Support Vector Machines, Generalization Bounds, and Transduction

Rodney Stephen Kroon

Thesis presented in partial fulfilment
of the requirements for the degree of
Master of Commerce
at the University of Stellenbosch

Promoter: Prof. C. W. Omlin
December 2003

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: .... Date: ....

Abstract

A common technique for function estimation is the support vector machine (SVM), proposed in the 1990s. This thesis aims to place the SVM in the learning theory framework, by considering the major issues in supervised learning, and investigating how SVMs deal with them.

The principal considerations when designing a new supervised learning technique are discussed shortly to provide context. We then consider the SVM algorithm, its inherent complexity control, implementation issues, applications and model selection. There is a particular focus on model selection through generalization bounds. The result is hopefully a clear, introductory tutorial on SVMs. The SVM is placed in its context as a regularization problem, and its relationship to structural risk minimization (SRM) is made clear.

We also consider the analogue of the SVM in the transductive learning environment, where the potential test points are known before constructing the learning machine. In this situation, the training and test points together can be seen as a finite input space. Using this, we derive scale-sensitive probably approximately correct (PAC) bounds which are applicable to transduction. The bounds can be applied to classification (including multi-class problems) and regression. We apply the bounds to classes of linear functions, providing transductive bounds on a number of techniques, including SVMs, ridge regression, and least squares estimation.

Keywords: SVM, tutorial, transduction, PAC bounds, model selection, covering numbers

Opsomming

'n Algemene tegniek vir funksieberaming is die ondersteuningspuntmetode (SVM) wat in die negentigerjare voorgestel is. Hierdie tesis poog om dié metode in die raamwerk van die leerteorie te plaas, deur die manier waarop die SVM die hoofoorwegings van leer onder toesig aanpak, te ondersoek.

Die belangrikste faktore om in ag te neem met die ontwerp van 'n nuwe metode vir leer onder toesig word kortliks bespreek om konteks te verskaf. Ons bespreek dan die SVM algoritme, sy inherente kompleksiteitsbeheer, implementeringsoorwegings, toepassings, en modelkeuse. Daar is veral 'n fokus op modelkeuse deur middel van veralgemeningsgrense. Dit het hopelik 'n duidelike inleiding tot die SVM as gevolg. Die SVM word in die konteks van reguleringsprobleme geplaas, en sy verhouding aan strukturele risikominimering word uitgewys.

Ons oorweeg ook die parallel van die SVM in transduksie, waar die moontlike toetspunte bekend is voor die bou van die leermasjien. In hierdie geval, kan die leer- en toetsdata saam as 'n eindige toevoerruimte beskou word. Deur hiervan gebruik te maak, lei ons skaal-sensitiewe waarskynlik byna korrek grense wat van toepassing op transduksie is, af. Dié grense kan vir klassifikasie (insluitende meervoudige-teiken klassifikasie) en regressie toegepas word. Ons pas die grense toe op klasse lineêre funksies, en verkry sodoende transduktiewe grense op 'n verskeidenheid tegnieke, insluitend die SVM, rifregressie, en kleinste kwadrate beraming.

Sleutelbegrippe: SVM, tutoriaal, transduksie, PAC grense, modelkeuse, bedekkingsnommers

For God, who has blessed me richly.
May the fruits of my labour please You.

Contents

List of Figures

Acknowledgements

I would like to thank the following people and organizations:

- my family, Dalene de Beer, and her family (in particular Ria de Beer), for their support and encouragement;
- my promoter, Christian Omlin;
- my fellow students who provided a pleasant working environment the last few years, in particular James Connan, Walter Andrag, Tiaan Wessels, Dane Walsh, Andries Kruger, Ben Bredenkamp and Sean Snyders;
- Sarel Steel for his encouragement, and also his helpful discussions and proofreading;
- Yann Guermeur for his helpful comments and discussions, and his interest in my work;
- Bernhard Schölkopf for prompt, helpful replies to my questions;
- Alexander Smola, for his useful comments and corrections;
- the various researchers who sent me papers in response to email requests;
- anonymous referees for comments on my work;
- Konrad Scheffler, Andries Kruger, Ben Bredenkamp and Bruce Bartlett for proofreading parts, and suggesting improvements and corrections;

- the South African Technology for Human Resources and Industry Programme (THRIP) for funding;
- Standard Bank for their bursary support;
- Andries Kruger and Karen Müller for help with LaTeX;
- Andrew Inggs for the use of his LaTeXthesis template;
- the administrators of the UCI and DELVE repositories [17, 184]; and
- all my other friends who kept me from giving up on this in exasperation.

Chapter 1

Introduction

1.1 Motivation

The support vector machine (SVM) [20, 24, 36, 166] has proven to be an extremely useful machine learning technique in the last decade. SVMs are based on the structural risk minimization (SRM) [190, 191] inductive principle. SRM aims at successful learning by minimizing bounds on generalization error.

Another result of research in SRM was the proposal of transductive learning [190, 191]. In 1999, Thorsten Joachims proposed the transductive support vector machine (TSVM) in the context of text classification [89, 93, 95, 171].

This thesis places SVMs in the learning theory framework, by investigating how they address the typical issues common to all supervised learning techniques. The result is an easily accessible introductory tutorial on SVMs. Most mathematical results are relegated to an appendix, to aid the flow of the main text. Particular attention is given to the use of generalization bounds for model selection. Due to the amount of work which has been done on SVMs and kernel techniques in the last decade, many of the sections are of necessity sparsely detailed. In these cases, I have attempted to provide useful references for the interested reader.

1.2 Problem Statement

One of the problems often encountered by the new graduate student's first meetings with SVMs, is that the usual introductory material assumes a large amount of mathematical background. This is no great obstacle for students who wish only to apply the SVM in a particular problem area: to them, examining each plank of the theoretical hull of the SVM ship is hardly necessary to instruct it which direction to sail. However, it does serve as somewhat of an impediment to the aspiring repairman, and ship-builder.

This is the gap which I have attempted to fill with this thesis: making the theoretical aspects of the SVM more easily accessible to new students in the field.

The use of generalization bounds for model selection is especially relevant for SVMs. In the case of transductive learning, many of the familiar techniques can be employed to provide such bounds. Part of this thesis is devoted to the derivation of generalization bounds for transductive learning[1], in particular for the TSVM and related techniques. It is hoped that the approach followed is clear enough to help guide students through the derivation of other bounds.

1.3 Objectives

This thesis then attempts to provide an introductory tutorial on SVMs, which is highly accessible to students without an extensive mathematical background. The tutorial also shows how SVMs handle the principal issues of supervised learning techniques. Mathematical background was placed in an appendix as far as possible, so as not to obscure the main principles underlying the SVM.

We also hope to use the derivation of a slightly improved version of a known generalization bound for the special case of a finite input space to illustrate

[1] The bounds are derived for a finite input space. Considering the points in the finite training set and test set as the input space yields bounds on transductive learning.

the principles behind probably approximately correct (PAC) bounds. The importance of such bounds and other error estimates for model selection will be made clear.

1.4 Thesis Outline

We begin by considering the learning milieu, and where SVMs can be placed in the learning framework in Chapter 2. In this chapter, we also highlight the major issues of supervised learning techniques. It is in this context that we consider the SVM in Chapters 3 and 4. Chapter 3 introduces the SVM algorithm for various problem types, and considers implementation issues and applications of SVMs. Chapter 4 considers the problem of model selection from the perspective of generalization error. A generalization bound applicable to transductive learning is derived, and applied to linear methods using techniques described in the chapter. We conclude by reviewing what has been achieved in Chapter 5, and consider some avenues for further research and open problems.

Chapter 2

Learning Theory

In this chapter, we provide an overview of some important aspects of learning theory and its development, and give an introduction to risk minimization approaches to learning.

2.1 Introduction

Learning attempts to identify the structure in a system and incorporate it into some model [196]. The typical problems encountered in learning are discussed in Section 2.2. In Section 2.3, we introduce and discuss common concerns which must be addressed by learning techniques. In Section 2.4, we give a short history of the development of learning theory.

Many learning techniques make use of some form of risk minimization, which is outlined in Section 2.5. The major approaches to risk minimization are structural and empirical risk minimization (SRM and ERM), which are distinguished in Sections 2.6 and 2.7.

In Section 2.8, we discuss the ideas and purpose of statistical learning theory (SLT) with respect to SRM. This work leads to the development of the SVM that will be introduced in Chapter 3.

2.2 The Learning Milieu

There are two major types of learning, supervised and unsupervised. The role of unsupervised learning is the detection (exploratory techniques) or verification (confirmatory techniques) of structure in the input data. These techniques can thus make the data easier to understand and process sensibly; one frequently finds unsupervised learning techniques used for preprocessing step before applying supervised learning techniques (e.g. [12]). Two of the major fields in exploratory unsupervised learning are data clustering [52, 88] and dimension reduction techniques such as principal components analysis (PCA) [97] and factor analysis [103, 102]).

Supervised learning involves estimating some relationship between input and output values, with the aid of training data which is typically composed of a number of input-output pairs. These techniques will be the focus of the rest of the thesis.

Fields where SVMs have already been successfully applied include *pattern recognition* or *classification* [20, 36], *regression estimation* [193], *density estimation* [193, 196] and *time series prediction* [54, 125, 127]. These are the three major types of *supervised learning* or *function estimation* problems [190, 191, 196]. A brief outline of these three problem types is given below, although the classification problem will be given preference when the SVM is introduced.

2.2.1 Classification

Classification is the problem of assigning a situation, event, object or observation to one or more of a finite number of prespecified classes [46]. Any pattern classification problem can be broken up into a number of simpler *dichotomous* or 2-class problems. For this reason, the dichotomous problem receives extensive treatment in the literature. Better solutions to the dichotomous problem also lead naturally to improved results in *polychotomous* or multi-class problems [190].

Typically, the training set for classification problems consists of a number of observations, together with the class membership of each observation.

2.2.2 Regression Estimation

In many cases, one is interested in finding the average value of a phenomenon, given the observation of a different phenomenon, i.e. given an observation x, one is interested in estimating the average value of a variable y [190]. This is the problem of regression estimation. Typically, one is only interested in doing this if one believes that some relationship exists between x and y. In most cases, y is a scalar value, while x is a vector.

The training data for regression estimation is usually a set of observations x_i and a realization y_i associated with each observation.

Time Series Prediction

A special case of regression estimation is that of time series prediction. Here, the problem is that of estimating the next observation in a temporal sequence, given the last n observations. In this case, x is a vector generated by concatenating the last n observations, while y is the next observation.

The training data in this case is a sequence of observations, or a *time series*.

One major stumbling block in time series prediction is *nonstationarity* [83]. A time series is said to be stationary if the points of the time series are independent of the actual time at which they occur. Thus, nonstationary time series need to be provided the time of each observation as part of the input, and the output will depend on the time. Of course, the introduction of time as an extra parameter makes the prediction problem more complex.

Another troublesome aspect, not limited to regression, is dependence between observations. This complication is usually sidestepped by deriving theory based on the assumption of independence of observations.

2.2.3 Density Estimation

The problem of density estimation is one of the central problems of mathematical statistics [190], and involves the estimation of the probability density function (p.d.f.) $f(t)$ of a distribution, given a sample cumulative distribution function (sample c.d.f.) $F_s(t)$.

The Glivenko-Cantelli theorem states that the sample c.d.f converges in probability to the underlying c.d.f as the sample size grows. The complication is that, in order to obtain the p.d.f. from the c.d.f. $F(z)$, one has to effectively solve an operator equation, specifically a Fredholm equation of the first kind [105], since the p.d.f. and c.d.f. are related by

$$\int_{-\infty}^{\infty} \theta(z-t) f(t) dt = F(z) \ , \tag{2.1}$$

where the kernel θ is the Heaviside function.

Problems of this type—viz. numerical differentiation problems—are typically ill-posed [190]; thus, a small variation in F, e.g. using a sample c.d.f. as an approximation of the underlying c.d.f., can lead to a large variation in the estimated function f.[1] Thus, techniques such as regularization must be employed in order to ensure a more reliable estimate.

Operator Inversion

This problem is also described by Vladimir Vapnik as that of interpreting results of indirect experiments [190]; it is a simpler case of the density estimation problem, in that a solution of the density estimation leads naturally to a solution of this problem. It can be stated as follows:

Problem 1 Operator Inversion. Estimate the solution f of an operator equation $Af = F$, given A, and observations $(x_1, y_1), \cdots, (x_l, y_l)$, with $y_i = F(x_i)$.

If we know F and A^{-1} exists, the solution can be determined as $f = A^{-1}F$. However, this problem is ill-posed (the solution obtained can be very sensi-

[1]For more on ill-posed problems, see Section A.6.2.

tive to the observations), and regularization techniques must thus be used.

2.2.4 The Unified View: Function Estimation

All of the above problem types can be seen as special cases of the general *function estimation* problem, where one considers the problem of best estimating a function describing the sought relationship. In the case of classification, the function is from the set of possible inputs to the set of categories. For regression, the range of the function is usually $\mathbb{R}$, and in the case of density estimation the function we seek is the density function. The most common approaches to this technique involve the concept of *risk minimization*, and traditionally the principle of ERM has been used. However, the pioneering work of Vapnik, Chervonenkis, and others [190, 191] led to the alternative principle of SRM, linked to the regularization techniques of Tikhonov et al. This principle makes use of the concept of *capacity* of a class of functions, instead of trying to minimize the error on a training set at all costs.

2.3 Issues in Supervised Learning

Supervised learning techniques usually involve choosing a model for the solution, and then finding the model parameters which provide the best fit of the given data to the model. However, the concept of best fit is subject to various definitions, leading one, in the context of risk minimization, to the two distinct approaches of ERM and SRM (see Section 2.7).

The training procedures, i.e. the processes used to find the optimal model parameters, for statistical learning techniques often involve the optimization of some risk, likelihood [166, Section 3.2], entropy or information function [76, Chapter 10]. This optimization can be done by a variety of techniques, from quadratic programming (QP) to gradient descent methods to direct analytic solution.

When approaching a supervised learning problem, or a new supervised learning technique, a number of issues must be addressed, some of the most important being:

- model selection,
- parameter selection,
- complexity control,
- efficient implementation,
- handling noisy data,
- quantifying model accuracy,
- how to make use of prior knowledge,
- and rule extraction from the trained model.

These issues are discussed in more detail below.

2.3.1 Model Selection

The model selection process begins with the choice of learning technique. Whether one decides to model the data using a hidden Markov model (HMM) [42, 66, 99], a Radial Basis Function (RBF) network [76, Chapter 5], or a SVM is an integral part of the model selection process. In many cases, the process continues after this choice: in the case of SVMs and other kernel methods, the choice of kernel used is also a part of model selection; if the data is modelled as a finite mixture distribution [7], the choice of distributions used in the mixture forms part of the model selection. With artificial neural networks (ANNs) [76, Chapters 3–4], specifying the architecture of the network (number of neurons at each layer) will be part of the process, etc.

2.3.2 Parameter Selection

Most training algorithms for supervised learning techniques make use of a number of parameters which must be selected before the model can be fitted to the training data. Examples of this are the learning rate in ANNs [76, Section 3.7], and the complexity-accuracy tradeoff parameter in SVMs [76, Section 6.3].

Various techniques, many of which are experimental, have been devised for determining these parameters, but to date the most common and popular technique is cross-validation, detailed in Section 4.2 [74, 91]. Parameter selection is often considered to be a part of model selection.

2.3.3 Complexity Control

Complexity control refers to the well-known tradeoff between generalization and accuracy of the fitted model on the training set. In essence, if the model allows an arbitrary complex function to be estimated, then the training set can be learned perfectly. However, this learned function typically does not perform well on unseen data (or *generalize* well)—this phenomenon is usually referred to as *overfitting*. Conversely, a model which can only generate simpler functions may have a mediocre performance on the training set, while generalizing better than the previous case. A good reference for more information on this tradeoff—also known as the *bias/variance dilemma*—and why it is unavoidable, is the classic paper by Geman, Bienenstock and Doursat [61].

There are many ways of handling this tradeoff: e.g. in the case of SVMs, there is a parameter which specifies the relative importance to be attached to smoothness, which favours less complex functions, versus accuracy on the training set.

Another important issue to bear in mind here is that penalizing the complexity of functions less complex than the actual target function is counterproductive. In this case, overly simple estimates of the function will form which perform poorly on both the training set and the test set. This is known as *underfitting*.

Vapnik proposes the use of SRM rather than ERM to handle the tradeoff in [190], while Baum and Haussler investigate the tradeoff with respect to the size of ANNs in [13]. The number of nodes and/or layers in the network typically corresponds to the complexity of the model, as does the number of states in a deterministic finite automaton (DFA) used to model a phenomenon, or the number of states assumed for a HMM. Thus, it becomes clear that the architecture of the learning model is something that must be considered carefully if one hopes to be successful.

2.3.4 Efficient Implementation

Many algorithms developed in the early 20th century have only begun to be implemented in practical situations in the last 20 years, since their high computational requirements made earlier implementation infeasible. Many techniques in use today are still limited by their inability to perform well in real-time systems, or efficiently handle large data sets.

Clearly the efficient implementation of learning algorithms is a central issue. Work in this field includes caching more effectively, working set algorithms and probabilistic speedups. All of these techniques, and others are used for more efficient training of SVMs (see Section 3.11).

2.3.5 Handling Noisy Data

In most practical situations, measurements are subject to errors from various sources, e.g. instrumental inaccuracies, chemical impurities and human error. However, one cannot simply discard the data with erroneous information for two major reasons: first, usually the data is still typically reasonably accurate and the loss of information from discarding the data outweighs the loss incurred by using the data with its error component included responsibly; and secondly, one generally does not know which data points are subject to error.

Most learning techniques can take noisy or erroneous data into consideration by some assumption. In classification problems, the assumption is often that of a fixed probability of misclassification, while regression techniques often assume data includes an additive error component with a zero mean Gaussian distribution. However, care must be taken, since deviations from these assumptions can obviously impact negatively on the performance of the algorithm in question.

2.3.6 Model Accuracy

Model accuracy refers to the ability of the model to generalize to unseen data. Generalization is a term inherited from psychology [76], where it refers to the similar phenomenon of psychological models performing well on new subjects. As mentioned in Section 2.3.3, there is a tradeoff between generalization and accuracy on the training set—this problem is exacerbated by the presence of noisy data (see Section 2.3.5).

There are three factors affecting generalization [74, pages 198–199], viz.:

- the training set
- the model architecture
- the complexity of the problem

Generalization improves if the sample distribution of the training data starts to resemble the underlying distribution from which data is drawn more closely, which usually occurs as the training set size increases. The other two items are intertwined as mentioned in Section 2.3.3: the architecture of the model must suit the complexity of the problem; more complex problems and models require more data to train thoroughly than simpler ones.

Given that the complexity of the problem is usually fixed, and often unknown, one can investigate one of two problems, depending on the situation:

either finding a sufficient training set size given an appropriate model, or determining the best architecture given the size of the training set.

Another significant problem is that the generalization error is unknown. Fortunately it can be approximated and bounded by various techniques. By far the most common technique of estimating the generalization error currently is that of cross-validation [74, Section 7.10], which, taken to the extreme of allowing n to be the amount of data available, leads to leave-one-out bounds for both estimating and bounding the generalization error. Other bounds, based on the Vapnik-Chervonenkis dimension (VC-dimension) or fat-shattering dimension of the model used, can also be utilized. These bounds are discussed in more detail in Section 4.2.

2.3.7 Handling Prior Knowledge

In many cases, one has access to certain rules or patterns in the data. If techniques are developed to incorporate this knowledge into the model prior to training, it can decrease training times, and improve model accuracy [170, Chapters 3–4]. Using only prior knowledge and no learning is one extreme case, e.g. where the existing knowledge is encoded into a database, and decisions are made based on these rules. At the other extreme, one has learning techniques which cannot yet utilize prior knowledge at all.

Most techniques lie somewhere in between, with techniques proposed for utilizing certain forms of prior knowledge exhibiting good experimental results, although not thoroughly theoretically motivated. Examples of attempts to utilize prior knowledge in machine learning include [15, 58, 59, 78, 124, 148, 153, 170].

2.3.8 Rule Extraction

One of the main reasons for humans' traditional lack of acceptance of machine learning techniques is their inability to understand what the model is doing from its representation, e.g. a representation of a neural network as a collection of weight vectors [128, Section 3.1.6].

For this reason, techniques have been developed which extract 'rules' from some kinds of trained models, such as extracting DFAs which approximate recurrent neural networks [128] or recurrent RBFs [59]. In general, however, this field is still very open with many opportunities.

2.4 Development of Learning Theory

One can see the inductive learning problem as that of estimating a function given approximate values of the function at a number of points. As pointed out by Vapnik [190], the traditional approach to such a situation, where one is attempting to solve a wide class of problems, is to focus on the simplest case in great detail. Once the theory for the simplest case is relatively well developed, it can hopefully be extended to more general cases.

In the case of the learning problem, a simple case is that of estimating a constant function, given function values subject to noise at certain points. Study of this problem leads one naturally to the approach of ERM (see Section 2.7). However, the drawback to this approach is that it does not necessarily perform well when applied to more complex learning problems. The reason for this is that even the class of functions admitting two possible outputs is considerably more complex, i.e. has more *capacity*, than the set of constant functions. This realization has led to the development of the SRM principle [190].

The class of functions with two possible outputs are generally associated with pattern recognition (or decision theory), and this field's groundwork was laid in the statistical arena by Nyman and Pearson (1928,1933), with their work on building good hypothesis tests by making use of the probability of error. This probability of error corresponds to the risk function used in many models today, and the concepts of loss and the risk function was first formalized in 1939 by Wald [46].

If one knows the marginal distribution of the outputs, given the input, as well as the loss function, one can theoretically derive the optimal estimator of the underlying relationship. This optimal estimator is known as the *Bayes estimator* in the case of misclassification loss. Usually, however, although the loss function may be known, the necessary marginal distribution is unknown.

The first technique for pattern recognition was based on two normally distributed populations, where a linear discriminant function is used to differentiate between the two populations. This technique was formulated in 1936 by R. A. Fisher—who is also well-known for developing the concept of the *sufficient statistic* [46]—and is widely known today as Fisher's linear discriminant (FLD), or linear discriminant analysis [36]. This technique yields a Bayes classifier if the model assumptions hold, and is still popular today, mainly due to its algorithmic simplicity and accuracy on simple problems. The use of the linear discriminant function implicitly defines a separating hyperplane between the populations considered.

In order to perform FLD, the parameters of the normal distributions must be estimated from the data available [36]. The traditional approach to parameter estimation is that of *maximum likelihood* (ML)—find the parameters which maximize the probability of generating the observed data set [46]. The parameters generated with this technique are also known as *maximum a posteriori* estimates. Approaches making use of entropy and information theory led to the minimum entropy and maximum mutual information approaches to parameter estimation, and Dempster's expectation-maximization algorithm [41, 195].

Since the advent of the computer, previously infeasible approaches to building classifiers have begun to attract attention. Among these are the *nonparametric* approaches to density estimation, where the joint input-output distribution necessary for building classifiers is directly estimated from the data, instead of assuming it to be of a certain distribution type and estimating the distribution parameters. While the pioneering work and foundational theory of this field was laid before computers became readily available, large scale implementation lagged behind somewhat. Probably the best-known method for nonparametric density estimation is that of Parzen windows, a technique formalized by M. Rosenblatt in 1956, but named after E. Parzen, who used forerunners of the technique for other purposes, and derived certain useful properties of the technique. This technique was generalized to the multivariate case by Cacoullos in 1966 [46, 120].

Some of the shortcomings of the Parzen window approach to density estimation were addressed by the introduction of k_n-nearest-neighbours estimation, which is the basis of the well-known k-nearest-neighbour decision rule [46, 74]. Another popular technique was based on the concept of calculating the resultant charge on a new point if the given data represents point charges in input space—this is known as the method of potential functions. These techniques of density estimation (Parzen windows, k_n-nearest-neighbours, potential functions) all make use of the concept of estimating the true density by combining what are known as kernel functions centred at the available data points. An entire field centring on this approach has been developed, and is known as kernel discriminant analysis. More on the origins of this field can be found in [72].

Another major stumbling block which cannot be overcome simply by the brute-force computing power available today, but requires human ingenuity in addition, is Bellman's famed 'curse of dimensionality': as the dimension of the input data increases, solving the problem with the original data becomes infeasible very quickly. Besides the computational effort of performing computations in higher dimensions, the density of the data in input space decreases rapidly, resulting in very unreliable estimates. To address this issue, dimension reduction techniques were investigated and harnessed for use in the decision theory stable. Dimension reduction techniques are techniques which transform the data from input space into a lower-dimensional *feature space*, while attempting to maintain the underlying structure of the original data as far as possible. It turns out that FLD is a dimension reduction technique which reduces the dimension of the input data to one, and Bryan extended FLD to a technique known as multiple discriminant analysis in 1951 [46]. This technique can then be used to reduce the dimensionality of the input data to the desired dimension as a preprocessing step to utilizing some standard decision theory technique. Of course, this is not the only technique used for dimension reduction, and today PCA is a very popular dimension reduction technique [97]. The variety of dimension reduction techniques is due to each technique trying to capture different aspects of the structure of the input data.

Besides the use of FLD as a dimension reduction technique, Fisher's work also laid the cornerstone for the use of discriminant functions for classification. As mentioned before, Fisher's discriminant function defines a hyperplane separating the two classes. In the first half of the 20th century, many scientists in the neurological field had started to study the use of computational approaches based on simplified models of nervous system processes. There were many opinions on how best to approach this problem, and the one which led to the development of many of the ANN-style algorithms we encounter today, was an attempt to model neural networks mathematically by means of a symbolic logic model [139]. This attempt led to the McCulloch-Pitts model of the neuron in 1943, which was in turn the basis of Frank Rosenblatt's perceptrons[2] [139]—a development which Vapnik describes in [191] as the bright event characterizing the first period of research of the learning problem. McCulloch and Pitts showed that a sufficiently complex network of their 'neurons' could theoretically compute any computable function.[3] This result is considered to be the origin of ANNs and the study of artificial intelligence as a field in its own right. Their work also influenced von Neumann when he was building EDVAC, a successor of the ENIAC [76]. It turns out that simple perceptrons define a separating hyperplane between two classes, exactly like FLD [139].[4] The difference between the two techniques, though, was that FLD attempted to construct a separating hyperplane between two known probability distributions, while the perceptron attempted to construct the hyperplane on the basis of labelled examples [191]. This problem is known as inductive inference, and the perceptron *learnt* the decision rule by *training* a number of weights which specified the algebraic form of the hyperplane.

[2]Not M. Rosenblatt, who developed Parzen windows.

[3]Effectively this means that most functions one would consider can be modelled to arbitrary accuracy with a large enough network.

[4]More complex perceptrons can generate piecewise-linear decision surfaces.

In the initial work by Rosenblatt, he worked primarily with what he termed the *elementary* perceptron. This was mainly because his model of the neuron, based on the McCulloch-Pitts model, used an 'all-or-nothing' activation function[5], which made the development of training techniques for more sophisticated perceptron networks difficult. It should be borne in mind that the perceptron was originally proposed as a simplistic model of the brain, and not as a pattern recognition tool [139]. Rosenblatt's perceptron was the first of a long line of learning machines, and in particular was the ancestor of the ANN. The term learning machine refers to the existence of both a model architecture and a training algorithm which optimizes the model according to some criterion, e.g. the number of errors on the training data.

Statistical learning theory (SLT) also arose as a result of the perceptron. In 1962, Novikoff proved a theorem regarding the convergence of the perceptron training algorithm under certain conditions. This theorem was expanded upon by Aizerman, Braverman and Rozonoer in 1964 to provide a probabilistic bound on the generalization error of a perceptron on unseen data [191]. Regularization theory for solving ill-posed problems (see Section A.6.2) was also developed in the 1960's, when it was also discovered that nonparametric density estimation is, in general, ill-posed [191]. The theoretical analysts of learning theory also developed SRM [190], which was based on the ideas of algorithmic complexity and capacity of a class of functions, concepts developed in the 1960s and 1970s [191].

In the 1980's, a continuous discriminant function (specifically a sigmoid function [98]) was considered as a replacement for the 'all-or-nothing' function used by Rosenblatt in his perceptrons. This made the analysis of the network easier, and led to the discovery in 1986 of the *back-propagation* algorithm by Rumelhart, Hinton and Williams [113][6], a gradient descent technique for simultaneously optimizing all the weights in the ANN [191, 76]. This discovery sparked the beginning of the large-scale use of neural networks in practice.

[5] This means that the neuron either fired (output a 1) or didn't fire (output a -1).

[6] A forerunner of back-propagation was developed by Paul Werbos in 1974.

In the last decade, the emphasis in the field of machine learning has shifted away from blind reliance on ANNs. Increasing attention is being given to alternative learning methods, including RBF networks, SVMs, HMMs [42, 66, 99], regularization networks, self-organizing maps (SOMs), and more. Bounds on generalization error, and other theoretical motivations for the use of various architectures are being investigated more actively, and play a greater role in the choice of learning technique [191].

2.5 Risk—True and Empirical

Learning can be considered as a curve-fitting problem, i.e. generating a function which best fits the underlying relationship one is trying to estimate. In order to apply mathematical and statistical techniques to this problem, the concept of best fit needs to be specified more precisely. The most common approach is the use of a risk functional, which typically utilizes the concept of a loss function.

A loss function $L(y, y_0)$ is a function which measures the loss suffered or cost incurred by approximating y with y_0. A typical loss function for dichotomous classification problems is

$$L(y, y_0) = \begin{cases} 0 & \text{if } y = y_0 \\ 1 & \text{if } y \neq y_0 \end{cases} \quad , \tag{2.2}$$

while the root sum of squared errors $L_2(y, y_0) = \|y - y_0\|_2^2$ and sum of absolute errors $L_1(y, y_0) = \|y - y_0\|_1$ are commonly used for regression.

If one knows the loss function, a reasonable way of assessing the suitability of a candidate function f as an estimator is to consider the mean error incurred by using this estimator of the correct value y, i.e. one investigates the *true risk functional*

$$R(f) = \int L(y, f(x))dP(x, y) \quad , \tag{2.3}$$

where $P(x, y)$ is the joint probability of obtaining an input x labelled y.

The problem with this measure of suitability is that the distribution $P(x, y)$ is usually unknown. However, under the assumption that the training data $(x_1, y_1), \cdots, (x_l, y_l)$ is fairly representative of the underlying distribution [7], the *empirical risk functional*,

$$R_{\text{emp}}(f) = \frac{1}{l} \sum_{i=1}^{l} L(y_i, f(x_i)) \ ,$$

should be a fair approximation of $R(f)$. With this in mind, two schools of thought developed regarding function estimation, particularly in the field of classification.

2.6 Theoretical Analysis vs. Applied Analysis

With the example loss function (2.2), the empirical risk function $R_{\text{emp}}(f_0)$ equals the proportion of the training data misclassified using f_0 as the decision function. Hence, minimizing the number of classification errors on the training data leads to a lower empirical risk functional, and hence hopefully to a lower mean error on the entire distribution.

The word 'hopefully' in the previous paragraph is what separates theoretical analysis from applied analysis of learning processes. Applied analysis focuses on minimizing the empirical risk functional, believing that this will naturally lead to lower mean errors. On the other hand, theoretical analysts feel there may be better ways to reduce the true risk functional, and have focused their attention in that direction [191]. This work has led to the SRM approach, in contrast to the ERM approach of the applied analysts.

2.7 ERM vs. SRM

The principle of ERM is that of approximating functions purely by minimizing the empirical risk functional. However, the empirical risk functional

[7] Note that although this assumption is usually reasonable, there are cases where it is seriously violated, and care must be taken when applying machine learning in these scenarios.

is only an approximation of the true risk functional, and one cannot be sure that these two functionals attain their minimum at the same point in hypothesis space [191]. This was the subject of work by Vapnik and Chervonenkis, who introduced the concept of the VC-dimension as a measure of the capacity of a class of functions, and used it to put a bound on the true risk in terms of the empirical risk and the capacity of the hypothesis space. This led to the concept of SRM [190, 196]: slowly expanding the capacity of the hypothesis space, and minimizing the empirical risk after each expansion. Roughly speaking, the empirical risk will decrease as the capacity increases, but the true risk bound will attain a minimum at a certain hypothesis space complexity.[8]

More recently, Vapnik has also introduced the concept of vicinal risk minimization, which is a generalization of ERM taking estimates of the underlying distribution of the training inputs x_i into account.[9] The approach is analogous to that of Parzen windows and other kernel discriminant analysis techniques, and encompasses a number of existing algorithms. For more details, see [35].

2.8 Statistical Learning Theory

SLT entails the use of mathematics and statistics for deriving learning algorithms that should perform well, and the investigation of their properties. The learning problem is treated as one of function estimation, where one must select the best approximating function out of a specified class, given a training set [196].

Most of the theory also assumes that data is generated from a fixed distribution, which is usually unknown, and that the functions are mappings from an input space $\mathbb{R}^n$ to some subset of $\mathbb{R}$ [196].

The goal of learning is then the minimization of the true risk functional (2.3)

[8] Other inductive principles also make use of this concept, including minimum description length and the method of sieves [162].

[9] He originally hinted at the concept as *local risk minimization* in [191].

over all functions in the given function class. The approach used for minimizing this functional is called the *induction principle*—two major induction principles are the ERM and SRM of Section 2.7. One begins with some problem or function to be estimated, and an associated loss function; together they are known as the *domain*. One then selects an induction principle, and a class of functions from which the estimate must be selected; finally one utilizes an algorithm which implements the induction principle on the class of functions, subject to the loss function [196].

Statistical learning theory is then interested in finding robust low-risk learning processes, and studying their behaviour. Vapnik [191] considers the four major aspects of this theory to be:

- the theory of consistency of learning processes
- the theory of nonasymptotic rates of convergence of learning processes
- the theory of controlling generalization of learning processes, and
- the theory of constructing learning algorithms.

Consistency concerns itself with the necessary and sufficient conditions for learning processes to converge to that function in the class of functions which minimizes the true risk, as the sample size increases—and also the rate of this convergence.

Bounds on nonasymptotic rates of convergence can be derived in terms of some relatively abstract concepts, most notably the annealed entropy, growth and generalized growth functions. In order to be useful, these functions must be bounded by more concrete functions. An example of this is the bound on the growth function making use of the VC-dimension, derived independently by Vapnik and Chervonenkis, by Sauer, and by Shelah. Work in this field revolves around finding better bounds using the existing concepts, and finding other concepts which allow one to find other, tighter bounds.

The theory of controlling generalization concerns itself with the search for an inductive principle with good generalization ability on small training samples. It was this field of learning theory which led to the SRM principle (Section 2.7), and the idea of transductive learning (Section 4.4).

The development of new inductive principles leads to the need for algorithms which effectively implement them. A classic example of this is the development of SVMs as an algorithm to implement SRM, albeit only approximately (see Section 3.5). The development of the concept of transductive learning has led to the development of the TSVM (see Section 4.4). In any case, the central issues with the development of new algorithms discussed in Section 2.3 must be considered in this field.

2.9 Summary

Learning theory concerns itself with efficiently finding a function that describes some observed phenomenon well. Typically, this phenomenon is described by a number of observed examples, and the learning process involves using these examples to model the underlying relationship as well as possible. This is known as supervised learning, and the three major supervised learning problems are classification, regression estimation, and density estimation. These problems can all be seen as special cases of the more general problem of function estimation.

By considering this problem mathematically as one of minimizing a risk functional, the field of statistical learning theory has been developed. Two major inductive principles of this theory are ERM and SRM, and various learning algorithms have been developed to implement these principles. A number of issues must be considered when developing learning algorithms, in order to make them useful, effective and feasible.

Chapter 3

Support Vector Machines

This chapter introduces the SVM from the perspective of supervised learning and statistical learning theory. The derivation is presented, and various extensions and implementation issues are outlined.

3.1 Introduction

Since its introduction by Bernhard Boser, Isabelle Guyon and Vladimir Vapnik in 1992 [20], the SVM has been applied to a variety of problems, mostly those of supervised learning.

