Sparse Nonlinear Discriminants

Der Fakultät für Elektrotechnik und Informationstechnik der
Otto-von-Guericke-Universität Magdeburg
zur Erlangung des akademischen Grades

Doktoringenieur (Dr. -Ing.)

am 26. Juni 2007 vorgelegte Dissertation
von **Dipl. -Ing. Edin Andelić**

Dedicated to

Yvonne

Zusammenfassung

Diese Dissertation beschäftigt sich mit Trainingsalgorithmen für das maschinelle Lernen und deren Anwendungen für die Klassifikation, die Regression und die automatische Spracherkennung. Es wird im Speziellen das überwachte Lernen betrachtet, das auch als Lernen aus Beispielen bezeichnet wird.

Anhand einer kurzen Einführung in die statistische Lerntheorie wird gezeigt, dass sich das überwachte Lernen als ein Funktionsschätzungsproblem formulieren lässt, bei dem sich die Klasse der linearen Funktionen als besonders geeignet für die Generalisierungsfähigkeit der Lösung erweist. Die Leistungsfähigkeit der linearen Funktionen wird zusätzlich durch die Anwendung sogenannter Kernfunktionen gesteigert, die es in effektiver Weise erlauben bestimmte Algorithmen in nichtlineare Räume zu transformieren. Ein wichtiger Vorteil von Kernfunktionen ist, dass die vom gegebenen Algorithmus geschätzten Funktionen in diesem neuen nichtlinearen Raum weiterhin linear sind, so dass die theoretischen Vorteile lineare Funktionen in Bezug auf die Generalisierungsfähigkeit erhalten bleiben. Ein solcher Algorithmus, der sich mit Kernfunktionen nichtlinear transformieren lässt, ist die Diskriminante. Es werden eine Reihe ordnungsrekursiver Algorithmen hergeleitet, die es erlauben, mit vertretbarem Aufwand die durch Kernfunktionen induzierte nichtlineare Version der Diskriminante zu berechnen. Die Algorithmen basieren einerseits auf der Tatsache, dass sich das Diskriminantenprob-

lem als äquivalent zu einer kleinsten-Quadrate-Regression erweist. Andererseits zeigt sich, dass sich die Lösung stark ausdünnen lässt, in dem Sinne dass nur ein kleiner Teil der Trainingsdaten für die Lösung ausgewählt wird. Dies reduziert sowohl den Aufwand beim Training als auch beim Testen erheblich.

Darüberhinaus wird die Tatsache genutzt, dass sich die Ausgaben der Diskriminanten probabilistisch interpretieren lassen. Die so gewonnenen Wahrscheinlichkeiten werden als Emissionswahrscheinlichkeiten für Hidden-Markov-Modelle verwendet und innerhalb eines automatischen Spracherkenners getestet. Schließlich werden die vorgestellten Algorithmen für Klassifikations- und Regressionsaufgaben in einer großen Sammlung von Experimenten auf wohlbekannten Datenbasen ausgewertet und mit anderen bewährten Lernalgorithmen verglichen.

Abstract

This Dissertation considers training algorithms for machine learning and their applications for classification, regression and automatic speech recognition. Particularly, supervised learning, which is also called learning from samples, is considered.

Starting with a short introduction into statistical learning theory it is shown, that supervised learning can be formulated as a function estimation problem where the class of linear functions turns out to be an appropriate choice for obtaining solutions with high generalization ability. The performance of linear functions may then be enhanced further by the use of the so-called kernel functions, which allow effectively to transform certain algorithms into nonlinear spaces. An important advantage of the kernel functions is that the estimated functions are still linear in the new nonlinear space such that the theoretical benefits of linear functions regarding the generalization ability are preserved. The discriminant approach turns out to be appropriate for being transformed into nonlinear spaces using kernel functions. We propose some order-recursive algorithms which allow to estimate a nonlinear kernel-induced version of the discriminant with a reasonable cost. These algorithms are based on two facts. First, the discriminant approach is equivalent to a certain least-squares regression. Second, the solution can be sparsified in the sense that only a small fraction of the training points is used for the

solution such that the cost for training and testing is remarkably reduced.

Furthermore, we use the fact that the outputs of the discriminants may be interpreted probabilistically. The resulting probabilities are used as emission probabilities for Hidden-Markov-Models and tested within an automatic speech recognizer. Finally, the proposed algorithms are evaluated for classification and regression tasks using a large collection of well-known databases and the results are compared with other state-of-the-art learning algorithms.

Danksagung

Ich möchte meinen tief empfundenen Dank all meinen ehemaligen Arbeitskollegen am Lehrstuhl Kognitive Systeme in Magdeburg zum Ausdruck bringen. Es war eine wunderbare Zeit, die ich nie vergessen werde. Ich hoffe, ihr empfindet das zumindest ähnlich. Meinem Promotionsbetreuer Prof. Wendemuth danke ich allen voran. Andreas, danke für Dein Vertrauen und Deine Unterstützung!

Desweiteren bin ich dem Land Sachsen-Anhalt zu großem Dank verpflichtet. Diese Arbeit wurde durch ein Promotionsstipendium des Landes Sachsen-Anhalt unterstützt.

Last but not least, möchte ich meiner Familie und meiner geliebten Freundin Yvonne danken. Ohne ihre Unterstützung, vor allem in schwierigen Situationen, wäre diese Arbeit nicht möglich gewesen.

Contents

Zusammenfassung **iii**

Abstract **v**

Acknowledgements **vii**

1 Introduction **1**

1.1 Supervised Learning 1

1.2 Problem Formulation and Outline of the Thesis 3

1.3 Motivation 4

2 Mathematical Background **7**

2.1 An Induction Principle 7

2.2 Consistency 9

2.3 Capacity Measures and Structural Risk Minimization 12

2.4 Support Vector Machines 14

2.4.1 The Linearly Separable Case 14

2.4.2 The Linearly Inseparable Case 19

2.5 Kernel Functions 21

2.5.1 The Kernel Trick 22

2.5.2 Kernel-Induced Feature Spaces 25

2.6 Summary 29

3 Discriminants 30

3.1 Linear Discriminants 30

3.2 Equivalence to Least-Squares 35

3.3 Kernel-Based Discriminants 37

3.3.1 Regularization 40

3.4 Sparse Approximations 41

3.4.1 Nonlinear Pseudodiscriminants 44

3.4.2 Orthogonal Least Squares 52

3.4.3 Recursive Least Squares 58

3.5 Summary 63

4 Application of Nonlinear Discriminants for Automatic Speech Recognition 64

4.1 Components of ASR 65

4.1.1 Sub-word modeling with HMMs 68

4.2 Using Nonlinear Discriminants for ASR 72

4.2.1 Probabilistic Outputs 72

4.3 Implementation 74

4.4 Experiments 74

4.4.1 Experimental Setup 74

4.4.2 Results 75

4.5 Summary 76

5 Experiments for Classification and Regression 78

5.1 Classification 78

5.1.1 Optical Character Recognition 79

5.1.2 Other Benchmarks 81

5.2 Regression 82
5.3 Summary 86

6 Conclusion and Future Work **88**

Declaration **92**

Persönlicher Werdegang **93**

List of Figures

2.1 Example of overfitting: The dashed function perfectly describes the given data points but is very likely to overfit whereas the linear function is more likely to generalize to unseen samples. 10

2.2 Illustration of consistency: By minimizing the training error we decrease the empirical risk. At the same time the bound on the complexity of our function class gets worse. Thus, the goal is to find an optimal trade-off between the complexity and the empirical risk, i. e. to minimize expected risk. 15

2.3 Margins and hyperplanes: A linear classifier is defined by the hyperplane's (solid line) normal vector $\mathbf{w}$ and the bias b. Each side of the hyperplane correspond to one class. The margin is the minimal distance between any training instance and the hyperplane, i. e. the distance between the solid and the dotted lines. 17

2.4 The simple XOR-problem: A correct classification of all outcomes of the binary XOR-function can not be achieved by linear functions. 22

2.5 The data are mapped by a nonlinear mapping into the feature space and the problem may become linearly separable when we choose an appropriate nonlinear map. The linear directions in the new feature space correspond to nonlinear ones in the original input space. 23

3.1 Geometrical interpretation of the LD approach. The two classes (black and white circles) are projected (dashed lines) onto the discriminating direction **w** such that the variances $\sigma_{\pm}$ within the classes are minimized and the means $\mu_{\pm}$ of the two projected classes are maximally separated. 32

3.2 Comparison of the decision boundaries obtained by the full KFD and our proposed approximation (NPD) on Ripley's dataset. The training samples are also shown and the ten selected basis centers used by the NPD are encircled. 51

4.1 Principle components of an ASR system 65

4.2 Feature extraction from a speech signal. Every 'hop-size' (or shift-size) seconds a vector of features is computed from the speech samples in a window of length 'window-size'. 66

4.3 Monophone, biphone, and triphone HMMs for the English word "bat" [b ae t]. 'sil' stands for silence at the beginning and end of the utterance, which is modeled as a 'phone', too. . 71

5.1 Decision boundaries on the two spirals classification problem using the NPD with different number of basis centers. 80

5.2 Example fit to a noisy sinc function for OROLS using 50 randomly generated points for training and testing. The standard deviation of the Gaussian noise is 0.1. The Root Mean Square Error (RMSE) is 0.0269 in this case. 9 points are selected as basis centers. 84

5.3 RMSE of fits to a noisy sinc function w. r. t. different training set sizes using OROLS. 1000 randomly generated points are used for testing. The standard deviation of the Gaussian noise is 0.1 in all runs. The results are averaged over 100 runs for each size. 85

5.4 RMSE of fits to a noisy sinc function w. r. t. different noise levels using OROLS. 100 / 1000 randomly generated points are used for training / testing. The results are averaged over 100 runs for each noise level. 85

List of Tables

4.1 Extract from a dictionary 68

4.2 Results on the RM1 Feb'89 test set. 76

5.1 Datasets used for the classification experiments. 81

5.2 Test errors in % on 5 benchmark datasets. The one-vs-rest approach is used. Average fraction of selected basis centers in % within parentheses. 82

5.3 Estimation of generalization errors on 13 benchmark data sets in % with standard deviations and sparsity levels in % within brackets (best result in **bold face**, second *emphasized*). 83

5.4 Average RMSE for the sinc experiment using the SVM and OROLS. 50 / 1000 randomly generated points are used for training / testing. The standard deviation of the Gaussian noise is 0.1 in all runs. The results are averaged over 100 runs. 84

5.5 Mean Square Error (MSE) with standard deviations for the Boston and Abalone dataset using different methods. 86

Chapter 1

Introduction

This chapter describes the contributions of this thesis and sketches its organization. Furthermore, for readers who are not familiar with machine learning this chapter provides a very brief and rough overview of the most important motivations and central questions referring to this field. Especially, supervised learning is discussed.

1.1 Supervised Learning

The topic of this thesis is the development and application of training algorithms for supervised learning which is often referred to as learning from examples. In supervised learning the training set is always labeled with known targets. There are two settings for a supervised learning scenario, namely classification and regression. In case of classification the labels have discrete values indicating the class the corresponding training input belongs to. Consider for instance Optical Character Recognition (OCR). In OCR the problem setting is as follows. The training set contains pictures of digits or characters hand-written by various writers. These pictures may be rep-

resented by vectors of fixed length where each entry contains e. g. the grey scale of the corresponding pixel. The labeling of the inputs is usually done by hand and indicates which class each input belongs to, i. e. in case of OCR which digit or character is represented by each picture. Note the difference between the inputs which could be raw data (e. g. pixels representing a digit) and the desired class (e. g. "2") which is symbolic. Other symbolic classes may be defined on the same raw data, such as "even" or "below 5". In most cases it is assumed that the labels reflect the true input-output relation. The machine learning algorithm is trained using this set of labeled inputs. As stated in the previous section the goal is that the trained model generalizes well. This means in the OCR case that a digit or character which is not included in the training set or is even written by a different writer should be classified correctly by the model.

For regression the labels are continuously valued. A basic example for this setting is fitting a real-valued function to some observed data points which are usually corrupted by noise. In this example the problem of generalization becomes very apparent. Obviously, it is always possible to find a very complicated function that perfectly describes the data. However, it is very unlikely that such a function will perform well on unseen data. Hence, for good generalization it is crucial how complicated the class is from which one chooses the function. In the following chapter it will be shown how a function class can be characterized.

1.2 Problem Formulation and Outline of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 describes the mathematical framework for machine learning and rewiews the basic mathematical definitions and theorems from learning theory regarding the following questions:

- What do we mean when we say learning?
- When is learning possible?
- What does generalization mean?
- How can we achieve a good generalization ability?

Furthermore, a famous state-of-the-art machine learning algorithm - the Support Vector Machine (SVM) is presented. SVMs are directly derived from the insights of statistical learning theory. Finally, a special class of learning machines, namely kernel-machines are considered. They are based on so-called kernel-functions which allow easily to turn linear algorithms like the SVM into nonlinear ones. The most important properties and advantages of kernel-functions are discussed. A particularly important advantage of kernel-based algorithms is that the non-linearities are induced implicitly by the kernel functions. Thus, this reduces the problem of finding a possibly very complicated and high-dimensional nonlinear function to tuning a single parameter of one function. This is very much in contrast to e. g. neural networks where the nonlinear directions have to be designed explicitly. However, the drawback of kernel-based algorithms is that in most cases a quadratic matrix of the same dimension like the number of training samples is involved during the training. This is clearly not reasonable especially

for large datasets and there is a need of efficient kernel-based training algorithms. There has been a large body of work related to efficient training of SVMs since it became apparent that SVMs perform surprisingly well in combination with kernel functions. Another promising approach is the so-called discriminant. Discriminants may be easily kernelized like SVMs and perform very competitively on various tasks. The development of training algorithms for discriminants with a reasonable cost is the central topic of this thesis. In Chapter 3 linear and nonlinear discriminants are considered. Discriminants are originally used for classification where the linear discriminant finds a direction which in some sense optimally separates inputs of one class from the other. It is reviewed how linear discriminants may be turned into nonlinear ones using kernel-functions in order to obtain more flexible classification and regression models. Furthermore, it is shown that this discriminative approach is closely related to a least-squares regression onto the labels and using this insight a variety of order-recursive algorithms for nonlinear discriminants is presented. Chapter 4 summarizes Automatic Speech Recognition (ASR) and shows how the probabilistically interpreted outputs of nonlinear discriminants may be applied for ASR. In Chapter 5 a large collection of experimental results for classification regression using the developed algorithms is presented. Finally, concluding remarks and an outlook for future work is given in chapter 6.

1.3 Motivation

During the last decades, very successful research and development in information and computer technology lead to the ability to store and process huge amounts of data. This progress was and is still very fruitful for many

research fields. To mention just a few examples, astrophysics, meteorology and human genome research benefit greatly from state-of-the-art computer technology. Furthermore, the internet is probably the most famous example of a vast repository of all kinds of information which changes constantly and grows at an exponential rate. However, the question is how to extract useful information from such a huge amount of data and how to derive rules that explain the characteristics of these data in order to gain a deeper insight into the underlying processes.

One option could be to develop a model that e. g. physically describes how the data are generated. This approach works well in restricted cases where a-priori knowledge is available. However, the problem is that a lot of phenomena not only in research but also in one's everyday life are very hard and sometimes impossible to be modeled reasonably. Consider for instance human genome research. It is not fully understood yet how the interplay between the human genes looks like and how the genes code useful information. Very complex models exist which only partially describe the situation.

At this point, machine learning which is a subdiscipline of artificial intelligence comes into play. Machines that learn are algorithmic systems that are trained instead of engineered. The machine is faced with a set of observations - the training set - which can be e. g. measurements carried out in some experiments. The task of every machine learning algorithm is on the one hand to explain the relations between the observations themselves or between the observations and some known or desired targets. On the other hand such a model should not only explain the observed data used for the training but also predict certain relations for unseen inputs that lie in the same domain. Speaking more mathematically, the task is to find a function that lies in a certain space and predicts the relations related to unseen inputs

as correctly as possible. This property is called generalization and plays the central role in machine learning research. The machine learning approach is inductive because it relies exclusively on the observed data and does not assume any physically or elsewise motivated model. Thus, one may say that machine learning is about letting the data speak.

Chapter 2

Mathematical Background

This chapter summarizes the most important insights from statistical learning theory which is the mathematical basis for machine learning. It is discussed by means of statistical learning theory what machine learning aims at and what are the necessary and sufficient condition for learning. Furthermore, kernel functions are discussed. The use of kernel functions is an elegant way to formulate linear learning algorithms in nonlinear spaces and hence to overcome the limited performance of linear functions.

2.1 An Induction Principle

As mentioned in the previous chapter the topic of this thesis is supervised learning. In a supervised learning scenario one is faced with empirical observations

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

In most cases the domain $\mathcal{X} \in \mathbb{R}^d$ is considered as a vector space where d denotes the dimension of the inputs. Note, however, that $\mathcal{X}$ needs not necessarily be a vector space. $\mathcal{X}$ is only required to be a set. $\mathcal{Y}$ denotes the

domain from which the labels are sampled. Throughout this thesis a set of given inputs is ordered in matrix form and denoted as $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ where M is the number of given training instances. A label set is given in vector form and denoted as $\mathbf{y} = \{y_1, \ldots, y_M\}^T$ which is contained in $\mathbb{R}$ for regression and in $\{1, -1\}^M$ for classification.

In statistical learning theory it is always assumed that the training data are sampled independently and identically from an unknown but fixed distribution P on the set $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. These kind of data are often called independently and identically distributed (iid). Assuming for a while that we could compute the quantities $P(y|\mathbf{x})$ the learning problem could be stated as finding a function f that belongs to a certain function class $\mathcal{F}$ such that the risk

$$R(f) = \int_{\mathcal{X}\times\mathcal{Y}} l(\mathbf{x}, y, f(\mathbf{x}))\, dP(\mathbf{x}, y) \tag{2.2}$$

is minimized. f is a function the learning machine can implement, i. e. the function by which the predictions for each data sample are made, and l is a non-negative loss function which is a measure for the error of the predictions. Since we assumed the knowledge of $P(y|\mathbf{x})$ the learning problem could easily be solved simply by finding a function f^* for which the a posteriori probability reaches a maximum, i. e. ,

$$f^*(\mathbf{x}) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}} P(y|\mathbf{x}) \tag{2.3}$$

where $P(y|\mathbf{x})$ may be computed by Bayes' law

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} = \frac{P(\mathbf{x}, y)}{P(\mathbf{x})}. \tag{2.4}$$

Unfortunately, only in constructed or trivial cases we have access to P. In general we do not know anything about P. What we do know is the training data. Thus, we may resort to what is called an induction principle. Instead of minimizing the true risk (2.2) we may perform an Empirical Risk

Minimization (ERM):

$$R_{emp}(f) = \sum_{i=1}^{M} l(\mathbf{x}_i, y_i, f(\mathbf{x_i})). \tag{2.5}$$

The law of large numbers tells us that the empirical risk converges to the true risk as the number of instances M goes to infinity. Moreover, the convergence in probability has an exponential rate which can be seen in the following theorem.

Theorem 1 (Hoeffding) *Let ζ_i, $i \in \{1, \ldots, M\}$ be M independent samples of a bounded random variable ζ, with values in the interval $[a, b]$. Then for any $\varepsilon > 0$,*

$$P\left\{\left|\frac{1}{M}\sum_{i=1}^{M}\zeta_i - \mathbf{E}(\zeta)\right|\right\} \leq \exp\left(-\frac{2M\varepsilon^2}{(b-a)^2}\right). \tag{2.6}$$

2.2 Consistency

However, theorem 1 does not imply that the minimum of the empirical risk converges to the minimum of the true risk in probability for all functions in $\mathcal{F}$. This is the question of consistency which plays a central role in statistical learning theory. If we denote the function that minimizes the empirical risk on the basis of a given training set $\mathbf{X}$ by $f^{M\ 1}$ then the ERM is said to be consistent if

$$|R_{emp}(f^M) - R(f^M)| \underset{M\to\infty}{\to} 0 \tag{2.7}$$

for all functions in $\mathcal{F}$.

In order to see that ERM alone is not consistent in general and thus does not lead to successful learning, consider the illustrative regression example in

[1]Note that f^M needs not to be unique.

Fig. 2.1. The noisy training data may be perfectly described by a polynomial of high degree for instance. The empirical risk is zero but the polynomial may take arbitrary values in regions that do not belong to the training set. This phenomenon is often referred to as overfitting. In contrast the linear function exhibits residual errors but it is on the other hand more predictive with respect to unseen samples. For classification one can find a similar example. If we do not restrict the function class $\mathcal{F}$ and allow arbitrary functions to be implemented by the learning machine a possible choice would be a function that simply memorizes the data, i. e. , a function that takes values y_i for $\mathbf{x} = \mathbf{x}_i$ and 1 otherwise. Again the empirical risk on the training data is zero but the learning machine will almost never predict an unseen sample correctly. Thus, if we do not restrict the function class we always can find infinitely many functions which minimize the empirical risk. It turns out that

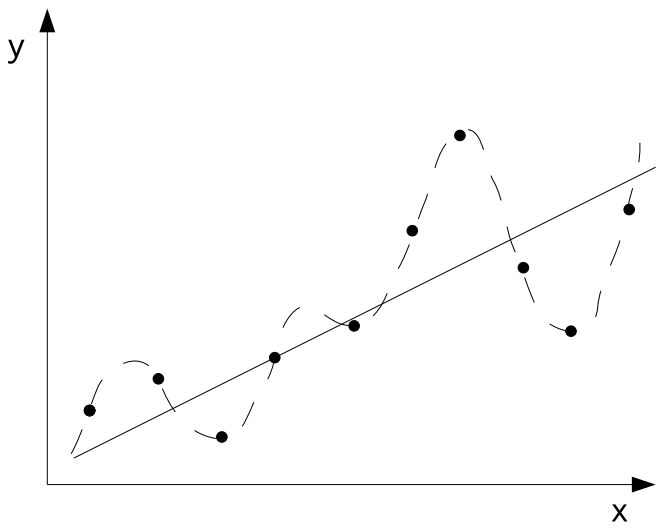

Figure 2.1: Example of overfitting: The dashed function perfectly describes the given data points but is very likely to overfit whereas the linear function is more likely to generalize to unseen samples.

successful learning crucially depends on a trade-off between the complexity of $\mathcal{F}$ and the minimization of the empirical risk. Thus, what we need is a generalized version for the law of large numbers, i. e , a condition that guarantees that the empirical risk consistently converges to the actual risk for all functions $f \in \mathcal{F}$. Without going too much into detail we state a key theorem [Vapnik and Chervonenkis, 1991] from statistical learning theory specifying the condition for consistency of ERM.

