.g. $\mathbf{x}_1 + \Delta\mathbf{x}_1$). We then rearrange the system such, that

the we can solve it for the Δ-variables, yielding the steps. First define

$$
\begin{aligned}
\mathbf{b}_1 - A_{11}\mathbf{x}_1 - A_{12}\mathbf{x}_2 - A_{13}\mathbf{x}_3 + \mathbf{w}_1 &=: \rho_1 \\
\mathbf{b}_2 - A_{21}\mathbf{x}_1 - A_{22}\mathbf{x}_2 - A_{23}\mathbf{x}_3 + \mathbf{w}_2 &=: \rho_2 \\
\mathbf{r} - \mathbf{w}_1 - p &=: \alpha \\
\mathbf{l}_1 - \mathbf{x}_1 + \mathbf{g}_1 &=: \nu_1 \\
\mathbf{l}_2 - \mathbf{x}_2 + \mathbf{g}_2 &=: \nu_2 \\
\mathbf{u} - \mathbf{x}_1 - \mathbf{t} &=: \tau
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{c}_1 - A_{11}^{\top}\mathbf{y}_1 - A_{21}^{\top}\mathbf{y}_2 - \mathbf{z}_1 + \mathbf{s} + H_{11}\mathbf{x}_1 + H_{12}\mathbf{x}_2 + H_{13}\mathbf{x}_3 &=: \sigma_1 \\
\mathbf{c}_2 - A_{12}^{\top}\mathbf{y}_1 - A_{22}^{\top}\mathbf{y}_2 - \mathbf{z}_2 + H_{21}\mathbf{x}_1 + H_{22}\mathbf{x}_2 + H_{23}\mathbf{x}_3 &=: \sigma_2 \\
\mathbf{c}_3 - A_{13}^{\top}\mathbf{y}_1 - A_{23}^{\top}\mathbf{y}_2 + H_{31}\mathbf{x}_1 + H_{32}\mathbf{x}_2 + H_{33}\mathbf{x}_3 &=: \sigma_3 \\
\mathbf{y}_1 + \mathbf{q} - \mathbf{v}_1 &=: \beta_1 \\
\mathbf{y}_2 - \mathbf{v}_2 &=: \beta_2
\end{aligned}
$$

$$
\begin{aligned}
\mu G_1^{-1}\mathbf{e} - \mathbf{z}_1 - G_1^{-1}\Delta G_1 \Delta \mathbf{z}_1 &=: \gamma_{\mathbf{z}_1} \\
\mu G_2^{-1}\mathbf{e} - \mathbf{z}_2 - G_2^{-1}\Delta G_2 \Delta \mathbf{z}_2 &=: \gamma_{\mathbf{z}_2} \\
\mu P^{-1}\mathbf{e} - \mathbf{q} - P^{-1}\Delta P \Delta \mathbf{q} &=: \gamma_{\mathbf{q}} \\
\mu T^{-1}\mathbf{e} - \mathbf{s} - T^{-1}\Delta T \Delta \mathbf{s} &=: \gamma_{\mathbf{s}} \\
\mu V_1^{-1}\mathbf{e} - \mathbf{w}_1 - V_1^{-1}\Delta V_1 \Delta \mathbf{w}_1 &=: \gamma_{\mathbf{w}_1} \\
\mu V_2^{-1}\mathbf{e} - \mathbf{w}_2 - V_2^{-1}\Delta V_2 \Delta \mathbf{w}_2 &=: \gamma_{\mathbf{w}_2}.
\end{aligned}
$$

These quantities will become the new right-hand sides. Here we have already decided for a specific way to linearize the non-linear equations. The system now

becomes:

$$\begin{aligned}
A_{11}\Delta\mathbf{x}_1 + A_{12}\Delta\mathbf{x}_2 + A_{13}\Delta\mathbf{x}_3 - \Delta\mathbf{w}_1 &=: \rho_1\\
A_{21}\Delta\mathbf{x}_1 + A_{22}\Delta\mathbf{x}_2 + A_{23}\Delta\mathbf{x}_3 - \Delta\mathbf{w}_2 &=: \rho_2\\
\Delta\mathbf{w}_1 + \Delta\mathbf{p} &=: \alpha\\
\Delta\mathbf{x}_1 - \Delta\mathbf{g}_1 &=: \nu_1\\
\Delta\mathbf{x}_2 - \Delta\mathbf{g}_2 &=: \nu_2\\
\Delta\mathbf{x}_1 + \Delta\mathbf{t} &=: \tau
\end{aligned}$$

$$\begin{aligned}
A_{11}^\top\Delta\mathbf{y}_1 + A_{21}^\top\Delta\mathbf{y}_2 + \Delta\mathbf{z}_1 - \Delta\mathbf{s} - H_{11}\Delta\mathbf{x}_1 - H_{12}\Delta\mathbf{x}_2 - H_{13}\Delta\mathbf{x}_3 &=: \sigma_1\\
A_{12}^\top\Delta\mathbf{y}_1 + A_{22}^\top\Delta\mathbf{y}_2 + \Delta\mathbf{z}_2 - H_{21}\Delta\mathbf{x}_1 - H_{22}\Delta\mathbf{x}_2 - H_{23}\Delta\mathbf{x}_3 &=: \sigma_2\\
A_{13}^\top\Delta\mathbf{y}_1 + A_{23}^\top\Delta\mathbf{y}_2 - H_{31}\Delta\mathbf{x}_1 - H_{32}\Delta\mathbf{x}_2 - H_{33}\Delta\mathbf{x}_3 &=: \sigma_3\\
-\Delta\mathbf{y}_1 - \Delta\mathbf{q} + \Delta\mathbf{v} &=: \beta
\end{aligned}$$

$$\begin{aligned}
G_1^{-1}Z_1\Delta\mathbf{g}_1 + \Delta\mathbf{z}_1 &=: \gamma_{z_1}\\
G_2^{-1}Z_2\Delta\mathbf{g}_2 + \Delta\mathbf{z}_2 &=: \gamma_{z_2}\\
P^{-1}Q\Delta\mathbf{p} + \Delta\mathbf{q} &=: \gamma_q\\
ST^{-1}\Delta\mathbf{t} + \Delta\mathbf{s} &=: \gamma_s\\
V_1^{-1}W_1\Delta\mathbf{v}_1 + \Delta\mathbf{w}_1 &=: \gamma_{w_1}\\
V_2^{-1}W_2\Delta\mathbf{v}_2 + \Delta\mathbf{w}_2 &=: \gamma_{w_2},
\end{aligned}$$

where in the last six equations we set the values of the corresponding diagonal matrices to the current value of these variables in the predictor step and to the value found in this step in the corrector step.