Section 3.2 introduces the basic concept and derivation of the SVM for dichotomous classification. After the basic algorithm is introduced, techniques for handling nonlinearity (Section 3.3)and inseparable data (Section 3.4) are considered. The connection between SVMs and SRM is summarised in Section 3.5. The application of SVMs to regression estimation is then presented in Section 3.6. Next, we consider how support vector (SV) classification and regression can be viewed as regularization problems in Section 3.7, before considering the use of SV techniques for operator inversion and density estimation in Section 3.8.

We review some algorithms closely related to the original SVM algorithm in Section 3.9, and Section 3.10 investigates techniques for extending the binary classification SVM to polychotomous problems. Some of the more common techniques for efficient implementation of SVMs are briefly discussed in Section 3.11. Finally, Section 3.12 discusses the performance of SVMs on benchmark and real-world problems.

3.2 The Basic Concept

The concept of SVMs is derived from that of an optimal separating hyperplane, a hyperplane which separates the data points for the two classes in input space without error, and has maximal least distance to any data point [20, 36, 190]. In fact, the SVM algorithm constructs exactly such a hyperplane in the simplest case, which we discuss below. Short introductions to the SVM algorithm are common in many papers utilizing the technique, some more extensive discussions can be found in [20, 24, 27, 36, 71, 76, 156, 166, 191, 193, 196]

Consider the case of dichotomous pattern recognition, given a training set $(x_1, y_1), \ldots, (x_l, y_l)$, where l is the number of training examples, and $y_i \in \{-1, 1\}$ is the class label for $x_i \in \mathbb{R}^N$.

In addition, we assume that the training data is linearly separable. This means that there is some hyperplane in input space such that all the x_i's for which $y_i = 1$ lie on the opposite side of the hyperplane to those for which $y_i = -1$.

It then follows that a $w \in \mathbb{R}^N$ and $b \in \mathbb{R}$ exist such that

$$y_i(w \cdot x_i + b) \geq 1, \quad i \in \{1, \ldots, l\} \ , \tag{3.1}$$

with equality occurring in the case of at least one x_i.

The set of hyperplanes $\{x : w \cdot x + b = 0\}$ that satisfy the above criteria (called *canonical* separating hyperplanes) is the same as the set of all separating hyperplanes, but multiple representations of the same hyperplane are normalized to a single representation of the hyperplane in canonical form [191, Section 5.4.2].

The next step is selecting the canonical separating hyperplane for which the closest x_i is furthest from the hyperplane.[1] Now, the distance of x_i from the plane $w \cdot x + b = 0$ is $\frac{|w \cdot x_i + b|}{\|w\|}$. For all the canonical separating hyperplanes, the minimum of the numerator is 1, so the problem reduces to maximizing the *margin* $\frac{1}{\|w\|}$. $\|w\|$ is also a measure of the *smoothness*[2] of the function $w \cdot x + b$ [166, Section 1.2].[3] A set of smooth functions[4] (functions with low $\|w\|$) is desirable because less data points are needed to find the optimal estimator in such a set.

Now, maximizing $\frac{1}{\|w\|}$ is equivalent to minimizing $\|w\|$ or $\frac{1}{2}\|w\|^2 = \frac{1}{2} w \cdot w$. Thus we have a constrained optimization problem:

$$\text{Minimize} \quad \frac{1}{2} w \cdot w \tag{3.2}$$

$$\text{subject to} \quad (3.1). \tag{3.3}$$

Introducing a vector of Lagrange multipliers $\alpha = (\alpha_1, \ldots, \alpha_l) \geq 0$ to handle the constraints (3.1), one obtains the Lagrangian (see Section A.3.3)

$$L_P(w, b, \alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{l} \alpha_i \{y_i(w \cdot x_i + b) - 1\} \ .$$

The solution to the problem lies at the saddle point of the Lagrangian function L_P, which must now be minimized with respect to w and b, and simultaneously maximized with respect to α (subject to $\alpha \geq 0$). Taking partial derivatives with respect to w and b, and setting them to zero yields

$$\frac{\partial L_P}{\partial w}(w, b, \alpha) = w - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \ , \tag{3.4}$$

$$\frac{\partial L_P}{\partial b}(w, b, \alpha) = \sum_{i=1}^{l} \alpha_i y_i = 0 \ . \tag{3.5}$$

Equation (3.4) shows that w is a linear combination of the x_i's, while the second equation indicates that the sum of coefficients of data points from each

[1] If one requires that $b = 0$, then the w generating such a hyperplane will be a Vapnik-Chervonenkis *generalized portrait* of the training set for $k = -1$—see [190, Addendum I.2].

[2] Also called flatness, or stability. A rather technical description of smoothness can be found in [62, Section 2].

[3] The smaller $\|w\|$ is, the less $w \cdot x + b$ changes with a fixed change in x.

[4] That is, a function class with small complexity.

class is equal. In addition, the Karush-Kuhn-Tucker (KKT) complementarity conditions (see Section A.3.2) between the α_i and their corresponding constraints can be exploited to find out more about the linear combination. The relevant KKT complementarity conditions are

$$\alpha_i\{y_i(w \cdot x_i + b) - 1\} = 0, \quad i \in \{1, \ldots, l\} \ . \tag{3.6}$$

Thus, $\alpha_i = 0$ unless $y_i(w \cdot x_i + b) - 1 = 0$; or, in other words only those data points x_i closest to the hyperplane (at a distance of $\frac{1}{\|w\|}$) can have nonzero Lagrange multipliers α_i. Usually, these points form a small proportion of the data [20], and so SVMs are said to learn a decision function with a *sparse* representation [23, 25]. These few data points used to calculate w are called the SVs, support patterns or extreme vectors, a term first used by Vapnik [190, Addendum I.2]. An effect of the learning algorithm is that removing non-SVs from the training set and retraining the SVM yields the same solution [24].

Substituting the equality constraints (3.4) and (3.5) into L, one obtains the equivalent Wolfe dual problem (see Section A.3.4) of maximizing

$$\begin{aligned}
& L_D(\alpha) \\
= \ & \frac{1}{2}(\{\sum_{i=1}^{l} \alpha_i y_i x_i\} \cdot \{\sum_{j=1}^{l} \alpha_j y_j x_j\}) - \sum_{i=1}^{l} \alpha_i \{y_i((\sum_{j=1}^{l} \alpha_j y_j x_j) \cdot x_i + b) - 1\} \\
= \ & \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j - b \sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i \\
= \ & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j
\end{aligned} \tag{3.7}$$

with respect to α, subject to $\alpha \geq 0$ and (3.5) (The constraint in (3.4) falls away since w is not present in (3.7)). Now the expression (3.7) for L_D is a quadratic form, and the constraints are all linear in the α_i's. This is then a quadratic programming problem, which can be optimized by various computer packages (see Section 3.11 for more information).

Suppose we have an α^o which maximizes L_D subject to the given constraints. Let w^o and b^o be the parameters of the corresponding optimal hyperplane. It follows from (3.4) that $w^o = \sum_{i=1}^{l} \alpha_i^o y_i x_i$. In order to calculate b^o, however, one must make use of the KKT complementarity conditions (3.6). For any nonzero α_i^o, it follows that

$$y_i(w^o \cdot x_i + b^o) = 1 \ ,$$

so that

$$b^o = y_i - w^o \cdot x_i \ , \tag{3.8}$$

since $y_i = \pm 1$. Solutions to the quadratic programming problem are usually determined by computers, so for reasons of numerical stability, b^o is usually taken to be the mean of the values calculated using (3.8) for each nonzero α_i^o [24, Section 3.2].

Once the separating hyperplane has been determined, it can be used to classify new data points by considering which side of the hyperplane they lie on. Thus the decision or classification function for a new point $x \in \mathbb{R}^N$ is

$$\mathrm{sgn}(w^o \cdot x' + b^o) \ ,$$

or equivalently

$$\mathrm{sgn}(\sum_{i=1}^{l} \alpha_i^o y_i x_i \cdot x + b^o) \ .$$

3.3 The Kernel Trick

Up to now, the common linear kernel was used in the SVM formulation. The kernel trick of Section A.7.3 was actually first popularized as a generalization of the linear SVM to high-dimensional feature spaces [20]. The motivation for this is Cover's theorem (see Section A.7.2), which essentially states that in many cases where data is not linearly separable in a low-dimensional space, a transformation of the data to a higher-dimensional feature space can yield linear separability.

In the SVM formulation of Section 3.2, a linear decision surface or hyperplane is generated by solving the quadratic programming problem generated by trying to maximize the margin, i.e. the distance between the hyperplane and the closest data point, or from a different viewpoint, enforce smoothness. Substituting an appropriate kernel function K for the inner product results in a linear decision surface in a new feature space H_K, a Hilbert space, which corresponds to a nonlinear decision surface in the original input space.

Thus, in order to generate such a nonlinear decision surface, one must maximize

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

subject to $\alpha \geq 0$ and $\sum_{i=1}^{l} \alpha_i y_i = 0$.

In the previous section, we assumed the inputs x_i came from $\mathbb{R}^n$. However, if we used any other Hilbert space as input space, the same reasoning works, and we obtain the same results, with one exception. If the underlying mapping from input space[5] to feature space associated with K is $\varphi : \mathbb{R}^N \to H_K$, all x_i in the final results are replaced by $\varphi(x_i)$. An important effect of using the kernel trick is then that $w^o = \sum_{i=1}^{l} \alpha_i^o y_i \varphi(x_i)$. Now, in many cases, the feature space H_K associated with K is infinite-dimensional, and in most other cases, φ is unknown or difficult to determine, which makes an explicit calculation of w^o infeasible. This is fortunately not as much of a stumbling block as it may at first seem, since it is the decision function $\mathrm{sgn}(w^o \cdot \varphi(x) + b^o)$ for $x \in \mathbb{R}^N$ that is of interest. Substituting the expression for w^o into this decision function yields

$$\begin{aligned} & \mathrm{sgn}((\sum_{i=1}^{l} \alpha_i^o y_i \varphi(x_i)) \cdot \varphi(x) + b^o) \\ = \; & \mathrm{sgn}(\sum_{i=1}^{l} \alpha_i^o y_i \varphi(x_i) \cdot \varphi(x) + b^o) \\ = \; & \mathrm{sgn}(\sum_{i=1}^{l} \alpha_i^o y_i K(x_i, x) + b^o) \;\; , \end{aligned}$$

where the b^o used is typically the mean (over nonzero α_i^o) of the values obtained from

$$b^o = y_i - w^o \cdot \varphi(x_i)$$

[5]Note that the kernel trick can be applied with any set as input space, not only N-dimensional Euclidean space, as long as a suitable kernel is defined.

$$= y_i - \left(\sum_{j=1}^{l} \alpha_j^o y_j \varphi(x_j) \right) \cdot \varphi(x_i)$$

$$= y_i - \sum_{j=1}^{l} \alpha_j^o y_j K(x_j, x_i) \ .$$

The choice of the kernel function K is often difficult, and the precise role of the kernel function is a thorny issue in theoretical analysis. Connections between the kernel function and regularization theory have helped in this regard [168], and in particular have allowed the design of kernel functions incorporating prior knowledge [164].[6] Various kernel functions are discussed in Section 4.3.

3.4 Handling Inseparability

A serious limitation of the original SVM proposed in [20] was its inability to handle training data which was not linearly separable in the feature space corresponding to the chosen kernel K. This problem was addressed by Corinna Cortes and Vladimir Vapnik in 1995 [36] with their proposal of the *soft margin* hyperplane. This has since become the standard approach used in SVMs[7], and an outline of the technique is given here.

The derivation done here is once again for a linear decision surface, the extension by using the kernel trick is identical to the linearly separable case.

The approach of the soft margin SVM is to allow the constraints (3.1) to be violated, but in such a way that any such violation will increase the objective function. Thus we change our interpretation of the margin $\frac{1}{||w||}$: now, the margin is the distance which must be attained between the separating hyperplane and a point in order to avoid penalization. We proceed by introducing *slack variables* $\xi = (\xi_1, \ldots, \xi_l)$. The constraints are modified

[6] The connection between SVMs and regularization is discussed in Section 3.7.

[7] Although the kernel trick aims to make the data linearly separable in feature space, we still need to cater for cases where separability is not achieved.

to

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad i \in \{1, \dots, l\} \tag{3.9}$$

for $\xi \geq 0$, and the objective function is changed to

$$\frac{1}{2} w \cdot w + C\Gamma(\sum_{i=1}^{l} \xi_i) \ , \tag{3.10}$$

where $C \in \mathbb{R}$ is a parameter determining the extent to which violations of the original constraints (3.1) should be penalized, and Γ is a convex, monotonically increasing function on $\mathbb{R}^+$. The reason for these requirements on Γ is that it ensures that the objective function remains convex. In particular, for $\Gamma(x) = x^\beta$, with β any positive integer, the problem is a convex program, and for $\beta = 1$ or $\beta = 2$, a quadratic program as before [24]. In these cases, the problem can once again be solved with available quadratic programming packages (see Section 3.11). Henceforth, we use $\Gamma(x) = x$.[8]

The derivation of the quadratic programming problem is similar to that in Section 3.2. First, Lagrange multipliers are introduced for the constraints. $\alpha \geq 0$ is once again used for the constraints (3.9), and the multipliers $\eta = (\eta_1, \cdots \eta_l) \geq 0$ to enforce the constraint $\xi \geq 0$.

So our primal Lagrangian is

$$\begin{aligned} L_P(w, b, \xi, \alpha, \eta) &= \frac{1}{2} w \cdot w + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i \{ y_i (w \cdot x_i + b) - 1 + \xi_i \} \\ &\quad - \sum_{i=1}^{l} \eta_i \xi_i \ . \end{aligned} \tag{3.11}$$

Taking partial derivatives with respect to w, b and ξ_i yields

$$\begin{aligned} \frac{\partial L_P(w, b, \xi, \alpha, \eta)}{\partial w} &= w - \sum_{i=1}^{l} \alpha_i y_i x_i \ , \\ \frac{\partial L_P(w, b, \xi, \alpha, \eta)}{\partial b} &= -\sum_{i=1}^{l} \alpha_i y_i \ , \\ \frac{\partial L_P(w, b, \xi, \alpha, \eta)}{\partial \xi_i} &= C - \alpha_i - \eta_i \ . \end{aligned}$$

[8]This will cause the slack variables to drop away in the dual formulation [24].

By setting these derivatives to zero, we obtain the two familiar constraints

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{3.12}$$

and

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \ , \tag{3.13}$$

and the new constraint

$$C = \alpha_i + \eta_i \ . \tag{3.14}$$

Substituting (3.12) into L_P from (3.11), and using (3.13) and (3.14) to obtain the Wolfe dual problem proceeds as follows:

$$\begin{aligned}
L_D(\alpha) &= \frac{1}{2}(\{\sum_{i=1}^{l} \alpha_i y_i x_i\} \cdot \{\sum_{j=1}^{l} \alpha_j y_j x_j\}) + C\sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \eta_i \xi_i \\
&\quad - \sum_{i=1}^{l} \alpha_i \{y_i((\sum_{j=1}^{l} \alpha_j y_j x_j) \cdot x_i + b) - 1 + \xi_i\} \\
&= \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j + \sum_{i=1}^{l} C\xi_i - \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\
&\quad -b\sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i - \sum_{i=1}^{l} \alpha_i \xi_i - \sum_{i=1}^{l} \eta_i \xi_i \\
&= \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j + \sum_{i=1}^{l} (C - \alpha_i - \eta_i)\xi_i \\
&= \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \ .
\end{aligned}$$

So the modification to the objective function has no effect on the dual objective function. However, there is a change in the constraints. For the separable case, we required $\alpha \geq 0$ and $\sum_{i=1}^{l} \alpha_i y_i = 0$. In the current problem, we also require that $\alpha_i \leq C$. This follows from (3.14), together with $\alpha \geq 0$ and $\eta \geq 0$. Thus the modification for linear inseparability has the effect of limiting the search space for α.

Once an optimal α^o has been found for the problem above, the corresponding w^o and b^o can be found precisely as in Section 3.2. Note that the restriction

$\alpha_i \leq C$ limits the influence any one point can have on the determination of the soft-margin separating hyperplane.

A tradeoff exists between increasing the margin ($\frac{1}{\|w\|}$) and decreasing the violation of the original constraints ($\Gamma(\sum_{i=1}^{l} \xi_i)$). This tradeoff is controlled by the regularization parameter C. How best to choose this parameter is a contentious issue, and a number of techniques exist for this purpose (see Section 4.3 for more discussion on this topic).

3.5 SVMs and SRM

The SVM algorithm is an approximate implementation of SRM. For the separable case, this is done by keeping the training error fixed at zero (eliminating the bias), and then minimizing the norm $\|w\|$ (minimizing the variance). The reason for this is that the generalization error can be bounded by the sum of two terms, one being the training error, and the other being a function of $\|w\|$—viz. the capacity of the set of hyperplanes with norm less than $\|w\|$ [191].

In the inseparable case, one is concerned with optimizing the objective function which depends on the soft bounds (which correspond to the bias) as well as $\|w\|$. This corresponds to finding a w^* optimizing the soft bounds (or training error) for all possible $\|w\|$, and then selecting a w^o from these solutions optimizing the generalization error bound.

However, there is one violation of the SRM principle in the implementation of SVMs. According to the SRM principle, the form of each function class (in SVMs, all hyperplanes with $\|w\| \leq k$) must be specified before the training data is known. SVMs, however, only consider hyperplanes which can be written in terms of the training data, so that the function class implemented by a SVM is dependent on the training set.[9] This problem can be circumvented when the input space is finite, and the technique for doing this is the basis of the TSVM described in Section 4.4 [191, Section 5.11].

[9] The DD-SRM inductive principle, discussed briefly in the next chapter, solves this problem.

3.6 SV Regression and Time Series Prediction

The SRM principle concerns itself with minimizing the sum of a term reflecting the loss suffered due to errors on the training set, and a term reflecting the capacity of the function class from which an estimator is selected.

In the separable case, it is possible to obtain a zero-loss performance on the training set. Once the data becomes inseparable, however, the loss suffered by an incorrect classification must be specified. Referring to the constraints (3.9), and the objective function (3.10), we see that for each data point in the training set not achieving the margin, the objective function is increased. This increase in the objective function corresponds to the loss function employed in training the SVM.

By modifying this loss function concept, one can create SVMs for regression estimation [164, 166, 193]. Here we view the hyperplane in feature space as the estimator of the function, and penalize deviations of data points from the hyperplane. Once again, $\|w\|$ corresponds to the capacity of the function class, so that the objective function will be

$$\frac{1}{2} w \cdot w + C \sum_{i=1}^{l} L(y_i, w \cdot x_i + b) \ .$$

The standard loss function employed for SV regression is the ε-insensitive loss function

$$L_\varepsilon(y_i, y) = \max(0, |y_i - y| - \varepsilon) \ .$$

This penalizes only training points which differ from the estimator by more than ε, and then in a linear fashion. The quadratic program and its constraints can be seen by considering the problem using soft constraints (the linear kernel is assumed, the kernel trick can be applied if desired):

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2} w \cdot w + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & |y_i - w \cdot x_i - b| \le \varepsilon + \xi_i, \ \ i \in \{1, \cdots, l\}, && (3.15) \\ \text{and} \quad & \xi \ge 0 \ . && (3.16) \end{aligned}$$

However, before continuing with Lagrange multipliers, taking derivatives and back-substituting as before, we must rewrite the constraints to eliminate the absolute value, in order to make the derivatives we need continuous. We rewrite (3.15) as

$$y_i - w \cdot x_i - b \leq \varepsilon + \xi_i$$

and

$$w \cdot x_i + b - y_i \leq \varepsilon + \xi_i^* \ ,$$

with nonnegative ξ_i and ξ_i^*. This time one uses Lagrange multipliers α_i and α_i^* for the loss function constraints and η_i and η_i^* for the nonnegativity constraints. This yields the primal Lagrangian

$$\begin{aligned} L_P(w, b, \xi, \xi^*, \alpha, \alpha^*, \eta, \eta^*) &= \frac{1}{2} w \cdot w + C \sum_{i=1}^{l} (\xi_i + \xi_i^*) - \sum_{i=1}^{l} (\eta_i \xi_i + \eta_i^* \xi_i^*) \\ &\quad - \sum_{i=1}^{l} \alpha_i (\varepsilon + \xi_i - y_i + w \cdot x_i + b) \\ &\quad - \sum_{i=1}^{l} \alpha_i^* (\varepsilon + \xi_i^* + y_i - w \cdot x_i - b) \ . \end{aligned}$$

An important thing to bear in mind is that at least one of a variable-starred variable pair (e.g. ξ_i^* and ξ_i) is always zero. One proceeds by taking the derivatives of L_P with respect to b, w, ξ_i and ξ_i^*, and setting them equal to zero. This yields

$$\begin{aligned} \frac{\partial L_P}{\partial b} = 0 \ &\Rightarrow \ \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) = 0 \ , \\ \frac{\partial L_P}{\partial w} = 0 \ &\Rightarrow \ w = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) x_i \ , \\ \frac{\partial L_P}{\partial \xi_i} = 0 \ &\Rightarrow \ C = \alpha_i + \eta_i \ , \\ \frac{\partial L_P}{\partial \xi_i^*} = 0 \ &\Rightarrow \ C = \alpha_i^* + \eta_i^* \ . \end{aligned}$$

Substituting this information into L_P, one can eliminate the η and η^*, and obtains the dual formulation of maximizing

$$L_D(\alpha_i, \alpha_i^*) = -\frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) x_i \cdot x_j - \varepsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i (\alpha_i - \alpha_i^*)$$

subject to

$$\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \ ,$$

with $\alpha_i, \alpha_i^* \in [0, C]$. Again, w^o can be expanded as a linear combination of the x_i's, and b^o can be computed from the data points with Lagrange multipliers in the open interval $(0, C)$.

This SV regression technique can be used for time series prediction [54, 125, 127] by creating data points in accordance with Takens' embedding theorem [125]. The training set $(x_1, y_1), \ldots, (x_l, y_l)$ is derived from a time series $p_1, \ldots, p_{l+N}$, by $x_i = (p_i, p_{i+1}, \ldots, p_{i+N-1})$, a sequence of N time series points, and $y_i = p_{i+N}$, the next time series point. To predict a new point in the time series, one uses the vector containing the previous N time series points as input [125].

3.7 SVM as a Regularization Problem

Regularization is discussed in Section A.6.2. In this section, we attempt to employ regularization techniques to the problem of minimizing the risk incurred by estimating the true function by a function chosen from an hypothesis space $\mathcal{F} = \{f(x, \zeta) : \zeta \in \Lambda\}$. Assume we use a regularizer $Q(\zeta)$, and our loss function is L. Then our problem becomes that of minimizing

$$\sum_{i=1}^{l} L(y_i, f(x_i, \zeta)) + \lambda Q(\zeta) \ .$$

In the particular case of SVMs, the feature space H_K associated with the kernel K is a Hilbert space, so that we can use the regularizer $Q(\alpha) = \|f(\cdot, \alpha)\|^2 = \|w\|^2$. The loss at the training points is just the value of the slack variables at these points, so that the regularization problem is minimizing

$$\sum_{i=1}^{l} \xi_i + \lambda \|w\|^2 \ ,$$

which is clearly equivalent to the objective function (3.10).

By modifying the loss function, SV regression can also clearly be cast as a regularization problem. Another good reference on SVMs and regularization is [53].

3.8 SV Operator Inversion and Density Estimation

In this section, we discuss an extension of SV regression for solving linear operator equations as proposed by Vapnik et al. [193].

Consider Problem 1. Suppose we seek a solution of the form $f_w(t) = w \cdot \phi(t) = \sum_{i=1}^{\infty} w_i \phi_i(t)$ for some w, i.e. we want f to be linear in some sequence of functions ϕ.

The problem then reduces to finding an appropriate w. This can be reduced to a SV regression problem once we utilize

$$F_w(x) = Af_w(t) = A\sum_{i=1}^{\infty} w_i \phi_i(t) = \sum_{i=1}^{\infty} w_i A\phi_i(t) \ .$$

Defining $\psi_i(x) = A\phi_i(t)$, this equals

$$\sum_{i=1}^{\infty} w_i \psi_i(x) = w \cdot \psi(x) \ .$$

Thus, the original problem of approximating $f(t)$ by $f_w(t)$ can be solved by finding the w yielding the best estimate of $F(x)$ by $w \cdot \psi(x)$ given data $(x_1, y_1), \ldots, (x_n, y_n)$. This is a regression problem and can thus be solved by SV techniques. However, in order to employ SV techniques, we need $K(x_i, x_j) = \psi(x_i) \cdot \psi(x_j)$ to be an admissible SV kernel (i.e. $\psi(x)$ must represent the mapping from input space to feature space for the regression problem).

Now SV regression will yield a weight vector as an expansion on support vectors $w = \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) \psi(x_i)$. Thus the estimated solution of the operator equation is given by

$$f_w(t) \quad = \quad \{\sum_{i=1}^{n} (\alpha_i^* - \alpha_i) \psi(x_i)\} \cdot \phi(t)$$

$$= \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)\mathcal{K}(x_i, t) \ ,$$

where $\mathcal{K}$ is called a *cross kernel.*

The selection of ϕ is thus a matter for care in this approach. It must be selected so that a good estimate can be found in the span of the component functions, while generating a suitable kernel and cross-kernel function. Since the forms of these functions are dependent on A, a standard kernel cannot generally be used as is, although the use of spline functions has led to analytic kernel and cross-kernel functions in [193].

Weston proposes a modification of this technique for SV density estimation in [196]. In this case, one uses the sample c.d.f. to generate training data. Given a sample $s = (x_1, \ldots, x_l)$ from an unknown density, one constructs the sample c.d.f.

$$F_s(x) = \frac{1}{l}\sum_{i=1}^{l}\theta(x - x_i) \ .$$

Then the problem is that of solving the operator equation (2.1) given training data

$$(x_1, F_s(x_1)), \ldots, (x_l, F_s(x_l) \ .$$

Weston notes that a problem with this technique is that the estimate it yields for the probability density does not satisfy the requirements of a probability density (everywhere nonnegative, and integrates to one). He incorporates these constraints by developing a technique analogous to classical density estimation kernel techniques (such as Parzen windows) using a regularization argument—for more details consult [196].

3.9 SVM Variants

The SVM, as an implementation of the SRM principle, was an entirely new approach to machine learning. Further research has led to a number of variations on the standard technique, and also variations of other algorithms based on the kernel trick. A few of the more well-known SVM variants are discussed here.

Probably the most influential SVM variant is the ν-SVM [156]. Its most attractive feature is the addition of the parameter ν to the regular SVM, which provides a simultaneous upper bound on the fraction of errors and lower bound on the fraction of the data which are support vectors. Moreover, the fractions converge to ν with probability 1. With this technique, the value of ε need not be prespecified, but comes from the choice of ν. A guideline for the choice of ν is given in [166, Section 6.4]. Importantly, this formulation corresponds to the normal SVM formulation for a corresponding choice of parameters, i.e. this variant does not affect the final result, but is an alternative way to select the parameters. This variant is available in most SVM packages. An extension of this proposal is that of varying the width of the lossless region around a regression estimate for heteroscedastic problems [166].

Another popular SVM variant, the linear programming SVM, is outlined in [25]. This approach attempts to enforce smoothness by minimizing the ℓ_1-norm of α, instead of the squared ℓ_2-norm of w. One of the effects of this change is that a linear programming problem results instead of a quadratic program. Many techniques already exist for solving such problems quickly and efficiently.

For classification, the minimization done by the SVM algorithm is equivalent to minimizing a probabilistic upper bound on the generalization error of possible hyperplane decision surfaces in feature space. An extension along these lines for classification is Jason Weston's leave-one-out (loo) SVM (LOO-SVM). This makes use of an alternative upper bound on the generalization error due to Jaakkola and Haussler based on the loo cross-validation error [87, 196, 197]. One of the advantages of this technique is that no SVM parameters have to be selected. Adaptive margin SVMs (AM-SVMs) have also been formulated as a generalization of LOO-SVMs, where a controlling regularization parameter is introduced [196]. Unfortunately, this technique cannot be extended to regression, since deriving an equivalent to Jaakkola and Haussler's bound in this case leads to a worse bound than Vapnik's original bound in [191]. For more on generalization bounds, see Section 4.2.

Weston also proposes multi-kernel SVMs for classification, regression and density estimation [196]. This technique makes use of what he terms a *kernel dictionary*, a set of kernel functions for the learning machine to utilize as necessary.

By trying to directly reduce the ℓ_0-norm[10] of the weight vector w of the decision surface in feature space, one can form alternative approaches to SVMs. This problem can be approximated by constrained optimization problems where the objective function is the ℓ_1- or ℓ_2-norm of w. This approach leads to the ℓ_1- and ℓ_2- **a**pproximation of the ze**ro**-norm **m**inimization SVM detailed in [198]. This then corresponds to a feature selection approach, since the l_0-norm of w is precisely the number of features used to construct the decision surface. For good generalization, one should try to reduce this number of components to an acceptable level, but not necessarily try to minimize it at all costs [198]. Other feature selection variants of SVMs outlined in [198] include approaches based on recursive feature elimination, correlation coefficients and R^2W^2.[11]

Saunders et al. propose the extension of ridge regression by means of the kernel trick in [140]. This extension is a variant of the SVM for regression where the ε-insensitive loss function is used with $\varepsilon = 0$, the vector ξ is derived using equality constraints, and Γ is replaced by the ℓ_2-norm of ξ. A result of this formulation is that a linear and not quadratic programming problem is obtained, which can be solved easily. An extension to classification, the least squares SVM (LS-SVM), has been proposed in [177, 176]. Unfortunately, in these formulations, the sparsity of the original SVM is lost—all the data points become support vectors. Pruning the support vectors with small coefficients is proposed to overcome this problem in [176, 174]. Other proposed extensions to these techniques include the weighted LS-SVM for more robust regression [173], an extension to polychotomies [178], and recurrent LS-

[10] The ℓ_0-norm of a vector is its number of nonzero components.

[11] R^2W^2 is discussed in Section 4.2.

SVMs for modelling multi-step time series prediction [175].

By viewing the SVM regularization problem, and modifying the regularization term, one can obtain a number of these variants (e.g. by replacing the regularizer $w \cdot w$ with $\sum_{i=1}^{l} |\alpha_i|$, one obtains the LP-SVM from the regular SVM [196]. Using yet another regularizer leads to AM-SVMs. An effect of this is that the asymptotic behaviour of these variants is identical (since the effect of the regularization term diminishes as the size of the training set increases).

Other learning machines have been derived from the SVM by considering *version space*, the space of all possible separating hyperplanes consistent with the data (this is a subset of the dual of input space). SVMs select the decision surface in the centre of the largest hypersphere which can be inscribed in version space [25]. However, alternative approaches based on this representation can be considered. The hyperplane corresponding to the Bayes point (approximately the centre of mass of version space) is the basis of the Bayes point machine (BPM) [79, 80, 81]. This approach has performed well, with statistically significant better performances over regular SVMs in some cases. Unfortunately, determining the Bayes point is difficult, and even determining the centre of mass of version space is computationally intensive, utilizing a *billiard algorithm*, where a simulated billiard ball travels uniformly through version space. An alternative approach which is less computationally intensive is the Analytic Centre Machine (ACM) [25]. A disadvantage of the BPM and ACM is that the sparse representation of the decision function is lost. On the other hand, an approach related to SVMs which has exhibited a much higher degree of sparseness experimentally, is Michael Tipping's relevance vector machine [183].

Olvi Mangasarian and his associates have also proposed a number of SVM variants, including active SVMs, proximal SVMs and smooth SVMs [60, 64, 109].

An important variant of the SVM for our purposes, the TSVM, will be discussed in more detail in Section 4.4.

3.10 Multi-class Support Vector Machines

One shortcoming of the SVM classifier we have introduced is that it is restricted to dichotomous classification. This issue has received much attention in the last six years, with two major avenues followed.

First, there are a number of techniques which can generally be utilized for extending dichotomous pattern recognition tools to the polychotomous case. Secondly, there have been attempts to modify the underlying architecture of the SVM.

A number of the techniques below have been compared for efficiency and accuracy in [84]. We have verified the accuracy results of that paper, with the few differences in results being attributed mostly to random sample fluctuations when performing cross-validation, and different storage precision in the cache [112]. Although not specifically investigated, the running times of the experiments also seemed to correlate well.

3.10.1 Standard Multi-class Techniques

One-against-the-rest

Since the initial optical character recognition (OCR) application of SVMs (see Section 3.12), the *one-per-class* or *one-versus-rest* (1-v-r) technique has been used [20, 36].

This technique entails training an SVM for each class which distinguishes between that class and other classes. A new example is then classified by testing it on each of these SVMs: typically, the new data point is then taken to belong to the class whose SVM outputs the highest value, prior to taking the sign [194, 196].

There are two major problems with the 1-v-r approach. First, there is no guarantee that all of the SVMs' output range is identical or even similar. Depending on the difficulty of each 1-v-r problem, the output range of the different SVMs will differ. Proposals to handle this, including normalization of the SVM outputs and feeding the SVM outputs into a single layer perceptron, are discussed in [121], with the perceptron significantly enhancing performance. The second complaint levelled at the 1-v-r technique is that it does not take into account the mutual exclusivity of class membership. The result of this is a great loss in discriminative power when there is no class i for which $P\left(\text{class}(x) = i\right) > 0.5$. More detail on this issue is given in [109].

Despite these shortcomings, this technique is easy to implement, displays fairly good performance empirically, and is still widely used.

One-against-one

The *pairwise coupling* or *one-versus-one* (1-v-1) technique involves constructing a SVM for each pair of classes, which distinguishes between them. Each SVM tests a new data point, and votes for the class it places the data point in. The point is then placed in the class receiving the most votes. Ties can be broken by considering the SVM outputs prior to taking the sign [194, 196].

This is the technique used in the popular SVM implementation package LIBSVM [29]. One of the reasons given for this is that although the number of SVMs that need to be trained grow quadratically with the number of classes (opposed to the linear growth of the 1-v-r approach), the optimization problems encountered are much smaller, since only data from the 2 classes involved are used for training each SVM. A drawback of this, however, is that smaller samples result in higher variance, so that the generalization performance of the individual SVMs may suffer [109].

Hastie and Tibshirani [73] propose some modifications to improve the efficiency of pairwise coupling techniques in general, which mostly rely on probabilistic outputs of the classifier. However, they also briefly present a technique for approximating class probabilities from outputs of dichotomous SVMs. Another method for extracting class probabilities for 2-class SVMs is detailed in [133].

Directed Acyclic Graphs

John Platt proposed the directed acyclic graph (DAG) SVM in order to compensate for two shortcomings of the 1-v-1 approach: the lack of a probabilistic generalization bound for 1-v-1 SVM approaches, and the quadratic (in the number of classes k) time complexity for evaluating new points.

Platt's technique makes use of decision DAGs, and consists of once again training one classifier for each pair of classifiers. However, to evaluate the class membership of a new point, the point is presented serially to only $N-1$ of the classifiers. The losing class in each classifier is eliminated as a possible class for the test point, so that ultimately the point is assigned to the remaining class. The decision as to which SVMs to present the data point to is taken by following a decision tree, specifically a DAG (hence the term decision DAG). Details on the construction of this tree and the technique are in [135].

The evaluation time of this technique is linear in the number of classes, and probabilistic generalization bounds are also derived for this approach.

Output Codes

Error-correcting code theory involves the design of codes which are resistant to deformation. This is useful when data is being transmitted along a carrier which may deform the data, or stored on magnetic media. Error detection techniques such as parity bit checking may be able to identify such errors, but are unable to correct them. Thus, more robust techniques for data description have been developed, such as Hamming codes [8, Chapter IV].

Using these codes to represent one's data allows one to recover the original data even if limited damage is done to the data. By viewing this damage as noise, and considering the output of a number of binary SVMs the noisy output of the code for the class membership of the input, one can use similar techniques to extract the 'original' class membership.

