PAC-Bayesian Pattern Classification with Kernels

Theory, Algorithms, and an Application to the Game of Go

vorgelegt von
Diplom-Physiker (Dipl.-Phys.)

Thore Graepel

Fakultät IV — Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
— Dr. rer. nat. —

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Heinrich Klar
Berichter: Prof. Dr. Ulrich Kockelkorn
Berichter: Prof. Dr. Fritz Wysotzki

Tag der wissenschaftlichen Aussprache: 12. Juli 2002

Berlin 2002

D 83

Für meine Eltern

Abstract

The thesis deals with problems of pattern classification in the framework of machine learning. The focus of the work is on *kernel methods* for the supervised classification of objects. The thesis gives a detailed introduction into the field of kernel algorithms and learning theory. New contributions include learning theoretical results in the PAC-Bayesian framework, efficient sampling algorithms for Bayesian classification in kernel space, and an application of kernel methods to pattern analysis in the game of Go.

Learning Theory In the *PAC-Bayesian framework* we derive new bounds on the prediction error of linear classifiers (in kernel space) in terms of the normalised *margin* achieved on the training sample, taking into account both the concentration of the training data and the margin distribution. Assuming *sparseness* of the dual variables we extend the PAC-Bayesian framework to data-dependent hypotheses. Finally, we prove "egalitarian" bounds on the probability of finding classifiers with high prediction error in subsets of hypothesis space with low empirical risk—results that emphasise the importance of model selection.

Learning Algorithms We discuss Bayesian classification in kernel space and identify *Bayesian transduction* and the *Bayes point machine* as optimal procedures for classification in a Bayesian sense. Assuming a uniform prior over normalised weights for a given kernel we devise sampling schemes for the resulting piecewise constant posterior: The *kernel Gibbs sampler*, a Markov chain Monte Carlo method able to deal with label noise, and the *permutational perceptron sampler* for large scale sampling. The classification techniques are tested on handwritten-digit recognition tasks and compare favourably to support vector machines when rejecting test data based on confidence measures.

Application We apply kernel classifiers to the domain of pattern classification in the *Japanese game of Go*, which serves as a test domain for classification problems involving symbolic or structured objects. In order to learn mappings from points on the board to numbers that indicate territorial status or move quality we define the *common fate graph* as a compact representation for Go positions and show how a feature space based on *relative subgraph features* can be constructed. Building on the this representation we train a support vector machine to learn the quality of moves from a collection of Go problems and from actual 9×9 game records. As a result, we obtain classifiers that solve certain Go problems and play 9×9 Go at a non-trivial level of playing strength.

Zusammenfassung

Die Arbeit beschäftigt sich mit der Kern-basierten überwachten Klassifikation von Mustern im Rahmen des maschinellen Lernens. Sie gibt eine detaillierte Einführung in die Themen Lerntheorie und Kern-basierte Algorithmen. Neue Beiträge bestehen in lerntheoretische Ergebnisse im Rahmen der PAC-Bayesianischen Theorie, neuen *sampling* Algorithmen für bayesianische Klassifikation in Kernräumen, und in der Anwendung von Kernmethoden auf Mustererkennung im japanische Brettspiel Go.

Lerntheorie Im Rahmen der PAC-Bayesianischen Theorie werden neue Schranken an den Vorhersagefehler linearer Klassifikatoren (in Kernräumen) hergeleitet, in die nicht nur der normierte *margin*, sondern auch die Konzentration der Trainingsdaten und die Verteilung der reellwertigen Ausgaben eingehen. Unter der Annahme dünnbesetzter dualer Variablen, wird die PAC-Bayesianische Theorie auf datenabhängige Hypothesenräume erweitert. Es werden "egalitäre" Schranken an die Wahrscheinlichkeit bewiesen, daß ein Klassifikator einen hohen Vorhersagefehler hat, obwohl er sich in einer Untermenge befindet, die im Mittel einen niedrigen Trainingsfehler aufweist. Diese Ergebnisse unterstreichen die Wichtigkeit von Modellselektion.

Lernalgorithmen *Bayesian transduction* und die *Bayes point machine* werden als optimale Klassifikationsverfahren im bayesianischen Sinne identifiziert. Es werden *sampling* Algorithmen für den resultierenden stückweise konstanten Posterior entwickelt: der *kernel Gibbs sampler,* ein MCMC Verfahren geeignet für den Fall verrauschter Klasseninformation und der *permutational perceptron sampler* für Probleme mit vielen Trainingsdaten. Beide Verfahren werden auf Klassifikationsproblemen zur Erkennung handgeschriebener Ziffern getestet und erweisen sich der *support vector machine* besonders bei der Angabe von Konfidenzwerten als überlegen.

Anwendung Als Anwendungsbeispiel für Kernklassifikatoren wird das Problem der Mustererkennung im japanischen Brettspiel Go betrachtet, das hier als Prototyp eines Klassifikationsproblems auf symbolischen bzw. strukturierten Objekten dient. Als kompakte Repräsentation von Go-Stellungen wird der *common fate graph* entwickelt, auf dessen Grundlage sogenannte *relative subgraph features* einen geeigneten Merkmalsraum aufspannen. Davon ausgehend wird eine *support vector machine* trainiert, um die Qualität von Zügen aus einer Sammlung von Go-Problemen und 9×9 Spielprotokollen zu lernen. Die resultierenden Klassifikatoren lösen bestimmte Problemstellungen und spielen 9×9 Go mit einer beachtlichen Spielstärke.

Contents

List of Symbols

This list contains important symbols that are used throughout the text. The list omits notation that is standard, self-explanatory or used only locally in the text.

Symbol	Meaning	Page
$\langle \cdot, \cdot \rangle$	inner product on ℓ_2^n	12
$\|\cdot\|_p$	ℓ_p-norm	12
$\|\cdot\|$	ℓ_2-norm	12
$\mathbf{0}_m$	m-vector of zeros	177
$\mathcal{A}$	learning algorithm	5
α_i , $\boldsymbol{\alpha}$	expansion coefficient (vector) of weight vector	19
Bayes	Bayes classification strategy	69
C	soft margin penalisation parameter	21
$\mathcal{C}(\cdot)$	compression function	99
$d(\cdot\,\cdot)$	metric	23
δ	confidence level of bounds	52
$\delta(\cdot)$	Dirac´s delta function	34
E	expectation value	4,201
$\mathbf{e}_i$	ith unit vector	6
$\varepsilon(\cdot)$	bound on prediction error	52
F_X	probability distribution function on $\mathcal{X}$	200
f_X	probability density on $\mathcal{X}$	27,200
$\mathrm{fat}_\mathcal{F}(\cdot)$	fat-shattering dimension of $\mathcal{F}$	61
G_{fgr}	full graph representation of a Go position	162
G_{cfg}	common fate graph of a Go position	163
$G_\mathcal{H}$	growth function of $\mathcal{H}$	55
Gibbs	Gibbs classification strategy	70
$\gamma(f, \boldsymbol{z})$	functional margin of f on training sample $\boldsymbol{z}$	16
$\gamma(\mathbf{w}, \boldsymbol{z})$	geometrical margin of weight vector $\mathbf{w}$ on training sample $\boldsymbol{z}$	16
$\Gamma(\mathbf{w}, \boldsymbol{z})$	normalised margin of weight vector $\mathbf{w}$ on training sample $\boldsymbol{z}$	82
$\boldsymbol{\Gamma}$	normalised margin distribution	88
$\mathcal{H}$	hypothesis space	4
$\mathcal{H}_\mathcal{K}$	hypothesis space of linear classifiers	12
$h(\cdot)$	hypothesis (classifier)	4
h^*	risk minimising hypothesis	9
$\mathbf{I}_n$	identity matrix of dimension n	22
I	indicator function	199

$I_{d,m}$	set of index vectors $\mathbf{i}$	99
$\mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)$	vector of indices of support vectors	20
$\mathbf{K}$	kernel or Gram matrix	19
$\mathcal{K}$	kernel feature space	11
$k(\cdot,\cdot)$	kernel function	31
$\mathrm{kum}(\mathcal{C})$	cummerbund of convex set $\mathcal{C}$	85
$\mathcal{L}(\cdot)$	likelihood	65
$L(\cdot)$	Lagrangian	177
$l(\cdot,\cdot)$	loss function	4
l_{0-1}	zero-one loss function	7
m	size of training sample $\boldsymbol{z}$	4
$\mathcal{N}$	covering number	60
n	dimensionality of feature space $\mathcal{K}$	11
$\mathbf{P}_{\mathsf{X}}$	probability measure on $\mathcal{X}$	4,200
$\mathrm{pol}(\mathcal{C})$	polar of convex set $\mathcal{C}$	86,202
Π_m	set of permutation functions on $\{1,\ldots,m\}$	54
$\pi(\cdot)$	permutation function	54
$\boldsymbol{\phi}(\cdot)$	feature mapping $\boldsymbol{\phi}:\mathcal{X}\to\mathcal{K}$	11
$R[h]$	risk functional for hypothesis h	4
$\hat{R}[h,\boldsymbol{z}]$	empirical risk of hypothesis h on training sample $\boldsymbol{z}$	5
$R_m[\mathcal{A}]$	generalisation error of algorithm $\mathcal{A}$	5
$\mathcal{R}(\cdot)$	reconstruction function	99
$\varsigma(\boldsymbol{x})$	data radius of input data $\boldsymbol{x}$	16
σ	bandwidth of Gaussian RBF kernel	39
$\mathcal{U}(\cdot)$	update function	113
$\mathbf{U}_{\mathsf{X}}$	uniform measure over $\mathcal{X}$	109
$V(\boldsymbol{z})$	version space	6
$\mathrm{VCdim}(\mathcal{H})$	VC dimension of $\mathcal{H}$	56
$\mathrm{vol}(\mathcal{C})$	volume of set $\mathcal{C}$ under uniform measure	5
$W(\cdot)$	Wolfe dual of quadratic programme	19
$\mathcal{W}$	ball of normalised weight vectors $\mathbf{w}$	12
$\mathbf{w}$	weight vector	12
$\mathcal{X}$	input space	3
x_i, $\boldsymbol{x}$	input sequence $(x_1,\ldots,x_m)\in\mathcal{X}^m$ of training sample	4
$\mathbf{x}$	input point mapped to feature space	11
$\vec{x}$	input vector in case $\mathcal{X}$ is a vector space	37
$\Xi(\boldsymbol{x})$	normalised data concentration of input data $\boldsymbol{x}$	87
ξ_i,$\boldsymbol{\xi}$	(vector of) soft margin slack variables	21
$\mathcal{Y}$	output space	3
y_i, $\boldsymbol{y}$	output sequence $(y_1,\ldots,y_m)\in\mathcal{Y}^m$ of training sample	4
Υ	logical formula	199
$\mathcal{Z}$	input-output space $\mathcal{X}\times\mathcal{Y}$	3
z_i, $\boldsymbol{z}$	training sample $(z_1,\ldots,z_m)\in(\mathcal{X}\times\mathcal{Y})^m$	4

Preface

Pursuing a doctorate is a bit like life itself. There are friends and foes, ups and downs, and—possibly most importantly—there is a beginning and an end. The grand theme of this thesis is *learning*, and—looking back—I have to admit that I learned a lot—not only about learning.

It all started by my decision to study physics out of curiosity about natural phenomena. Interested in the functioning of the brain I took a course in neural networks, received a research studentship in the field at the computer science department, and ended up doing my diploma project on the topic of neural networks. While the original work of my diploma thesis focused on the optimisation aspects of learning my subsequent work including this thesis focuses on the concept of *generalisation* in learning. How does this topic relate to my original plan of studying natural phenomena? It does so in at least two ways.

1. All the empirical sciences are based upon the concept of drawing general conclusions from particular observations. It is this general concept which is studied by learning theory. In this sense, machine learning and statistics can be considered sciences of inference. They study models of inductive inference and can thus be seen as a meta-science or a variant of applied epistemology.

2. Machine learning shares many aspects with the empirical sciences themselves. Due to the complexity of the data and the complexity of the algorithms applied machine learning is effectively an experimental science—where experimental is understood to include numerical simulations on a computer. Some of the results in this thesis were explicit attempts at formulating learning theoretical results that can—at least partially—be confirmed by experiments.

From the beginning of my scientific work I was encouraged to actively participate in scientific life, to visit conferences, and to publish. Also I did not have any given topic for this thesis from the beginning. Rather the results and algorithms emerged piece by piece, each giving new directions for further results and more ideas. This approach is risky because it does not guarantee the creation of a fully consistent final body of work. Another fallacy of the approach is one that may be observed throughout the whole field of machine learning: Since it is so easy to combine two earlier ideas into a new one (the gist of creativity) and to implement the resulting algorithm the field is swamped with publications on more or less original learning algorithms, which are often not tested

sufficiently in practice and which are not honestly compared to existing methods. In other words, there is a danger of moving from problem-oriented research to purely idea-oriented research. With these caveats in mind, however, active participation in scientific life is a great source of motivation and progress, and I would recommend it to anyone planning to do a PhD. As a result this thesis cannot be considered as my answer to any particular assigned question. Instead, it is the documentation of three years of continuous doctoral research.

However, it is not intended as a mere collection of results. As the word *thesis* and more so the word *defence* (viva) indicate the original idea of a doctoral thesis was to put forward certain scientific viewpoints and to be able to defend them before one's peers. With this idea in mind I composed this thesis from my past publications while trying to emphasise a line of argument ranging from the duality of example- and hypothesis-based learning algorithms via the necessity of data-dependent bounds to the insight that the task of model selection may often be more important than the search of the best hypothesis within a given model.

Acknowledgements

Writing up a doctoral thesis is a lonely business. The scientific work that leads to it is not. I would like to thank all those people who contributed in various ways to this work.

Most of the time of my doctorate I spent at the Computer Science Department of the Technical University of Berlin. I was fortunate enough to be a member of two research groups—one after another: The Neural Information Processing group of Prof. Klaus Obermayer and the statistics group of Prof. Ulrich Kockelkorn. Despite the conflict we had I would like to thank Prof. Obermayer for his earlier support. It is fair to say that his encouragement and scientific advice are among the main reasons why I started my doctoral work and why I am still pursuing a scientific career. After I had left Prof. Obermayer's group Prof. Kockelkorn made it possible to continue my doctoral work in his group. I am very grateful to Prof. Kockelkorn not only for providing an asylum during the rest of my time at TU Berlin and for becoming my doctoral advisor, but also for granting me interesting insights and advice from the statistician's point of view. As a result I am sometimes even tempted to almost call myself a statistician. Also, I thank Prof. Wysotzki for accepting the responsibility of being my second official advisor and I apologise for delivering more pages than I had originally intended.

From the Neural Information Processing group I would like to mention Christian Piepenbrock, Hendrik Purwins, and Peter Adorjan with whom I had interesting discussions—not only about science—and all three of which made me feel a little bit at home in that group. Also, many thanks go to Jürgen Schweiger and Jörg Betzin from the Statistics group for welcoming me in their group and for constantly reminding me of the mysterious foundations of statistics. I take particular pleasure in mentioning Peter Bollmann-Sdorra. Officially, I gratefully acknowledge his important contributions to this work, in particular, his proof of what we subsequently called Bollmann's Lemma. Unofficially, I would like to thank him for the many controversial and stimulating discussions we had in his office with topics ranging from business via politics to science.

Among my colleagues and friends I am probably most indebted to Matthias Burger and Ralf Herbrich. Matthias—first friend, later also colleague—greatly supported me through all my time at TU Berlin. He often is the first with whom to discuss personal as well as scientific matters: Any idea passing his scepticism is well prepared for entering the world. Ralf—first colleague, later also friend—had the most significant influence on this thesis and also on myself. Many results of this thesis are the results of very enjoyable brain-storming sessions both in front of the blackboard and in various pubs all over the world. Ralf's outstanding skills together with his enthusiasm for life in general and machine learning in particular were a constant source of inspiration, knowledge, energy, and—fun. For a taste of it try Ralf's great new book *Learning Kernel Classifiers* (Herbrich 2001).

During my doctorate I was fortunate enough to progress in the chain of knowledge and to be able to accept the responsibility of advising diploma students. In particular, I would like to acknowledge the work of Mike Goutrie and Marco Krüger who carried out the simulations on the application of machine learning to the game of Go that form parts of Chapter 10. We had many stimulating discussions about Go programming, and their skills and enthusiasm turned the duty of advising them into an exciting joint research project. In the context of machine Go I would also like to thank Nici Schraudolph and Thomas Wolf for providing their databases of Go game records and computer generated Go problems, respectively.

One of the great merits of modern science is its international flavour. On two occasions I was granted the opportunity of prolonged research stays at foreign institutions. I would like to thank Prof. Shun-ichi Amari for his great hospitality during my stay at the RIKEN Brain Science Institute, Tokyo, Japan, in the summer of 1999. Later, in the spring of 2000, Ralf and I had the chance to visit Bob Williamson at the Australian National University in Canberra, Australia. This visit turned out to be an excellent combination of outrageous fun and most productive scientific work, and I would like to thank Bob for making this trip such a great success.

On many occasions I had the opportunity to discuss and work together with machine learning researchers from all over the world. In particular, I enjoyed working with Colin Campbell, Bernhard Schölkopf, John Shawe-Taylor, and Alex Smola. Although we did not work together directly I would like to mention how deeply I was impressed by David McAllester whose PAC-Bayesian theorems lie at the heart of some of the key results of this thesis.

After work I was often magically drawn to the place of my oldest friend Marcus Funke for a match of Backgammon. Ironically, Marcus almost succeeded in shattering my scientific and statistical world view by consistently throwing exactly those dice combinations he needed and thus proving any large deviation bound empirically wrong. Thanks to him for reminding me of the difference between theory and practice.

Finally, I would like to thank my parents just for being there. You are the best.

1. Introduction and Background

1.1. Introduction

The field of *machine learning* (ML) has its roots in the ancient human dream of creating an automaton that is able to think and learn. Realistically today, machine learning aims at developing computer programs that improve their performance automatically by trial and error or by processing examples of preconceived solutions to the task at hand (Mitchell 1997). With the human brain as a constant reminder of what may one day be possible to achieve, the general challenge is so demanding that machine learning naturally draws from such diverse disciplines as psychology, biology, physics, computer science, and statistics. The spectrum of intentions behind the research ranges from practical applications to understanding the fundamentals underlying learning and abstraction. In particular, complementary areas of research are the development of practical learning systems and the theoretical analysis of learning. Ideally, these two areas of machine learning interact and benefit from one another's results.

The field of machine learning can roughly be divided into three major categories: *supervised learning*, *unsupervised learning*, and *learning in dynamic environments*. This distinction reflects the different types of feedback a learning machine receives based on its output or behaviour. Supervised learning comprises tasks such as classification and regression (see, e.g. Bishop (1995)), where the learning machine is provided with input and output data. The problem then is to learn a mapping from input space to output space so as to be able to predict output values for new as yet unseen input data well, according to some specified criterion. Also the problem of ordering things, often referred to as ordinal regression (Herbrich, Graepel, and Obermayer 2000), belongs to this category. In unsupervised learning (see, e.g. Ripley (1996) and Haykin (1994)) the learning machine is provided with only the input data. This task can be looked upon as the detection of presupposed structure within the input data according to some specified criterion. The problems of density estimation (Devroye and Györfi 1985) and probabilistic modelling (Heckerman 1996), dimensionality reduction (Jolliffe 1986), and clustering (Jain and Dubes 1988) are prototypical tasks of unsupervised learning. Learning in dynamic environments is often considered in the framework of reinforcement learning (Sutton and Barto 1998). Conceptually, this task lies somewhere in between supervised and unsupervised learning because the learning machine receives feedback for its actions in the form of a reward (or punishment) signal.

While all these different learning tasks involve various interesting theoretical and practical issues, the question of *generalisation* lies at their very heart. After all, the aim

of learning from examples is to infer an underlying general dependency from particular training data. How such an undertaking may be feasible at all has been an object of philosophical debate for quite some time (Popper 1980; Popper 1981). The question is still influential on the field among those who are not fully satisfied with tackling technicalities (Vapnik 1995). In the context of machine learning the development of learning algorithms can be considered as a practical attempt to solve this longstanding problem. In contrast, learning theory may be seen as an attempt to formalise the problem and thus to make possible a rigorous mathematical analysis.

This thesis primarily deals with the supervised learning task of classification. The reasons are twofold: First, the problem of classification can be thought of as the *drosophila melanogaster* of learning theory—it is simple enough to be studied in detail, yet it reveals many of the problems encountered in the study of learning in general. Secondly, the problem of classification learning is relevant in practice. Problems of classification or categorisation can be found in almost all areas of human intellectual endeavour. Thus it comes as no surprise that applications of machine classification range from the classification of documents (Joachims 1998), via the identification of particles in high-energy physics (Barabino et al. 1999) to the classification of genomic data (Brown et al. 2000), to name just a few.

One natural segmentation of the various results conceivable in the field of machine learning is that into theory, algorithms, and applications. In this thesis I would like to present contributions to these three aspects of the field. The remainder of this introduction will prepare the ground for the results to follow by introducing the basic learning task. All the subsequent chapters—both theoretical and practical—will build on this general framework.

The subsequent two chapters, Chapters 2 and 3, are intended to provide an overview of the relevant background necessary to understand the new results presented in the later chapters. First, Chapter 2 on learning algorithms aims at introducing the fundamentals of kernel-based classification. In particular, we emphasise the conceptual position of kernel-based classification between the paradigms of hypothesis-based classification and example-based classification. The subsequent Chapter 3 on generalisation theory serves to introduce the PAC-Bayesian framework of learning, building on elements of both the PAC/VC framework and the Bayesian framework. The chapter on learning algorithms appears before the one on learning theory because machine learning is in a sense an empirical science, and many developments in learning theory have been a consequence of practical experience with learning algorithms. The order of presentation thus follows from putting observation before modelling.