Theorem 2 *One-sided uniform convergence in probability,*

$$\lim_{M\to\infty} P\left\{\sup_{f\in\mathcal{F}}(R(f) - R_{emp}(f) > \varepsilon)\right\} = 0, \tag{2.8}$$

for all $\varepsilon > 0$ is a necessary and sufficient condition for nontrivial consistency of ERM for a given function class $\mathcal{F}$. □

Theorem 2 tells us that the convergence of the *worst case* over all functions $f \in \mathcal{F}$ is necessary and sufficient for nontrivial consistency. Consistency is said to be trivial if we restrict the function class too much. This can be seen as the converse of overfitting and is often referred to as underfitting. For instance, if we allow only one function to be implemented by the learning machine then ERM would be trivially consistent since $R(f) = R_{emp}(f) =$ const.

Theorem 2 whilst being theoretically very appealing does not provide us with a measure for the complexity of the function class leading to computable bounds for generalization. In the following we will outline possibilities for doing so.

2.3 Capacity Measures and Structural Risk Minimization

So far, we have seen that doing ERM alone is no guarantee for successful learning and that consistency is a crucial property of a learning machine. We will now concentrate on the question if there are any principles that allow us to choose only such function classes for learning that fulfill Theorem 2? It turns out that the complexity of the functions in the class $\mathcal{F}$ is crucial for consistency and hence for fulfilling Theorem 2. But what does complexity mean and how can we control the complexity of a function class? Roughly speaking, the complexity of a function class is determined by the number of different possible outcomes when choosing functions from this class. There are a number of different complexity measures for functions classes. Among others, popular measures are covering numbers, annealed entropy, Vapnik Chervonenkis entropy (VC entropy) and the VC dimension, or the Rademacher complexity [Vapnik, 1998]. In the following we will present the notion of the VC dimension in detail and quantify its implications for the so called Structural Risk Minimization (SRM) which motivated the introduction of a very famous learning machine, namely the Support Vector Machine (SVM) [Cortes and Vapnik, 1995] [Schölkopf and Smola, 2002].

The VC dimension for a function class $\mathcal{F}$ is defined as the maximum number of training points that can be shattered (i. e. separated) by $\mathcal{F}$. If the VC dimension is h, then there exists at least one set of h points that can be shattered by the corresponding function class. But note that in general it will not be true that every set of h points can be shattered. The VC dimension can be used to bound the left-hand side in Theorem 2, i. e.

$$P\left\{\sup_{f\in\mathcal{F}}(R(f)-R_{emp}(f)>\varepsilon)\right\}\leq H(\mathcal{F},M,\varepsilon), \tag{2.9}$$

where H is a function that depends on the complexity of the function class $\mathcal{F}$, the size of the training set M and the chosen precision ε. For a random draw of the training sample $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ we can easily turn this kind of bounds into the following form. With probability $1 - \delta$, the actual risk can be bounded as

$$R(f) \leq R_{emp}(f, \mathcal{Z}) + \tilde{H}(\mathcal{F}, M, \delta). \tag{2.10}$$

The function $\tilde{H}$ is a penalty term which measures the degree of uncertainty. Independently of our chosen capacity measure, this penalty term usually increases monotonically with a higher precision $1-\delta$ and a higher complexity of $\mathcal{F}$ and decreases monotonically with a higher number of training samples M.

It can be seen from the bound above that successful learning can be achieved by finding a function that produces a small empirical error and at the same time keeps the penalty term $\tilde{H}$ small. However, it should be noted that the bound (2.10) only holds for learning machines with finite function classes. For instance, learning machines like k-nearest neighbors which can implement function classes with infinite VC dimension work well in practice. Thus, the bound (2.10) is only a sufficient and not a necessary condition for nontrivial consistency.

When we use the 0/1-loss

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

for the empirical risk R_{emp}, then for the case of two-class classification problems the VC theory provides us with the following bound [Vapnik and Chervonenkis, 1974].

Theorem 3 *Let h denote the finite VC dimension of the function class $\mathcal{F}$. For all $\delta > 0$ and $f \in \mathcal{F}$ the inequality bounding the risk*

$$R(f) \leq R_{emp}(f, \mathcal{Z}) + \sqrt{\frac{h(\ln(\frac{2M}{h}) + 1) - \ln(\frac{\delta}{4})}{M}} \tag{2.12}$$

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

Structural Risk Minimization (SRM) [Cortes and Vapnik, 1995] is based on these insights. The penalty term in (2.10) depends on the capacity of the chosen class of functions, whereas the empirical risk and the actual risk depend on one particular realization of this function class. The goal of SRM is to choose a subset of the function class, such that the risk bound for that subset is minimized. To this end, we must introduce some structure into the entire class of functions the learning machine can implement by dividing this class into nested subsets $\mathcal{F}_1 \subseteq \ldots \subseteq \mathcal{F}_n$ with non-decreasing VC dimensions $h_1 \leq \ldots h_n$. Now we can proceed as follows. For each subset, the empirical risk is minimized by choosing the optimal realization f_i of the corresponding subset $\mathcal{F}_i$. At the end we choose that trained machine whose sum of empirical risk and VC confidence is minimal. This procedure is illustrated in Fig. 2.2. In the following we shall present a famous practical implementation of this principle.

2.4 Support Vector Machines

2.4.1 The Linearly Separable Case

We consider the two-class classification problem with some labeled training data $\{\mathbf{x}_i, y_i\}, \quad i = 1, \ldots, M, \quad y_i \in \{1, -1\}, \quad \mathbf{x}_i \in \mathcal{X} = \mathbb{R}^d$. Furthermore,

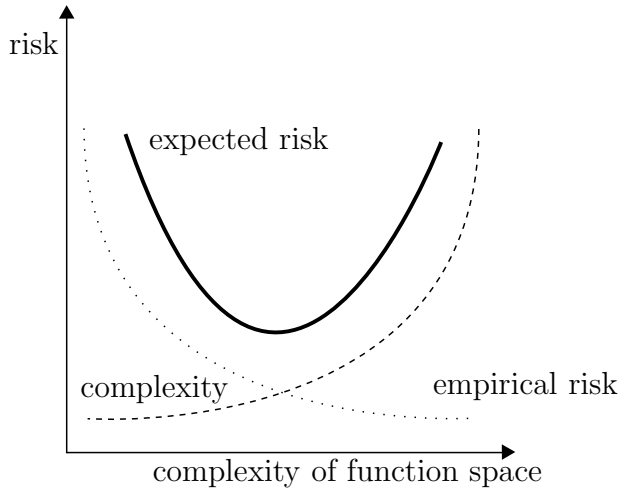

Figure 2.2: Illustration of consistency: By minimizing the training error we decrease the empirical risk. At the same time the bound on the complexity of our function class gets worse. Thus, the goal is to find an optimal trade-off between the complexity and the empirical risk, i. e. to minimize expected risk.

we assume that the data are linearly separable, i. e. there is a linear function

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{2.13}$$

of the training points $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) < 0$ whenever $y = -1$ and $f(\mathbf{x}) \geq 0$ otherwise. We define that the function class $\mathcal{F}$ is described by all possible linear hyperplanes of the form (2.13), i. e.

$$\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R} | f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b\}. \tag{2.14}$$

Thus, each hyperplane is completely described by its normal vector $\mathbf{w}$ and the offset b. We can use for instance the VC dimension to measure the complexity of this function class. It is straightforward to show that a hyperplane of the given form can separate maximally $d + 1$ points in a d-dimensional vector space like $\mathcal{X} = \mathbb{R}^d$ for all possible labelings. Thus, the VC dimension of our

assumed function class is $h = d + 1 < \infty$ and the application of the bound given in (2.10) makes sense. However, we can not apply SRM yet since we do not have a nested structure of function classes. We may construct the desired nested structure by limiting the function classes in the form

$$\mathcal{F}_\Lambda = \{f : \mathcal{X} \to \mathbb{R} | f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \quad \|\mathbf{w}\| < \Lambda\}. \tag{2.15}$$

It is clear that $\mathcal{F}_{\Lambda_i} \subseteq \mathcal{F}_{\Lambda_j}$ for $\Lambda_i \leq \Lambda_j$. However, if the data are separable by $\text{sign}(\mathbf{w}^T\mathbf{x} + b)$ then they are separable using any positive multiple of $(\mathbf{w}, b)$ and hence all function classes would have the same VC dimension since they contain the same hyperplane in different representations. What we aim at is that $h(\mathcal{F}_{\Lambda_i}) \leq h(\mathcal{F}_{\Lambda_j})$ for $\Lambda_i \leq \Lambda_j$. To this end, we need a unique representation for each hyperplane. One way to achieve this is to define a canonical representation for each hyperplane by scaling the normal vector $\mathbf{w}$ and adjusting the offset b such that none of the training points produces an output smaller than one, i. e.

$$\min_{i=1,\ldots,M} |f(\mathbf{x}_i)| = 1. \tag{2.16}$$

Using this canonical representation we are able to measure how good the separation of the data by the separating hyperplane is. This is the concept of the so-called margin. The margin is defined as the minimal euclidean distance between any training point and the separating hyperplane. This is illustrated in Fig. 2.3. The margin can be measured by the length of the normal vector $\mathbf{w}$ since we assumed that the hyperplanes are in canonical form. This can be seen by the following example. Consider two training points $\mathbf{x}_1$ and $\mathbf{x}_2$ belonging to two different classes. Each of the training points is located on the edge of the margin with $\mathbf{w}^T\mathbf{x}_1 + b = 1$ and $\mathbf{w}^T\mathbf{x}_2 + b = -1$. Then the margin is given by the minimal distance between them measured

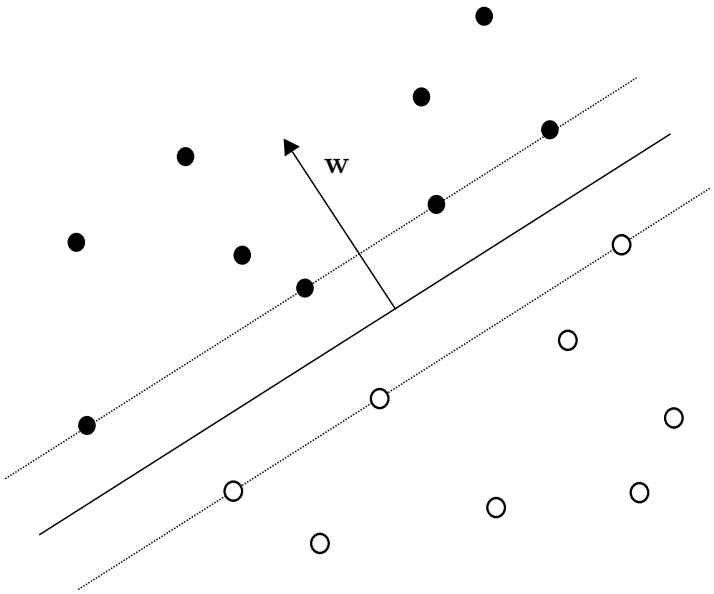

Figure 2.3: Margins and hyperplanes: A linear classifier is defined by the hyperplane's (solid line) normal vector $\mathbf{w}$ and the bias b. Each side of the hyperplane correspond to one class. The margin is the minimal distance between any training instance and the hyperplane, i. e. the distance between the solid and the dotted lines.

perpendicularly to the hyperplane, i. e.

$$\frac{\mathbf{w}}{\|\mathbf{w}\|}(\mathbf{x}_1 - \mathbf{x}_2) = \frac{2}{\|\mathbf{w}\|}. \tag{2.17}$$

Thus, the smaller the norm of the normal vector, the higher the margin. Moreover, it was shown in [Cortes and Vapnik, 1995] that the VC dimension of a function class $\mathcal{F}_\Lambda$ which is restricted to canonical hyperplanes is bounded by

$$h \leq \min(\Lambda^2 R^2 + 1, d + 1) \text{ for } \|\mathbf{w}\| < \Lambda \tag{2.18}$$

where R is the radius of the smallest sphere containing the data. The advantage of using canonical hyperplanes becomes apparent now. By bounding the margin we can effectively reduce the influence of a growing dimensionality d.

Clearly, the radius R grows with the dimensionality d. On the other hand, it is easier to construct a large margin for high dimensional data. Thus, the VC dimension can be directly controlled by bounding the margin and this fact enables us to apply the SRM principle.

The SVM is an algorithm which is motivated by the described SRM principle and the link between the VC dimension and the margin. As argued above it is desirable to achieve a large margin restricted to canonical hyperplanes. This can be expressed as a quadratic optimization problem in the form

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.19}$$

$$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x} + b) \geq 1 \quad \forall i = 1, \ldots, M. \tag{2.20}$$

The constraint ensures that none of the training points produces an output greater than one. Thus, every hyperplane that is a solution of (2.19) is a canonical hyperplane as well. The optimization problem above is in its primal version and can be solved directly by a quadratic optimizer. An important property of this optimization problem is that due to its convexity all solutions are global solutions. This is very much in contrast to e. g. neural networks.

Another possibility to solve this optimization problem is to form its dual version. Due to the convexity of the quadratic problem the primal and the dual are closely connected, i. e. if the primal is infeasible then the dual is unbounded and vice versa. Furthermore, if both are feasible primal and dual reach the same objective function value at the optimal solution. We shall see in the next section that the dual version exhibits the important property that it allows to formulate SVMs in nonlinear spaces using kernel functions.

In order to form the dual we introduce Langrangian multipliers $\alpha_i \geq 0,\ i = 1, \ldots, M$ for each constraint in (2.19). The Lagrangian may then be

written as

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{M} \alpha_i (y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) \tag{2.21}$$

where $\boldsymbol{\alpha}$ contains all Lagrangian multipliers. The goal is to minimize the Lagrangian with respect to $\mathbf{w}$ and b and to maximize it with respect to α_i. At the optimal solution the following saddle point equations hold.

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{M} \alpha_i y_i = 0 \tag{2.22}$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{M} \alpha_i y_i \mathbf{x}_i. \tag{2.23}$$

Substituting (2.22) and (2.23) into the Lagrangian (2.21) yields the dual optimization problem

$$\max_{\boldsymbol{\alpha}} \left(\sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) \right) \tag{2.24}$$

$$\text{subject to} \quad \alpha_i \geq 0,\ i = 1, \ldots, M \tag{2.25}$$

$$\sum_{i=1}^{M} \alpha_i y_i = 0. \tag{2.26}$$

When we solve this optimization problem we obtain the Lagrangian coefficients α_i and the desired decision function is given by

$$f(\mathbf{x}) = \text{sign}\ (\mathbf{w}^T\mathbf{x} + b) \tag{2.27}$$

$$= \text{sign}\ \left(\sum_{i=1}^{M} \alpha_i y_i (\mathbf{x}^T \mathbf{x}_i) + b \right). \tag{2.28}$$

2.4.2 The Linearly Inseparable Case

So far we have only considered the linearly separable case where the empirical risk is zero. Unfortunately, in most practical cases this assumption does not hold and the optimization problem (2.24) has no feasible solutions at all. In

order to overcome this problem we may relax the hard-margin constraints (2.20) by introducing the so-called slack variables, i. e. we allow for some errors in the form

$$y_i(\mathbf{w}^T\mathbf{x} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, M. \tag{2.29}$$

These constraints are often called the soft-margin constraints. Now we allow that some training points lie inside the margin area. But note that as long as all slacks are smaller than one, this is still a linearly separable problem. The goal is to bound the VC dimension of our function class by maximizing the margin and at the same time to minimize the empirical risk that is given by $\sum_i \xi_i$. This can be expressed as a modified primal optimization problem in the form

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{M} \xi_i \tag{2.30}$$

where $C > 0$ is a free regularization parameter which determines the trade-off between the capacity of the function class and the empirical risk. Now the dual optimization problem reads

$$\max_{\boldsymbol{\alpha}} \left(\sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) \right) \tag{2.31}$$

$$\text{subject to} \quad \alpha_i \geq 0 \geq C, \; i = 1, \ldots, M \tag{2.32}$$

$$\sum_{i=1}^{M} \alpha_i y_i = 0. \tag{2.33}$$

The difference of this modified optimization problem to the formulation without slack variables is that the Lagrange parameters α_i are limited by the so called box constraints $\alpha_i \geq 0 \geq C, \; i = 1, \ldots, M$.

A very important property of the SVM is the sparsity of the solution. By sparsity we mean the fact that only a fraction of the training instances are used to construct the decision function. To see that the SVM solution

is sparse consider the Karush-Kuhn-Tucker conditions (KKT-conditions). These conditions are second order optimality conditions and are necessary and in many cases sufficient for most optimization problems. Fortunately, the KKT-conditions are of particular simplicity for the SVM case.

$$\alpha_i = 0 \quad \rightarrow \quad y_i f(\mathbf{x}_i) \geq 1 \quad \text{and} \quad \xi_i = 0 \tag{2.34}$$

$$0 < \alpha_i < C \quad \rightarrow \quad y_i f(\mathbf{x}_i) = 1 \quad \text{and} \quad \xi_i = 0 \tag{2.35}$$

$$\alpha_i = C \quad \rightarrow \quad y_i f(\mathbf{x}_i) \leq 1 \quad \text{and} \quad \xi_i \geq 0. \tag{2.36}$$

Thus, only the Lagrangian parameters α which correspond to training instances $\mathbf{x}_i$ on the edge of or inside the margin area have non-zero entries. These training points are called the support vectors and gave the SVM its name.

Intuitively speaking, the SVM has shown that sparsity is an indication for a good generalization ability. In the SVM case, sparsity followed directly from the proposed optimization problem that arose from the desired application of the SRM-principle. We will exploit the importance of sparsity for a good generalization in the next chapter.

2.5 Kernel Functions

The SVM case has shown that the restriction to linear functions is reasonable in order to control the complexity of the function class effectively and hence to achieve a good generalization ability. However, so far we have not discussed the ability of linear functions to minimize the empirical risk which is of the same importance for successful learning. It turns out, that linear functions perform poorly for most practical cases when it comes to minimize the empirical risk. For instance, consider the easy two-dimensional XOR-problem in Fig. 2.4.

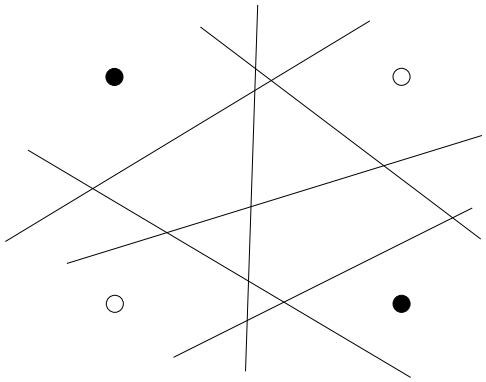

Figure 2.4: The simple XOR-problem: A correct classification of all outcomes of the binary XOR-function can not be achieved by linear functions.

We can see that there is no linear direction that is capable of classifying the binary outputs of the XOR-function correctly. Thus, the question is how we can enhance the empirical performance of linear functions *and* at the same time benefit from their theoretical properties.

2.5.1 The Kernel Trick

Clearly, we could first preprocess the data by a nonlinear mapping $\mathbf{\Phi}$ in the form

$$\mathbf{\Phi}: \mathbb{R}^d \rightarrow \mathcal{E} \tag{2.37}$$

$$\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x}) \tag{2.38}$$

in order to obtain sufficiently rich directions in the nonlinear feature space $\mathcal{E}$. Instead of working in the original input space $\mathcal{X}$ we may now apply a linear algorithm in the feature space $\mathcal{E}$ using the non-linearly mapped inputs

$$\{(\mathbf{\Phi}(\mathbf{x}_1), y_1), \ldots, (\mathbf{\Phi}(\mathbf{x}_M), y_M)\} \subseteq (\mathcal{E} \times \mathcal{Y})^M \tag{2.39}$$

The hope is that in the nonlinear feature space the problem becomes linearly separable like illustrated in Fig. 2.5. However, there are two problems with

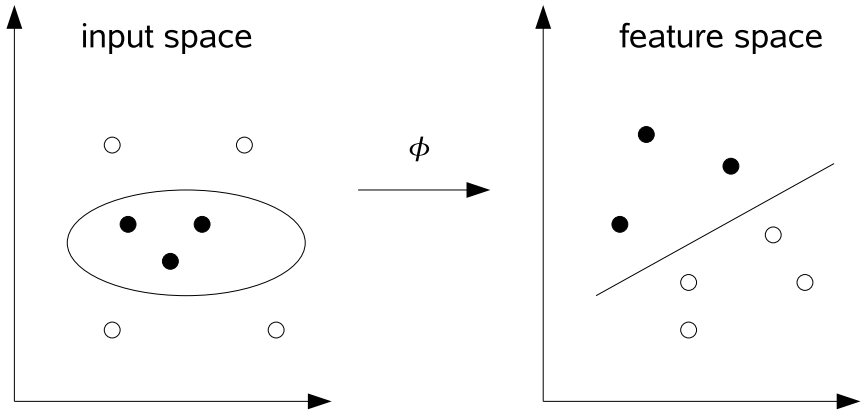

Figure 2.5: The data are mapped by a nonlinear mapping into the feature space and the problem may become linearly separable when we choose an appropriate nonlinear map. The linear directions in the new feature space correspond to nonlinear ones in the original input space.

this approach. First, prior knowledge is required about the problem at hand in order to construct an appropriate nonlinear mapping. Second, due to memory requirements we are restricted to nonlinear mappings which are not too high dimensional. To see this consider the following example. The n-th order monomials which are often used in image processing can easily be constructed for $n = 2$.

$$\mathbf{\Phi} : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \tag{2.40}$$

$$\mathbf{x} = (x_1, x_2)^T \rightarrow (z_1, z_2, z_3)^T := (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T \tag{2.41}$$

Clearly, in this case we may easily carry out the mapping $\mathbf{\Phi}$ directly on the data. However, the problem becomes intractable for higher order monomials. For instance, if the input space consists of high dimensional vectors (e. g. $16\times$ 16 pixels images resulting in 265-dimensional vectors) then evaluating all 5th

order monomials would amount to working in a feature space with

$$\binom{265+5-1}{5} \approx 10^{10} \tag{2.42}$$

dimensions. This is clearly infeasible. Now the kernel functions come into play. It turns out that we can compute scalar products between non-linearly mapped inputs using a kernel function k. It is easy to show this fact for the second order monomials from above.

$$\begin{aligned}
\mathbf{\Phi}^T(\mathbf{x})\mathbf{\Phi}(\mathbf{x}^{'}) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(x_1^{'2}, \sqrt{2}x_1^{'}x_2^{'}, x_2^{'2})^T && (2.43)\\
&= ((x_1, x_2)(x_1^{'}, x_2^{'})^T)^2 && (2.44)\\
&= (\mathbf{x}^T\mathbf{x}^{'})^2 && (2.45)\\
&= k(\mathbf{x}, \mathbf{x}^{'}) && (2.46)
\end{aligned}$$

This generalizes to all n-th order monomials, i. e. the kernel function

$$k(\mathbf{x}, \mathbf{x}^{'}) = (\mathbf{x}^T\mathbf{x}^{'})^n \tag{2.47}$$

computes a scalar product in the space of all n-th order monomials [Schölkopf and Smola, 2002]. This is an example of the so-called *kernel trick*. Speaking more generally, the kernel trick is to formulate an algorithm exclusively in terms of scalar products and to replace the scalar products by kernel functions. Thereby we are able to perform the algorithm in the feature space $\mathcal{E}$ without even knowing the underlying map $\mathbf{\Phi}$ since $\mathbf{\Phi}$ is implicitly induced by the kernel function. Now it is apparent why we formulated the SVM optimization problem in its dual version (2.24). The dual version contains the input data exclusively in term of scalar products $\mathbf{x}_i^T\mathbf{x}_j$ which can be replaced by $k(\mathbf{x}_i, \mathbf{x}_j)$ and the SVM may be solved linearly in $\mathcal{E}$ yielding nonlinear directions in the original input space $\mathcal{X}$.