What remains is to solve this huge system of equations twice in each iteration. Luckily, the system is very sparse, compared to its size (see next equation). The approach taken in (Vanderbei and Shanno, 1997) is to use some predefined pivoting to reduce this system until any further reduction will lead to a uncontrollable fill-in of non-zero elements. We again follow (Vanderbei and Shanno, 1997) in which order to solve for the variables. Do do so, we first rewrite the problem in one big matrix (cf. system on page 122). We see that we can trivially (i.e. with low computational effort) solve this system for $\Delta\mathbf{t}$, $\Delta\mathbf{g}_1$, $\Delta\mathbf{g}_2$, $\Delta\mathbf{p}$, $\Delta\mathbf{v}_1$ and $\Delta\mathbf{v}_2$:

$$\Delta\mathbf{t} = S^{-1}T(\gamma_s - \Delta\mathbf{s}), \tag{A.12}$$
$$\Delta\mathbf{g}_1 = G_1Z_1^{-1}(\gamma_{z_1} - \Delta\mathbf{z}_1), \tag{A.13}$$
$$\Delta\mathbf{g}_2 = G_2Z_2^{-1}(\gamma_{z_2} - \Delta\mathbf{z}_2), \tag{A.14}$$
$$\Delta\mathbf{p} = PQ^{-1}(\gamma_q - \Delta\mathbf{q}), \tag{A.15}$$
$$\Delta\mathbf{v}_1 = V_1W_1^{-1}(\gamma_{w_1} - \Delta\mathbf{w}_1), \tag{A.16}$$
$$\Delta\mathbf{v}_2 = V_2W_2^{-1}(\gamma_{w_2} - \Delta\mathbf{w}_2). \tag{A.17}$$

Resubstituting this solutions in the equations on page 122 we get the new, smaller system shown on page 123. Again, we can still easily solve this system for the

$$
\begin{bmatrix}
-ST^{-1} & & & & & & & & & & & & & & -I & & \\
 & -G_1^{-1}Z_1 & & & & & & & & & & & -I & & & & \\
 & & -G_2^{-1}Z_2 & & & & & & & & & & & -I & & & \\
 & & & -P^{-1}Q & & & & & & & & -I & & & & & \\
 & & & & & & & & & -I & & -I & & & & I & \\
 & & & & & & & & & & -I & & & & & & I \\
 & & & & & & -H_{11} & -H_{12} & -H_{13} & A_{11}^\top & A_{21}^\top & & I & & -I & & \\
 & & & & & & -H_{21} & -H_{22} & -H_{23} & A_{12}^\top & A_{22}^\top & & & I & & & \\
 & & & & & & -H_{31} & -H_{32} & -H_{33} & A_{13}^\top & A_{23}^\top & & & & & & \\
\hline
 & & & & -I & & A_{11} & A_{12} & A_{13} & & & & & & & & \\
 & & & & & -I & A_{21} & A_{22} & A_{23} & & & & & & & & \\
 & & & -I & -I & & & & & & & & & & & & \\
 & -I & & & & & I & & & & & & & & & & \\
 & & -I & & & & & I & & & & & & & & & \\
-I & & & & & & -I & & & & & & & & & & \\
 & & & & I & & & & & & & & & & & V_1^{-1}W_1 & \\
 & & & & & I & & & & & & & & & & & V_2^{-1}W_2
\end{bmatrix}
\begin{bmatrix}
\Delta \mathbf{t} \\ \Delta \mathbf{g}_1 \\ \Delta \mathbf{g}_2 \\ \Delta \mathbf{p} \\ \Delta \mathbf{w}_1 \\ \Delta \mathbf{w}_2 \\ \Delta \mathbf{x}_1 \\ \Delta \mathbf{x}_2 \\ \Delta \mathbf{x}_3 \\ \Delta \mathbf{y}_1 \\ \Delta \mathbf{y}_2 \\ \Delta \mathbf{q} \\ \Delta \mathbf{z}_1 \\ \Delta \mathbf{z}_2 \\ \Delta \mathbf{s} \\ \Delta \mathbf{v}_1 \\ \Delta \mathbf{v}_2
\end{bmatrix}
=
\begin{bmatrix}
-\gamma_{\mathrm{s}} \\ -\gamma_{\mathrm{z}_1} \\ -\gamma_{\mathrm{z}_2} \\ -\gamma_{\mathrm{q}} \\ \beta_1 \\ \beta_2 \\ \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \rho_1 \\ \rho_2 \\ -\alpha \\ \nu_1 \\ \nu_2 \\ -\tau \\ \gamma_{\mathrm{w}_1} \\ \gamma_{\mathrm{w}_2}
\end{bmatrix}
$$

$$
\left[\begin{array}{cccccccccccccc|c}
-V_1W_1^{-1} & & & & & -I & & & & & & & & & \Delta\mathbf{w}_1 \\
& -V_2W_2^{-1} & & & & & & & & & & & & -I & \Delta\mathbf{w}_2 \\
& & -H_{11} & -H_{12} & -H_{13} & A_{11}^T & A_{21}^T & & & & & & & & \Delta\mathbf{x}_1 \\
& & -H_{21} & -H_{22} & -H_{23} & A_{12}^T & A_{22}^T & & & I & & & & & \Delta\mathbf{x}_2 \\
& & -H_{31} & -H_{32} & -H_{33} & A_{13}^T & A_{23}^T & & & & & & & & \Delta\mathbf{x}_3 \\
\hline
-I & & A_{11} & A_{12} & A_{13} & & & & & & & & & & \Delta\mathbf{y}_1 \\
& -I & A_{21} & A_{22} & A_{23} & & & & & & & & & & \Delta\mathbf{y}_2 \\
& & & & & & & PQ^{-1} & & & & & & & \Delta\mathbf{q} \\
-I & & & I & & & & & G_1Z_1^{-1} & & & & & & \Delta\mathbf{z}_1 \\
& & & & I & & & & & G_2^{-1}Z_2 & & & & & \Delta\mathbf{z}_2 \\
& & & & & -I & & & & & & & & S^{-1}T & \Delta\mathbf{s}
\end{array}\right]
=
\left[\begin{array}{c}
\beta_1 - V_1W_1^{-1}\gamma_{w_1} =: \hat{\beta}_1 \\
\beta_2 - V_2W_2^{-1}\gamma_{w_2} =: \hat{\beta}_2 \\
\sigma_1 \\
\sigma_2 \\
\sigma_3 \\
\rho_1 \\
\rho_2 \\
-\alpha + PQ^{-1}\gamma_q =: -\hat{\alpha} \\
\nu_1 + G_1Z_1^{-1}\gamma_{z_1} =: \hat{\nu}_1 \\
\nu_2 + G_2Z_2^{-1}\gamma_{z_2} =: \hat{\nu}_2 \\
-\tau + S^{-1}T\gamma_s =: -\hat{\tau}
\end{array}\right]
$$

variables $\Delta\mathbf{q}$, $\Delta\mathbf{z}_1$, $\Delta\mathbf{z}_2$ and $\Delta\mathbf{s}$ by:

$$\Delta\mathbf{q} = P^{-1}Q(\Delta\mathbf{w}_1 - \hat{\alpha}), \tag{A.18}$$
$$\Delta\mathbf{z}_1 = G_1^{-1}Z_1(\hat{\nu}_1 - \Delta\mathbf{x}_1), \tag{A.19}$$
$$\Delta\mathbf{z}_2 = G_2^{-1}Z_2(\hat{\nu}_2 - \Delta\mathbf{x}_2), \tag{A.20}$$
$$\Delta\mathbf{s} = ST^{-1}(\Delta\mathbf{x}_1 - \hat{\tau}). \tag{A.21}$$