Following a chiastic pattern the subsequent three chapters, Chapters 4, 5, and 6 constitute the theoretical contribution of the thesis. The analysis emphasises the influence of two particular quantities for the generalisation ability of classifiers: their margin on the training sample and their representational sparseness. Both quantities have been put forward as candidates for explaining the generalisation performance of learning machines such as the support vector machine (Vapnik 1998) and are subject to ongoing debate in the machine learning community. Chapter 4 expands on ideas presented in Herbrich (2001) by thoroughly exploring the geometry of the weight space of linear classifiers under

a uniform prior over normalised weights in the PAC-Bayesian framework. New contributions include the incorporation of apriori knowledge about the input distribution and the consideration of the distribution of margins instead of just the minimal margin itself. Chapter 5 focuses on the complementary issue of sparse representation of classifiers in the compression framework with a focus on a dual representation of classifiers based on the input sample. The main contribution of the chapter lies in the derivation of PAC-Bayesian bounds for the case of data-dependent hypotheses. The third theoretical chapter, Chapter 6, has a somewhat political flavour: What is the fraction of good classifiers among those classifiers that are consistent with the training sample? Answers are provided by what we call egalitarian bounds and by the discovery of the classical perceptron algorithm as an abundant source of sparse classifiers.

The subsequent three chapters, Chapters 7, 8, and 9 make up the practical, algorithmic part of the thesis. Chapter 7 discusses Bayesian learning in kernel space and presents Bayesian transduction as a full-fletched Bayesian approach to the problem of the prediction of labels. The Bayes point machine is shown to be an efficient approximation to the fully Bayesian approach—more amenable to real-world problems than transduction. Both, Bayesian transduction and the Bayes point machine, rely on the ability to sample from the Bayesian posterior over weight space. Efficient sampling schemes to this end are provided in Chapter 8. Experiments based on these sampling schemes are presented in Chapter 9.

Finally, we consider an application of machine learning to a complex domain, the game of Go, in Chapter 10. The Japanese game of strategy serves as an example for the importance of finding a suitable representation for the learning task at hand and is used as a test bed for exploring graph-based representations.

1.2. The Basic Learning Task

Supervised learning aims at discovering and modelling the underlying dependency between two sets commonly referred to as *input space* $\mathcal{X}$ and *output space* $\mathcal{Y}$, which will be jointly referred to as the *input-output* space $\mathcal{Z}$ according to the following definition:

Definition 1 (Input-Output space). We call

1. $\mathcal{X}$ the input space,
2. $\mathcal{Y}$ the output space, and
3. $\mathcal{Z} \stackrel{\text{def}}{=} \mathcal{X} \times \mathcal{Y}$ the joint input–output space

of the learning problem.

Learning is based on a *training sample* $\boldsymbol{z}$ of size m which is assumed to be drawn iid from some unknown probability measure over $\mathcal{Z}$,

$$\mathsf{P}_{\mathsf{Z}} \stackrel{\text{def}}{=} \mathsf{P}_{\mathsf{XY}} .$$

Definition 2 (Training sample). Given an input–output space $\mathcal{Z}$ and a probability measure $\mathbf{P}_{\mathsf{Z}}$ thereon we call an m-tuple

$$\boldsymbol{z} \in \mathcal{Z}^m$$

drawn iid from $\mathbf{P}_{\mathsf{Z}}$ a training sample of size m. Given $\boldsymbol{z} = ((x_1, y_1), \ldots, (x_m, y_m))$ we will call the pairs (x_i, y_i) training examples. Also we use the notation $\boldsymbol{x} = (x_1, \ldots, x_m)$ and similarly $\boldsymbol{y} = (y_1, \ldots, y_m)$.

The goal of supervised learning is to find a functional relationship (or hypothesis) $h : \mathcal{X} \to \mathcal{Y}$ that predicts the output for new, possibly yet unseen test examples $x \in \mathcal{X}$ in the best possible way according to some optimality criterion. The hypotheses considered in learning are contained in the hypothesis space.

Definition 3 (Hypothesis and hypothesis space). Given an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ we call a function

$$h : \mathcal{X} \to \mathcal{Y}$$

an hypothesis and a set

$$\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$$

of hypotheses an hypothesis space.

Various criteria for success in learning are conceivable. We will focus on criteria based on a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that assigns a real-valued *loss* $l(\hat{y}, y)$ to pairs of outputs $(\hat{y}, y)$.

Definition 4 (Loss function). Given an output space $\mathcal{Y}$ we call a function

$$l : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$$

a loss function on $\mathcal{Y}$.[1]

A useful measure of success of a given hypothesis h based on a given loss function l is its (true) *risk* defined as follows:

Definition 5 (Risk functional). Given a loss function l, a hypothesis space $\mathcal{H}$, and a probability measure $\mathbf{P}_{\mathsf{Z}}$ the functional $R : \mathcal{H} \to \mathbb{R}$ given by

$$R[h] \stackrel{\text{def}}{=} \mathbf{E}_{\mathsf{XY}}[l(h(\mathsf{X}), \mathsf{Y})]$$

is called the (true) risk functional on $\mathcal{H}$. Given an hypothesis h we call $R[h]$ its *prediction error*.[2]

We are now in a position to define the basic task of supervised learning:

[1] The loss function l is assumed piecewise continuous such that under a Borel measure on $\mathcal{X} \times \mathcal{Y}$ we have that $l(h(\mathsf{X}), \mathsf{Y})$ is also a random variable.

[2] We assume that $R[h]$ exists.

Definition 6 (Supervised learning task). Given an input space $\mathcal{X}$, an output space $\mathcal{Y}$, a training sample $\boldsymbol{z} \sim \mathbf{P}_{\mathbf{Z}^m}$, an hypothesis space $\mathcal{H}$, and a loss function l, find an hypothesis h such that its risk $R[h]$ is as close as possible to the risk $R[h^*]$ of $h^* = \operatorname{arginf}_{h \in \mathcal{H}} R[h]$.

Now that the task is defined let us consider the procedures that actually lead from training samples to hypotheses: *learning algorithms*.

Definition 7 (Learning algorithm). Given an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ we call a mapping

$$\mathcal{A} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$$

a learning algorithm.

In order to assess the quality of a learning algorithm we can use the following measure.

Definition 8 (Generalisation error of algorithms). Suppose we are given a fixed learning algorithm $\mathcal{A} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{Y}^{\mathcal{X}}$. Then, for any fixed training sample size $m \in \mathbb{N}$ the *generalisation error* $R_m[\mathcal{A}]$ of $\mathcal{A}$ is defined by

$$R_m[\mathcal{A}] \stackrel{\text{def}}{=} \mathbf{E}_{\mathbf{Z}^m}[R[\mathcal{A}(\mathbf{Z})]] ,$$

which is the expected prediction error of the hypotheses found by the algorithm.

Note that for any loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ a small generalisation error $R_m[\mathcal{A}]$ of the algorithm $\mathcal{A}$ guarantees a small prediction error for most randomly drawn training samples $\boldsymbol{z}$ because by Markov's inequality, Theorem 59, we have for all $\varepsilon > 0$,

$$\mathbf{P}_{\mathbf{Z}^m}(R[\mathcal{A}(\mathbf{Z})] > \varepsilon \cdot \mathbf{E}_{\mathbf{Z}^m}[R[\mathcal{A}(\mathbf{Z})]]) \leq \frac{1}{\varepsilon} .$$

Hence we can view $R_m[\mathcal{A}]$ also as a performance measure of $\mathcal{A}$'s hypotheses for randomly drawn training samples $\boldsymbol{z}$.

Of course the main problem lies in the fact that we do not possess knowledge of the probability measure $\mathbf{P}_{\mathbf{Z}}$ and hence are not able to evaluate the risk functional $R[h]$ for a given hypothesis h. However, the iid training sample $\boldsymbol{z}$ makes it possible to at least estimate the risk $R[h]$. This risk estimator is called the empirical risk.

Definition 9 (Empirical risk). Given a training sample $\boldsymbol{z} \sim \mathbf{P}_{\mathbf{Z}^m}$, a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, and an hypothesis $h \in \mathcal{H}$ we call

$$\hat{R}[h, \boldsymbol{z}] \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} l(h(x_i), y_i)$$

the empirical risk of h on $\boldsymbol{z}$. An hypothesis h with $\hat{R}[h, \boldsymbol{z}] = 0$ is called *consistent* with $\boldsymbol{z}$.

According to the law of large numbers (Feller 1950; Feller 1966) for a given hypothesis h the empirical risk $\hat{R}[h, \boldsymbol{z}]$ converges *almost surely* to the true risk $R[h]$ if the latter exists. As a consequence, one of the corner stones of machine learning is the *principle of empirical risk minimisation (ERM)* (Vapnik and Chervonenkis 1971).

Principle of Empirical Risk Minimisation (ERM) Given an hypothesis space $\mathcal{H}$, a loss function l, and a training sample $\boldsymbol{z}$ choose the hypothesis $h \in \mathcal{H}$ that minimises the empirical risk.

A realisation of th ERM principle is given by what will be called an empirical risk minimiser or ERM algorithm defined as follows:

Definition 10 (Empirical risk minimising learning algorithm). We call a learning algorithm with

$$\mathcal{A}(\boldsymbol{z}) \stackrel{\text{def}}{=} \operatorname*{argmin}_{h \in \mathcal{H}} \hat{R}[h, \boldsymbol{z}]$$

an *empirical risk minimising learning algorithm* or for short *ERM algorithm* and denote it by $\mathcal{A}_{\text{erm}}$.

While the almost sure convergence of the empirical risk to the true risk holds for any fixed hypothesis $h \in \mathcal{H}$ this may not be true for the empirical risk minimiser $\mathcal{A}_{\text{erm}}(\boldsymbol{z}) \in \mathcal{H}$ because $\mathcal{A}_{\text{erm}}(\boldsymbol{z})$ itself depends on the random draw of the training sample $\boldsymbol{z}$. This is the motivation in learning theory not to consider just the convergence for fixed hypotheses $h \in \mathcal{H}$ but to consider *convergence uniformly* across all of hypothesis space $\mathcal{H}$, a problem that will be further discussed in Chapter 3.

Let us now turn to the *classification learning task* which is a special type of supervised learning task and will be the main topic of this work.

Definition 11 (Classification learning task). The classification learning task is a supervised learning task with an output space $\mathcal{Y}$ of finite cardinality $|\mathcal{Y}|$. The elements of the output set $\mathcal{Y}$ are referred to as *class labels.*

When considering training samples for classification learning tasks it will be useful to denote by $\mathbf{i}_y(\boldsymbol{z})$ the set of indices of training examples belonging to class y, i.e. $\mathbf{i}_y(\boldsymbol{z}) \stackrel{\text{def}}{=} \{i \in \{1, \ldots, m\} \,|\, (x_i, y_i) \in \boldsymbol{z},\, y_i = y\}$. In relation to the ERM principle it will be of interest to consider those hypotheses with minimal error in the sense of the following definition:

Definition 12 (Version space and generalised version space). Given an hypothesis space $\mathcal{H}$, a loss function l, and a training sample $\boldsymbol{z}$ the subset $V(\boldsymbol{z}) \subseteq \mathcal{H}$ of hypothesis space

$$V(\boldsymbol{z}) \stackrel{\text{def}}{=} \left\{ h \in \mathcal{H} \,\middle|\, \hat{R}[h, \boldsymbol{z}] = \min_{h \in \mathcal{H}} \hat{R}[h, \boldsymbol{z}] \right\}$$

is called generalised version space. If $\min_{h \in \mathcal{H}} \hat{R}[h, \boldsymbol{z}] = 0$ then we call the subset $V(\boldsymbol{z}) \subseteq \mathcal{H}$ simply version space.

Given that $\mathcal{Y}$ has finite cardinality $|\mathcal{Y}|$, i.e. $\mathcal{Y} = \{c_1, \ldots, c_{|\mathcal{Y}|}\}$, the loss function $l : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$ can only take $|\mathcal{Y}|^2$ different values. It is thus conveniently represented by a loss matrix

$$\mathbf{L}_{[i,j]} = l_{ij} \stackrel{\text{def}}{=} l(c_i, c_j) \,.$$

Denoting by $\mathbf{e}_{c_i}$ the i-th $|\mathcal{Y}|$–dimensional unit vector the loss function l can be expressed as

$$l(\hat{y}, y) = \mathbf{e}'_{\hat{y}} \mathbf{L} \mathbf{e}_y .$$

Obviously, it is reasonable to have $l_{ii} = 0$ for all i because correct classifications should not incur a non-zero loss. However, there may be good reasons to allow for different losses incurred by different inter-class confusions.

1. In many real-world problems the loss l_{ij}, $i \neq j$ incurred by a misclassification of class c_i for class c_j may not be the same for all pairs (c_i, c_j) of class labels. As an example, consider the problem of medical diagnosis: Confusing condition c_1 with condition c_2 may have consequences of very different magnitude and severity as compared to confusing condition c_1 with condition c_3 depending on the different options for therapy.

2. Even given that the different inter-class confusions incur the same real loss, i.e. $l_{ij} = l_0 = \text{const.}$ the classification problem may be unbalanced in the sense that the marginal class probabilities $\mathsf{P}_{\mathsf{Y}}(c_i) = \mathsf{E}_{\mathsf{X}}\left[\mathsf{P}_{\mathsf{Y}|\mathsf{X}=x}(c_i)\right]$ vary from class to class. Now, consider the so-called *default classifiers* $h_{c_i}(x) \stackrel{\text{def}}{=} c_i$ that assign every input x to the same class c_i and have a risk of

$$R[h_{c_i}] = \mathsf{E}_{\mathsf{Y}}[l(c_i, \mathsf{Y})] = \sum_{j=1}^{|\mathcal{Y}|} l(c_i, c_j) \mathsf{P}_{\mathsf{Y}}(c_j) .$$

   In order to equalise the risks of the default classifiers we can choose $l(c_i, c_j) = l_0 \left((|\mathcal{Y}| - 1) \mathsf{P}_{\mathsf{Y}}(c_j)\right)^{-1} \mathsf{I}_{i \neq j}$ which results in a constant risk of $R[h_{c_i}] = l_0$.

While, in fact, many real-world problems are multi-class problems in the sense that $|\mathcal{Y}| > 2$ this thesis deals primarily with the simples case of $|\mathcal{Y}| = 2$, the *binary classification learning task.*

Definition 13 (Binary classification learning task). The binary classification learning task is a classification learning task with an output space $\mathcal{Y}$ of cardinality $|\mathcal{Y}| = 2$. For notational convenience the elements of the output space $\mathcal{Y}$ are $\mathcal{Y} = \{-1, +1\}$.

Also, if not otherwise stated we will assume that the loss function takes only two different values and is defined as follows:

Definition 14 (Zero-one loss). Given an output space $\mathcal{Y}$ the zero-one loss $l_{0-1} : \mathcal{Y} \times \mathcal{Y} \to \{0, 1\}$ is defined as

$$l_{0-1}(\hat{y}, y) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{for} \quad \hat{y} = y \\ 1 & \text{for} \quad \hat{y} \neq y \end{cases} .$$

Note that this can also be written as $l_{0-1}(\hat{y}, y) = \mathsf{I}_{\hat{y} \neq y}$, where I is the indicator function.

The appealing property of the zero-one loss is that the corresponding risk $R[h]$ of an hypothesis $h \in \mathcal{H}$ can be identified with the probability of error of that hypothesis,

$$\begin{aligned} R[h] &= \mathbf{E}_{\mathsf{XY}}\left[l_{0-1}\left(h\left(\mathsf{X}\right), \mathsf{Y}\right)\right] \\ &= \mathbf{E}_{\mathsf{XY}}\left[\mathbf{I}_{h(\mathsf{X})\neq\mathsf{Y}}\right] . \\ &= \mathbf{P}_{\mathsf{XY}}\left(h\left(\mathsf{X}\right) \neq \mathsf{Y}\right) , \end{aligned}$$

which constitutes a very natural measure of quality for a classifier h.

Although the restriction of the classification task to two classes may at first sight appear rather severe, it turns out that the more general case of $|\mathcal{Y}| > 2$ can be solved based on the solution of the case $|\mathcal{Y}| = 2$. In particular, let us consider two basic strategies (see, e.g., Weston and Watkins (1999)):

1. **One-against-One.** Learn a binary classifier $h_{(c_i,c_j)}$ for every pair (c_i, c_j) of classes $c_i, c_j \in \mathcal{Y}$, $c_i \neq c_j$ based on a reduced training sample

$$\boldsymbol{z}_{(c_i,c_j)} \stackrel{\text{def}}{=} \{(x_k, y_k) \in \boldsymbol{z} \,|\, y_k = c_i \ \vee \ y_k = c_j\}$$

   containing only those inputs that are labelled either c_i or c_j. A test input $x \in \mathcal{X}$ is classified by combining these $(|\mathcal{Y}| - 1)^2/2$ binary classifiers as follows: Count to which class the ensemble of binary classifiers assigns the input x most often and assign it to that class. Unfortunately, this procedure is not always unique because the voting procedure may produce a tie. In this case other criteria, e.g. the real–valued output of a binary classifier (if available), may be used to break the tie.

2. **One-against-All.** Alternatively, the multi-class problem can be reduced to $|\mathcal{Y}|$ binary classification tasks. In this case learn a binary classifier $h_{(c_i)}$ for every class $c_i \in \mathcal{Y}$ against a new class $\bar{c}$ consisting of the examples from the remaining classes. Again these $|\mathcal{Y}|$ binary classifiers are combined into a multi-class classifier for a test input $x \in \mathcal{X}$ by voting and by some procedure for tie-breaking.

More elaborated methods based on error correcting codes can be found in Allwein et al. (2000) and Dietterich and Bakiri (1995). These procedures may only partly be satisfactory for multi-class problems due to the possible ties in voting. However, they have been shown to work well in practice and they make clear that once a good solution to the binary classification problem is found the multi-class problems can be solved as well. In the remainder of this work the focus will thus be on the task of binary classification learning. First, we will describe basic algorithms for classification learning in Chapter 2. In the subsequent Chapter 3 we will explore mathematical models of learning. Both chapters serve to prepare the grounds for the new results presented in subsequent chapters.

2. Algorithms for Classification Learning

Since Fisher's famous discriminant analysis (Fisher 1936) a plethora of algorithms for classification have been developed (see, e.g. Duda and Hart (1973), Bishop (1995), Ripley (1996), and Herbrich (2001)) for various applications as well as for their own sake. It is the aim of this chapter to give an overview of existing algorithms and to work out some of their conceptual features in order to present kernel classifiers in a reasonable context.

Consider the probabilistic learning scenario as specified in Section 1.2. Assuming that $\mathbf{P}_{\mathsf{XY}}$ is known, the classification task would be rather simple to solve: Calculate the conditional probability $\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}$ using

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y) = \frac{\mathbf{P}_{\mathsf{XY}}(x,y)}{\sum_{y' \in \mathcal{Y}} \mathbf{P}_{\mathsf{XY}}(x,y')} = \frac{\mathbf{P}_{\mathsf{XY}}(x,y)}{\mathbf{P}_{\mathsf{X}}(x)} .$$

Then we have the following theorem (see, e.g., Duda and Hart (1973))

Theorem 1 (Risk minimiser). *Given an input space $\mathcal{X}$, an output space $\mathcal{Y}$ of finite cardinality, a probability measure $\mathbf{P}_{\mathsf{XY}}$, the hypothesis space $\mathcal{H} = \mathcal{Y}^{\mathcal{X}}$, and the zero-one loss l_{0-1} the hypothesis $h^* \in \mathcal{H}$ that minimises the risk $R[h]$ is given by*

$$h^*(x) \stackrel{\text{def}}{=} \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y) ,$$

which is also termed the Bayes classifier. The risk $R[h^*]$ of the Bayes classifier is termed the Bayes risk.

Proof. For the risk $R[h]$ of an hypothesis h we have

$$\begin{aligned} R[h] &= \mathbf{E}_{\mathsf{XY}}[l(h(\mathsf{X}), \mathsf{Y})] \\ &= \mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x}[l(h(\mathsf{X}), \mathsf{Y})]\right] \\ &= \mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x}\left[\mathbf{I}_{h^*(\mathsf{X}) \neq \mathsf{Y}}\right]\right] \\ &= \mathbf{E}_{\mathsf{X}}\left[\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(h(\mathsf{X}) \neq \mathsf{Y})\right] \\ &= \mathbf{E}_{\mathsf{X}}\left[1 - \mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(h(\mathsf{X}) = \mathsf{Y})\right] \end{aligned}$$

For every $x \in \mathcal{X}$ this expression is minimised by $h^*(x) = \operatorname{argmax}_{y \in \mathcal{Y}} \mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y)$. □

The above theorem motivates a simple two step procedure for classification:

1. Estimate the probability measure $\mathbf{P}_{\mathsf{XY}}$.

2. Calculate the Bayes classifier h^*.

Unfortunately, it turns out that it is very difficult to get a good estimate of $\mathbf{P}_{\mathsf{XY}}$ and hence of $\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}$ solely on the basis of the training sample $\boldsymbol{z}$. In order to see this we note that $\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}$ can only reliably be estimated at points x where $\mathbf{P}_{\mathsf{X}}(x)$ can be estimated well. For the sake of generalisation, however, we want to classify new yet unseen points x. In particular, such an estimation is difficult in high-dimensional input spaces. This fact was already known to Fisher (Fisher 1936) who proposed to use linear discriminant functions even for the case that the class-conditional densities $\mathbf{f}_{\mathsf{X}|\mathsf{Y}=y}$ belong to some known parameterised family of densities.