In this case, the binary SVMs are usually trained to distinguish between groups of classes, and the output of all the SVMs for a test example together form a binary string. This string is compared to a number of *codewords*, strings representing each of the classes. These codewords can be constructed in a fashion so that they are robust to errors on a limited number of the SVMs. The class whose codeword most closely matches the output string of the SVMs is then taken as the class of the test example. The measure of distance used here is typically the Hamming distance[12]

More complete details on the operation of these error-correcting output codes can be found in [44, 45]. Proposals for improving on this technique by designing better codes for multi-class problems can also be found in these papers. Other such work includes that on continuous codes by Crammer and Singer [37], and the design of codes which are dependent on the training data rather than defined a priori [122]. In [38], Crammer and Singer also point out a shortcoming of output code techniques: the independence of the binary classification problems implies that correlations between classes cannot be taken into account by such a technique.

Other Techniques

By viewing each SVM classifier as a competition between the classes it is trying to distinguish, one can view the above four strategies as tournaments with the winner being the chosen class. Other winner selection strategies from game theory can also be utilized. Many tournaments select a winner without a full round-robin pairing taking place, but by only playing selected opponents, e.g. the classic knock-out approach. Binary SVMs can be organized into tournaments in the same way to determine a winner [196].

An important aspect to consider when designing multi-class classifiers in this way is that while most games have some random element, classifiers are usually deterministic, so that tournament structures which permit rematches may not be so desirable. Interestingly, the choice of tournament structure

[12] The number of differing bits between two codewords [8, Chapter IV.1].

is still a major issue in real tournaments today, as the optimal structure is highly debatable.

3.10.2 Extending the Standard SVM

Work on extending the SVM to the polychotomous case directly usually proceeds by the identification of certain desirable properties of the binary SVM, and an attempt to generalize the formulation so that as many of these properties as possible are maintained in the multi-class case.

To date, these desirable properties, and hence the resulting formulations, have differed widely. Platt et al. assert in [135] that the problem of construction of multi-class SVMs is currently unsolved.

MC-SVM

The first direct extension of SVMs directly to the polychotomous case, the multi-class SVM (MC-SVM), was proposed in 1998 by Weston and Watkins, and was also independently derived by Vapnik and Blanz [68, 200, 196]. This technique can also be used for generalizing LP-SVMs to the multi-class LP-SVM.

The approach taken here is that of generalizing the objective function to be minimized. Weight vectors $w_1, \cdots, w_k$, biases $b_1, \cdots, b_k$ and slack vectors $\xi_1, \cdots, \xi_k$ are introduced and (3.10) becomes

$$\frac{1}{2}\sum_{i=1}^{k} w_i \cdot w_i + C\sum_{i=1}^{k}\sum_{j=1}^{l}\xi_{ij} \quad ,$$

with slightly modified constraints due to a generalization of the loss function being used.

After optimizing the weights and biases, the decision function reduces to

$$f(x) = \arg\max_i w_i \cdot x + b_i \quad .$$

This effectively corresponds to assigning each class a score for a new test example, and the point is assigned to the class with the highest score. The binary SVM is a special case, where the scores are effectively subtracted from each other. One could argue that 1-v-r SVMs also employ this technique; however, the difference between the techniques is that the weights used to generate the scores are optimized simultaneously in this technique. Weston also discusses a single optimization problem formulation of the 1-v-r technique in [196]. In [26, p161], Campbell states that the MC-SVMs exhibit similar performance to the 1-v-r technique, although [196] alleges that due to the lower number of SVs of their approach, the expected generalization error should be lower.[13]

Guermeur et al.'s SRM Approach

Guermeur et al. [68] make use of a multi-class generalization of the margin concept, to derive a probabilistic generalization bound. This bound has two terms, corresponding to the empirical risk and capacity terms of the bound used as the basis of SRM.

In a manner similar to the derivation of linearly separable SVMs, an optimization problem is formulated which finds the optimal tradeoff between empirical risk and capacity. The resulting technique can also be extended to utilize kernels, and handle nonseparable data.

Crammer and Singer's Approach

The margin concept is also central to Crammer and Singer's multi-class SVM. They extend the concept of separating hyperplanes to multi-class problems, and then attempt to maximize the generalized margin in the same way SVMs do in the binary case.

The weight vector w is replaced by a matrix M, with the columns of M essentially being class-specific weight vectors. An optimization problem essentially identical to that for SVMs results, except that the squared norm of the weight vector is replaced by the squared Frobenius norm of M, and the

[13]Note that this suggestion is entirely based on using the number of SVs as an upper bound on the loo error.

constraints are modified to take into account the generalized margin with slack variables.

Details on the derivation of the decision function can be found in [38], together with proposals for efficient implementation of the algorithm, including decomposition and caching (see Section 3.11).

Lee et al.'s Sum-to-zero Approach

Lee et al. approached the generalization of the SVM by considering its asymptotic behaviour [109]. Previous work by some of the authors showed that the binary SVM decision rule for an input x asymptotically tends to $\mathrm{sgn}[P(x \in \mathrm{class1}) - \frac{1}{2}]$, i.e. SVMs implement a Bayes classifier, the optimal classification rule, in the limit.

Their approach is then to generalize the SVM by modifying the desired and actual output to a vector whose components sum to zero (i.e. class labels are represented by vectors whose components sum to zero). This results in an optimization problem very similar to those already encountered above, but with a sum-to-zero constraint on the output vector (which has one component for each class). The decision function then selects the class corresponding to the greatest component of the output vector. This approach maintains the property of asymptotic application of the Bayes rule.

Techniques for handling unequal misclassification costs and sample bias which maintain the optimal asymptotic behaviour are also presented.

M-SVM and MC-LS-SVM

Erin Bredensteiner and Kristin Bennett propose the M-SVM formulation by applying to SVMs the techniques used to generalize the Robust Linear Programming approach of Mangasarian et al. to Multicategory Discrimination [22]. In addition, the new formulation still follows a margin maximization approach and can make use of the kernel trick to generate nonlinear decision functions.

Finally, in [178], a technique for generalizing LS-SVMs (see Section 3.9) is proposed. However, on closer inspection, the technique suggested turns out to be that presented by Weston as the single step 1-v-r technique [196, Section 3.5], modified for the LS-SVM case.

3.10.3 Multi-labelled Classification

Another problem commonly encountered is that of *multi-labelled* classification. Here, an example must be assigned to the subset of categories to which it belongs. A common example is text categorization, e.g. the classification of news articles, where one article can be relevant in any number of fields. To compound the problem, one also does not usually know how many categories a new example belongs to.

For this type of problem, a simple variation of the 1-v-r approach is commonly used. For each category, a SVM is trained which discriminates between articles in and not in that category. New examples are then presented to these SVMs, and placed in those categories for which the SVM accepts the example. An enhancement of this technique is the use of thresholds: only if the SVM output for a category exceeds a certain learnt threshold, is that category selected [77, 89].

Elisseeff and Weston [49] describe a more general approach which they call the ranking approach, which attempts to rank the various categories by minimizing an expression called the *ranking loss*. This technique leads to a generalization of MC-SVM for multi-labels, which the authors term Rank-SVM. Finally, a learning problem is solved in order to predict the number of categories to assign a new example to, and the appropriate number of highest ranked categories are selected. The advantage of this approach over the 1-v-r variation is its ability to cope with correlations between categories.

3.11 Implementing SVMs

A variety of SVM implementations and utilities are available.[14] Well-known packages include Chang and Lin's LIBSVM [29] and BSVM [28], SVM^{light} by Thorsten Joachims [90], Ryan Rifkin's SVMFu [137], the Royal Holloway University of London (RHUL) SVM package [141] and SVMTorch, the SVM component of the Torch machine learning library from the Dalle Molle Institute for Perceptual Artificial Intelligence.[15] The SVM Matlab toolbox by Steve Gunn[16] is also popular.

The choice of package will depend on the optimization techniques employed in the various packages, as well as the nature of the problem one is faced with. Core to the implementation of an SVM is an efficient technique for solving the QP problem obtained in the derivations above. According to [20], this problem 'is amenable to efficient numeric optimization algorithms', and a number of books exist detailing such algorithms, e.g. [57, 114, 115]. The original SVM implementation discussed in [20] makes use of a popular technique attributed to M.J.D. Powell, detailed in [115].

Extreme care must be taken when implementing these algorithms to ensure both efficiency and numerical stability, and for this reason many SVM implementations make use of quadratic optimization packages or library routines. If the implementation is not numerically stable, the algorithm may not converge, or may converge much more slowly than expected. These packages then need to be able to reliably handle convex quadratic programming problems with linear constraints [24]. [123] details several such packages, but does not cover packages introduced in the last decade. Some optimization packages covered in this review and recommended in Alex Smola's SV regression tutorial [166, Section 5.1] are MINOS, OSL and CPLEX. LOQO, a commercial package authored by R.J. Vanderbei [189], is also sometimes used.

[14] See http://www.kernel-machines.org/software.html.

[15] See http://www.idiap.ch/index.php?content=SVMTorch&IncFile=PageType&UrlTemplateType=3&cPathContenu=pages/contenuTxt/Projects/Torch/.

[16] See http://www.isis.ecs.soton.ac.uk/resources/svminfo.

The optimization technique he uses (*primal-dual path following*) can also be implemented manually, with optimizations for the support vector QP problem built in. Another technique for optimizing the Lagrangian is the gradient ascent approach of the Kernel-Adatron algorithm, detailed in [27].

SVM algorithms are processor and memory intensive, and for this reason much work has focused on optimizing the algorithm's memory and processor usage. Some of the more common techniques are discussed shortly below.

3.11.1 Chunking

The concept of chunking was already proposed by Vapnik in 1982 [190] as a modification to his 'special algorithm' for constructing separating hyperplanes. The technique was already implemented in the introductory papers on SVMs [20, 36], and was the first technique developed for handling large data sets. Initially (in [20]), the technique could only handle separable data, but the inseparable case was addressed in [36].

The idea of chunking is to only work with a portion, or *chunk*, of the data set at a time. Training is then done iteratively, with the support vectors from one chunk being maintained as data points, discarding the rest, and replacing them with enough data points which are classified incorrectly by the current approximate solution to refill the chunk. Eventually the appropriate support vectors are assimilated by this process, since training continues until the KKT conditions are met, yielding an optimal solution (the training set is processed multiple times).

The major advantage of this *working set* technique is that only information relevant to a small portion of the training set need reside in memory, namely that of the chunk currently under consideration; the memory requirement of this technique is then quadratic in the size of the working set.[17]

[17]Assuming the number of support vectors is less than the size of the working set—otherwise the working set size must expand to fit all the support vectors.

3.11.2 The Decomposition Method

A refinement of the chunking technique by Osuna et al. led to the more advanced *decomposition* method described in [92, 129, 141] for classification, and adapted for regression by Smola and Schölkopf in [166]. This technique involves selecting the working set from data points violating the KKT conditions (see Section A.3.2), while holding the other variables steady. This technique performs well in practice, but must be implemented carefully, since it does not necessarily converge [166]. An algorithm for a popular implementation of this technique is also provided in [29].

One of the important aspects of this technique is how best to select the working set. Joachims suggests a technique based on finding the variables in which direction the steepest descent of the objective function occurs. Details are in [92].

The major advantage of this technique over the regular chunking algorithm is that it can cope with situations where the memory is insufficient to handle all the support vectors simultaneously [166, Section 5.5.2]. Another technique which overcomes this problem is Sequential Minimal Optimization (SMO).

3.11.3 Sequential Minimal Optimization

SMO, developed by John Platt, is essentially a chunking algorithm where the size of the working set is 2. In this case, the optimization problem can be solved analytically instead of using an optimization package. This speeds up the training considerably, with the upper bound on processing time changing from a cubic to a quadratic [132]. In addition, the memory requirements for this technique scale linearly with the training set size.

Details on this technique can be found for classification in [131, 132, 149], and additional heuristics for improving the speed can be found in [56, 134]. Smola and Schölkopf [166] propose an extension of this technique to regression, as do Flake and Lawrence [56]. There is also a summary in [77]. SMO is employed in the LIBSVM package [29].

3.11.4 Shrinking

Shrinking, proposed by Thorsten Joachims for his SVM^{light} package, makes use of the concept of a *bounded* support vector. This is a support vector for which the corresponding Lagrange multiplier at the solution is C. In the same way that the solution of a SVM is not affected by discarding the training data which are not support vectors, one can discard the bounded support vector variables from the optimization problem, by fixing their contribution in the α vector to C, and making slight modifications to the objective function.

However, one does not know a priori which inputs will correspond to non-SVs or bounded SVs. Thus a heuristic is employed, which assumes that data points consistently displaying such behaviour for a number of iterations is likely to turn out to be one of the two. Such data points are then periodically removed from the optimization problem, effectively *shrinking* it. Once the shrunken problem is optimized, it is tested for optimality on the entire problem, and if it is not optimal, the solution is used as a starting point for further optimization. More details on this algorithm can be found in [29, 92].

3.11.5 Caching

Caching is a well-known technique for speeding up algorithms which are computationally and/or memory intensive. Typically, values that one hopes will be used regularly are stored in memory to prevent unnecessary re-evaluation or retrieval from disk. This process was pioneered in hardware where the operating system uses a small, fast *cache memory* similarly to speed regular memory access [180, Section 2.2.5]. The technique is also used for handling programs and data sets which are too large for memory by a technique known as *paging*. Various strategies can be used for determining which values, instructions or data should be cached. A discussion of paging and page replacement (and thus caching) policies can be found in [180, Section 6.1].

In the case of large data sets, the kernel matrix is often too large to be stored in memory. However, the columns of the kernel matrix corresponding to unbounded support vectors are used much more often than other columns in the solution of the optimization problem. For this reason, caching these columns can be very useful. However, since one does not know which data points these will be, a *least recently used* caching policy is typically followed. In this way, the appropriate columns are naturally cached.

Caching policies for SVM implementations are discussed in [29, 56, 92]. In particular, it should be noted that Flake and Lawrence allege that caching can degrade performance if not implemented with certain modifications to improve the cache hit rate and thus diminish the cache administration overhead [56].

3.11.6 Other Techniques

Another technique for QP, known as *successive overrelaxation* has been applied to SVMs by Mangasarian and Musicant [119]. This technique has been specifically designed for the efficient solution of large systems of equations, and has shown to perform well on the SV optimization problem, with training times substantially outperforming SMO on some benchmarks.

A probabilistic speedup is proposed for feature extraction problems in [163], and presented for machine learning in [167]. The algorithm suggested can not be applied directly to SVMs, although the authors suggest a way in which it may possibly be applied as a topic for further research. This technique yields a good, although not optimal, solution in a considerably shorter time. Interestingly, the result used for this work is derived from Hoeffding's inequality [82], which is central to the generalization bounds we derive in Chapter 4.

3.12 Benchmarks and Applications of SVMs

SVMs and SVM variants have been shown to be versatile tools, able to effectively handle a wide variety of tasks. Particularly in the case of classification and regression problems, good results have been achieved, often equalling or bettering the performance of previous state-of-the-art methods [143].

Many of these results have been obtained during the development of various optimizations and variants of SVMs, e.g. [85, 89, 196, 205]. In order to demonstrate the efficiency of new techniques, they are usually tested first on some form of 'toy problem', and often also on some benchmark problems.

A toy problem is one where the test data is generated in such a way that the optimal result is known a priori. The results obtained can then be compared to the optimal result. Due to differences in difficulty of various toy problems, a number of repositories of benchmark problems have been established. These repositories contain the data necessary to test one's proposed algorithm, thus facilitating standardization of results. In addition, the problems in these repositories vary in difficulty, allowing one to evaluate one's algorithm on a simple problem, or a problem with 'real-world' complexity. In fact, some problems in such repositories are representations of real-world tasks.

The most often used repositories in the SVM literature are the UCI [17] and DELVE [184] repositories. The UCI repository is older and larger than the DELVE repository, but the DELVE repository is better organized, and software is also provided for managing the data sets provided. Recently, the IDA repository has been established at GMD FIRST [63]. This repository aims to overcome the shortcomings inherent in the other repositories—more details can be found in [126].

In addition to the repositories and toy examples, competitions are occasionally held where machine learning techniques are employed to solve various problems. These problems can then be used later as benchmarks for testing one's algorithm. Examples of such competitions are the annual Knowledge Discovery and Data Mining (KDD) Cup [3, 199], and the Santa Fe Time Series competition [182]. Another set of such benchmark problems are those that were employed by the STATLOG project for evaluation and characterization of classification algorithms [51].

3.12.1 Classification

The most popular classification toy problems are those which involve finding a decision surface between different classes where points of each class are given in $\mathbb{R}^2$. Such problems and their solutions are easy to present and understand in diagrams, making them very popular. Examples of such problems are hand-crafted examples (e.g. [20, Figure 2] and [24, Figures 7 and 9]), those where data for each class is generated randomly from an underlying distribution [174], and those where class membership is determined by satisfying some criterion (e.g. the two spirals problem [177, 178, 205]).

In the UCI repository, SVMs have been applied predominantly to the *vowel* (11-class speaker identification based on vowel pronunciation) [38, 84, 121, 200, 196], *pima-indians-diabetes* (predicting diabetes status) [27, 187, 196, 197], *breast-cancer-wisconsin* (diagnosing breast tumours) [27, 39, 196, 197], *wine* (using chemical analysis to determine the origin of wines—3-class) [22, 84, 200, 196], *glass* (6-class glass origin classification problem) [22, 38, 121, 200, 196], and *adult* (predicting whether family income exceeds a threshold) [90, 133, 134, 149] data sets. Other UCI data sets used include *letter* (written letter identification) [38, 84], *isolet* (spoken letter identification) [38], *dermatology* (6-class disease identification) [121], *iris* (Fisher's 3-class iris data) [84, 200, 196], *ionosphere* (binary classification of radar returns) [27, 187], *sonar* (classification of sonar signals) [27, 39], *soybean* (soybean disease diagnosis) [121, 200, 196], *liver-disorders* (classification of alcohol-related liver disorders in males) [187], and a number of data sets from the STATLOG project, notably *satimage* [38, 84], *vehicle* [84], *shuttle* [38, 84], *segment* [84, 205], *German credit* [187], and *heart disease* [187, 196, 197].

Most of the DELVE repository's classification benchmarks are replications of those in the UCI repository, with the notable exception of the *Titanic* data set, where the goal is to predict whether a passenger on the Titanic survived—results on this data set can be found, inter alia, in [187, 196, 197].

The IDA classification repository consists of 13 benchmark problems, including 11 from the UCI repository and the Titanic problem from the DELVE repository. All of the problems are binary classification problems—in the cases where the original problem was not a binary problem, categories have been grouped [126]. The data is normalized so that the input vectors have zero mean and unit variance, and pre-defined splits between training and test sets are provided. In this way, benchmark tests can be more standardised, and results reported using this repository can be treated more confidently. Tests from this repository have been used on SVM variants in e.g. [126, 196, 197].

The initial real world focus of SVMs at AT&T Labs was on the problem of isolated handwritten digit recognition—basic OCR, where the characters are the 10 digits [149]. The main data sets for this application were the United States Postal Service (USPS) and National Institute of Standards and Technologies (NIST) databases. A reduced version of the NIST database, the MNIST data set is also popular. These databases contain images of handwritten digits, with the USPS images recorded from actual mail items. Many authors have published results with SVMs and variants on these data sets, regularly rivalling or outperforming state-of-the-art techniques [20, 22, 27, 36, 38, 39, 126, 134, 157, 153, 149, 143, 167, 191, 196]. Other techniques modified using the kernel trick have also been applied to this problem with good results, e.g. kernel PCA [150, 143]. SVM techniques for character recognition have now developed to the stage where authors are proposing kernels for application to on-line handwriting recognition [85].

The next major field to receive the attention of the SVM community was that of image processing. Attention was given to face detection in images, i.e. detecting if and where in an image faces are present [77, 90, 129]. Similar approaches can be used for other objects, too, an example being the pedestrian detection system of Papageorgiou et al. [85]. Another task is that of distinguishing between individuals' faces, or face recognition. A database often used for evaluating face recognition tasks is the Olivetti Research Labs database of face images and work here includes [26, 137]. Once again, the techniques can be applied to other objects, and a short overview of applications in 3D object recognition can be found in [85]. Other techniques where SVMs have been applied to images, are those of automatically categorizing images [30, 31], detecting the orientation of images [194], and detecting steganography, or the concealment of messages, in image representations [85].

The proliferation of the Internet has led to the availability of large amounts of information. In order to use this information effectively, techniques are necessary to find relevant information. The pioneering work on text categorization of news articles and websites was done by Thorsten Joachims [77, 89, 90, 93, 91, 96, 95], with significant contributions from Susan Dumais et al. [77]. Campbell notes Drucker et al.'s work on spam categorization [26], and recent work has also been done on authorship attribution of newspaper articles with SVMs [43].

Another field where SVMs are performing particularly well, and are gaining popularity, is that of bioinformatics. This popularity was reflected by the 2001 Knowledge Discovery and Data Mining (KDD) Cup problems coming solely from the bioinformatics field [3]. TSVMs have been used to improve on the prizewinning performance in one of the problems for the 2001 competition, improving the success rate from 68.4% to 81.6% [199]. SVMs have been used for translation initiation site (TIS) recognition [126, 206], a field where modifying the kernel to reflect prior knowledge has given good results. TISs are points in a nucleotide sequence where protein encodings start. Functional properties of new protein sequences can be determined by means of protein homology detection, a task also successfully tackled with SVMs in [86].

SVMs have also been used for cancer diagnosis. Beyond the Wisconsin breast cancer benchmark, SVMs have been used for survival-time classification [111] and chemotherapy recommendation [110] for patients. Another linear programming SVM variant is currently implemented as a breast cancer diagnostic system at the University of Wisconsin hospital, which has exhibited generalization of 99.48%, an almost 3% improvement over the best machine learning alternative reported [85]. [124] presents the use of SVMs for handling a component of a system for drug administration recommendation in intensive care units.

The use of SVMs for functional classification of gene expression data from DNA microarray hybridization experiments is investigated for yeast in [23, 49, 198]. Using gene expression data for cancer, specifically leukemia, diagnosis is investigated by Xu et al. in [205]. Weston et al. also apply their work in [198] to cancer diagnosis in the case of a colon cancer and a large B-Cell lymphoma problem. SVMs have thus performed very well in biological applications to date, and are quickly entrenching themselves as a useful tool in bioinformatics.

Other fields where SVM classification has been applied include high-energy physics, engine knock detection and database marketing [85].

3.12.2 Regression and Time Series Prediction

The SVM was originally proposed for classification only, but in 1995, Vapnik formulated SVMs for regression estimation [191], and his technique was modified for analysing time series data in [127], with other modifications proposed in [54, 125].

The most common toy example in regression estimation is that of estimating the sinc function $\text{sinc}(x) = \frac{\sin(x)}{x}$, with the difficulty being controlled by adding noise from different distributions [54, 136, 156, 176, 173, 193]

A number of well-known artificial benchmark time-series have been used for testing SV regression techniques, including Ikeda maps [125], the Lorenz time series [125], motorcycle data [173] the highly chaotic Mackey-Glass time series [56, 125, 127, 136], and data set D from the Santa Fe competition, where an SVM implementation improved on the previous record by 37% [85, 54, 127, 136]

Other benchmark problems used from the UCI and DELVE repositories are the *abalone* data set (forecasting the age of abalone) (UCI) [167], the *Boston housing* problem (forecasting house prices in a region of Boston) [33, 156, 143, 167, 172, 173, 196], and *kin-32fh* (predicting the position of a robotic arm) [33, 196].

SV regression has also been used for predicting the temperature of underground cables [85], and attempting to forecast financial time series [186].

3.12.3 Other Application Fields

A SV technique for solving linear operator equations was proposed by Vapnik et al. in [193], and this technique was extended to density estimation by Jason Weston in [196]. SV approaches to novelty detection have also been proposed by Tax and Duin, and Scholkopf et al. [25, 181].

3.13 Summary

This chapter presented the SVM for classification, regression estimation, and density estimation. As an approximate implementation of SRM, the algorithm is designed with inherent capacity control. A number of SVM variants and extensions of the SV classification technique to polychotomous problems have been proposed. Techniques have also been developed to reduce the computational complexity of SVM implementations. In addition, SVMs have exhibited excellent performance on many benchmark and practical problems, and are fast becoming a popular method in the machine learning toolbox.

Chapter 4

Generalization Bounds and Transductive Learning

Bounds and estimates on generalization error and their use in model selection are discussed, with special attention to transductive learning.

4.1 Introduction

This chapter begins by examining techniques for estimating or bounding the generalization error of learning techniques, with a focus on SVMs, in Section 4.2. We continue in Section 4.3 by considering how these estimates and bounds can be used for model selection. Section 4.4 presents transductive learning and the TSVM, and finally a generalization bound for finite input spaces is derived and applied to transductive learning and TSVMs in Section 4.5.

4.2 Error Estimates and Generalization Bounds

Error estimates and bounds on generalization performance of an automated system are important because they provide a certain degree of confidence in the system. For this purpose, generalization bounds are more desirable than error estimates, but are usually more difficult to obtain.

In general, PAC theory provides probabilistic bounds on the generalization error, while cross-validation provides an estimator of the generalization error. However, knowledge of the distribution of the cross-validation error estimate would also allow one to construct (probable) bounds on generalization performance.

4.2.1 Training Error and Hold-out Estimates

The simplest, and initially most obvious, estimator of generalization is the error rate obtained on a given training set. There is, however, a major flaw with this approach: the estimate obtained is biased downwards, or understates the generalization error. The extent of this bias depends on the learning algorithm, but if the technique allows overfitting to occur, for example, the training error could be very small or even zero, while the generalization error is substantial. In general, exposure of the learning algorithm to the outputs corresponding to certain data points during training enhances its ability to correctly classify those data points.

The most simple way to overcome this is by the use of a hold-out test set. Given data, one splits it into a training set s of size l and a test set s' of size l'. The learning machine is then trained on the training data, and its generalization assessed on the test set is denoted by $p_{\text{err}}^{\text{ho}}(s, s')$. This is an unbiased estimator of the generalization error of the machine trained on a sample of length l. Typically, however, one can obtain better performance by training on the entire data set provided. Assuming that training on more data yields better expected generalization (a reasonable assumption), $p_{\text{err}}^{\text{ho}}(s, s')$ is then an upwardly biased estimate of the true error of the machine trained on the entire sample. This bias decreases as the test set size decreases, while the variance of the estimator increases. Thus, in order to find a reliable estimate, a tradeoff between the initial training and test set size must be found [91]. A major advantage of this approach is that the error estimate can be computed quickly and easily. The technique of cross-validation discussed below can be seen as averaging the results of a number of hold-out tests.

4.2.2 Cross-validation Estimates

In this section, we describe the mechanism of k-fold cross-validation, and its extreme case, loo cross-validation. Suppose we are given a data set s of size l. Assume k is a factor of l.[1] We then divide the training data into k *folds* $s_1, \cdots, s_k$, and use each fold as a test set for the learning machine constructed using the other $k-1$ folds as training data. Let the number of errors on fold i be e_i, and set $\frac{l}{k} = l'$, the size of each fold. Then the error rate on fold i is $\frac{e_i}{l'}$, and the k-fold cross-validation error estimate is $p^{cv}_{\mathrm{err}}(s_1, \cdots, s_k) = \frac{1}{k}\sum_{i=1}^{k}\frac{e_i}{m} = \frac{1}{l}\sum_{i=1}^{k} e_i$. We denote the loo-error, where $k = l$, by $p^{\mathrm{loo}}_{\mathrm{err}}(s)$.

The cross-validation error estimate is an unbiased estimate of the generalization error of the machine with a training set of size $l - l'$. Since one expects a learning algorithm's performance to improve with an increase in the training set size, this introduces a very slight positive bias to the estimator as an estimate of the error with a training set of size l. We say the estimator is *almost* unbiased. In addition, the variance of this estimator decreases as the number of folds k increases. Thus loo-estimation (also referred to as *moving control* estimation [190]) provides the least biased, lowest variance cross-validation estimate of the generalization error. However, with these advantages comes the additional computational time required to repeat the learning process l times, instead of a smaller k.

In order to overcome this computational burden, techniques have been proposed for estimating and bounding the loo-error in a reduced time. We investigate some of these approaches in the case of SVMs below.

From Chapter 3, we know that data points which are not SVs do not affect the solution obtained. Utilizing this and other SVM-specific properties can yield efficient estimates of the loo-error. Since non-SVs are always classified correctly, it follows that loo-errors can only occur on the support vectors. We thus have the following bound on the loo-error: $P^{\mathrm{loo}}_{\mathrm{err}}(s) \leq \frac{\text{Number of SVs}}{l}$. Taking the expected value over all possible samples s of size l, we obtain [191,

[1] Deviations from this have a very minor impact on the results derived.

Theorem 5.2]

$$\mathbb{E}[P_{\mathrm{err}}(n-1)] = \mathbb{E}[P_{\mathrm{err}}^{\mathrm{loo}}(s)] \leq \frac{\mathbb{E}[\text{Number of SVs}]}{n} \quad ,$$

a bound on the expected true error for a machine trained on a sample of size $n-1$, $P_{\mathrm{err}}(n-1)$. However, Burges [24, Section 7.4] says he has yet to find a use for this bound, since, although it may be tighter, it is not as predictive as PAC bounds.[2] In addition, it is difficult to evaluate the expected number of SVs.

A useful application of the loo-error result above is that once the SVM has been trained on the full training set, the exact loo-error can be obtained by performing loo-testing *on the SVs only*, since we know that training with a non-SV removed will yield the same solution and the point will be classified correctly [192].

More stringent requirements on data points than merely being support vectors lead to tighter bounds on the loo-error, and even quicker calculation of the exact error. A development in this direction is a result by Jaakkola and Haussler [87], which says (for SVMs) that a point x_j can only be an error with the loo-technique if the classification obtained by $\mathrm{sgn}(\sum_{i \neq j} \alpha_i K(x_j, x_i))$ differs from the correct label.[3] This results in both a lower bound on loo-error, and a quicker technique for exact loo-error evaluation.

Joachims [91], however, notes that this result is derived for kernel machines without a bias term b, and hence is not directly applicable to regular SVMs. He proposes an alternative bound on the loo-error which he obtains by investigating the different optimization problems for the SVM with and without a data point left out. He derives a relationship between the solutions in this case, and uses this relationship to derive a necessary condition for a data point to yield an error in the loo-technique in terms of the solution vector α and the slack variables ξ. This bound can be strengthened in the

[2] Thus PAC bounds would be more appropriate for model selection purposes.

[3] Note that this corresponds to evaluating the decision function, omitting the term corresponding to x_j.

case of unbiased SVMs[4], and can be evaluated efficiently. Since the bounds are proposed for text classification, Joachims presents bounds on precision, recall and the F1 measure as well. An overview can be found in [94], with full details in [91].

Vapnik and Chapelle derive another bound on the loo-error, relying on a concept they term the *span* of a SV. They then give a necessary condition for a data point x_j to be misclassified by the loo-technique in terms of its Lagrange multiplier α_j, its span S_j, the diameter of the smallest sphere containing all the data points, and the error penalization parameter C. In addition, under an assumption which is usually approximately valid[5], the loo-error can be calculated. The resulting *span-rule* is empirically shown to produce good quality estimates of the loo-error. The interested reader is referred to [32, 192].

The major advantage of these bounds and estimates of loo-error is the property that they reduce the time taken to calculate the loo-error or bound it. This is particularly important for model selection, where parameters (e.g. the regularization parameter of SVMs) are usually selected empirically by investigating the error estimates or bounds for various combinations of parameters. Note that since all of the techniques require the solution (or Lagrange multipliers) of the trained SVM, the optimal model parameters cannot usually be determined analytically. An additional caveat is that by Scheffer [142], who warns that such empirical parameter selection may lead to *meta-level overfitting*[6]

4.2.3 PAC Bounds

A major shortcoming of the estimates above is that they do not provide a principled way to take into consideration the variance of the estimate—i.e. we have no way of assessing how *confident* we are in the error estimate,

[4]Note that in high-dimensional feature spaces, the removal of a bias term will usually not make much difference to the model.

[5]The assumption is usually invalid, but the violation and its impact are typically small.

[6]i.e. "If you optimize over a bound, it stops being one" [162].

beyond that a larger sample size induces more confidence.

The PAC model provides a framework for the derivation of bounds on the generalization error of a learning machine. These bounds are based on the capacity of the class of hypotheses the learning machine can implement. SRM utilizes these bounds for learning machines implementing nested classes of hypotheses of increasing capacity.

Most bounds studied in PAC theory are based on *uniform convergence results*. Such theorems investigate the convergence of frequencies of events to their underlying probabilities, and the rate of this convergence.[7] A number of such results are provided by Vapnik in [190]. The essential idea of generalization bounds in the PAC framework is to derive an upper bound on the integral of the upper tail of the distribution of the generalization error [26].

The foundations of the PAC model were laid by Valiant in [185], where he investigated the problem of learning Boolean functions. Important extensions of this work were the extension of the model to binary functions over domains other than $\{0,1\}^n$, e.g. $\mathbb{R}^N$ [18, 19], the decision theoretic approach pioneered by Haussler [75], the development of *p-concepts* by Kearns and Schapire [101] (where the output of the binary function is probabilistic), and techniques for handling noisy or erroneous data [100]. The techniques employed for p-concept learning, as extended in [5], can also be employed for investigating PAC learning in regression problems. The interested reader is referred to Dana Angluin's survey [2], and Martin Anthony's excellent introduction to PAC learning (with special reference to ANNs) [4].

Key concepts for understanding many of the PAC model results are various measures of capacity of hypothesis classes. Important examples of such measures are the VC-dimension and the fat-shattering dimension. The papers above, and others, utilize these quantities to state necessary and/or sufficient conditions for certain classes of hypotheses to be *learnable*.[8] One such

[7] An example of a bound not utilizing an uniform convergence result is one based on sample compression by Littlestone and Warmuth, outlined in [40, Section 4.4]. The number of support vectors plays the role of d when applying this bound to SVMs.

[8] Informally, an hypothesis class $\mathcal{F}$ is learnable if there is an algorithm which can approximate any function $f \in \mathcal{F}$ arbitrarily well, given sufficient examples labelled according

condition is that the VC-dimension of the hypothesis class must be finite.[9] In this case, the number of samples necessary for learning is dependent on the VC-dimension.

These measures are often used to bound the growth function in order to provide bounds on generalization. Examples of such bounds are used for illustration purposes in the discussion which follows. More efficient techniques for bounding the growth function by means of *entropy numbers* and *observed covering numbers* have recently been fruitful. The pioneers of these approaches are Alex Smola, Bernhard Schölkopf, John Shawe-Taylor and Robert Williamson [151, 152, 169, 202, 203, 201, 204]. The derivation of these tighter bounds will also receive attention.

Shawe-Taylor and Williamson, together with Peter Bartlett and Martin Anthony, were also the proponents of data-dependent SRM (DD-SRM) [158, 159]. SVMs, although originally designed as an approximate implementation of SRM, fully implement the DD-SRM inductive principle. Thus the bounds from this theory can be applied directly to SVMs. Many of these bounds also depend on the growth function and are traditionally bounded by the VC-dimension. However, we can also tighten these bounds using the entropy number approach mentioned above.

We will concentrate on the classification problem, with a few remarks afterwards on the multi-class problem and SV regression. The first generalization bounds were derived for finite hypothesis classes, in terms of the cardinality of the hypothesis class. These techniques were extended to handle infinite hypothesis classes by using the cardinality of an ϵ-net of the hypothesis class in a slightly modified bound [190]. This approach constructs distribution-free bounds[10] on generalization in terms of the growth function. An example of a bound using this quantity (modified from [6][11]) is

$$P_{s\in X^l}\left(\exists f\in\mathcal{F} \text{ with } \frac{P(f(x)=1)-\frac{1}{l}\sum_{i=1}^{l}\theta(h(x_i)=1)}{\sqrt{P(f(x)=1)}}\geq\epsilon\right)$$

to f.

[9] For regression, it is the fat-shattering function which must be finite.

[10] Bounds independent of the underlying distribution from which samples are drawn.