Defining the diagonal matrices

$$E_1 = (V_1W_1^{-1} + P^{-1}Q)^{-1}, \tag{A.22}$$
$$E_2 = (V_2W_2^{-1})^{-1}, \tag{A.23}$$
$$D_1 = ST^{-1} + G_1^{-1}Z_1, \tag{A.24}$$
$$D_2 = G_2^{-1}Z_2, \tag{A.25}$$

we can rewrite the system on page 123 as:

$$\begin{bmatrix} -E_1^{-1} & & & & & -I & \\ & -E_2^{-1} & & & & & -I \\ & & -H_{11}-D_1 & -H_{12} & -H_{13} & A_{11}^\top & A_{21}^\top \\ & & -H_{21} & -H_{22}-D_2 & -H_{23} & A_{12}^\top & A_{22}^\top \\ & & -H_{31} & -H_{32} & -H_{33} & A_{13}^\top & A_{23}^\top \\ -I & & A_{11} & A_{12} & A_{13} & & \\ & -I & A_{21} & A_{22} & A_{23} & & \end{bmatrix} \begin{bmatrix} \Delta\mathbf{w}_1 \\ \Delta\mathbf{w}_2 \\ \Delta\mathbf{x}_1 \\ \Delta\mathbf{x}_2 \\ \Delta\mathbf{x}_3 \\ \Delta\mathbf{y}_1 \\ \Delta\mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \hat{\beta}_1 - P^{-1}Q\hat{\alpha} \\ \hat{\beta}_2 \\ \sigma_1 - G_1^{-1}Z_1\hat{\nu}_1 - ST^{-1}\hat{\tau} \\ \sigma_2 - G_2^{-1}Z_2\hat{\nu}_2 \\ \sigma_3 \\ \rho_1 \\ \rho_2 \end{bmatrix}.$$

Finally, we solve for $\Delta\mathbf{w}_1$ and $\Delta\mathbf{w}_2$ through

$$\Delta\mathbf{w}_1 = -E_1(\hat{\beta}_1 - P^{-1}Q\hat{\alpha} + \Delta\mathbf{y}_1), \tag{A.26}$$
$$\Delta\mathbf{w}_2 = -E_2(\hat{\beta}_2 + \Delta\mathbf{y}_2), \tag{A.27}$$

arriving at the final system

$$\begin{bmatrix} -H_{11}-D_1 & -H_{12} & -H_{13} & A_{11}^\top & A_{21}^\top \\ -H_{21} & -H_{22}-D_2 & -H_{23} & A_{12}^\top & A_{22}^\top \\ -H_{31} & -H_{32} & -H_{33} & A_{13}^\top & A_{23}^\top \\ A_{11} & A_{12} & A_{13} & E_1 & \\ A_{21} & A_{22} & A_{23} & & E_2 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}_1 \\ \Delta\mathbf{x}_2 \\ \Delta\mathbf{x}_3 \\ \Delta\mathbf{y}_1 \\ \Delta\mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \sigma_1 - G_1^{-1}Z_1\hat{\nu}_1 - ST^{-1}\hat{\tau} \\ \sigma_2 - G_2^{-1}Z_2\hat{\nu}_2 \\ \sigma_3 \\ \rho_1 - E_1(\hat{\beta}_1 - P^{-1}Q\hat{\alpha}) \\ \rho_2 - E_2\hat{\beta}_2 \end{bmatrix}. \tag{A.28}$$

At this point, at least if do not make any special assumptions about the structure of H and A no further trivial reduction is possible. However, the advantage of deriving an individual interior point code for a specific problem like e.g. KFD (cf. Section 3.5.5) is, that there might be further reduction possible, making the last system even smaller. If finally this system can not be reduced any further it has to be solved using standard numerical methods. Updates are done such, that we maintain positivity of the constraint variables (cf. Vanderbei, 1997, for details). It is worthwhile to note, that the standard mathematical programming literature usually assumes that both, H and A are very sparse, structured matrices. Then solving the final system can be very efficient by using a clever pivoting technique. However, in kernel based learning we usually have that either H or A contain the complete kernel matrix K which, usually, is not sparse. Recently Achlioptas et al. (2002) proposed to replace the dense kernel matrix by a sparse approximation. This would clearly be an alternative here. Also recently, there has been some interest in low-rank approximations of the kernel matrix or its inverse using incomplete Cholesky factorizations (cf. Smola and Schölkopf, 2000; Bach and Jordan, 2002). For specific kernel problems this might also yield good approximate solutions here. Some authors also proposed to use iterative methods to solve this last system which could e.g. then be parallelized over a cluster of computers (e.g. Freund and Nachtigal, 1994; Freund and Jarre, 1996). The system to solve is a symmetric, indefinite system. Hence many iterative techniques like QMR, SYMMLQ or GMRES (cf. Golub and van Loan, 1996, and references there in), to name only a few, are possible. The central problem however is, that this system will become more and more ill-conditioned as we approach the final solution, thus requiring more and more iterations.

Starting, Stopping and Inbetween

We will not discuss the issue of how to initialize the variables, how to monitor the stopping conditions and how to update the factor μ in great detail here. They all play a crucial role in successfully implementing an interior point code, e.g. one that converges fast and reliably. The latter is indeed very difficult to achieve for any mathematical optimization technique. Implementations that are numerically stable and have the suitable "tricks" built in are very difficult to program and one reason why commercial packages like CPLEX (CPL, 1994) are rather expensive.

Initialization is done by solving the reduced system once for D and E being the identity matrix and the right-hand site suitably adjusted. Then a value for the other variables is computed and appropriately adjusted away from zero for the non-negative variables.

Termination is rather trivial: We monitor the primal and dual infeasibility and the duality gap. If all are below predefined thresholds we know that we are very close to the optimal solution and terminate.

Dealing with the different situations during the optimization is much more difficult. First we have to adjust the value of μ. Using the current variable assignment it can be computed as

$$\mu = \frac{\mathbf{g}^\top \mathbf{z} + \mathbf{p}^\top \mathbf{q} + \mathbf{s}^\top \mathbf{t} + \mathbf{w}^\top \mathbf{v}}{2M_1 + M_2 + 2N_1 + N_2}$$

Since we wish to make progress to the optimal solution Vanderbei (1997) suggest to use a value of μ that is one tenth of that we compute. Furthermore it is suggested to reduce μ even more if we can make large steps, i.e. do not have to shorten the step directions much to maintain positivity. More details on how to implement the particular interior point strategy presented here can be found on (Vanderbei, 1997). There is also a free implementation available on the web page of R. Vanderbei. However, this is also a general purpose implementation and can not take into account the special structure of e.g. KFD problem.

Interior Point Codes for KFD

We now use this notation to derive interior point codes especially adopted to KFD, sparse KFD and linear sparse KFD.