2.5.2 Kernel-Induced Feature Spaces

Now the question arises under which conditions a kernel function corresponds to scalar products of non-linearly mapped inputs, i. e. when is a kernel a valid kernel? It will turn out that the answer is simple. Every symmetric, positive definite function is a valid kernel. In order to justify this theoretically we have to show that the underlying nonlinear mapping associated with such a kernel always exist. In the following we will outline two theoretical reasonings that are often used to identify these kernel-induced feature spaces. For further details see e. g. [Schölkopf and Smola, 2002] and the references therein.

Let us first introduce some notations and definitions.

Definition 1 A function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ of two variables is called a *positive definite kernel* iff it is symmetric, that is, $k(\mathbf{x}^{'}, \mathbf{x}) = k(\mathbf{x}, \mathbf{x}^{'})$ for any two objects $\mathbf{x}, \mathbf{x}^{'} \in \mathcal{X}$ and positive definite, that is,

$$\sum_{i,j=1}^{M} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \tag{2.48}$$

for any $M > 0$, any choice of objects $\mathbf{x}_1, \ldots, \mathbf{x}_M \in \mathcal{X}$ and any choice of real numbers $\alpha_1, \ldots, \alpha_M \in \mathbb{R}$. □

If the last inequality is always strictly positive k is called a *strictly positive definite* kernel. In particular, positive definite kernels are exactly those giving rise to a positive definite Gram matrix or kernel matrix $\mathbf{K}$ with the elements $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. Note, that for a matrix to be positive definite, it is necessary to be symmetric and non-negative on the diagonal.

Reproducing Kernels

Given a real-valued, positive definite kernel function k we are able to define a functional space $\mathcal{H}$ as a set of functions $f: \mathcal{X} \to \mathbb{R}$ of the form

$$f(\mathbf{x}) = \sum_{i=1}^{M} \alpha_i k(\mathbf{x}_i, \mathbf{x}), \tag{2.49}$$

for $M > 0$, $\alpha_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathcal{X}$ together with their limits under the norm

$$\|f\|_{\mathcal{H}}^2 = \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j). \tag{2.50}$$

It can be shown that this norm is independent of the representation (2.49) of $f(\mathbf{x})$. Furthermore, $\mathcal{H}$ is a Hilbert space with a dot product defined for two elements $f(\mathbf{x}) = \sum_i^N \alpha_i k(\mathbf{x}_i, \mathbf{x})$ and $g(\mathbf{x}) = \sum_j^M \alpha_j' k(\mathbf{x}_j', \mathbf{x})$ by

$$\langle f, g \rangle = \sum_{i}^{N} \sum_{j}^{M} \alpha_i \alpha_j' k(\mathbf{x}_i, \mathbf{x}_j'). \tag{2.51}$$

An important property of this construction is that the value $f(\mathbf{x})$ of a function $f \in \mathcal{H}$ at a point $\mathbf{x} \in \mathcal{X}$ can be expressed as a dot product in $\mathcal{H}$,

$$f(\mathbf{x}) = \langle f, k(\cdot, \mathbf{x}) \rangle \tag{2.52}$$

where we denote by $k(\cdot, \mathbf{x})$ the kernel where the first argument is free and the second is fixed to $\mathbf{x}$.

In particular, taking $f(\cdot) = k(\cdot, \mathbf{x}')$ we find the following reproducing property

$$\Big\langle k(\cdot, \mathbf{x}'), k(\cdot, \mathbf{x}) \Big\rangle = k(\mathbf{x}', \mathbf{x}). \tag{2.53}$$

The last equality shows one possible way to identify a feature space associated with the kernel k. When we define the feature map $\mathbf{\Phi}$ as

$$\mathbf{\Phi} : \mathcal{X} \to \mathcal{H} \text{ with } \mathbf{\Phi}(\mathbf{x}) = k(\cdot, \mathbf{x}) \tag{2.54}$$

we see by equation (2.53) that the kernel k acts as a dot product of $\mathbf{\Phi}(\mathbf{x})$ and $\mathbf{\Phi}(\mathbf{x}^{'})$. Thus, we can take the Hilbert space $\mathcal{H}$ as one realization of the feature space $\mathcal{E}$ associated with the kernel k. A Hilbert space $\mathcal{H}$ constructed in this way is called a Reproducing Kernel Hilbert Space (RKHS). In the following we give a formal definition of a RKHS [Aronszajn, 1950].

Definition 2 (Reproducing Kernel Hilbert Space (RKHS)) Let $\mathcal{X}$ be a nonempty 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 (RKHS) endowed with the dot product $\langle \cdot, \cdot \rangle$ if there exists a function k with the properties that

1. k has the reproducing property $f(\mathbf{x}) = \langle f, k(\cdot, \mathbf{x}) \rangle$ for all $f \in \mathcal{H}$, in particular, $\langle k(\cdot, \mathbf{x}^{'}), k(\cdot, \mathbf{x}) \rangle$
2. k spans $\mathcal{H}$, i. e. , $\mathcal{H} = \text{span}\{k(\cdot, \mathbf{x}) | \mathbf{x} \in \mathcal{X}\}$. □

It can be shown that the kernel k for such a RKHS is uniquely determined [Aronszajn, 1950].

Mercer Kernels

A second way to identify a feature space associated with a kernel is based on Mercer's Theorem [Mercer, 1909].

Theorem 4 (Mercer's Theorem) *Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^d$ and $L^2(\mathcal{X})$ be the space of square integrable functions f over $\mathcal{X}$. Furthermore, let $k(\cdot, \cdot)$ be a continuous symmetric function such that the integral operator $T_k : L^2(\mathcal{X}) \to L^2(\mathcal{X})$,*

$$T_k f(\mathbf{x}) = \int_{\mathcal{X}} k(\mathbf{x}, \mathbf{x}^{'}) f(\mathbf{x}^{'}) d\mathbf{x}^{'} \tag{2.55}$$

is positive, that is

$$\int \int_{\mathcal{X} \times \mathcal{X}} k(\mathbf{x}, \mathbf{x}^{'}) g(\mathbf{x}^{'}) g(\mathbf{x}) d\mathbf{x}^{'} d\mathbf{x} \geq 0 \tag{2.56}$$

for all $g \in L^2(\mathcal{X})$, then we can expand $k(\mathbf{x}, \mathbf{x}')$ in a uniform convergent series on $\mathcal{X} \times \mathcal{X}$ as

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d_{\mathcal{H}}} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{x}') \tag{2.57}$$

where $\{\psi_i\}_{i=1}^{d_{\mathcal{H}}} \in L^2(\mathcal{X})$ is an orthogonal set of eigenfunctions of the integral operator T_k normalized in such way that $\|\psi_i\|_{L_2} = 1$. The $\{\lambda_i\}_{i=1}^{d_{\mathcal{H}}}$ are the corresponding positive associated eigenvalues of the integral operator T_k where $d_{\mathcal{H}}$, the dimension of this Hilbert space, is either $d_{\mathcal{H}} \in \mathbb{N}$ or $d_{\mathcal{H}} = \infty$. The function k is called a Mercer kernel. □

Now if we take $\mathcal{H} = L_2^{d_{\mathcal{H}}}$ and the mapping $\mathbf{\Phi}$ as

$$\mathcal{X} \to L_2^{d_{\mathcal{H}}}, \ \mathbf{\Phi}(\mathbf{x}) = (\sqrt{\lambda_i}\psi_i(\mathbf{x}))_{i=1,\ldots,d_{\mathcal{H}}} \tag{2.58}$$

we see from the expansion (2.57) that the Mercer kernel k acts as a dot product in $L_2^{d_{\mathcal{H}}}$.

It can be shown that a kernel is a Mercer kernel if and only if it is a positive definite kernel. Furthermore, there is a close connection between Mercer kernels and RKHSs. It turns out that for every Mercer kernel k defined over $\mathcal{X} \subset \mathbb{R}^d$ there exists a RKHS $\mathcal{H}$ of functions defined over $\mathcal{X}$ for which k is the reproducing kernel. Remarkably, the converse also holds [Christianini and Shawe-Taylor, 2000]. For any RKHS the corresponding reproducing kernel is a Mercer kernel.

Some most widely used kernel functions are given below.

$$\text{Gaussian kernel (RBF-kernel):} \quad k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{2.59}$$

$$\text{Polynomial kernel:} \quad k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + \theta)^d \tag{2.60}$$

$$\text{Sigmoidal kernel:} \quad k(\mathbf{x}, \mathbf{x}') = \tanh(\kappa\mathbf{x}^T\mathbf{x}' + \theta) \tag{2.61}$$

Note that the dimension of the feature space associated with the Gaussian kernel is infinite dimensional.

2.6 Summary

We have demonstrated by means of statistical learning theory that the restriction of the function class our learning machine can implement is crucial for successful learning. This fact served as a theoretical justification for choosing linear functions for learning since the capacity of linear functions can easily be controlled. We presented the SVM as an example for how these insights can be exploited algorithmically. In the SVM-case the capacity of the function class is controlled by defining canonical hyperplanes and then maximizing the margin. The underlying principle is the so-called structural risk minimization. However, the problem with linear functions is that their ability to minimize the empirical risk is very poor. In order to overcome this problem one can use kernel functions. Kernel functions allow an elegant transformation of any linear algorithm into a nonlinear one as long as one can express the algorithm exclusively in terms of dot products. The advantage of such a construction is that the transformed algorithm is still linear in some nonlinear space. Thereby we are able to have both a good generalization and good ability to minimize the empirical risk.

Chapter 3

Discriminants

The discriminative approach has a long historical tradition. Starting with Linear Discriminants (LDs) which were first introduced in [Fisher, 1936] we will show that LDs have very motivating and appealing statistical properties in case of classification along with a clear geometrical interpretation. The equivalence of the discriminative approach to a least-squares regression and the possibility of constructing nonlinear discriminants using Mercer kernel functions lay the foundations for the derivation of some incremental learning algorithms for nonlinear discriminants. For convenience we will concentrate on discriminants for two-class (binary) problems but as we shall see later the multiclass problem may easily be reduced to binary classification problems.

3.1 Linear Discriminants

Consider a training set $\mathbf{X} = \{\mathbf{X}_+, \mathbf{X}_-\}$ belonging to an input space $\mathcal{X}$ and consisting of M samples which are split into two classes. Let the classes be labeled with -1 and 1 defining a corresponding label vector $\mathbf{y} = \{-1, 1\}^M$. The number of samples labeled with 1 and -1 is $|\mathbf{X}_+| = M_+$ and $|\mathbf{X}_-| = M_-$,

respectively. The corresponding class means are

$$\mathbf{m}_{\pm} = \frac{1}{M_{\pm}} \sum_{\mathbf{x} \in \mathbf{X}_{\pm}} \mathbf{x}. \tag{3.1}$$

Fisher's idea was to classify the training samples by finding a direction $\mathbf{w}$ which separates the class means and at the same time minimizes the variances within the classes after projection onto $\mathbf{w}$. Thus, we need a measure that quantifies how far the class means are separated *and* how compact the projected training instances that belong to the same class are. Such a measure is the so-called Rayleigh quotient. In particular, we intend to obtain a one-dimensional discriminative function

$$f: \quad \mathcal{X} \to \mathbb{R}; \quad f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} \tag{3.2}$$

such that the so-called Rayleigh quotient

$$R(\mathbf{w}) = \frac{(\mu_+ - \mu_-)^2}{\sigma_+ + \sigma_-} \tag{3.3}$$

is maximized with

$$\mu_{\pm} = \mathbf{w}^T \mathbf{m}_{\pm} \quad \text{and} \quad \sigma_{\pm} = \sum_{\mathbf{x} \in \mathbf{X}_{\pm}} (\mathbf{w}^T \mathbf{w} - \mu_{\pm})^2 \tag{3.4}$$

denoting the means and unnormalized variances of the corresponding training samples after projection onto $\mathbf{w}$. Inserting (3.4) into (3.3) yields

$$R(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{3.5}$$

with

$$\mathbf{S}_B = (\mathbf{m}_+ - \mathbf{m}_-)(\mathbf{m}_+ - \mathbf{m}_-)^T \tag{3.6}$$

and

$$\mathbf{S}_W = \sum_{i, y_i = -1} (\mathbf{x}_i - \mathbf{m}_-)(\mathbf{x}_i - \mathbf{m}_-)^T + \sum_{i, y_i = 1} (\mathbf{x}_i - \mathbf{m}_+)(\mathbf{x}_i - \mathbf{m}_+)^T \tag{3.7}$$

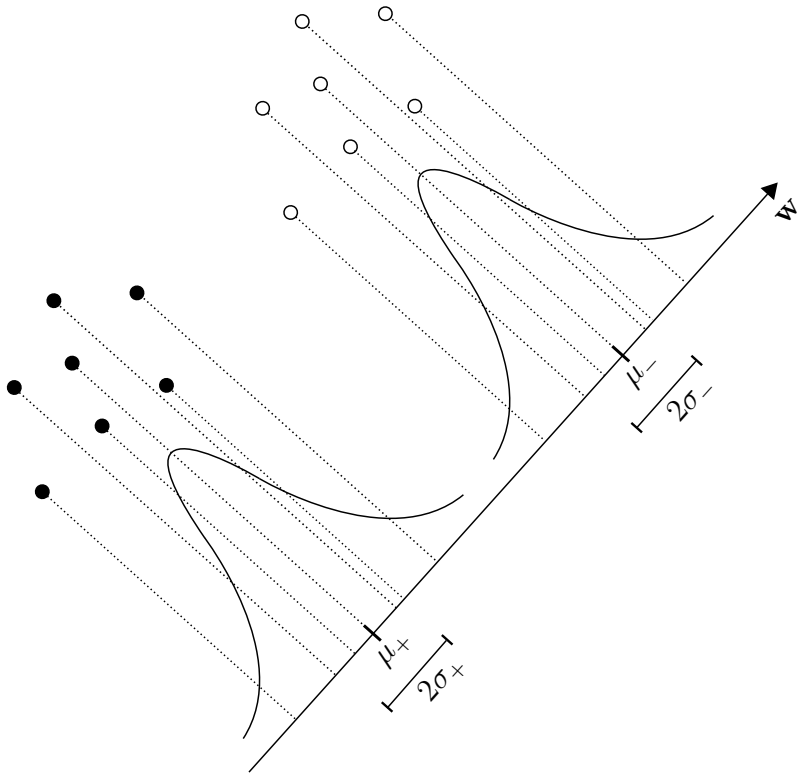

Figure 3.1: Geometrical interpretation of the LD approach. The two classes (black and white circles) are projected (dashed lines) onto the discriminating direction $\mathbf{w}$ such that the variances $\sigma_\pm$ within the classes are minimized and the means $\mu_\pm$ of the two projected classes are maximally separated.

denoting the unnormalized between-class and within-class covariance matrices (often referred to as scatter matrices), respectively. The geometrical interpretation of the LD approach is illustrated in figure 3.1.

Differentiating (3.5) with respect to $\mathbf{w}$ leads to the generalized eigenvalue problem

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w} \tag{3.8}$$

with

$$\lambda = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} = R(\mathbf{w}). \tag{3.9}$$

Thus, $\mathbf{w}$ must be a generalized eigenvector of (3.8). To see that the leading eigenvector (the eigenvector that corresponds to the largest eigenvalue λ_{max}) of (3.8) is the optimal (optimal in the sense that (3.8) is maximized) solution $\mathbf{w}_{opt}$ of (3.8), we assume the converse, i. e. , we assume that there exists another eigenvalue $\hat{\lambda} < \lambda_{max}$ with a corresponding eigenvector $\hat{\mathbf{w}}$ such that $R(\mathbf{w}_{opt}) < \hat{\lambda}$. Then by definition

$$\hat{\lambda} = \frac{\hat{\mathbf{w}}^T \mathbf{S}_B \hat{\mathbf{w}}}{\hat{\mathbf{w}}^T \mathbf{S}_W \hat{\mathbf{w}}} = R(\hat{\mathbf{w}}) > R(\mathbf{w}_{opt}). \tag{3.10}$$

The last inequality is a contradiction to our assumption that $\mathbf{w}_{opt}$ is the optimal solution of (3.8) and hence the leading eigenvector of (3.8) is the optimal solution. However, note that $\mathbf{w}_{opt}$ is not unique. We see that by definition of the Rayleigh coefficient $R(\mathbf{w})$ only the direction of $\mathbf{w}$ matters and not its length. Thus, every scaled version of $\mathbf{w}_{opt}$ is also a solution of (3.8). However, all these solutions are equivalent to each other since the scaling factor has no impact on the resulting discriminative function. For instance,

$$\mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{m}_+ - \mathbf{m}_-) \tag{3.11}$$

is a solution of (3.8) since by definition of $\mathbf{S}_B$ the vector $\mathbf{S}_B\mathbf{w}$ lies in the direction of $\mathbf{m}_+ - \mathbf{m}_-$.

The existence of a global solution makes the LD a very motivating approach. Furthermore, the examination of the Bayes optimality of the LD shows that this approach yields an optimal solution for normally distributed classes assuming equally structured covariance matrices [Duda and Hart, 1973] [Bishop, 1995]. This can be shown as follows.

If our assumption holds that the classes are normally distributed with the same covariance matrix $\mathbf{\Sigma}$ we can write the class-conditional densities in the

form

$$p(\mathbf{x}|y=1) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_+)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{m}_+)\right), \quad (3.12)$$

$$p(\mathbf{x}|y=-1) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_-)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{m}_-)\right). \quad (3.13)$$

Without loss of generality the posterior probability for the case $y = 1$ can then be computed using Bayes' law yielding

$$P(y=1|\mathbf{x}) = \frac{p(\mathbf{x}|y=1)P(y=1)}{p(\mathbf{x}|y=1)P(y=1) + p(\mathbf{x}|y=-1)P(y=-1)} \quad (3.14)$$

which can be expressed as a standard logistic function in the following form

$$P(y=1|\mathbf{x}) = \frac{1}{1+\exp(-t)} \quad (3.15)$$

with

$$t = \log\left(\frac{p(\mathbf{x}|y=1)P(y=1)}{p(\mathbf{x}|y=-1)P(y=-1)}\right). \quad (3.16)$$

Inserting (3.12) and (3.13) into (3.15) we obtain

$$t = \mathbf{w}^T\mathbf{x} + b \quad (3.17)$$

with the linear direction

$$\mathbf{w} = \mathbf{\Sigma}^{-1}(\mathbf{m}_+ - \mathbf{m}_-) \quad (3.18)$$

and the bias

$$b = \frac{1}{2}\mathbf{m}_-^T\mathbf{\Sigma}^{-1}\mathbf{m}_- - \frac{1}{2}\mathbf{m}_+^T\mathbf{\Sigma}^{-1}\mathbf{m}_+ + \log\left(\frac{P(y=1)}{P(y=-1)}\right). \quad (3.19)$$

Since the covariance matrix $\mathbf{\Sigma}$ is a scaled version of the scatter matrix which we defined in (3.32) the solution (3.18) is up to a scaling factor identical with the solution (3.11). As argued above, a scaling factor is not relevant for the resulting discriminative function and hence the LD approach is Bayes optimal for this case.

3.2 Equivalence to Least-Squares

Another interesting and for the following derivations crucial fact about LDs is their equivalence to a standard least-squares regression onto the labels [Duda and Hart, 1973]. Least-squares regression aims to find the weight vector $\tilde{\mathbf{w}} \in \mathbb{R}^{M+1}$ which minimizes the squared residual

$$\tilde{\mathbf{e}} = \|\tilde{\mathbf{X}}\tilde{\mathbf{w}} - \mathbf{y}\|^2 = (\tilde{\mathbf{X}}\tilde{\mathbf{w}} - \mathbf{y})^T(\tilde{\mathbf{X}}\tilde{\mathbf{w}} - \mathbf{y}) \tag{3.20}$$

where the matrix $\tilde{\mathbf{X}} = [\mathbf{X}, \mathbf{1}]$ is constructed from the original data matrix $\mathbf{X}$ by adding an additional column $\mathbf{1}$ consisting of ones and the new weight vector $\tilde{\mathbf{w}} = (\mathbf{w}; b)$ contains a bias b. Differentiating (3.20) with respect to $\tilde{\mathbf{w}}$ gives rise to the so called normal equation

$$\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\tilde{\mathbf{w}} - \tilde{\mathbf{X}}^T\mathbf{y} = \mathbf{0} \tag{3.21}$$

which is the necessary and sufficient condition for the minimum. It follows directly from (3.21) that $\tilde{\mathbf{w}}$ is given by

$$\tilde{\mathbf{w}} = (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{y} = \mathbf{X}^\dagger\mathbf{y} \tag{3.22}$$

provided that the matrix $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ is not singular, i. e. , provided that the columns of $\tilde{\mathbf{X}}$ are linearly independent. The matrix $\mathbf{X}^\dagger$ is called the Moore-Penrose-Inverse or pseudoinverse of $\mathbf{X}$ and yields an unbiased estimation of $\tilde{\mathbf{w}}$ with the smallest Euclidean norm. In order to see that this least-squares solution is equivalent to LDs given by (3.11) let us rewrite (3.20) in the following form

$$\tilde{\mathbf{e}} = \left\| \begin{bmatrix} \mathbf{X}_+^T & \mathbf{1}_+ \\ \mathbf{X}_-^T & \mathbf{1}_- \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} - \begin{bmatrix} -\mathbf{1}_+ \\ \mathbf{1}_- \end{bmatrix} \right\|^2 . \tag{3.23}$$

Here we assume that the target vector $\mathbf{y}$ contains the binary class labels $\{1, -1\}$ and hence the vector $\mathbf{1}_\pm = 1, \ldots, 1$ has the length $M_\pm$.