[11] This tightens a bound by Vapnik in [190].

$$\leq \quad 4\mathcal{N}(\mathcal{F}, 2l)e^{-\frac{\epsilon^2 l}{4}} \quad ,$$

where $\mathcal{F}$ is an hypothesis class, and s is a sample of length l. Using Sauer's inequality (Theorems 50 and 51), one can obtain the following bound on the generalization error of a learning machine implementing $\mathcal{F}$, and trained on an l-sample s [24]:

$$R(h) \leq R_s^0(h) + \sqrt{\frac{v(\log(\frac{2l}{v}) + 1) - \log(\frac{\delta}{4})}{l}} \tag{4.1}$$

with probability at least $1 - \delta$, where $v = \mathrm{VC}(\mathcal{H})$, the VC-dimension of $\mathcal{H}$.

Unfortunately, the VC-dimension of all hyperplanes in an N-dimensional space is $N + 1$ [40, 191], so at first glance it seems as if most kernels do not benefit from this theorem.[12] However, the SVM hyperplanes depend on the input data, which constrains the hypothesis space and decreases its VC-dimension. In this case, we can bound the VC-dimension by [9, 147, 158, 191] $v \leq R^2W^2$, where R is the radius of a sphere containing the training data in feature space, and $\|w\| \leq W$. As mentioned in Section 3.2, the distance between the SVM hyperplane and the nearest example (in the separable case) is $\gamma = \frac{1}{\|w\|}$, so that $v \leq \frac{R^2}{\gamma^2}$, the square of the ratio of the radius of the smallest sphere enclosing the training data to the margin attained by the SVM on the data. This provides us with another motivation for maximizing the margin: a larger margin yields better generalization bounds.

Equation (4.1) is only valid for a fixed hypothesis class. SRM attempts to find the optimal classifier in nested function classes of hyperplanes of increasing complexity. In the SVM context, our original hypothesis class is

$$\mathcal{H} = \{w \cdot x + b : w \in H_K, b \in \mathrm{I\!R}\} \ .$$

Using the R^2W^2 bound on VC-dimension, we can use the nested set of hypothesis classes[13]

$$\mathcal{H}(1)_\infty \subset \mathcal{H}(2)_\infty \subset \cdots \subset \mathcal{H}(i)_\infty \subset \cdots \subset \mathcal{H}(t)_\infty \ ,$$

[12] Remember that kernels construct hyperplanes in high or infinite-dimensional space.

[13] We need only consider a finite number of H_i since beyond a certain VC-dimension, the bound becomes trivial.

where $\mathcal{H}(i) = \{w \cdot x + b : \|w\| \leq i\}$. We can now search for the w_i minimizing the bound (4.1) for each $\mathcal{H}(i)_\infty$ with probability $1 - \frac{\delta}{i}$. As a result, the smallest bound should hold with probability $1 - \delta$, by means of the union bound of Theorem 54 [158].

Note that $\mathcal{H}(i)$ depends on the training data. Now, (4.1) only holds if the hypothesis classes are specified prior to obtaining the training data. Thus the approach outlined in the previous paragraph, although it may perform well empirically, is theoretically flawed. This dilemma is solved by the DD-SRM inductive principle. The SVM, with its data-dependent hypothesis classes, fully implements this generalization of SRM. The initial work on DD-SRM bounds focused on classification problems where a zero training error could be attained [158, 159]. The bounds makes use of an argument using covering numbers, the fat-shattering dimension, and a *luckiness* function.

An example of such a data-dependent result is that if a training data set for an SVM, contained in a ball of radius R centred at the origin, is correctly classified by the SVM, then with probability $1 - \delta$, the generalization error is bounded by

$$\frac{2}{l}\left(v \log(\frac{8el}{v}) \log(32l) + \log(\frac{8l}{\delta})\right) \quad ,$$

where $v = \lfloor \frac{577R^2}{\gamma^2} \rfloor$ [159, Theorem 4.9].

[10, Corollary 9] extends the results to allow nonzero empirical error. The paper shows that the true error of any hypothesis $h \in \mathcal{H}$ is bounded with probability $1 - \eta$ for all margins $\gamma \in (0, 1]$ simultaneously by

$$R_s^\gamma(h) + \sqrt{\frac{2}{m}(d \ln(\frac{34em}{d}) \log(578m) + \ln(\frac{8}{\gamma\delta}))} \quad , \tag{4.2}$$

where d is the *fat-shattering dimension* of $\mathcal{H}$ at scale $\frac{\gamma}{32}$ (see Section A.8). $R_s^\gamma(h)$ is the empirical margin risk with margin γ (see Definition 63). The behaviour of this bound with respect to γ is important: as γ increases, the empirical margin risk increases, while the second term, which is a *scale-sensitive* bound on capacity, decreases. We can thus bound the risk with probability $1 - \eta$ by the minimum of (4.2) over all $\gamma \in (0, 1]$. A variant of this approach, known as *margin percentile bounds*, is presented in [40].

Another method proposed to generate bounds in the case of inseparable data, or data achieving a small margin, is embedding the feature space into another space. The proposed embedding moves the points not achieving the margin further away from the hyperplane. It turns out that this technique [40, 160, 161] is analogous to the use of the soft-margin approach of Cortes and Vapnik [36]. The results of this work then provide PAC bounds on soft-margin SVMs.

Extensions of the PAC model and uniform convergence results for countable ranges have been derived, e.g. [6, 14, 68]. In the case of an uncountable range, such as $\mathbb{R}$, another approach is called for. The major approach to obtaining generalization bounds in the case of regression is by treating the regression problem as a classification problem (then bounding the error of this problem): in this context, not attaining a certain accuracy with the regression function at a point is considered a classification error.[14] Anthony and Bartlett introduce this approach and refer to it as *approximating from interpolated examples* [5]. This paper shows the equivalence of finite fat-shattering function of $\mathcal{H}$ and $\mathcal{H}$ being able to approximate functions (regress) into $\mathbb{R}$ from interpolated examples (training points). Generalization bounds in the form of sample complexity results are also provided.

The application of this technique applied to SV regression to derive bounds is outlined in [40, Chapter 4]. This technique is also used for the regression bound derived in Section 4.5. One can also derive PAC bounds on the expected loss for other loss functions based on the uniform convergence results for real-valued functions provided by Vapnik [190, 191].

4.2.4 Generalization via Covering Numbers

An alternative to using capacity measures such as the VC-dimension to bound the growth function, and indirectly, covering numbers, is bounding the covering numbers directly using entropy numbers. This more direct approach will hopefully yield smaller generalization bounds.

[14] Note the parallel to the ε-insensitive loss function: this technique bounds the expected loss using this loss function.

Many approaches to bounding generalization error involve bounding the expected ϵ-covering number of the hypothesis space $\mathbb{E}[\mathcal{N}(\epsilon, \mathcal{F}, s)]$ by the ϵ-growth function $\mathcal{N}(\epsilon, \mathcal{F}, l) = \sup_{s \in X^l} \mathcal{N}(\epsilon, \mathcal{F}, s)$. This is done to generate distribution-free bounds.[15] However, in the case of SVMs, the ϵ-growth function is calculated by taking the sup over either the entire feature space (recalling the older bounds using VC-dimension $N+1$), or a hypersphere containing the transformed data points in feature space (the bounds using $\frac{R^2}{\gamma^2}$). These bounds can be strengthened by taking the sup over any smaller set containing the image of input space under the feature space mapping induced by the kernel.

A classic example of this is that the image of any input in feature space lies on the unit sphere, using the Gaussian kernel, since

$$\|\varphi(x)\|^2 = \varphi(x) \cdot \varphi(x) = K(x, x) = e^{-\frac{\|x-x\|^2}{2\sigma^2}} = e^0 = 1 \ .$$

Thus in order to take the sup of the covering number in feature space for arbitrary samples in input space, one needs only to consider the sup over the unit sphere in feature space. This limitation is clearly dependent on the kernel.

In particular, the approach we sketch here, originally detailed in [203], utilizes the fact that the images of data points in input space lie in an axis-parallel parallelepiped[16]. By taking the sup over the smallest ellipsoid containing this parallelepiped, one can obtain improved bounds. Directly obtaining the covering numbers is difficult, but bounds can be placed on them in terms of entropy numbers of certain linear operators.

More specifically, we utilize Mercer's theorem (34) for a Mercer kernel K. In this case,

$$\|\varphi(x)\|^2 = K(x, x) = \sum_{j=1}^{\infty} \lambda_j \psi_j(x) \psi_j(x) = \sum_{j=1}^{\infty} (\sqrt{\lambda_j} \psi_j(x))^2 \ ,$$

[15] If one knows the distribution $P(x, y)$, one may be able to generate a better distribution-specific bound by directly bounding the expected value. See the convergence results in [191, Chapter 3].

[16] A parallelepiped is a generalization to higher dimensions of a rectangle in 2 dimensions, and is a box-like structure. More details can be found in Section A.5.

with the series converging. We consider the feature space of K to be the space of sequences, and use $\varphi_j(x) = \sqrt{\lambda_j}\psi_j(x)$. Then $\varphi(x) \in \ell_2$. We also assume[17] there is a kernel-dependent constant C_K bounding $|\psi_j(x)|$ for all $x \in X$ and $j \in \mathbb{N}$, so that $|(\varphi(x))_j| \leq \sqrt{\lambda_j}C_k$. It follows that $\varphi(x)$ lies within an axis parallel parallelepiped of sidelengths $\sqrt{\lambda_j}C_k$ in ℓ_2 for any $x \in X$.

Now we can construct an ellipsoid $\mathcal{E}_D$ containing this parallelepiped, such that it is the image of the unit ball in ℓ_2: this ellipsoid is created by scaling each component of the input, so that we are dealing with a diagonal map D which maps the j'th component of the input, x_j, to $d_j x_j$. We need to ensure that $D(x) \in \ell_2$ for all x in the unit ball, and that the ellipsoid contains the parallelepiped. This is satisfied if we let $d_j = C_K\|(\frac{\sqrt{\lambda_j}}{r_j})\|_{\ell_2} r_j$.[18] Here (r_j) is a positive sequence chosen so that the norm in the expression for d_j is finite. Note that the inverse operator A^{-1} maps the parallelepiped into the unit ball.

We now specify the SVM hypothesis classes we will consider. For a given c, we define the hypothesis class

$$\mathcal{H}(c) = \{f_w : x \to x \cdot w : x \in \mathrm{span}(\varphi(X)), \|w\| \leq c\} \ .$$

Note that these classes do not allow a bias term $+b$.

We define the *evaluation map* for a sample $s = s_1, \ldots, s_l$ as

$$S_{\varphi(s)}(w) = (\varphi(s_1) \cdot w, \ldots, \varphi(s_l) \cdot w) \ ,$$

where the range is in ℓ_∞^l. Thus $S_{\varphi(s)}$ evaluates a hypothesis on the sample s.

In order to evaluate a sample for every hypothesis in $\mathcal{H}(c)$, we consider the operator $T = S_{\varphi(s)}c$. The image of the unit ball under this operator

[17] This is a stricter requirement than necessary. A looser requirement which holds in nearly all practical situations is adequate. For more details see the note in [70, page 241], and [154].

[18] Technically, this step also requires that the measure of the input space be bounded. Usually, one is working with $\mathbb{R}^N$ and the Lebesgue measure. In this case, bounding the input space is sufficient.

corresponds to the evaluation of the sample s on all hypotheses with $\|w\| \leq c$ (i.e. $\mathcal{H}(c)$). Suppose we can bound the entropy number $\epsilon_n(T)$ of T by ϵ_0 for all samples. This means there is an ϵ_0-covering of the image of the unit ball with n or fewer elements for all samples, and thus $\mathcal{N}(\epsilon_0, \mathcal{H}(c), l) \leq n$.

We make use of the ellipsoid $\mathcal{E}_A$ defined above by rewriting T in terms of A^{-1} and A. The resulting decomposition of T allows us to obtain tighter bounds on the entropy numbers. Since

$$x \cdot w = (AA^{-1}x) \cdot w = (A^{-1}x) \cdot (A^T w) = (A^{-1}x) \cdot (Aw)$$

(where the last equality follows since A is a diagonal operator),

$$T = S_{\varphi(s)}c = (S_{A^{-1}\phi(s)})(Ac) \ .$$

We apply Theorem 45 to obtain

$$\epsilon_n(T) \leq \min \left\{ \|T\|, p\|T\| \sqrt{\frac{\log(1 + \frac{l}{\log n+1})}{\log n + 1}}) \right\} \ .$$

We also have that $\|T\| \leq \|S_{A^{-1}\phi(s)}\| \|A\| c$. We now show that $\|S_{A^{-1}\phi(s)}\| \leq 1$ for all samples s:

$$\begin{aligned} \sup_{s \in X^l} \|S_{A^{-1}\phi(s)}\|_{\ell_\infty^l} &= \sup_{s \in X^l} \left\{ \max_{1 \leq i \leq l} (A^{-1}\phi(s_i) \cdot w), \|w\| \leq 1 \right\} \\ &\leq \sup_{s \in X^m} \left\{ \max_{1 \leq i \leq l} \|A^{-1}\phi(s_i)\| \|w\|, \|w\| \leq 1 \right\} \\ &\leq 1 \ , \end{aligned}$$

since the construction of A implies that $A^{-1}\phi(s_i)$ lies in the unit ball in ℓ_2.

It follows that for any sample s,

$$\epsilon_n(T) \leq \|A\| c \min \left\{ 1, p\sqrt{\frac{\log(1 + \frac{l}{\log n+1})}{\log n + 1}}) \right\} \ .$$

Other bounds on $\epsilon_n(T)$ can be derived similarly, and these bounds inherently yield bounds on the covering numbers we seek.

Note that the eigenvalues and eigenfunctions used in this argument depend on the kernel used, so that this technique is kernel specific (although it can be used for any Mercer kernel). Also note that this technique does not cater for a bias term. Bias can be handled by directly modifying the covering number obtained, as is done in the bounds we derive in Section 4.5.

4.3 Model Selection

A significant problem when training and implementing a support vector machine, as with many other learning models, is the configuration of the model to the problem under investigation. This process is known as model selection, and a variety of techniques and heuristics are applied to do this for SVMs.

The use of the kernel trick to generalize the optimal margin classifier algorithm to SVMs led to dramatic improvements in performance, at the expense of higher computational requirements. In some cases the performance of the linear kernel $K(x_i, x_j) = x_i \cdot x_j$ is adequate. In most cases, however, the data is not almost linearly separable, and kernels can be of great use.

Empirically, it has been observed that the mere use of a high-dimensional feature space in the SVM algorithm may lead to improved performance, this performance being similar for a wide variety of types of kernel [145]. This means that although judicious choice of a kernel and its parameters may lead to even more improved performance, one may be able to obtain satisfactory performances by the use of a number of standard kernels, a number of which are discussed below. On the other hand, good model selection can lead to the groundbreaking results reported for SVMs on benchmarks and applications, not merely adequate performance [126]. However, as mentioned in [147], the exact choice of kernel function seems to be less important than the accuracy-complexity tradeoff parameter C.

With the traditional SVM classifier, model selection involves selecting the form of the kernel function, its parameters, and C, the parameter determining the bias-variance tradeoff. In the case of SV regression, the parameters of the loss function must also be selected (e.g. ε for the ε-insensitive loss). For other variants, other parameters are required (e.g. ν in ν-SVMs), but the principles discussed here can usually be applied similarly in those cases.

4.3.1 Form of Kernel Function

In the early theoretical development of the SVM, the role of the kernel function was one of the least understood aspects of the technique. In [24], Burges says that the choice of kernel is 'perhaps the biggest limitation of the support vector approach'. However, work by Smola et al. [164] has established a link between the choice of kernel and regularization operators. Thus, the kernel influences not only the class of functions from which the decision surface is selected, but also the type of regularization applied by the SVM [149, 165].

In addition, the good empirical performance of SVMs was difficult to understand until this connection with regularization theory was established. Realizing that the kernel function not only maps data to a high-dimensional space, but also implicitly performs regularization, has helped researchers to understand why SVMs are achieving such good results [165].

The initial use of the kernel trick was restricted to Mercer kernels. In particular, three families of such kernels are very popular, and all result in familiar classifiers [24, 166]:

- The *polynomial kernel* $K(x_i, x_j) = (x_i \cdot x_j + 1)^c$ yields a classifier where the decision surface is a p-th order polynomial in the data ($c \in \mathbb{N}$).
- The *Gaussian kernel* $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$ corresponds to a Gaussian RBF classifier [157].
- The *sigmoidal kernel* $K(x_i, x_j) = \tanh(\alpha x_i \cdot x_j - c)$ corresponds to a type of three-layer neural network [77]. Note that this kernel is not

an admissible Mercer kernel for certain values of κ and δ (particularly for $\alpha < 0$ or $c < 0$).

The difference between the SVM approach and the traditional approach can be seen by considering the following. By training using the SVM approach with a Gaussian kernel, the number of RBF centres is automatically determined, while this would be part of the model selection process for RBF classifiers. Similarly, the number of nodes in the perceptron corresponding to the sigmoidal kernel solution is automatically determined [24].

[126] also presents the inverse multiquadric kernel $K(x_i, x_j) = \frac{1}{\sqrt{\|x_i - x_j\|^2 + c^2}}$ as a common kernel function, although we have not yet seen it applied to SVMs.

Vapnik et al. propose the use of kernels which generate B_n-splines or Fourier expansions (using *Dirichlet* kernels) for function approximation in [193]. Smola and Schölkopf [164, 165] show that only B_n-splines with odd n are admissible kernels, and thus term them $B_{(2n+1)}$-splines. A discussion of these kernels and the Gaussian kernel from a regularization point of view is also presented in these papers.

Another contribution by Smola and Schölkopf is proposing the use of kernels generated by conditionally positive definite functions [164]. Schölkopf suggests their use when the kernel trick is applied to algorithms using only the distance between points, and also for representing translational invariance as prior knowledge in SVMs [144, 145].

Other kernels available in the RHUL SVM [141] include V. Vovk's real polynomial $K(x_i, x_j) = \frac{1-(x_i \cdot x_j)^c}{1 - x_i \cdot x_j}$ (including the limiting case as $c \to \infty$), a generalization of the polynomial kernel, the *semilocal* kernel $K(x_i, x_j) = [x_i \cdot x_j + 1]^c e^{-\|x_i - x_j\|^2 \sigma^2}$, and a proposed analysis of variance (ANOVA) kernel by Vapnik. More on ANOVA decomposition kernels can be found in [172].

Schölkopf et al. discuss the construction of kernels for incorporating prior knowledge in the context of OCR in [153]. This is an early example of the general trend that has been developing in SVMs: for optimal results, kernels should be tailored for the problem at hand, by incorporating the prior knowledge available, specifying the type of regularization, or using other characteristics of the problem.

An example of using a different characteristic of the problem is the development of *cluster kernels* for semi-supervised learning. In semi-supervised learning, one is provided with labelled training examples, but also typically a large amount of unlabelled data (which could be the a priori known test set). A typical assumption made to utilize this data to enhance learning machines is that data in the same class tends to appear in clusters. [34] then focuses on the development of these cluster kernels, which attempt to avoid placing decision surfaces through clusters (or regions of high data density) in input space.

However, the development of appropriate kernels for a task can be a relatively complex task. For this reason, Weston proposes multi-kernel SVMs in [196]. This makes use of a pre-defined *dictionary* of candidate kernel functions $K_1, K_2, \ldots, K_\kappa$, for each of which a weight vector is calculated simultaneously in a generalization of the SVM optimization problem. These weight vectors are then combined into a decision function of the form

$$f(x,a) = \sum_{i=1}^{l} (\sum_{j=1}^{\kappa} a_{ij} K_j(x_i, x)) + b \ .$$

Other potential benefits of this proposal are its robustness, its ability to generate more varied decision surfaces in a controlled manner, and that one can use multiple kernels, each one appropriate in only certain parts of input space [196].

4.3.2 Kernel and SVM Parameters

This section focuses on techniques for determining SVM parameters (e.g. C and ε), as well as the parameters of the kernel function (e.g. p for the polynomial kernel, or δ and κ for the sigmoidal kernel).

[142] and [126] both list three typical approaches to model selection for

determining numerical parameters, viz. Bayesian model selection (including minimum description length approaches [191]), complexity penalization and regularization (including PAC theory), and hold-out testing (including cross-validation). All of these approaches can be used for SVMs. Despite their differences, they all attempt to choose the parameters which minimize some bound on or estimate of the generalization error.

Bayesian model selection is based on MacKay's *evidence framework*, presented in [117, 118]. This is a general model selection approach, and was customised for application to SVMs by Kwok in 1999 [106, 107]. This approach uses three levels of Bayesian inference to build a classifier. The first level approximately corresponds to the training of the SVM for a given set of parameters (i.e. determining the support vector coefficients and the bias). The second level typically corresponds to finding the optimal regularization parameter by an iterative technique, and the third level to determining the kernel parameters by considering the *model evidence*. This technique is, however, not widely used for SVM model selection.

Complexity control approaches typically attempt to select parameters which optimize bounds on the generalization errors in terms of function class complexity. This amounts to choosing a parameter which minimizes a certain generalization bound. For example, the bounds derived in Section 4.5 can be used for model selection for transductive SVMs.

The most popular model selection techniques are hold-out testing methods, most notably cross-validation [74, 91]. This process is described in Section 4.2. Typically, cross-validation is performed on the trained machine for a number of different parameter values. The combination of parameters yielding the lowest estimate are chosen for the model. The major disadvantage of this approach is that the SVM must be trained k times for each combination of parameters. Also, in practice, the selection of parameters in this way is often a multi-phase process, where a rough range of parameters is first identified by cross-validation, and a second phase then searches that range more thoroughly.

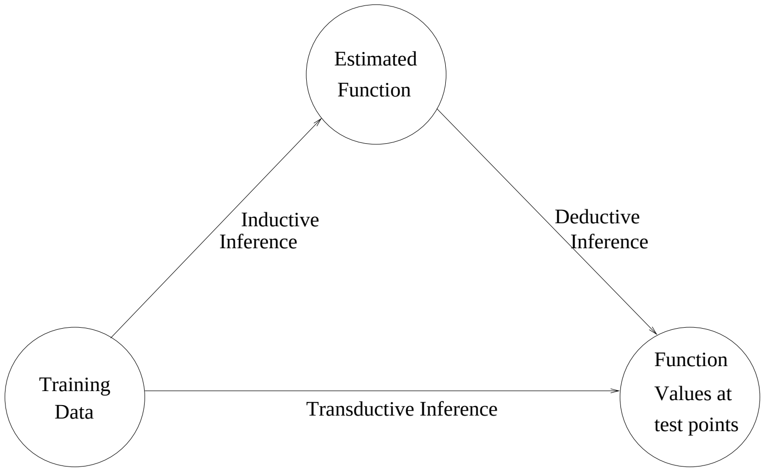

Figure 4.1: Types of Learning

4.4 Transductive Learning: Modifying the Support Vector Machine

In [191], Vapnik proposed his main principle for solving problems with limited information:

> *When solving a given problem, try to avoid solving a more general problem as an intermediate step.*

It is this principle which led to the development of the transductive support vector machine, which will be discussed below.

Figure 4.1 (from [191, page 169]) illustrates the distinction between inductive and transductive learning. Inductive learning attempts to learn a function representing the dependence between the inputs and outputs. The output for a new point is then found by deductive learning—plugging the input into the learnt dependence. However, if the finite set of future input points for the learning problem are known a priori, the entire dependence need not be estimated, only the values of the dependence at the given points—a simpler problem. This technique of *estimating the function at given points* is called transductive learning [33, 65, 93].

The transductive approach is generally more accurate on the given test set, and thus can provide similar performance with smaller training sets [33]. In addition, applying transductive learning is a technique for overcoming the violation of SRM in the original SVM formulation (see Section 4.2). The drawback is that the machine is then designed to provide outputs only for a fixed, finite set of *predefined* inputs. A typical example of such a situation is when one must categorize the data in a database. Classifying each datum manually can be costly (in time and/or money), so a subset of the data is categorized and a learning machine is then trained on those data points to classify the other points. Techniques for selecting which data points to classify is another research problem [26, Section 2.6], and possible techniques for this include active selection [118] and cluster analysis [52, 88].

Following the introduction of transductive inference by Vapnik [190, 191], and his provision in 1998 of a uniform convergence result of training error to test error similar to that originally provided for SVMs, a practical approximate implementation for larger data sets, the TSVM, was proposed in 1999 by Thorsten Joachims for text classification (the binary case) [93]. Essentially, Joachims proposes a technique for approximately solving the transductive inference optimization problem presented by Vapnik. Transductive Regression (based on Ridge Regression, which is linked to SVMs, and linear discriminant techniques) for function estimation was also proposed in 1999 by Olivier Chapelle et al. [33].

A core concept in understanding transductive inference is that sets of functions in the inductive case which would have equal outputs at the given test points are essentially equivalent. Thus, the test set partitions the space of possible functions into equivalence classes, where f and g are in the same class if $f(x) = g(x)$ for all x in the test set. Thus, in the case where the function to be estimated has a finite range (e.g. classification), the inductive hypothesis space is partitioned into a finite transductive hypothesis space. The problem then reduces to finding an appropriate hypothesis class, or set of labels in the classification case.

Formally, given a training set $(x_1, y_1), \ldots, (x_l, y_l)$, with $y_i \in \{1, -1\}$, and a test set $x_1^\dagger, \ldots, x_k^\dagger$. Then, the optimization problem for the linearly inseparable case, analogously to (3.9) and (3.10), is

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2} w \cdot w + C \sum_{i=1}^{l} \xi_i + C^\dagger \sum_{i=1}^{k} \xi_i^\dagger \\ \text{subject to} \quad & y_i (w \cdot x_i + b) \geq 1 - \xi_i, \;\; i \in \{1, \ldots, l\} \\ & y_i^* (w \cdot x_i^\dagger + b) \geq 1 - \xi_i^\dagger, \;\; i \in \{1, \ldots, k\} \\ & \xi \geq 0 \\ & \xi_\dagger \geq 0 \;\; , \end{aligned} \tag{4.3}$$

where the minimization is done over the weight, bias, slack variables, and *the set of labels for the test set* $\{y_i^\dagger : i \in 1, \ldots, k\}$. Note that this formulation involves finding a hyperplane which separates the proposed class division of the training and test data maximally.

Since the labels are not continuous, but are either 1 or -1, the obvious approach is a combinatorial one: try all possible combinations of labels, and see which leads to the lowest objective function, but this approach quickly becomes infeasible, even for small test sets. Joachims' solution approximates this by a local search over the possible labels. He first trains a regular inductive SVM on the training set to obtain an initial proposed labelling of the test set, and then tunes this by changing labels which lead to a reduction in the objective function (4.3). More details on the algorithm can be found in [93], where Joachims also proves that his algorithm converges in a finite number of steps. Although Joachims does not discuss it, the kernel trick can be applied to his technique for generating nonlinear decision surfaces. An interesting project would be the direct extension of TSVMs to polychotomous classification.

Importantly, the objective function above has two user-defined parameters, C and $C^\dagger$. In fact, Joachims' proposed algorithm has three: however, one parameter is specific to his implementation. This parameter specifies the number of test examples in the positive class. The use of this parameter is important to make his algorithmic implementation tractable and to counter potentially poor choices of cost parameters. Using this constraint, however, effectively restricts his search of the hypothesis space, possibly yielding nonoptimal solutions. Choosing these parameters, along with the choice of a kernel and its parameters, forms the problem of model selection.[19]

Although a transductive SV regression approach is theoretically possible, the resulting optimization problem is very complex. Chapelle et al. [33] propose a form of transductive regression based on ridge regression.[20] Ridge regression estimates can be found analytically, and this advantage carries over to the transductive regression they propose.

4.5 Generalization for Transductive Learning

In this section, a simple margin-dependent PAC bound on the generalization error for learning in a finite input space is presented. One can apply these bounds to transductive learning by considering the finite set of points in the training set and the test set together as a finite input space. The bound provided holds for dichotomous and polychotomous classification, as well as regression using the ε-insensitive loss function. The bound allows training errors, and, to our knowledge, is the first covering number PAC bound specifically for arbitrary learning algorithms in finite input spaces[21]. As a result, we can potentially strengthen the better known bounds in various ways, particularly for our focus area, transductive learning, when the training set is substantially larger than the test set.[22] The new bound depends on the empirical margin error, the size of the training set, and the covering number of the hypothesis class measured at a scale related to the margin. Although certain technicalities common in the derivation of similar bounds

[19] Note that if a model selection technique selects good cost parameters, Joachim's third parameter may be unnecessary except from the perspective of computational cost.

[20] Ridge regression can be seen as a variant of SVM, by using a linear kernel and squared error loss.

[21] Lanckriet et al. proposed bounds for transductive learning in 2002 [108]. Their technique provides bounds on the transductive test error for kernel methods, subject to their proposed learning approach, involving semi-definite programming [188], being followed. Their proofs make use of Bartlett et al.'s recent work on Rademacher complexities, as in e.g. [11].

[22] Note that such problems are much less common than the opposite situation.

are not employed in this bound, we hope that the derivation of this bound will still be valuable to the novice.

Many bounds utilize a symmetrization argument on a squashed function class to obtain bounds in terms of covering numbers on a double sample.[23] This bound, however, relies on covering numbers of the entire finite input space. Although the size of the input space (training set and test set) in transductive problems may exceed twice the size of the training set, resulting in larger covering numbers, tighter bounds on the covering numbers of the entire hypothesis space may still be easier to obtain. The bound is applied to SVMs, using a technique for bounding the covering numbers of the SVM hypothesis class in terms of the eigenvalues of a Gram matrix. Bounds for SVMs both with and without bias terms, are presented. The bounds also hold for other estimation techniques which select estimates from the same hypothesis class as SVMs, such as least squares estimation and kernel ridge regression. These bounds can also be used for model selection for SVMs.

4.5.1 Assumptions, Definitions and Notation

We assume a finite input space S, and that the labels associated with positive and negative examples are 1 and -1 respectively in the dichotomous case. In the polychotomous case, labels are assumed to come from the set $\{1, \cdots, k\}$, and for regression we assume the range of the underlying function is $\mathbb{R}$. Denoting the set of outputs by Y, we further assume an underlying distribution $P(x, y)$ on $Z = S \times Y$, from which examples are generated independently.[24]

For each type of problem, we consider a set of hypotheses $\mathcal{H}$ with which one can generate outputs in Y. For regression, the hypothesis is simply the real-valued function. In the case of dichotomous classification, the outputs are generated by taking the sign of a real-valued function. In the polychotomous case, we make use of a dummy variable target encoding of the class

[23] This approach makes use of Theorem 66 with $m = l$.

[24] Note that in practice, examples for transductive learning are not generated independently for the training set, or the test set.

membership. Each hypothesis is a vector of k real-valued functions, with the sign of each component indicating membership of the corresponding class. Since membership is unique, we transform the vector in order to ensure that no more than one of the component functions is positive, by means of the canonical operator.

For the rest of the section, we assume that all hypotheses in the hypothesis space $\mathcal{H}$ for polychotomous classification have already been subjected to the canonical operator[25], i.e. the hypothesis space is of the form $\{\Delta f : h \in \mathcal{H}'\}$ for some other hypothesis space $\mathcal{H}'$. The target of this modified hypothesis is then a binary k-component vector y, with value 1 in the component representing the correct class, and value -1 in the other components.

4.5.2 An Error Bound for Transductive Learning

We intend to derive a probabilistic generalization bound on the expected risk, in terms of maximum deviation from the empirical margin risk. That is, we want to bound, for some $\gamma \geq 0$,

$$P_{s \in Z^m} \left(\sup_{h \in \mathcal{H}} \left(R\left(h\right) - R_s^{\gamma}\left(h\right) \right) \geq \epsilon \right) \quad ,$$

where the probability is with respect to all possible m-samples.[26]

Theorem 2. *For any $\gamma \geq 0$, and $m \in \mathbb{N}$,*

$$P_{s \in Z^m} \left(\sup_{h \in \mathcal{H}} \left(R\left(h\right) - R_s^{\gamma}\left(h\right) \right) \geq \epsilon \right) \leq \mathcal{N}\left(\frac{\gamma}{2}, \mathcal{H}, d_S \right) e^{-2m\epsilon^2} \quad .$$

Proof. Let $\mathcal{P}$ be a $\frac{\gamma}{2}$-cover of $\mathcal{H}$ with respect to d_S of minimal cardinality.

- *Dichotomous classification:* Then, for any $h \in \mathcal{H}$, there is a $p\left(h\right) \in \mathcal{P}$ such that
$$\sup_{x \in S} \left\{ \left| h\left(x\right) - p\left(h\right)\left(x\right) \right| \right\} \leq \frac{\gamma}{2} \quad .$$

[25] The results still hold if this coding is not used, but the utility of the resulting risk definitions below is doubtful.

[26] Since the sample is drawn independently according to $P\left(x, y\right)$, the distribution is derived from the product of the probabilities of drawing each element in the sample.

It follows by the triangle inequality that, if $h(x)\,y \leq 0$, then

$$p(h)(x)\,y \leq h(x)\,y + \frac{\gamma}{2}y \leq \frac{\gamma}{2} \ .$$

Referring to Definition 62, we conclude that

$$R(h) \leq \sum_{(x,y)\in Z} \theta\left(p(h)(x)\,y \leq \frac{\gamma}{2}\right) P(x,y) \ .$$

It also follows from the triangle inequality that, if $p(h)(x)\,y \leq \frac{\gamma}{2}$, then

$$h(x)\,y \leq p(h)(x)\,y + \frac{\gamma}{2}y \leq \frac{\gamma}{2} + \frac{\gamma}{2}y \leq \gamma \ .$$

Now, referring to Definition 63, we obtain

$$R_s^{\gamma}(h) \geq \frac{1}{m}\sum_{i=1}^{m} \theta\left(p(h)(x_i)\,y_i \leq \frac{\gamma}{2}\right) \ .$$

Now set $\tau(p,x,y) = \theta\left(p(x)\,y \leq \frac{\gamma}{2}\right)$.

- *Polychotomous classification:* Then, for any $h \in \mathcal{H}$, there is a $p(h) \in \mathcal{P}$ such that

$$\sup_{x\in S, k\in Y} \{|h_k(x) - (p(h))_k(x)|\} \leq \frac{\gamma}{2} \ .$$

It follows by the triangle inequality that, if $\min_{k\in Y}\{h_k(x)\,y_k\} \leq 0$, then

$$\min_{k\in Y}\{(p(h))_k(x)\,y_k\} \leq \min_{k\in Y}\{h_k(x)\,y_k\} + \frac{\gamma}{2}y \leq \frac{\gamma}{2} \ .$$

Referring to Definition 62, we conclude that

$$R(h) \leq \sum_{(x,y)\in Z} \theta\left(\min_{k\in Y}\{(p(h))_k(x)\,y_k\} \leq \frac{\gamma}{2}\right) P(x,y) \ .$$

It also follows from the triangle inequality that

$$\min_{k\in Y}\{h_k(x)\,y_k\} \leq \min_{k\in Y}\{(p(h))_k(x)\,y_k\} + \frac{\gamma}{2}y \leq \frac{\gamma}{2} + \frac{\gamma}{2}y \leq \gamma$$

if $\min_{k\in Y}\{(p(h))_k(x)\,y_k\} \leq \frac{\gamma}{2}$.
Now, referring to Definition 63, we obtain

$$R_s^{\gamma}(h) \geq \frac{1}{m}\sum_{i=1}^{m} \theta\left(\min_{k\in Y}\{(p(h))_k(x_i)\,y_{ik}\} \leq \frac{\gamma}{2}\right) \ .$$

Now set $\tau(p,x,y) = \theta\left(\min_{k\in Y}\{p_k(x)\,y_k\} \leq \frac{\gamma}{2}\right)$.