KFD For the plain KFD approach (cf. (3.37)) with either $\|\mathbf{w}\|^2$ or $\|\mathbf{ff}\|^2$ regularizer we define

$$\mathbf{x}_3 = \begin{bmatrix} \mathbf{ff} \\ , \\ b \end{bmatrix}, H_{33} = \begin{bmatrix} CP & & \\ & I & \\ & & 0 \end{bmatrix}, \mathbf{r} = 0, \mathbf{c} = 0, A_{13} = \begin{bmatrix} K & I & \mathbf{1} \end{bmatrix}, \mathbf{b}_1 = \mathbf{y}.$$

where P is either K or I, depending on which regularizer we choose. All other parts of the general QP (A.6) are empty. This problem is exactly the KFD problem. Substituting these quantities into the reduced KKT (A.28) we get

$$\begin{bmatrix} -CP - D_{31} & & & K^\top \\ & -I - D_{32} & & I \\ & & -D_{33} & \mathbf{1}^\top \\ K & I & \mathbf{1} & E_1 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{ff} \\ \Delta_, \\ \Delta\mathbf{b}_1 \\ \Delta\mathbf{y}_1 \end{bmatrix} = \begin{bmatrix} \text{some right hand side} \end{bmatrix},$$

where E_1 and D_{31}, D_{32}, D_{33} are some diagonal matrices. This system can be further reduced by pivoting on $\Delta_,$, yielding:

$$\begin{bmatrix} -CP - D_{31} & & K^\top \\ & -D_{33} & \mathbf{1}^\top \\ K & \mathbf{1} & E_1 + (I + D_{32})^{-1} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{ff} \\ \Delta\mathbf{b}_1 \\ \Delta\mathbf{y}_1 \end{bmatrix} = \begin{bmatrix} \text{some other right hand side} \end{bmatrix}.$$

For KFD with $P = K$ the only possible pivot left would be $\Delta\mathbf{y}_1$. However, doing so would result in a very complex system containing terms of the form $K^\top DK$ where D is some diagonal matrix. For small M and hence small K this might make sense but we do not carry out this step explicitly. For the $P = I$ variant however, we can still easily pivot in $\Delta\mathbf{ff}$, yielding

$$\begin{bmatrix} 0 & \mathbf{1}^\top \\ \mathbf{1} & \frac{1}{C}K(-CI + D_{31})^{-1}K^\top + E_1 + I \end{bmatrix} \begin{bmatrix} \Delta\mathbf{b}_1 \\ \Delta\mathbf{y}_1 \end{bmatrix} = \begin{bmatrix} \text{some other right hand side} \end{bmatrix}.$$

It is interesting to note how this system resembles the system of linear equalities presented in (3.64) that could be solved alternatively.

Sparse KFD, Linear Sparse KFD We will not write down the reduced KKT systems for these approaches explicitly but only describe how they can be cast into the general formulation (A.6). The derivation of the reduced KKT system and its possible further reduction is straight forward.

To represent the sparse KFD problem (cf. (3.53)) in terms of the general QP (A.6) we define:

$$\mathbf{x}_2 = \begin{bmatrix} \mathbf{ff}_+ \\ \mathbf{ff}_- \end{bmatrix}, \qquad \mathbf{x}_3 = \begin{bmatrix} \text{,} \\ b \end{bmatrix}, \qquad \mathbf{c} = \begin{bmatrix} C\mathbf{1} \\ C\mathbf{1} \end{bmatrix},$$

$$H_{33} = \begin{bmatrix} I & \\ & 0 \end{bmatrix}, \qquad A_{12} = \begin{bmatrix} K & -K \end{bmatrix}, \qquad A_{13} = \begin{bmatrix} I & \mathbf{1} \end{bmatrix},$$

$$\mathbf{r} = 0, \qquad \mathbf{b}_1 = \mathbf{y}, \qquad \mathbf{l}_2 = 0.$$

Again, all unspecified quantities are empty or all zero. For linear sparse KFD (LSKFD) we get

$$\mathbf{x}_2 = \begin{bmatrix} \mathbf{ff}_+ \\ \mathbf{ff}_- \\ \text{,}_+ \\ \text{,}_- \end{bmatrix}, \qquad \mathbf{x}_3 = \begin{bmatrix} b \end{bmatrix}, \qquad \mathbf{c} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ C\mathbf{1} \\ C\mathbf{1} \end{bmatrix},$$

$$A_{12} = \begin{bmatrix} K & -K & I & -I \end{bmatrix}, \qquad A_{13} = \begin{bmatrix} \mathbf{1} \end{bmatrix},$$

$$\mathbf{r} = 0, \qquad \mathbf{b}_1 = \mathbf{y}, \qquad \mathbf{l}_2 = 0.$$

For all variants of KFD we notice that deriving a specialized reduced KKT system for the interior point optimizer actually makes sense: Compared to the general reduced KKT system (A.28) substantial further reductions are possible making it easier and faster to solve the reduced KKT system.

Appendix B

Proofs

This appendix contains some proofs from Chapter 4.

B.1 Proof of Lemma 4.2

Proof. To begin with, let us rewrite the covariance matrix (4.1) in a different, equivalent way. We have

$$\begin{aligned} C_{\mathcal{X}} &= \frac{1}{M-1}\sum_{x\in\mathcal{X}}(\mathbf{x}-\mathbf{m}_{\mathcal{X}})(\mathbf{x}-\mathbf{m}_{\mathcal{X}})^{\top} \qquad \text{(B.1)}\\ &= \frac{1}{M-1}\sum_{x\in\mathcal{X}}\left(\mathbf{x}\mathbf{x}^{\top}-\mathbf{x}\mathbf{m}_{\mathcal{X}}^{\top}-\mathbf{m}_{\mathcal{X}}^{\top}\mathbf{x}+\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^{\top}\right)\\ &= \frac{1}{M-1}\sum_{x\in\mathcal{X}}\mathbf{x}\mathbf{x}^{\top}+\frac{M}{M-1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^{\top}-\frac{M}{M-1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^{\top}-\frac{M}{M-1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^{\top}\\ &= \frac{1}{M-1}\sum_{x\in\mathcal{X}}\mathbf{x}\mathbf{x}^{\top}-\frac{M}{M-1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^{\top}. \end{aligned}$$

Lets us apply this to the definition of $C_{\mathcal{X}\cup x}$:

$$C_{\mathcal{X}\cup x} := \frac{1}{M}\sum_{x\in\mathcal{X}\cup x}\mathbf{x}\mathbf{x}^{\top}-\frac{M+1}{M}\mathbf{m}_{\mathcal{X}\cup x}\mathbf{m}_{\mathcal{X}\cup x}^{\top}.$$

First we split this expression into parts containing only old examples $\mathbf{x}\in\mathcal{X}$ and the new example (with a slight abuse of notation also denoted by $\mathbf{x}$).

$$C_{\mathcal{X}\cup x}=\frac{1}{M}\sum_{x\in\mathcal{X}}\mathbf{x}\mathbf{x}^{\top}+\frac{1}{M}\mathbf{x}\mathbf{x}^{\top}-\frac{1}{M(M+1)}(M^2\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^{\top}+\mathbf{x}\mathbf{x}^{\top}+M\mathbf{m}_{\mathcal{X}}\mathbf{x}+M\mathbf{m}_{\mathcal{X}}^{\top}),$$

where we used that $\sum_{x\in\mathcal{X}\cup x}\mathbf{x}=M\mathbf{m}_{\mathcal{X}}+\mathbf{x}$. Now we expand all terms and use the relation $\frac{1}{M}-\frac{1}{M(M+1)}=\frac{1}{(M+1)}$ to collect the $\mathbf{x}\mathbf{x}^{\top}$ terms. Furthermore, we split the

term $\frac{M}{M+1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^\top$ we would get into the sum of $\frac{1}{M+1}$ and $\frac{M-1}{M+1}$.