More generally, in order to cope with this problem various strategies have been proposed. These can generally be divided into two categories: example-based approaches that aim at a local estimation of $\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}$ and hypothesis-based approaches that try to select an hypothesis h from a fixed restricted hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$. Note that the approaches of hypothesis-based and example-based classification may be viewed as counterparts of parametric (Lindsey 1996) and non-parametric statistics (Fraser 1957), respectively.

2.1. Hypothesis-Based Classification

Let us first consider learning algorithms that select a classifier from a given hypothesis space $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$.

Definition 15 (Hypothesis-based learning algorithm). Given an input space $\mathcal{X}$, an output space $\mathcal{Y}$, and an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ we call a mapping

$$\mathcal{A} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H}$$

an hypothesis-based learning algorithm.

As will be seen in Chapter 3 it is crucial for effective generalisation that the expressive power of $\mathcal{H}$ is limited. For finite cardinality $\left|\mathcal{Y}^{\mathcal{X}}\right|$ one would therefore often require for the hypothesis space that $|\mathcal{H}| \ll \left|\mathcal{Y}^{\mathcal{X}}\right| = |\mathcal{Y}|^{|\mathcal{X}|}$, thus enforcing what is called an *inductive bias* in Mitchell (1997).

In order to proceed and to formulate concrete learning algorithms we need to be able to describe the inputs $x \in \mathcal{X}$ in some way. To this end let us define *input features*.

Definition 16 (Feature and feature set). Given an input space $\mathcal{X}$ and some set Φ we call a function $\phi : \mathcal{X} \to \Phi$ that assigns an element of Φ to each x a *feature* (also called attribute) and the set Φ its associated *feature set*.

Note that this definition is very general and does not specify the precise nature of Φ. In particular, we can look upon any hypothesis $h \in \mathcal{Y}^{\mathcal{X}}$ as a feature associated with the feature set $\mathcal{Y}$. Of course, this is only natural because the class membership we wish to predict is obviously a feature of an input x. Depending on the nature of the set Φ and any further structure defined on it different types of features can be distinguished.

1. A feature is called *nominal,* if $|\Phi|$ is finite and Φ has no further structure. Comparison of two realisations $\phi(x_1)$ and $\phi(x_2)$ of a nominal feature can only lead to either of two results: $\phi(x_1) = \phi(x_2)$ or $\phi(x_1) \neq \phi(x_2)$. The associated operation on an ensemble of objects is *counting.*

2. A feature is called *ordinal* if some order relation is defined on Φ. Comparison of two realisations $\phi(x_1)$ and $\phi(x_2)$ of the feature may lead to either of three results: $\phi(x_1) = \phi(x_2)$, $\phi(x_1) > \phi(x_2)$, or $\phi(x_1) < \phi(x_2)$. The associated operation on an ensemble of objects is *ordering.*

3. A feature is called *metric* if a metric d is defined on Φ. Comparison of two realisations $\phi(x_1)$ and $\phi(x_2)$ of the feature leads to a real-valued result $d(\phi(x_1), \phi(x_2)) \in [0, \infty)$ as might be the case for the results of quantitative measurement.

Depending on the types of features that are available for a given learning problem different hypothesis spaces $\mathcal{H}$ and associated learning algorithms have been developed. For nominal features various hypothesis spaces such as decision trees and other representations of logical formulae such as DNFs (disjunctive normal forms) and CNFs (conjunctive normal forms) have been examined (Mitchell 1997).

Typical learning algorithms like C4.5 (Quinlan 1993) search these hypothesis spaces in a pre-specified order depending on some measure of complexity of hypotheses and aim at finding a compromise between empirical risk and hypothesis complexity. We will primarily be interested in real-valued metric features.

Definition 17 (Real-valued feature). Given an input space $\mathcal{X}$ we call a function

$$\phi : \mathcal{X} \to \mathbb{R}$$

a *real-valued feature.*

In many learning problems the inputs $x \in \mathcal{X}$ are given in terms of a number of real-valued features and can thus be considered as vectors in a real vector space called the *feature space.*

Definition 18 (Real-valued feature space). Given an input space $\mathcal{X}$ and n features $\phi_i : \mathcal{X} \to \mathbb{R}$ on $\mathcal{X}$ we define a *feature mapping*

$$\boldsymbol{\phi} : \mathcal{X} \to \mathcal{K} \subseteq \mathbb{R}^n$$

and we call $\mathcal{K}$ the *feature space.*

Up to this point we have distinguished between the inputs $x \in \mathcal{X}$ and their *representations* $\boldsymbol{\phi}(x) \in \mathcal{K}$. While it is conceptually important to maintain this distinction objects are generally thought of in terms of their representation. It is thus only natural to follow suit in notation and to abbreviate the input x mapped to feature space by $\boldsymbol{\phi}(x)$ as $\mathbf{x}$, i.e., we use $\mathbf{x} \stackrel{\text{def}}{=} \boldsymbol{\phi}(x)$.

Remark. Although we focus on real-valued features, in practice it is possible to convert nominal and ordinal features of finite feature set cardinality $|\Phi|$ to real-valued features. Consider a feature $\phi : \mathcal{X} \to \{a_1, \ldots, a_{|\Phi|}\}$ with $|\Phi|$ finite. This $|\Phi|$-valued feature can be converted to $|\Phi|$ binary real-valued features $\phi_i : \mathcal{X} \mapsto \{0, 1\}$ given by $\phi_i(x) \stackrel{\text{def}}{=} \mathbf{I}_{\phi(x)=a_i}$. This method is used in text classification (Joachims 1998) where a feature in a bag-of-words representation is given by the occurrence or non-occurrence of a term in a document. We will return to this type of representation in Chapter 10 when we characterise graphs by the occurrence of given subgraphs. Conceptually, however this procedure is satisfactory neither for nominal nor for binary features: For the binary features the resulting hypercube structure in feature space $\mathcal{K}$ appears to be rather unnatural. In addition, if the original feature was ordinally scaled, this structure is typically lost by the above procedure. Finally, the resulting binary features ϕ_i display an odd dependency due to the fact that for any x only one ϕ_i may be non-zero.

2.1.1. Primal Perceptron Learning

Given objects $x \in \mathcal{X}$ with their corresponding representation $\mathbf{x} \in \ell_2^n$ let us consider the hypothesis space of linear classifiers.

Definition 19 (Hypothesis space of linear classfiers). Given an input space $\mathcal{X}$, an output space $\mathcal{Y} = \{-1, +1\}$, and a feature mapping $\boldsymbol{\phi} : \mathcal{X} \mapsto \mathcal{K} \subseteq \ell_2^n$ we define the hypothesis space $\mathcal{H}_\mathcal{K}$ of linear classifiers $h_\mathbf{w} : \mathcal{X} \to \{-1, +1\}$ by[1]

$$\mathcal{H}_\mathcal{K} \stackrel{\text{def}}{=} \{h_\mathbf{w} : x \mapsto \operatorname{sign}(\langle \mathbf{x}, \mathbf{w} \rangle) \,|\, \mathbf{w} \in \mathcal{W}\} .$$

The set $\mathcal{W}$ is the ball

$$\mathcal{W} \stackrel{\text{def}}{=} \left\{ \mathbf{w} \in \mathcal{K} \,\middle|\, \|\mathbf{w}\|^2 = 1 \right\}$$

of weight vectors $\mathbf{w}$ with unit norm, $\|\mathbf{w}\|^2 = 1$. By $\langle \cdot, \cdot \rangle : \mathcal{K} \times \mathcal{K} \to \mathbb{R}$ we denote the inner or scalar product given by $\langle \mathbf{x}, \mathbf{w} \rangle \stackrel{\text{def}}{=} \sum_{i=1}^n x_i w_i$ for all $\mathbf{x}, \mathbf{w} \in \mathcal{K}$ and by $\|\cdot\| : \mathcal{K} \to [0, \infty)$ we denote the Euclidean norm $\|\mathbf{w}\| \stackrel{\text{def}}{=} \sqrt{\langle \mathbf{w}, \mathbf{w} \rangle}$ for all $\mathbf{w} \in \mathcal{K}$.

Remark. Note that the normalisation condition $\|\mathbf{w}\|^2 = 1$ in Definition 19 does not actually restrict the hypothesis space $\mathcal{H}_\mathcal{K}$ because we have for all $x \in \mathcal{X}$, for all $\mathbf{w} \in \mathcal{K}$, and for all $c \in \mathbb{R}^+$

$$h_{c\mathbf{w}}(x) = \operatorname{sign}(\langle \mathbf{x}, c\mathbf{w} \rangle) = \operatorname{sign}(\langle \mathbf{x}, \mathbf{w} \rangle) = h_\mathbf{w}(x) .$$

Choosing $c = \|\mathbf{w}\|^{-1}$ by default we achieve a one-to-one mapping of hypotheses h and weight vectors $\mathbf{w}$. Of course, this is also true for inputs $\mathbf{x}$ mapped to feature space: For

[1] Note the special case that $\operatorname{sign}(\langle \mathbf{x}, \mathbf{w} \rangle) = 0$ with the usual definition of the sign function. Two solutions to this problems are possible. Either the sign function is redefined asymmetrically such that $\operatorname{sign}(0) = 1$ (which we assume here and in the following) or we consider an output space $\mathcal{Y} = (-1, 0, +1)$ and redefine the loss such that $l(0, -1) = 1$ and $l(0, +1) = 1$, i.e., such that a vanishing output is always considered as an error.

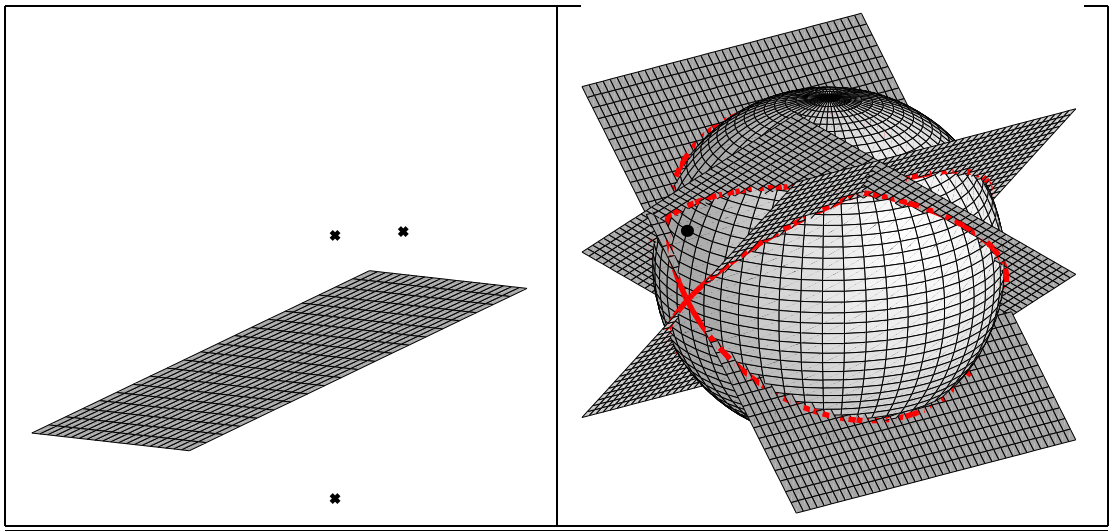


Figure 2.1.: Duality of feature space and hypothesis space for linear classifiers. **Feature space** (left): Three inputs $x_1, x_2, x_3 \in \mathcal{X}$ are represented as vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathcal{K}$ in feature space. An hypothesis $h_{\mathbf{w}} \in \mathcal{H}_{\mathcal{K}}$ associated with a weight vector $\mathbf{w} \in \mathcal{K}$ describes a plane $\{\mathbf{x} \in \mathcal{K} \mid \langle \mathbf{x}, \mathbf{w} \rangle = 0\}$ in feature space separating points in one half-space $\{\mathbf{x} \in \mathcal{K} \mid \langle \mathbf{x}, \mathbf{w} \rangle > 0\}$ from those in the other half-space $\{\mathbf{x} \in \mathcal{K} \mid \langle \mathbf{x}, \mathbf{w} \rangle < 0\}$. **Hypothesis space** (right): The hypothesis space $\mathcal{H}_{\mathcal{K}}$ can be visualised as the surface $\mathcal{W} = \{\mathbf{w} \in \mathcal{K} \mid \|\mathbf{w}\|^2 = 1\}$ of a sphere. Each input x can be thought of as a plane $\{\mathbf{w} \in \mathcal{K} \mid \langle \mathbf{x}, \mathbf{w} \rangle = 0\}$ separating hypotheses in one half-space $\{\mathbf{w} \in \mathcal{K} \mid \langle \mathbf{x}, \mathbf{w} \rangle > 0\}$ from those in the other half-space $\{\mathbf{w} \in \mathcal{K} \mid \langle \mathbf{x}, \mathbf{w} \rangle < 0\}$. Version space $V(\boldsymbol{z}) \subset \mathcal{H}_{\mathcal{K}}$ is the spherical triangle on the left side of the image. It contains the weight vector $\mathbf{w}$ depicted as a black dot corresponding to the separating hyperplane on the left.

all $\mathbf{x}, \mathbf{w} \in \mathcal{K}$ and for all $c \in \mathbb{R}^+$ we have

$$\text{sign}\left(\langle c\mathbf{x}, \mathbf{w}\rangle\right) = \text{sign}\left(\langle \mathbf{x}, \mathbf{w}\rangle\right) .$$

Thus under this hypothesis space we can normalise our feature vectors $\mathbf{x}$ without loss of generality.

As it stands the hypothesis space $\mathcal{H}_\mathcal{K}$ of linear classifiers is limited to hyperplanes $\{\mathbf{x} \in \mathcal{K} : \langle \mathbf{x}, \mathbf{w}\rangle = 0\}$ passing through the origin. While this restriction may not be of great concern in high-dimensional feature spaces it may nevertheless be desirable to introduce a threshold $w_0 \neq 0$. This is easily achieved by adding a constant feature $\phi_0(x) = 1$ and introducing a corresponding component w_0 of the weight vector.

Given the hypothesis space $\mathcal{H}_\mathcal{K}$ of linear classifiers let us now consider the associated learning problem of finding a good hypothesis $h \in \mathcal{H}_\mathcal{K}$ given a training sample $\boldsymbol{z}$. The principle of empirical risk minimisation recommends to look for the hypothesis $h^* \in \mathcal{H}_\mathcal{K}$ with minimal empirical risk. As it turns out we have to distinguish between the case of $\hat{R}[h^*, \boldsymbol{z}] = 0$ and the case of $\hat{R}[h^*, \boldsymbol{z}] \neq 0$.

Definition 20 (Linear separability). We call a training sample $\boldsymbol{z}$ linearly separable if there exists an hypothesis $h \in \mathcal{H}_\mathcal{K}$ (a hyperplane) such that

$$\hat{R}[h, \boldsymbol{z}] = 0 ,$$

that is, if there exists a version space $V(\boldsymbol{z}) \subseteq \mathcal{H}_\mathcal{K}$.

Remark. If a training sample is linearly separable, then there may exist infinitely many hypotheses $h \in \mathcal{H}_\mathcal{K}$ with empirical risk $\hat{R}[h, \boldsymbol{z}] = 0$. Thus the application of the ERM principle does not lead to a unique solution of the learning problem.

Drawing on further learning theoretical considerations we will later see that among the solutions with minimal empirical risk we might have reason to single out specific solutions such as the maximum margin solution (Boser et al. 1992) or the Bayes point (Herbrich et al. 2001).

In the case of linearly separable training data $\boldsymbol{z}$ there exists a variety of different learning procedures (an excellent overview is given in Duda et al. (2001)). Possibly the simplest such procedure is the online perceptron algorithm with fixed increment which was suggested by Rosenblatt around 1960 (Rosenblatt 1958; Rosenblatt 1962). The algorithm proceeds as follows (pseudo code of the algorithm can be found in Appendix C, Algorithm 1).

1. Initialise the weight vector $\mathbf{w}_0 = \mathbf{0}$. Set $t = 0$.
2. Cycle through the training sample $\boldsymbol{z}$ and check for every training example $z_i = (x_i, y_i)$ if it is correctly classified by the current weight vector $\mathbf{w}_t$.
   a) if $y_i \langle \mathbf{x}_i, \mathbf{w}_t\rangle \leq 0$ (wrong classification) then update the weight vector $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$ and proceed with $t \leftarrow t + 1$.
   b) if $y_i \langle \mathbf{x}_i, \mathbf{w}_t\rangle > 0$ (correct classification) then proceed.

3. Repeat 2. until all training examples in $\boldsymbol{z}$ are classified correctly.

Note that although we consider the perceptron algorithm as a learning algorithm it is in a more general sense an algorithm for solving systems of linear inequalities. The procedure is called *mistake driven* because only the occurrence of a classification mistake leads to an update of the weight vector. During learning, the length $\|\mathbf{w}\|$ of the weight vector is not kept at $\|\mathbf{w}\| = 1$ and thus strictly speaking the resulting classifiers $h_{\mathbf{w}}$ are not elements of $\mathcal{H}_{\mathcal{K}}$. However, since normalisation does not change the resulting classification given by a linear classifier, we can always normalise $\mathbf{w}$ after learning using $\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\|$ and thus get back to $\mathcal{H}_{\mathcal{K}}$.

2.1.2. Minover Heuristic and Margin Enforcement for Perceptron

Before we consider the convergence properties of the perceptron algorithm let us introduce two useful modifications:

1. The perceptron algorithm assumes a fixed order of the training examples in $\boldsymbol{z}$ and cycles through the data in that order. As an alternative, one can always select that training example z_i whose input vector $\mathbf{x}_i$ has the smallest overlap with the current weight vector, i.e., one chooses training example z_i with $i(t) = \operatorname{argmin}_{i \in \{1,\ldots,m\}} y_i \langle \mathbf{x}_i, \mathbf{w}_t \rangle$, a procedure referred to as the *minover heuristic*. The intuition behind this heuristic is that one should aim at selecting those training examples for updating first that would change the direction of the current weight vector the most.

2. The perceptron algorithm performs an update of the weight vector only if a classification error occurs. Instead, the condition for a weight update can be replaced by $y_i \langle \mathbf{x}_i, \mathbf{w}_t \rangle \leq \rho^2$. This modification has the effect that correct classification by a certain margin is enforced, a condition that will later be discussed in the context of margin bounds (see Subsection 3.1.3 and Chapter 4) and the maximum margin perceptron (see Subsection 2.1.3).

The resulting minover algorithm (Guyon and Storck 2000) with enforced margin ρ^2 is given by the following steps.

1. Using some randomly chosen training example $z_i = (x_i, y_i)$ initialise the weight vector at $\mathbf{w}_0 = y_i \mathbf{x}_i$. Set $t = 0$.

2. Go through the training sample $\boldsymbol{z}$ and choose training example $z_{i(t)} = \left(x_{i(t)}, y_{i(t)}\right)$ such that $i(t) = \operatorname{argmin}_{i \in \{1,\ldots,m\}} y_i \langle \mathbf{x}_i, \mathbf{w}_t \rangle$.

   a) if $y_{i(t)} \left\langle \mathbf{x}_{i(t)}, \mathbf{w}_t \right\rangle \leq \rho^2$ (margin error w.r.t. ρ^2) then update the weight vector $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{i(t)} \mathbf{x}_{i(t)}$ and proceed with $t \leftarrow t + 1$.
   b) if $y_{i(t)} \left\langle \mathbf{x}_{i(t)}, \mathbf{w}_t \right\rangle > \rho^2$ (correct by at least ρ^2) then proceed.

3. Repeat 2. until all training examples in $\boldsymbol{z}$ are classified correctly with margin ρ^2.

In order to give a result on the number of update steps until convergence we need to introduce two quantities which together quantify the linear separability of the training sample. In order to state the result we need to introduce a quantity called the margin.

Definition 21 ((Geometric) margin). Given a training example z_i, a training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, a feature mapping $\boldsymbol{\phi}$, and a weight vector $\mathbf{w} \in \mathcal{K}$ we call

$$\gamma(\mathbf{w}, z_i) \stackrel{\text{def}}{=} \frac{y_i \langle \mathbf{x}, \mathbf{w} \rangle}{\|\mathbf{w}\|}$$

the (*geometric*) *margin* of weight vector $\mathbf{w}$ on training example z_i, and

$$\gamma(\mathbf{w}, \boldsymbol{z}) \stackrel{\text{def}}{=} \min_{z_i \in \boldsymbol{z}} \gamma(\mathbf{w}, z_i)$$

the (*geometric*) *margin* of weight vector $\mathbf{w}$ on training sample $\boldsymbol{z}$.

Since the margin $\gamma(\mathbf{w}, \boldsymbol{z})$ is not scale invariant it has to be considered w.r.t. a quantity that describes the extent of the data: the data radius.

Definition 22 (Data radius). Given a training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$ we call the quantity

$$\varsigma(\boldsymbol{x}) \stackrel{\text{def}}{=} \max_{x_i \in \boldsymbol{x}} \|\mathbf{x}_i\|$$

the data radius.

It turns out that the number of update steps a perceptron takes until convergence can be bounded in terms of margin and data radius (Rosenblatt 1958; Rosenblatt 1962).