- *Regression:* Then, for any $h \in \mathcal{H}$, there is a $p(h) \in \mathcal{P}$ such that

$$\sup_{x \in S} \{|h(x) - p(h)(x)|\} \leq \frac{\gamma}{2} \ .$$

It follows by the triangle inequality that, if $|h(x) - y| \geq \varepsilon$, then

$$|p(h)(x) - y| \geq |h(x) - y| - |h(x) - p(h)(x)| \geq \varepsilon - \frac{\gamma}{2} \ .$$

Referring to Definition 62, we conclude that

$$R(h) \leq \int_{(x,y) \in Z} \theta\left(|p(h)(x) - y| \geq \varepsilon - \frac{\gamma}{2}\right) dP(x,y) \ .$$

It also follows from the triangle inequality that

$$|h(x) - y| \geq |p(h)(x) - y| - |p(h)(x) - h(x)| \geq \left(\varepsilon - \frac{\gamma}{2}\right) - \frac{\gamma}{2} \leq \varepsilon - \frac{\gamma}{2}$$

if $|p(h)(x) - y| \geq \varepsilon - \frac{\gamma}{2}$.

Now, referring to Definition 63, we obtain

$$R_s^{\gamma}(h) \geq \frac{1}{m} \sum_{i=1}^{m} \theta\left(|p(h)(x_i) - y_i| \geq \varepsilon - \frac{\gamma}{2}\right) \ .$$

Now set $\tau(p, x, y) = \theta\left(|p(x) - y| \geq \varepsilon - \frac{\gamma}{2}\right)$.

We now define

$$\Omega_s(p) = \sum_{(x,y) \in Z} \tau(p, x, y) P(x,y) - \frac{1}{m} \sum_{i=1}^{m} \tau(p, x_i, y_i) \ ,$$

the deviation between the expected and empirical margin error of p with margin $\frac{\gamma}{2}$.

From the upper bound on expected risk, and lower bound on empirical margin risk derived for each case above, it follows (with summation over Z replaced by integration in the regression case) that

$$\begin{aligned}
& P_{s \in Z^m}\left(\sup_{h \in \mathcal{H}} \left(R(h) - R_s^{\gamma}(h)\right) \geq \epsilon\right) \\
\leq \ & P_{s \in Z^m}\left(\sup_{h \in \mathcal{H}} \Omega_s(p(h)) \geq \epsilon\right) \\
\leq \ & P_{s \in Z^m}\left(\sup_{p \in \mathcal{P}} \Omega_s(p) \geq \epsilon\right) \\
\leq \ & \mathcal{N}\left(\frac{\gamma}{2}, \mathcal{H}, d_S\right) \sup_{p \in \mathcal{P}} P_{s \in Z^m}\left(\Omega_m(p) \geq \epsilon\right) \ . \qquad (4.4)
\end{aligned}$$

We can now apply Theorem 61 since we see that in all cases the first term of Ω_s is the expected value corresponding to the sample average calculated in the second term. Since the τ are indicator functions, $b_i = 1$ in the theorem, so we can rewrite (4.4) as

$$\begin{aligned} & \mathcal{N}\left(\frac{\gamma}{2}, \mathcal{H}, d_S\right) \sup_{p \in \mathcal{P}} P_{s \in Z^m}\left(E\left(S_m\right) - S_m\right) \\ \leq\ & \mathcal{N}\left(\frac{\gamma}{2}, \mathcal{H}, d_S\right) \sup_{p \in \mathcal{P}} e^{-2m\epsilon^2} \\ =\ & \mathcal{N}\left(\frac{\gamma}{2}, \mathcal{H}, d_S\right) e^{-2m\epsilon^2} \ . \end{aligned}$$

□

The corresponding bound obtained for classification in an arbitrary input space in [10] is $2\mathcal{N}\left(\frac{\gamma}{2}, \pi_\gamma\left(\mathcal{H}\right), 2m\right) e^{-\frac{m\epsilon^2}{2}}$. Here the use of $2m$ in the covering number denotes the sup of the covering numbers over all samples of length $2m$. Let us consider the differences between our result and this one.

First, we note that for an input space where $|S|$ is much larger than $2m$, the bound of Bartlett will most likely be superior. But in the case of transduction where our test set is small relative to the training set, our input space cardinality is closer to m than to $2m$, which could yield significantly better results.

The factor 2 improvement before the covering number, and the factor 4 improvement in the exponent, are due to the fact that we do not need to use a result based on a double sample argument[27], since we bound over the entire input space.

For the classification problems, our result can also be strengthened further to include the squashed function class used by Bartlett. The covering number over the function class $\mathcal{H}$ can simply be replaced by the covering number of the squashed function class $\pi_\gamma\left(\mathcal{H}\right)$. Replacing an hypothesis by its squashed equivalent does not have any effect on either of the risks we consider[28], and so we can apply the result to the squashed function class, which will typically have a considerably smaller covering number (see Theorem 49).

[27] This is the result used to obtain (1) in [10].

[28] Squashing outputs from beyond an absolute value of γ to an absolute value of γ do not change their classification, and other points are not affected.

We also note that the same proof (with summation replaced by integration) holds for infinite input spaces. In this case, however, the covering numbers are usually very large, often infinite, so that the resulting bounds are trivial.

We thus conclude that our bound will be useful for problems with a finite input space, when the training set comprises a substantial portion of the space. This is often the case in transductive learning when the test set is small.

Theorem 3. *For any $\gamma \geq 0$ and $m \in \mathbb{N}$,*

$$P_{s \in Z^m}\left(\exists h \in \mathcal{H} : R(h) \geq R_s^{\gamma}(h) + \sqrt{\frac{1}{2m}\left(\ln \mathcal{N}\left(\frac{\gamma}{2}, \mathcal{H}, d_S\right) - \ln \delta\right)}\right) \leq \delta \ .$$

Proof. The result follows by setting the right hand side of the formula in Theorem 2 to δ, and solving for ϵ. □

We now have a bound holding for a γ chosen a priori. To make the bound useful, we would like to be able to select γ after observation of the trained machine on the training data. In order to achieve this, we apply Theorem 68:

Theorem 4. *Define*

$$\Psi(x, y) = \sqrt{\frac{1}{2m}\left(\ln \mathcal{N}(x, \mathcal{H}, d_S) - \ln y\right)} \ .$$

For any $m \in \mathbb{N}$, $r > 0$ and $0 < \delta < 1$,

$$P_{s \in Z^m}\left[\bigcup_{\gamma \in (0, r]}\left(\exists h \in \mathcal{H} : R(h) \geq R_s^{\gamma}(h) + \Psi\left(\frac{\gamma}{4}, \frac{\delta\gamma}{2r}\right)\right)\right] \leq \delta \ .$$

Proof. For any $m \in \mathbb{N}$, we work in the probability space $\left(Z^m, 2^{Z^m}, P_{s \in Z^m}\right)$. Define

$$E(\alpha_1, \alpha_2, \delta) = \left\{s \in Z^m : \exists h \in \mathcal{H} : R(h) \geq R_s^{\alpha_2}(h) + \Psi\left(\frac{\alpha_1}{2}, \delta\right)\right\} \ .$$

We proceed by showing that these sets satisfy the requirements of Theorem 68:

1. For $0 < \alpha \leq r$ and $0 < \delta \leq 1$,

$$E(\alpha, \alpha, \delta) = \left\{ s \in Z^m : \exists h \in \mathcal{H} : R(h) \geq R_s^\alpha(h) + \Psi\left(\frac{\alpha}{2}, \delta\right) \right\}$$

so that $P(E(\alpha, \alpha, \delta)) \leq \delta$, by Theorem 3.

2. This requirement follows since all sets of samples are measurable.

3. From the definition of empirical margin risk,

$$\alpha \leq \alpha_2 \Rightarrow R_s^\alpha(h) \leq R_s^{\alpha_2}(h)$$

We also know that a covering number is an increasing function of the resolution (Theorem 48), so that

$$\alpha_1 \leq \alpha \Rightarrow \mathcal{N}(\frac{\alpha_1}{2}, \mathcal{H}, d_S) \geq \mathcal{N}(\frac{\alpha}{2}, \mathcal{H}, d_S)$$

and finally

$$0 < \delta_1 \leq \delta \Rightarrow \ln \delta_1 \leq \ln \delta$$

Putting this together yields, for $0 < \alpha_1 \leq \alpha \leq \alpha_2 \leq r$ and $0 < \delta_1 \leq \delta \leq 1$

$$R_s^{\alpha_2}(h) + \Psi\left(\frac{\alpha_1}{2}, \delta_1\right) \geq R_s^\alpha(h) + \Psi\left(\frac{\alpha}{2}, \delta\right) \quad .$$

This means that

$$\begin{aligned} E(\alpha_1, \alpha_2, \delta_1) &= \{s \in Z^m : \exists h \in \mathcal{H} : R(h) \geq R_s^{\alpha_2}(h) + \Psi\left(\frac{\alpha_1}{2}, \delta_1\right) \\ &\subseteq \{s \in Z^m : \exists h \in \mathcal{H} : R(h) \geq R_s^\alpha(h) + \Psi\left(\frac{\alpha}{2}, \delta\right) \\ &= E(\alpha, \alpha, \delta) \ . \end{aligned}$$

Applying Theorem 68 yields, for $0 < a, \delta < 1$,

$$P_{s \in Z^m}\left[\bigcup_{\alpha \in (0, r]} \left(\exists h \in \mathcal{H} : R(h) \geq R_s^\alpha(h) + \Psi\left(\frac{\alpha a}{2}, \frac{\delta \gamma (1-a)}{r}\right)\right)\right] \leq \delta \ .$$

Setting $\alpha = \gamma$ and choosing $a = \frac{1}{2}$ yields the result. □

It may now seem that we have merely transferred the a priori choice of γ to the choice of r. However, we have good guidelines for the choice of r:

- For regression we only consider $\gamma < \varepsilon$, so we can set $r = \varepsilon$.[29]

[29] This corresponds to a type of scaling on the output—we could have scaled all outputs in regression by a factor of $\frac{1}{\varepsilon}$ and simply used $r = 1$.

- In the case of classification, the constant value of r can often be related to the learning algorithm in use. For example, a value of $r = 1$ is usually suitable for SVMs, since the learning algorithm attempts to obtain a margin of 1 on the training data.[30]

4.5.3 An Application to TSVMs

In order to obtain practical bounds, we must still bound the covering number appearing in this result. Of course the bound on the covering number depends on the function class for the learning algorithm. We investigate the class of functions corresponding to an inner product with one point in a sphere of radius c, on a finite measure subset X of an inner product space

$$\mathcal{H}(c) = \{f_w : x \to \langle w, x\rangle : |w| \leq c\} = c\mathcal{H}(1) \ .$$

We choose these $\mathcal{H}(c)$ since they form the nested hypothesis space structure for SVMs [20, 24, 36, 67, 166, 193] and TSVMs [93] using the linear kernel (without bias). We shall bound $\mathcal{N}\left(\frac{\gamma}{4}, \mathcal{H}(c), d_S\right)$ by considering the eigenvalues of the Gram matrix G with respect to the points of S. We shall employ the following bound on covering numbers [152, 146].

Theorem 5. *For $u \in \mathbb{N}$, let $v = (v_1, \cdots, v_u)$ be a vector in X, and let G be the Gram matrix induced by v. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_u$ be the eigenvalues of G. Set $\lambda_{u+1} = 0$. Define, for all $n \in \mathbb{N}$,*

$$k\,(n) = \min\left\{ j \in \{1, \cdots, u\} : \lambda_{j+1}^{j} \leq \frac{\lambda_1 \cdots \lambda_j}{n^2} \right\} \ ,$$

and for $n = 2, 3, \cdots$,

$$\Upsilon\,(n) = 6 \min_{j \in \{1, \cdots, n-1\}} \left[2^{\frac{1-j}{k(2^j-1)}} \left(\lambda_1 \cdots \lambda_j\right)^{\frac{1}{2k(2^j-1)}} c\,(n,j) \right] \ ,$$

where

$$c\,(n,j) = \min\left(1, 1.86\sqrt{\frac{\log\left(\frac{u}{n-j}+1\right)}{n-j}}\right)$$

Then, for $n \in \mathbb{N}$ and $\epsilon > \Upsilon\,(n)$,

$$\mathcal{N}\left(\epsilon, \mathcal{H}(1), d_v\right) \leq 2^{n-1} \ .$$

[30] Note that margin here refers to the difference between the output of the SVM and 0, not the distance between the hyperplane and the input point in feature space.

Combining this result with Theorem 4 and Theorem 47 yields (setting v to a vector enumerating S)

Theorem 6. *Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{|S|}$ be the eigenvalues of the Gram matrix induced by the vector v which enumerates S. Set $\lambda_{|S|+1} = 0$. Let $k(n)$ and $\Upsilon(n)$ be as in Theorem 5.*

Then, simultaneously for all $\gamma \in (4c\Upsilon(n), r]$,

$$P_{s \in Z^m}\left(\exists h \in \mathcal{H}(c) : R(h) \geq R_s^\gamma(h) + \sqrt{\frac{1}{2m}\left(n \ln 2 - \ln \frac{\delta\gamma}{r}\right)}\right) \leq \delta \ .$$

This bound applies to SV function classes which do not incorporate a bias term. In order to allow a bias term, we make use of a concept proposed in [203, Sect. 9.1]. For a metric d, suppose we are given an $\frac{\epsilon}{2}$-cover V of a function class $\mathcal{F}$, and a bound B on the norm of the bias b. Then $W = \left\{B - (2i+1)\frac{\epsilon}{2} : i \in \left\{0, \cdots, \lceil 2\frac{B}{\epsilon} \rceil\right\}\right\}$ is an $\frac{\epsilon}{2}$-cover of the constant functions with norm bounded by B. From V and W, we can construct an ϵ-cover of $\mathcal{F}_B = \{f + b : f \in \mathcal{F}, -B \leq b \leq B\}$, namely

$$V_B = \{v + w : v \in V, w \in W\}$$

It follows that $\mathcal{N}(\epsilon, \mathcal{F}_B, d) \leq \lceil 2\frac{B}{\epsilon} \rceil \mathcal{N}\left(\frac{\epsilon}{2}, \mathcal{F}, d\right)$.[31] Combining this with our bound involving covering numbers, and Theorem 5, yields

Theorem 7. *Choose $B > 0$.[32] Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{|S|}$ be the eigenvalues of the Gram matrix induced by the vector v which enumerates S. Set $\lambda_{|S|+1} = 0$. Let $k(n)$ and $\Upsilon(n)$ be as in Theorem 5.*

Then, simultaneously for all $\gamma \in (8c\Upsilon(n), r]$,

$$P_{s \in Z^m}\left(\exists h \in \mathcal{H}(c)_B : R(h) \geq R_s^\gamma(h) + \sqrt{\frac{\left(n \ln 2 + \ln\left(\frac{r}{\delta\gamma}\right)\lceil\frac{8B}{\gamma}\rceil\right)}{2m}}\right) \leq \delta.$$

Theorems 6 and 7 provide bounds which are only based on the Gram matrix of dot products. The kernel trick (see Section A.7.3) can thus be employed, directly providing bounds for dichotomous classification and regression TSVMs using Mercer kernels. The bounds remain exactly the same,

[31] The cover they construct in the cited paper seems to be a $\frac{3\epsilon}{2}$-cover instead of an ϵ-cover, but the principle remains the same.

[32] The choice of B should be influenced by the discussion in [201, Sect. 5.3].

except that all calculations are then based on the Gram matrix with i, j-entry $K(v_i, v_j)$.

4.5.4 Discussion and Conclusions

This section proposes probabilistic generalization bounds on the risk of learning algorithms in finite input spaces, and hence also transductive learning. Although generic bounds for inductive learning also apply in the transductive case, this PAC bound enhances existing bounds by using the finiteness of the input space implied by transductive problems. The bound allows the use of covering numbers of the squashed function class [10, 67] present in other classification bounds, and also has the advantage of allowing training errors. We note that our bound is most effective when the test set size is small compared to the training set size[33].

A possible improvement on the results presented may be the use of other concentration of measure inequalities, such as the result by Boucheron et al. in [21], instead of Hoeffding's inequality.

One of the limitations of the bounds presented in this paper is their use for polychotomous classification. The role of the canonical operator is to enforce an analog of the dichotomous margin on the outputs of the hypothesis. However, due to the formulation of the operator, it will probably be difficult to directly obtain the covering numbers of $\Delta\mathcal{H}$. A result linking the covering numbers of $\mathcal{H}$ and $\Delta\mathcal{H}$ could be useful in this regard.

In [151], Schölkopf et al. suggest two possible generalizations of their results on bounding SVM hypothesis classes with entropy numbers: applying the result to transduction, and handling nonzero training errors. Both of these are addressed in this section.

TSVMs cannot generate all functions in $\mathcal{H}(c)$, but only functions of the

[33] This makes sense, since we then have a labelling of a relatively large proportion of our input space. In this sense, the work of Lanckriet et al. [108] is useful, since it bounds the error on the test set.

form

$$\mathcal{H}'(c) = \left\{ f_\alpha : x \to \sum_{i=1}^{|S|} \alpha_i \langle x, v_i \rangle : \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \alpha_i \alpha_j \langle v_i, v_j \rangle \leq c \right\} .$$

By investigating covering numbers of this function class directly, one should be able to obtain further improvements to the bounds derived. Work on bounding these covering numbers has been done in [201].

The bounds derived could potentially be used for SVM model selection, either for determining the regularization constant, which depends on c, or to determine certain kernel parameters, as in [152].

A limitation of the SVM bounds is $\Upsilon(n)$ in Theorems 6 and 7. For Theorem 6 to be nontrivial, we need $\Upsilon(n) < \frac{r}{8c}$, and in the case of Theorem 6, $\Upsilon(n) < \frac{r}{4c}$.

Another interesting avenue for further research in transductive bounds is the derivation of distribution-dependent bounds, particularly for the uniform distribution common in practice.

4.6 Summary

A major aspect of implementing a machine learning technique is the problem of model selection. This involves selecting the model or machine architecture and the parameters for this model or architecture in order to obtain good generalization.

This is usually approached by choosing a model which seems suitable for the learning problem at hand, and further making decisions which minimize some estimate of or bound on the generalization error.

A number of such estimates and bounds are available. The most popular model selection strategy currently is cross-validation. However, for computationally intensive models, model selection using PAC bounds may be preferable.

Generally, PAC bounds make use of dimension-like quantities such as the VC-dimension. More recently, an entropy number approach to bounding covering numbers has been proposed for linear hypothesis classes. This approach may yet be fruitfully applied to other classes of hypotheses.

A generalization bound in terms of covering numbers is presented and applied to transductive learning in the last section. This bound applies to regression and multi-class classification. The entropy number approach mentioned above is applied to this bound to yield bounds not only on TSVMs, but also on the application of other techniques yielding linear hypotheses for transductive learning.

Chapter 5

Conclusion and Future Research Directions

This thesis set out to place SVMs in the learning theory framework by considering how they handle the major issues encountered in supervised learning. All of the major issues identified were addressed, excepting rule extraction from the trained SVM. This is a very open field with many possibilities.

A property of SVMs (and other regularization techniques) which can sometimes cause trouble is their sensitivity to the scaling of data. For this reason, the IDA repository [63] has scaled all factors in its data sets to zero mean and unit variance, so that techniques can be compared easily. The problem means that scaling the data can alter the performance of the technique. Generally, the zero-mean, unit-variance solution is proposed as a heuristic solution, although the best strategy for handling scaling still seems to be an open problem.

Another useful potential extension of SVMs is a technique for handling missing data or *incomplete* vectors, i.e. data points where not all attributes are available.

The TSVM for transductive classification has been proposed, and used for multi-class problems, by the standard multi-class techniques. A direct multi-class extension of the TSVM based on an uniform convergence result may be interesting new work. In addition, no transductive SVM for regression has yet been proposed, possibly due to the computational complexity involved. A more tractable linear method for transductive regression, based on ridge regression, has been proposed, however [33]. It may be useful to consider extending this technique using the kernel trick.

In this thesis, a new generalization bound applicable to transductive learning was derived. This bound was applied to classes of linear methods, and nonlinear techniques derived from them with the kernel trick (including the SVM). The entropy number technique employed to apply the bound to these linear methods has thus far only been applied to classes of linear functions. Yet, there may be considerable potential in applying the technique to other hypothesis classes. A related problem to this is pointed out by Alex Smola [166]: currently, model selection using generalization bounds is inaccessible to most practitioners.

A number of shortcomings of current generalization bounds derive from the assumptions underlying the derivation. Addressing these shortcomings could be the basis of future research. A typical example of this is the assumption that training and test data are generated independently according to the underlying distribution of input-output space. This assumption precludes the use of the results when applying active selection for selecting the training data or on-line learning. Most bounds also are unable to cope with sampling bias, and unequal misclassification costs, both situations common in practice.

Appendix A

Mathematical Background

This appendix covers mathematical background useful for the understanding of this thesis, including some functional analysis, statistics and optimization theory.

A.1 Notation and Comments

Throughout this thesis, log represents logarithm base 2, while ln represents the natural logarithm.

The notation $\alpha > \beta$ for vectors or sequences α and β, means that the components of α exceed the corresponding components of β. A similar interpretation holds for equality, and other inequalities. When β is a scalar, the meaning is that one should consider a vector with all components equal to β.

If $f(x)$ is a function of a vector x in $\mathbb{R}^N$, $\frac{df}{dx}(x)$ denotes the vector of partial derivatives $(\frac{\partial f}{\partial x_1}(x), \cdots, \frac{\partial f}{\partial x_N}(x))$. Similarly partial differentiation with respect to a vector represents the vector of partial derivatives.

We denote the cardinality (number of elements) of a set A by $|A|$.

In the background following, certain sacrifices to technicalities and generalities had to be made to keep the appendix a manageable size. If the background does not help clarify a topic, it is recommended you consult the various texts cited, but be warned that many of them are even more technical, and assume deeper background knowledge.

A.2 Useful Functions and Concepts

Hypothesis space is the set of functions from which a learning algorithm selects its solution. Version space is the subset of acceptable solutions in hypothesis space. For example, the perceptron algorithm tries to construct a separating hyperplane between two classes. The set of all hyperplanes is then the hypothesis space, and version space is then the set of all hyperplanes which successfully separate the two classes.

This section further details some of the useful functions employed in the thesis.

The sign function $\mathrm{sgn}(x)$ maps a nonzero real number x to $\frac{x}{|x|}$, and maps 0 to 1; i.e. $\mathrm{sgn}(x) = 1$ if $x \geq 0$, and -1 otherwise.

An *indicator function* θ is a function which maps predicates (or assertions) into $\{0, 1\}$,

$$\theta(p) = \begin{cases} 0 & \text{if } p \text{ is false,} \\ 1 & \text{if } p \text{ is true.} \end{cases}$$

In the special case where p is of the form $x > 0$ for some x, the indicator function is the *Heaviside* function [126], and we often just write $\theta(x)$:

$$\theta(x) = \begin{cases} 0 & \text{for } x < 0, \\ 1 & \text{for } x \geq 0. \end{cases}$$

The sigmoid, logistic or hyperbolic tangent function [179, page 594], is

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad .$$

This is a monotone increasing function mapping $\mathbb{R}$ onto $(0, 1)$.

The concept of the *canonical operator* comes from [67]:

Definition 8 Canonical Operator. The canonical operator is a mapping Δ from a vector-valued function h to Δh as follows: for any $x \in \mathrm{dom} h$, let $M_1(x)$ be the smallest index l satisfying $h_l(x) = \max_k h_k(x)$, and $M_2(x)$ the smallest index $l \neq M_1(x)$ satisfying $h_l(x) = \max_{k \neq M_1(x)} h_k(x)$. Then

$$(\Delta h)_k(x) = \begin{cases} \frac{1}{2}\left(h_k(x) - h_{M_2(x)}(x)\right) & \text{if } k = M_1(x), \\ \frac{1}{2}\left(h_k(x) - h_{M_1(x)}(x)\right) & \text{otherwise.} \end{cases}$$

The effects of applying the canonical operator to h is a new function Δh such that no more than one component of Δh is positive, and at least one component is nonnegative.

Finally, we consider the *squashing operator*

$$\pi_\gamma(h)(x) = \begin{cases} \gamma & \text{if } h(x) \geq \gamma, \\ -\gamma & \text{if } h(x) \leq -\gamma, \\ h(x) & \text{otherwise.} \end{cases}$$

Essentially, $\pi_\gamma(h(x))$ yields $h(x)$, but squashes any values with absolute value exceeding γ back to $\gamma \mathrm{sgn}(h(x))$.

We can apply either of these operators to a set of functions. For example, applying the squashing operator to a function class $\mathcal{H}$ yields

$$\pi_\gamma(\mathcal{H}) = \{\pi_\gamma(h) : h \in \mathcal{H}\} \quad .$$

A.3 Optimization

In this section we assume familiarity with basic multivariate calculus, particularly differentiation.

Optimization concerns itself with finding the extreme values of a function, often subject to some constraints. Since minimizing a function f is equivalent to maximizing $-f$, we need only consider the problem of finding the minimum of a function. This section in particular lacks rigour, attempting to rather explain the motivation behind the techniques employed in the thesis. An excellent book covering the material here is [130].

A.3.1 Constrained Optimization

In many cases, we are interested in minimizing f over only a subset of its domain. Usually, this subset can be specified in terms of constraints, inequalities of the form $g(x) \leq 0$ and equations of the form $h(x) = 0$. These are called inequality and equality constraints, respectively. Our problem is then to minimize an *objective function* $f(x)$, subject to

$$\begin{aligned} g_i(x) \leq 0 \quad & \text{for } i \in \{1, \cdots, m\} \ , \quad \text{and,} \\ h_i(x) = 0 \quad & \text{for } i \in \{1, \cdots, n\} \ . \end{aligned}$$

We call this the *primal problem* (P). We assume $x \in \mathbb{R}^N$. The subset of $\mathbb{R}^l$ for which all the constraints of the optimization problem hold is called the *feasible region* of the problem.

We now introduce the concept of convexity of a set and a function [104, Appendix II].

Definition 9. A subset S of a vector space is convex if for any two points $x, y \in S$,

$$\alpha x + (1 - \alpha) y \in S$$

for all $\alpha \in [0, 1]$.

This definition says that if the line segment joining any two points in a set is always contained in the set, we call the set convex.

Definition 10. A real-valued function g is said to be convex if, for any $x, y \in \text{dom } g$,

$$g(\alpha x + (1 - \alpha) y) \leq \alpha g(x) + (1 - \alpha) g(y)$$

for all $\alpha \in [0, 1]$.

An intuitive way of understanding this is that the line segment joining $g(x)$ and $g(y)$ must lie above the graph of g. This is then equivalent to saying that the set of points above the graph is convex.

The concept of convexity is important for two main reasons. First, under mild assumptions which we assume hold throughout, a convex function has only one minimum. Without the assumption of convexity, a function may have multiple local minima, which confounds many minimization techniques. The second benefit convexity yields is because the intersection of a finite number of convex sets is convex. This means that if all the constraints are convex functions, the feasible region turns out to be the intersection of a number of convex sets, and is hence convex. This turns out to be very convenient.

The much simpler case of minimizing convex functions under convex constraints is called *convex programming.* The function and constraints in this case are together called a convex program.

A.3.2 The Karush-Kuhn-Tucker Conditions

We assume in this section that all functions and constraints are well-behaved, and in particular are twice differentiable. This is the case throughout the thesis, except in the case where absolute values are encountered. This can usually be remedied by replacing the constraint $|g(x)| - c \leq 0$ by the pair of constraints $g(x) - c \leq 0$ and $-g(x) - c \leq 0$.

We recall that in order for a point x^o to be an unconstrained extremum of f, we need $\frac{df}{dx}(x^o) = 0$. It can be shown that for x^o to be a constrained extremum, we need [16, Section 1.4]

- x^o to satisfy the constraints,
- for some vectors $\kappa^o \geq 0$ and λ^o,

$$\frac{df}{dx}(x^o) + \sum_{i=1}^{m} \kappa_i^o \frac{dg_i}{dx}(x_0) + \sum_{i=1}^{n} \lambda_i^o \frac{dh_i}{dx}(x) = 0 \text{ , with}$$

- $\sum_{i=1}^{m} \kappa_i^o g_i(x_0) = 0$.

In the case of a convex program, if we replace the last requirement by the stricter *KKT complementarity condition*

$$\kappa_i g_i(x_0) = 0 \text{ for all } i \in \{1, \cdots, m\} \ ,$$

the conditions become necessary and sufficient for an extremum [138, Theorem 28.3]. These three conditions (the first two above and this one), are known as the KKT (also often Kuhn-Tucker) conditions.

A vector pair (κ^o, λ^o) satisfying the KKT conditions is called a Kuhn-Tucker vector [138, Section 28]. For such a Kuhn-Tucker vector, consider the function

$$f_{KT}(x) = f(x) + \sum_{i=1}^{m} \kappa_i^o g_i(x) + \sum_{i=1}^{n} \lambda_i^o h_i(x) \ \ .$$

Suppose we find the extrema of $f_{KT}(x)$ without constraints, and obtain a set of solutions D. Consider the subset D_0 of D of points satisfying the original constraints. Finally, we eliminate from D_0 any point x for which $g_i(x) < 0$ for a nonzero κ_i. Call the resulting set D_1. We see that D satisfies the second necessary condition above, D_0 satisfies the first two, and D_1 satisfies all three. Thus, by selecting an appropriate subset of the extrema of f_{KT}, we obtain all the extrema of f.

Once again, in the case of a convex program, things are easier. In this case, f_{KT} is convex since it is a sum of convex functions. This means it has only one global minimum, x^o. Thus D has only one element, and since our original convex program must have a minimum contained in D, it follows that x^o is the unique solution of the convex program, without the need to construct D_0 and D_1 [138, Corollary 28.1.1].

A.3.3 Lagrange Multipliers

We now consider the *primal Lagrangian* of (P),

$$L_P(x, \kappa, \lambda) = f(x) + \sum_{i=1}^{m} \kappa_i g_i(x) + \sum_{i=1}^{n} \lambda_i h_i(x) \ \ .$$

Note that L_P is similar to f_{KT}, but now κ and λ are variables of the function, instead of forming a pre-defined Kuhn-Tucker vector. We call these variables Lagrange multipliers. The primal Lagrangian is important due to the important Kuhn-Tucker theorem [138, Corollary 28.3.1], stated in an extremely simplified form here:

Theorem 11. *Let (P) be a convex program. A point x^o solves (P) if and only if there exist some (κ^o, λ^o) such that $(x^o, \kappa^o, \lambda^o)$ is a saddle point of L_P (with respect to minimizing in x and maximizing in (κ, λ)).*

This theorem makes use of the concept of a saddle point:

Definition 12. A vector pair (x_s, y_s) is called a saddle point [138, page 281] of a function $f(x, y)$ (with respect to minimizing in x and maximizing in y) if for all x, y,

$$f(x, y_s) \geq f(x_s, y_s) \geq f(x_s, y) \quad .$$

Thus, for a saddle point (x_s, y_s), setting $y = y_s$ and considering $f(x, y_s)$ as a function of x, f attains a minimum at $x = x_s$. Similarly f as a function of y with $x = x_s$ achieves a maximum at $y = y_s$.

For convex programs, we are then interested in finding the saddle point of the primal Lagrangian. An elegant method of doing this makes use of the concept of *duality*.

A.3.4 Duality

Given a primal convex program (P), we can rewrite f in terms of L_P over the feasible region. Let x be a point in the feasible region. Then

$$\max_{\kappa \geq 0, \lambda} L_P(x, \kappa, \lambda) = \max_{\kappa \geq 0, \lambda} \left[f(x) + \sum_{i=1}^{m} \kappa_i g_i(x) + \sum_{i=1}^{n} \lambda_i h_i(x) \right] \quad .$$

Since x is in the feasible region, $g_i(x) \leq 0$ for $i \in \{1, \cdots, m\}$, and $h_i(x) = 0$ for $i \in \{1, \cdots, n\}$, so that the maximum is achieved when $(\kappa, \lambda) = 0$, and equals $f(x)$.

Thus, we can rewrite (P) as (P'):

$$\text{find} \qquad \min_x \left\{ \max_{\kappa, \lambda} \{L(x, \kappa, \lambda)\} \text{ subject to } \kappa \geq 0 \right\}$$

$$\begin{array}{ll} \text{subject to} & g_i(x) \le 0, \;\; i \in \{1, \cdots, m\} \\ \text{and} & h_i(x) = 0, \;\; i \in \{1, \cdots, n\} \;\; . \end{array}$$

We now propose an analog of (P') called the minimax dual of (P'). This is obtained by reversing the order of optimization in (P'), and is denoted by (D'):

$$\begin{array}{ll} \text{find} & \max_{\kappa,\lambda} \left\{ \min_x \{L(x,\kappa,\lambda)\} \text{ subject to } g_i(x) \le 0, \;\; i \in \{1, \cdots, m\} \right. \\ & \left. \text{and } h_i(x) = 0, i \in \{1, \cdots, n\} \right\} \\ \text{subject to} & \kappa \ge 0 \;\; . \end{array}$$

Note that the minimax dual of (D') is once again (P').

We now come to the importance of duality for finding the saddle point: $(x^o, \kappa^o, \lambda^o)$ is the saddle point of L_P if and only if x^o is the solution of (P), $(x^o, \kappa^o, \lambda^o)$ is the solution of (D) and

$$f(x^o) = L_P(x^o, \kappa^o, \lambda^o) \;\; .$$

In addition, for any x satisfying the primal constraints,

$$f(x) \ge L_P(x, \kappa, \lambda)$$

for any (κ, λ) satisfying $\kappa \ge 0$ and $\frac{\partial L_P}{\partial x}(x, \kappa, \lambda) = 0$. These statements form the essence of Wolfe's duality theorems [130, Section 12.4].

The value of $f(x) - L_P(x, \kappa, \lambda)$ for appropriate κ, λ is called the duality gap, and is the basis for the *primal-dual path following* optimization technique. Here, the optimal x is found by sequential updates, decreasing the duality gap until it reaches 0.

Consider any fixed pair (κ, λ). Since (P) is a convex program, $L_P(x, \kappa, \lambda)$ is a convex function of x. It follows that a point x^* satisfying the constraints minimizes $L_P(x, \kappa, \lambda)$ subject to the primal constraints if and only if

$$\frac{\partial L_P}{\partial x}(x^*, \kappa, \lambda) = 0 \;\; .$$

Thus (D') is equivalent to the so called Wolfe dual problem (D):

$$
\begin{array}{ll}
\text{find} & \max\limits_{x,\kappa,\lambda}\{L_P(x,\kappa,\lambda)\} \\
\text{subject to} & \kappa \geq 0, \\
& g_i(x) \leq 0, \;\; i \in \{1,\cdots,m\}, \\
& h_i(x) = 0, \;\; i \in \{1,\cdots,n\} \\
\text{and} & \dfrac{\partial L_P}{\partial x}(x,\kappa,\lambda) = 0 \;\; .
\end{array}
$$

A.3.5 Quadratic Programming

A special form of convex programming, quadratic programming, occurs when the constraint functions g_i and h_i are linear functions of x, and the objective function is a quadratic function of x.[1] Suppose (P) is a quadratic programming problem.

We can then write the constraints as

$$
\begin{aligned}
g_i(x) = g_{i0} + \sum_{j=1}^{N} g_{ij}x_j \quad &\leq \quad 0 \;\; , \\
h_i(x) = h_{i0} + \sum_{j=1}^{N} h_{ij}x_j \quad &= \quad 0 \;\; ,
\end{aligned}
$$

and the objective function as

$$
f(x) = f_0 + \sum_{j=1}^{N} f_j x_j + \sum_{j=1}^{N}\sum_{k=1}^{N} f_{jk}x_i x_j \;\; .
$$

We consider the Wolfe dual (D) of (P). This requires

$$
\begin{aligned}
& \frac{\partial L_P}{\partial x_j}(x,\kappa,\lambda) \\
= \;\; & \frac{\partial f}{\partial x_j}(x) + \sum_{i=1}^{m} \kappa_i \frac{\partial g_i}{\partial x_j}(x) + \sum_{i=1}^{n} \lambda_i \frac{\partial h_i}{\partial x_j}(x) \\
= \;\; & 0 \text{ for } j \in \{1,\cdots N\} \;\; .
\end{aligned}
$$

[1] If the function to be minimized is linear, the problem is called a linear programming problem, and it can be easily solved by a technique known as the *simplex method*, detailed in [104, Section 5.3] (although more efficient techniques exist).