$$C_{\mathcal{X}\cup\mathsf{x}} = \frac{1}{M}\sum_{\mathsf{x}\in\mathcal{X}}\mathbf{x}\mathbf{x}^\top - \frac{M-1}{M+1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^\top - \frac{1}{M+1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^\top + \frac{1}{M+1}\mathbf{x}\mathbf{x}^\top - \frac{1}{M+1}\mathbf{m}_{\mathcal{X}}\mathbf{x}^\top - \frac{1}{M+1}\mathbf{x}\mathbf{m}_{\mathcal{X}}^\top.$$

If we multiply the complete expression by $\frac{M-1}{M}$ we have to multiply each term in the expression by the inverse, i.e. $\frac{M}{M-1}$, and get

$$C_{\mathcal{X}\cup\mathsf{x}} = \frac{M}{M-1}\left(\frac{1}{M-1}\sum_{\mathsf{x}\in\mathcal{X}}\mathbf{x}\mathbf{x}^\top - \frac{M^2}{M^2-1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^\top + \frac{M}{M^2-1}\mathbf{x}\mathbf{x}^\top - \frac{M}{M^2-1}\mathbf{m}_{\mathcal{X}}\mathbf{x}^\top - \frac{M}{M^2-1}\mathbf{x}\mathbf{m}_{\mathcal{X}}^\top\right),$$

Noting that $\frac{M^2}{M^2-1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^\top = -\frac{M}{M-1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^\top + \frac{M}{M^2-1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^\top$ we get

$$C_{\mathcal{X}\cup\mathsf{x}} = \frac{M}{M-1}\left(\frac{1}{M-1}\sum_{\mathsf{x}\in\mathcal{X}}\mathbf{x}\mathbf{x}^\top - \frac{M}{M-1}\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^\top + \frac{M}{M^2-1}\left[\mathbf{m}_{\mathcal{X}}\mathbf{m}_{\mathcal{X}}^\top + \mathbf{x}\mathbf{x}^\top - \mathbf{m}_{\mathcal{X}}\mathbf{x}^\top - \mathbf{x}\mathbf{m}_{\mathcal{X}}^\top\right]\right).$$

Using (B.2) and collecting all terms with $\frac{M}{M^2-1}$ in front into a quadratic expression proofs the first statement of the Lemma.

The proof for the second part, i.e. for the update rule of $C_{\mathcal{X}\setminus\mathsf{x}}$ follows along the same lines. Finally to show what happens if we exchange one example is simply an application of what we just proved for the removal and the addition on an example, i.e.

$$\begin{aligned}
C_{(\mathcal{X}\setminus\mathsf{x})\cup\mathsf{x}} &= \frac{M-2}{M-1}\left[C_{\mathcal{X}\setminus\mathsf{x}} + \frac{M-1}{(M-1)^2-1}\left(\mathbf{m}_{\mathcal{X}\setminus\mathsf{x}} - \mathbf{x}\right)\left(\mathbf{m}_{\mathcal{X}\setminus\mathsf{x}} - \mathbf{x}\right)^\top\right] \\
&= \frac{M-2}{M-1}\left[\frac{M-1}{M-2}\left[C_{\mathcal{X}} - \frac{M}{(M-1)^2}\left(\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i\right)\left(\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i\right)^\top\right]\right. \\
&\qquad \left. + \frac{M-1}{(M-1)^2-1}\left(\mathbf{m}_{\mathcal{X}\setminus\mathsf{x}} - \mathbf{x}\right)\left(\mathbf{m}_{\mathcal{X}\setminus\mathsf{x}} - \mathbf{x}\right)^\top\right] \\
&= C_{\mathcal{X}} - \frac{M}{(M-1)^2}\left(\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i\right)\left(\mathbf{m}_{\mathcal{X}} - \mathbf{x}_i\right)^\top \\
&\qquad + \frac{1}{M}\left(\mathbf{m}_{\mathcal{X}\setminus\mathsf{x}_i} - \mathbf{x}\right)\left(\mathbf{m}_{\mathcal{X}\setminus\mathsf{x}_i} - \mathbf{x}\right)^\top.
\end{aligned}$$

□

B.2 Proof of Lemma 4.9

Proof. Here we reproduce in detail the proof given in Bartlett and Mendelson (2002). Let $\epsilon_1, \ldots, \epsilon_M$ be independent Rademacher random variables. Then from

the Definition (4.17) for $\widehat{\mathcal{R}}_M$ we get:

$$
\begin{aligned}
\widehat{\mathcal{R}}_M(\mathcal{G}) &= \mathbb{E}_\epsilon \left[\frac{2}{M} \sup_{g\in\mathcal{G}} \left| \sum_{i=1}^{M} \epsilon_i g(X_i) \right| |X_i \right] && \text{(definition of } \widehat{\mathcal{R}}_M\text{)} \\
&= \mathbb{E}_\epsilon \left[\frac{2}{M} \sup_{\|\mathbf{u}\|\le R} \left| \sum_{i=1}^{M} \epsilon_i \langle \mathbf{u}, X_i \rangle \right| |X_i \right] && \text{(definition of } \mathcal{G}\text{)} \\
&= \mathbb{E}_\epsilon \left[\frac{2}{M} \sup_{\|\mathbf{u}\|\le R} \left| \langle \mathbf{u}, \sum_{i=1}^{M} \epsilon_i X_i \rangle \right| |X_i \right] && \text{(linearity of scalar product)} \\
&\le \mathbb{E}_\epsilon \left[\frac{2}{M} \left| \frac{\langle \sum_{i=1}^{M} \epsilon_i X_i, \sum_{i=1}^{M} \epsilon_i X_i \rangle}{R \| \sum_{i=1}^{M} \epsilon_i X_i \|} \right| |X_i \right] && \text{(worst case is u} = \textstyle\sum(\dots)/(R\|\sum(\dots)\|\text{)} \\
&= \frac{2}{RM} \mathbb{E}_\epsilon \left[\left(\sum_{i,j=1}^{M} \epsilon_i \epsilon_j \langle X_i, X_j \rangle \right)^{\frac{1}{2}} |X_i \right] && \text{(def. of norm and scalar products)} \\
&\le \frac{2}{RM} \left(\sum_{i,j=1}^{M} \mathbb{E}_\epsilon \left[\epsilon_i \epsilon_j \langle X_i, X_j \rangle |X_i \right] \right)^{\frac{1}{2}} && \text{(Jensen's inequality)} \\
&= \frac{2}{RM} \left(\sum_{i=1}^{M} \mathbb{E}_\epsilon \left[\epsilon_i^2 \langle X_i, X_i \rangle |X_i \right] \right)^{\frac{1}{2}} && \text{(} \mathbb{E}_\epsilon[\epsilon_i \epsilon_j] = 0\text{)} \\
&= \frac{2}{RM} \left(\sum_{i=1}^{M} \langle X_i, X_i \rangle \right)^{\frac{1}{2}} && \text{(} \mathbb{E}_\epsilon[\epsilon_i^2] = 1\text{)} \quad \text{(B.2)} \\
&\le \frac{2}{\sqrt{M}}.
\end{aligned}
$$

The chain of inequalities up to Equation B.2 shows the claim. □

B.3 Proof of Lemma 4.10

Proof. We want to show that $\|E\|_2$ is strongly concentrated around its expectation. We do this by showing that $\|E\|_2$ is stable in the sense of Definition 4.2. Then applying McDiarmid's inequality (cf. Theorem 4.1) yields the result. We will use that $f(\mathbf{x}) \le R^2$ and Equation (4.19).