Then the normal equation reads

$$\begin{bmatrix} \mathbf{X}_+ & \mathbf{X}_- \\ \mathbf{1}_+^T & \mathbf{1}_-^T \end{bmatrix} \begin{bmatrix} \mathbf{X}_+^T & \mathbf{1}_+ \\ \mathbf{X}_-^T & \mathbf{1}_- \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{X}_+ & \mathbf{X}_- \\ \mathbf{1}_+^T & \mathbf{1}_-^T \end{bmatrix} \begin{bmatrix} -\mathbf{1}_+ \\ \mathbf{1}_- \end{bmatrix}. \tag{3.24}$$

When we use the definition of the within-class scatter matrix and the sample means the normal equation can be written as

$$\begin{bmatrix} \mathbf{S}_W + \mathbf{m}_+\mathbf{m}_-^T & M_+\mathbf{m}_+ + M_-\mathbf{m}_- \\ (M_+\mathbf{m}_+ + M_-\mathbf{m}_-)^T & M_+ + M_- \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} M_-\mathbf{m}_- - M_+\mathbf{m}_+ \\ M_- - M_+ \end{bmatrix}. \tag{3.25}$$

Solving the second equation for b yields

$$b = \frac{M_- - M_+ - (M_+\mathbf{m}_+ + M_-\mathbf{m}_-)^T\mathbf{w}}{M_+ + M_-} \tag{3.26}$$

and inserting this into the first equation we obtain

$$\mathbf{S}_W\mathbf{w} = -\left(\frac{M_+M_-}{M_+ + M_-}\mathbf{S}_B\mathbf{w} + \frac{M_+^2 + M_-^2}{M_+ + M_-}(\mathbf{m}_- - \mathbf{m}_+)\right) \tag{3.27}$$

and since by definition of $\mathbf{S}_B$ the vector $\mathbf{S}_B\mathbf{w}$ must lie in the direction of $\mathbf{m}_- - \mathbf{m}_+$ the least-squares solution

$$\mathbf{w} = \eta\mathbf{S}_W^{-1}(\mathbf{m}_- - \mathbf{m}_+) \tag{3.28}$$

is once again up to an unimportant scaling factor η identical with the LD solution (3.11). The equivalence of the LDs to least-squares regression is another explanation of their Bayes optimality because there is a close connection between the loss function we choose and the assumed noise model for our predictions. It is well known that the squared loss function corresponds to a Gaussian noise model - the same noise model we assumed when proving the Bayes optimality of LDs. Hence, the least-squares model can be regarded as the maximum a posteriori estimator corresponding to a probability model with gaussian noise and gaussian weight prior.

3.3 Kernel-Based Discriminants

In the previous chapter we stated that controlling the complexity of the function class the learning machine can implement is mandatory for successful learning. Due to this theoretical insight linear functions with a controllable complexity are an appropriate choice for both constructing SVMs and discriminants. However, the restriction to linear directions in the input space is a drawback at the same time. In the previous chapter we saw that linear functions are not able to solve even the simple XOR-problem for all possible labelings. Thus, the question arises if we are able to obtain sufficiently rich discrimination directions without setting the theoretical benefits of linear functions aside. The answer is yes if we use kernel functions which we introduced in the previous chapter. Not surprisingly a kernel-based reformulation of the LD approach was proposed at the same time by several researchers [Mika, 2002] [Baudat and Anouar, 2000] [Roth and Steinhage, 1999].

Instead of working in the original input space $\mathcal{X}$ a nonlinear mapping $\mathbf{\Phi} : \mathcal{X} \rightarrow \mathcal{E}$ is applied to the data $\mathbf{X}$. Now the LD can be linearly performed with the mapped samples $\{\mathbf{\Phi}(\mathbf{x}_1), \mathbf{\Phi}(\mathbf{x}_2), \ldots, \mathbf{\Phi}(\mathbf{x}_M)\} \in \mathcal{E}$ in the feature space $\mathcal{E}$ yielding nonlinear directions in the input space $\mathcal{X}$. Note that our goal is not to explicitly define the mapping $\mathbf{\Phi}$ since in the following derivations we are rather interested in applying the kernel trick to LDs, i. e. , in replacing the dot-products $\mathbf{\Phi}^T(\mathbf{x})\mathbf{\Phi}(\mathbf{x})$ by kernel functions. The only thing we have to be sure of concerning $\mathbf{\Phi}$ is that it exists and as we have seen in the last chapter, the use of Mercer kernels guarantees the existence of such a mapping.

Within this framework we are able to formulate a generalized version of

the Rayleigh coefficient in the feature space $\mathcal{E}$ in the form

$$R^{\mathbf{\Phi}}(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B^{\mathbf{\Phi}} \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W^{\mathbf{\Phi}} \mathbf{w}} \tag{3.29}$$

where

$$\mathbf{S}_B^{\mathbf{\Phi}} = (\mathbf{m}_+^{\mathbf{\Phi}} - \mathbf{m}_-^{\mathbf{\Phi}})(\mathbf{m}_+^{\mathbf{\Phi}} - \mathbf{m}_-^{\mathbf{\Phi}})^T \tag{3.30}$$

with

$$\mathbf{m}_\pm^{\mathbf{\Phi}} = \frac{1}{M_\pm} \sum_{\mathbf{x} \in \mathbf{X}_\pm} \mathbf{\Phi}(\mathbf{x}) \tag{3.31}$$

and

$$\mathbf{S}_W^{\mathbf{\Phi}} = \sum_{i, y_i = -1} (\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{m}_-^{\mathbf{\Phi}})(\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{m}_-^{\mathbf{\Phi}})^T + \sum_{i, y_i = 1} (\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{m}_+^{\mathbf{\Phi}})(\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{m}_+^{\mathbf{\Phi}})^T \tag{3.32}$$

However, there is a problem with this approach. Since the feature space $\mathcal{E}$ has usually a very high or even infinite dimension we are not able to compute the discriminating direction $\mathbf{w}$ directly since $\mathbf{w}$ is of the same dimension. But as we will see $\mathbf{w}$ lies in the span of $\mathbf{\Phi}(\mathbf{x}_i), \quad i = 1, \ldots, M$ and hence takes the form

$$\mathbf{w} = \sum_{i=1}^{M} \alpha_i \mathbf{\Phi}(\mathbf{x}_i) \tag{3.33}$$

for some $\boldsymbol{\alpha} \in \mathbb{R}^M$. We will show this in the following.

Given the training data $\mathbf{X}$, any $f \in \mathcal{E}$ can be decomposed into a linear subspace $f_\parallel$ which is spanned by all mapped training instances $\mathbf{\Phi}(\mathbf{x}_i)$ and its orthogonal complement $f_\perp$, which satisfies $\langle f_\perp, \Phi(\mathbf{x}_i) \rangle = 0, \quad \forall \mathbf{x}_i \in \mathbf{X}$. Then the application of f for an arbitrary training instance $\mathbf{x}_j$ yields

$$\begin{aligned} f(\mathbf{x}_j) &= \left\langle f_\parallel + f_\perp, \mathbf{\Phi}(\mathbf{x}_j) \right\rangle = \left\langle \left(\sum_{i=1}^{M} \alpha_i \mathbf{\Phi}(\mathbf{x}_i) + f_\perp \right), \mathbf{\Phi}(\mathbf{x}_j) \right\rangle \\ &= \sum_{i=1}^{M} \alpha_i \left\langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \right\rangle \\ &= \sum_{i=1}^{M} \alpha_i k(\mathbf{x}_i, \mathbf{x}_j). \end{aligned} \tag{3.34}$$

The last equation follows from the fact that we use reproducing kernels. We see that it suffices to consider only the part of $\mathbf{w}$ which lies in the span of $\mathbf{\Phi}(\mathbf{x}_i)$. Furthermore, using the theory of reproducing kernels [Mercer, 1909] which shows that $\mathcal{E}$ is induced by positive definite kernel functions defining the inner product

$$k(\mathbf{x}, \mathbf{x}') = \left\langle \mathbf{\Phi}(\mathbf{x}), \mathbf{\Phi}(\mathbf{x}') \right\rangle \tag{3.35}$$

we are able to avoid to perform the mapping $\mathbf{\Phi}$ explicitly. This is often referred to as the *kernel trick*. At this point it is possible to re-formulate the generalized Rayleigh coefficient exclusively in terms of dot products without even knowing $\mathbf{\Phi}$. Following [Mika, 2002] we can write the between-class and within-class scatter matrices $\mathbf{S}_B^{\mathbf{\Phi}}$ and $\mathbf{S}_W^{\mathbf{\Phi}}$ exclusively in terms of dot products $\left\langle \mathbf{\Phi}(\mathbf{x}), \mathbf{\Phi}(\mathbf{x}') \right\rangle$. Thus, after some algebraic manipulations we are able to formulate these matrices in the feature space $\mathcal{E}$ using kernel functions in the form

$$\mathbf{S}_B^k = (\mathbf{m}_+^k - \mathbf{m}_-^k)(\mathbf{m}_+^k - \mathbf{m}_-^k)^T \tag{3.36}$$

with

$$(\mathbf{m}_\pm^k)_j = \frac{1}{M_\pm} \sum_{\mathbf{x} \in \mathbf{X}_\pm} k(\mathbf{x}, \mathbf{x}_j) \tag{3.37}$$

and

$$\mathbf{S}_W^k = \mathbf{K}_+(\mathbf{I}_+ - \mathbf{M}_+)\mathbf{K}_+^T + \mathbf{K}_-(\mathbf{I}_- - \mathbf{M}_-)\mathbf{K}_-^T. \tag{3.38}$$

The matrices $\mathbf{K}_\pm$ have the size $M \times M_\pm$ with elements

$$(\mathbf{K}_\pm)_{ij} = k(\mathbf{x}_i, \mathbf{x}_j),\ \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_\pm. \tag{3.39}$$

$\mathbf{I}_\pm$ are identity matrices with the size $M_\pm \times M_\pm$ and $\mathbf{M}_\pm$ are the matrices with all elements set to $\frac{1}{M_\pm}$ with the size $M_\pm \times M_\pm$. Then, using these *kernelized* matrices the generalized Rayleigh coefficient takes the form

$$R^k(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T \mathbf{S}_B^k \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \mathbf{S}_W^k \boldsymbol{\alpha}} \tag{3.40}$$

and again we arrived at a generalized eigenvalue problem which can be solved by finding the leading eigenvector of the matrix $(\mathbf{S}_W^k)^{-1}\mathbf{S}_B^k$. Alternatively, as we have argued for the case of LDs that only the direction and not the length of the discriminating direction is important we can obtain an equivalent solution by

$$\boldsymbol{\alpha} = (\mathbf{S}_W^k)^{-1}(\mathbf{m}_+^k - \mathbf{m}_-^k) \tag{3.41}$$

and the projection of a new sample $\mathbf{x}$ onto $\mathbf{w}$ is given by

$$\boldsymbol{\Phi}^T(\mathbf{x})\mathbf{w} = \sum_{i=1}^{M} \alpha_i k(\mathbf{x}, \mathbf{x}_i). \tag{3.42}$$

Kernel-based Discriminants are often called Kernel Fisher Discriminants (KFD) in the literature [Mika, 2002] [Baudat and Anouar, 2000] [Roth and Steinhage, 1999].

3.3.1 Regularization

However, there is a problem with this approach. From the last equation we see that the complexity of the discriminant function scales with the number of training instances. Thus, all training instances are used in the testing phase which is highly undesirable. Moreover, in order to avoid overfitting and numerical instabilities (note that the matrix $\mathbf{S}_W^k$ is at most of rank $(M-2)$) we have to employ some form of regularization. The proof that the KFD is equivalent to a least-squares regression onto the labels is completely analogous to the presented proof concerning the equivalence between LDs and least-squares. We can use this equivalence to obtain a regularized version of the KFD using regularized least-squares which is also called Ridge regression. The idea is to incorporate regularization controlled by a continuous parameter λ into the model yielding a regularized squared loss in the form

$$L(\boldsymbol{\alpha}) = \|\mathbf{K}\boldsymbol{\alpha} - \mathbf{y}\|^2 + \lambda\|\boldsymbol{\alpha}\|^2. \tag{3.43}$$

Ridge regression [Rifkin et al., 2003] penalizes the norm of the solution yielding flat directions in the RKHS, which are robust against outliers caused by e. g. noise. Another possibility to introduce regularization is to use the so-called Least-Squares SVMs (LS-SVMs), which were proposed in [Suykens and Vandewalle, 1999]. The close relation between the LS-SVM and the Kernel Fisher Discriminant (KFD) was shown in [Van Gestel et al., 2002]. While the SVM has a large margin interpretation the LS-SVM formulation is related to the Ridge regression approach for classification with binary targets and to the KFD. The optimization problem associated with LS-SVMs contains equality constraints. This leads to a linear set of equations in the dual space which can be solved using e. g. the conjugate gradient method for large data sets or a direct method for a small number of data. The solution may then be pruned [De Kruif and De Vries, 2003] [Hoegaerts et al., 2004] in a second stage to obtain a sparse solution.

As a conclusion we can state that all the approaches we discussed so far are closely related to each other and can be regarded as instances of Least-Squares Models (LSMs). However, in contrast to SVMs, due to the use of the squared loss LSMs are not sparse in general and the methods of direct regularization mentioned above are not able to regain sparsity. In the following we will outline some approaches that aim at introducing sparsity for LSMs.

3.4 Sparse Approximations

One way to obtain a sparse solution is to impose a sparsity constraint into the problem itself. The cleanest way would be to replace the regularization term in (3.43) by a L_0-norm regularizer in the form $\lambda\|\boldsymbol{\alpha}\|_0$. The constraint would

then be the number of nonzero elements in the weight vector. However, this approach is algorithmically intractable since it yields a NP-hard combinatorial search problem [Natarajan, 1995]. One way to overcome this problem is to look for a solution where the most points are separated linearly. This is a NP-hard problem of the same structure. However, approximate solutions exist without changing the L_0-norm [Wendemuth, 1995]. Another way to proceed is to approximate the L_0-norm by L_1-norm regularization like in basis pursuit [Chen et al., 1998]. This approach favors solutions with a small L_1-norm. The resulting problem is a convex programming problem with a unique and sparse solution.

These direct approaches are not pursued here. In the following we restrict ourselves to greedy methods which impose sparsity on the solution using subset selection like in [Billings and Lee, 2002] [Nair et al., 2002]. The difference between subset selection and a direct convex programming approach is that the sparsity is directly controlled and does not depend on a regularization parameter like λ in (3.43).

In our framework subset selection may be stated as the following problem. Find m columns of the $M \times M$ Gram matrix $\mathbf{K}$ such that

$$\min_{\boldsymbol{\alpha}_m} \|\mathbf{K}_m \boldsymbol{\alpha}_m - \mathbf{y}\| \leq \epsilon, \quad \text{for } m \ll M \tag{3.44}$$

where $\mathbf{K}_m$ denotes the reduced $M \times m$ Gram matrix consisting of the chosen columns, $\boldsymbol{\alpha}_m \in \mathbb{R}^m$ is the corresponding truncated weight vector and ϵ denotes the interpolation error. Such a sparse approximate interpolator can be interpreted as a discrete regularization in the sense that now regularization is controlled by discrete decisions whether to consider a particular column of the Gram matrix or not. In [Natarajan, 1999] a theoretical justification for regularization via sparse approximate interpolation is given. It was shown that the interpolation error ϵ and the noise intensity in the target vector $\mathbf{y}$

will tend to cancel out if ϵ is chosen a priori to be the noise intensity.

However, the question is how to choose the most relevant samples. In order to find the best subset of fixed size we would have to perform an exhaustive search in a discrete space consisting of $\binom{M}{m}$ possible choices which is clearly a NP-hard combinatorial search problem . Hence, one is restricted to suboptimal search strategies. The greedy algorithms in the literature which employ subset selection can be classified into forward selection [Mallat and Zhang, 1993] [Natarajan, 1995] [Grote and Huckle, 1997] [Smola and Bartlett, 2001] and backward selection [Couvreur and Bresler, 2000]. The advantage of backward selection is that provable convergence bounds exist. It was shown in [Couvreur and Bresler, 2000] that under certain circumstances backward selection is able to find an optimum in the sense that the resulting solution is equivalent to a direct approach using the L_0-norm regularization. This is a theoretically appealing result but the proof is not constructive. This means that there is no algorithmic way to evaluate whether the assumption holds on which the proof relies [Couvreur and Bresler, 2000]. Moreover, backward selection is computationally very expensive since in the first iteration all columns of the Gram matrix are considered as possible choices for the final model. The consequence is that the full Gram matrix has to be computed, stored and factorized prior to sequentially annihilating columns which are found to be least relevant with respect to the current residual error. This is rather prohibitive for large datasets. In contrast, the computational cost and the memory requirements associated with forward selection are much lower than those of backward selection and tends to be much cheaper than the direct convex programming approaches [Nair et al., 2002] at least for the case $m \ll M$. Forward selection starts with an empty training set and adds sequentially one sample that is most relevant according to a certain criterion

(e. g. decreasing the residual error). Thus, especially in case of large data sets forward selection is a practical method. The drawback of forward selection is that in contrast to backward selection no provable bounds exist concerning the convergence to the direct approach using the L_0-norm regularization. Nevertheless, as we shall see later, in most applications forward selection is very competitive to other state-of-the-art methods and we will restrict our further discussion to forward selection.

There are several slightly different contributions to this approach. In [Nair et al., 2002] an external algorithm which is based on elementary Givens rotations is used to update the QR-decomposition of the reduced Gram matrix in order to construct sparse models. The modified Gram Schmidt orthogonalization is used in [Billings and Lee, 2002] and [Chen et al., 1991] for the orthogonal decomposition of the Gram matrix. They also apply forward selection in a second step to obtain sparse models. This method is known as Orthogonal Least Squares (OLS). However, the OLS algorithm requires the computation and the storage of the full Gram matrix which is prohibitive for large datasets. We will now discuss some simple and efficient alternatives [Andelić et al., 2006b] [Andelić et al., 2007] to the methods discussed above.

3.4.1 Nonlinear Pseudodiscriminants

Motivated by the equivalence of the discriminant approach to a least-squares regression onto the labels we present in the following a computationally very efficient way for constructing LSMs in a RKHS within a forward selection rule with low memory requirements [Andelić et al., 2006b]. The proposed method exploits the positive definiteness of the Gram matrix for an order-recursive thin update of the pseudoinverse which represents the optimal solution in

the least-squares sense.

In a supervised learning problem one is faced with a training data set $\mathcal{D} = \{\mathbf{x}_i, y_i\}, i = 1 \dots M$. Here, $\mathbf{x}_i$ denotes an input vector of fixed size and y_i is the corresponding target value which is contained in $\mathbb{R}$ for regression or in $\{1, -1\}$ for binary classification. It is assumed that $\mathbf{x}_i \neq \mathbf{x}_j$, for $i \neq j$.

We focus on sparse approximations of models of the form

$$\hat{\mathbf{y}} = \mathbf{K}\boldsymbol{\alpha} + \mathbf{e}. \tag{3.45}$$

where $\mathbf{e}$ denotes the residual error. The use of Mercer kernels $k(\cdot, \mathbf{x})$ [Mercer, 1909] gives rise to a symmetric positive definite Gram Matrix $\mathbf{K}$ with elements $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ defining the subspace of the RKHS in which learning takes place. The weight vector $\boldsymbol{\alpha} = \{b, \alpha_1, \dots, \alpha_M\}$ contains a bias term b with a corresponding column $\mathbf{1} = \{1, \dots, 1\}$ in the Gram matrix.

Consider the overdetermined least-squares-problem

$$\hat{\boldsymbol{\alpha}}_m = \underset{\boldsymbol{\alpha}_m}{\operatorname{argmin}} \|\mathbf{K}_m\boldsymbol{\alpha}_m - \mathbf{y}\|^2 \tag{3.46}$$

in the m-th forward selection iteration with the reduced Gram matrix $\mathbf{K}_m = [\mathbf{1}\ \mathbf{k}_1 \dots \mathbf{k}_m] \in \mathbb{R}^{M\times(m+1)}$ where $\mathbf{k}_i = (k(\cdot, \mathbf{x}_1), \dots, k(\cdot, \mathbf{x}_M))^T,\ i \in \{1, \dots, m\}$ denotes one previously unselected column of the full Gram matrix. We denote the reduced weight vector as $\boldsymbol{\alpha}_m = \{b, \alpha_1, \dots, \alpha_m\} \in \mathbb{R}^{m+1}$ and the target vector as $\mathbf{y} = (y_1, \dots, y_M)^T$. Among all generalized inverses of $\mathbf{K}_m$ the pseudoinverse

$$\mathbf{K}_m^\dagger = (\mathbf{K}_m^T\mathbf{K}_m)^{-1}\mathbf{K}_m^T \tag{3.47}$$

is the one that has the lowest Frobenius norm [Ben-Israel and Greville, 1977]. Thus, the corresponding solution

$$\hat{\boldsymbol{\alpha}}_m = \mathbf{K}_m^\dagger\mathbf{y} \tag{3.48}$$

has the lowest Euclidean norm.

In order to see how the knowledge of $\mathbf{K}_{m-1}^{\dagger}$ can be used to obtain the current pseudoinverse $\mathbf{K}_m^{\dagger}$ we have to partition $\mathbf{K}_m$ and $\boldsymbol{\alpha}_m$ in the form

$$\mathbf{K}_m = [\mathbf{K}_{m-1}\mathbf{k}_m] \tag{3.49}$$

$$\boldsymbol{\alpha}_m = (\boldsymbol{\alpha}_{m-1}\alpha_m)^T \tag{3.50}$$

and to set $\alpha_m = \alpha_{m0} = const.$ Then the square loss becomes

$$L(\boldsymbol{\alpha}_{m-1}, \alpha_{m0}) = \|\mathbf{K}_{m-1}\boldsymbol{\alpha}_{m-1} - (\mathbf{y} - \mathbf{k}_m\alpha_{m0})\|^2. \tag{3.51}$$

The minimum of (3.51) in the least-squares-sense is given by

$$\hat{\boldsymbol{\alpha}}_{m-1} = \mathbf{K}_{m-1}^{\dagger}(\mathbf{y} - \mathbf{k}_m\alpha_{m0}). \tag{3.52}$$

Inserting (3.52) into (3.51) yields

$$L(\alpha_{m0}) = \|(\mathbf{I} - \mathbf{K}_{m-1}\mathbf{K}_{m-1}^{\dagger})\mathbf{k}_m\alpha_{m0} - (\mathbf{I} - \mathbf{K}_{m-1}\mathbf{K}_{m-1}^{\dagger})\mathbf{y}\|^2 \tag{3.53}$$

with $\mathbf{I}$ denoting the identity matrix of appropriate size.