Now

$$
\begin{aligned}
\frac{\partial g_i}{\partial x_j}(x) &= g_{ij} \ , \\
\frac{\partial h_i}{\partial x_j}(x) &= h_{ij} \ , \text{and} \\
\frac{\partial f}{\partial x_j}(x) &= f_j + \sum_{j=1}^{N}\sum_{k=1}^{N}(f_{jk} + f_{kj})x_k \ .
\end{aligned}
$$

Substituting these formulae into the N equations above yields N linear constraints in the N unknowns x_j. Substituting these constraints into L_P and the constraints g_i and h_i where possible, one can eliminate some of the primal variables x_j from L_P and the constraints. The resulting Lagrangian function is called the dual Lagrangian L_D.

The Wolfe dual problem (D) can then be formulated as the problem of maximizing L_D subject to $\kappa \geq 0$, the constraints which were not used in obtaining L_D from L_P, and the modified constraints obtained after substitution. The partial derivative constraints which have been used to modify L_P to L_D will automatically hold.

Example

Let (P) be the problem of minimizing

$$f(x, y) = xy$$

subject to $x = y$ and $x \geq 3$. We can clearly see this is equivalent to minimizing x^2, subject to $x \geq 3$, so that the optimal solution is $(x, y) = (3, 3)$, where $f(x, y) = 9$. Let us see how this is obtained with Lagrange multipliers.

In this case, using Lagrange multipliers $\kappa \geq 0$ and λ, we have the primal Lagrangian,

$$L_P(x, y, \kappa, \lambda) = xy + \kappa(3 - x) + \lambda(x - y)$$

To compute the Wolfe dual, we take the partial derivatives

$$\frac{\partial L_P}{\partial x}(x, y, \kappa, \lambda) = y - \kappa + \lambda = 0 \tag{A.1}$$

$$\frac{\partial L_P}{\partial y}(x, y, \kappa, \lambda) = x - \lambda = 0 \tag{A.2}$$

$$\tag{A.3}$$

From (A.1), we have $y = \kappa - \lambda$, and from (A.2), we have $x = \lambda$. Substituting these into L_P, we obtain

$$\begin{aligned} L_D(\kappa, \lambda) &= \lambda(\kappa - \lambda) + \kappa(3 - \lambda) + \lambda(\lambda - (\kappa - \lambda)) \\ &= \lambda\kappa - \lambda^2 + 3\kappa - \lambda\kappa + 2\lambda^2 - \lambda\kappa \\ &= \lambda^2 - \lambda\kappa + 3\kappa \ . \end{aligned}$$

Substituting into the constraints, we have $\lambda = \kappa - \lambda$, and $\lambda \geq 3$.

Since we used each partial derivative to eliminate a variable, (P) is equivalent to this dual problem (D): maximize L_D subject to $\kappa \geq 0$, $\lambda \geq 3$ and $\kappa = 2\lambda$.

In this case the dual problem does not seem to be any simpler than the original problem, but for more complex problems, it is often the case. We can solve (D) intuitively by substituting $\kappa = 2\lambda$, yielding the problem of maximizing

$$\begin{aligned} L_D(\lambda) &= \lambda^2 - 2\lambda^2 + 6\lambda \\ &= -\lambda^2 + 6\lambda \end{aligned}$$

subject to $2\lambda \geq 0$ and $\lambda \geq 3$. Clearly we need only consider the second constraint, as it implies the first. This is a parabola with turning point $\frac{-6}{-2} = 3$, so that the optimal solution is $\lambda = 3$, where $L_D(\lambda) = 9$, the optimal value of $f(x, y)$. In addition, $x = \lambda = 3$, and $y = \kappa - \lambda = 2\lambda - \lambda = 3$.

This is essentially the approach followed in the SVM formulations of Sections 3.2 and 3.4, when partial derivatives are set to 0, and the resulting formulae substituted back into the primal Lagrangian, Note that, in these cases, as in the example, the dual Lagrangians L_D no longer utilize any of the primal variables of the original optimization problems.[2]

[2] A result of this is that the original constraints fall away in the dual formulation, as do (3.4) in the basic formulation and (3.12) in the inseparable case.

A.4 Spaces

We assume familiarity with vector spaces [105, Section 2.1], in particular Euclidean N-space ($\mathbb{R}^N$).

Definition 13. A σ-algebra [1, Chapter 3] on a set X is a collection of subsets of X closed under finite intersection, complementation, and countable union.

A simple example (and the only one we use) of a σ-algebra on any set X is the collection of all its subsets, known as its *power set*, and denoted by 2^X.

Definition 14. A measure [1, Chapter 3] on a σ-algebra S is a function $\mu : S \to [0, \infty)$ satisfying

- $\mu(\emptyset) = 0$; and
- for any disjoint sequence of sets $\{A_n\}$ from S,

$$\mu(\bigcup_{n=1}^{\infty} A_n) = \sum_{n=1}^{\infty} \mu(A_n) \quad .$$

Definition 15. A measure space [50] is a triple (X, S, μ), with X a set, S a σ-algebra on X, and μ a measure on S. If $\mu(X) = 1$, the triple is a probability space, and μ is called a probability measure.

We can associate with any probability space random variables which behave in a certain fashion: the variables are elements of X (the domain), and are elements of some of the elements of S (possible events). $\mu(S)$ then measures the probability of a random variable belonging to S (i.e. the probability of the event occurring).

Suppose we have a probability measure μ defined on a σ-algebra on X. From μ we can generate a function $f : X \to [0, \infty)$ by $f(x) = \mu(\{x\})$. Such a function f is called a p.d.f..

Definition 16. A metric (or distance function) [47, Section I.6] in a set X is a function d satisfying, for all $x, y, z \in X$,

- $d(x,y) \geq 0$;
- $d(x,y) = 0$ if and only if $x = y$;
- $d(x,y) = d(y,x)$; and
- $d(x,y) \leq d(x,z) + d(z,y)$.

An example of a metric in $\mathbb{R}^n$ is the Euclidean distance between two points

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)} = \|x - y\| \ .$$

Another example used in this thesis is Hamming distance. Here $d(x,y)$ is defined as the number of differing bits in the sequences x and y. It is easily verified that this d satisfies the requirements of a norm.

Informally, a metric d is a dissimilarity measure: the more dissimilar x and y are, the greater the value of $d(x,y)$.

Definition 17. A normed space [47, Section II.3] is a vector space[3] X, together with a function $|\cdot| : X \to [0,\infty)$ satisfying, for all $x,y \in X$,$\alpha \in \mathbb{R}$,

- $|x| = 0$ if and only if $x = 0$;
- $|x+y| \leq |x| + |y|$; and
- $|\alpha x| = |\alpha||x|$.

Such a function is called a norm.

Note that the Euclidean metric above can also be defined in terms of the norm of the difference between the two points. If X corresponds to a normed space, this is a general approach to deriving a metric from a norm.

A common norm defined on matrices is the Frobenius norm. If A is a $m \times n$ matrix, the Frobenius norm of A is just the Euclidean norm in $\mathbb{R}^{mn}$ of the vector obtained by concatenating the rows of A. Thus the squared Frobenius norm of a matrix A is just the sum of the squared elements of A.

[3] We use $\mathbb{R}$ as the scalar field.

Definition 18. An inner product space [105, Section 3.1] is a vector space X, together with a function $\langle \cdot, \cdot \rangle : X \times X \to \mathbb{R}$ satisfying[4], for all $x, y, z \in X$,$\alpha \in \mathbb{R}$,

- $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$;
- $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle$;
- $\langle x, y \rangle = \langle y, x \rangle$;
- $\langle x, x \rangle \geq 0$; and
- $\langle x, x \rangle = 0$ if and only if $x = 0$.

Such a function is called an inner product.

$\langle x, y \rangle$ is often written as $x \cdot y$, and this is the notation used through most of the thesis.

A well-known example of an inner product on $\mathbb{R}^n$ is the so-called dot-product

$$x \cdot y = \sum_{i=1}^{n} x_i y_i \quad .$$

Note that in this case $(x \cdot x)^{\frac{1}{2}} = \|x\|$. Once again, this is a general rule for obtaining a norm from an inner product. Thus we see that an inner product space is a normed space, and a normed space is a metric space.

We also see that we can obtain the norm of ℓ_2 from the inner product

$$x \cdot y = \sum_{i=1}^{\infty} x_i y_i$$

and the norm for $L_2(A)$ from

$$f \cdot g = \int_A f(x) g(x) dx \quad .$$

In Euclidean space, one can obtain the formula $x \cdot y = \|x\| \|y\| \cos \theta$, where θ is the angle between x and y. In this sense, we can see the inner product

[4] The requirements differ slightly with other scalar fields.

as a similarity measure, in that if x and y are vectors in similar directions (small θ), $x \cdot y$ will be larger than if the angle between them was larger (with the inner product dropping to 0 when the angle reaches $\frac{\pi}{2}$).

We now turn our attention to various normed spaces, that we deal with in the thesis:

- ℓ_p^m, for $p \in [1, \infty)$, is the inner product space consisting of all m-component vectors, using the ℓ_p-norm, defined as

  $$\|x\|_p = (\sum_{i=1}^{m} |x_i|^p)^{\frac{1}{p}} \quad .$$

  We see that our usual interpretation of $\mathbb{R}^N$ as N-dimensional Euclidean space is equivalent to the space ℓ_2^N. For $p = \infty$, we define a norm as follows:

  $$\|x\|_\infty = \sup_{i=1}^{m} |x_i| \quad .$$

  This norm follows by taking the limit of the p-norm.

- If we let the number of components in the vectors of ℓ_p^m become infinite, we obtain the collection of spaces ℓ_p. The norms are defined in the same way as for ℓ_p^m, except that the sums are taken to infinity. However, the set used is not all infinite sequences, but instead only the set of sequences for which the sum in the norm converges. Thus ℓ_p is the set of sequences

  $$\{x : \|x\|_p \text{ is finite. }\}$$

  where

  $$\|x\|_p = (\sum_{i=1}^{\infty} |x_i|^p)^{\frac{1}{p}} \quad .$$

  Similarly to the m-dimensional case, we can extend the definition to ℓ_∞.

- Finally we consider the L_p spaces. These are similar to ℓ_p, except that we consider the space of real-valued continuous functions[5]. Thus

[5] Actually,we consider the completion of this space under the metric defined below.

we can see a point in the space (i.e. a function) as an uncountable 'sequence'. We extend the norm by replacing the sum with an integral (with the domain of integration being a parameter to the space), and keep the concept of restricting the set to those functions for which the norm converges. Thus $L_p(A)$ is the set of functions

$$\{f : \|f\|_p \text{ is finite. }\}$$

with

$$\|f\|_p = (\int_A |f(x)|^p dx)^{\frac{1}{p}} \ .$$

The extension to the space $L_\infty(A)$ of functions bounded on A is similar, obtaining norm

$$\|f\|_\infty = \sup_{x \in A} |f(x)| \ .$$

If A is an interval in $\mathbb{R}$, it is standard to omit the parentheses, thus writing $L_p[a,b]$ for $L_p([a,b])$, for example.

We can obtain an inner product from the norm for $p = 2$ in all the cases above, so that ℓ_2^m, ℓ_2 and $L_2(A)$ are inner product spaces with the respective inner products:

$$\begin{aligned}
\langle x, y \rangle_{\ell_2^m} &= \sum_{i=1}^{m} x_i y_i \ , \\
\langle x, y \rangle_{\ell_2} &= \sum_{i=1}^{\infty} x_i y_i \ , \\
\langle f, g \rangle_{L_2(A)} &= \int_A f(x) g(x) dx \ .
\end{aligned}$$

Definition 19. An Hilbert space [105, Section 3.1] is a complete inner product space.

Here completeness means that any Cauchy series in the space converges. We will not go into the technical aspects of this, but rather just mention that this means that $\mathbb{R}^n$, $L_2[a,b]$ and ℓ_2 are Hilbert spaces [105, Section 3.1].

Definition 20. A Banach space is a complete normed space.

Once again we do not go into detail on completeness, but note that any Hilbert space is also a Banach space, using the norm derived from the inner product.

A.5 Geometric Concepts

In this section, we present some common geometric concepts from $\mathbb{R}^2$ and $\mathbb{R}^3$, and consider generalizations to $\mathbb{R}^N$ and other inner product spaces.

We first consider the line in $\mathbb{R}^2$, $b + w_1x_1 + w_2x_2 = 0$, and the plane in $\mathbb{R}^3$, $b + w_1x_1 + w_2x_2 + w_3x_3 = 0$. The line or plane in these cases is just the set of points satisfying the equivalent equation $w \cdot x + b = 0$. We call the extension of this concept to an inner product space a hyperplane, the set of points satisfying the equation $w \cdot x + b = 0$. We see that for a given inner product space, any w and b yield a hyperplane.

A useful formula here is the distance between a point x and a hyperplane, which is

$$d = \frac{w \cdot x + b}{\|w\|}$$

The next geometric figures we consider are the circle in $\mathbb{R}^2$ and the sphere in $\mathbb{R}^3$. Given a centre point c and a radius R, we can describe these figures as the set of points satisfying $d(x, c) = \|x - c\| = R$. In other inner product spaces and $\mathbb{R}^N$, we call this set of points a hypersphere. Sometimes the terminology refers to the points surrounded by the hypersphere, i.e. points satisfying $d(x, c) \leq R$.

The ellipse in $\mathbb{R}^2$ and the ellipsoid in $\mathbb{R}^3$ can be seen as 'squashed' or 'stretched' versions of the disk and sphere. For any two points c_1 and c_2, and a constant R, we can consider the set of points [179, Definition 33] such that

$$d(x, c_1) + d(x, c_2) = 2R \ .$$

This set of points in an inner product space is called an ellipsoid. In the special case where $c_1 = c_2$, the ellipsoid is a hypersphere. If we have a basis

$\nu_1, \cdots, \nu_i, \cdots$ for the inner product space, we can also generalize the ellipsoid in terms of it being a 'squashed' or 'stretched' version of a hypersphere. If we have a hypersphere, we can obtain an ellipsoid from it by scaling each component of all points in the hypersphere, i.e. given scalars $\lambda_1, \cdots, \lambda_i, \cdots$, and a point $x = \sum_{i=1}^{\infty} x_i \nu_i$ in an hypersphere, then $\sum_{i=1}^{\infty} \lambda_i x_i \nu_i$ is the corresponding point on the ellipsoid. Conversely, by scaling with the inverses of the λ_i's we can move from the ellipsoid back to the hypersphere. If we consider scaling the unit hypersphere (the hypersphere with radius 1), each sequence λ defines an unique ellipsoid.

Finally we shall discuss the axis-parallel parallelepiped. The parallelepiped is the high-dimensional generalization of the 2-dimensional parallelogram. The axis-parallel version is then the generalization of rectangles in two dimensions and rectangular prisms in three dimensions, where the sides are parallel to the axes (i.e. parallel to the basis vectors). The rectangular prism[6] can be seen as any point for which each component lies in some closed interval, e.g. if we use the same interval $[0, 1]$ for all three components, the rectangular prism obtained is just the unit cube. This formulation is then easily generalized to high dimensions. For a given basis $\nu_1, \cdots, \nu_i, \cdots$ of an inner product space, we select corresponding intervals $[a_1, b_1], \cdots, [a_i, b_i], \cdots$. These intervals then characterize the axis-parallel parallelepiped consisting of all points $x = \sum_{i=1}^{\infty} x_i \nu_i$ with $x_i \in [a_i, b_i]$ for all i. In all such cases we consider in this thesis, $a_i = -b_i$, so that the parallelepiped is symmetric about the origin.

A.6 Operators

A.6.1 Introductory Concepts

Definition 21. An operator [105, Section 2.6] is a mapping $T : V \to W$, with $V \subset X$, and X and W vector spaces.

Definition 22. A functional [105, Section 2.6] is an operator where $V = X$ and $W = \mathbb{R}$.

[6] Strictly, the interior of a rectangular prism.

Definition 23. A linear operator [105, Section 2.6] is an operator satisfying, for all, $x, y \in V, \alpha \in \mathbb{R}$

- $T(x+y) = T(x) + T(y)$; and
- $T(\alpha x) = \alpha T(x)$.

As an example, the identity operator, $T(x) = x$ is a linear operator, while for $V = \mathbb{R}$, the operators $T(x) = x^2$ and $T(x) = |x|$ are not. These statements are easily by verifying whether T meets the requirements above.

We now consider a more complicated example. We consider the integral operator on $L_\infty(-\infty, \infty)$,

$$T_K(f)(z) = \int_{-\infty}^{\infty} K(z,t) f(t) dt \ ,$$

where K is a continuous function called the *kernel* of the integral operator. From the properties of integrals, we can see that the linearity requirements above hold, so that this operator is linear.

In the special case[7] where $K(z,t) = \theta(z-t)$, the resulting integral operator is the one used to obtain a c.d.f. from a p.d.f..

Definition 24. Let V and W be normed spaces. A linear operator T is bounded [105, Section 2.6] if there is a constant c such that $\|T(x)\| \leq c\|x\|$ for all $x \in V$. We call the smallest such c the norm of T, denoted $\|T\|$.

In the case of the identity operator above, $c = 1$ is sufficient, and also the smallest possible, so that $\|T\| = 1$. We denote the set of bounded linear operators from V into W by $\mathcal{L}(V, W)$.

We turn our attention now to operator equations. An operator equation is an equation of the form $T(f) = F$, where one tries to find f in terms of F. It is the solution of these problems which form the focus of the next section. For the density estimation problem, T is an integral operator, and we obtain an integral equation or *Fredholm equation* [105, Section 5.4],

$$\int_{-\infty}^{\infty} k(z,t) f(t) dt = F(z) \ . \tag{A.4}$$

[7]Note that this K is not continuous, so extra footwork is actually needed here.

A.6.2 Ill-posed Problems and Regularization

Large portions of this section are based on [190, Appendix to Chapter 1].

Ill-posed problems are operator equations which are not considered well-posed by some criterion, traditionally well-definedness in the Tikhonov sense (or, as a special case, in the Hadamard sense). Before presenting these criteria, the concept of stability of the solution of an operator equation must be introduced.

Definition 25. A solution f^*, of an operator equation $Tf = F$ ($T : A \to X$), is said to be *stable* if, for all $\epsilon > 0$, there exists a $\delta(\epsilon)$ such that

$$d(Tf, F) \leq \delta(\epsilon) \Rightarrow d(f, f^*) \leq \epsilon \ .$$

What this means is that a small change in the right hand side of an operator equation will only cause a small change in the solution to the equation, a concept very similar to continuity of an operator.

Definition 26. The problem of solving an operator equation $Tf = F$ is well-posed in the Tikhonov sense on a subset A' of A if, for all $F \in TA'$, the equation has exactly one solution in A', and this solution is stable.

If $A' = A$ and $X = TA$, the problem is said to be well-posed in the Hadamard sense.

Thus, it follows that the problem of solving an operator equation can be ill-posed in the Hadamard sense for three main reasons, viz. if for any $F \in X$:

1. no solution to the problem exists.
2. a solution exists, but it is not unique.
3. the unique solution to the problem is not stable.

It follows that the only possible reason for ill-posedness in the case of an one-to-one operator T is instability of the solutions. Since the operator is one-to-one, it has an inverse T^{-1}, such that $f = T^{-1}F$.

Theorem 27. *If the inverse of an operator $T : A \to X$ exists, and is continuous, the problem of solving the operator equation $Tf = F$ is well-posed in the Hadamard sense.*

Proof. Since the inverse exists, it follows that for all $F \in X$ the equation has exactly one solution f^* in A. All that remains is to show that the solution is stable. Choose an $\epsilon > 0$. By the continuity of T^{-1}, there is a $\delta(\epsilon)$ such that

$$d(F', F) \leq \delta(\epsilon) \Rightarrow d(T^{-1}F', f^*) \leq \epsilon \ .$$

Setting $F' = Tf$ yields the definition of stability, and so the problem is well-posed. □

This theorem also holds for well-posedness in the Tikhonov sense, when A is replaced by A' and X by TA'. A result of this is that by restricting the domain of the operator one can sometimes generate well-posed problems in the Tikhonov sense from problems ill-posed in the Hadamard sense.

These concepts of stability and well-posedness are important because we often wish to solve an operator equation where we only have an approximation of the right hand side (usually some practical observation, subject to noise, measurement error, and random fluctuation). For well-posed problems this is not a problem, and if the inverse exists, we can apply Theorem 27 to obtain a fair approximation of the correct solution.

However, if the inverse operator is not continuous, the solution will typically be unstable. Making use of the concept of *complete continuity* of an operator, we can show that many common operator equations are ill-posed, in particular Fredholm equations (A.4). In this case, we need to employ regularization techniques to obtain stable (also sometimes termed robust) solutions.

One can consider the problem of solving an operator equation $Tf = F$ as the problem of minimizing $\|Tf - F\|$. One can modify the concept of stability to minimizing functionals. Now, although the problem of minimizing $\|Tf - F\|$ remains ill-posed, we consider the problem of rather minimizing

$$R_\lambda(f, F) = \|Tf - F\| + \lambda\Omega(f) \ ,$$

where $\lambda \in \mathbb{R}$ is called a regularization constant, and $\Omega(f)$, the stabilizer or *regularizer*, is a nonnegative functional meeting certain minor technical requirements. This minimization problem is well-posed: if we have a close approximation F' of F, the solution minimizing $R_\lambda(f, F')$ will be close to the solution minimizing $R_\lambda(f, F)$.

Now, the solution minimizing $R_\lambda(f, F)$ is not the same as the solution f^* minimizing $\|Tf - F\|$. We solve this by finding a sequence of solutions of $R_\lambda(f, F')$ converging to f^*. We consider a sequence (F_n) of approximations to F, with $\|F_n - F\| \leq \frac{1}{n}$.

If our choice of $\lambda(n)$ satisfies certain conditions with respect to the accuracy $\frac{1}{n}$ of the approximation F_n[8], the sequence of solutions minimizing $R_{\lambda(n)}(f, F_n)$ converges to f^*.

Most operator equation problems we will consider are set in Hilbert spaces. It can be shown that a suitable regularizer in the case of Hilbert spaces is $\Omega(f) = \|f\|^2$. Thus, if we have a sequence $\lambda(n)$ converging slowly enough to 0, we know that the sequence of solutions minimizing $\|Tf - F\| + \lambda(n)\|f\|^2$ converges to the solution of the operator equation.

Although this theory is elegant, the application is usually less so, due to a number of problems. First, one usually does not have a sequence of approximations to F, just a single best approximation F'. In addition, one does not even know how close F' is to F, and hence what value of λ to use with F'. Assume for the moment we do know an appropriate value of λ. The strategy is then to select the f^*_λ minimizing $R_\lambda(f, F')$. Then f^*_λ is an element of a sequence converging to the solution of the operator equation, and is considered a good estimate of f^* on the basis of this. In practice, the choice of λ is a thorny issue.[9] The only guideline we have is that if we feel F' is a good approximation of F, we should use a small λ, and if we are not as confident, we should use a larger λ. Usually the λ is determined experimentally by cross-validation (i.e. a number of λ's are tried, and the solution f_λ exhibiting the best performance is used).

[8] Simply, $\lambda(n) \to 0$, as $n \to \infty$, but $\lambda(n)n^2$ must not converge to 0. See [190, Theorem A.1].

[9] Note that it is the asymptotic behaviour of the λ_n sequence which determines whether it is acceptable or not. Hence, any choice of λ_n for a fixed n is acceptable. However, the quality of the approximation f^*_λ will depend on the choice of an appropriate λ.

A.7 The Kernel Trick

This section presents Mercer's theorem for Hilbert-Schmidt (HS) operators, and discusses its implication for algorithms making use of inner products, via the kernel trick.

A.7.1 Hilbert-Schmidt Operators and Eigenvalues

Definition 28. Let X be a Hilbert space which is also a measure space with measure μ. We call an operator $T : L_2(X) \to L_2(X)$ a Hilbert-Schmidt (HS) operator [48, Exercise XI.8.44] if there is some function K satisfying

$$\int_{s \in X} \int_{t \in X} |K(s,t)|^2 d\mu(t) d\mu(s) = c < \infty \ , \tag{A.5}$$

so that T is the integral operator with kernel K. This kernel is unique, and thus we can specify T by its kernel, writing T_K. In addition, T_K is bounded, with norm $\sqrt{c}$.

If K is a bounded function, and $\mu(X) < \infty$, the integral in (A.5) is finite, so that the integral operator T_K is a HS operator.

Furthermore, it can be shown that HS operators are compact [48, Theorem XI.6.6]. This is important because it tells us something about the eigenvalues of these operators.

Definition 29. For a linear operator $T : V \to V$, an eigenvalue [105, Section 7.2] of T is a number λ such that the inverse of $T - \lambda I$ does not exist (is not invertible). The set of eigenvalues is also called the *point spectrum*.

It can be shown that $T - \lambda I$ is not invertible if and only if there is a nonzero $x \in V$ such that $(T - \lambda I)(x) = 0$, i.e. $Tx = \lambda x$. Such a v is called an eigenvector corresponding to the eigenvalue λ. In the case where the elements of V are functions, eigenvectors are also called *eigenfunctions*. Note that if x is an eigenfunction corresponding to λ, then so is αx for any scalar α.

Suppose $T : \mathrm{I\!R}^N \to \mathrm{I\!R}^N$ is the operator representing multiplication by the $N \times N$ matrix A. Then λ is an eigenvalue of T (or A) if $A - \lambda I$ is not invertible, i.e. if $\det(A - \lambda I) = 0$. This determinant can then be expanded into a degree N polynomial in λ, called the characteristic equation of A. We thus know that A has at least one (possibly complex) and at most N different eigenvalues.

The following is from Theorem 8.3-1 in [105]:

Theorem 30. *A compact operator has countably many eigenvalues.*

This means that we can consider the sequence λ of eigenvalues of any HS operator.

Definition 31. An operator is self-adjoint when $T(f) \cdot g = f \cdot T(g)$.

Theorem 32. *If K is symmetric (i.e. $K(s,t) = K(t,s)$), the HS operator T_K is self-adjoint.*

Proof.

$$\begin{aligned} T_K(f) \cdot g &= \int_{t \in X} T_K(f)(t) g(t) d\mu(t) \\ &= \int_{t \in X} \int_{s \in X} K(t,s) f(s) d\mu(s) g(t) d\mu(t) \\ &= \int_{t \in X} \int_{s \in X} K(t,s) f(s) g(t) d\mu(s) d\mu(t) \quad . \end{aligned}$$

Now we can replace $K(t,s)$ by $K(s,t)$ from symmetry, and reverse the order of integration. Then $T_K(f)$ equals

$$\begin{aligned} & \int_{s \in X} \int_{t \in X} K(s,t) g(t) f(s) d\mu(t) d\mu(s) \\ = & \int_{s \in X} f(s) \int_{t \in X} K(s,t) g(t) d\mu(t) d\mu(s) \\ = & \int_{s \in X} f(s) T_K(g)(s) d\mu(s) \\ = & f \cdot T_K(g) \quad . \end{aligned}$$

$\square$

This is useful because all the eigenvalues of a self-adjoint operator are real numbers [105, Theorem 9.1-1].

Definition 33. A bounded, self-adjoint linear operator is nonnegative if $T(f) \cdot f \geq 0$ for all f.[10]

All the eigenvalues of a nonnegative operator are nonnegative [105, Problem 9.3.5].

Thus, an HS operator T_K is nonnegative, and hence its eigenvalues are a countable set of nonnegative real numbers, if K is symmetric, and if

$$
\begin{aligned}
T_K(f) \cdot f &= \int_{t \in X} T_K(f)(t) f(t) d\mu(t) \\
&= \int_{t \in X} \int_{s \in X} K(t,s) f(s) f(t) d\mu(s) d\mu(t) \geq 0 \qquad \text{(A.6)}
\end{aligned}
$$

for all f.

Theorem 34 Mercer, 1909. *Let X be an Hilbert space and a measure space with finite measure. Let the bounded, symmetric function K on X^2 be such that the Hilbert-Schmidt operator T_K is nonnegative. Let $\lambda_1, \cdots, \lambda_n, \cdots$ be an enumeration of the nonzero eigenvalues of T_K, with $\psi_1, \cdots, \psi_n, \cdots$ the corresponding eigenfunctions scaled such that $\|\psi_i\| = 1$. Then*

- *$\lambda \in \ell_1$, i.e. $\sum_{i=1}^{\infty} \lambda_i < \infty$.*
- *$\psi_j \in L_\infty(X)$, i.e. ψ_j is a bounded function on X.*
- *$\sum_{j=1}^{\infty} \lambda_j \psi_j(x) \psi_j(y)$ converges to $k(x,y)$ absolutely and uniformly*[11] *for all (x,y).*

We now note that the sum in the last requirement is just a weighted form of the ℓ_2 inner product of $\psi(x)$ and $\psi(y)$, and that this weighted form $(\langle x, y\rangle_{\ell_{2,\lambda}} = \sum_{j=1}^{\infty} \lambda_j x_j y_j)$ is also a valid inner product for ℓ_2 (since $\lambda > 0$). Suppose, in the above case, we consider the resulting mapping $\psi : X \to \ell_2$. Then this theorem tells us we can calculate $\langle \psi(x), \psi(y)\rangle_{\ell_{2,\lambda}}$ for any

[10] Such operators are sometimes called positive.

[11] For more on uniform convergence, see Section A.10.1. A sequence is absolutely convergent if its corresponding sequence of norms converges.

$x, y \in X$ simply as $K(x, y)$. Given an input space X, a kernel satisfying the requirements of Mercer's theorem is called a *Mercer kernel.*

In many cases, the number of nonzero eigenvalues of T_K is finite, say m. In this case, the mapping ψ maps into ℓ_2^m, but the essential idea stays the same.

A.7.2 Cover's Theorem

Consider an algorithm F, which takes a collection of vectors $x_1, \ldots, x_l \in X$ as input. Suppose F can be formulated in such a way that the inputs x_i are only used in the algorithm in inner products. Suppose we wish to generalize this algorithm. One approach would be transform the vectors into a different space, and then performing the algorithm on the transformed vectors. For example, if we wanted to generalize the standard linear regression algorithm, taking inputs $(x, y) \in \mathbb{R}^2$, we could consider the transformation

$$\varphi\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} x \\ y \\ x^2 \\ y^2 \\ xy \end{bmatrix},$$

and then apply linear regression on the resulting vectors in $\mathbb{R}^5$. Note that if we have an algorithm utilizing inner products in a space, we are essentially working with linear functions in that space. Thus the resulting regression function, although linear in $\mathbb{R}^5$, would be nonlinear in the original input vector (x, y). Similarly, originally linear classification algorithms generalized using this technique will typically obtain nonlinear decision surfaces in the original space.

What is the motivation of generalizing algorithms in this manner? The justification for classification can be attributed to a theorem shown by Thomas Cover in 1965, which essentially states [76, Section 5.2] that transforming a low-dimensional input space nonlinearly to a high-dimensional *feature space* generally improves the probability of the resulting problem being linearly separable.[12]

There are at first sight a number of problems with this proposal. First, utilizing high-dimensional spaces lends itself easily to overfitting. Secondly, in higher-dimensional space, data is necessarily sparse. Thirdly, transforming the data and calculating the inner product can be prohibitive or even impossible in very high- and infinite-dimensional feature spaces.

The second problem is not as serious as it seems at first. If one looks closely, although extra features are being calculated, we are only considering a subspace of $\mathbb{R}^5$: those points where the third component is the square of the first component, the fourth component the square of the second, and the final component the product of the first two. When one considers the available data in this subspace, it is just as dense as in the original problem.

The first problem is linked to the idea of fitting a polynomial of degree $p-1$ through p points: by using enough features, any problem can be made linearly separable. This is indeed a problem, and one should keep it in mind.

The third problem is the focus of the kernel trick: how can we calculate the inner products after transformation feasibly for high- and infinite-dimensional feature spaces. In this light, the discussion after Theorem 34 may make more sense.

A.7.3 Using The Kernel Trick

Although the mathematics necessary to apply what is now commonly known as the kernel trick was already available in the mid-twentieth century, the technique of using the kernel trick to generalize algorithms involving inner products on the data only, and not using the data in any other way, only became popular with the advent and subsequent success of SVMs. Subse-

[12] More technically, Cover's separability theorem states that the probability that N data points drawn independently from the same distribution, and each labelled -1 or 1 randomly with equal probability, can be seperated linearly after a transformation is $\left(\frac{1}{2}\right)^{N-1}\sum_{m=0}^{m_1-1}\begin{pmatrix} N-1 \\ m \end{pmatrix}$. Here m_1 is the number of degrees of freedom of separating hyperplanes in the feature space corresponding to the transformation. Note that this is a cumulative binomial probability, and approaches 1 as m_1 approaches N.

quent uses of the technique were Kernel PCA [155], Kernel SOMs [116], and many others.

We know from Section A.7.2 that an algorithm using inner products can be generalized by replacing the inner product function by calculating the inner product on a transformation of the input space. Consider the transformation $\varphi : X \to Y$, with Y a Hilbert space. Suppose we have a function K such that

$$k(x, y) = \varphi(x) \cdot \varphi(y) \quad .$$

We can then use this function to transform and calculate inner products in one step. This shortcut for transforming two points from an input space X into a higher-dimensional feature space (ℓ_2) and then calculating the inner product in feature space, simply by evaluating k on the original points, is called the *kernel trick*.[13]

A function K with which one can perform the kernel trick is called an admissible kernel. Kernels satisfying the requirements of Mercer's Theorem 34 are called Mercer kernels. It is clear that Mercer kernels are admissible.

Usually the space X used is $\mathbb{R}^N$. Now, with the usual Lebesgue measure, $\mathbb{R}^N$ is not a finite measure space. This can be circumvented by using the Lebesgue measure on a sufficiently large subset (large enough to handle any reasonable input to the algorithm in question), and setting the measure of the rest of $\mathbb{R}^N$ to 0. Mercer's theorem is stated from this perspective in [26, Appendix 2]. A discussion of various Mercer kernels is given in Section 4.3.

A useful concept when discussing kernels is the Gram matrix:

Definition 35. The Gram matrix of a symmetric kernel K on a set of points $x_1, \cdots, x_m$ is the $m \times m$ matrix G with $G_{ij} = K(x_i, x_j)$.

Note that if the input space X is a finite set of points $\{v_1, \cdots, v_u\}$, the requirement in (A.6 for positivity of the HS operator becomes[14]

$$\sum_{i=1}^{u} \sum_{j=1}^{u} K(v_i, v_j) f(v_i) f(v_j) \geq 0$$

[13] For an example of applying the kernel trick with a simple kernel, see [24, Sect. 4].

[14] Assuming an uniform measure on the elements.

for all f. If we set $x_i = f(v_i)$, this means that for all vectors x, we need $x^T G x \geq 0$. A matrix A satisfying $x^T A x \geq 0$ for all x is said to be positive semi-definite. In order to apply Mercer's theorem to a kernel function K on a finite space, we thus need K to be symmetric and its associated Gram matrix on the input space to be positive semi-definite. We call such a kernel a positive definite kernel.

Some algorithms (such as the nearest neighbour classification algorithm) only make use of the distances between points, instead of inner products between points in general. This means that we may be able to generalize these algorithms to use the kernel trick with less requirements on the kernel function. This is because the distance between two points x and y is simply $\|x-y\| = (x-y)\cdot(x-y)$, so that our kernel function will only have to represent inner products when both components are equal. This approach leads to the wider class of conditionally positive definite kernels. The interested reader is referred to [144].

A.8 Capacity

This section is devoted to introducing the concept of capacity of a class of functions, and various techniques used for measuring capacity.

A.8.1 Definitions

We consider a class $\mathcal{F}$ of real-valued functions mapping X into $\mathbb{R}$. For a finite set of points $S = \{v_1, \cdots, v_u\}$ in X, we are interested in a measure of closeness between elements of $\mathcal{F}$ evaluated on the points in S. A possible measure of this type is defining for $f, g \in \mathcal{F}$

$$d_{1,S}(f,g) = \sum_{i=1}^{u} |f(v_i) - g(v_i)| \ ,$$

the sum of distances between $f(v_i)$ and $g(v_i)$. Thus we implicitly consider the distance between the vectors $f(S) = (f(v_1), \cdots, f(v_u))$ and $g(S) = (g(v_1), \cdots, g(v_u))$ as points in ℓ_1^u.