$$
\begin{aligned}
\max_{f\in\mathcal{F}} \left| \mathbb{E}_{\mathcal{Y}}[f(Y)] - \frac{1}{M}\sum_{i=1}^{M} f(x_i) + \frac{1}{M} f(x_i) - \frac{1}{M} f(x_i') \right| & \\
&\le \max_{f\in\mathcal{F}} \left(\left| \mathbb{E}_{\mathcal{Y}}[f(Y)] - \frac{1}{M}\sum_{i=1}^{M} f(x_i) \right| + \frac{1}{M} |f(x_i) - f(x_i')| \right) \\
&\le \|E\|_2 + \frac{1}{M} \max_{f\in\mathcal{F}} |f(x_i) - f(x_i')| \\
&\le \|E\|_2 + \frac{2R^2}{M}.
\end{aligned}
$$

Similarly:

$$\begin{aligned}
&\max_{f\in\mathcal{F}}\left|\mathbb{E}_{\mathcal{Y}}[f(Y)]-\frac{1}{M}\sum_{i=1}^{M}f(x_i)+\frac{1}{M}f(x_i)-\frac{1}{M}f(x_i')\right| \\
&\qquad\geq \max_{f\in\mathcal{F}}\left(\left|\mathbb{E}_{\mathcal{Y}}[f(Y)]-\frac{1}{M}\sum_{i=1}^{M}f(x_i)\right|-\frac{1}{M}\left|f(x_i)-f(x_i')\right|\right) \\
&\qquad\geq \|E\|_2-\frac{1}{M}\max_{f\in\mathcal{F}}\left|f(x_i)-f(x_i')\right| \\
&\qquad\geq \|E\|_2-\frac{2R^2}{M}.
\end{aligned}$$

Hence we can apply McDiarmid with $c=c_i=\frac{2R^2}{M}$ for all $i=1,\ldots,M$. □

Bibliography

D. Achlioptas, F. McSherry, and B. Schölkopf. Sampling techniques for kernel methods. In T.G. Diettrich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Proccessing Systems*, volume 14, 2002.

M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale–sensitive Dimensions, Uniform Convergence, and Learnability. *Journal of the ACM*, 44(4): 615–631, 1997.

J.A. Anderson. Logistic discrimination. In P.R. Kirshnaiah and L.N. Kanal, editors, *Classification, Pattern Recognition and Reduction of Dimensionality*, volume 2 of *Handbook of Statistics*, pages 169–191. North Holland, Amsterdam, 1982.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

F.R. Bach and M.I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

P. Bartlett, O. Bousquet, and S. Mendelson. Localized rademacher complexities. In J. Kivinen and R.H. Sloan, editors, *Proceedings COLT*, volume 2375 of *Lecture Notes in Computer Science*, pages 44–58. Springer, 2002.

P.L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 2002. to appear.

G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

K.P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In P. Langley, editor, *Proceedings, 17th ICML*, pages 65–72, San Francisco, 2000. Morgan Kaufmann.

K.P. Bennett and O.L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

R. Beran and M.S. Srivastava. Bootstrap tests and confidence regions for functions of a covariance matrix. *Annals of Statistics*, 13(1):95–115, 1985.

D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

S.A. Billings and K.L Lee. Nonlinear Fisher discriminant analysis using a minimum squared error cost function and the orthogonal least squares algorithm. *Neural Networks*, 15(2):263–270, 2002.

C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html, a huge collection of artificial and real-world data sets.

B. Blankertz, G. Curio, and K-R. Müller. Classifying single trial EEG: Towards brain computer interfacing. In T.G. Diettrich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Proccessing Systems*, volume 14, 2002.

B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, March 2002.

B. Bradshaw, B. Schölkopf, and J. Platt. Kernel methods for extracting local image semantics. unpublished manuscript, private communication, 2000.

L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, July 1997.

C.J.C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 375–381, Cambridge, MA, 1997. MIT Press.

C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20: 273–297, 1995.

Using the CPLEX Callable Library. CPLEX Optimization Incorporated, Incline Village, Nevada, 1994.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

G.B. Dantzig. *Linear Programming and Extensions*. Princeton Univ. Press, Princeton, NJ, 1962.

A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Journal of Machine Learning Research*, 2001. To appear in special issue on Support Vector Machines and Kernel Methods.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Number 31 in Applications of Mathematics. Springer, New York, 1996.

K.I. Diamantaras and S.Y. Kung. *Principal Component Neural Networks*. Wiley, New York, 1996.

M. Doljansky and M. Teboulle. An interior proximal algorithm and the exponential multiplier method for semidefinite programming. *SIAM J. Optim.*, 9(1):1–13, 1998.

R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. John Wiley & Sons, 1973.

B. Efron and R.J. Tibshirani. Improvements on cross-validation: the .632+ bootstrap method. *J. Amer. Statist. Assoc*, 92:548–560, 1997.

S.C. Eisenstat and I.C.F. Ipsen. Three absolute pertubation bounds for matrix eigenvalues imply relative bounds. *SIAM Journal on Matrix Analysis and Applications*, 20(1):149–158, 1998.

M. Elad, Y. Hel-Or, and R. Keshet. Pattern detection using a maximal rejection classifier. *Pattern Recognition Letters*, 23(12):1459–1471, 2002.

R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

R.W. Freund and F.A. Jarre. A qmr-based interior-point algorithm for solving linear programs. *Math. Programming, Series B*, 1996.

R.W. Freund and N.M. Nachtigal. A new krylov-subspace method for symmetric indefinite linear systems. In *Proceedings of the 14th IMACS World Congress*, 1994.

Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55 (1):119–139, 1997.

J. Friedman, T. Hastie, and R.J. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Department of Statistics, Sequoia Hall, Stanford Univerity, July 1998.

J.H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, 2nd edition, 1990.

U. Garczarek. *Classification Rules in Standardized Partition Spaces*. PhD thesis, Universität Dortmund, Department of Statistics, 2002.

T.v. Gestel, J.A.K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vanderwalle. Bayesian framework for least squares support vector machine classifiers, Gaussian process and kernel Fisher discriminant analysis. Technical report, Katholieke Universiteit Leuven, August 2001.

F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10:1455–1480, 1998.

F. Girosi, M. Jones, and T. Poggio. Priors, stabilizers and basis functions: From regularization to radial, tensor and additive splines. Technical Report A.I. Memo No. 1430, Massachusetts Institute of Technology, June 1993.

G.H. Golub and C.F. van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, London, 3rd edition, 1996.

T. Graepel, R. Herbrich, B. Schölkopf, A.J. Smola, P.L. Bartlett, K.-R. Müller, K. Obermayer, and R.C. Williamson. Classification on proximity data with LP-machines. In D. Willshaw and A. Murray, editors, *Proceedings of ICANN'99*, volume 1, pages 304–309. IEE Press, 1999.

T. Graepel, R. Herbrich, and J. Shawe-Taylor. Generalization error bounds for sparse linear classifiers. In *Proc. COLT*, pages 298–303, San Francisco, 2000. Morgan Kaufmann.

A.J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. In *Proc. 10th Annu. Conf. on Comput. Learning Theory*, pages 171–183. ACM, 1997.

D.J. Hand. *Kernel discriminant analysis*. Research Studies Press, New York, 1982.

T.J. Hastie, A. Buja, and R.J. Tibshirani. Penalized discriminant analysis. *Annals of Statistics*, 23:73–102, 1995.