Theorem 2 (Margin perceptron convergence). *Given a training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$ such that the data radius is $\varsigma \equiv \varsigma(\boldsymbol{x})$ and such that there exists a weight vector $\mathbf{w}^* \in \mathcal{K}$ with margin $\gamma^* \equiv \gamma(\mathbf{w}^*, \boldsymbol{z}) > 0$ the perceptron algorithm with enforced margin ρ^2 terminates after at most M update steps with*

$$M = \frac{2\rho^2 + \varsigma^2}{\gamma^{*2}},$$

and the margin $\gamma(\mathbf{w}_M, \boldsymbol{z})$ of the solution $\mathbf{w}_M$ on the training sample is at least

$$\gamma(\mathbf{w}_M, \boldsymbol{z}) \geq \frac{\rho^2}{2\rho^2 + \varsigma^2}\gamma^*.$$

The proof of the perceptron convergence theorem can be found in Appendix A.1. Let us consider variants of the perceptron algorithm with different enforced margin ρ^2 in the light of the theorem. Consider three cases:

1. For an enforced margin of $\rho^2 = 0$ we obtain the standard perceptron convergence result $M \leq \varsigma^2 \gamma^{*-2}$ but cannot give a lower bound greater zero on the obtained margin $\gamma(\mathbf{w}_M, \boldsymbol{z})$.

2. For an enforced margin of $\rho^2 > 0$ the bound on the number of updates deteriorates. This is very plausible because the size of the target region is effectively shrunk by enforcing a margin $\rho^2 > 0$. However, we can now guarantee that the margin $\gamma(\mathbf{w}_M, \boldsymbol{z})$ of the obtained classifier $\mathbf{w}_M$ has at least a certain value which is proportional to γ^* and is an increasing function of the enforced margin ρ^2.

3. For the case $\rho^2 \to \infty$ the number M of update steps becomes unbounded. However, the lower bound on the obtained margin $\gamma(\mathbf{w}_M, \boldsymbol{z})$ approaches $\frac{\gamma^*}{2}$, i.e., we have a guarantee for a margin within a factor of 2 of the maximum.

One way of understanding the perceptron convergence theorem is to consider the perceptron algorithm as a search algorithm in hypothesis space. The version space or feasible region $V(\boldsymbol{z}) \subseteq \mathcal{H}_\mathcal{K}$ is the target area of the search. Then the margin γ^* can be considered a measure of the size of version space $V(\boldsymbol{z})$ as compared to the size of the whole hypothesis space $\mathcal{H}_\mathcal{K}$. The theorem indicates that the larger the target area of the search is the fewer steps are needed to find it. Additionally enforcing a margin ρ corresponds to a relative decrease in size of the target area and thus leads to an increase of the number of steps required to find it. We will return to the perceptron convergence theorem in Chapter 6.

Various other approaches to perceptron learning exist varying from least-square to linear programming approaches (see Duda et al. (2001) for an extensive overview). For this work it will be sufficient to consider one more approach to perceptron learning: the *maximum stability* or *maximum margin* perceptron. This approach is particularly elegant and has recently received possibly more attention than it deserves in the context of *support vector machines (SVMs)*.

2.1.3. The Maximum Margin Perceptron

As was demonstrated in the context of the perceptron convergence theorem, the existence of a classifier $h_{\mathbf{w}^*}$ with large margin $\gamma^* = \gamma(\mathbf{w}^*, \boldsymbol{z}) > 0$ on the training sample $\boldsymbol{z}$ is a crucial quantity for determining the speed of convergence of the perceptron algorithm. As will be seen in Subsection 3.1.3 and Chapter 4 learning theoretical results show that the risk of a classifier $h_\mathbf{w}$ can be bounded in terms of its own margin $\gamma(\mathbf{w}, \boldsymbol{z})$, indicating that one should aim at finding a classifier $h_\mathbf{w}$ with maximal margin. For a more intuitive explanation of the robustness of the maximum margin hyperplane classifier consider Figure 2.2: The classification of the maximum margin hyperplane is most robust w.r.t. perturbations of the input points orthogonal to the hyperplane. In the following we will explore the maximum margin classifier and its properties in some detail. The reason is that the maximum margin classifier lies at the heart of the SVM, whose invention initiated the recent boost of interest in kernel classifiers. In addition, the maximum margin classifier helps to introduce important concepts with regard to kernel methods and will serve as a base line algorithm for the remainder of this thesis.

For a linearly separable training sample $\boldsymbol{z}$, the optimisation problem of the maximum margin classifier can be stated as maximising the margin $\gamma(\mathbf{w}, \boldsymbol{z})$.

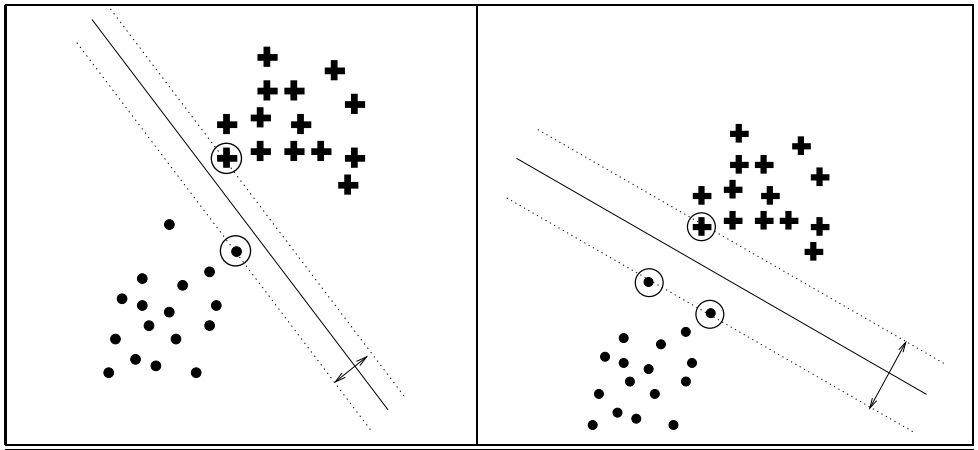


Figure 2.2.: Illustration of the maximum margin-hyperplane: Shown are inputs of two different classes (dots and crosses), separated by an arbitrary hyperplane (left) and by the maximum margin hyperplane (right). Intuitively, the maximum margin hyperplane on the right is more stable because its resulting classification is more robust w.r.t. perturbations of the input points orthogonal to the hyperplane. The distance between the two outer dotted lines is twice the margin, i.e., 2γ. The circled inputs are called "support vectors".

Definition 23 (Maximum margin hyperplane). Given a linearly separable training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$, we define the maximum margin hyperplane as the solution of the optimisation problem

$$\text{maximise}_{\mathbf{w}\in\mathbb{R}^n, w_0\in\mathbb{R}}\ \gamma\left(\mathbf{w}, \boldsymbol{z}\right) = \min_{z_i\in\boldsymbol{z}} \frac{y_i\left(\langle \mathbf{x}_i, \mathbf{w}\rangle + w_0\right)}{\|\mathbf{w}\|}\,.$$

Note that we explicitly reintroduce the bias term w_0 here because it makes the optimisation problem shift-invariant. In order to obtain a convex optimisation problem we fix the minimal real-valued output of the classifier to 1, i.e., we require for all $i \in \{1, \ldots, m\}$ that

$$y_i\left(\langle \mathbf{x}_i, \mathbf{w}\rangle + w_0\right) \geq 1\,. \tag{2.1}$$

Then for a weight vector $\mathbf{w}$ on a training sample $\boldsymbol{z}$ with training examples $(\mathbf{x}^+, +1) \in \boldsymbol{z}$ and $(\mathbf{x}^-, -1) \in \boldsymbol{z}$ that meet (2.1) as an equality the margin is given by

$$\gamma\left(\mathbf{w}, \boldsymbol{z}\right) = \frac{1}{2}\left(\frac{\langle \mathbf{x}^+, \mathbf{w}\rangle + w_0}{\|\mathbf{w}\|} - \frac{\langle \mathbf{x}^-, \mathbf{w}\rangle + w_0}{\|\mathbf{w}\|}\right) = \frac{1}{\|\mathbf{w}\|}$$

Thus we can find the maximum margin hyperplane by solving a quadratic programme according to the following theorem.

Theorem 3 (Maximum margin hyperplane quadratic programme). *Given a linearly separable training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$, the*

quadratic programme

$$\text{minimise}_{\mathbf{w}} \quad \|\mathbf{w}\|^2 \tag{2.2}$$

$$\text{subject to} \quad \forall i \in \{1,\ldots,m\}: \quad y_i \left(\langle \mathbf{x}_i, \mathbf{w}\rangle + w_0\right) \geq 1 \tag{2.3}$$

yields the maximum margin hyperplane $\mathbf{w}^*$ *with margin* $\gamma = \|\mathbf{w}^*\|^{-1}$.

It turns out that this problem can be solved by transforming it into its corresponding dual formulation.

Theorem 4 (Dual of maximum margin hyperplane quadratic programme). *Given a linearly separable training sample* $\boldsymbol{z}$*, a feature space* $\mathcal{K}$*, a feature mapping* $\boldsymbol{\phi}$ *and a solution* $\boldsymbol{\alpha}^*$ *of the quadratic programme*

$$\begin{aligned} \text{minimise}_{\boldsymbol{\alpha}} \quad & W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i \alpha_i = 0 \\ & \forall i \in \{1,\ldots,m\}: \quad \alpha_i \geq 0 \end{aligned}$$

then the weight vector $\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i$ *yields the maximum margin hyperplane with margin* $\gamma = \|\mathbf{w}^*\|^{-1}$.

The proof of this theorem can be found in Appendix A.2. Theorem 4 leads to the following interesting observations. The training data enter the optimisation problem only in terms of their inner product or Gram matrix.

Definition 24 (Gram matrix). Given an input sample $\boldsymbol{x}$, a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$ the $m \times m$-matrix $\mathbf{K}$ with elements

$$\mathbf{K}_{ij} \stackrel{\text{def}}{=} \langle \mathbf{x}_i, \mathbf{x}_j\rangle \ ,$$

is called the *Gram or kernel matrix.*

This property indicates that the optimisation problem is invariant under orthogonal transformations (rotation, mirroring). In Subsection 2.3 this feature will be exploited to generalise the maximum margin hyperplane to non-linear function classes by the application of the so-called *kernel trick* leading to the well-known support vector machine (SVM).

The solution weight vector $\mathbf{w}^*$ can be described as a linear combination of training examples

$$\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i \, . \tag{2.4}$$

This will turn out to be a crucial condition for the application of the kernel trick to be discussed in Subsection 2.3. The value of w_0 does not appear in the dual formulation. We can determine w_0, however, from the primal constraints according to

$$w_0^* = -\frac{1}{2}\left(\max_{y_i=-1}\langle \mathbf{x}_i, \mathbf{w}^*\rangle + \min_{y_i=+1}\langle \mathbf{x}_i, \mathbf{w}^*\rangle\right).$$

We find from the Karush-Kuhn-Tucker conditions that the optimal solutions $\boldsymbol{\alpha}^*$, $(\mathbf{w}^*, w_0^*)$ satisfy for all $i \in \{1, \ldots, m\}$ that

$$\alpha_i^* \left[y_i \left(\langle \mathbf{x}_i, \mathbf{w}^*\rangle + w_0^*\right) - 1\right] = 0\,. \tag{2.5}$$

These equations imply that all those training examples z_i have coefficients $\alpha_i^* = 0$ for which

$$y_i \left(\langle \mathbf{x}_i, \mathbf{w}^*\rangle + w_0^*\right) > 1\,. \tag{2.6}$$

Those inputs with $\alpha_i^* \neq 0$ are termed "support vectors" and have the property that they satisfy the constraints (2.3) with equality.

Definition 25 (Support vectors). Given the expansion (2.4) of the weight vector $\mathbf{w}^*$ we call those vectors $\mathbf{x}_i$ *support vectors,* for which $\alpha_i^* \neq 0$. We call $\mathbf{i}_{\text{sv}}(\boldsymbol{\alpha}^*)$ the set of indices of support vectors

$$\mathbf{i}_{\text{sv}}(\boldsymbol{\alpha}^*) \stackrel{\text{def}}{=} \{i \in \{1, \ldots, m\} \mid \alpha_i^* \neq 0\}$$

Often the number $|\mathbf{i}_{\text{sv}}(\boldsymbol{\alpha}^*)|$ of support vectors is much less than the number m of training examples, $|\mathbf{i}_{\text{sv}}(\boldsymbol{\alpha}^*)| \ll m$, resulting in a so-called sparse solution. This sparsity has implications for both computational complexity and generalisation properties. While the latter will be discussed in depth in Chapter 5 the reduction in computational complexity is easily seen when considering the computational costs associated with the evaluation of an SVM classifier at a given test input $\mathbf{x}$,

$$h_{\boldsymbol{\alpha}^*}(\mathbf{x}) = \operatorname{sign}\left(\sum_{i \in \mathbf{i}_{\text{sv}}(\boldsymbol{\alpha}^*)} \alpha_i^* y_i \langle \mathbf{x}_i, \mathbf{x}\rangle\right)$$

which are proportional to $|\mathbf{i}_{\text{sv}}(\boldsymbol{\alpha}^*)|$ rather than m and make possible a faster classification than would be possible with a "dense" classifier.

Finally, we should note that the margin $\gamma(\mathbf{w}^*, \boldsymbol{z})$ of the maximum margin hyperplane $\mathbf{w}^*$ is easily calculated using $\gamma(\mathbf{w}^*, \boldsymbol{z}) = \|\mathbf{w}^*\|^{-1}$, the linear expansion (2.4), and the Karush-Kuhn-Tucker conditions (2.5)

$$\gamma(\mathbf{w}^*, \boldsymbol{z}) = \|\boldsymbol{\alpha}^*\|_1^{-\frac{1}{2}}\,.$$

Up to this point the maximum margin classifier has been discussed under the assumption of a linearly separable training sample. This assumption is realistic for problems in which the dimensionality n of the feature space $\mathcal{K}$ is greater than the number m of

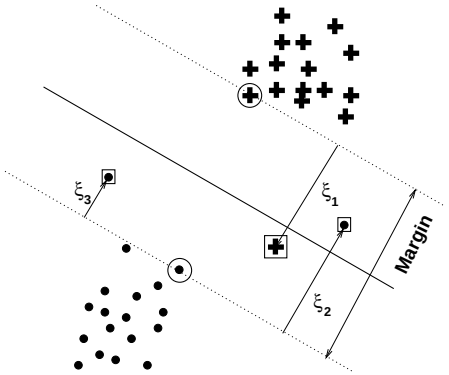


Figure 2.3.: Illustration of the soft-margin hyperplane. Shown are two clouds of data points corresponding to two different classes (dot and plus) in a binary classification problem. The two classes are separated by a large-margin hyperplane whose margin is delimited by support vectors (circled). Data points $\mathbf{x}_i$ falling into the margin region have non-zero slack-variables ξ_i indicating by how much the constraints are violated. Thus a large-margin solution is obtained although the data are not even linearly separable.

training samples, $n \gg m$, as is, for example, the case in problems of text categorisation with a bag-of-words representation (Joachims 1998). In the more usual case of $n \ll m$, however, linear separability of the training sample can in general not be expected. For the maximum margin classifier we will consider two possible ways of dealing with this problem (Cortes and Vapnik 1995).

In the context of optimisation the problem of non-separable data can be expressed as the non-existence of a feasible region. This is the case if the primal constraints (2.3) cannot be satisfied. In order to deal with this problem the constraints are relaxed using so-called slack-variables ξ_i leading to the soft-margin classifier.

Definition 26 (Soft margin hyperplane). Given a training sample $\boldsymbol{z}$ (not necessarily linearly separable), a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$, we define the p-norm ($p \in \{1, 2\}$) soft-margin hyperplane with penalisation parameter C as the solution of the optimisation problem

$$\begin{aligned} &\text{minimise}_{\mathbf{w}} && \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i^p \\ &\text{subject to} \quad \forall i \in \{1, \ldots, m\} && y_i \left(\langle \mathbf{x}_i, \mathbf{w} \rangle + w_0 \right) \geq 1 - \xi_i \\ & && \xi_i \geq 0 \end{aligned}$$

Let us consider the two cases $p = 1$ and $p = 2$ in turn by deriving the respective dual formulations of the optimisation problems. The results will be given in terms of the following two theorems whose proofs can be found in the Appendix. For the case $p = 1$ we have the following theorem (proof in Appendix A.3):

Theorem 5 (1-norm soft margin dual). *Given a training sample $\boldsymbol{z}$ (not necessarily linearly separable), a feature space $\mathcal{K}$, a feature mapping $\boldsymbol{\phi}$ and a solution $\boldsymbol{\alpha}^*$ of the quadratic programme*

$$\begin{aligned} \text{minimise}_{\boldsymbol{\alpha}} \quad & W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i \alpha_i = 0 \\ & \forall i \in \{1, \ldots, m\}: \quad C \geq \alpha_i \geq 0\,, \end{aligned}$$

then the weight vector $\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i$ together with w_0^* chosen such that $\forall i \in \mathbf{i}_{\text{sv}}(\boldsymbol{\alpha}^*)$ we have $y_i(\langle \mathbf{x}_i, \mathbf{w}^* \rangle + w_0^*) = 1$, yields the 1-norm soft-margin hyperplane.

The theorem shows that the optimisation problem essentially corresponds to the optimisation problem of Theorem 4 with the only difference that we now have constraints $C \geq \alpha_i \geq 0$ instead of $\alpha_i \geq 0$ for all $i \in \{1, \ldots, m\}$. These constraints on the dual variables α_i are referred to as *box constraints* because the dual parameter vector $\boldsymbol{\alpha}$ is restricted to a box of size C. Interpreting the size of coefficient α_i as indicating the contribution of training example z_i to the solution of the optimisation problem the box constraints can be seen as putting a limit on the contribution of individual training examples.

The following theorem gives the dual formulation of the soft-margin classifier with $p = 2$ (proof in Appendix A.4):

Theorem 6 (2-norm soft margin dual). *Given a training sample $\boldsymbol{z}$ (not necessarily linearly separable), a feature space $\mathcal{K}$, a feature mapping $\boldsymbol{\phi}$ and a solution $\boldsymbol{\alpha}^*$ of the quadratic programme*

$$\begin{aligned} \text{minimise}_{\boldsymbol{\alpha}} \quad & W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \left(\langle \mathbf{x}_i, \mathbf{x}_j \rangle + \frac{1}{C} \mathbf{I}_{i=j} \right) \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i \alpha_i = 0 \\ & \forall i \in \{1, \ldots, m\}: \quad \alpha_i \geq 0\,, \end{aligned}$$

then the weight vector $\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i$ together with w_0^* chosen such that $\forall i \in \mathbf{i}_{\text{sv}}(\boldsymbol{\alpha}^*)$ we have $y_i(\langle \mathbf{x}_i, \mathbf{w}^* \rangle + w_0^*) = 1 - \frac{\alpha_i^*}{C}$, yields the 2-norm soft-margin hyperplane with margin

$$\gamma = \left(\|\boldsymbol{\alpha}^*\|_1 - \frac{1}{C} \|\boldsymbol{\alpha}^*\|_2^2 \right)^{-\frac{1}{2}}.$$

Interestingly, the optimisation problem again essentially corresponds to the hard-margin case, Theorem 4, with the difference that the Gram matrix $\mathbf{G}$ is replaced by a modified Gram matrix $\tilde{\mathbf{G}}$ by adding a constant term $\frac{1}{C}$ to its diagonal,

$$\tilde{\mathbf{G}} = \mathbf{G} + \frac{1}{C} \mathbf{I}_m\,,$$

where $\mathbf{I}_m$ is the $m \times m$-identity matrix. Due to this simple modification of the Gram matrix the resulting algorithm is conceptually simpler and easier to analyse than the 1-norm soft margin formulation (Shawe-Taylor and Cristianini 2000a). Both formulations, however, have the advantage that they can be cast as QP problems and be solved by the application of standard software for solving large QP problems. With the idea of a dual formulation of optimisation problems in mind let us now turn to the concept of example-based classification.

2.2. Example-Based Classification

Hypothesis-based learning algorithms select an appropriate global hypothesis from a fixed hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ given a priori according to an inductive principle such as the principle of empirical risk minimisation (ERM). In contrast, example-based classification takes a more local approach. It is based on a simple yet intuitive principle:

Principle of Similarity: Given an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ similar inputs $x \in \mathcal{X}$ should be labelled by similar outputs $y \in \mathcal{Y}$.

In order to apply this principle it is necessary to conceive of some measure of similarity or dissimilarity in both input space $\mathcal{X}$ and output space $\mathcal{Y}$. It turns out to be a very fruitful and powerful approach to transfer the problem of inference again to the domain of geometry as was done in the previous section by introducing a feature space (see Definition 16). Let us thus restrict our discussion to the class of *metric* distance measures, defined as follows:

Definition 27 (Metric). Given a set $\mathcal{X}$ we call a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ a *metric* if we have for all $x_1, x_2, x_3 \in \mathcal{X}$

1. $d(x_1, x_2) = d(x_2, x_1)$ (Symmetry)
2. $d(x_1, x_2) \geq 0$ with $d(x_1, x_2) = 0 \implies x_1 = x_2$ (Definiteness)
3. $d(x_1, x_2) + d(x_2, x_3) \geq d(x_1, x_3)$ (Triangle Inequality)

This restriction is useful because the definition of a metric captures the essential properties of both spatial distance as well as conceptual dissimilarity. Note that the zero-one loss $l_{0-1} : \mathcal{Y} \times \mathcal{Y} \to \{0, 1\}$ constitutes a metric: It satisfies symmetry as well as definiteness and the triangle inequality as is easily verified for $|\mathcal{Y}|$ finite simply by checking the three conditions for all combinations of arguments.