Note that the vector

$$\mathbf{q}_m = (\mathbf{I} - \mathbf{K}_{m-1}\mathbf{K}_{m-1}^{\dagger})\mathbf{k}_m = \mathbf{k}_m - \mathbf{K}_{m-1}(\mathbf{K}_{m-1}^{\dagger}\mathbf{k}_m) \tag{3.54}$$

is the residual corresponding to the least-squares regression of $\mathbf{K}_{m-1}$ onto $\mathbf{k}_m$. Hence, $\mathbf{q}_m$ is a nullvector if and only if $\mathbf{k}_m$ is a nullvector unless $\mathbf{K}$ is not strictly positive definite. However, due to the use of positive definite Mercer kernels $\mathbf{k}_m$ can not be a nullvector since at least $k(\mathbf{x}, \mathbf{x}) > 0$. A problem can arise if the Gram matrix is not *strictly* positive definite or ill-conditioned which can be the case even if we use Mercer kernels. Thus, to ensure strictly positive definiteness of $\mathbf{K}$, it is mandatory to add a small positive constant ε to the main diagonal of the full Gram matrix in the form $\mathbf{K} \rightarrow \mathbf{K} + \varepsilon\mathbf{I}$. This form of regularization smoothes the solution similarly to the Ridge regression which we discussed above.

Forward selection may then be performed using this strictly positive definite Gram matrix. In the following $\mathbf{k}_m \neq \mathbf{0}$ is assumed.

The minimum of (3.53) is met at

$$\hat{\alpha}_{m0} = \mathbf{q}_m^\dagger(\mathbf{I} - \mathbf{K}_{m-1}\mathbf{K}_{m-1}^\dagger)\mathbf{y} \tag{3.55}$$

Noting that the pseudoinverse of a vector is given by

$$\mathbf{q}_m^\dagger = \frac{\mathbf{q}_m^T}{\|\mathbf{q}_m\|^2} \tag{3.56}$$

equation (3.55) may be written as

$$\begin{aligned} \hat{\alpha}_{m0} &= \frac{\mathbf{q}_m^T(\mathbf{I} - \mathbf{K}_{m-1}\mathbf{K}_{m-1}^\dagger)\mathbf{y}}{\|\mathbf{q}_m\|^2} \\ &= \frac{\mathbf{k}_m^T(\mathbf{I} - \mathbf{K}_{m-1}\mathbf{K}_{m-1}^\dagger)^T(\mathbf{I} - \mathbf{K}_{m-1}\mathbf{K}_{m-1}^\dagger)\mathbf{y}}{\|\mathbf{q}_m\|^2}. \end{aligned} \tag{3.57}$$

The matrix

$$\mathbf{P}_m = \mathbf{I} - \mathbf{K}_{m-1}\mathbf{K}_{m-1}^\dagger \tag{3.58}$$

is an orthogonal projection matrix which implies it being symmetric and idempotent, i. e.

$$\mathbf{P}_m\mathbf{P}_m^T = \mathbf{P}_m^T\mathbf{P}_m = \mathbf{P}_m. \tag{3.59}$$

Noting the last equality and the definition of $\mathbf{q}_m$ in (3.54) equation (3.57) simplifies to

$$\hat{\alpha}_{m0} = \mathbf{q}_m^\dagger\mathbf{y}. \tag{3.60}$$

Combining (3.60) with (3.52) the current weight vector $\hat{\boldsymbol{\alpha}}_m$ may be updated as

$$\hat{\boldsymbol{\alpha}}_m = \begin{bmatrix} \hat{\boldsymbol{\alpha}}_{m-1} \\ \hat{\alpha}_{m0} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{m-1}^\dagger - \mathbf{K}_{m-1}^\dagger\mathbf{k}_m\mathbf{q}_m^\dagger \\ \mathbf{q}_m^\dagger \end{bmatrix}\mathbf{y} \tag{3.61}$$

revealing the update

$$\mathbf{K}_m^\dagger = \begin{bmatrix} \mathbf{K}_{m-1}^\dagger - \mathbf{K}_{m-1}^\dagger\mathbf{k}_m\mathbf{q}_m^\dagger \\ \mathbf{q}_m^\dagger \end{bmatrix} \tag{3.62}$$

for the current pseudoinverse.

In the m-th iteration $O(Mm)$ operations are required for these updates. The memory requirement for the proposed method is of the same order. Note that the computation of the pseudoinverse from scratch would require $O(Mm^3)$ operations in each iteration. In the following we refer to the described method as Nonlinear Pseudodiscriminants (NPDs).

Forward Selection

The question which criteria are possible for choosing a new basis center with the corresponding column of the Gram matrix in each iteration is still open. Obviously, a reasonable goal of every forward selection scheme could be to select the columns of the Gram matrix that provide the greatest reduction of the residual. Methods like basis matching pursuit [Mallat and Zhang, 1993], order-recursive matching pursuit [Natarajan, 1995] or probabilistic approaches [Smola and Schölkopf, 2000] are several contributions to this issue. In [Nair et al., 2002], forward selection is performed in a computationally very efficient way by simply choosing the column of the Gram matrix that corresponds to the entry with the highest absolute value in the current residual. The reasoning is that the residual provides the direction of the maximum decrease in the cost function $0.5\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} - \boldsymbol{\alpha}^\mathbf{T}\mathbf{y}$, since the Gram matrix is strictly positive definite. The latter method is used in the following but note that the NPDs may be applied within any of the above forward selection rules.

Furthermore, one advantage of the NPDs is that the corresponding residual may be updated with a negligible computational cost.

Consider the residual

$$\hat{\mathbf{e}}_m = \mathbf{y} - \hat{\mathbf{y}}_m = \mathbf{y} - [\mathbf{K}_{m-1}\mathbf{k}_m]\,\hat{\boldsymbol{\alpha}}_m \tag{3.63}$$

in the m-th iteration. Inserting (3.61) into (3.79) yields

$$\begin{aligned}
\hat{\mathbf{e}}_m &= \mathbf{y} - [\mathbf{K}_{m-1}\mathbf{k}_m] \begin{bmatrix} \mathbf{K}_{m-1}^\dagger - \mathbf{K}_{m-1}^\dagger \mathbf{k}_m \mathbf{q}_m^\dagger \\ \mathbf{q}_m^\dagger \end{bmatrix} \mathbf{y} \\
&= \mathbf{y} - \hat{\mathbf{y}}_{m-1} + \mathbf{K}_{m-1}\mathbf{K}_{m-1}^\dagger \mathbf{k}_m \mathbf{q}_m^\dagger \mathbf{y} - \mathbf{k}_m \mathbf{q}_m^\dagger \mathbf{y} \\
&= \hat{\mathbf{e}}_{m-1} - (\mathbf{q}_m^\dagger \mathbf{y})(\mathbf{k}_m - \mathbf{K}_{m-1}\mathbf{K}_{m-1}^\dagger \mathbf{k}_m) \\
&= \hat{\mathbf{e}}_{m-1} - (\mathbf{q}_m^\dagger \mathbf{y})\mathbf{q}_m. \qquad (3.64)
\end{aligned}$$

The current residual may be updated without even knowing the weight vector $\hat{\boldsymbol{\alpha}}_m$. The residual update requires $\mathcal{O}(M)$ operations in each iteration. This is a considerable saving compared to a re-computation of the current residual in each iteration in the form

$$\hat{\mathbf{e}}_m = \mathbf{y} - \mathbf{K}_m \mathbf{K}_m^\dagger \mathbf{y} \qquad (3.65)$$

which requires $\mathcal{O}(Mm)$ operations. We are able to compute the least-squares solution $\hat{\boldsymbol{\alpha}}_m$ in one shot

$$\hat{\boldsymbol{\alpha}}_m = \mathbf{K}_m^\dagger \mathbf{y} \qquad (3.66)$$

after the forward selection is stopped. It is possible to determine the number of basis functions using crossvalidation or one may use for instance the Bayesian Information Criterion or the Minimum Description Length as alternative stopping criteria.

The NPD is summarized in pseudocode in Algorithm 1. In order to illustrate that the NPD is a reasonable approximation of the standard KFD which uses all training instances as basis centers we use the well known synthetic Ripley data set. This is a linearly non-separable two-class classification problem. The Ripley dataset consists of 250 training and 1000 testing examples. The Bayes error rate for this dataset is 8%. We use the Gaussian kernel for both methods with the same kernel width and the same regularization

Algorithm 1 Nonlinear Pseudo-Discriminants

Require: Training data $\mathbf{X}$, labels $\mathbf{y}$, kernel, ε

Initializations: $\lambda \leftarrow 0$, $m \leftarrow 1$, $\mathbf{K}_1 = \mathbf{1}$, $\mathbf{K}_1^\dagger = \frac{1}{M}\mathbf{1}^T$

while λ changes significantly and $\mathbf{K}_m$ is not ill-conditioned **do**

Update $\hat{\mathbf{e}}_m$

find the index i_{opt} of the entry of $\hat{\mathbf{e}}_m$ with the highest absolute value

$I_{opt} \leftarrow \{I_{opt}, i_{opt}\}$

$I \leftarrow I \setminus \{i_{opt}\}$

Compute $\mathbf{k}_{i_{opt}}$

$\mathbf{K}_m \leftarrow [\mathbf{K}_{m-1}\mathbf{k}_{i_{opt}}]$

Update $\mathbf{K}_m^\dagger$ using $\mathbf{k}_{i_{opt}}$

$m \leftarrow m + 1$

end while

return $\hat{\boldsymbol{\alpha}}_m, I_{opt}$

constant. In figure 3.4.1 we see that in the region where the data are not separable the decision boundaries of the two methods are almost identical. The test error of the NPD on this dataset is 9.2% which is quite favorable compared to the test error of the full KFD (9.6%). This is a promising first result for the following empirical studies.

We have seen that some sort of regularization is mandatory for numerical stability and a good generalization ability especially in case of kernel-methods where possibly infinite dimensional spaces are involved. Sparse approximations like the NPD are one way to deal with this problem since a sparse interpolator can be seen as a discrete regularization. However, the sparsity principle alone is not entirely able to prevent overfitting. For instance, in some cases where the data are highly affected by noise even a sparse model may fit into noise and cause overfitting. Now the question arises if there is a

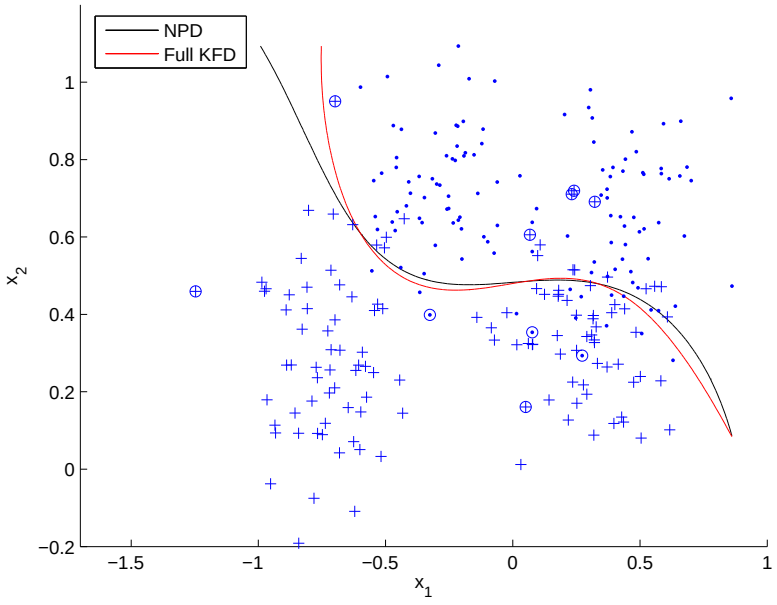

Figure 3.2: Comparison of the decision boundaries obtained by the full KFD and our proposed approximation (NPD) on Ripley's dataset. The training samples are also shown and the ten selected basis centers used by the NPD are encircled.

way to combine sparse approximations with a direct regularization method, i. e. , can we find a sparse solution for minimizing the regularized loss

$$L_R(\boldsymbol{\alpha}) = \|\mathbf{K}\boldsymbol{\alpha} - \mathbf{y}\|^2 + \lambda\|\boldsymbol{\alpha}\|^2. \tag{3.67}$$

within a forward selection scheme?

When we use the reduced Gram matrix $\mathbf{K}_m$ then the modified problem is

$$\begin{aligned} & \underset{\boldsymbol{\alpha}_m}{\operatorname{argmin}} L_R(\boldsymbol{\alpha}_m) \\ = \; & \underset{\boldsymbol{\alpha}_m}{\operatorname{argmin}} (\mathbf{y} - \mathbf{K}_m\boldsymbol{\alpha}_m)^T(\mathbf{y} - \mathbf{K}_m\boldsymbol{\alpha}_m) + \lambda\boldsymbol{\alpha}_m^T\boldsymbol{\alpha}_m \\ = \; & \underset{\boldsymbol{\alpha}_m}{\operatorname{argmin}} (\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{K}_m\boldsymbol{\alpha}_m + \boldsymbol{\alpha}_m^T\mathbf{K}_m^T\mathbf{K}_m\boldsymbol{\alpha}_m) + \lambda\boldsymbol{\alpha}_m^T\boldsymbol{\alpha}_m. \end{aligned} \tag{3.68}$$

Setting the derivative of L_R with respect to $\boldsymbol{\alpha}_m$ equal to zero yields

$$\begin{aligned} \frac{\partial L_R}{\partial \boldsymbol{\alpha}_m} &= (\mathbf{K}_m^T \mathbf{K}_m \boldsymbol{\alpha}_m - \mathbf{K}_m^T \mathbf{y}) + \lambda \mathbf{I}_m \boldsymbol{\alpha}_m = 0 \\ &\Longrightarrow (\mathbf{K}_m^T \mathbf{K}_m + \lambda \mathbf{I}_m)\boldsymbol{\alpha}_m = \mathbf{K}_m^T \mathbf{y}. \end{aligned} \tag{3.69}$$

Thus, minimizing the regularized loss L_R yields

$$\tilde{\boldsymbol{\alpha}}_m = (\mathbf{K}_m^T \mathbf{K}_m + \lambda \mathbf{I}_m)^{-1} \mathbf{K}_m \mathbf{y} = \tilde{\mathbf{K}}_m^{\dagger} \mathbf{y}. \tag{3.70}$$

It can be seen that for $\lambda \to 0$ we obtain the pseudoinverse which is the solution of the ordinary least-squares problem. Thus, for $\lambda > 0$ the matrix $\tilde{\mathbf{K}}_m^{\dagger}$ can be seen as a regularized version of the pseudoinverse. The problem however is that with the methods outlined before the regularized pseudoinverse can not be updated order-recursively in contrast to the NPD since $\mathbf{I} - \mathbf{K}_{m-1}\tilde{\mathbf{K}}_{m-1}^{\dagger}$ is not an orthogonal projection for $\lambda \neq 0$. In the following we will outline a possible way to overcome this problem.

3.4.2 Orthogonal Least Squares

In order to apply a direct regularization method within forward selection we have to find a way to reveal the individual impact of each regressor on the final model. A classical way [Chen et al., 1991] of doing so is to consider an orthogonal decomposition of the Gram matrix in the form

$$\mathbf{K} = \mathbf{Q}\mathbf{U} \tag{3.71}$$

where

$$\mathbf{U} = \begin{bmatrix} 1 & u_{1,2} & \cdots & u_{1,M} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{M,M} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{3.72}$$

and

$$\mathbf{Q} = [\mathbf{q}_1 \dots \mathbf{q}_M] \tag{3.73}$$

with orthogonal columns that satisfy $\mathbf{q}_i^T \mathbf{q}_j = 0$, for $i \neq j$. If we replace the Gram matrix of the regression model (3.45) by this orthogonal decomposition an equivalent model can be written as

$$\mathbf{y} = \mathbf{Q}\tilde{\boldsymbol{\alpha}} + \mathbf{e} \tag{3.74}$$

where the orthogonal weights satisfy the triangular system

$$\mathbf{U}\boldsymbol{\alpha} = \tilde{\boldsymbol{\alpha}}. \tag{3.75}$$

Thus, knowing $\mathbf{U}$ and $\tilde{\boldsymbol{\alpha}}$, the original weights can easily be recovered. Due to the fact that $\mathbf{U}$ is a upper triangular matrix the linear system of equations above can easily be solved through backsubstitution.

The classical Orthogonal Least Squares (OLS) algorithm [Chen et al., 1991] uses the modified Gram-Schmidt procedure to perform an orthogonalization of the full Gram matrix K. Starting from $n = 1$, the columns $\mathbf{k}_i,\ n+1 < i < M$ of $\mathbf{K}$ are made orthogonal to the nth column at the nth stage. This operation is repeated for $1 < n < M - 1$. Thus, all columns of $\mathbf{K}$ have to be computed and stored. The memory requirement of this method is $\mathcal{O}(M^2)$. This is rather prohibitive especially for large datasets. In the following we will present a thin update scheme [Andelić et al., 2007] for the reduced Gram matrix $\mathbf{K}_m$ which reduces the memory requirement to $\mathcal{O}(Mm)$.

Order-Recursive Orthogonal Least Squares

Recall that the matrix $\mathbf{P}_m$ which we defined in (3.58) is a projection matrix. Thus, every projection $\mathbf{q}_m = \mathbf{P}_m \mathbf{k}_m$ lies in a subspace which is orthogonal

to $\mathbf{K}_{m-1}$ and it follows immediately that $\mathbf{q}_i^T \mathbf{q}_j = 0$, for $i \neq j$. Hence, an orthogonal decomposition

$$\mathbf{K}_m = \mathbf{Q}_m \mathbf{U}_m \tag{3.76}$$

of the reduced Gram matrix is given by the orthogonal matrix

$$\mathbf{Q}_m = \left[\mathbf{Q}_{m-1} \mathbf{q}_m\right] \tag{3.77}$$

and the upper triangular matrix

$$\mathbf{U}_m = \left[\begin{pmatrix} \mathbf{U}_{m-1} \\ \mathbf{0}_{m-1}^T \end{pmatrix} (\mathbf{Q}_m^T \mathbf{Q}_m)^{-1} \mathbf{Q}_m^T \mathbf{k}_m\right]. \tag{3.78}$$

In the m-th iteration $\mathcal{O}(Mm)$ operations are required for all these updates. Note that the inversion of the matrix $\mathbf{Q}_m^T \mathbf{Q}_m$ is trivial since this matrix is diagonal. However, the condition number of the matrix $\mathbf{Q}_m$ increases as the number of selected columns m grows. Thus, to ensure numerical stability it is important to monitor the condition number of this matrix and to terminate the iteration if the condition number exceeds a predefined value unless another stopping criterion is reached earlier. We refer to the proposed method as Order-Recursive Orthogonal Least Squares (OROLS).

Regularization and Selection of Basis Centers

Consider the residual

$$\tilde{\mathbf{e}}_m = \mathbf{y} - \hat{\mathbf{y}}_m = \mathbf{y} - \mathbf{Q}_m \tilde{\boldsymbol{\alpha}}_m \tag{3.79}$$

in the m-th iteration. The vector $\tilde{\boldsymbol{\alpha}}_m$ contains the orthogonal weights.

The regularized square residual is given by

$$\begin{aligned} \tilde{E}_m &= \tilde{\mathbf{e}}_m^T \tilde{\mathbf{e}}_m + \lambda \tilde{\boldsymbol{\alpha}}_m^T \tilde{\boldsymbol{\alpha}}_m \\ &= \mathbf{y}^T \tilde{\mathbf{P}}_m \mathbf{y} \end{aligned} \tag{3.80}$$

where λ denotes a regularization parameter. The minimum of (3.80) is given by

$$\begin{aligned} \tilde{\mathbf{P}}_m &= \mathbf{I} - \mathbf{Q}_m(\mathbf{Q}_m^T\mathbf{Q}_m + \lambda\mathbf{I}_m)^{-1}\mathbf{Q}_m^T \\ &= \tilde{\mathbf{P}}_{m-1} - \frac{\mathbf{q}_m\mathbf{q}_m^T}{\lambda + \mathbf{q}_m^T\mathbf{q}_m}. \end{aligned} \tag{3.81}$$

Thus, the current residual corresponding to the regularized least squares problem may be updated as

$$\begin{aligned} \tilde{\mathbf{e}}_m &= (\tilde{\mathbf{P}}_{m-1} - \frac{\mathbf{q}_m\mathbf{q}_m^T}{\lambda + \mathbf{q}_m^T\mathbf{q}_m})\mathbf{y} \\ &= \tilde{\mathbf{e}}_{m-1} - \mathbf{q}_m\frac{\mathbf{y}^T\mathbf{q}_m}{\lambda + \mathbf{q}_m^T\mathbf{q}_m}. \end{aligned} \tag{3.82}$$

The orthogonal weights

$$(\tilde{\boldsymbol{\alpha}}_m)_i = \frac{\mathbf{y}^T\mathbf{q}_i}{\lambda + \mathbf{q}_i^T\mathbf{q}_i}, \quad 1 \leq i \leq m. \tag{3.83}$$

can be computed when the forward selection is stopped. The original weights can then be recovered by

$$\hat{\boldsymbol{\alpha}}_m = \mathbf{U}_m^{-1}\tilde{\boldsymbol{\alpha}}_m \tag{3.84}$$

which is an easy inversion since $\mathbf{U}_m$ is upper triangular.

Now we can choose in each iteration a previously unselected column $\mathbf{q}_i$ which corresponds to the highest absolute value in the current residual and add it to $\mathbf{Q}_{m-1}$. It remains to decide when to stop the iterations. In case of the NPD we outlined that one way to determine the number of basis functions is to use crossvalidation or to use for instance the Bayesian Information Criterion or the Minimum Description Length as alternative stopping criteria. It turns out that for OROLS there is another possibility to derive a stopping criterion which is given by the following reasoning.

It was shown in [Wahba, 1979] that the problem of minimizing the *expected* mean of squared residuals which are given in (3.80) w. r. t. the regularization parameter λ is equivalent to minimizing the so-called Generalized

Cross Validation (GCV) w. r. t. λ. Therefore the GCV is a reasonable criterion for choosing a λ that ensures a good generalization ability.

The idea presented in [Gu and Wahba, 1991] and [Orr, 1995] is to re-estimate the λ in each iteration using the GCV. The convergence of λ can then be used as a stopping criterion. We will now summarize the results. For details see [Orr, 1995].