T.J. Hastie and R.J. Tibshirani. Discriminant analysis by gaussian mixtures. *Journal of the Royal Statistical Society*, pages 155–176, 1996.

D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz, July 1999.

R. Herbrich and T. Graepel. Large scale bayes point machines. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 528–534. MIT Press, 2001.

R. Herbrich, T. Graepel, and C. Campbell. Bayesian learning in reproducing kernel Hilbert spaces. Technical report, Technical University of Berlin, 1999. TR 99-11.

R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, August 2001.

R. Herbrich and R.C. Williamson. Algorithmic luckiness. *Journal of Machine Learning Research*, 3:175–212, September 2002.

P.J. Huber. *Robust Statistics*. John Wiley and Sons, New York, 1981.

T.S. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *J. Comp. Biol.*, 7:95–114, 2000.

T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

I.M. Johnstone. On the distribution of the largest principal component. Technical report, Department of Statistics, Stanford University, August 2000.

N. Karmarkar. A new polynomial algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

W. Karush. Minima of functions of several variables with inequalities as side constraints. Master's thesis, Dept. of Mathematics, Univ. of Chicago, 1939.

S.S. Keerthi and S.K. Shevade. SMO algorithm for least squares SVM formulations. Technical Report CD-02-08, National Univ. of Singapore, 2002.

S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. Technical Report CD-99-14, National University of Singapore, 1999. `http://guppy.mpe.nus.edu.sg/~mpessk`.

L.G. Khachiyan. A polynomial time algorithm in linear programming. *Doklady Akademia Nauk SSSR*, 244:1093–1096, 1979. also: USSR Computational Mathematics and Math. Phys., 20:53–72, 1980.

A.N. Kolmogorov. Stationary sequences in hilbert spaces. *Moscow University Mathematics*, 2:1–40, 1941.

V.I. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. In E. Gine, D. Mason, and J.A. Wellner, editors, *High Dimensional Probability II*, number 47 in Progress in Probability, pages 443–459, Boston, 2000. Birkhäuser.

H.W. Kuhn and A.W. Tucker. Nonlinear programming. In *Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics*, pages 481–492, Berkeley, 1951. University of California Press.

T. Kurita and T. Taguchi. A modification of kernel-based Fisher discriminant analysis for face detection. In *Proceedings of the 5th IEEE conference on automatic face and gesture recognition*, pages 300–304, Vancouver, Canada, 2002.

Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46:191–202, 2002.

N. Littlestone and M. Warmuth. Relating data compression and learnability. Technical report, University of California at Santa Cruz, USA, June 10 1986.

Q. Liu, R. Huang, H. Lu, and S. Ma. Face recognition using kernel based Fisher discriminant analysis. In *Proceedings of the 5th IEEE conference on automatic face and gesture recognition*, pages 197–201, 2002.

D.G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Co., Reading, second edition, May 1984. ISBN 0-201-15794-2. Reprinted with corrections in May, 1989.

A. Luntz and V. Brailowsky. On estimation characters obtained in statistical procedure of recognition. *Technicheskaya Kibernetica*, 3, 1969. In russian.

O.L. Mangasarian. Mathematical programming in data mining. *Data Mining and Knowledge Discovery*, 42(1):183–201, 1997.

O.L. Mangasarian and D.R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, March 2001.

L. Mason, P.L. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. Technical report, Department of Systems Engineering, Australian National University, 1998.

W.S. McCulloch and W. Pitts. A logistic calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, pages 148–188, 1989.

G.J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons, 1992.

S. Mehrotra. On the implementation of a (primal-dual) interior point method. *SIAM Journal on Optimization*, 2:575–602, 1992.

F. Meinecke, A. Ziehe, M. Kawanabe, and K.-R. Müller. A resampling approach to estimate the stability of one- or multidimensional independent components. To appear in IEEE Transactions on Biomedical Engineering, 2002.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

S. Mika. Kernalgorithmen zur nichtlinearen Signalverarbeitung in Merkmalsräumen. Master's thesis, Technische Universität Berlin, November 1998.

S. Mika, G. Rätsch, and K.-R. Müller. A mathematical programming approach to the kernel Fisher algorithm. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 591–597. MIT Press, 2001a.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999a.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A.J. Smola, and K.-R. Müller. Invariant feature extraction and classification in kernel spaces. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 526–532. MIT Press, 2000.

S. Mika, B. Schölkopf, A.J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de–noising in feature spaces. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 536–542. MIT Press, 1999b.

S. Mika, A.J. Smola, and B. Schölkopf. An improved training algorithm for kernel Fisher discriminants. In T. Jaakkola and T. Richardson, editors, *Proceedings AISTATS 2001*, pages 98–104, San Francisco, CA, 2001b. Morgan Kaufmann.

A.J. Miller. *Subset Selection in Regression*. Chapman and Hall, London, UK, 1990.

J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

V.A. Morozov. *Methods for Solving Incorrectly Posed Problems*. Springer Verlag, 1984.

K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12 (2):181–201, 2001.

S. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, New York, NY, 1996.

B.K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computation*, 24:227–234, 1995.

M. Opper and D. Haussler. Generalization performance of Bayes optimal classification algorithm for learning a perceptron. *Physical Review Letters*, 66:2677, 1991.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, pages 276–285, New York, 1997. IEEE.

E. Parzen. On estimation of probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 2001.

T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

G. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Neues Palais 10, 14469 Potsdam, Germany, October 2001.

G. Rätsch, A. Demiriz, and K. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1-3):193–221, 2002. Special Issue on New Methods for Model Selection and Model Combination. Also NeuroCOLT2 Technical Report NC-TR-2000-085.

G. Rätsch, S. Mika, B. Schölkopf, and K.-R. Müller. Constructing boosting algorithms from SVMs: an application to one-class classification. *IEEE PAMI*, 24 (9):1184–1199, September 2002. Earlier version is GMD TechReport No. 119, 2000.

G. Rätsch, S. Mika, and M.K. Warmuth. On the convergence of leveraging. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural information processings systems*, volume 14, 2002. Longer version also NeuroCOLT Technical Report NC-TR-2001-098.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, March 2001. also NeuroCOLT Technical Report NC-TR-1998-021.

G. Rätsch, B. Schölkopf, A.J. Smola, S. Mika, T. Onoda, and K.-R. Müller. Robust ensemble learning. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 207–219. MIT Press, Cambridge, MA, 2000a.

G. Rätsch, B. Schölkopf, A.J. Smola, S. Mika, T. Onoda, and K.-R. Müller. Robust ensemble learning for data mining. In H. Terano, editor, *Proc. PAKDD'00*, Lecture Notes in Artificial Intelligence. Springer, April 2000b.

G. Rätsch, B. Schölkopf, A.J. Smola, K.-R. Müller, T. Onoda, and S. Mika. ν-Arc: Ensemble learning in the presence of outliers. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 561–567. MIT Press, 2000c.

G. Rätsch, M. Warmuth, S. Mika, T. Onoda, S. Lemm, and K.-R. Müller. Barrier boosting. In *Proc. COLT*, pages 170–179, San Francisco, 2000d. Morgan Kaufmann.

B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics*, 27:832–837, 1956.