Suppose we were provided with a metric perfectly suited to the classification task at hand. From such a distance measure we might expect that the distance between two objects of the same class is small and the the distance between two objects of different classes is large. More formally, given a binary classification learning task as defined in Definition 13 and a positive constant $\hat{d}$ we might require from a suitable distance measure d that

$$\mathbf{P}_{(\mathsf{XY})^2}\left(\mathsf{Y}_1 = \mathsf{Y}_2 \;\Rightarrow\; d(\mathsf{X}_1, \mathsf{X}_2) < \hat{d}\right) = 1$$

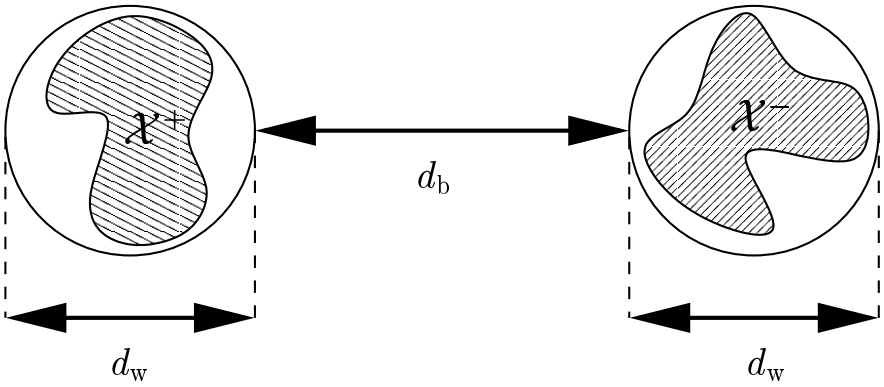


Figure 2.4.: Illustration of a perfect distance measure in input space $\mathcal{X}$ for classification learning. Let $\mathcal{X}^+$ be the support of $\mathbf{P}_{\mathsf{X}|\mathsf{Y}=+1}$ and $\mathcal{X}^-$ the support of $\mathbf{P}_{\mathsf{X}|\mathsf{Y}=-1}$. In this case the two classes are ideally separated w.r.t. the distance measure d which in this illustration is given by the Euclidean distance $d(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|$. The within-class distances are at most d_w and the between-class distances are at least d_b such that $\hat{d}$ with $d_\mathrm{w} < \hat{d} < d_\mathrm{b}$ exists.

and

$$\mathbf{P}_{(\mathsf{XY})^2}\left(\mathsf{Y}_1 \neq \mathsf{Y}_2 \;\Rightarrow\; d(\mathsf{X}_1, \mathsf{X}_2) > \hat{d}\right) = 1\,,$$

accordingly. This situation is depicted in Figure 2.4 where the supports $\mathcal{X}^+$ and $\mathcal{X}^-$ of two class conditional probability measures $\mathbf{P}_{\mathsf{X}|\mathsf{Y}=+1}$ and $\mathbf{P}_{\mathsf{X}|\mathsf{Y}=-1}$ are shown. The maximal within-class distance is d_w and the minimal between-class distance is d_b. Hence there exists a positive constant $\hat{d}$ with $d_\mathrm{w} < \hat{d} < d_\mathrm{b}$. Given such a perfect distance measure the learning task becomes trivial: Provided we have access to just one example of each class, $z^+ = (x^+, +1)$ and $z^- = (x^-, -1)$ we can classify any test example z drawn from $\mathbf{P}_{\mathsf{XY}}$ correctly simply by determining the nearest neighbour of x in $\{x^+, x^-\}$ and labelling x accordingly.

Of course, a distance measure perfect for a given learning task is rarely possible let alone known and available. Note that such a distance measure may only exist if the two class-conditional probability measures have no overlap. However, the general idea of considering the labels of the nearest neighbours of a test input in a given metric is a powerful idea and leads to the class of nearest neighbour classifiers.

2.2.1. Nearest Neighbour Classifiers

The basic idea of the K-nearest-neighbour (KNN) classifier is to determine the K training examples closest in a given metric to the test input and to assign to the test input the label of the majority of those examples. The KNN classifier was first proposed by Fix and Hodges (1951a) (see also Fix and Hodges (1951b)). It is thus a classic among the classification techniques and is still used as a benchmark today (LeCun 1998). In order to give a clean definition of the KNN classifier we need a definition of neighbourhood.

Definition 28 (Neighbourhood function). Given an input sample $\boldsymbol{x}$, an index $i \in \{1, \ldots, m\}$ and a test input $x \in \mathcal{X}$ we define the neighbourhood function $\Pi_d : \{1, \ldots, m\} \times \mathcal{X} \times \mathcal{X}^m \to \{1, \ldots, m\}$ with respect to a metric d on $\mathcal{X}$ as

$$\Pi_d(i; x, \boldsymbol{x}) \stackrel{\text{def}}{=} |\{x_j \in \boldsymbol{x} \,|\, d(x, x_j) < d(x, x_i)\}| + |\{x_j \in \boldsymbol{x} \,|\, d(x, x_j) = d(x, x_i) \wedge j \leq i\}| .$$

The neighbourhood function is a permutation function on the training sample. Note that the second summand serves as a tie-breaker in case of equal distances. Given the neighbourhood function we can now define the KNN classifier.

Definition 29 (K-nearest-neighbour classifier). Given an input space $\mathcal{X}$, an output space $\mathcal{Y}$, and a metric d on $\mathcal{X}$ the K-nearest-neighbour classifier is given by

$$h_{\text{knn}}(x; \boldsymbol{z}) \stackrel{\text{def}}{=} \operatorname*{argmax}_{y \in \mathcal{Y}} |\{(x_i, y_i) \in \boldsymbol{z} \,|\, \Pi_d(i; x, \boldsymbol{x}) \leq K \wedge y_i = y\}| + t(y) \tag{2.7}$$

where $t : \mathcal{Y} \to \left[0, \frac{1}{2}\right]$ is a tie-breaker function for the case of equal distances between inputs.

Note that—strictly speaking—KNN is a learning algorithm in the sense of Definition 7 because it takes as input a training sample $\boldsymbol{z}$ and returns an hypothesis $h \in \mathcal{Y}^{\mathcal{X}}$. However, to simplify notation we will denote the KNN classifier by h_{knn} keeping in mind its dependency on the training sample.

For binary classification the problem of voting ties is easily solved by choosing K odd. It should be noted that given a metric d the only parameter to choose is K. The KNN classifier is based on a local estimate of the class probabilities $\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}$ in the sense that

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y) = \frac{\mathbf{P}_{\mathsf{XY}}(x, y)}{\mathbf{P}_{\mathsf{X}}(x)} \approx \frac{1}{K} |\{(x_i, y_i) \in \boldsymbol{z} \,|\, \Pi_d(i; x, \boldsymbol{x}) \leq K \wedge y_i = y\}| ,$$

and thus implements a local estimate of the Bayes classifier from Theorem 1. The KNN classifier belongs to the class of example-based classifiers because the process of classification itself requires the training sample to be available as can be seen from (2.7). This is in contrast to hypothesis-based learning algorithms in which the training sample is required for learning, i.e., for the selection of a hypothesis $h \in \mathcal{H}$, but is not required for the classification of new examples. Despite its simplicity, the KNN classifier has a number of remarkable properties.

1. For a fixed value of K the error of the KNN classifier is asymptotically bounded by $2 \cdot R[h^*]$, i.e., in the limit of $m \to \infty$ the KNN classifier´s error does not exceed twice the Bayes risk (see Theorem 1). If K is increased with increasing m such that $m^{-1}K \to 0$ even universal consistency can be shown, i.e., the KNN classifier is asymptotically Bayes risk efficient. These results hold for arbitrary data distributions $\mathbf{P}_{\mathsf{XY}}$ and for a wide selection of metrics (see Devroye et al. (1996) for these and more results). This type of consistency result cannot be given for purely hypothesis-based classifiers with a fixed hypothesis space.

2. From a naive point of view the computational complexity of KNN is $\mathcal{O}(mKn_{\mathrm{d}})$ where n_{d} is the cost of one distance evaluation. However, with K and n_{d} fixed one may achieve $\mathcal{O}\left(m^{\frac{1}{d}}\right)$ worst-case time and $\mathcal{O}(\log m)$ expected time. This reduction is achieved, e.g., by the application of multi-dimensional search trees (see Devroye et al. (1996)).

3. By deleting certain data points, the complexity of the final classifier can be reduced (Duda and Hart 1973). This deletion scheme makes the classifier sparse and the remaining points resemble the support vectors defined in Definition 25.

4. The so-called (K, L)-Nearest Neighbour classifier (Hellman 1970) requires that the majority class wins the election by at least $L > \frac{K}{2}$, thus enforcing a voting margin. If such a margin is not achieved no decision is made by the classifier. This discrete margin resembles the real-valued margin defined in Definition 21.

5. The KNN classifier is easily extended to the case of multi-class learning. The K nearest neighbours simply vote among more than two alternatives.

6. No feature space or similar representation is necessary for the application of KNN. It suffices to have a metric available on the input space $\mathcal{X}$. The classifier is thus applicable to a wide range of problems independent of the particular representation.

The simple KNN classifier thus emerges as a versatile and powerful tool for classification, in particular for the case that little is known about the data at hand. A closely related classification scheme is the moving window estimator.

2.2.2. Moving Window Classifier

The moving window classifier works as follows: For a given test input $x \in \mathcal{X}$ consider all the inputs $x_i \in \boldsymbol{x}$ which lie within a given radius around x in the metric d. Among the labels of those inputs make a voting and decide for the majority class. In the case of a voting tie, some random or non-random tie-breaking procedure is invoked. More formally, the moving window classifier is defined as follows

Definition 30 (Moving Window Classifier). Given an input space $\mathcal{X}$, an output space $\mathcal{Y}$, and a metric d on $\mathcal{X}$ the moving window classifier with radius D is given by

$$h_{\mathrm{mw}}(x; \boldsymbol{z}) \stackrel{\mathrm{def}}{=} \operatorname*{argmax}_{y \in \mathcal{Y}} |\{(x_i, y_i) \in \boldsymbol{z} \mid d(x, x_i) \leq D \wedge y_i = y\}| + t(y) \tag{2.8}$$

where $t : \mathcal{Y} \to \left[0, \frac{1}{2}\right]$ is a tie-breaker function.

The moving window classifier shares many of the properties of the K-nearest neighbour classifier, among them universal strong consistency under conditions equivalent to those for KNN (Devroye et al. 1996). Effectively, the moving window classifier and the KNN classifier just differ in the way they determine which examples are used for voting.

The performance of the moving window estimator is strongly dependent on the parameter D, i.e., on the radius around the test input in which training data are considered for voting. If D is too small, then too few training data fall into the ball of radius D around the test input and the estimate of the class probability $\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y)$ is poor due to the lack of samples. The region of interest for classification may not even be fully covered. If D is too large then the estimate of $\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}$ lacks spatial resolution and—for $D \to \infty$—simply approaches the relative frequencies of classes in the sample.

In a sense, compared to the moving window classifier KNN adjusts the effective voting radius D in a data-depending fashion. Let $\Pi_d^{-1} : \{1,\ldots,m\} \times \mathcal{X} \times \mathcal{X}^m \to \{1,\ldots,m\}$ be the inverse of the neighbourhood function Π_d in its first argument. Then the data-dependent voting radius of KNN would be given by

$$D_{\mathrm{Knn}}(x, \boldsymbol{z}) = d\left(x, x_{\Pi_d^{-1}(i;x,\boldsymbol{x})}\right) ,$$

effectively choosing a small value of D in areas of high density and a high value of D in areas of small density. The moving window classifier is just a special case of the more general class of classifiers based on a kernel density estimation of the class-conditional probability densities.

2.2.3. Kernel Density Estimation Classifiers

At the beginning of Chapter 2 it was argued that knowledge of the probabilities $\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}$ would enable us to find the Bayes classifier h^* and thus to perform as well as possible given $\mathbf{P}_{\mathsf{XY}}$. Assuming the existence of a density $\mathbf{f}_{\mathsf{X}|\mathsf{Y}=y}$ in input space $\mathcal{X} \subseteq \ell_2^n$ by Bayes´ theorem we have

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y) = \frac{\mathbf{f}_{\mathsf{X}|\mathsf{Y}=y}(\mathbf{x})\,\mathbf{P}_{\mathsf{Y}}(y)}{\mathbf{f}_{\mathsf{X}}(\mathbf{x})} .$$

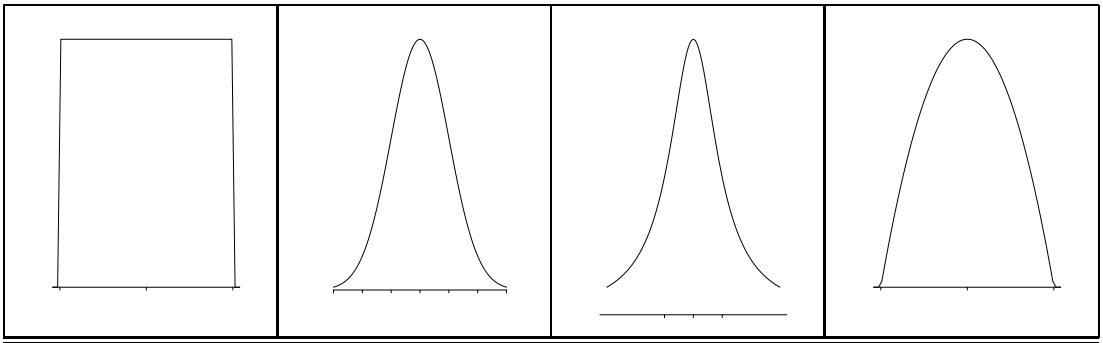
Thus given suitable estimates for $\mathbf{f}_{\mathsf{X}|\mathsf{Y}=y}$ and $\mathbf{P}_{\mathsf{Y}}$ we can find an approximation to the Bayes classifier $h^*(x) = \mathrm{argmax}_{y\in\mathcal{Y}} \mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y)$. Clearly, we can estimate $\mathbf{P}_{\mathsf{Y}}(y)$ by the number $m_y \stackrel{\mathrm{def}}{=} |\mathbf{i}_y|$ of examples of class y in the training sample over the total number of examples,

$$\hat{\mathbf{P}}_{\mathsf{Y}}(y) = \frac{m_y}{m} .$$

The density $\mathbf{f}_{\mathsf{X}|\mathsf{Y}=y}$ can be estimated by the Parzen window or kernel density estimator given by the following two definitions (see Silverman (1986) for various heuristics and Devroye and Györfi (1985) for an in-depth theoretical treatment of non-parametric density estimation):

Definition 31 (Radial density kernel). A function $k : \mathbb{R}^+ \to \mathbb{R}$ is called an *radial density kernel* if it satisfies

1. $k(r) \geq 0$ for all $r \geq 0$ (positive)
2. $\int_0^\infty k(r)\, dr < \infty$ (normalisable).



(a) Uniform (b) Gaussian (c) Cauchy (d) Epanechnikov

Figure 2.5.: Various radially symmetric density kernels $k(x)$ in $\mathbb{R}$ suitable for kernel density estimation. (a) Uniform or naive kernel $k_{\text{uniform}}(r) \sim \mathbf{I}_{0\leq r\leq 1}$, (b) Gaussian kernel $k_{\text{Gauss}}(r) \sim \exp\left(-r^2\right)$, (c) Cauchy kernel $k_{\text{Cauchy}}(r) \sim 1/\left(1+r^2\right)$, (d) Epanechnikov kernel $k_{\text{Epanechnikov}}(r) \sim \left(1-r^2\right)\mathbf{I}_{0\leq r\leq 1}$.

Various radial density kernels are shown in Figure 2.5. Given the definition of a radial density kernel we can proceed and define the kernel density estimator as follows:

Definition 32 (Kernel density estimator). Given the input space $\mathcal{X} = \ell_2^n$ with elements $\vec{x} \in \mathcal{X}$[2], an input sample $\boldsymbol{x}$ and a density kernel k satisfying

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} k\left(\|\vec{x}\|_2\right) d\vec{x} = 1\,,$$

the kernel density estimator with bandwidth D is defined as

$$\hat{\mathsf{f}}_{\mathsf{X}}(\vec{x}) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} \frac{1}{D} k\left(\frac{\|\vec{x}-\vec{x}_i\|}{D}\right).$$

Kernel density estimators are very flexible tools for density estimation. However, they are also very sensitive to the bandwidth parameter D, which determines the spatial resolution of the density estimate. Using the kernel density estimator defined in Definition 32 for estimating the class conditional densities $\hat{\mathsf{f}}_{\mathsf{X}|\mathsf{Y}=y}$ gives

$$\hat{\mathsf{f}}_{\mathsf{X}|\mathsf{Y}=y}(\vec{x}) = \frac{1}{m_y}\sum_{i\in \mathbf{i}_y(\boldsymbol{z})} \frac{1}{D} k\left(\frac{\|\vec{x}-\vec{x}_i\|}{D}\right).$$

[2] If the input space $\mathcal{X}$ is a vector space ℓ_2^n we denote its elements by $\vec{x} \in \mathcal{X}$ to indicate that $\vec{x}$ is a vector. As will be seen in Section 2.3 the kernel induces another feature representation $\mathbf{x} \stackrel{\text{def}}{=} \boldsymbol{\phi}(\vec{x})$ for which the bold symbol will be reserved.

Now, we can complete our estimation of $\hat{h}^*$ for the Bayes classifier h^* by

$$\begin{aligned}
\hat{h}^*(\vec{x}) &= \operatorname*{argmax}_{y\in\mathcal{Y}} \hat{\mathbf{P}}_{\mathbf{Y}|\mathbf{X}=\vec{x}}(y) \\
&= \operatorname*{argmax}_{y\in\mathcal{Y}} \hat{\mathbf{P}}_{\mathbf{Y}}(y)\,\hat{\mathbf{f}}_{\mathbf{X}|\mathbf{Y}=y}(\vec{x}) \\
&= \operatorname*{argmax}_{y\in\mathcal{Y}} \frac{m_y}{m}\frac{1}{m_y}\sum_{i\in\mathbf{i}_y(\boldsymbol{z})}\frac{1}{D}k\left(\frac{\|\vec{x}-\vec{x}_i\|}{D}\right) \\
&= \operatorname*{argmax}_{y\in\mathcal{Y}} \sum_{i\in\mathbf{i}_y(\boldsymbol{z})} k\left(\frac{\|\vec{x}-\vec{x}_i\|}{D}\right) \\
&= \operatorname{sign}\left(\sum_{i\in\mathbf{i}_{+1}(\boldsymbol{z})} k\left(\frac{\|\vec{x}-\vec{x}_i\|}{D}\right) - \sum_{i\in\mathbf{i}_{+1}(\boldsymbol{z})} k\left(\frac{\|\vec{x}-\vec{x}_i\|}{D}\right)\right) \\
&= \operatorname{sign}\left(\sum_{i=1}^{m} y_i k\left(\frac{\|\vec{x}-\vec{x}_i\|}{D}\right)\right)
\end{aligned}$$

This result naturally leads to the definition of a classifier based on kernel density estimation:

Definition 33 (Kernel density estimation classifier). Given the input space $\mathcal{X} = \ell_2^n$ the output space $\mathcal{Y} = \{+1, -1\}$, a training sample $\boldsymbol{z}$, a radial density kernel k, and a kernel bandwidth D the *kernel density estimation classifier* is given by

$$h_{\text{kde}}(\mathbf{x}) \stackrel{\text{def}}{=} \operatorname{sign}\left(\sum_{i=1}^{m} y_i k\left(\frac{\|\vec{x}-\vec{x}_i\|}{D}\right)\right).$$

Clearly, the moving window classifier is a special case of the kernel density estimation classifier with the uniform kernel k_{unif}. The use of kernels different from k_{unif} such as those shown in Figure 2.5 results in smoother decision regions and usually prevents voting ties as can be found by the uniform weighting of examples using k_{unif}. In addition, if the kernel used satisfies $k(r) > 0$ for all $r > 0$ the problem of uncovered regions in input space vanishes. In addition it should be noted that kernel density estimation classifiers also enjoy the property of universal consistency for regular kernels (Devroye et al. 1996).

2.3. Kernel-Based Classification

In the previous two subsections we introduced two different types of classification schemes: hypothesis-based and example-based. In hypothesis-based classification, once a hypothesis space $\mathcal{H}$ is chosen, learning can be considered as an optimisation or constraint satisfaction problem as in the case of perceptron learning. In contrast, the example-based approach can hardly be considered as learning, because once the metric and some algorithm-specific parameters are chosen, the classifier is given as a function of training sample $\boldsymbol{z}$ and test input $x \in \mathcal{X}$. Both approaches have their merits.

1. Hypothesis-based classification may be computationally complex during learning, i.e., in the optimisation phase. Often the optimisation problems involved have local optima, as for example in multi-layer neural networks or in perceptron learning for problems that are not linearly separable. In this case it may not even be feasible to find the global optimum in a reasonable amount of time. Often approximation schemes must be invoked to make the optimisation possible. In contrast, example-based classification as considered here does not involve any optimisation.

2. Hypothesis-based classification is often faster at classifying new inputs as compared to example-based classification. At least this is often the case in the regime $m \gg n$ where there are more training examples m than feature dimensions n. This is due to the fact that hypothesis-based classification roughly takes $\mathcal{O}(n)$ for classification, while example-based classification takes $\mathcal{O}(m)$. Of course, more elaborate estimates of the computational complexities are possible if details about the different learning scenarios are known.

3. Hypothesis-based classification requires the selection of an appropriate hypothesis space $\mathcal{H}$. This option makes it possible to facilitate learning by applying apriori domain-specific knowledge to the selection or construction of the hypothesis space. If such knowledge is not available, however, it may be difficult to find an appropriate hypothesis space. In contrast, example-based classification does not require the specification of an hypothesis space. Instead, algorithm-specific parameters have to be tuned and, in particular, a suitable metric has to be found.