We can consider the vectors $f(S)$ and $g(S)$ as points in ℓ_p^u for other values of p as well, yielding alternative metrics $d_{p,S}$. In particular, we are interested in the case $p = \infty$. Then we define

$$d_S(f,g) = d_{\infty,S}(f,g) = \sup_{i=1}^{u} |f(v_i) - g(v_i)| \ ,$$

the maximum deviation between f and g on S.

Now,we consider a class $\mathcal{F}$ of real vector-valued functions with k components. For this case the previous argument will still work if we replace $|f(v_i)-g(v_i)|$ by $\|f(v_i) - g(v_i)\|$ throughout. Of course, the resulting metric depends on the norm used in $\|f(v_i) - g(v_i)\|$. As before, we can treat the vectors $f(v_i)$ and $g(v_i)$ as points in ℓ_q^k, giving different norms $d_{q,p,S}$ for different choices of q. Once again, we are interested in the case $q = \infty$, yielding the metric

$$\begin{aligned} d_S(f,g) &= d_{\infty,\infty,S}(f,g) \\ &= \sup_{i=1}^{u} \|f(v_i) - g(v_i)\|_\infty \\ &= \sup_{i=1}^{u} \sup_{j=1}^{k} |f_j(v_i) - g_j(v_i)| \ , \end{aligned}$$

where f_j and g_j denote the j-th components of f and g. This is then the maximum deviation between a component of f and g over the points in S.

Definition 36. An ϵ-net or ϵ-cover of a function class $\mathcal{H}$ with respect to a metric d is a set of functions P_ϵ such that for any $h \in \mathcal{H}$, there is a $p(h) \in P_\epsilon$ with $d(h, p(h)) \leq \epsilon$.

Definition 37. The ϵ-covering number of a function class $\mathcal{H}$ with respect to a metric d is the minimal cardinality[15] of an ϵ-net of $\mathcal{H}$ with respect to d. This covering number is denoted by $\mathcal{N}(\epsilon, \mathcal{H}, d)$.

Definition 38. The nth entropy number [203, Section 2] of a function class $\mathcal{H}$, with respect to a metric d is defined by

$$\epsilon_n(\mathcal{H}, d) = \inf\{\epsilon > 0 : \mathcal{N}(\epsilon, \mathcal{H}, d) \leq n\} \ .$$

In other words, the nth entropy number is the finest resolution of a cover with n or fewer elements.

[15]The cardinality of a set is its number of elements.

Definition 39. The nth entropy number of a bounded linear operator T is the nth entropy number of the image of the unit ball, i.e.

$$\epsilon_n(T) = \epsilon_n(T(U), d) \ ,$$

where U is the unit ball[16] in the domain of T, and d is the metric employed in the range of T.

Definition 40. The ϵ-entropy of a function class $\mathcal{H}$ on samples of length l is the expectation, over all samples s of length l, of the ϵ-covering number of $\mathcal{H}$ with respect to d_s,

$$H(\epsilon, \mathcal{H}, l) = \mathbb{E}_{s \in X^l} \mathcal{N}(\epsilon, \mathcal{H}, d_s) \ .$$

Definition 41. The ϵ-growth function[17] of a function class $\mathcal{H}$, $\mathcal{N}(\epsilon, \mathcal{H}, l)$ is defined as

$$\sup_{s \in X^l} \mathcal{N}(\epsilon, \mathcal{H}, d_s) \ .$$

Thus the growth function returns the maximal ϵ-covering number of a l-sample s.

We note that $H(\epsilon, \mathcal{H}, l) \leq \mathcal{N}(\epsilon, \mathcal{H}, l)$—the ϵ-entropy is bounded by the ϵ-growth function. In the case of classification, the covering numbers are independent of $\epsilon < 1$, so the ϵ is usually omitted from the notation.

A.8.2 Useful Results

The following are results used in the bound derived in the thesis.

Theorem 42. *Given a set of functions $\mathcal{F}$, a finite set of points X. For any $\epsilon \geq 0$, and any sample s from X, an ϵ-cover of $\mathcal{F}$ with respect to d_X also covers $\mathcal{F}$ with respect to d_s.*

Proof. Let P_ϵ be an ϵ-cover of $\mathcal{F}$ with respect to d_X. Then, for any $f \in \mathcal{F}$, there is a $p(f) \in P_\epsilon$ such that $d_X(f, p(f)) \leq \epsilon$. That is,

$$\max_{x \in X} \{|f(x) - p(f)(x)|\} \leq \epsilon \ .$$

[16] A hypersphere of radius 1, centred at 0.

[17] We use the definition by Anthony in [6]. Vapnik [191] defines the growth function as the natural logarithm of this value.

Since the points of the sample s are all in X, $d_s(f, p(f)) \leq \epsilon$, and P_ϵ is an ϵ-cover of $\mathcal{F}$ with respect to d_s. $\square$

Theorem 43. *Given a bounded linear operator T, $\epsilon_1(T) \leq \|T\|$.*

Proof. Although equality holds [203, Section 2], we only use the inequality above, which is easily proved:

$$\begin{aligned} \epsilon_1(T) &= \epsilon_1(T(U)) \\ &= \inf\{\epsilon > 0 : \mathcal{N}(\epsilon, T(U), d) \leq 1\} \ . \end{aligned}$$

Thus $\epsilon_1(T)$ is the radius of the smallest ball containing $T(U)$. We know that a ball of radius $\|T\|$ about the origin does this, from the definition of the norm of T, and the result follows. $\square$

Theorem 44 [203]. *Let $\sigma = (\sigma_1, \cdots, \sigma_j, \cdots)$ be a nonincreasing sequence of nonnegative numbers. Then, for all $n \in \mathbb{N}$, and for $D : \ell_p \to \ell_p$, where*

$$D(x) = (\sigma_1 x_1, \ldots, \sigma_j x_j, \ldots) \ ,$$

we have that

$$\sup_{j \in \mathbb{N}} n^{-\frac{1}{j}} \left(\prod_{i=1}^{j} \sigma_i\right)^{-\frac{1}{j}} \leq \epsilon_n(D) \leq 6 \sup_{j \in \mathbb{N}} n^{-\frac{1}{j}} \left(\prod_{i=1}^{j} \sigma_i\right)^{-\frac{1}{j}} \ .$$

Theorem 45 [203]. *Let S be a bounded linear operator from a Hilbert Space into ℓ_∞^m. There is a constant $0 < p \leq 103$ such that for all $m \in \mathbb{N}$, and $1 \leq n \leq 2^{m-1}$,*

$$\epsilon_n(S) \leq min \left(\|S\|, p\|S\| \sqrt{\frac{\log(1 + \frac{m}{\log n + 1})}{\log n + 1}} \right) \ .$$

It is hypothesised that the value of p is close to 1.86 [203, Theorem 8].

Theorem 46 [203]. *Let E, F, G be Banach spaces, with $R \in \mathcal{L}(F, G)$ and $S \in \mathcal{L}(E, F)$. Then, for $n, t \in \mathbb{N}$,*

$$\epsilon_{nt}(RS) \leq \epsilon_n(R)\epsilon_t(S)$$

Since $\epsilon_1(T) \leq \|T\|$, it follows that

$$\epsilon_n(RS) \leq \epsilon_n(R)\|S\|$$

and

$$\epsilon_n(RS) \leq \|R\|\epsilon_n(S) \quad .$$

Theorem 47. *For $c > 0$, and a finite set S,*

$$\mathcal{N}\left(\epsilon, c\mathcal{F}, d_S\right) = \mathcal{N}\left(\frac{\epsilon}{c}, \mathcal{F}, d_S\right) \quad .$$

Proof. Let P be an $\frac{\epsilon}{c}$-cover of $\mathcal{F}$ with respect to d_S. We show that cP is an ϵ-cover of $c\mathcal{F}$ with respect to d_S: For any $f' \in c\mathcal{F}$, there is a $f \in \mathcal{F}$ with $f' = cf$. Associated with this f is a $p(f) \in P$ such that $d_S(f, p(f)) \leq \frac{\epsilon}{c}$. Thus $d_S(cf, cp(f)) = cd_S(f', p(f)) \leq \epsilon$. □

Theorem 48. *$\epsilon \leq \epsilon'$ implies*

$$\mathcal{N}\left(\epsilon, \mathcal{F}, d\right) \geq \mathcal{N}\left(\epsilon', \mathcal{F}, d\right)$$

for any function class $\mathcal{F}$.

Proof. This follows because any ϵ-cover of $\mathcal{F}$ is also an ϵ'-cover of $\mathcal{F}$. □

Theorem 49. *Let P be an ϵ-cover of a function class $\mathcal{F} : X \to Y$ with respect to the metric d_A, for finite $A \subseteq X$. Then $\pi_\gamma(P)$ is an ϵ- cover of $\pi_\gamma(\mathcal{F})$ with respect to d_A.*

Proof. Consider any $g \in \pi_\gamma(\mathcal{F})$. Then there is some $f \in \mathcal{F}$ such that $g = \pi_\gamma(f)$. Now, there is some $p(f) \in P$ such that $d_A(f, p(f)) \leq \epsilon$. Thus $d_A(g, \pi_\gamma(p(f))) \leq \epsilon$: if if were not so, there would be some $x \in A$ such that $d(\pi_\gamma(f)(x), \pi_\gamma(p(f))(x)) > \epsilon$. It would then follow that $d(f(x), p(f)(x)) > \epsilon$, contrary to the choice of p. □

This theorem implies that $\mathcal{N}(\epsilon, \pi_\gamma(\mathcal{F}), d_{l_\infty(A)}) \leq \mathcal{N}(\epsilon, \mathcal{F}, d_{l_\infty(A)})$.

A.8.3 Dimension-like Capacity Measures

Various dimension-like capacity measures have been proposed. In this section, we will concentrate on the two most popular, the VC-dimension, and the fat-shattering function.

The VC-dimension was originally proposed as a measure of capacity for sets of indicator functions, and was thus useful for bounding classification error. In the case of indicator functions, using a 0/1-encoding, and the d_S norm, the covering numbers are independent of the ϵ used[18]. In this case, since the indicator functions output a 0 or 1 for each element of S, the covering number turns out to be the number of different label assignments to the elements of S.[19] Since there are only $2^{|S|}$ possible labellings of S, we see that the covering number, and thus also the growth function, is bounded above by $2^{|S|}$. The VC-dimension of a function class $\mathcal{F}$ is then the length $\mathrm{VC}(\mathcal{F})$ of the longest sample which can be labelled in all $2^{\mathrm{VC}(\mathcal{F})}$ possible ways by elements of $\mathcal{F}$.[20]

We see that for $l \leq \mathrm{VC}(\mathcal{F})$,

$$\mathcal{N}(\mathcal{F}, l) = 2^l \quad ,$$

and for $l > \mathrm{VC}(\mathcal{F})$,

$$\mathcal{N}(\mathcal{F}, l) < 2^l \quad .$$

The second inequality can be strengthened further, showing that the growth function only grows polynomially when sample lengths exceed the VC-dimension. This stronger result, derived independently by Sauer, by Shelah and by Vapnik and Chervonenkis, is sometimes called the VCSS lemma, but is usually known as Sauer's lemma:

Theorem 50 Sauer's Lemma. *Consider a set $\mathcal{F}$ of indicator functions. Then*

$$\mathcal{N}(\mathcal{F}, l) \leq \sum_{i=0}^{\mathrm{VC}(\mathcal{F})} \binom{l}{i}$$

[18] As long as ϵ is smaller than $\frac{1}{2}$.

[19] Every function outputs a labelling of S, and the elements of the cover are also labellings of S.

[20] Note that the VC-dimension can be infinite.

This result may not seem useful at first, but its utility becomes apparent when used in conjunction with the following result by Blumer et al [19, Proposition A.2.1]:

Theorem 51. *For $l > d \geq 1$,*

$$\sum_{i=0}^{d} \binom{l}{i} \leq 2\frac{l^d}{d!} \leq \left(\frac{el}{d}\right)^d .$$

When $\mathcal{N}(\mathcal{F}, d_S) = 2^l$, we say that $\mathcal{F}$ *shatters* the points of S. With this terminology, the VC-dimension of $\mathcal{F}$ is the length of the largest sample shattered by $\mathcal{F}$ [191]. The motivation for the definition of VC-dimension is then that a function class which can only shatter small samples has less capacity than one which is able to shatter larger samples. This concept is generalized with the fat-shattering function.

Suppose we no longer consider binary functions for classification, but rather use real-valued functions, where the classification is obtained by comparing the output to some threshold value. We will usually be more confident in our classification the more the output of the classification function differs from the threshold. The fat-shattering function measures the capacity of the function class to *achieve a given margin*.

Definition 52. Let $\gamma \geq 0$. We say a function class $\mathcal{F}$ γ-shatters a l-sample s relative to a vector r if for each binary l-vector b, there is a $f_b \in \mathcal{F}$ such that

$$f_b(s_i) \begin{cases} > & r_i + \gamma \quad \text{if } b_i = 1 \\ \leq & r_i - \gamma \quad \text{if } b_i = 0 \end{cases}$$

From this viewpoint, shattering refers to 0-shattering relative to the origin. Clearly, the larger γ is, the more difficult it will be for a function class to γ-shatter a sample, i.e. its capacity to γ-shatter decreases. γ is sometimes referred to as the *width of shattering.* We now extend the VC-dimension concept to include the width of shattering, yielding the fat-shattering function [201, Definition 2.1].

Definition 53. The fat-shattering function of a function class $\mathcal{F}$ is a function mapping a real number γ to the size of the largest sample which can be

γ-shattered relative to some vector. We call the value of the fat-shattering function of $\mathcal{F}$ at γ the fat-shattering dimension of $\mathcal{F}$ at *scale* γ.

The VC-dimension and fat-shattering function can be used for measuring capacity for regression functions by considering regression estimation as a classification problem using the ε-insensitive loss.

A.9 Statistical Concepts

This section deals with a few well-known statistical concepts, laws and distributions.

Theorem 54. *Consider a probability space (X, S, μ). Let $A_1, \cdots, A_n$ be elements of S. Then*

$$\mu(\bigcup_{i=1}^{n} A_i) \leq \sum_{i=1}^{n} \mu(A_i) \ .$$

In other words, the probability of at least one event in a set occurring is not more than the sum of the probabilities of each event occurring.

Definition 55. Consider a probability space defined on $\mathbb{R}$, with a p.d.f. f. Then the c.d.f. F is defined by

$$F(x) = \int_{-\infty}^{x} f(z)dz = \int_{-\infty}^{x} 1df(z) \ . \tag{A.7}$$

This means that $F(x)$ is the probability a random variable corresponding to the probability space does not exceed x.

This definition can be extended to spaces defined on $\mathbb{R}^N$.

The Gaussian (or normal) distribution is a commonly encountered family of probability density functions on $\mathbb{R}$, and is defined by

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \ , \tag{A.8}$$

where μ and σ are parameters specifying which member of the family.

Definition 56. The expected value of a real-valued function g of a continuous random variable X' associated with a probability space (X, S, μ) is

$$\mathbb{E}g(X') = \int_X g(x)f(x)dx \quad ,$$

where $f(x)$ is the associated p.d.f..

When X' is discrete, and p is the probability mass function, the formula is

$$\mathbb{E}g(X') = \sum_{x \in X} g(x)p(x) \quad .$$

In the case where $g(x) = x$, we call the expected value of $g(X')$ the mean of X'. For X' distributed Gaussian, $\mathbb{E}X' = \mu$ is the mean.

Using the concept of the expected value, we can define variance:

Definition 57. The variance of a function g of a variable X' associated with a probability space (X, S, μ) is

$$\text{var}g(X') = (\mathbb{E}(g(X') - \mathbb{E}g(X'))^2 \quad ,$$

the expected squared deviation of $g(X')$ from its expected value.

The variance of $g(x) = x$ is then the expected squared deviation of X' about its mean. For X' distributed Gaussian, $\text{var}X' = \sigma^2$.

A.9.1 Density Estimation and Estimators

In many cases, one has access to realizations of a random variable, but does not know the probability measure of the underlying probability space. The process of using these realizations to gather information about the underlying measure is called *density estimation.* This is because the process tries to identify the p.d.f., which characterizes the measure.

There are two major approaches to density estimation: parametric and non-parametric. Parametric approaches assume the p.d.f. comes from a family of such functions characterized by a set of parameters, and then focuses on estimating these parameters. An example of this is assuming the p.d.f. comes from the Gaussian family (A.8), and then trying to estimate μ and

σ. Nonparametric approaches do not make such restrictive prior assumptions on the form of the p.d.f., attempting to obtain the p.d.f. under as few assumptions as possible.

Well known techniques for obtaining the parameters for parametric density estimation are the ML approach and Bayesian inference. Common techniques for nonparametric estimation are kernel density estimation and Parzen windows. A good introduction to both parametric and nonparametric density estimation is Chapters 3 and 4 of the classic book by Duda and Hart [46].

When one does not have access to the measure underlying a random variable, one can often use functions of a sample of realizations of the random variable to estimate properties of the random variable, notably the mean and variance (such as using data to estimate μ and σ^2 for a Gaussian distribution). Such functions are called *estimators*.

We can consider the bias and variance of estimators as measures of their quality. Suppose we want to use an estimator $g(s)$ (with s a sample) to estimate some property $p(X')$. The variance of the estimator is just the variance of the function g. The bias of the estimator g is just $\mathbb{E}_{s\in X^l} g(s) - p(X')$, a seemingly sensible measure of quality of the estimator g. We say an estimator is unbiased if its bias is 0, biased upwards if the bias is positive, and biased downwards if the bias is negative.

An example of an unbiased estimator is using a sample average as an estimator for the mean μ. Here,

$$\begin{aligned}
\mathbb{E}\frac{1}{l}\sum_{i=1}^{l} s_i &= \frac{1}{l}\sum_{i=1}^{l}\mathbb{E}s_i \\
&= \frac{1}{m}\sum_{i=1}^{m}\mu \\
&= \mu \ ,
\end{aligned}$$

where μ is the common mean of all the realizations of the random variable.

A.10 Bounding Risk

A.10.1 Modes of Convergence

Informally speaking, a sequence of elements of a set converges if the elements of the sequence become closer and closer to a fixed point in the set. However, when trying to make this concept formal, things can get a little tricky, particularly when we are working with sets of functions, or with random variables. This section outlines the modes of convergence mentioned in this thesis. Familiarity with limits is assumed.[21]

Definition 58. Let (X, d) be a metric space. Suppose $x = x_1, \cdots, x_n, \cdots$ is a sequence in X, and $y \in X$. We say x converges to y if $\lim_{n\to\infty} d(x_i, y) = 0$, and write $x \longrightarrow y$.

Convergence of functions in the normed space L_∞ is called *uniform convergence.* When we consider sequences of random variables, things get a little more complicated:

Definition 59. Let (X, d) be a metric space. Let $x = x_1, \cdots, x_n, \cdots$ be a sequence of random variables, with x_i associated with the probability space (X, S_i, μ_i). Let y be a random variable associated with (X, S_y, μ_y). We say the random sequence x converges to the random variable y with probability t if the probability of convergence of a realization of the sequence x to a realization of y is t, i.e.

$$P(x \longrightarrow y) = t \ .$$

If $t = 1$, we also say that x converges to y *almost surely* or *almost everywhere*, and write $x \longrightarrow_{ae} y$. Convergence with probability 1 is also sometimes called *strong* convergence.

Definition 60. Let (X, d) be a metric space. Let $x = x_1, \cdots, x_n, \cdots$ be a sequence of random variables, with x_i associated with the probability space (X, S_i, μ_i). Let y be a random variable associated with (X, S_y, μ_y). We say the random sequence x converges to the random variable y in probability $(x \longrightarrow_p y)$ if, for any $\delta > 0$,

$$\lim_{n\to\infty} P(d(x_n, y) > \delta) = 0 \ .$$

[21] See any introductory calculus textbook, e.g. [179].

This is also known as weak convergence.

These formulations may at first seem similar, but there is an important difference. If we consider the original definition of convergence, we note that both modes of convergence are similar to that definition, with a modification for probability (since we are dealing with a sequence of random variables). From this viewpoint, we see that the difference arises in where the probability is taken. For weak convergence, we consider the limit of the probability, while for strong convergence, we consider the probability of the limit. It should also be noted that saying something holds with probability 1 is not equivalent to saying it holds for all elements of X. It says that the probability of a random variable attaining a value for which it does not hold, is 0. Thus, there may be points for which it does not hold, but there is no chance that the random variable will take these values.

When we consider convergence of functions of random variables, we obtain so-called uniform convergence results. Essentially we can then consider weak or strong uniform convergence.

For example, the Glivenko-Cantelli theorem [191, Section 2.11], states that the sequence of sample c.d.f.'s of a random variable $F_l(z) = \frac{1}{l}\sum_{i=1}^{l}\theta(z-x_i)$ (where the sample random variables are $x_1, \cdots, x_l$) as the sample length increases, exhibits weak uniform convergence to the underlying c.d.f $F(z)$, i.e.

$$(F_l)\ l \overrightarrow{\to \infty}_p\ F \ ,$$

or, more explicitly,

$$\lim_{l\to\infty} P(\|F_l(z) - F(z)\|_\infty > \delta) = 0 \ .$$

If we further expand the L_∞ norm, we see the theorem shows that

$$\lim_{l\to\infty} P(\sup_z |F_l(z) - F(z)| > \delta) = 0 \ .$$

A.10.2 Hoeffding's Theorem

Central to our proof will be the following result by Wassily Hoeffding [82]:

Theorem 61. *Let $\tau_1, \cdots, \tau_m$, $\tau_i \in [a_i, a_i + b_i]$, be a sequence of independent, identically distributed bounded random variables. Denote their average by S_m. Then, for any $\epsilon \geq 0$,*

$$Pr\left\{\mathbb{E}(S_m) - S_m \geq \epsilon\right\} \leq \exp\left(-\frac{2m^2\epsilon^2}{\sum_{i=1}^{m} b_i^2}\right) \quad .$$

A.10.3 Risk Definitions

In this section, we assume the inputs come from a finite set S, the outputs are in a set Y, and $Z = S \times Y$. Given an hypothesis $h \in \mathcal{H}$, we define the expected risk and empirical margin risk for h, adapted from the risks in [67], for each learning task:

Definition 62 Expected Risk. The *expected risk* $R(h)$ of h is:

- For dichotomous classification, the probability that $h(x)\,y \leq 0$ for a random pair (x, y) chosen according to $P(x, y)$, i.e.

$$R(h) = \sum_{(x,y)\in Z} \theta\left(h(x)\,y \leq 0\right) P(x, y) \quad .$$

  Note that this assumes that $h(x) = 0$ always contributes to the expected risk.

- For polychotomous classification, the probability that $h_k(x)\,y_k \leq 0$ for some $k \in Y$ for a random pair (x, y) chosen according to $P(x, y)$, i.e.

$$R(h) = \sum_{(x,y)\in Z} \theta\left(\min_k\left(h_k(x)\,y_k\right) \leq 0\right) P(x, y) \quad .$$

- For regression, the probability that $h(x)$ deviates from y by more than ε for a random pair (x, y) chosen according to $P(x, y)$, i.e.

$$R(h) = \int_{(x,y)\in Z} \theta\left(|h(x) - y| > \varepsilon\right) dP(x, y) \quad .$$

Note that the definition for regression corresponds to the use of an ε-insensitive loss function, and that integration is necessary here since Y is infinite in this case.

Definition 63 Empirical Margin Risk. The *empirical margin risk with margin* $\gamma \geq 0$, $R_s^\gamma(h)$, of h over a sample $s = \{(x_i, y_i) \in Z : i \in \{1, \cdots, m\}\}$ is:

- For dichotomous classification, the proportion of examples in the sample for which $h(x_i)\, y_i < \gamma$, i.e.

$$R_s^\gamma(h) = \frac{1}{m} \sum_{i=1}^{m} \theta\left(h(x_i)\, y_i < \gamma\right) \quad .$$

- For polychotomous classification, the proportion of examples in the sample for which $h_k(x_i)\, y_{ik} \leq \gamma$ for some $k \in Y$, i.e.

$$R_s^\gamma(h) = \frac{1}{m} \sum_{i=1}^{m} \theta\left(\min_k \left(h_k(x_i)\, y_{ik}\right) \leq \gamma\right) \quad .$$

- For regression, the proportion of examples in the sample for which $h(x_i)$ deviates from y_i by more than $\varepsilon - \gamma$, i.e.

$$R_s^\gamma(h) = \frac{1}{m} \sum_{i=1}^{m} \theta\left(|h(x_i) - y_i| > \varepsilon - \gamma\right) \quad .$$

  We also require $\varepsilon > \gamma$ in this case.

Making use of the squashing operator, we can define two risk-like functions for dichotomous classification, which are often useful as bounds on the risks above in derivations (once again derived from [67]):

Definition 64. For $\gamma \geq 0$,

$$\mathcal{R}^\gamma(h) = \sum_{(x,y) \in Z} \theta(|\pi_\gamma(h(x)) - \gamma y| \geq \gamma) P(x, y) \quad .$$

Definition 65. For $\gamma \geq 0$ and a sample $s = \{(x_i, y_i) \in Z : i \in \{1, \cdots, m\}\}$,

$$\mathcal{R}_s^\gamma(h) = \frac{1}{m} \sum_{i=1}^{m} \theta(|\pi_\gamma(h(x_i)) - \gamma y_i| \geq \gamma) \quad .$$

A.10.4 Traditional Approach to Bounds

In this section we sketch a common approach to bounding generalization error in the case of dichotomous classification. Suppose $(x, y) \in Z$ contributes to $R(h)$ (i.e. is such that $h(x)y \leq 0$). Then, for $\gamma > 0$, $\pi_\gamma(h(x))y \leq 0$. Thus, for $\gamma \geq 0$,

$$|\pi_\gamma(h(x)) - \gamma y| = |\pi_\gamma(h(x))| + \gamma \geq \gamma \ ,$$

so that (x, y) also contributes to $\mathcal{R}^\gamma(h)$, implying that

$$R(h) \leq \mathcal{R}^\gamma(h) \ . \tag{A.9}$$

We wish to bound, for some $\gamma \geq 0$,

$$P_{s\in Z^m}\left(\sup_{h\in\mathcal{H}}(R(h) - R_s^\gamma(h)) \geq \epsilon\right) \ . \tag{A.10}$$

From (A.9), for all $\gamma \geq 0$,

$$\sup_{h\in\mathcal{H}}(\mathcal{R}^\gamma(h) - R_s^\gamma(h)) \geq \sup_{h\in\mathcal{H}}(R(h) - R_s^\gamma(h)) \ .$$

Together with (A.10), this yields

$$P_{s\in Z^m}\left(\sup_{h\in\mathcal{H}}(R(h) - R_s^\gamma(h)) \geq \epsilon\right) \leq P_{s\in Z^m}\left(\sup_{h\in\mathcal{H}}(\mathcal{R}^\gamma(h) - R_s^\gamma(h)) \geq \epsilon\right)$$

where the probability is with respect to possible samples of size m.[22] This is a generalization of a result presented in [69]:

Theorem 66. *For all $\gamma \geq 0$, and all $m, l \in \mathbb{N}$,*

$$\begin{aligned} & P_{s\in Z^m}\left(\sup_{h\in\mathcal{H}}(\mathcal{R}^\gamma(h) - R_s^\gamma(h)) \geq \epsilon\right) \\ \leq \ & 2P_{s\in Z^m, t\in Z^l}\left(\sup_{h\in\mathcal{H}}(\mathcal{R}_t^\gamma(h) - R_s^\gamma(h)) \geq \epsilon - \frac{1}{l}\right) \ , \end{aligned}$$

where the second probability is with respect to all pairs of samples of lengths m and l.

[22] Since the sample is drawn independently according to $P(x, y)$, the distribution is derived from the product of the possibilities of drawing each element in the sample.

Proof.

$$\begin{aligned}
& P_{s\in Z^m, t\in Z^l}\left(\sup_{h\in\mathcal{H}}(\mathcal{R}_t^\gamma(h) - R_s^\gamma(h)) \geq \epsilon - \frac{1}{l}\right) \\
= & \sum_{(s,t)\in Z^{m+l}} \theta\left[\sup_{h\in\mathcal{H}}(\mathcal{R}_t^\gamma(h) - R_s^\gamma(h)) - \epsilon + \frac{1}{l}\right] P(s,t) \ , \qquad \text{(A.11)}
\end{aligned}$$

where $P(s,t)$ is the probability of obtaining the samples s and t.

Since $P(s,t) = P(s)P(t)$, we can rewrite (A.11) as

$$\sum_{s\in Z^m} P(s)\left[\sum_{t\in Z^l} P(t)\theta[\sup_{h\in\mathcal{H}}(\mathcal{R}_t^\gamma(h) - R_s^\gamma(h)) - \epsilon + \frac{1}{l}]\right] \ . \qquad \text{(A.12)}$$

Now, we restrict the first sum over

$$Q = \left\{s \in Z^m : \sup_{h\in\mathcal{H}}(\mathcal{R}^\gamma(h) - R_s^\gamma(h)) \geq \epsilon\right\} \ .$$

Then (A.12) is greater than or equal to

$$\sum_{s\in Q} P(s)I(s) \ , \qquad \text{(A.13)}$$

with

$$I(s) = \sum_{t\in Z^l} P(t)\theta\left[\sup_{h\in\mathcal{H}}(\mathcal{R}_t^\gamma(h) - R_s^\gamma(h)) - \epsilon + \frac{1}{l}\right] \ .$$

Now, for any $s \in Q$, there exists an $h(s)$ satisfying

$$\mathcal{R}^\gamma(h(s)) - R_s^\gamma(h(s)) \geq \epsilon \ .$$

It follows that

$$\begin{aligned}
I(s) & \geq \sum_{t\in Z^l} P(t)\theta\left[\mathcal{R}_t^\gamma(h(s)) - R_s^\gamma(h(s)) - \epsilon + \frac{1}{l}\right] \\
& = \sum_{t\in Z^l} P(t)\theta\left[\mathcal{R}_t^\gamma(h(s)) - \mathcal{R}^\gamma(h(s)) + \mathcal{R}^\gamma(h(s)) - R_s^\gamma(h(s)) - \epsilon + \frac{1}{l}\right] \\
& \geq \sum_{t\in Z^l} P(t)\theta\left[\mathcal{R}_t^\gamma(h(s)) - \mathcal{R}^\gamma(h(s)) + \frac{1}{l}\right] \\
& = P_t(\mathcal{R}_t^\gamma(h(s)) \geq \mathcal{R}^\gamma(h(s)) - \frac{1}{l}) \\
& = P_{t\in Z^l}(l\mathcal{R}_t^\gamma(h(s)) \geq l\mathcal{R}^\gamma(h(s)) - 1) \ . \qquad \text{(A.14)}
\end{aligned}$$

Now, from the definition of the left hand risk in the probability in (A.14), $l\mathcal{R}_t^\gamma(h(s))$ has a binomial distribution with parameters $(l, \mathcal{R}^\gamma(h(s)))$. Thus, the probability in (A.14) is the probability that the binomial variable exceeds

$$M = l\mathcal{R}^\gamma(h(s)) - 1$$

which is 1 less than its mean. Due to a result shown by Jogdeo and Samuels, we know that the median of a binomial distribution is greater than M [55]. Thus we can bound $I(s)$ from below by $\frac{1}{2}$.

Returning to (A.13), we then have

$$\begin{aligned} & P_{s\in Z^m, t\in Z^l}\left(\sup_{h\in\mathcal{H}}(\mathcal{R}_t^\gamma(h) - R_s^\gamma(h)) \geq \epsilon - \frac{1}{l}\right) \\ \geq\ & \sum_{s\in Q} P(s)\frac{1}{2} \\ =\ & \frac{1}{2}P_s\left(\sup_{h\in\mathcal{H}}(\mathcal{R}^\gamma(h) - R_s^\gamma(h)) \geq \epsilon\right) \quad . \end{aligned}$$

Multiplying by 2 yields the result. □

This theorem is usually used for the case $m = l$ (the case in [69]).

Attention is now given to bounding the probability in the right hand side of Theorem 66. This is usually done by a permutation argument: given the points in two equal-sized samples, we consider all possible pairs of equal-sized samples generated from these points, and investigate the proportion of these pairs with risk deviation exceeding ϵ.[23]

A.10.5 Results Used in Deriving Transductive Bounds

Both the traditional approach and the approach used in this thesis make use of the following result:

Theorem 67. *For a finite set A, and a function $f(x,y)$,*

$$P(\sup_{a\in A} f(x,a) > \epsilon) \leq |A| \sup_{a\in A} P(f(x,a) > \epsilon) \quad .$$

[23] For more details, see [10, 69].

Proof.

$$\begin{aligned}
P(\sup_{a\in A} f(x,a) > \epsilon) &= \int_{x\in X} \theta(\sup_{a\in A} f(x,a) > \epsilon) f(x) dx \\
&\leq \int_{x\in X} |\{a \in A : f(x,a) > \epsilon\}| f(x) dx \\
&= \int_{x\in X} (\sum_{a\in A} \theta(f(x,a) > \epsilon)) f x dx \\
&= \sum_{a\in A} \int_{x\in X} \theta(f(x,a) > \epsilon) f x dx \\
&= \sum_{a\in A} P(f(x,a) > \epsilon) \\
&\leq |A| \sup_{a\in A} P(f(x,a) > \epsilon) \ \ ,
\end{aligned}$$

with the integrals replaced by sums for a discrete random variable X. □

We also utilize the following generalization of a result by Bartlett [10, Proposition 8]:

Theorem 68. *Let $(X, \mathcal{F}, P)$ be a probability space, and let*

$$\{E(\alpha_1, \alpha_2, \delta) : 0 < \alpha_1, \alpha_2 \leq r \text{ and } 0 < \delta \leq 1\}$$

be a set of events satisfying the following conditions:

1. *for all $0 < \alpha \leq r$ and $0 < \delta \leq 1$, $P(E(\alpha, \alpha, \delta)) \leq \delta$;*
2. *for all $0 < a < 1$ and $0 < \delta \leq 1$,*

   $$\bigcup_{\alpha\in(0,r]} E(\alpha a, \alpha, \delta\alpha(1-a))$$

   is measurable; and
3. *for all $0 < \alpha_1 \leq \alpha \leq \alpha_2 \leq r$ and $0 < \delta_1 \leq \delta \leq 1$,*

   $$E(\alpha_1, \alpha_2, \delta_1) \subseteq E(\alpha, \alpha, \delta) \ \ .$$

Then, for $0 < \alpha < r$ and $0 < \delta < 1$,

$$P\left(\bigcup_{\alpha\in(0,r]} E\left(\alpha a, \alpha, \frac{\delta\alpha(1-a)}{r}\right)\right) \leq \delta \ \ .$$

Proof.

$$
\begin{aligned}
& P\left(\bigcup_{\alpha\in(0,r]} E\left(\alpha a, \alpha, \frac{\delta\alpha\,(1-a)}{r}\right)\right) \\
= \; & P\left(\bigcup_{i=0}^{\infty}\{E\left(\alpha a, \alpha, \frac{\delta\alpha\,(1-a)}{r}\right) : \alpha \in (ra^{i+1}, ra^{i}]\}\right) \\
\leq \; & P\left(\bigcup_{i=0}^{\infty} E\left(ra^{i+1}, ra^{i+1}, \delta a^{i}\,(1-a)\right)\right) \\
\leq \; & \sum_{i=0}^{\infty} \delta a^{i}\,(1-a) \\
= \; & \delta\,(1-a)\sum_{i=0}^{\infty} a^{i} \\
= \; & \delta \; .
\end{aligned}
$$

Appendix B

List of Abbreviations

This appendix lists the abbreviations used in this thesis, their page of first occurrence, and what they stand for.