V. Roth and V. Steinhage. Nonlinear discriminant analysis using kernel functions. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 568–574. MIT Press, 2000.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

P. Ruján. Playing billiard in version space. *Neural Computation*, 9:197–238, 1996.

S. Saitoh. *Theory of Reproducing Kernels and its Applications*. Longman Scientific & Technical, Harlow, England, 1988.

C. Saunders, A. Gammermann, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521, 1998a.

C. Saunders, M.O. Stitson, J. Weston, L. Bottou, B. Schölkopf, and A.J. Smola. Support vector machine reference manual. Technical Report CSD-TR-98-03, Royal Holloway University, London, 1998b.

B. Schölkopf. *Support vector learning*. Oldenbourg Verlag, Munich, 1997.

B. Schölkopf. The kernel trick for distances. In T.K. Leen, T.G. Diettrich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors. *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999a.

B. Schölkopf, R. Herbrich, and A.J. Smola. A generalized representer theorem. In D.P. Helmbold and R.C. Williamson, editors, *COLT/EuroCOLT*, volume 2111 of *LNAI*, pages 416–426. Springer, 2001.

B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A.J. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, September 1999b.

B. Schölkopf, S. Mika, A.J. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction *via* approximate pre-images. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, pages 147 – 152, Berlin, 1998a. Springer Verlag.

B. Schölkopf, A. Smola, R.C. Williamson, and P.L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207 – 1245, 2000. also NeuroCOLT Technical Report NC-TR-1998-031.

B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

B. Schölkopf, A.J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998b.

J. Schürmann. *Pattern Classification: a unified view of statistical and neural approaches*. Wiley, New York, 1996.

A. Shashua. On the relationship between the support vector machine for classification and sparsified Fisher's linear discriminant. *Neural Processing Letters*, 9 (2):129–139, April 1999.

J. Shawe-Taylor, P.L. Bartlett, and R.C. Williamson. Structural risk minimization over data-dependent hierachies. *IEEE Transactions on Information Theory*, 44 (5):1926–1940, 1998.

J. Shawe-Taylor, P.L. Bartlett, R.C. Williamson, and M. Anthony. A framework for structural risk minimization. In *Proc. COLT*. Morgan Kaufmann, 1996.

J. Shawe-Taylor, N. Cristianini, and J. Kandola. On the concentration of spectral properties. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

J. Shawe-Taylor and R.C. Williamson. A PAC analysis of a Bayesian estimator. Technical Report NC2-TR-1997-013, Royal Holloway, University of London, 1997.

P.Y. Simard, Y.A. LeCun, J.S. Denker, and B. Victorri. Transformation invariance in pattern recognition – tangent distance and tangent propagation. In G. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524, pages 239–274. Springer LNCS, 1998.

A.J. Smola. *Learning with Kernels*. PhD thesis, Technische Universität Berlin, 1998.

A.J. Smola and P.L. Bartlett. Sparse greedy gaussian process regression. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 619–625. MIT Press, 2001.

A.J. Smola, S. Mika, and B. Schölkopf. Quantization functionals and regularized principal manifolds. Technical Report NC-TR-98-028, Royal Holloway College, University of London, UK, 1998a.

A.J. Smola, S. Mika, B. Schölkopf, and R.C. Williamson. Regularized principal manifolds. *Journal of Machine Learning Research*, 1:179–209, June 2001.

A.J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In P. Langley, editor, *Proc. ICML'00*, pages 911–918, San Francisco, 2000. Morgan Kaufmann.

A.J. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998b.

A.J. Smola, R.C. Williamson, S. Mika, and B. Schölkopf. Regularized principal manifolds. In Paul Fischer and Hans Ulrich Simon, editors, *Proceedings of EuroCOLT 99)*, volume 1572 of *LNAI*, pages 214–229, Berlin, March 1999. Springer.

G.W. Stewart. Error and pertubation bounds for subspaces associated with certain eigenvalue problems. *SIAM Reviews*, pages 727–764, 1973.

G.W. Stewart and J. Sun. *Matrix pertubation theory*. Academic Press, 1990.

M. Stitson, A. Gammerman, V.N. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector regression with ANOVA decomposition kernels. Technical Report CSD-97-22, Royal Holloway, University of London, 1997.

J.A.K. Suykens and J. Vanderwalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

M. Talagrand. A new look at independence. *Annals of Probability*, 24:1–34, 1996.

A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-posed Problems*. W.H. Winston, Washington, D.C., 1977.

M.E. Tipping. The relevance vector machine. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 652–658. MIT Press, 2000.

K. Tsuda. Support vector classifier with asymmetric kernel functions. Technical Report TR-98-31, Electrotechnical Laboratory, Japan, 1998.

K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.R. Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14:2397–2414, 2002.

K. Tsuda, G. Rätsch, S. Mika, and K.-R. Müller. Learning to predict the leave-one-out error of kernel based classifiers. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Artificial Neural Networks — ICANN'01*, pages 331–338. Springer Lecture Notes in Computer Science, Vol. 2130, 2001.

A. Turing. Computing machinery and intelligence. *Mind*, 59:433–560, 1950.

A.W. van der Vaart and J.A. Wellner. *Weak Convergence and Empirical Processes*. Springer, 1996.

J.L.M. van Dorsselaer, M.E. Hochstenbach, and H.A. van der Vorst. Computing probabilistic bounds for extreme eigenvalues of symmetric matrices with the Lanczos method. *SIAM Journal on Matrix Analysis and Applications*, 22(3): 837–852, 2000.

R.J. Vanderbei. Interior-point methods: Algorithms and formulations. *ORSA Journal on Computing*, 6(1):32–34, 1994.

R.J. Vanderbei. LOQO user's manual – version 3.10. Technical Report SOR-97-08, Princeton University, Statistics and Operations Research, 1997. Code available at http://www.princeton.edu/~rvdb.

R.J. Vanderbei and D.F. Shanno. An interior point algorithm for nonconvex nonlinear programming. Technical Report SOR-97-21, Statistics and Operations Research, Princeton University, 1997.

V.N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, Berlin, 1982.

V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, 1995.

V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

V.N. Vapnik and A.Y. Chervonenkis. *Theory of Pattern Recognition*. Nauka, Moskow, 1974. in Russian.

V.N. Vapnik and A.Y. Chervonenkis. The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 1(3):283–305, 1991.

T. Watkin. Optimal learning with a neural network. *Europhysics Letters*, 21: 871–877, 1993.

C. Watkins. Dynamic alignment kernels. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50, Cambridge, MA, 2000. MIT Press.

J.H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, England, 1965.

C.K.I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In V. Tresp T.K. Leen, T.G. Dietrich, editor, *NIPS*, volume 13, pages 682–688. MIT Press, 2001.

R.C. Williamson, A.J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. NeuroCOLT Technical Report NC-TR-98-019, Royal Holloway College, University of London, UK, 1998.

J. Xu, X. Zhang, and Y. Li. Kernel MSE algorithm: a unified framework for KFD, LS-SVM and KRR. In *Proceedings of IJCNN*, pages 1486–1491, 2001.

M.-H. Yang. Face recognition using kernel methods. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

M-H. Yang, N. Ahuja, and D. Kriegman. Face recognition using kernel eigenfaces. In *Proceedings of the 2000 IEEE International Conference on Image Processing*, pages 37–40, Vancouver, Canada, September 2000.

A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites. *BioInformatics*, 16(9):799–807, September 2000.