4. Based on the previous point, hypothesis- and example-based approaches often differ in the interpretability of the resulting classifiers. The hypotheses chosen by hypothesis-based classification schemes often provide interesting information about the classification problem at hand in terms of the features used. Linear classifiers based on a weight vector $\mathbf{w} \in \mathcal{K}$ provide information through the different weights w_i associated with different features ϕ_i. Similarly, decision trees constitute a hierarchy of feature decisions that may provide information about the importance of features w.r.t. the given classification problem. Note, however, that this need not be the case as can be seen from the weights of multi-layer perceptrons that may be difficult to interpret. However, example-based approaches as considered here do not provide any such information.

It turns out that kernel-based classification provides us with a bridge between hypothesis-based and example-based classification.

2.3.1. Kernel Classifiers

Let us consider the hypothesis space $\mathcal{H}_{\mathcal{K}}$ of linear classifiers in a feature space $\mathcal{K}$ as given in Definition 19, containing hypotheses of the type

$$h_{\mathbf{w}}(x) \stackrel{\text{def}}{=} \operatorname{sign}\left(\langle \boldsymbol{\phi}(x), \mathbf{w} \rangle\right),$$

where the mappings $\phi_i : \mathcal{X} \to \mathbb{R}$ provide the feature space representation of the inputs $x \in \mathcal{X}$ such that inputs x are mapped to the feature space $\mathcal{K}$. Also let $\mathbf{w} \in \mathcal{K}$. We assume that the weight vector $\mathbf{w}$ can be expanded as a linear combination of training inputs $\boldsymbol{\phi}(x_i)$ as in the dual representation of $\mathbf{w}$ in the maximum margin classifier duals in Theorem 4,

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \boldsymbol{\phi}(x_i) .$$

Then the linear classifiers in $\mathcal{K}$ are of the form

$$h_{\mathbf{w}}(x) = \operatorname{sign}\left(\sum_{i=1}^{m} \alpha_i \langle \boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x) \rangle\right) .$$

The essence of kernel-based classification is to merge the two steps of (i) mapping the inputs $x \in \mathcal{X}$ to $\mathcal{K}$ by the application of $\boldsymbol{\phi}$ and (ii) of calculating the inner product $\langle \boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x) \rangle$ in $\mathcal{K}$ by the application of a *kernel* according to the following definition:

Definition 34 (Kernel). Given an input space $\mathcal{X}$, a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$ a *kernel* is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that for all $x_1, x_2 \in \mathcal{X}$

$$k(x_1, x_2) \stackrel{\text{def}}{=} \langle \boldsymbol{\phi}(x_1), \boldsymbol{\phi}(x_2) \rangle . \tag{2.9}$$

Assuming that we can compute the kernel for a given feature space $\mathcal{K}$ directly we can now define the *kernel classifier*. This definition will be central to the remainder of this work because most of the theorems to follow will be applicable to kernel classifiers, the algorithms to follow will be related to kernel classifiers, and the applications will be carried out using kernel classifiers.

Definition 35 (Kernel Classifier). Given an input space $\mathcal{X}$, an output space $\mathcal{Y} = \{+1, -1\}$, a training sample $\boldsymbol{z} \in (\mathcal{X} \times \mathcal{Y})^m$, a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and m coefficients $\alpha_i \in \mathbb{R}$ a kernel classifier is defined by

$$h_{(\boldsymbol{\alpha}, \boldsymbol{z})}(x) \stackrel{\text{def}}{=} \operatorname{sign}\left(\sum_{i=1}^{m} \alpha_i k(x_i, x)\right)$$

Given a kernel k a kernel classifier is thus characterised by its coefficient vector $\boldsymbol{\alpha}$ of m coefficients α_i, one for each data point. A learning algorithm in the sense of Definition 7 for kernel classifiers is a procedure for determining the vector $\boldsymbol{\alpha}$ of coefficients. Several aspects of kernel classifiers are remarkable:

1. The kernel classifier is inherently linear in the feature space $\mathcal{K}$ and can thus be analysed using well-known concepts and methods from classical linear classification.

2. The evaluation of a kernel classifier is at most $\mathcal{O}(m)$, and possibly less due to vanishing coefficients α_i. The computational complexity of evaluating the classifier is independent of the dimensionality n of the feature space if the computational complexity of evaluating the kernel is. As we will later see this is frequently the case.

3. The kernel classifier shares aspects of both hypothesis-based classification (Section 2.1) and example-based classification (Section 2.2). It is hypothesis-based because it is characterised by an associated weight vector $\mathbf{w} \in \mathcal{K}$. It is example-based because evaluation of the classifiers requires at least a subset of the training sample.

In order to be able to design kernels for given learning tasks it will be necessary to determine, what properties a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ must satisfy such that an associated feature space in the sense of Definition 34 exists. To this end we will characterise kernels mathematically.

2.3.2. Properties of Kernels

From the relation $k(x, \tilde{x}) = \langle \boldsymbol{\phi}(x), \boldsymbol{\phi}(\tilde{x}) \rangle$ it is clear that a kernel function must be symmetric $k(x, \tilde{x}) = k(\tilde{x}, x)$ and satisfy the Cauchy-Schwarz inequality $k^2(x, \tilde{x}) \leq k(x, x) k(\tilde{x}, \tilde{x})$ simply because the corresponding inner product $\langle \boldsymbol{\phi}(x), \boldsymbol{\phi}(\tilde{x}) \rangle$ has these properties by definition. However, these are only necessary conditions for k being a kernel. The following theorem gives us a necessary and sufficient criterion for kernels on input spaces $\mathcal{X}$ of finite cardinality $|\mathcal{X}| = m$.

Theorem 7 (Positive definite kernel matrix). *Given an input space $\mathcal{X} = \{x_1, \ldots, x_m\}$ and a symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, then k is a kernel function if and only if the $m \times m$-matrix $\mathbf{K}$ with elements*

$$\mathbf{K}_{ij} = k(x_i, x_j)$$

is positive semi-definite (has non-negative eigenvalues).

Proof. If k is a kernel we have

$$\mathbf{K}_{ij} = \langle \boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x_j) \rangle .$$

Hence we have for all $\boldsymbol{\alpha} \in \mathbb{R}^n$ that

$$\boldsymbol{\alpha}' \mathbf{K} \boldsymbol{\alpha} = \left\langle \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}(x_i), \sum_{j=1}^{n} \alpha_j \boldsymbol{\phi}(x_j) \right\rangle = \left\| \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}(x_i) \right\|^2 \geq 0,$$

i.e., $\mathbf{K}$ is positive semidefinite.

If $\mathbf{K}$ is positive semidefinite then there exists an eigenvalue decomposition where $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$ contains the eigenvalues $\lambda_i \geq 0$ on it diagonal and the columns of $\mathbf{V}$ are the corresponding eigenvectors $\mathbf{v}_i$. Hence we have can write

$$\mathbf{K}_{ij} = \left\langle \sqrt{\boldsymbol{\Lambda}} \mathbf{v}_i, \sqrt{\boldsymbol{\Lambda}} \mathbf{v}_j \right\rangle := \langle \boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x_j) \rangle ,$$

where the feature mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{K}$ is given by

$$\boldsymbol{\phi} : x_i \to \sqrt{\boldsymbol{\Lambda}} \mathbf{v}_i ,$$

showing that k is indeed a kernel according to Definition 34 □

Theorem 7 was characterised by an input space $\mathcal{X}$ of finite cardinality, and a feature space $\mathcal{K}$ of finite dimensionality. Both limitations are overcome by Mercer´s theorem from functional analysis. In discussing this theorem we assume some basic knowledge about Hilbert spaces (for a detailed exposition of Hilbert spaces see Debnath and Mikusinski (1998)). In particular, the fact that we are dealing with function spaces on the one hand and spaces of countable dimensionality on the other hand motivates the introduction of two particular Hilbert spaces:

Definition 36 (Hilbert space of countable square-summable sequences). We denote by $\ell_2(\boldsymbol{\mu})$ the Hilbert space of countable square-summable sequences of $\mathbf{x} = (x_1, \ldots, x_i, \ldots)$ with $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_i, \ldots)$ a countable sequence of positive real numbers, i.e.,.

$$\ell_2(\boldsymbol{\mu}) \stackrel{\text{def}}{=} \left\{ \mathbf{x} = (x_i)_{i \in \mathbb{N}} : \|\mathbf{x}\|^2_{\ell_2(\boldsymbol{\mu})} = \sum_{i=1}^{\infty} \mu_i x_i^2 < \infty \right\}.$$

Then we can define the inner product

$$\langle \mathbf{x}, \tilde{\mathbf{x}} \rangle_{\ell_2(\boldsymbol{\mu})} \stackrel{\text{def}}{=} \sum_{i=1}^{\infty} \mu_i x_i \tilde{x}_i .$$

Besides these sequences we need the Hilbert space of square integrable continuous functions.

Definition 37 (Hilbert space of square-integrable continuous functions). We denote by $L_2(X)$ the Hilbert space of square-integrable continuous functions $f : X \to \mathbb{R}$ with $X \subset \ell_2^n$, i.e.,

$$L_2(X) \stackrel{\text{def}}{=} \left\{ f : X \to \mathbb{R} : \|f\|^2_{L_2} = \int_X f^2(\mathbf{x}) \, \mathrm{d}\mathbf{x} < \infty \right\}.$$

Then we can define the inner product

$$\langle f, g \rangle_{L_2} \stackrel{\text{def}}{=} \int_X f(\mathbf{x}) \cdot g(\mathbf{x}) \, \mathrm{d}\mathbf{x} .$$

In analogy with the finite case given in Theorem 7 Mercer´s theorem (Mercer 1909; König 1986) characterises kernel functions in terms of the eigen spectrum of their associated integral operator. We will simply quote the theorem here as it can be found in Cristianini and Shawe-Taylor (2000) and discuss some of its consequences for working with kernels.

Theorem 8 (Mercer´s Theorem). *Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^n$. Suppose k is a continuous symmetric function such that the integral operator $T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$,*

$$(T_k f)(\cdot) \stackrel{\text{def}}{=} \int_{\mathcal{X}} k(\cdot, \mathbf{x}) f(\mathbf{x}) \, \mathrm{d}\mathbf{x}$$

is positive, that is

$$\int_{\mathcal{X}} \int_{\mathcal{X}} k(\mathbf{x}, \tilde{\mathbf{x}}) f(\mathbf{x}) f(\tilde{\mathbf{x}}) \, \mathrm{d}\mathbf{x} \mathrm{d}\tilde{\mathbf{x}} \geq 0 \tag{2.10}$$

for all $f \in L_2(\mathcal{X})$. *Then* k *is a kernel and we can expand* $k(\mathbf{x}, \tilde{\mathbf{x}})$ *in a uniformly convergent series (on* $\mathcal{X} \times \mathcal{X}$*) in terms of* T_k*'s eigen functions* $\phi_j \in L_2(X)$, *normalised in such a way that* $\|\phi_j\|_{L_2} = 1$, *and positive associated eigenvalues* $\lambda_j > 0$ *as*

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \sum \lambda_j \phi_j(\mathbf{x}) \phi_j(\tilde{\mathbf{x}}) .$$

Let us first consider the relation to the case of an input space $\mathcal{X}$ of finite cardinality $|\mathcal{X}|$ given in Theorem 7. The positivity condition (2.10) corresponds to the positive semi-definiteness of the kernel matrix in Theorem 7. In fact, Theorem 7 is a special case of Mercer's theorem for a finite subset $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathcal{X}$ of points $\mathbf{x}_i$. This can be seen by choosing

$$f(\mathbf{x}) = \sum_{i=1}^{m} v_i \delta(\mathbf{x}_i) , \tag{2.11}$$

where $\delta : \mathbb{R}^n \to [0, \infty)$ is Dirac's delta function and can be expressed as the limit of functions in L_2. Then condition (2.10) becomes

$$\begin{aligned} 0 &\leq \int_{\mathcal{X}} \int_{\mathcal{X}} k(\mathbf{x}, \tilde{\mathbf{x}}) f(\mathbf{x}) f(\tilde{\mathbf{x}}) \, \mathrm{d}\mathbf{x}\mathrm{d}\tilde{\mathbf{x}} \\ &= \sum_{i=1}^{m} \sum_{j=1}^{m} v_i v_j \int_{\mathcal{X}} \int_{\mathcal{X}} k(\mathbf{x}, \tilde{\mathbf{x}}) \delta(\mathbf{x}_i) \delta(\tilde{\mathbf{x}}_j) \, \mathrm{d}\mathbf{x}\mathrm{d}\tilde{\mathbf{x}} \\ &= \sum_{i=1}^{m} \sum_{j=1}^{m} v_i v_j k(\mathbf{x}_i, \tilde{\mathbf{x}}_j) \\ &= \mathbf{v}' \mathbf{K} \mathbf{v} , \end{aligned}$$

which is just the condition of positive semi-definiteness for the kernel matrix $\mathbf{K}$. Similarly, if the positivity condition (2.10) does not hold for some functions $f \in L_2$ we can always conceive of a mesh $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathcal{X}$ of points such that the resulting kernel matrix $\mathbf{K}$ has negative eigenvalues. As a consequence, the condition for k being a kernel function in the sense of Mercer's theorem can be expressed alternatively as follows (Saitoh 1988):

Corollary 1 (Matrix formulation of Mercer's condition). *The kernel operator* T_k *from Mercer's theorem 8 is positive if and only if for any finite subset* $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathcal{X}$ *the corresponding kernel matrix* $\mathbf{K}$ *is positive semi-definite.*

In the remainder of this thesis the expression *Mercer kernel* or simply *kernel* will refer to a kernel function satisfying the condition of Mercer's theorem. Let us now consider the feature mapping implied by Mercer's theorem, Theorem 8.

Definition 38 (Mercer features). Given a kernel function k satisfying Mercer's condition, then the mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{K}$ into the feature space $\mathcal{K} = \ell_2(\boldsymbol{\lambda})$, i.e.,

$$\mathbf{x} \mapsto \boldsymbol{\phi}(x) = (\phi_1(x), \ldots, \phi_j(x), \ldots) ,$$

is called the *Mercer feature mapping* and the associated features $\phi_i(x)$ are called *Mercer features.*

Clearly, the Mercer features satisfy the kernel relation

$$\begin{aligned}\left\langle \boldsymbol{\phi}\left(x\right),\boldsymbol{\phi}\left(\tilde{x}\right)\right\rangle_{\ell_2(\boldsymbol{\lambda})} &= \sum_{i=1}^{\infty}\lambda_i\boldsymbol{\phi}\left(x\right)\boldsymbol{\phi}\left(\tilde{x}\right)\\ &= k\left(x,\tilde{x}\right),\end{aligned}$$

with λ_i the eigenvalues of the integral operator T_k. The features are orthonormal in $L_2\left(\mathcal{X}\right)$. Note, however, that a rescaling of the features is possible, i.e., given a sequence $\mathbf{b}=\left(b_1,\ldots,b_i,\ldots\right)$ of real numbers we can choose the feature mapping $\boldsymbol{\phi}:\mathcal{X}\rightarrow\mathcal{K}$ into the feature space $\mathcal{K}=\ell_2\left(\boldsymbol{\mu}\right)$ with $\mu_i=\frac{\lambda_i}{b_i^2}$,

$$x\mapsto\boldsymbol{\phi}\left(x\right)=\left(b_1\phi_1\left(x\right),\ldots,b_j\phi_j\left(x\right),\ldots\right),$$

which also satisfies the kernel condition (2.9).

The gist of kernel methods building on Mercer´s theorem lies in the complementarity of kernel function k and feature space $\mathcal{K}$. Given a set of real-valued features we can always calculate the kernel or inner product matrix $\mathbf{K}$ and be sure that it is non-negative by construction. However, building on Mercer´s theorem we may not even need to construct a feature space explicitly, We can instead conceive of an appropriate kernel function k and verify by Mercer´s theorem if that particular kernel corresponds to an inner product in some feature space $\mathcal{K}$. We can thus implicitly work in that feature space $\mathcal{K}$ by carrying out inner products in $\mathcal{K}$ without explicitly performing the mapping $\boldsymbol{\phi}:\mathcal{X}\rightarrow\mathcal{K}$ or even without knowing the exact nature of $\mathcal{K}$.

In practice, we can operate geometrically in the feature space $\mathcal{K}$ as long as we restrict ourselves to the span of data points $\boldsymbol{\phi}\left(x\right)$ whose inner products $\left\langle\boldsymbol{\phi}\left(x_i\right),\boldsymbol{\phi}\left(x_j\right)\right\rangle$ are known and given by the kernel matrix $\mathbf{K}$ with elements $\mathbf{K}_{ij}\overset{\text{def}}{=}k\left(x_i,x_j\right)=\left\langle\boldsymbol{\phi}\left(x_i\right),\boldsymbol{\phi}\left(x_j\right)\right\rangle$. Let us briefly consider some basic geometrical operations in $\mathcal{K}$. Given a vector $\mathbf{w}\in\mathcal{K}$ that can be expressed as a linear combination of vectors $\boldsymbol{\phi}\left(x_i\right)$,

$$\mathbf{w}=\sum_{i=1}^{m}\alpha_i\boldsymbol{\phi}\left(x_i\right),$$

with expansion coefficients $\alpha_i\in\mathbb{R}$, its squared norm $\left\|\mathbf{w}\right\|^2$ can be calculated as

$$\begin{aligned}\left\|\mathbf{w}\right\|^2 &= \left\langle\mathbf{w},\mathbf{w}\right\rangle\\ &= \left\langle\sum_{i=1}^{m}\alpha_i\boldsymbol{\phi}\left(x_i\right),\sum_{i=1}^{m}\alpha_j\boldsymbol{\phi}\left(x_j\right)\right\rangle\\ &= \sum_{i=1}^{m}\sum_{i=1}^{m}\alpha_i\alpha_j\left\langle\boldsymbol{\phi}\left(x_i\right),\boldsymbol{\phi}\left(x_j\right)\right\rangle\\ &= \sum_{i=1}^{m}\sum_{i=1}^{m}\alpha_i\alpha_j k_{ij}\\ &= \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}.\end{aligned}$$

Similarly, given another vector $\tilde{\mathbf{w}} \in \mathcal{K}$ again expressed using expansion coefficients $\tilde{\alpha}_i \in \mathbb{R}$,

$$\tilde{\mathbf{w}} = \sum_{i=1}^{m} \tilde{\alpha}_i \boldsymbol{\phi}(x_i) \ ,$$

we can calculate the inner product $\langle \mathbf{w}, \tilde{\mathbf{w}} \rangle$ in $\mathcal{K}$ simply by

$$\begin{aligned} \langle \mathbf{w}, \tilde{\mathbf{w}} \rangle &= \left\langle \sum_{i=1}^{m} \alpha_i \boldsymbol{\phi}(x_i) \, , \sum_{i=1}^{m} \tilde{\alpha}_j \boldsymbol{\phi}(x_j) \right\rangle \\ &= \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \tilde{\alpha}_j \langle \boldsymbol{\phi}(x_i) \, , \boldsymbol{\phi}(x_j) \rangle \\ &= \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \tilde{\alpha}_j k_{ij} \\ &= \boldsymbol{\alpha}' \mathbf{K} \tilde{\boldsymbol{\alpha}} \, . \end{aligned}$$

Finally, the squared Euclidean distance $\|\mathbf{w} - \tilde{\mathbf{w}}\|^2$ in $\mathcal{K}$ is calculated using

$$\begin{aligned} \|\mathbf{w} - \tilde{\mathbf{w}}\|^2 &= \|\mathbf{w}\|^2 - 2 \langle \mathbf{w}, \tilde{\mathbf{w}} \rangle + \|\tilde{\mathbf{w}}\|^2 \\ &= \boldsymbol{\alpha}' \mathbf{K} \boldsymbol{\alpha} - 2 \boldsymbol{\alpha}' \mathbf{K} \tilde{\boldsymbol{\alpha}} + \tilde{\boldsymbol{\alpha}}' \mathbf{K} \tilde{\boldsymbol{\alpha}} \\ &= (\boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}})' \mathbf{K} (\boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}}) \end{aligned}$$

The linear expansion of the weight vector $\mathbf{w} \in \mathcal{K}$ is the condition for doing geometry in feature space. We will see that this expansion is a natural consequence for many algorithms that are based on linear functions in $\mathcal{K}$. In particular, it follows from the dual representation of the quadratic programs in support vector learning or from the algorithmic procedure in (dual) perceptron learning. More generally, this expansion follows for the solution of certain optimisation problems from the so-called *representer theorem* by Grace Wahba that was published as early as 1970 in the context of spline models (Kimeldorf and Wahba 1970). It has recently been generalised by Schölkopf et al. (2001) to more general cost functions.

Theorem 9 (Representer Theorem). *Suppose we are given a fixed mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{K} \subseteq \ell_2^n$, a training sample $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{Z}^m$, a cost function $c : \mathcal{X}^m \times \mathcal{Y}^m \times \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ strictly monotonically decreasing in its third argument and the class of linear functions in $\mathcal{K}$ as given in Definition 19. Then any $\mathbf{w}_{\boldsymbol{z}} \in \mathcal{W}$ defined by*

$$\mathbf{w}_{\boldsymbol{z}} = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \ c(\boldsymbol{x}, \boldsymbol{y}, (\langle \mathbf{x}_1, \mathbf{w} \rangle, \ldots, \langle \mathbf{x}_m, \mathbf{w} \rangle))$$

admits a representation of the form

$$\exists \boldsymbol{\alpha} \in \mathbb{R}^m : \qquad \mathbf{w}_{\boldsymbol{z}} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i \, .$$

The proof of the theorem is basically a simple projection argument and can be found in Appendix A.