Abbreviation	*Page*	*In full*
1-v-1	43	One-versus-one
1-v-r	42	One-versus-rest
ACM	41	Analytic Centre Machine
AM-SVM	39	Adaptive Margin Support Vector Machine
ANN	9	Artificial Neural Network
ANOVA	76	Analysis of Variance
BPM	41	Bayes Point Machine
c.d.f.	7	Cumulative Density Function
DAG	44	Directed Acyclic Graph
DD-SRM	67	Data-dependent Structural Risk Minimization
DFA	11	Deterministic Finite Automaton
ERM	4	Empirical Risk Minimization
FLD	15	Fisher's Linear Discriminant
HMM	9	Hidden Markov Model
HS	119	Hilbert Schmidt
KDD	58	Knowledge Discovery and Data Mining
KKT	27	Karush-Kuhn-Tucker
loo	39	Leave-one-out

Abbreviation	*Page*	*In full*
LOO-SVM	39	Leave-one-out Support Vector Machine
LP-SVM	39	Linear Programming Support Vector Machine
LS-SVM	40	Least Squares Support Vector Machine
MC-SVM	46	Multi-class Support Vector Machine
ML	15	Maximum Likelihood
NIST	57	National Institute of Standards and Technologies
OCR	42	Optical Character Recognition
PAC	3	Probably Approximately Correct
PCA	5	Principal Components Analysis
p.d.f.	7	Probability Density Function
QP	8	Quadratic Programming
RBF	9	Radial Basis Function
RHUL	50	Royal Holloway University of London
SLT	4	Structural Learning Theory
SOM	19	Self-organizing Map
SMO	52	Sequential Minimal Optimization
SRM	1	Structural Risk Minimization
SV	24	Support Vector
SVM	1	Support Vector Machine
TIS	58	Translation Initiation Site
TSVM	1	Transductive Support Vector Machine
USPS	57	United States Postal Service
VC	13	Vapnik-Chervonenkis

Bibliography

[1] Charalambos D. Aliprantis and Owen Burkinshaw. *Principles of Real Analysis.* Academic Press, Inc., second edition, 1981.

[2] Dana Angluin. Computational learning theory: Survey and selected bibliography. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 351–369. ACM Press, 1992.

[3] Annual KDD cup - web page for 2001 competition. http://www.cs.wisc.edu/~dpage/kddcup2001/.

[4] Martin Anthony. Probabilistic analysis of learning in artificial neural networks: The PAC model and its variants. Technical Report NC-TR-94-003, NeuroCOLT, 1994.

[5] Martin Anthony and Peter L. Bartlett. Function learning from interpolation. Technical Report NC-TR-94-013, NeuroCOLT, 1994.

[6] Martin Anthony and John Shawe-Taylor. A result of Vapnik with applications. *Discrete Applied Mathematics*, 47:207–217, 1993.

[7] Peter Arcidiacono and John P. Jones. Finite mixture distributions, sequential likelihood and the EM algorithm, September 2000.

[8] Jean-Pierre Barthélemy, Gèrard Cohen, and Antoine Lobstein. *Algorithmic Complexity and Communication Problems.* UCL Press, London, 1996. Original French version: 'Complexité algorithmique. Application à des problèmes de communication. Masson, Paris, 1996.' Translated by Catherine Fritsch-Mignotte and Maurice Mignotte.

[9] Peter Bartlett and John Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander Johannes Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, USA, 1998.

[10] Peter L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

[11] Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Local Rademacher complexities. Submitted, November 2002.

[12] Gaston Baudat and Fatiha Anouar. Kernel-based methods and function approximation. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '01)*, pages 1244–1249, Washington, D.C., July 2001.

[13] Eric B. Baum and David Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, January 1989.

[14] Shai Ben-David, Nicolò Cesa-Bianchi, David Haussler, and Philip M. Long. Characterizations of learnability for classes of $\{0, \ldots, n\}$-valued functions. Technical Report NC-TR-95-032, NeuroCOLT, 1995.

[15] Yoshua Bengio. A connectionist approach to speech recognition. *International Journal on Pattern Recognition and Artificial Intelligence*, 7(4):647–668, 1993.

[16] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Computer Science and Applied Mathematics. A Series of Monographs and Textbooks. Academic Press, Inc., New York, 1982.

[17] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[18] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 273–282. ACM Press, 1986.

[19] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965, 1989.

[20] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir Naumovich Vapnik. A training algorithm for optimal margin classifiers. In David Haussler, editor, *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburg, PA, 1992. ACM Press.

[21] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. A sharp concentration inequality with applications. Technical Report NC-TR-99-057, NeuroCOLT, May 1999.

[22] Erin J. Bredensteiner and Kristin P. Bennett. Multicategory classification by support vector machines. *Computational Optimizations and Applications*, 12:53–79, 1999.

[23] Michael P. S. Brown, William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, Jr., and David Haussler. Support vector machine classification of microarray gene expression data. UCSC-CRL 99-09, University of California, Santa Cruz, Santa Cruz, CA, June 1999.

[24] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2):121–167, 1998.

[25] Colin Campbell. Kernel methods: A survey of current techniques. Draft of a tutorial submission appearing in Neurocomputing, volume 48, pages 63–84, 2000.

[26] Colin Campbell. An introduction to kernel methods. In Robert J. Howlett and Lakhmi C. Jain, editors, *Radial Basis Function Networks 1. Recent Developments in Theory and Applications (Studies in Fuzziness and Soft Computing Vol. 66)*, chapter 7, pages 155–192. Physica-Verlag, Heidelberg, New York, March 2001.

[27] Colin Campbell and Nello Cristianini. Simple learning algorithms for training support vector machines. Technical report, Department of Engineering Mathematics, University of Bristol, Bristol, 1998.

[28] Chih-Chung Chang and Chih-Jen Lin. BSVM website. http://www.csie.ntu.edu.tw/~cjlin/bsvm/.

[29] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a Library for Support Vector Machines (Version 2.2)*. National Taiwan University, Taipei. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[30] Olivier Chapelle. Support vector machines for image classification. Master's thesis, l'Ecole Normale Supérieure de Lyon, August 1998. French title: Support Vector Machines et Classification d'Images.

[31] Olivier Chapelle, Patrick Haffner, and Vladimir Naumovich Vapnik. SVMs for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5), 1999.

[32] Olivier Chapelle and Vladimir Naumovich Vapnik. Model selection for support vector machines. In S.A. Solla, T.K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

[33] Olivier Chapelle, Vladimir Naumovich Vapnik, and Jason A. E. Weston. Transductive inference for estimating values of functions. In *Advances in Neural Information Processing Systems 12*, Cambridge, Mass., 1999. MIT Press.

[34] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. Submitted to Neural Information Processing Systems 2002 (nformation NIPS02).

[35] Olivier Chapelle, Jason A. E. Weston, Léon Bottou, and Vladimir Naumovich Vapnik. Vicinal risk minimization. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2000.

[36] Corinna Cortes and Vladimir Vapnik. Support-Vector networks. *Machine Learning*, 20(3):273–297, 1995.

[37] Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46, 2000.

[38] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, pages 265–292, December 2001.

[39] Nello Cristianini, Colin Campbell, and John Shawe-Taylor. Dynamically adapting kernels in support vector machines. In S.A. Solla, T.K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, 1998. Also NeuroCOLT Technical Report NC-TR-98-017, Royal Holloway College, University of London, UK, 1998.

[40] Nello Cristianini and John Shawe-Taylor. *Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, UK, 2001.

[41] Thomas Dean, Sonia Leach, and Hagit Shatkay. Expectation maximization. Available on web at http://www.cs.brown.edu/research/ai/dynamics/tutorial/Documents/ExpectationMaximization.html. Notes for a tutorial on dynamical systems at Brown University.

[42] Thomas Dean, Sonia Leach, and Hagit Shatkay. Hidden Markov models. Available on web at

http://www.cs.brown.edu/research/ai/dynamics/tutorial/Documents/HiddenMarkovModels.html. Notes for a tutorial on dynamical systems at Brown University.

[43] Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass. Authorship attribution with support vector machines, 2000. Poster presented at The Learning Workshop (4-7 April 2000) in Snowbird, Utah. Available for download from http://ais.gmd.de/~leopold/snowbird.ps.

[44] Thomas G. Dietterich and Ghulum Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In T.L. Dean and K. McKeown, editors, *Proceedings of the Ninth AAAI National Conference on Artificial Intelligence*, pages 572–577, Menlo Park, CA, 1991. AAAI Press.

[45] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[46] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[47] Nelson Dunford and Jacob T. Schwartz. *Linear Operators Part I: General Theory*, volume VII of *Pure and Applied Mathematics: A Series of Texts and Monographs.* Interscience Publishers, Inc., New York, August 1957.

[48] Nelson Dunford and Jacob T. Schwartz. *Linear Operators Part II: Spectral Theory — Self Adjoint Operators in Hilbert Space*, volume VII of *Pure and Applied Mathematics: A Series of Texts and Monographs.* Interscience Publishers, Inc., New York, July 1963.

[49] André Elisseeff and Jason A. E. Weston. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems, vol 14 (NIPS01)*, Cambridge, Mass., 2001. MIT Press.

[50] Eric Weisstein's world of mathematics. http://mathworld.wolfram.com.

[51] Evaluation/characterization of classification algorithms. http://www.liacc.up.pt/ML/statlog/.

[52] Brina S. Everitt. *Cluster Analysis*. Edward Arnold, London, third edition, 1993.

[53] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. A unified framework for regularization networks and support vector machines. Technical Report A.I. Memo No. 1654/C.B.C.L. Paper No. 171, Massachusetts Institute of Technology: Artificial Intelligence Laboratory/Center for Biological and Computational Learning (Department of Brain and Cognitive Sciences), Cambridge, Mass., March 2000. Also appeared as 'Regularization Networks and Support Vector Machines' in 'Advances in Large-Margin Classifiers (Neural Information Processing) (Alexander Johannes Smola, Peter L. Bartlett, Bernhard Schölkopf, and D. Schuurmans, eds.), ch. 10, MIT Press, 2000'.

[54] Rodrigo Fernández. Predicting time series with a local support vector regression machine. In *Proceedings of ACAI'99*, 1999.

[55] Enrique Figueroa and Christian Houdré. Lossless coding through the concentration of measure phenomenon. Available for download at www.math.gatech.edu/~houdre/research/papers/lossless.pdf.

[56] Gary William Flake and Steve Lawrence. Efficient SVM regression training with SMO. *Machine Learning*, 2001. Also Technical report for NEC Research Institute, 1999.

[57] Roger Fletcher. *Practical Methods of Optimization, Volume 1: Unconstrained Optimization*. John Wiley and Sons, 1980.

[58] Paolo Frasconi, Marco Gori, Marco Maggini, and Giovanni Soda. Unified integration of explicit knowledge and learning by example in

recurrent networks. *Knowledge and Data Engineering*, 7(2):340–346, 1995.

[59] Paolo Frasconi, Marco Gori, Marco Maggini, and Giovanni Soda. Representation of finite state automata in recurrent radial basis function networks. *Machine Learning*, 23(1):5–32, 1996.

[60] Glenn Fung and Olvi L. Mangasarian. Proximal support vector machine classifiers. In *Knowledge Discovery and Data Mining*, pages 77–86, 2001.

[61] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, January 1992.

[62] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

[63] GMD FIRST (IDA) binary classification benchmark repository. http://ida.first.gmd.de/~raetsch/data/benchmarks.htm.

[64] Francisco J. González-Castaño and Robert R. Meyer. Projection support vector machines. Mathematical Programming Technical Report MP-00-05, University of Wisconsin, November 2000.

[65] T. Graepel, Ralf Herbrich, and K. Obermayer. Bayesian transduction. In *Advances in Neural Information Processing Systems 1999*, 1999.

[66] Leslie Grate, Richard Hughey, Kevin Karplus, and Kimmen Sjölander. Tutorial: Stochastic modeling techniques: Understanding and using hidden Markov models. Technical report, University of California, Santa Cruz, June 1996.

[67] Yann Guermeur. A simple unifying theory of multi-class support vector machines. Rapport de recherche 4669, Institut National de Recherche en Informatique et en Automatique, December 2002. ISRN INRIA/RR–4669–FR+ENG.

[68] Yann Guermeur, André Elisseeff, and Hélène Paugam-Moisy. A new multi-class SVM based on a uniform convergence result. In S.-I. Amari, Christopher Lee Giles, M. Gori, and V. Piuri, editors, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000)*, 2000.

[69] Yann Guermeur, André Elisseeff, and Dominique Zelus. Bounding the capacity measure of multi-class discriminant models. Technical Report NC2-TR-2002-123, NeuroCOLT, February 2002.

[70] Ying Guo, Peter L. Bartlett, John Shawe-Taylor, and Robert C. Williamson. Covering numbers for support vector machines. *IEEE Transactions on Information Theory*, 48(1):239–250, January 2002.

[71] Isabelle Guyon, Bernhard Boser, and Vladimir Naumovich Vapnik. Automatic capacity tuning of very large VC-dimension classifiers. In *Advances in Neural Information Processing Systems 5 (NIPS'92)*, pages 147–155, San Mateo, CA, 1993. Morgan Kaufmann.

[72] David J. Hand. *Kernel Discriminant Analysis*, volume 2 of *Pattern Recognition & Image Processing Research Studies Series*. Research Studies Press, John Wiley and Sons, Chichester, 1982. Editor: Dr Josef Kittler.

[73] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. Technical report, Stanford University and University of Toronto, November 1996. Condensed version also appeared in Advances in Neural Information Processing Systems 10, 1998.

[74] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Series in Statistics. Springer, 2001.

[75] David Haussler. Part 1: Overview of the probably approximately correct (PAC) learning framework. In P. Smolensky, M. C. Mozer, and D. E. Rumelhart, editors, *Mathematical Perspectives on Neural Networks*. Lawrence Erlbaum Publishers, Mahwah, NJ, 1996.

Contains reprinted material from 'Decision Theoretic Generalizations of the PAC Model for Neural Nets and Other Learning Applications', Information and Computation, Vol. 100, September, 1992, pp. 78-150.

[76] Simon Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, New Jersey, second edition, 1999. ISBN: 0-13-273350-1.

[77] Marti A. Hearst, Bernhard Schölkopf, Susan Dumais, Edgar Osuna, and John Platt. Trends and controversies: Support vector machines. *IEEE Intelligent Systems*, July 1998.

[78] David Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, March 1995. Revised November 1996. Slides available at ftp://ftp.research.microsoft.com/pub/dtg/david/tutorial.ps.

[79] Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayes point machines: Estimating the Bayes point in kernel space. In *International Joint Conference on Artificial Intelligence Workshop on Support Vector Machines*, pages 23–27, 1999.

[80] Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayesian learning in reproducing kernel Hilbert spaces. Technical Report TR 99-11, Department of Computer Science, Technical University of Berlin, Franklin Street 28/29, 10587 Berlin, July 1999. http://stat.cs.tu-berlin.de/publications.

[81] Ralf Herbrich, Thore Graepel, and Colin Campbell. Robust Bayes point machines. In *Proceedings of ESANN 2000*, pages 49–54, 2000.

[82] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, March 1963.

[83] Kevin D. Hoover. Nonstationary time series, cointegration, and the principle of the common cause. *British Journal for the Philosophy of Science*, 2002. Forthcoming.

[84] Chih-Wei Hsu and Chih-Jen Lin. A comparison on methods for multi-class support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan, 2001.
http://www.csie.ntu.edu.tw/~cjlin/papers/multisvm.ps.gz.

[85] Isabelle Guyon's list of SVM applications.
http://www.clopinet.com/isabelle/Projects/SVM/applist.html.

[86] Tommi S. Jaakkola, Mark Diekhans, and David Haussler. A discriminative framework for detecting remote protein homologies, 1998.
http://www.cse.ucsc.edu/research/compbio/discriminative/Jaakola2-1998.ps.

[87] Tommi S. Jaakkola and David Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*. Morgan Kaufmann, 1999.

[88] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[89] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. LS VIII-Report 23, Fachbereich Informatik, Universität Dortmund, Dortmund, Germany, November 1997. Revised April 1998.

[90] Thorsten Joachims. Making large-scale SVM learning practical. LS VIII-Report 24, Fachbereich Informatik, Universität Dortmund, Dortmund, Germany, June 1998.
http://www-ai.cs.uni-dortmund.de/svm_light.

[91] Thorsten Joachims. Estimating the generalization performance of a svm efficiently. LS VIII-Report 25, Fachbereich Informatik, Universität Dortmund, Dortmund, Germany, December 1999.

[92] Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J.C. Burges, and Alexander Johannes Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 41–56. MIT Press, Cambridge, MA, 1999. Also published as LS VIII-Report 24 of the Fachbereich Informatik, Universitt Dortmund.

[93] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1999.

[94] Thorsten Joachims. Estimating the generalization performance of a SVM efficiently. In *Proceedings of the International Conference on Machine Learning (ICML)*. Morgan Kaufmann, 2000.

[95] Thorsten Joachims. A statistical learning model of text classification for support vector machines. In *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 2001.

[96] Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor. Composite kernels for hypertext categorisation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[97] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[98] Michael I. Jordan. Why the logistic function? A tutorial discussion on probabilities and neural networks. Computational Cognitive Science Report 9503, Massachusetts Institute of Technology, August 1995.

[99] Tapas Kanungo. Hidden Markov models. Slides for a tutorial on hidden Markov models. http://www.cfar.umd.edu/~kanungo, March 1998.

[100] Michael J. Kearns and Ming Li. Learning in the presence of malicious errors. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 267–280. ACM Press, 1988.

[101] Michael J. Kearns and Robert E. Schapire. Efficient distribution-free learning of probabilistic concepts. In Stephen Jose Hanson, George A. Drastal, and Ronald L. Rivest, editors, *Computational Learning Theory and Natural Learning Systems, Volume I: Constraints and Prospect*. Bradford/MIT Press, 1994. Also in 31st Annual IEEE Symposium on Foundations of Computer Science, pp. 382–391, 1990.

[102] Jae-On Kim and Charles W. Mueller. Introduction to factor analysis: Statistical methods and practical issues. Technical Report 07-014, Sage University, 1978. Volume 14 in the Quantitative Applications in the Social Sciences Series.

[103] Jae-On Kim and Charles W. Mueller. Introduction to factor analysis: What it is and how to do it. Technical Report 07-013, Sage University, 1978. Volume 13 in the Quantitative Applications in the Social Sciences Series.

[104] Delia Koo. *Elements of Optimization*. Heidelberg Science Library. Springer-Verlag, 1977.

[105] Erwin Kreyszig. *Introductory Functional Analysis with Applications*. Wiley Classics Library. John Wiley and Sons, 1989.

[106] James Tin-Yau Kwok. Integrating the evidence framework and the support vector machine. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 177–182, Bruges, Belgium, April 1999.

[107] James Tin-Yau Kwok. The evidence framework applied to support vector machines. *IEEE Transactions on Neural Networks*, 11(5):1162, September 2000.

[108] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semi-definite programming. Technical Report UCB/CSD-02-1206, Division of Computer Science, University of California, Berkeley, October 2002.

[109] Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory support vector machines. Technical Report TR 1043, Department of Statistics, University of Wisconsin-Madison, September 2001. To appear, Proceedings of the 33rd Symposium on the Interface.

[110] Yuh-Jye Lee, Olvi L. Mangasarian, and W. H. Wolberg. Breast cancer survival and chemotherapy: A support vector machine analysis. Technical Report 99-10, Data Mining Institute, December 1999. Also in DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 55 (2000), pp. 1–10.

[111] Yuh-Jye Lee, Olvi L. Mangasarian, and W. H. Wolberg. Survival-time classification of breast cancer patients. Technical Report 01-03, Data Mining Institute, March 2001.

[112] Chih-Jen Lin, 2001. Personal Communication.

[113] Richard P. Lippmann. An introduction to computing with neural nets. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35:4–22, April 1987.

[114] F. A. Lootsma, editor. *Numerical Methods for Non-linear Optimization.* Academic Press, London, 1972.

[115] David Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, 1984.

[116] Donald MacDonald and Colin Fyfe. The kernel self organising map. In *First International NAISO Symposium on Engineering of*

Intelligent Systems (EIS 2000). National and Artificial Intelligence Systems Organization, 2000.

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

[118] David J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736, 1992.

[119] Olvi L. Mangasarian and David R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10(5):1032–1037, September 1999. Also Mathematical Programming Technical Report 98-18.

[120] Timothy Masters. *Advanced Algorithms for Neural Networks*, pages 158–192. John Wiley & Sons, 1995. ISBN: 0471105880.

[121] Eddy Mayoraz and Ethem Alpaydin. Support vector machine for multi-class classification. IDIAP-RR 6, IDIAP, 1998.

[122] Eddy Mayoraz and Miguel Moreira. On the decomposition of polychotomies into dichotomies. In Douglas H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning*, pages 219–226. Morgan Kaufmann, 1997.

[123] Jorge J. Moré and Stephen J. Wright. *Optimization Software Guide*. Society for Industrial and Applied Mathematics, Philadelphia, 1993.

[124] Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1999.

[125] Sayan Mukherjee, Edgar Osuna, and Federico Girosi. Nonlinear prediction of chaotic time series using support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII - Proceedings of the 1997 IEEE Workshop*, pages 511–520, New York, 1997. IEEE.

[126] Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, March 2001.

[127] Klaus-Robert Müller, Alexander Johannes Smola, Gunnar Rätsch, Bernhard Schölkopf, J. Kohlmorgen, and Vladimir Naumovich Vapnik. Predicting time series with support vector machines. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks - ICANN '97*, volume 1327 of *Lecture Notes in Computer Science*, pages 999–1004, Berlin, 1997. ICANN, Springer. Also in 'Advances in Kernel Methods — Support Vector Learning, pages 243 – 253. MIT Press, Cambridge, MA, 1999' as 'Using Support Vector Machines for Time Series Prediction'.

[128] Christian W. Omlin and C. Lee Giles. Symbolic knowledge representation in recurrent neural networks: Insights from theoretical models of computation. In I. Cloete and J.M. Zurada, editors, *Knowledge-Based Neurocomputing*. MIT Press, 2000.

[129] Edgar Osuna, Robert Freund, and Federico Girosi. Support vector machines: Training and applications. Technical Report AIM-1602, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1997. Also Center for Biological and Computational Learning, Department of Brain and Cognitive Sciences, C.B.C.L. Paper No. 144.

[130] Michael J. Panik. *Classical Optimization: Foundations and Extensions*, volume 16 of *Studies in Mathematical and Managerial Economics*. North-Holland/American Elsevier, 1976.

[131] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998.

[132] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander Johannes Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999.

[133] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Alexander Johannes Smola, Peter L. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large-Margin Classifiers (Neural Information Processing)*, chapter 5. MIT Press, March 1999. Also published as 'Probabilities for SV Machines' in 'Advances in Large Margin Classifiers', Chapter 5. http://research.microsoft.com/~jplatt.

[134] John C. Platt. Using sparseness and analytic QP to speed training of support vector machines. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, Cambridge, MA, 1999. MIT Press.

[135] John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large margin DAGs for multiclass classification. In S.A. Solla, T.K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

[136] Gunnar Rätsch, Ayhan Demiriz, and Kristin P. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. NeuroCOLT2 Technical Report 85, Royal Holloway College, University of London, Egham, UK, September 2000. Accepted for publication in the Machine Learning special issue on 'New Methods for Model Selection and Model Combination'.

[137] Ryan Rifkin and Michelle Nadermann. *SvmFu: Software For SVMs*. Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. http://www.ai.mit.edu.

[138] R. Tyrrell Rockafellar. *Convex Analysis.* Princeton Mathematical Series. Princeton University Press, Princeton, New Jersey, 1970.

[139] Frank Rosenblatt. *Principles of Neurodynamics.* Spartan Books, Washington, D.C., 1962.

[140] C. Saunders, Alex Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521, San Francisco, CA, 1998. Morgan Kaufmann.

[141] C. Saunders, Mark O. Stitson, Jason Weston, L. Bottou, Bernhard Schölkopf, and Alexander Johannes Smola. Support vector machine — reference manual. Technical Report CSD-TR-98-03, Department of Computer Science, Royal Holloway, University of London, Egham, UK, 1998. http://svm.dcs.rhbnc.ac.uk.

[142] Tobias Scheffer and Thorsten Joachims. Estimating the expected error of empirical minimizers for model selection. Technical Report TR-98-9, Computer Science Department, Technische Universität Berlin, Berlin, Germany, July 1998.

[143] Bernhard Schölkopf. Support vector learning. Available on the web, September 1999. Lecture notes and slides. http://www.first.gmd.de/~bs.

[144] Bernhard Schölkopf. The kernel trick for distances. Technical Report MSR-TR-2000-51, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, May 2000.

[145] Bernhard Schölkopf. Statistical learning and kernel methods. Technical Report MSR-TR-2000-23, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, February 2000. Lecture notes for a course taught at the Interdisciplinary College, Günne, Germany, in March 2000. http://research.microsoft.com/~bsc.

[146] Bernhard Schölkopf, 2003. Personal Communication.

[147] Bernhard Schölkopf, Christopher J. C. Burges, and Vladimir Naumovich Vapnik. Extracting support data for a given task. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings, First International Conference on Knowledge Discovery & Data Mining*, pages 252–257, Menlo Park, CA, 1995. AAAI Press.

[148] Bernhard Schölkopf, Christopher J. C. Burges, and Vladimir Naumovich Vapnik. Incorporating invariances in support vector learning machines. In C. von der Malsburg, W. von Seelen, J.C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks - ICANN'96 (Springer Lecture Notes in Computer Science, Vol 1112)*, pages 47–52. Springer, Berlin, 1996.

[149] Bernhard Schölkopf, Christopher J.C. Burges, and Alexander Johannes Smola, editors. *Advances in Kernel Methods: Support Vector Learning.* MIT Press, Cambridge, Massachusetts, 1999.

[150] Bernhard Schölkopf, Sebastian Mika, Christopher J.C. Burges, P. Knirsch, Klaus-Robert Müller, Gunnar Rätsch, and Alexander Johannes Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, September 1999.

[151] Bernhard Schölkopf, John Shawe-Taylor, Alexander Johannes Smola, and Robert C. Williamson. Generalization bounds via eigenvalues of the Gram matrix. Technical Report NC2-TR-1999-035, NeuroCOLT2, March 1999.

[152] Bernhard Schölkopf, John Shawe-Taylor, Alexander Johannes Smola, and Robert C. Williamson. Kernel-dependent support vector error bounds. In D. Willshaw and A. Murray, editors, *Proceedings of the International Conference on Artificial Neural Networks 1999*, volume 1, pages 103–108. IEE Conference Publications, 1999.

[153] Bernhard Schölkopf, P.Y. Simard, Alexander Johannes Smola, and Vladimir Naumovich Vapnik. Prior knowledge in support vector

kernels. In Michael I. Jordan, Michael J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processings Systems, Volume 10*, pages 640–646, Cambridge, MA, 1998. MIT Press.

[154] Bernhard Schölkopf and Alexander Johannes Smola. *Learning with Kernels — Support Vector Machines, Regularization, Optimization and Beyond.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2002.

[155] Bernhard Schölkopf, Alexander Johannes Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max-Planck-Institut für biologische Kybernetik, Arbeitsgruppe Bülthoff, Spemannstraße 38, 72076 Tübingen, Germany, December 1996. Also appeared in Neural Computation, 10:1299–1319, 1998. http://www.mpik-tueb.mpg.de/bu.html.

[156] Bernhard Schölkopf, Alexander Johannes Smola, Robert C. Williamson, and Peter L. Bartlett. New support vector algorithms. Technical Report NC-TR-98-031, NeuroCOLT, UK, 1998. Also appeared in Neural Computation, vol. 12, pp. 1207–1245, 2000.

[157] Bernhard Schölkopf, Kah-Kay Sung, Christopher J.C. Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Naumovich Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765, 1997.

[158] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. A framework for structural risk minimization. Technical Report NC-TR-96-032, NeuroCOLT, 1996.

[159] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998. Also NeuroCOLT Technical Report NC-TR-96-053, October 1996.

[160] John Shawe-Taylor and Nello Cristianini. Margin distribution bounds on generalization. Technical Report NC-TR-98-020, NeuroCOLT, July 1998.

[161] John Shawe-Taylor and Nello Cristianini. On the generalisation of soft margin algorithms. Technical Report NC-TR-00-082, NeuroCOLT, 2000.

[162] Alexander Johannes Smola, 2003. Personal Communication.

[163] Alexander Johannes Smola, Olvi L. Mangasarian, and Bernhard Schölkopf. Sparse kernel feature analysis. Technical Report 99-04, University of Wisconsin, Data Mining Institute, Madison, 1999.

[164] Alexander Johannes Smola and Bernhard Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. Technical Report 1064, GMD FIRST, April 1997. Also appeared in Algorithmica, 22:211–231, 1998.

[165] Alexander Johannes Smola and Bernhard Schölkopf. From regularization operators to support vector kernels. In *Advances in Neural Information Processings Systems 10*, pages 343–349, San Mateo, CA, 1998.

[166] Alexander Johannes Smola and Bernhard Schölkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT, Royal Holloway College, University of London, October 1998. Also published in Statistics and Computing 2001. http://www.neurocolt.com.

[167] Alexander Johannes Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 911–918, San Francisco, CA, 2000. Morgan Kaufmann.

[168] Alexander Johannes Smola, Bernhard Schölkopf, and Klaus-Robert Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998.

[169] Alexander Johannes Smola, John Shawe-Taylor, Bernhard Schölkopf, and Robert C. Williamson. The entropy regularization information criterion. In *Advances in Neural Information Processing Systems, 2000*, 2000.

[170] Sean Snyders. Inductive machine learning bias in knowledge-based neurocomputing. Master's thesis, University of Stellenbosch, Stellenbosch, South Africa, December 2002.

[171] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000. Also Technical Report 00-034 of Department of Computer Science and Engineering, University of Minnesota.

[172] Mark Stitson, Alex Gammerman, Vladimir Naumovich Vapnik, Volodya Vovk, Chris Watkins, and Jason Weston. Support vector regression with ANOVA decomposition kernels. Technical Report CSD-TR-97-22, Department of Computer Science, Royal Holloway College, University of London, Egham, Surrey TW20 0EX, England, November 1997. Also appeared in: Bernhard Schölkopf, Christopher J.C. Burges, and Alexander Johannes Smola, editors, Advances in Kernel Methods — Support Vector Learning, pages 285 – 291. MIT Press, Cambridge, MA, 1999.

[173] Johan A. K. Suykens, J. de Brabanter, L. Lukas, and Joos Vandewalle. Weighted least squares support vector machines: Robustness and sparse approximation. *Neurocomputing*, 48(1–4):85–105, 2001.

[174] Johan A. K. Suykens, L. Lukas, and Joos Vandewalle. Sparse least square support vector machine classifiers. In *ESANN 2000 European Symposium on Artificial Neural Networks*, pages 37–42, 2000.

[175] Johan A. K. Suykens and Joos Vandewalle. Recurrent least squares support vector machines. *IEEE Transactions on Circuits and Systems - I*, 47(7):1109–1114, July 2000. Running title: Recurrent Support Vector Machines.

[176] Johan A.K. Suykens, L. Lukas, and Joos Vandewalle. Sparse approximation using least squares support vector machines. In *IEEE International Symposium on Circuits and Systems (ISCAS 2000)*. IEEE, 2000.

[177] Johan A.K. Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, June 1999.

[178] Johan A.K. Suykens and Joos Vandewalle. Multiclass least squares support vector machines. In *IJCNN '99 International Joint Conference on Neural Networks*, Washington D.C., 1999.

[179] Earl W. Swokowski, Michael Olinick, and Dennis Pence. *Calculus*. PWS Publishing Company, Boston, sixth edition, 1994.

[180] Andrew S. Tanenbaum. *Structured Computer Organization*. Prentice-Hall, Upper Saddle River, NJ, fourth edition, 1999.

[181] D.M.J. Tax and R.P.W. Duin. Data domain description using support vectors. In M. Verleysen, editor, *Proceedings of the European Symposium on Artificial Neural Networks (Bruges, April 21-23, 1999)*, pages 251–257, Brussels, 1999. D-Facto.

[182] The Santa Fe time series competition data. http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html.

[183] Michael E. Tipping. The relevance vector machine. In S.A. Solla, T.K. Leen, and Klaus Robert Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 652–658, Cambridge, Mass., 2000. MIT Press.

[184] University of Toronto's DELVE benchmark repository. http://www.cs.utoronto.ca/~delve/data/datasets.html.

[185] Leslie G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142, 1984.

[186] Tony van Gestel, Johan A. K. Suykens, Dirk-Emma Baestaens, Annemie Lambrechts, Gert R. G. Lanckriet, Bruno Vandaele, Bart de Moor, and Joos Vandewalle. Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks, Special Issue on Neural Networks in Financial Engineering*, 12(4):809–821, 2001.

[187] Tony van Gestel, Johan A. K. Suykens, Gert R. G. Lanckriet, Annemie Lambrechts, Bart de Moor, and Joos Vandewalle. Bayesian framework for least squares support vector machine classifiers, Gaussian processes and kernel Fisher discriminant analysis. Manuscript no. 2281, August 2001.

[188] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[189] R. J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 11:451–484, 1999.

[190] Vladimir Naumovich Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer Series in Statistics. Springer-Verlag, 1982. Translated by Samuel Kotz.

[191] Vladimir Naumovich Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[192] Vladimir Naumovich Vapnik and Olivier Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013–2036, 2000.

[193] Vladimir Naumovich Vapnik, Steven E. Golowich, and Alexander Johannes Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. Mozer, Michael I. Jordan, and T. Petsche, editors, *Advances in*

Neural Information Processing Systems 9, pages 281–287, Cambridge, MA, 1997. MIT Press.

[194] Dane A. Walsh. Image orientation using support vector machines. Master's thesis, University of Stellenbosch, Stellenbosch, South Africa, March 2003.

[195] Yair Weiss. Motion segmentation using EM—a short tutorial. Available on web, 1996. Class tutorial for a computational vision class at Massachusetts Institute of Technology. http://persci.mit.edu/people/yweiss/emTutorial.ps.

[196] Jason A. E. Weston. *Extensions to the Support Vector Method*. PhD thesis, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey, October 1999.

[197] Jason A. E. Weston. Leave-One-Out support vector machines. In *International Joint Conference on Artificial Intelligence (IJCAI'99)*, 1999.

[198] Jason A. E. Weston, André Elisseeff, Bernhard Schölkopf, and Michael E. Tipping. Use of the zero-norm with linear models and kernel methods. Submitted to the Journal of Machine Learning Research Special Issue on Variable and Feature Selection.

[199] Jason A. E. Weston, Fernando Pérez-Cruz, Olivier Bousquet, Olivier Chapelle, André Elisseeff, and Bernhard Schölkopf. Feature selection and transduction for prediction of molecular bioactivity for drug design. *Bioinformatics*, 1(1):1–8, 2002.

[200] Jason A. E. Weston and Chris Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England, May 1998.

[201] Robert C. Williamson, John Shawe-Taylor, Bernhard Schölkopf, and Alexander Johannes Smola. Sample based generalization bounds. Technical Report NC-TR-99-055, NeuroCOLT, 1999.

[202] Robert C. Williamson, Alexander Johannes Smola, and Bernhard Schölkopf. Entropy numbers, operators and support vector kernels. Technical Report NC-TR-98-023, NeuroCOLT, July 1998.

[203] Robert C. Williamson, Alexander Johannes Smola, and Bernhard Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. Technical Report NC-TR-98-019, NeuroCOLT, 1998.

[204] Robert C. Williamson, Alexander Johannes Smola, and Bernhard Schölkopf. Entropy numbers of linear function classes. In *Proceedings of the 13th Annual Conference on Computational Learning Theory*, pages 309–319. Morgan Kaufmann, San Francisco, 2000.

[205] Jianhua Xu, Xuegong Zhang, and Yanda Li. Kernel MSE algorithm: A unified framework for KFD, LS-SVM and KRR. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '01)*, pages 2:1486–1491, 2001.

[206] A. Zien, Gunnar Rätsch, Sebastian Mika, Bernhard Schölkopf, C. Lemmen, Alexander Johannes Smola, T. Lengauer, and Klaus-Robert Müller. Engineering support vector machine kernels that recognize translation initiation sites, 1999. Presented at the German Conference on Bioinformatics.