The GCV_m in the m-th iteration is given by

$$GCV_m = \frac{1}{M} \frac{\|\tilde{\mathbf{P}}_m \mathbf{y}\|^2}{\Big((1/M)\ \mathrm{trace}(\tilde{\mathbf{P}}_m)\Big)^2}. \tag{3.85}$$

Minimizing the GCV_m with respect to λ gives rise to a re-estimation formula for λ. An alternative way to obtain a re-estimation of λ is to maximize the Bayesian evidence [MacKay, 1992].

Differentiating (3.85) with respect to λ and setting the result to zero gives a minimum when

$$\mathbf{y}^T \tilde{\mathbf{P}}_m \frac{\partial \tilde{\mathbf{P}}_m \mathbf{y}}{\partial \lambda} \mathrm{trace}(\tilde{\mathbf{P}}_m) = \mathbf{y}^T \tilde{\mathbf{P}}_m^2 \mathbf{y} \frac{\partial \mathrm{trace}(\tilde{\mathbf{P}}_m)}{\partial \lambda}. \tag{3.86}$$

Noting that

$$\mathbf{y}^T \tilde{\mathbf{P}}_m \frac{\partial \tilde{\mathbf{P}}_m \mathbf{y}}{\partial \lambda} = \lambda \tilde{\boldsymbol{\alpha}}_m^T (\mathbf{Q}_m^T \mathbf{Q}_m + \lambda \mathbf{I}_m)^{-1} \tilde{\boldsymbol{\alpha}}_m \tag{3.87}$$

equation (3.86) can be rearranged to obtain the re-estimation formula

$$\lambda := \frac{[\partial \mathrm{trace}(\tilde{\mathbf{P}}_m)/\partial \lambda] \mathbf{y}^T \tilde{\mathbf{P}}_m^2 \mathbf{y}}{\mathrm{trace}(\tilde{\mathbf{P}}_m) \tilde{\boldsymbol{\alpha}}_m^T (\mathbf{Q}_m^T \mathbf{Q}_m + \lambda \mathbf{I}_m)^{-1} \tilde{\boldsymbol{\alpha}}_m} \tag{3.88}$$

where

$$\frac{\partial \mathrm{trace}(\tilde{\mathbf{P}}_m)}{\partial \lambda} = \sum_{i=1}^{m} \frac{\mathbf{q}_i^T \mathbf{q}_i}{(\lambda + \mathbf{q}_i^T \mathbf{q}_i)^2}. \tag{3.89}$$

The forward selection is stopped when λ stops changing significantly.

The computational cost for this update is $\mathcal{O}(m)$. The OROLS algorithm is summarized in pseudocode in Algorithm 2.

Algorithm 2 Order-Recursive Orthogonal Least Squares

Require: Training data $\mathbf{X}$, labels $\mathbf{y}$, kernel

Initializations: $\lambda \leftarrow 0$, $m \leftarrow 1$, $\mathbf{K}_1 = \mathbf{1}$, $\mathbf{K}_1^\dagger = \frac{1}{M}\mathbf{1}^T$, $\mathbf{Q}_1 = \mathbf{1}$, $\mathbf{U}_1 = [1]$, $I = \{1, \ldots, M\}$, $I_{opt} = \{\}$

while λ changes significantly and $\mathbf{Q}_m$ is not ill-conditioned **do**

find the index i_{opt} of the entry of the residual $\tilde{\mathbf{e}}_m$ with the highest absolute value

$I_{opt} \leftarrow \{I_{opt}, i_{opt}\}$

$I \leftarrow I \setminus \{i_{opt}\}$

Compute $\mathbf{k}_{i_{opt}}$

Compute $\mathbf{q}_{i_{opt}}$

$\mathbf{K}_m \leftarrow [\mathbf{K}_{m-1}\mathbf{k}_{opt}]$

$\mathbf{Q}_m \leftarrow [\mathbf{Q}_{m-1}\mathbf{q}_{opt}]$

Update $\mathbf{K}_m^\dagger$ and $\mathbf{U}_m$ using $\mathbf{k}_{opt}$ and $\mathbf{q}_{opt}$

Update λ

$m \leftarrow m + 1$

end while

return $\hat{\boldsymbol{\alpha}}_m, I_{opt}$

3.4.3 Recursive Least Squares

In some applications where the datasets are very large even the reduced memory requirements of NPDs and OROLS are too high. We will now outline one possibility to reduce the memory requirement from $\mathcal{O}(Mm)$ to $\mathcal{O}(m^2)$. The proposed method uses the idea presented in [Engel et al., 2003] which is to approximate the full $M \times M$ Gram matrix by an $m \times m$ matrix. The Recursive Least-Squares algorithm (RLS-algorithm) computes the least squares solution recursively by sequentially processing the training data $\mathbf{X}$. In each step $n; n = 1, \ldots, M$ the matrix inversion lemma is applied to obtain the $n \times n$ matrix $(\mathbf{K}_n^T \mathbf{K}_n)^{-1}$. The RLS-algorithm is widely used in the field of online adaptive filtering since it features fast convergence and a favorably low misadjustment to the optimal least square solution even in presence of noise [Haykin, 2002]. To exploit these advantageous properties of the RLS for machine learning purposes one has to prevent overfitting, i. e. one has to introduce sparsity to the recursive least square solution.

$$\min_{\mathbf{w},b} E(\mathbf{w}) = \sum_{i=1}^{M} \|\Phi^T(\mathbf{x}_i)\mathbf{w} + b - y_i\|^2 \tag{3.90}$$

with b denoting a bias term. By defining $\Phi \rightarrow (\Phi^T, 1)^T$ and $\mathbf{w} \rightarrow (\mathbf{w}^T, b)^T$ and by exploiting the fact that $\mathbf{w}$ can be expressed as an expansion of the $\Phi(\mathbf{x}_i)$ in the form

$$\mathbf{w} = \sum_{i}^{M} \alpha_i \Phi(\mathbf{x}_i) \tag{3.91}$$

equation (3.90) can be written as

$$\min_{\boldsymbol{\alpha}} E(\boldsymbol{\alpha}) = \|\mathbf{K}\boldsymbol{\alpha} - \mathbf{y}\|^2 \tag{3.92}$$

with

$$[\mathbf{K}]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) \tag{3.93}$$

denoting the symmetric $M \times M$ kernel matrix and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)^T$.

Unsupervised Sparsification

The idea proposed in [Engel et al., 2003] is to regard a training sample $\mathbf{x}_n$ for the expansion of the direction $\mathbf{w}_n$ only if $\mathbf{x}_n$ is approximately linearly independent of the m_{n-1} previously chosen samples $\tilde{\mathbf{x}}_j; j = 1, \ldots, m_{n-1}$. Thus, introducing the set of reduced coefficients $\mathbf{r}_n = (r_1, \ldots, r_{m_{n-1}})^T$ every $\mathbf{x}_n$ that fulfills the condition

$$\delta_n := \min_{\mathbf{r}_n} \left\| \sum_{j=1}^{m_{n-1}} r_j \Phi(\tilde{\mathbf{x}}_j) - \Phi(\mathbf{x}_n) \right\|^2 > \nu \tag{3.94}$$

is added to a dictionary $\mathbf{D}_n$ with ν being a small regularization parameter by which the level of sparsity is controlled. This sparsification method is unsupervised since one controls only ν blindly to the labels. The greater ν the more sparsity is achieved.

Equation (3.94) can be solved straightforwardly yielding

$$\delta_n = k_{nn} - \tilde{\mathbf{k}}_{n-1}^T(\mathbf{x}_n)\tilde{\mathbf{r}}_n > \nu \tag{3.95}$$

$$\text{with} \quad \tilde{\mathbf{r}}_n = \tilde{\mathbf{K}}_{n-1}^{-1}\tilde{\mathbf{k}}_{n-1}^T(\mathbf{x}_n) \tag{3.96}$$

where

$$\left[\tilde{\mathbf{K}}_{n-1}\right]_{i,j} = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) \tag{3.97}$$

$$(\tilde{\mathbf{k}}_{n-1}(\mathbf{x}_n))_i = k(\tilde{\mathbf{x}}_i, \mathbf{x}_n) \tag{3.98}$$

$$k_{nn} = k(\mathbf{x}_n, \mathbf{x}_n) \tag{3.99}$$

for $i, j = 1, \ldots, m_{n-1}$.

Note that the redefinition of Φ in (3.92) effects that $k(\mathbf{x}, \mathbf{x}')$ has to be redefined also as $k(\mathbf{x}, \mathbf{x}') + 1$ and ν may be chosen in the range

$$0 < \nu \leq 1. \tag{3.100}$$

By applying the previously described sparsification in each step n one may collect all reduced coefficients $\tilde{\mathbf{r}}_n$ defining the $n \times m_n$ reduction matrix $\mathbf{R}_n$. Then the corresponding direction $\mathbf{w}_n$ is approximated in the form

$$\mathbf{w}_n \approx \tilde{\mathbf{\Phi}}_n \tilde{\boldsymbol{\alpha}}_n \tag{3.101}$$

with

$$\tilde{\boldsymbol{\alpha}}_n := \mathbf{R}_n^T \boldsymbol{\alpha}_n \tag{3.102}$$

and

$$\tilde{\mathbf{\Phi}}_n = [\Phi(\tilde{\mathbf{x}}_1), \ldots, \Phi(\tilde{\mathbf{x}}_{m_n})]. \tag{3.103}$$

Inserting (3.101) in (3.90) and having in mind the redefinition of $\mathbf{w}$ and Φ in (3.92) yield

$$\min_{\tilde{\boldsymbol{\alpha}}_n} E(\tilde{\boldsymbol{\alpha}}_n) = \|\mathbf{R}_n \tilde{\mathbf{K}}_n \tilde{\boldsymbol{\alpha}}_n - \mathbf{y}_n\|^2 \tag{3.104}$$

with $\mathbf{y}_n = (y_1, \ldots, y_n)^T$. The least square solution of (3.104) is

$$\tilde{\boldsymbol{\alpha}}_n = \tilde{\mathbf{K}}_n^{-1} (\mathbf{R}_n^T \mathbf{R}_n)^{-1} \mathbf{R}_n^T \mathbf{y}_n. \tag{3.105}$$

Defining $\mathbf{P}_n = (\mathbf{R}_n^T \mathbf{R}_n)^{-1}$ the following recursions may be computed applying the matrix inversion lemma. For details see [Engel et al., 2003, Haykin, 2002].

In case of $\delta_n \leq \nu$ the dictionary $\mathbf{D}_n$ and $\tilde{\mathbf{K}}_n$ are unchanged. The update for $\mathbf{P}_n$ is

$$\mathbf{P}_n = \mathbf{P}_{n-1} - \frac{\mathbf{P}_{n-1} \tilde{\mathbf{r}}_n \tilde{\mathbf{r}}_n^T \mathbf{P}_{n-1}}{1 + \tilde{\mathbf{r}}_n^T \mathbf{P}_{n-1} \tilde{\mathbf{r}}_n}. \tag{3.106}$$

For $\delta_n > \nu$ the sample $\mathbf{x}_n$ is added to the dictionary and the matrices $\tilde{\mathbf{K}}_n^{-1}$ and $\mathbf{P}_n$ are updated:

$$\mathbf{P}_n = \begin{bmatrix} \mathbf{P}_{n-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{3.107}$$

$$\tilde{\mathbf{K}}_n^{-1} = \frac{1}{\delta_n} \begin{bmatrix} \delta_n \tilde{\mathbf{K}}_{n-1}^{-1} + \tilde{\mathbf{r}}_n \tilde{\mathbf{r}}_n^T & -\tilde{\mathbf{r}}_n \\ -\tilde{\mathbf{r}}_n^T & 1 \end{bmatrix} \tag{3.108}$$

The described unsupervised sparsification is summarized in Algorithm 3.

Algorithm 3 Unsupervised Sparsification

Require: ν, k

Initializations $\tilde{\mathbf{K}}_1^{-1} = [\frac{1}{k_{11}}]$, $\mathbf{P}_1 = [1]$, $\mathbf{D}_1 = \{\mathbf{x}_1\}$, $m = 1$

for $n = 2, \ldots, M$ **do**

get new sample $\mathbf{x}_n$

compute $\tilde{\mathbf{r}}_n$

compute δ_n

if $\delta_n > \nu$ **then**

$\mathbf{D}_n = \mathbf{D}_{n-1} \cup \{\mathbf{x}_n\}$

update $\tilde{\mathbf{K}}_n^{-1}$

update $\mathbf{P}_n$

$m \leftarrow m + 1$

else

$\mathbf{D}_n = \mathbf{D}_{n-1}$

update $\mathbf{P}_n$

end if

end for

return $\mathbf{P}_n, \mathbf{D}_n$

RLS-Filtering

After performing these recursions for the whole training set $\mathbf{X}$ one obtains the unsupervisedly reduced $m \times m$ matrices $\mathbf{P}_m$ and $\tilde{\mathbf{K}}_m^{-1}$. With the initializations $\mathbf{P}_{m,0} = \mathbf{P}_m$ and $\tilde{\boldsymbol{\alpha}}_{m,0} = (0, \ldots, 0)^T$ the standard RLS-algorithm is applied to compute (3.105) recursively. The matrix $\mathbf{P}_{m,n}$ is updated with (3.106). The RLS-update for $\tilde{\boldsymbol{\alpha}}_{m,n}$ is

$$\tilde{\boldsymbol{\alpha}}_{m,n} = \tilde{\boldsymbol{\alpha}}_{m,n-1} + \tilde{\mathbf{K}}_m^{-1}\mathbf{q}_{m,n}(y_n - \tilde{\mathbf{k}}_{n-1}^T(\mathbf{x}_n)\tilde{\boldsymbol{\alpha}}_{m,n-1}) \tag{3.109}$$

with

$$\mathbf{q}_{m,n} = \frac{\mathbf{P}_{m,n-1}\tilde{\mathbf{r}}_n}{1 + \tilde{\mathbf{r}}_n^T\mathbf{P}_{m,n-1}\tilde{\mathbf{r}}_n}. \tag{3.110}$$

The RLS-filtering of the data yields the reduced set of coefficients $\tilde{\boldsymbol{\alpha}}_m$. For the sake of a simple notation the index m will be omitted in the following. The pseudocode of the RLS-filtering procedure is given in Algorithm 4. In the

Algorithm 4 RLS-Filtering

```
Initialization: α̃_0 = (0, ..., 0)^T
for n = 1, ..., M do
    get new sample x_n
    compute r̃_n
    compute δ_n
    update q_n
    update P_n
    update α̃_n
end for
return D_n, α̃
```

following we will refer to the unsupervised sparsification with the subsequent RLS-filtering as Kernel Recursive Least Squares (KRLS).

3.5 Summary

Starting with an introduction of the discriminant approach we have seen that discriminants have very favorable properties as for their Bayes-optimality in certain classification settings. However, discriminants are not sparse in general. This is rather prohibitive for large datasets and can lead to poor generalization properties when no form of regularization is employed. We noticed that we have to overcome this problem especially when solving kernel-based discriminants. We have outlined how kernel-functions can be used to turn linear discriminants into nonlinear ones in order to enhance their empirical performance. Finally, we exploited the equivalence of discriminants to a least-squares regression onto the targets for the derivation of some forward selection algorithms which lead to sparse solutions and exhibit low costs in terms of memory and computational time.

Chapter 4

Application of Nonlinear Discriminants for Automatic Speech Recognition

We will now describe how the outputs of nonlinear discriminants can be probabilistically interpreted and where these probabilities can be used in an Automatic Speech Recognition (ASR) system. Speech recognition is a complicated task and state-of-the-art recognition systems are very complex. There are many different approaches for the implementation of the components. For further information the reader is referred to [Rabiner and Juang, 1993] [Gold and Morgan, 1999] [Huang et al., 2001]. Here we only want to provide an overview over ASR, some of its main difficulties, the basic components, their functionality and interaction.

4.1 Components of ASR

The general task of Automatic Speech Recognition (ASR) is to deduce an unknown sequence of words (text) from its observed acoustical realization, an utterance. We must thus "reverse" the process of speech production. Figure 4.1 shows the main components of an ASR system. In ASR systems

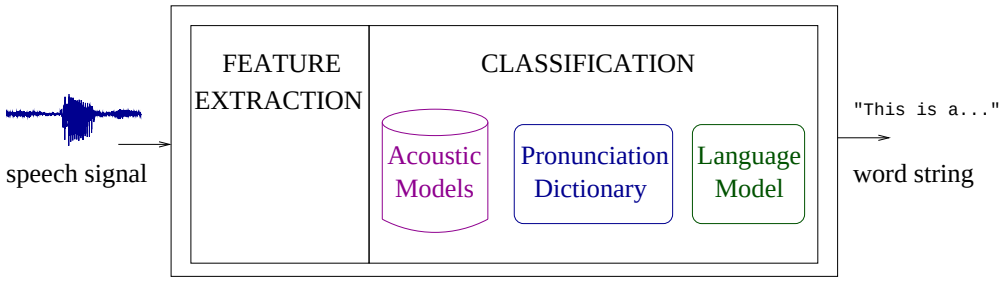

Figure 4.1: Principle components of an ASR system

acoustic information is sampled as a signal suitable for processing by computers and fed into a recognition process. The output of the system is a hypothesis for a transcription of the utterance. The first step is to convert the acoustic waveform into an electric signal for further processing. From this signal, certain properties (typically 15-30) are extracted over successive intervals because speech is short-time stationary, usually at around $10ms$. These properties build up the feature-vectors (frames) which should both be good in separating different classes of speech sounds as well as in suppressing irrelevant sources of variation. That means features are extracted that are *robust to acoustic variation* but *sensitive to linguistic content*. Put in other words, features that are discriminant and allow to distinguish between different linguistic units (e. g. , phones) are required. On the other hand the features should also be robust against noise and factors that are irrelevant for the recognition process (e. g. , the fundamental frequency of the

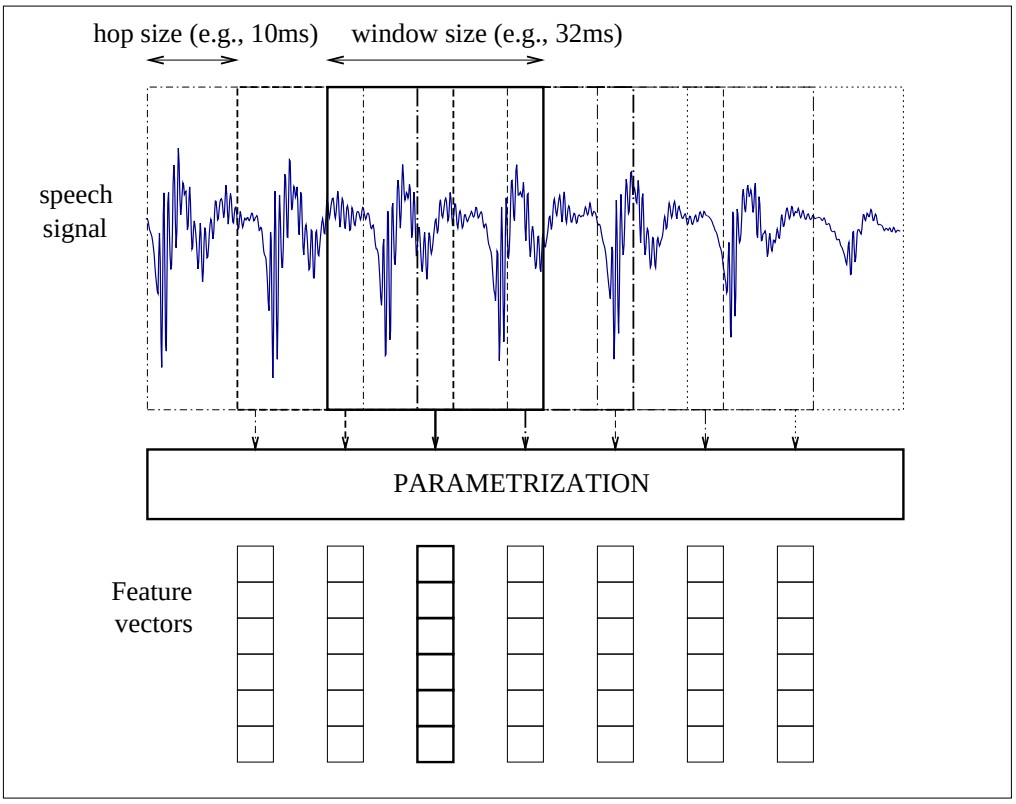

Figure 4.2: Feature extraction from a speech signal. Every 'hop-size' (or shift-size) seconds a vector of features is computed from the speech samples in a window of length 'window-size'.

speech signal). The number of features extracted from the waveform signal is commonly much lower than the number of signal samples, thus reducing the amount of data. The choice of suitable features varies depending on the classification technique.

Figure 4.2 indicates how features (or feature vectors) are derived from the speech signal. Typically, a frequency-domain based parametrization is performed to extract the features. Spectral analysis is performed, e. g. , every 10 ms on the speech samples in a window of, e. g. , 32 ms length. The

speech signal is regarded stationary in this time-scale. Although this is not strictly true, it is a reasonable approximation. For each frame a vector of parameters, the feature vector, is determined and handed to the next stage, the classification.

In the *classification* module the feature vectors are matched with reference patterns, which are called acoustic models. The reference patterns are usually Hidden Markov Models (HMMs). The HMM-states are assigned to smallest linguistic units of our choice (e. g. words, phones, subphone units). We train emission probabilities for the states, and transition probabilities between the states. One HMM is assigned for a whole utterance which is the full acoustic observation or, if hierarchically organized, smaller units thereof (words, phones, subphones). HMMs cope with temporal variation, which is important since the duration of individual phones may differ between the reference speech signal and the speech signal to be recognized. A linear normalization of the time axis is not sufficient here, since not all phones are expanded or compressed over time in the same way. For instance, stop consonants ("d", "t", "g", "k", "b", and "p") do not change their length much, whereas the length of vowels strongly depends on the overall speaking rate.

The *pronunciation dictionary* defines which combination of phones give valid words for the recognition. It can contain information about different pronunciation variants of the same word. Table 4.1 shows an extract of a dictionary. The words (graphemes) in the left column are related to their pronunciation (phones) in the right column (phone symbols like in the table are commonly used for the English language). The *language model* contains rudimentary syntactic information. Its aim is to predict the likelihood of specific words occurring one after another in a certain language. In a more formal description, the probability of the k-th word following the $(k-1)$

Table 4.1: Extract from a dictionary

word	*pronunciation*
INCREASE	ih n k r iy s
INCREASED	ih n k r iy s t
INCREASES	ih n k r iy s ah z
INCREASING	ih n k r iy s ih ng
INCREASINGLY	ih n k r iy s ih ng l iy
INCREDIBLE	ih n k r eh d ah b ah l

previous words is defined as $P(w_k|w_{k-1}, w_{k-2}, ..., w_1)$. In practice the context (number of previous words considered in the model) is restricted to $(n-1)$ words $P(w_k|w_{k-1}, w_{k-2}, ..., w_1) \approx P(w_k|w_{k-1}, w_{k-2}, ..., w_{k-n+1})$, and the resulting language model is called n-gram model.