We could only scratch the surface of the theory of kernel methods in this subsection large parts of which follow the line of reasoning given in Cristianini and Shawe-Taylor (2000). For a competent overview see also Schölkopf et al. (1999). The theory of kernels goes back as far as 1909 to Mercer´s seminal work (Mercer 1909). The application of kernel methods under the name of "potential functions" for learning was proposed and carried out by Aizerman et al. (1964). Later the application of kernel methods became popular due to the success of the support vector machine (Cortes and Vapnik 1995; Vapnik 1995; Vapnik 1998). There exists a rather deep theory in approximation theory and statistics on reproducing kernel Hilbert spaces which traces back mainly to Kimeldorf and Wahba (1970) and Wahba (1990). Finally, there are interesting relations to covariance functions studied in the context of Gaussian processes (see, e.g. Opper and Winther (2000a)) and krieging (see Matheron (1963)).

2.3.3. Constructing Kernels

Part of the power and conceptual elegance of kernel methods lies in the way kernels can be constructed from kernels. Once we have verified the Mercer condition, for example from Corollary 1, we can use certain closure properties of Mercer kernels to build kernels from kernels. These closure properties are given by the following theorem which is essentially quoted from Cristianini and Shawe-Taylor (2000) where also the proof can be found.

Theorem 10 (Closure properties of kernels). *Let k_1 and k_2 be kernels over $\mathcal{X} \times \mathcal{X}$, $\mathcal{X} \subseteq \ell_2^n$, $\vec{x}_1, \vec{x}_2 \in \mathcal{X}$, $a \in \mathbb{R}^+$, $f : \mathcal{X} \to \mathbb{R}$, $\boldsymbol{\phi} : \mathcal{X} \to \ell_2^{\tilde{n}}$ with k_3 a kernel over $\ell_2^{\tilde{n}} \times \ell_2^{\tilde{n}}$, $\mathbf{B}$ a symmetric positive semi-definite $n \times n$ matrix, $p : \mathbb{R} \to \mathbb{R}$ a polynomial $p(x) = \sum_{i=0}^{\infty} a_i x^i$ with non-negative coefficients $a_i \in \mathbb{R}^+$. Then the following functions are kernels:*

1. $k(\vec{x}_1, \vec{x}_2) = k_1(\vec{x}_1, \vec{x}_2) + k_2(\vec{x}_1, \vec{x}_2)$
2. $k(\vec{x}_1, \vec{x}_2) = a\, k_1(\vec{x}_1, \vec{x}_2)$
3. $k(\vec{x}_1, \vec{x}_2) = k_1(\vec{x}_1, \vec{x}_2) \cdot k_2(\vec{x}_1, \vec{x}_2)$
4. $k(\vec{x}_1, \vec{x}_2) = f(\vec{x}_1) \cdot f(\vec{x}_2)$
5. $k(\vec{x}_1, \vec{x}_2) = k_3(\boldsymbol{\phi}(\vec{x}_1), \boldsymbol{\phi}(\vec{x}_2))$
6. $k(\vec{x}_1, \vec{x}_2) = \vec{x}_1' \mathbf{B} \vec{x}_2$
7. $k(\vec{x}_1, \vec{x}_2) = p(k_1(\vec{x}_1, \vec{x}_2))$
8. $k(\vec{x}_1, \vec{x}_2) = \exp(k_1(\vec{x}_1, \vec{x}_2))$
9. $k(\vec{x}_1, \vec{x}_2) = \exp\left(-\frac{\|\vec{x}_1 - \vec{x}_2\|^2}{2\sigma^2}\right)$

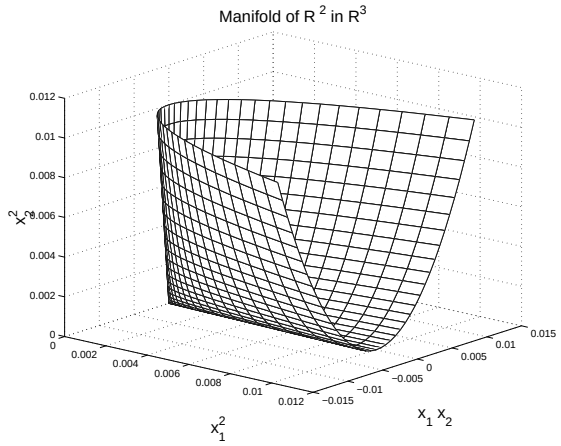


Figure 2.6.: Illustration of feature space $\mathcal{K}$ and feature mapping $\boldsymbol{\phi}$ for the homogenous polynomial kernel of degree $p = 2$.

In the light of this theorem let us consider some basic kernels in ℓ_2^n. Clearly, if we choose $\mathcal{X} \subseteq \ell_2^n$ and $\boldsymbol{\phi}(\vec{x}_1) = \vec{x}_1$ then the corresponding kernel k is simply the inner product $k(\vec{x}_1, \vec{x}_2) = \langle \vec{x}_1, \vec{x}_2 \rangle$ in $\mathcal{X}$, which is in accordance with part 6 of Theorem 10 with $\mathbf{B} = \mathbf{I}_n$. Possibly the simplest non-trivial kernel in $\mathbb{R}^n$ is the so-called homogenous polynomial kernel of degree p given by $k(\vec{x}_1, \vec{x}_2) = \langle \vec{x}_1, \vec{x}_2 \rangle^p$.

Example 1 (Polynomial kernel of degree 2). Consider the case $\mathcal{X} \subseteq \mathbb{R}^2$ and the polynomial kernel $k(\vec{x}_1, \vec{x}_2) = \langle \vec{x}_1, \vec{x}_2 \rangle^2$ of degree $p = 2$. A corresponding feature mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{K}$ with $\mathcal{K} \subset \mathbb{R}^3$ is given by

$$\boldsymbol{\phi}(\vec{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \end{pmatrix}.$$

We can verify the kernel condition $k(\vec{x}_1, \vec{x}_2) = \langle \boldsymbol{\phi}(\vec{x}_1), \boldsymbol{\phi}(\vec{x}_2) \rangle$ by evaluating the two sides of the equation component-wise and find equality. For the case $\mathcal{X} = [-1, 1] \times [-1, 1]$ and the homogenous polynomial kernel of degree $p = 2$ Figure 2.6 depicts the resulting manifold $\boldsymbol{\phi}(\mathcal{X})$ in $\mathbb{R}^3$. Note that the inherent dimensionality of the manifold is $\dim(\boldsymbol{\phi}(\mathcal{X})) = 2$ corresponding to the dimensionality $\dim(\mathcal{X}) = 2$ of $\mathcal{X}$. A (linear) hyperplane in $\mathcal{K}$ corresponds to a parabolic curve in $\mathcal{X}$ thus making it possible to do non-linear classification in $\mathcal{X}$ by doing linear classification in $\mathcal{K}$.

In general, the dimensionality $n_{\mathcal{K}}$ of the feature space $\mathcal{K}$ corresponding to the homogenous polynomial kernel $k(\vec{x}_1, \vec{x}_2) = \langle \vec{x}_1, \vec{x}_2 \rangle^p$ of degree p for $\mathcal{X} \subseteq \mathbb{R}^n$ is given by

$$n_{\mathcal{K}} = \binom{n + p - 1}{p}.$$

The distinct features ϕ_i in this case are the monomials of degree p. The inhomogeneous polynomial kernel $k(\vec{x}_1, \vec{x}_2) = (\langle \vec{x}_1, \vec{x}_2 \rangle + c)^p$ has a minimal corresponding feature space

of dimensionality

$$n_{\mathcal{K}} = \binom{n+p}{p}$$

spanned by all the monomials of up to and including p. The general appeal of these kernels comes from the fact that the kernel features correspond to interactions of up to p of the original n features. If the original features are, for example, binary indicators of word occurrences in a bag-of-words representation of documents (Joachims 1998) then a polynomial kernel of degree p would consider word co-occurrences of p words, i.e., bi-grams for $p = 2$ etc. Also in the field of handwritten-digit recognition, in particular on the well-known USPS and NIST datasets, polynomial kernels have proven to be very useful in conjunction with the support vector machine (Vapnik 1995).

Among the most popular kernels is the Gaussian radial-basis function kernel (see also part 9 of Theorem 10) given by

$$k_{\text{rbf}}(\vec{x}_1, \vec{x}_2) \stackrel{\text{def}}{=} \exp\left(-\frac{\|\vec{x}_1 - \vec{x}_2\|^2}{2\sigma^2}\right). \tag{2.12}$$

This kernel is often simply called *RBF kernel,* and is known to be practitioner´s best first guess if no further information about the problem at hand is available. The RBF kernel function can be approximated arbitrarily well by a sum of polynomials with positive coefficients and is thus the limit of a kernel function. Another consequence of the polynomial expansion is that the dimensionality of the associated feature space $\mathcal{K}$ can be thought of as infinite. Other remarkable properties of the RBF kernel include:

1. The RBF kernel is positive everywhere and—if normalised appropriately—constitutes a proper density. It thus acts as a local approximator and limits the influence any one single input has on the final classification.

2. RBF kernels are known to be universal approximators and have been successfully used in so-called RBF neural networks (Poggio and Girosi 1990).

3. The bandwidth parameter σ is continuous (in contrast to the degree p of polynomial kernels) and can thus be adapted by gradient-based optimisation methods such as gradient descent.

4. The RBF kernel is normalised in the sense that for any $\boldsymbol{\phi}(\vec{x}_1) \in \mathcal{K}$ we have $\|\boldsymbol{\phi}(\vec{x}_1)\|^2 = k_{\text{rbf}}(\vec{x}_1, \vec{x}_1) = 1$. Also it is positive, $k_{\text{rbf}}(\vec{x}_1, \vec{x}_1) > 0$ for all vectors $\vec{x}_1, \vec{x}_2 \in \mathcal{X}$. As a consequence the inputs in feature space lie on the positive 2^n-tant of the unit hyper-sphere. The normalisation can also be seen by considering

first the non-local kernel $k'(\vec{x}_1, \vec{x}_2) = \exp\left(\frac{\langle \vec{x}_1, \vec{x}_2 \rangle}{\sigma^2}\right)$ and normalising it by

$$\begin{aligned}
\frac{\langle \boldsymbol{\phi}(\vec{x}_1), \boldsymbol{\phi}(\vec{x}_2) \rangle}{\sqrt{\|\boldsymbol{\phi}(\vec{x}_1)\|^2 \|\boldsymbol{\phi}(\vec{x}_2)\|^2}} &= \frac{k'(\vec{x}_1, \vec{x}_2)}{\sqrt{k'(\vec{x}_1, \vec{x}_1)\, k'(\vec{x}_2, \vec{x}_2)}} \\
&= \frac{\exp\left(\frac{\langle \vec{x}_1, \vec{x}_2 \rangle}{\sigma^2}\right)}{\sqrt{\exp\left(\frac{\|\vec{x}_1\|^2}{\sigma^2}\right) \exp\left(\frac{\|\vec{x}_2\|^2}{\sigma^2}\right)}} \\
&= \exp\left(\frac{\langle \vec{x}_1, \vec{x}_2 \rangle}{\sigma^2} - \frac{1}{2}\frac{\|\vec{x}_1\|^2}{\sigma^2} - \frac{1}{2}\frac{\|\vec{x}_2\|^2}{\sigma^2}\right) \\
&= \exp\left(-\frac{\|\vec{x}_1 - \vec{x}_2\|^2}{2\sigma^2}\right).
\end{aligned}$$

5. For data that are not linearly separable in $\mathcal{X}$ we can always find a feature space $\mathcal{K}$ by the application of an RBF kernel of sufficiently small bandwidth σ such that the data are linearly separable in $\mathcal{K}$. Given that the minimal distance between inputs in the training sample is Δ we can always choose $\sigma \ll \Delta$ such that the training inputs $\boldsymbol{\phi}(\vec{x}_i)$ in $\mathcal{K}$ become virtually orthogonal and span a space of dimensionality m, in which m training examples are always linearly separable. However, such a procedure may not be advantageous because it may lead to over-fitting.

While the polynomial kernels and the RBF kernel are fairly generic and are frequently applied in domains where there is no extra knowledge available about the data, there exist various approaches for tailoring kernels for given problems. While we will not discuss these special kernels in detail here, it is interesting to consider the general spirit of how kernel methods adapt to various scenarios of learning.

1. One challenge is given by potential transformations under which a classification scheme is expected to show invariant performance. Consider the problem of handwritten digit recognition. Rotations and shifts of handwritten digits as well as a uniform change of gray level and uniform zooms are perceptually natural transformations under which we would wish a pattern recognition system to be invariant. Although this rather general question has not been adequately solved to date kernels have been constructed for the case of only slight transformations that can be linearly approximated in tangent space. This lead to the development of kernels that are robust w.r.t. a set of transformations if these are small enough (Schölkopf et al. 1998).

2. As has been demonstrated earlier, the dimensionality $\dim(\mathcal{K})$ of feature space $\mathcal{K}$ may be very large and even the subspace spanned by all the linear combinations of training samples may be of dimensionality m if the kernel matrix K has full rank. Possibly the simplest approach to dimensionality reduction is given by kernel PCA

(Schölkopf et al. 1998) which performs a principal component analysis on the kernel matrix $\mathbf{K} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}'$ and thus generates a set of non-linear orthogonal kernel features a subset of which can be recombined to give a low-rank approximation of the kernel matrix $\mathbf{K}$, thus effectively adapting the kernel to the covariance structure of the data in feature space. A similar approach has been suggested in (Cristianini et al. 2001) for the construction of a kernel from word-document matrices in information retrieval. These schemes may act as additional regularisers and may reduce the computational complexity of the final classifier.

3. While it is often recommendable in practice to incorporate any apriori knowledge at hand into a learning machine for a given problem this may not always be possible. Apriori knowledge may be difficult to code or might be present in the form of another, possibly unlabelled dataset. In this case it might be favourable to try to *learn the kernel.* This idea is not at all new. People in neural networks research have long argued that kernel techniques are comparable to two-layer feed-forward neural networks, emphasising, however, that kernel methods simply neglect the training of the first layer, which is implicitly given by the choice of the kernel function. Given a parameterised kernel it is shown in Cristianini et al. (1999) how the kernel parameter, e.g. the bandwidth σ in a Gaussian RBF kernel can be adapted automatically. Using one bandwidth parameter per input feature, Guyon et al. (2002) use a similar technique to perform automatic feature selection using gradient descent.

4. A very generic approach to learning the kernel from the data is proposed by Jaakkola and Haussler (1999). First the data is used to train a generative probabilistic model of the data in the maximum likelihood framework. Given such a model log-likelihood $l(x \| \boldsymbol{\theta})$ the feature vector of an input x is given by $\boldsymbol{\phi}(x) = \nabla_{\boldsymbol{\theta}} l(x \| \boldsymbol{\theta})$ and the *Fisher kernel* may be calculated as $k_{\text{fisher}}(x, \tilde{x}) = \left\langle \boldsymbol{\phi}(x), \mathbf{G}_{\boldsymbol{\theta}}^{-1} \boldsymbol{\phi}(x) \right\rangle$ and $\mathbf{G}_{\boldsymbol{\theta}}$ is the Fisher information matrix after which the kernel was named. Note that there is no reason to restrict the Fisher kernel idea to the case $\mathcal{X} = \mathbb{R}^n$. In fact, $\mathcal{X}$ may be any set of objects as long as we can define a generative model over $\mathcal{X}$. In fact, in Jaakkola et al. (1999) the input space $\mathcal{X}$ was given by genetic sequences and the probabilistic model was a hidden Markov model generating such sequences.

5. Considering a kernel as a measure of similarity it is straight-forward to construct kernels based on distances and vice versa (Graepel et al. 1999; Schölkopf 2000). In the simplest case this is achieved by using the relation

$$d(x, \tilde{x}) = \sqrt{k(x, x) - 2k(x, \tilde{x}) + k(\tilde{x}, \tilde{x})}$$

   in analogy to the Euclidean distance in ℓ_2^n.

6. In general it is desirable to be able to define kernels over domains $\mathcal{X}$ other than ℓ_2^n in order to extend the applicability of kernel methods to a wider range of domains with structured representation such as strings or graphs. This line of reasoning has been followed by Haussler (1999) in his work on convolution kernels and by Watkins

(2000) in his work on sequence kernels. The general idea is to measure the overlap between, e.g. two vectors, and use it to define an inner product. For strings such a measure would be the weighted sum of substrings co-occuring in the two strings between which the kernel is evaluated (Lodhi et al. 2001). In Chapter 10 a similar technique will be used to calculate kernels between graphs.

By now it should be clear that the application of kernels constitutes a very powerful tool for the construction of learning machines. In particular, the conceptual separation between learning algorithm and kernel facilitates progress: Given a new kernel it can be used in conjuction with a variety of learning algorithms. In turn, given a new learning algorithm it can be used in conjunction with a wide selection of kernels. In the following let us consider how exactly learning algorithms for linear classifiers can be combined with kernels.

2.3.4. The Nearest Mean Classifier

Consider the simple classifier that calculates the class means $\boldsymbol{\mu}_y$ for each class and assigns a new test point x to the class whose class mean $\boldsymbol{\mu}_y$ is closest to it. More formally, such a classifier is given by

$$\begin{aligned} h_{\mathrm{nm}}(x;\boldsymbol{z}) &\stackrel{\mathrm{def}}{=} \operatorname*{argmin}_y \left(\frac{1}{2}\left\|\mathbf{x}-\boldsymbol{\mu}_y\right\|^2\right) \\ &= \operatorname{sign}\left(\frac{1}{2}\left\|\mathbf{x}-\boldsymbol{\mu}_{-1}\right\|^2 - \frac{1}{2}\left\|\mathbf{x}-\boldsymbol{\mu}_{+1}\right\|^2\right), \end{aligned}$$

and the class means $\boldsymbol{\mu}_y$ are given by

$$\boldsymbol{\mu}_y = \frac{1}{m_y}\sum_{i\in \mathbf{i}_y(\boldsymbol{z})} \mathbf{x}_i$$

Clearly this classifier is a linear classifier as can be seen by multiplying out the argument of the sign-function,

$$\begin{aligned} \frac{1}{2}\left\|\mathbf{x}-\boldsymbol{\mu}_{-1}\right\|^2 - \frac{1}{2}\left\|\mathbf{x}-\boldsymbol{\mu}_{+1}\right\|^2 &= \left\langle \mathbf{x}, \boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1}\right\rangle + \frac{1}{2}\left\|\boldsymbol{\mu}_{-1}\right\|^2 - \frac{1}{2}\left\|\boldsymbol{\mu}_{+1}\right\|^2 \\ &= \langle \mathbf{x}, \mathbf{w}\rangle + w_0\,, \end{aligned}$$

with $\mathbf{w} = \left(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1}\right)$ and $w_0 = \frac{1}{2}\left(\left\|\boldsymbol{\mu}_{-1}\right\|^2 - \left\|\boldsymbol{\mu}_{+1}\right\|^2\right)$. Now, let us take the dual perspective and consider the expansion

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i\,,$$

which compared with the expression for $\mathbf{w}$ gives $\alpha_i = y_i$: the expansion coefficients are just given by the class labels. Introducing a kernel $k(x,\tilde{x}) = \langle \mathbf{x}, \tilde{\mathbf{x}}\rangle$ the nearest mean

classifier can be written as

$$h(x) = \operatorname{sign}\left(\langle \mathbf{x}, \mathbf{w} \rangle + w_0\right) \tag{2.13}$$

$$= \operatorname{sign}\left(\sum_{i=1}^{m} y_i k(x_i, x) + w_0\right), \tag{2.14}$$

where w_0 is given by

$$w_0 = \frac{1}{2}\left(\frac{1}{m_{-1}^2} \sum_{i \in \mathbf{i}_{-1}(\boldsymbol{z})} \sum_{j \in \mathbf{i}_{-1}(\boldsymbol{z})} k(x_i, x_j) - \frac{1}{m_{+1}^2} \sum_{i \in \mathbf{i}_{-1}(\boldsymbol{z})} \sum_{j \in \mathbf{i}_{+1}(\boldsymbol{z})} k(x_i, x_j)\right).$$

Now assume that the class means are equidistant from the origin, $\|\boldsymbol{\mu}_{+1}\|^2 = \|\boldsymbol{\mu}_{-1}\|^2$ and hence $w_0 = 0$. If we further assume that k is a density as defined in Definition 31 we see by comparing (2.14) with Definition 33 that the nearest mean classifier applied in a feature space $\mathcal{K}$ corresponding to a density kernel k results in a kernel density estimation classifier as given by Definition 33. Thus two seemingly different approaches to classification—one example-based, one hypothesis-based—lead to the same effective classification strategy. In addition, it appears that this classifier is not limited to the use of density kernels but can be used in conjunction with any of the kernels that satisfy Mercer´s condition. Since no learning is involved in the application of the nearest mean classifier it is extremely easy to apply in practice. However, there is a price to pay for this simplicity: In contrast to many other kernel classifiers, the nearest mean classifier is a dense classifier in the sense that none of the coefficients α_i are zero, i.e., $|\mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)| = m$. Hence, the evaluation of the nearest mean classifier for a given test example has computational complexity $\mathcal{O}(m)$.