4.1.1 Sub-word modeling with HMMs

In large vocabulary ASR systems, HMMs are used to represent sub units of words (such as phones). For English it is typical to have around 40 models (phones). The exact phone set depends on the dictionary that is used. Word models can be constructed as a combination of the sub word models.

Hidden Markov Models (HMMs) are used in ASR. A HMM is a stochastic finite state automaton (SFSA) built from a finite set of possible states $Q = \{q_1, \ldots, q_K\}$. Each of these states is associated with a specific probability distribution. A specific HMM M_i is, then, represented by a SFSA comprised of L_i states $S_i = \{s_1, \ldots, s_l, \ldots, s_{L_i}\}$ with each $s_l \in Q$, arranged according to a certain, most often predefined, topology.

Thus, HMMs can be used to model a sequence of feature-vectors $X = \{x_1, \ldots, x_N\}$ as a piecewise stationary process where each stationary seg-

ment is associated with a specific hidden (not directly observable) linguistic HMM state. This approach defines two concurrent stochastic processes: the sequence of HMM-states (modeling the temporal structure of speech), and a set of state output processes modeling the locally stationary property of the speech signal.

Since it is clearly infeasible to have a HMM for every possible utterance except in extremely constrained tasks, a hierarchical scheme is generally adopted to reduce the number of HMMs. First, entire sentences are modeled as sequences of words (constrained by a grammar). Furthermore, words are often concatenated from subword units (constrained by a lexicon), most commonly phones or triphones. One HMM with one or more states is then used to model these sub-word units.

Theory and methodology of HMMs are described in a number of sources, e. g. [Rabiner and Juang, 1993]. The fundamental equation describing this process is Bayes' rule, applied to speech recognition:

$$P(M|X,\Theta) = \frac{p(X|M,\Theta)P(M|\Theta)}{p(X|\Theta)} \tag{4.1}$$

in which Θ is the parameter set and $P(M|X,\Theta)$ is the posterior probability of the hypothesized HMM M given a sequence X of feature-vectors. Since this probability cannot be computed directly, it is usually split according to (4.1) into the acoustic model $p(X|M,\Theta)$ and a prior $P(M|\Theta)$ representing the language model.

The (full) acoustic likelihood is computed by expanding it into all possible state paths in M that can generate X:

$$p(X|M,\Theta) = \sum_{\forall S_j} p(X,S_j|M,\Theta) \tag{4.2}$$

where S_j are all possible paths of length N in M. For recognition, it is

usually approximated as

$$p^*(X|M,\Theta) = \max_{\forall S_j} p(X, S_j|M,\Theta) \tag{4.3}$$

which is also known as the "Viterbi"-approximation.

When decoding an observation X, we have to find the model M_j which maximizes $P(M|X,\Theta)$:

$$\begin{aligned} j &= \operatorname*{argmax}_{\forall i} P(M_i|X,\Theta) \\ &= \operatorname*{argmax}_{\forall i} p(X|M_i,\Theta)P(M_i|\Theta) \end{aligned} \tag{4.4}$$

since $P(X|\Theta)$ from (4.1) is a constant during recognition. The acoustic model $p(X|M_i,\Theta)$ is usually realized using Gaussian Mixture Models (GMMs). The parameters of the GMMs are estimated using the Expectation-Maximization-Algorithm (EM-Algorithm). However, the problem with the EM-Algorithm is that it only guarantees convergence to local optima. In contrast, discriminants exhibit global solutions as we have seen in the last chapter. Moreover, the outputs of discriminants can directly be interpreted as probabilities. Below we will describe how we can substitute GMMs by these probabilities [Andelić et al., 2006a] [Andelić et al., 2005] [Andelić et al., 2004].

In practice, the realization of one and the same phone differs a lot depending on its neighboring phones (the phone 'context'). Therefore *context dependent* phone models are most widely used. *Biphone models* consider either the left (preceding) or right (succeeding) phone, in *triphone models* both neighboring phones are taken into account, and for each phone different models are used for a different context. In Figure 4.3, the English word "bat" [b ae t] is shown in a monophone, biphone and triphone representation. The underlying sub-models for the phones or their combinations (in the bi- and triphone case) are in most cases HMMs. A phonetic alphabet of 40

phones results in a number of $40^3 = 64000$ possible triphones. However, only 3% of these theoretical possibilities occur in the language. Therefore we can train all existing triphone models The information coming from the language

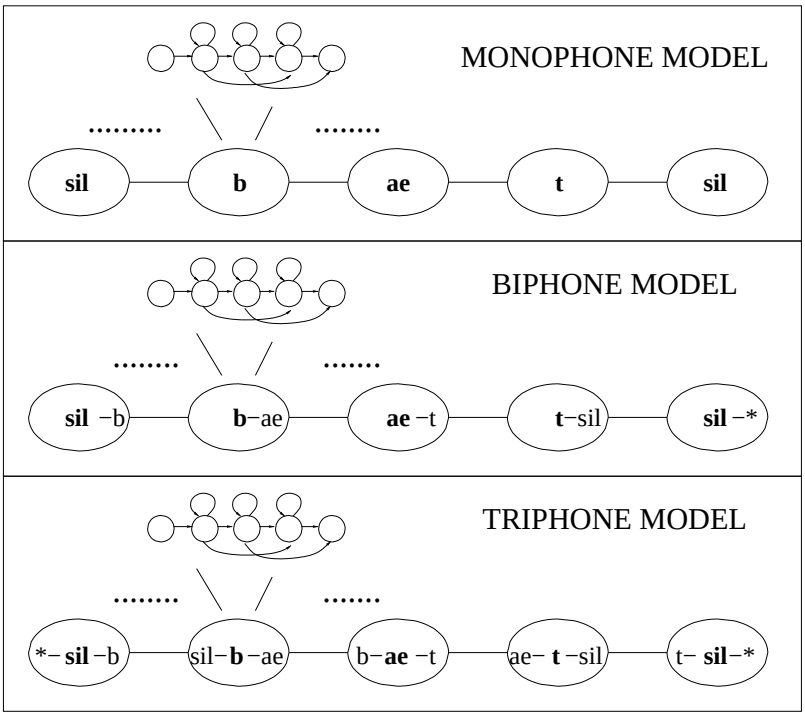

Figure 4.3: Monophone, biphone, and triphone HMMs for the English word "bat" [b ae t]. 'sil' stands for silence at the beginning and end of the utterance, which is modeled as a 'phone', too.

model and acoustic models as well as the information from the pronunciation dictionary has to be balanced during speech recognition. The performance of speech recognition systems is typically described in terms of word accuracy, A in %, defined as

$$A = 100 - \frac{S + I + D}{N} 100 \tag{4.5}$$

where N is the total number of words in the test set, and S, I, and D are the total number of substituted, inserted, and deleted words, respectively.

4.2 Using Nonlinear Discriminants for ASR

One of the advantages of Nonlinear Discriminants (NDs) is that their outputs may be interpreted as class-conditional probabilities due to the fact that they are normally distributed [Mika, 2002]. This is rather favorable compared to SVMs where the outputs have to be calibrated by a sigmoidal function in order to be interpretable as probabilities [Platt, 1998]. We will now describe how with the methods outlined in the last chapter the probabilistic outputs of NDs can be used to train the emission probabilities of HMM-states in ASR.

4.2.1 Probabilistic Outputs

Since the ND that is used here is a binary classifier the question arises how to deal with the probabilistic outputs in a multiclass scenario which we are actually faced with in ASR. Basically, there are two approaches for a multiclass discrimination problem. In the one-versus-rest (one-vs-rest) scheme [Schölkopf and Smola, 2002] each class is discriminated from all other classes. We will use this method together with KRLS. However, due to limitations in memory the one-vs-rest method is not accurately applicable when we use Nonlinear Pseudodiscriminants (NPDs) or Order-Recursive Least-Squares (OROLS). We use the one-versus-one (one-vs-one) classification method [Kressel, 1999] instead for the NPD and OROLS, i. e. we discriminate one class (one HMM-state in our case) from one other class.

Let us denote one arbitrary feature vector by $\mathbf{x}$ and the selected basis centers by $\mathbf{x}_n,\ n = 1, \ldots, m$. Furthermore, let k be a positive definite Mercer kernel. As we have seen in the last chapter, the discriminating direction $\mathbf{w}$ associated with a ND, can be expressed by an expansion in terms of

the nonlinearly mapped feature vectors. The methods derived in this thesis (NPD, OROLS, KRLS) can be used to compute the corresponding expansion coefficients $\alpha_n,\ n = 1, \ldots, m$. Note that the weight vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^T$ has to be normalized to length one in order to scale the projections that belong to different classification problems to the same range and thereby to obtain consistent probability estimates.

After training a ND associated with a classification problem which is to separate class i from class j, one single output, i. e. the projection of $\mathbf{x}$

$$q_{ij}(\mathbf{x}) = (\mathbf{w}\mathbf{x}) + b = \sum_{n=1}^{m} \alpha_n k(\mathbf{x}, \mathbf{x}_n) + b \tag{4.6}$$

onto the discriminating direction $\mathbf{w}$ may be used to compute the production probability of one single feature vector, i. e.

$$\begin{aligned} P_{ij}(\mathbf{x}|i) &= P_{ij}(q_{ij}(\mathbf{x})|i) \\ &= \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(q_{ij}(\mathbf{x}) - \mu_i)^2}{2\sigma_i^2}\right). \end{aligned} \tag{4.7}$$

The mean μ_i and the variance σ_i of the projected feature vectors $\mathbf{x}$ belonging to class i may be estimated consistently. Then the corresponding posterior probabilities $\nu_{ij} \equiv P_{ij}(i|\mathbf{x})$ may be obtained from each P_{ij} applying Bayes' rule. All pairwise probabilities ν_{ij} are transformed into one posterior probability using the pairwise coupling formula [Price et al., 1995]

$$P(i|\mathbf{x}) = \left[\sum_{j=1, j\neq i}^{T} \frac{1}{\nu_{ij}} - (T-2)\right]^{-1} \tag{4.8}$$

where T is the number of all classification problems. Every posterior probability $P(i|\mathbf{x})$ can then be transformed into a production probability applying again Bayes' rule $P(\mathbf{x}|i) \propto \frac{P(i|\mathbf{x})}{P(i)}$ where the probability $P(i)$ for the occurrence of class i is estimated by the relative frequency of the class in the training data. We use each of these production probabilities as an emission probability of the corresponding HMM-state.

4.3 Implementation

Our implementation of a hybrid HMM-system using acoustic models other than GMMs is based on the speech recognizer HTK [Young, 1996]. The framework uses a plugin-architecture which allows easy and flexible integration of arbitrary classifiers in HTK. An external classifier is contained in a Dynamically Loadable Library (DLL) with a few well-defined entry points. Measures are taken to make the classifier thread-safe. The DLL and the file from which to initialize the classifier must be specified to the recognizer in two additional fields in the HMM definition.

At startup, the classifier initializes its internal state from the file mentioned in the HMM definition. During recognition, the external classifier computes log-probabilities of speech frames $\mathbf{x}$ for given hypotheses h. In HTK, hypothesis are generated hierarchically, i.e. on sentence, word, and sub-word level. Due to this, a speech frame may be tested against the same hypothesis several times. Caching of results is performed to reduce computational overhead.

4.4 Experiments

4.4.1 Experimental Setup

We use the Resource Management 1 (RM1) corpus [Price et al., 1988]. The RM1 database is a collection of recordings of spoken sentences referred to naval resource management tasks. It was recorded with the support from the Defense Advanced Research Projects Agency (DARPA) Information Science and Technology Office. Different speakers of both genders with various US-American dialects read the sentences from written prompts in a low back-

ground noise environment.

We preprocess the speech data using a short-time FFT with a frame width of $25ms$ and a frame shift of $10ms$ followed by a cepstral analysis. The resulting feature-vectors contain 39 components (12 cepstral coefficients, the frame energy and the first and second order time differences). First, we compute a time alignment using a standard Gaussian mixture HMM decoder to get the state (label) for each feature-vector. The feature vectors have to be rescaled feature-wise to zero mean and standard deviation one in order to compensate the variability of the speech data due to different speakers. The test data are scaled using the means and standard deviations of the training features. The full 72-speaker training set (2880 sentences) is used for training. Our system is evaluated using the speaker independent Feb'89 test set. We use the standard RM word-pair grammar. Throughout all experiments the Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$ is used. The kernel parameter $\sigma = 8$ is optimized on the Oct'89 set. The forward selection for the NPD and KRLS during the training is terminated when the highest absolute value in the current residual is less than a predefined value. For OROLS we stop the forward selection when the regularization parameter λ has converged. We use 48 monophone models including a silence model to model the silence before and after the utterance and an optional short pause model with a single state to model the pauses between the words.

4.4.2 Results

Our baseline is a 3-state monophone HMM system trained on the full training set with GMMs consisting of 8 mixtures per state [Krüger et al., 2005]. The results are shown in table 4.2. The NPD and OROLS outperforms the HMM/GMM-system and is very competitive compared with a hybrid HMM-

based decoder using SVMs for the acoustic modeling [Krüger et al., 2005]. The results are very promising. Only the KRLS exhibits a poorer perfor-

classifier (3-state HMM)	Word accuracy
GMM with 8 mixtures	91.96%
SVM one-vs-one	94.10%
NPD one-vs-one	94.41%
OROLS one-vs-one	94.25%
KRLS one-vs-rest	90.1%

Table 4.2: Results on the RM1 Feb'89 test set.

mance on this task. We believe that the reason for this phenomenon is that we are limited in the number of selected basis centers when we use the one-vs-rest approach since in each classification problem all feature vectors are involved. Furthermore, it has to be noted that a standard HMM/GMM-system using triphones instead of monophones achieves about 97% word accuracy on the RM1 task. But note that we incorporate much more prior knowledge into the model when we use triphones since triphones are context-dependent and their acquisition requires a large amount of data. Thus, the use of monophones has its own justification. In situations where we are not able to acquire a large amount of speech data due to restrictions in terms of model complexity, our approach (including KRLS) could be a very good choice [Schafföner et al., 2006].

4.5 Summary

We have shown that the probabilistic outputs of nonlinear discriminants may be integrated into a hybrid HMM-based decoder for continuous speech recog-

nition. However, using kernel methods in the described way for a triphone system would be computationally intractable since there are too many classes for a one-vs-one classification. Thus, a possible step for future work could be to find a solution for this problem. For instance, Directed Acyclic Graphs (DAGs) [Platt et al., 2000] could be used instead of the one-vs-one scheme. DAGs could reduce the training and decoding times significantly and could make this approach applicable for larger datasets. Furthermore, finding better feature selection schemes and designing more appropriate kernels that lead to more accurate acoustic models could be focused.

Chapter 5

Experiments for Classification and Regression

To show the competitiveness of the proposed methods with other state-of-the art learning machines, we present and discuss in this chapter extensive empirical evaluations for classification and regression tasks. All benchmark datasets can be found in the UCI machine learning repository [Merz and Murphy,].

5.1 Classification

The first experiment for classification is the two spirals problem [Lang and Witbrock, 1988]. We do not perform this experiment in order to compare our results with other methods. We rather want to illustrate that kernel-methods are able to find an accurate decision boundary even if this boundary exhibits severe nonlinearities like in this case.

The two spirals problem is a synthetic problem. Nevertheless, it is known as a very hard task. For instance, neural networks are very hard to train on

this dataset. It can be seen from figure 5.1 that the NPD is able to construct a good decision boundary which is able to separate all training instances correctly even with 90 basis centers. Using 120 basis centers the decision boundary takes a smooth form with balanced distances in both directions towards the training data. This result indicates that the NPD can find solution which exhibit a large margin. This is very surprising insofar as discriminants are not constructed using the large margin principle. This suggests a good generalization ability of NPDs. Moreover, a SVM uses all training samples as support vectors [Billings and Lee, 2002] which indicates that SVMs are not necessarily maximally sparse.

As a conclusion we state that two important things are illustrated by this example. First, the use of Mercer kernel functions allows to incorporate powerful nonlinear directions into the model. And second, the NPD is a reasonable algorithmic approach to obtain a solution for these models.

5.1.1 Optical Character Recognition

For optical character recognition, 5 well-known benchmark datasets were chosen.

In all experiments the Gaussian kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{5.1}$$

is used. The kernel parameter σ is optimized using a 5-fold crossvalidation in all experiments. For the classification experiments the one-vs-rest approach is used to obtain a multiclass classification hypothesis.

The USPS dataset contains 256 pixel values of handwritten digits as training and testing instances. The letter dataset contains 20000 labeled samples. The character images were based on 20 different fonts and each letter within

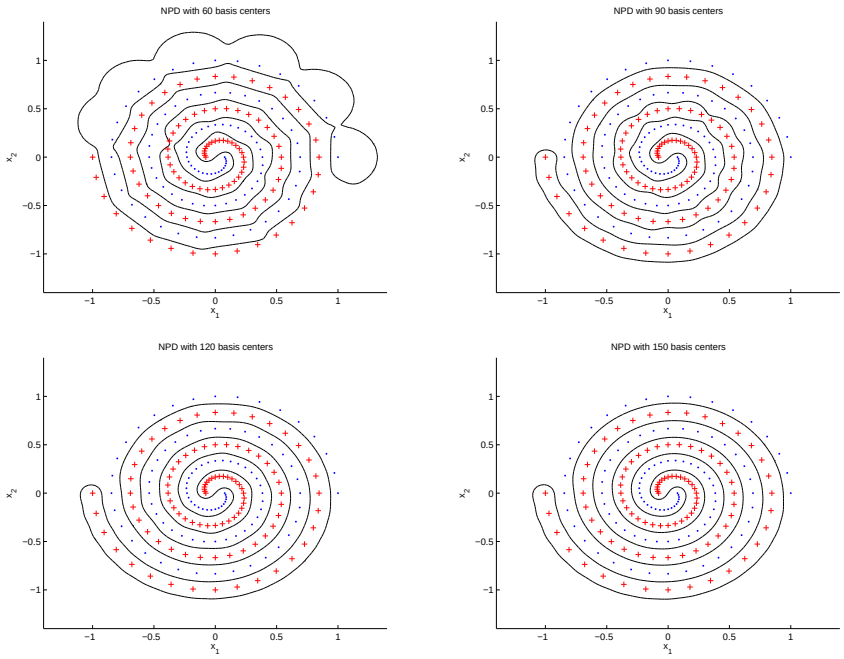

Figure 5.1: Decision boundaries on the two spirals classification problem using the NPD with different number of basis centers.

these fonts was randomly distorted to produce a dataset of unique stimuli. For this dataset no predefined split for training and testing exist. We used the first 16000 instances for training and the remaining 4000 instances for testing. Optdigits is a database of digits handwritten by Turkish writers. It contains digits written by 44 writers. The training set is generated from the first 30 writers and digits written by the remaining independent writers serve as testing instances. The database was generated by scanning and processing forms to obtain 32×32 matrices which were then reduced to 8×8. Pendigits contains pen-based handwritten digits. The digits were written down on a touch-sensitive tablet and were then resampled and normalized to a temporal sequence of eight pairs of (x, y) coordinates. The predefined test set is formed entirely from written digits produced by independent writers. The

satimage dataset was generated from Landsat Multi-Spectral Scanner image data. Each pattern contains 36 pixel values and a number indicating one of the six classes of the central pixel. The characteristics of the datasets are summarized in table 5.1. The recognition results can be seen in table 5.2. Especially for the optdigits and pendigits datasets NPD and OROLS appear to be significantly superior compared with SVMs. The performance on the remaining 3 datasets is comparable with SVMs. However, the NPD tends to select slightly more basis centers. We believe that at least for these datasets OROLS yields a better regularization due to the GCV criterion which adapts the regularization automatically.

Data set	# classes	# training	# testing
USPS	10	7291	2007
Letter	26	16000	4000
Optdigits	10	3823	1797
Pendigits	10	7494	3498
Satimage	6	4435	2000

Table 5.1: Datasets used for the classification experiments.

5.1.2 Other Benchmarks

13 artificial and real world benchmark datasets[1] for classification were chosen. For each of the 13 datasets randomly generated partitions for training and testing exist (20 partitions for Image and Splice and 100 partitions for all other). In all experiments the Gaussian kernel is used. The width of

[1]These datasets can be downloaded from http://ida.first.fhg.de/projects/bench/benchmarks.htm.

Data set	SVM	$OROLS$	NPD
USPS	4.3(3.9)	4.4(9.7)	4.4(10)
Letter	2.75(7)	2.61(4.3)	2.65(6.3)
Optdigits	2.73(12.1)	1.11(10.2)	1.2(10.4)
Pendigits	2.5(3.2)	1.66(1.9)	1.95(2.5)
Satimage	7.8(5.5)	8.2(7.5)	8.1(10.3)

Table 5.2: Test errors in % on 5 benchmark datasets. The one-vs-rest approach is used. Average fraction of selected basis centers in % within parentheses.

the kernel function, the regularization constant ε and the number of the selected input vectors (number of basis functions) are optimized on the first five training partitions of each datasets using a 5-fold crossvalidation for the NPD. For OROLS only the kernel width is optimized during a 5-fold crossvalidation procedure.

The NPD and OROLS are compared with Support Vector Machines ($SVMs$), Kernel Fisher Discriminant (KFD)(not sparse) and the Orthogonal Least Squares Algorithm (OLS). The results for these methods are taken from [Billings and Lee, 2002] [Rätsch et al., 2001]. It can be seen from table 5.3 that both NPD and OROLS are comparable or better than the other state-of-the-art classifiers.

5.2 Regression

For regression, we first want to illustrate how OROLS perform on noisy datasets since we expect OROLS to be robust against noise due to the automatically adapted regularization. To this end we use a synthetic dataset

Table 5.3: Estimation of generalization errors on 13 benchmark data sets in % with standard deviations and sparsity levels in % within brackets (best result in **bold face**, second *emphasized*).

Data set	*SVM*	*KFD*	*OLS*	*OROLS*	*NPD*
Banana	11.5±0.7(78)	10.8±0.5	10.7±0.5(93)	*10.6±0.4*(90)	**10.5±0.4**(87)
B.Cancer	26.0±4.7(42)	**24.8±4.6**	*25.8±4.7*(96)	26.8±4.8(95)	26.8±4.8(95)
Diabetis	23.5±1.7(57)	23.2±1.6	*23.1±1.8*(98)	*23.1±1.7*(93)	**23.0±1.8**(91)
F.Solar	**32.4±1.8**(9)	*33.2±1.7*	33.6±1.6(99)	33.6±1.7(95)	33.5±1.8(94)
German	**23.6±2.0**(58)	*23.7±2.2*	24.0±2.3(99)	23.8±2.1(91)	23.9±2.2(89)
Heart	*16.0±3.2*(51)	16.1±3.4	**15.8±3.4**(98)	16.1±3.4(91)	16.2±3.4(91)
Image	3.0±0.6(87)	4.8±0.6	**2.8±0.6**(78)	*2.9±0.6*(77)	**2.8±0.6**(74)
Ringnorm	1.7±0.1(62)	**1.5±0.1**	*1.6±0.1*(98)	1.8±0.1(95)	1.8±0.1(94)
Splice	*10.9±0.7*(31)	**10.5±0.6**	11.7±0.6(67)	11.7±0.7(55)	11.7±0.8(50)
Thyroid	4.8±2.2(79)	*4.2±2.0*	4.6±2.4(84)	*4.2±2.0*(83)	**4.1±1.9**(82)
Titanic	*22.4±1.0*(10)	23.3±2.0	*22.4±1.0*(93)	*22.4±1.0*(90)	**22.3±1.0**(73)
Twonorm	3.0±0.2(82)	**2.6±0.2**	*2.7±0.2*(97)	*2.7±0.2*(85)	**2.6±0.2**(75)
Waveform	**9.9±0.4**(60)	**9.9±0.4**	10.0±0.4(96)	10.0±0.4(60)	10.0±0.5(51)

based on the function $\mathrm{sinc}(x) = \sin(x)/x,\ x \in (-10, 10)$ which is corrupted by Gaussian noise. The sinc function is a good choice since the signal to noise ratio varies heavily for different function values. All training and testing instances are chosen randomly using a uniform distribution on the same interval. The results are illustrated in figures 5.2-5.4 and table 5.4. We see from Fig.5.2 that OROLS is able to identify the true function even in presence of noise with a high standard deviation. Moreover, Fig.5.3 shows that the Root Mean Square Error (RMSE) decreases rapidly with increasing number of training instance and converges to a highly confident value. In Fig.5.4 we see that the RMSE increases only in a ratio of about 0.3 with respect to the

noise standard deviation. OROLS is very competitive on this task compared to SVMs. We can conclude that the regularization incorporated in OROLS is responsible for this good generalization.

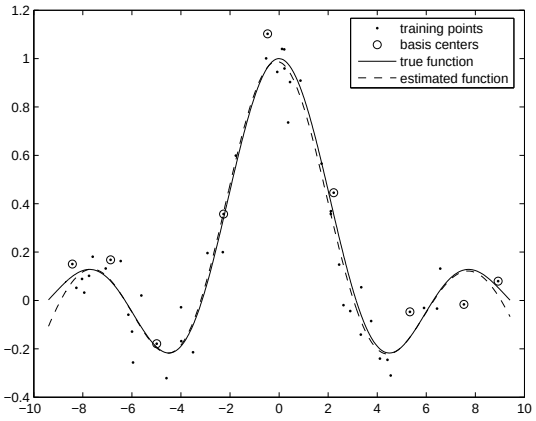

Figure 5.2: Example fit to a noisy sinc function for OROLS using 50 randomly generated points for training and testing. The standard deviation of the Gaussian noise is 0.1. The Root Mean Square Error (RMSE) is 0.0269 in this case. 9 points are selected as basis centers.

Method	RMSE
SVM	0.0519
OROLS	0.0431

Table 5.4: Average RMSE for the sinc experiment using the SVM and OROLS. 50 / 1000 randomly generated points are used for training / testing. The standard deviation of the Gaussian noise is 0.1 in all runs. The results are averaged over 100 runs.

Additionally, the two real world datasets Boston and Abalone, which are available from the UCI machine learning repository, are chosen. The hy-

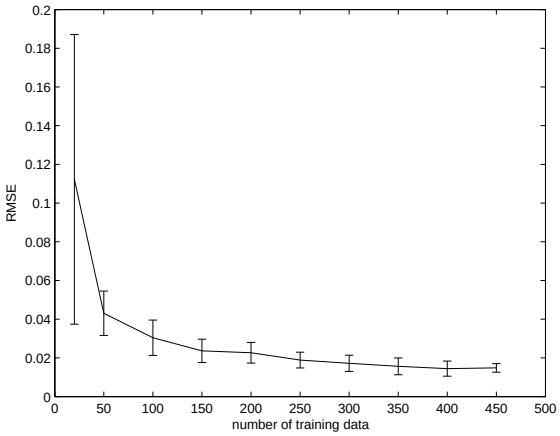

Figure 5.3: RMSE of fits to a noisy sinc function w. r. t. different training set sizes using OROLS. 1000 randomly generated points are used for testing. The standard deviation of the Gaussian noise is 0.1 in all runs. The results are averaged over 100 runs for each size.

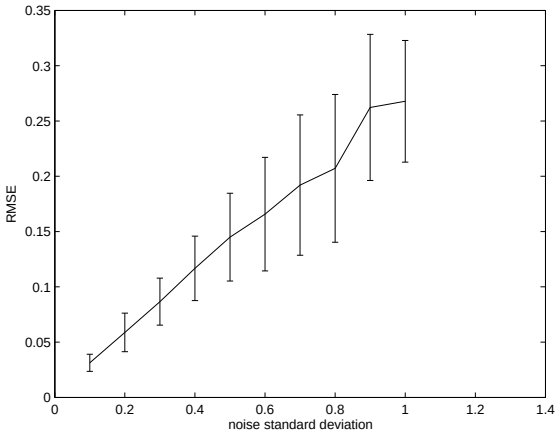

Figure 5.4: RMSE of fits to a noisy sinc function w. r. t. different noise levels using OROLS. 100 / 1000 randomly generated points are used for training / testing. The results are averaged over 100 runs for each noise level.

perparameters are optimized in a 5-fold crossvalidation procedure. For both datasets, random partitions of the mother data for training and testing are generated (100 (10) partitions with 481 (3000) instances for training and 25 (1177) for testing for the Boston and Abalone dataset, respectively). All continuous features are rescaled to zero mean and unit variance for both Abalone and Boston. The gender encoding (male / female /infant) for the Abalone dataset is mapped into $\{(1,0,0),(0,1,0),(0,0,1)\}$. The Mean Squared Error (MSE) of OROLS and NPD is compared with a forward selection algorithm based on a QR-decomposition of the Gram matrix [Nair et al., 2002]. The results in table 5.5 show that the MSE is improved significantly by OROLS and NPD. It should be noted that the best performance of OROLS and NPD for the Boston dataset is quite favorable compared with the best performance of SVMs (MSE 8.7 ± 6.8) [Schölkopf and Smola, 2002].

Dataset	QR	OROLS	NPD
Boston	8.35±5.67	7.92±3.46	7.66±3.66
Abalone	4.53±0.29	4.32±0.15	4.34±0.16

Table 5.5: Mean Square Error (MSE) with standard deviations for the Boston and Abalone dataset using different methods.

5.3 Summary

It was shown that the computationally efficient training algorithms derived in the last chapter are very competitive with other state-of-the-art methods. Both NPD and OROLS are forward selection methods and are very easy to implement with low memory requirements and reasonable scaling properties . The advantage of the OROLS algorithm is that due to the novel

orthogonal decomposition scheme the GCV can easily be incorporated as an effective stopping criterion. Furthermore, the GCV criterion allows to adapt the regularization parameter in each iteration automatically. Extensive empirical studies using synthetic and real-world benchmark datasets for classification and regression suggest that the proposed methods are able to construct models with a very competitive generalization ability. The advantage of the proposed methods compared with e. g. SVMs is their simplicity. Sparsity is achieved in a computationally efficient way by construction and can hence better be controlled than in the SVM case where a optimization problem is to be solved. Furthermore, in contrast to SVMs both OROLS and NPD allow an easy incorporation of multiple kernels, i e. the kernel parameters may be varied for different training instances in order to obtain more flexible learning machines. This possibility is not examined here and may be an interesting direction for future work. A further step for future work could be the development of the proposed algorithm for tasks like dimensionality reduction or online-learning.

Chapter 6

Conclusion and Future Work

In this thesis algorithms for supervised learning and their applications for classification, regression and speech recognition were considered.

After reviewing some basic facts from statistical learning theory we gained two very important insights. First, consistency is crucial for successful learning. And second, not the dimensionality of the data but the complexity of the function class we choose our functions from is important in order to obtain consistent learning rules. These facts served as a solid theoretical explanation for choosing learning machines that implement exclusively linear functions. Linear functions exhibit a small and easily controllable complexity but on the other hand their performance in real-world problems is very limited. To overcome this problem we introduced the so-called kernel functions which allow to turn every algorithm into a nonlinear one as long as the algorithm may be formulated exclusively in terms of dot-products. Using kernel functions we are able to implement a variety of nonlinear mappings implicitly. At the same time, we are able to keep the algorithm linear in the new nonlinear feature-space and to benefit thereby from the theoretical advantages of linear functions.

The SVM served as an important example to show how these theoretical insights together with the possibility to incorporate kernel functions can be used algorithmically. Furthermore, the SVM indicated that regularization is deeply connected with the notion of sparsity. Regularization is the process of restricting the function class and is hence an important technique for achieving consistent solutions. Regularization can appear in different forms. In the SVM case regularization is performed indirectly by maximizing the margin. The consequence is that we obtain a sparse solution. In contrast, the drawback of discriminants is that they are not sparse in general. This fact is also revealed by the equivalence between the discriminant approach and least-squares models. However, at the same time this equivalence indicates an important advantage of the discriminant approach since least-squares models are very favorable as for their Bayes-optimality in certain classification settings. Therefore we pursued a combination of these two concepts by incorporating sparsity into least-squares models. Thus, compared to the SVM we went the other way around and performed regularization in a discrete manner by controlling the sparsity directly.

After reviewing the theoretical justification for this procedure we developed greedy algorithms for solving sparse kernel-based discriminants. Thereby we exploited the fact that the use of Mercer kernels leads to a positive definite Gram matrix and hence we were able to examine only the current residual as a subset selection criterion or to approximate the full Gram matrix. Furthermore, starting with a simple update scheme for the pseudoinverse we derived an order-recursive algorithm for an orthogonal decomposition of the reduced Gram matrix without increasing the cost in terms of memory and computational time significantly. Thus we were able to regularize the solution even further by analytically penalizing the length of the solution. The

prize we had to pay for this improved regularization was the introduction of an additional parameter. On the other hand, this regularization parameter can make the solution more robust against noisy data and can be used as an effective stopping criterion.

In a large collection of experiments we demonstrated that the generalization abilities of the methods proposed in this thesis are very competitive with other state-of-the-art techniques. Since different datasets have widely varying characteristics it would be not legitimate to claim that one algorithm is generally superior over the others. But we believe that in cases where very sparse solutions can be achieved which is very often the case our methods are a very good choice due to their simplicity and reasonable cost. However, this dependance on the sparsity is also a drawback of our methods since they loose their advantageous properties when a sparse solution can not be achieved without sacrificing a good generalization. But other methods like the SVM can not guarantee a sparse solution neither. A further drawback at least of the NPD and OROLS is that they can not be applied on very large datasets. Therefore we were not able to use the NPD and the OROLS-algorithm for speech recognition in a one-vs-rest setting. A possible solution to this problem could be the design of special kernel functions which yield a sparse Gram matrix. For instance, we might set the kernel evaluations of two very distant points to zero. Such a method could be able to decrease the runtime and memory requirements of the proposed update schemes considerably. We leave this as a suggestion for future work.

A second direction of future work could be the examination of the interplay between sparsity and the accuracy of the class-conditional probabilities. An additional advantage of our methods is that their outputs can be interpreted probabilistically in a natural way. We have demonstrated this fact

in the speech recognition experiments. However, answering the question if there is a relation between sparsity and the class-conditional probabilities and if one can derive bounds for this relation could be of great theoretical and practical significance especially for speech recognition.

Declaration

No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Persönlicher Werdegang

29.01.1977: Geboren in Hagen, Westfalen

09/1983–06/1987: Grundschule Funckeparkschule in Hagen

09/1987–06/1996: Allgemeine Hochschulreife am Albrecht-Dürer-Gymnasium in Hagen (Abschlussnote: 1,7)

10/1997–12/2002: Studium der Elektrotechnik an der Ruhr-Universität Bochum (Abschlussnote: sehr gut (1,5))

Vertiefungsrichtungen: Signaltheorie, Digitale Signalverarbeitung, Elektromagnetische Felder

Titel der Diplomarbeit: *Aspekte einer einheitlichen Systemtheorie für reelle, komplexe und hyperkomplexe Multiratensysteme*
Betreuer: Prof. Dr. -Ing. Heinz G. Göckler

05/2003–12/2006: Doktorand am Lehrstuhl "Kognitive Systeme", Otto-von-Guericke-Universität Magdeburg
Betreuer: Prof. Dr. rer. nat. Andreas Wendemuth

seit 03/2007: Entwicklungsingenieur bei der Endress+Hauser Conducta GmbH + Co. KG in Gerlingen

Arbeitsgebiete: Digitale Signalverarbeitung, Methoden des maschinellen Lernens für die automatische Sensorwartung

Bibliography

[Andelić et al., 2004] Andelić, E., Schafföner, M., Katz, M., Krüger, S. E., and Wendemuth, A. (2004). Iterative implementation of the kernel fisher discriminant for speech recognition. In *9th Conference on Speech and Computer (SPECOM 2004), St. Petersburg, Russia*.

[Andelić et al., 2005] Andelić, E., Schafföner, M., Katz, M., Krüger, S. E., and Wendemuth, A. (2005). Acoustic modelling using kernel-based discriminants. In *10th International Conference on Speech and Computer (SPECOM 2005), October 17-19, 2005, University of Patras, Patras, Greece*.

[Andelić et al., 2006a] Andelić, E., Schafföner, M., Katz, M., Krüger, S. E., and Wendemuth, A. (2006a). A hybrid hmm-based speech recognizer using kernel-based discriminants as acoustic models. In *18th International Conference on Pattern Recognition (ICPR 2006), 20-24 August 2006, Hong Kong, China*, pages 1158–1161. IEEE Computer Society.

[Andelić et al., 2006b] Andelić, E., Schafföner, M., Katz, M., Krüger, S. E., and Wendemuth, A. (2006b). Kernel least-squares models using updates of the pseudoinverse. *Neural Computation*, 18(12):2928–2935.

[Andelić et al., 2007] Andelić, E., Schafföner, M., Katz, M., Krüger, S. E., and Wendemuth, A. (2007). Updates for nonlinear discriminants. In *20th International Joint Conference on Artificial Intelligence (IJCAI 2007), 06-12 January 2007, Hyderabad, India*. AAAI Press.

[Aronszajn, 1950] Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*.

[Baudat and Anouar, 2000] Baudat, G. and Anouar, F. (2000). Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404.

[Ben-Israel and Greville, 1977] Ben-Israel, A. and Greville, T. N. E. (1977). *Generalized Inverses: Theory and Applications*. Wiley.

[Billings and Lee, 2002] Billings, S. A. and Lee, K. L. (2002). Nonlinear fisher discriminant analysis using a minimum squared error cost function and the orthogonal least squares algorithm. *Neural Networks*, 15:263–270.

[Bishop, 1995] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

[Chen et al., 1991] Chen, S., Cowan, C. F. N., and Grant, P. M. (1991). Orthogonal least squares learning for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309.

[Chen et al., 1998] Chen, S., Donoho, D. L., and Saunders, M. A. (1998). Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61.

[Christianini and Shawe-Taylor, 2000] Christianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. N. (1995). Support vector networks. *Machine Learning*, 20:273–297.

[Couvreur and Bresler, 2000] Couvreur, C. and Bresler, Y. (2000). On the optimality of the backward greedy algorithm for the subset selection problem. *SIAM Journal on Matrix Analysis and Applications*, 21(3):797–808.

[De Kruif and De Vries, 2003] De Kruif, B. J. and De Vries, T. J. A. (2003). Pruning error minimization in least squares support vector machines. *IEEE Transactions on Neural Networks*, 14(3):696–702.

[Duda and Hart, 1973] Duda, R. O. and Hart, P. E. (1973). *Pattern classification and scene analysis*. John Wiley and Sons.

[Engel et al., 2003] Engel, Y., Mannor, S., and Meir, R. (2003). The kernel recursive least squares algorithm. Technical report, Interdisciplinary Center for Neural Computation, Jerusalem, Israel.

[Fisher, 1936] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188.

[Gold and Morgan, 1999] Gold, B. and Morgan, N. (1999). *Speech and audio signal processing: processing and perception of speech, and music*. John Wiley and Sons Inc.

[Grote and Huckle, 1997] Grote, M. J. and Huckle, T. (1997). Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18:838–853.

[Gu and Wahba, 1991] Gu, C. and Wahba, G. (1991). Minimizing gcv/gml scores with multiple smoothing parameters via the newton method. *SIAM Journal on Scientific and Statistical Computing*, 12(2):383–398.

[Haykin, 2002] Haykin, S. (2002). *Adaptive Filter Theory*. Prentice Hall, New Jersey.

[Hoegaerts et al., 2004] Hoegaerts, L., Suykens, J. A. K., Vanderwalle, J., and De Moor, B. (2004). A comparison of pruning algorithms for sparse least squares support vector machines. In *Proceedings of the 11th International Conference on Neural Information Processing (ICONIP 2004)*, Calcutta, India.

[Huang et al., 2001] Huang, X., Acero, A., and Hon, H. (2001). *Spoken Language processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall.

[Kressel, 1999] Kressel, U. (1999). *Advances in Kernel Methods: Support Vector Learning*, chapter Pairwise Classification and Support Vector Machines, pages 255–268. MIT Press.

[Krüger et al., 2005] Krüger, S. E., Schaföner, M., Katz, M., Andelić, E., and Wendemuth, A. (2005). Speech recognition with support vector machines in a hybrid system. In *9th European Conference on Speech Communication and Technology*, pages 993–996.

[Lang and Witbrock, 1988] Lang, K. J. and Witbrock, M. J. (1988). Learning to tell two spirals apart. In *Proceedings of the 1988 Connectionist Models Summer School*.

[MacKay, 1992] MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

[Mallat and Zhang, 1993] Mallat, S. and Zhang, Z. (1993). Matching pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, 41:3397–3415.

[Mercer, 1909] Mercer, J. (1909). Functions of positive and negative type and their connections to the theory of integral equations. In *Philos. Trans. Roy. Soc.*, pages A 209:415–446, London.

[Merz and Murphy,] Merz, C. J. and Murphy, P. M. The uci machine learning repository. http://mlearn.ics.uci.edu/MLRepository.html.

[Mika, 2002] Mika, S. (2002). *Kernel Fisher Discriminants*. PhD thesis, Technical University Berlin.

[Nair et al., 2002] Nair, P., Choudhury, A., and Keane, A. J. (2002). Some greedy learning algorithms for sparse regression and classification with mercer kernels. *Journal of Machine Learning Research*, 3:781–801.

[Natarajan, 1995] Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM Journal of Computing*, 25:227–234.

[Natarajan, 1999] Natarajan, B. K. (1999). On learning functions from noise-free and noisy examples via occam's razor. *SIAM Journal of Computing*, 29:712–727.

[Orr, 1995] Orr, M. (1995). Regularisation in the selection of radial basis function centres. *Neural Computation*, 7:606–623.

[Platt et al., 2000] Platt, J., Cristianini, N., and Shawe-Taylor, J. (2000). Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems 12*, pages 547–553.

[Platt, 1998] Platt, J. C. (1998). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

[Price et al., 1995] Price, D., Knerr, S., Personnaz, L., and Dreyfus, G. (1995). Pairwise neural network classifiers with probabilistic outputs. In *Advances in Neural Information Processing Systems*, volume 7. The MIT Press.

[Price et al., 1988] Price, P., Fisher, W., Bernstein, J., and Pallett, D. (1988). The darpa 1000-word resource management database for continuous speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*.

[Rabiner and Juang, 1993] Rabiner, L. and Juang, B. (1993). *Fundamentals of Speech Recognition*. Englewood Cliffs NJ.

[Rätsch et al., 2001] Rätsch, G., Onoda, T., and Müller, K.-R. (2001). Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320. also NeuroCOLT Technical Report NC-TR-1998-021.

[Rifkin et al., 2003] Rifkin, R., Yeo, G., and Poggio, T. (2003). Regularized least squares classification. In Suykens, J. A. K., Horvath, G., Basu, S., Micchelli, C., and Vandewalle, J., editors, *Advances in Learning Theory: Methods, Models and Applications*, volume 190 of *NATO Science Series III: Computer and Systems Sciences*, chapter 7, pages 131–154. IOS Press, Amsterdam.

[Roth and Steinhage, 1999] Roth, V. and Steinhage, V. (1999). Nonlinear discriminant analysis using kernel functions. In *Advances in Neural Information Processing Systems NIPS 12*, pages 568–574. MIT Press.

[Schafföner et al., 2006] Schafföner, M., Krüger, S. E., Andelić, E., Katz, M., and Wendemuth, A. (2006). Limited training data robust speech recognition using kernel-based acoustic models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*.

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. MIT Press.

[Smola and Bartlett, 2001] Smola, A. J. and Bartlett, P. L. (2001). Sparse greedy gaussian process regression. In *Advances in Neural Information Processing Systems NIPS 14*. MIT Press.

[Smola and Schölkopf, 2000] Smola, A. J. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 911–918. Morgan Kaufmann.

[Suykens and Vandewalle, 1999] Suykens, J. A. K. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300.

[Van Gestel et al., 2002] Van Gestel, T., Suykens, J. A. K., Lanckriet, G., Lambrechts, A., De Moor, B., and Vandewalle, J. (2002). A bayesian framework for least squares support vector machine classifiers, gaussian processes and kernel fisher discriminant analysis. *Neural Computation*, 14(5):1115–1147.

[Vapnik, 1998] Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley and Sons.

[Vapnik and Chervonenkis, 1974] Vapnik, V. N. and Chervonenkis, A. Y. (1974). *Theory of Pattern Recognition*. Nauka. in Russian.

[Vapnik and Chervonenkis, 1991] Vapnik, V. N. and Chervonenkis, A. Y. (1991). The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 3(1):283–305.

[Wahba, 1979] Wahba, W. (1979). How to smooth curves and surfaces with splines and cross validation. In *Proceedings of the 24th Conference on the Design of Experiments*. US Army Research Office.

[Wendemuth, 1995] Wendemuth, A. (1995). Learning the unlearnable. *Journal of Physics A*, 28:5423–5436.

[Young, 1996] Young, S. (1996). *The HTK-Book*. Cambridge University Press.