2.3.5. The Nearest Neighbour Classifier as a Kernel Classifier

It turns out that we can also view the KNN classifier considered in Subsection 2.2.1 as a kernel classifier. To this end let us rewrite the Definition 29 for binary classification as

$$h_{\mathrm{Knn}}(x; \boldsymbol{z}) = \operatorname{sign}\left(\sum_{i:\Pi_d(i;x,\boldsymbol{x}) \leq K} y_i\right).$$

Let us first consider the case of $K = 1$. In order to see the relation to a kernel classifier consider Figure 2.7. Note that for simplicity of presentation we disregard the possibility of distance ties. Then we have

$$h_{\mathrm{nn}}(x; \boldsymbol{z}) = \operatorname{sign}\left(\sum_{i:\Pi_d(i;x,\boldsymbol{x})=1} y_i\right) \tag{2.15}$$

$$= \operatorname{sign}\left(\sum_{i=1}^{m} \lim_{\sigma \to \infty} k_{d,\sigma}(x_i, x)\right), \tag{2.16}$$

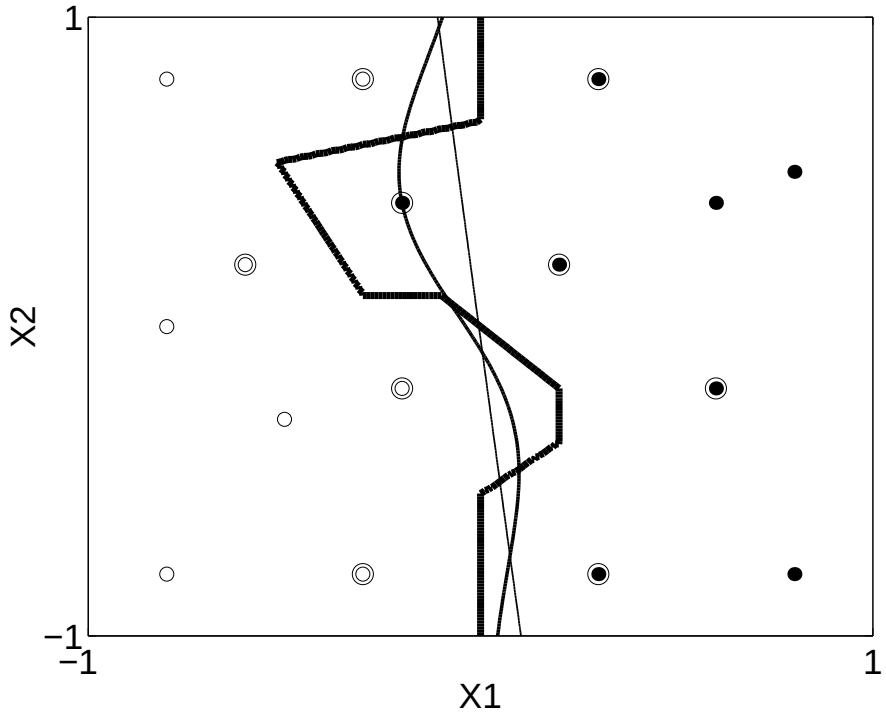


Figure 2.7.: Illustration of the convergence of the classifier based on class-conditional Parzen window density estimation to the 1-NN classifier in $\mathcal{X} = [-1, +1]^2 \subset \mathbb{R}^2$. For $\sigma = 5$ the decision surface (thin line) is almost linear, for $\sigma = 0.4$ the curved line (medium line) results, and for very small $\sigma = 0.02$ the piecewise linear decision surface (thick line) of 1-NN results. For 1-NN only the circled points contribute to the decision surface and form the compression sample.

where $k_{d,\sigma}$ may be any positive definite kernel $k_{d,\sigma} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ such that for all $x \in \mathcal{X}$ and for all $\boldsymbol{x} \in \mathcal{X}^m$ we have for all $j \in \{1, \ldots, m\}$,

$$\lim_{\sigma \to 0} \frac{k_{d,\sigma}\left(d^2\left(x, x_j\right)\right)}{\sum_{i=1}^{m} k_{d,\sigma}\left(d^2\left(x, x_i\right)\right)} = \begin{cases} 1 & \text{for} \quad d\left(x, x_j\right) = \min_{k \in \{1,\ldots,m\}} \left(x, x_k\right) \\ 0 & \text{otherwise} \end{cases} .$$

Such a kernel is, e.g., given by the RBF kernel

$$k_{d,\sigma}\left(r^2\right) = \exp\left(-\frac{r^2}{2\sigma^2}\right) .$$

Comparing (2.16) to the nearest mean classifier (2.14) we see that the 1-NN classifier can be written as a nearest mean classifier assuming $m_{+1} = m_{-1}$ and no threshold w_0.

Also the KNN classifier can be expressed as a kernel classifier. We define the set $I_{K,m} \subset \{1, \ldots, m\}^K$ as the set containing all index vectors of size $K \in \mathbb{N}$,

$$I_{K,m} \stackrel{\text{def}}{=} \left\{ (i_1, \ldots, i_K) \in \{1, \ldots, m\}^K \,|\, i_1 < i_2 < \cdots < i_K \right\} .$$

Then we have

$$\begin{aligned} h_{\mathrm{Knn}}(x;\boldsymbol{z}) &= \operatorname{sign}\left(\sum_{i:\Pi_d(i;x,\boldsymbol{x})\leq K} y_i\right) \\ &= \operatorname{sign}\left(\sum_{\mathbf{i}\in I_{K,m}} \prod_{i\in\mathbf{i}} \lim_{\sigma\to 0} k_{d,\sigma}(x,x_i) \sum_{j\in\mathbf{i}} y_j\right) . \end{aligned}$$

The idea here is that that summands $\mathbf{i}$ will dominate the sum for which the product $\prod_{i\in\mathbf{i}} \lim_{\sigma\to 0} k_{d,\sigma}(x,x_i)$ is the greatest. This will be the case for the subsample $\boldsymbol{z}_{\mathbf{i}}$ of the K training examples closest to x under the metric d. Since we would like to write h_{Knn} as a kernel classifier, we define a product kernel $\tilde{k} : \mathcal{X}^K \times \mathcal{X}^K \to \mathbb{R}$ on sequences $\boldsymbol{x}, \tilde{\boldsymbol{x}}$,

$$\tilde{k}(\boldsymbol{x},\tilde{\boldsymbol{x}}) \stackrel{\mathrm{def}}{=} \prod_{j=1}^{|\boldsymbol{x}|} k(x_j,\tilde{x}_j) .$$

Then we can express h_{Knn} as a kernel classifier

$$h_{\mathrm{Knn}}(x;\boldsymbol{z}) = \operatorname{sign}\left(\sum_{\mathbf{i}\in I_{K,m}} \beta_{\mathbf{i}} \lim_{\sigma\to 0} \tilde{k}_{d,\sigma}(\boldsymbol{x}_{\mathbf{i}},(x,\ldots,x))\right) ,$$

where we have defined

$$\tilde{k}_{d,0}(\boldsymbol{x},\tilde{\boldsymbol{x}}) \stackrel{\mathrm{def}}{=} \lim_{\sigma\to 0} \tilde{k}_{d,\sigma}(\boldsymbol{x},\tilde{\boldsymbol{x}})$$

and

$$\beta_{\mathbf{i}} \stackrel{\mathrm{def}}{=} \sum_{l=1}^{K} \alpha_{i_l} .$$

.

2.3.6. Kernel Perceptron Learning

Let us recall the simple procedure of the perceptron algorithm as outlined in Subsection 2.1 and given in pseudo code in Appendix C. After the initialisation of the weight vector $\mathbf{w} = \mathbf{0}$ updates of the weight vector take the form $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i\mathbf{x}_i$ for misclassified examples. As a consequence, the final weight vector $\mathbf{w}$ can be written as

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i .$$

The weight vector $\mathbf{w}$ never leaves the span of the training examples in feature space. From this representation of the weight vector it is immediately clear how a dual version of the perceptron algorithm with a kernel $k(x,\tilde{x}) = \langle x,\tilde{x}\rangle$ proceeds:

1. Initialise the coefficient vector $\boldsymbol{\alpha}_0 = \mathbf{0}$.

2. Cycle through the training sample $\boldsymbol{z}$ and check for every training example $z_i = (x_i, y_i)$ if it is correctly classified by the current classifier $h_{\boldsymbol{\alpha}_t}$.

   a) if $y_i \left(\sum_{j=1}^{m} (\boldsymbol{\alpha}_t)_j \, k(x_j, x_i) \right) \leq 0$ (wrong classification) then update the coefficient $\alpha_i \leftarrow \alpha_i + y_i$ and proceed with $t \leftarrow t + 1$.

   b) if $y_i \left(\sum_{j=1}^{m} (\boldsymbol{\alpha}_t)_j \, k(x_j, x_i) \right) > 0$ (correct classification) then proceed.

3. Repeat 2. until all training examples in $\boldsymbol{z}$ are classified correctly.

Note that the expansion of the weight vector leads to an even simpler update rule than in the primal perceptron algorithm. Of course, we can again apply the minover heuristic to select that training example z_i whose input vector $\mathbf{x}_i$ has the smallest overlap with the current weight vector, i.e., at each turn one chooses training example $z_{i(t)}$ with $i(t) = \operatorname{argmin}_{i \leq m} y_i \sum_{j=1}^{m} \alpha_j k(x_i, x_j)$. Also a margin ρ^2 can be enforced by using the update condition $y_i \sum_{j=1}^{m} \alpha_j k(x_i, x_j) \leq \rho^2$.

In comparison to the nearest mean classifier we might ask what we have gained. The main improvement over the nearest mean classifier lies in the sparseness of the dual perceptron solution. Empirically, it turns out that the solutions given by the dual perceptron are very sparse, i.e., the number $|\mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)|$ is small. Since the computational complexity of evaluating the resulting kernel classifier is $\mathcal{O}(|\mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)|)$ we have considerable computational savings at classification time at the price of higher computational costs at training time. Depending on the given circumstances, either solution may be preferable.

Remark 1 *(Soft margin perceptron).* Of course, the dual perceptron algorithm in the form given depends on data samples $\boldsymbol{z}$ that are linearly separable (see Definition 20) in feature space. As mentioned earlier this may not always be the case in practice. In the context of the maximum margin classifier in Subsection 2.1 we discussed two ways of handling in the quadratic programme training-data that were not linearly separable. In the dual formulation, the 1-norm and 2-norm slack penalisation lead to box constraints $0 \leq \alpha_i \leq C$ and the inner product modification $\langle \mathbf{x}_i, \mathbf{x}_j \rangle \rightarrow \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \frac{1}{C} \mathbf{l}_{i=j}$, respectively. Adapting these two strategies to the dual perceptron leads to the following two heuristics.

1-**norm**: Run the dual perceptron algorithm as usual but cut off any α_i at $\alpha_i = C$, and let it not exceed that value.

2-**norm**: Modify the kernel according to $k(x_i, x_j) \rightarrow k(x_i, x_j) + \frac{1}{C} \delta_{ij}$ or in matrix notation $\mathbf{K} \rightarrow \mathbf{K} + \frac{1}{C} \mathbf{I}_m$.

With these modifications the kernel perceptron algorithm constitutes a versatile and flexible tool for binary classification. Of course, we need to find some way of adjusting the additional kernel parameter C.

2.3.7. Support Vector Learning

The support vector machine (Cortes and Vapnik 1995) is—like most good ideas—essentially the result of combining two ingredients:

1. The maximum margin classifier (see Subsection 2.1.3)
2. Mercer kernels (see Subsections 2.3.2 and 2.3.3)

Let us reconsider the Wolf dual of the quadratic programme derived for the maximum margin classifier as given in Theorem 4,

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle ,$$

which is to be minimised w.r.t. $\boldsymbol{\alpha}$ subject to $\sum_{i=1}^{m} y_i \alpha_i = 0$ and $\alpha_i \geq 0$ for all $i \in \{1, \ldots, m\}$. In order to carry out the classification in a feature space $\mathcal{K}$ we can again replace the inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ by the kernel $k(x_i, x_j)$. Thus the data enter the optimisation problem only in terms of the kernel matrix $\mathbf{K}$. Given the solution vector $\boldsymbol{\alpha}^*$ we have for the resulting weight vector $\mathbf{w}^*$,

$$\mathbf{w}^* = \sum_{i \in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)}^{m} \alpha_i^* y_i \mathbf{x}_i ,$$

and for the resulting classifier

$$\begin{aligned} h(x) &= \operatorname{sign}\left(\langle \mathbf{w}^*, \mathbf{x} \rangle + w_0\right) \\ &= \operatorname{sign}\left(\sum_{i \in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \alpha_i^* y_i k(x_i, x) + w_0 \right) , \end{aligned}$$

where w_0 is given by

$$w_0^* = -\frac{1}{2} \left(\max_{y_j = -1} \sum_{i \in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \alpha_i^* y_i k(x_i, x_j) + \min_{y_j = +1} \sum_{i \in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \alpha_i^* y_i k(x_i, x_j) \right) .$$

Most of what has been said about the maximum margin classifier applies directly to the support vector machine, and the reader is referred to Subsection 2.1.3. Of course, the 1-norm and 2-norm soft margin variants of the maximum margin classifier can be used in conjunction with Mercer kernels as well. In the corresponding optimisation problems given in Theorem 5 and Theorem 6 the inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ are simply replaced by the kernel $k(x_i, x_j)$.

With regard to the implementation of support vector machines various optimisation methods have been suggested for solving the quadratic programmes involved. These techniques range from simple gradient descent algorithms such as the kernel adatron (Frieß et al. 1998) to the application of standard QP-solvers such as LOQO (Vanderbei 1994) based, e.g., on interior point methods. A particularly elegant implementation is the *sequential minimal optimiser (SMO),* that reduces the problem to chunks of two inputs, as scenario that can be solved analytically (Platt 1998).

2.3.8. Other Kernel-Based Learning Algorithms

The idea of *kernelisation* of known linear learning algorithms has proved to be very fruitful, at least in the amount of papers written. Various optimality criteria are possible and a great number of algorithms has been suggested to address various aspects of classification. For the sake of completeness I will name just a few of the recent developments here. An alternative to the soft-margin parameterisation as given in Theorems 5 and 6 was suggested under the name of ν-support vector machines in (Schölkopf et al. 2000). The ν-parameterisation has a natural interpretation in terms of the fraction of support vectors and is thus expected to be easier to adapt. Also, instead of constructing a quadratic programme, people have conceived of linear classifiers that are the solution to linear programmes under the name of *linear programming machines* (Smola et al. 1999; Graepel et al. 1999) an idea that goes back to Duda and Hart (1973). Fisher´s well-known linear discriminant was generalised to kernel spaces under the name of kernel fisher discriminant (KFD) in Mika et al. (2001). The sparseness of the solutions can be improved by the *relevance vector machine* (Tipping 2001) that imposes a prior on $\boldsymbol{\alpha}$ leading to sparse solutions. Coming from a slightly different background *Gaussian process classification* (Gibbs and Mackay 1998; Opper and Winther 2000b) also results in a kernel technique for classification, where the kernel is interpreted as a covariance function of a multi-dimensional Gaussian distribution.

Beside the problem of binary classification other supervised learning tasks have been tackled by the kernel approach: Multi-class classification problems are frequently encountered in practice and various kernel methods have been developed for solving them, either based on a combination of binary classifiers as discussed in Section 1.2 or by directly solving an optimisation problem involving multiple classes (Weston and Watkins 1999). Also the problem of regression can be solved in the kernel-framework. Support vector regression using the ε -insensitive loss function was suggested by Vapnik (1995) and carried further, for example, in Drucker et al. (1997). Of course, Gaussian processes were originally developed in a regression framework and were recently rediscovered as a substitute for feed-forward neural networks (Williams and Rasmussen 1996). Finally, the task of ordinal regression that aims at predicting ranks, i.e., ordered classes, has been successfully tackled in the kernel framework (Herbrich, Graepel, and Obermayer 2000).

In the context of unsupervised learning various classical methods are amenable to *kernalisation*. Probably the most basic algorithm is *kernel PCA* (Schölkopf et al. 1996), which performs principal component analysis in kernel space. Other approaches include *topographic kernel clustering* (Graepel and Obermayer 1998; Girolami 2001), and novelty detection using so-called *single-class SVMs* (Schölkopf et al. 2000).

3. Generalisation Theory of Classification

Computational/statistical learning theory creates and studies models of learning from examples. Since the phenomenon of learning in general is very complex and constitutes one of the highest cognitive abilities to be found in nature all the models conceived must have a very high degree of abstraction. While real-world biological learning processes involve explorative learning in dynamic environments, learning to learn, incomplete information, uncertainty, delayed and possibly lethal rewards, various time scales (between short-term and evolutionary learning), interacting objectives, biological constraints, and similar complications, successful models must isolate essential features of learning to be able to provide meaningful results. Nonetheless various aspects of learning such as knowledge representation, computational complexity, and generalisation may be studied under appropriate model assumptions.

We will focus on the question of generalisation. The key question is this: How is it possible to infer general dependencies from a particular finite sample of empirical data? In other words: How is generalisation possible, and what is necessary in order to achieve it? Conceptually, the question of generalisation and abstraction lies at the heart of learning theory as much as it lies at the heart of epistemology. In how far can we draw general conclusions from particular observations? Or more dramatic: What logical reason do we have to believe that the sun will rise tomorrow, if all we really know is that it has risen on every day of our lives before? Of course, learning theory cannot be expected to answer any of these fundamental questions directly. After all its conclusions crucially depend on the assumptions of the model. However, we might expect to gain at least some insights about the nature of learning and generalisation by studying particular models of learning.

The second more down-to-earth motivation for studying learning theory lies in the desire to understand and improve learning algorithms. While the abstract notions of learning theory might be far removed from biological learning in general, they are certainly able to conceptually capture interesting aspects of machine learning. There exists ample empirical evidence that machine learning algorithms do work well in practice (see, e.g., Michie et al. (1994) for an extensive collection of empirical results on a selection of benchmark problems). Learning theory provides the tools for analysing machine learning algorithms, to pinpoint criteria for their success, and to provide insights into the nature of their workings that allow us to further improve their performance. In particular, learning theory may offer solutions to the problem of model selection—which is crucial to the success of any learning machine.

The presentation of learning theoretical frameworks in this chapter will be based on

the learning scenario outlined in the introductory chapter. Section 3.1 gives an introduction to the PAC/VC framework of learning and provides the necessary background for subsequent chapters. The following section on Bayesian learning, Section 3.2, shows how the process of learning is modelled in a Bayesian framework. Both these sections together provide the necessary background for Section 3.3 on PAC-Bayesian theory, which reconciles the notions of PAC and Bayesian learning.

3.1. The PAC/VC Framework

The basic notions of PAC (probably approximately correct) learning were introduced to the machine learning community by Valiant (1984) in a seminal paper that essentially started the research field of *computational learning theory*. However, similar and more general models for frequentist inference regarding rates of uniform convergence had been studied earlier by Vapnik and Chervonenkis (1971) in the Soviet Union and were only later introduced to the machine learning community by Blumer et al. (1989). Many results of what is now known as *statistical learning theory* can already be found in Vapnik and Chervonenkis (1974). More recent presentations include Vapnik (1995) and Vapnik (1998), and the introductory book by Vidyasagar (1997). Other books on the subject include Kearns and Vazirani (1994), from the computational learning theory perspective and the excellent book by Herbrich (2001), focusing on learning theory for kernel classifiers. Asymptotic results, in particular for example-based classification rules, are emphasised in the book by Devroye et al. (1996).

Based on the model presented in Chapter 1 the essential feature of this modelling approach is that the m examples z_i from which a given dependency is to be inferred are drawn independently from the same probability distribution $\mathbf{P}_\mathsf{Z}$ resulting in a training sample $\boldsymbol{z}$ drawn from a joint distribution $\mathbf{P}_{\mathsf{Z}^m} = \prod_{i=1}^{m} \mathbf{P}_{\mathsf{Z}_i}$. The model thus assumes a stationary data source with independent observations, an assumption typically invalid if the examples result from a physical non-random time series. The iid assumption also implies that the learning machine is completely passive and has not influence whatsoever over the generation or selection of training examples. The criterion for successful learning in the model is a low prediction error $R[h]$ of an hypothesis h as defined in Definition 5. For binary classification $R[h]$ is the probability that h misclassifies an example drawn from $\mathbf{P}_\mathsf{Z}$: We assume that the test examples to be encountered are distributed identically to the available training data. While these restrictions seem necessary to obtain meaningful results, the PAC model does not make any assumptions about the distribution $\mathbf{P}_\mathsf{Z}$ itself, a property often referred to as *distribution free*. This is in contrast to a wide range of parametric statistical methods that typically make assumptions on the distribution of the data, and are hence often able to make more precise statements at the cost of stricter assumptions.

3.1.1. PAC Bounds on the Prediction Error

The original PAC model presented in Valiant (1984) defines the learnability of an hypothesis space $\mathcal{H}$ in terms of two quantities: *computational complexity* and *sample com-*

plexity. We will focus here on the aspect of sample complexity, which in this work takes the equivalent form of upper bounds on the prediction error $R[h]$ in terms of the number of samples m, the desired confidence δ, and quantities that are related to the hypothesis space $\mathcal{H}$. Given an algorithm $\mathcal{A}$ (see Definition 7) as a function of a random training sample $\mathbf{Z}$ the prediction error $R[\mathcal{A}(\mathbf{Z})]$ of the resulting hypothesis is a random variable. In the PAC framework we aim at bounding the probability that the prediction error exceeds a certain value ε. In particular, let us consider how we can derive bounds on the risk of empirical risk minimisers $\mathcal{A}_{\text{erm}}$. To this end let us consider the two hypotheses $h_{\boldsymbol{z}}$ and $h^* \in \mathcal{H}$:

1. $h_{\boldsymbol{z}} \stackrel{\text{def}}{=} \mathcal{A}_{\text{erm}}(\boldsymbol{z}) \in \mathcal{H}$ is an hypothesis that minimises the empirical risk, i.e. $h_{\boldsymbol{z}} = \operatorname{argmin}_{h \in \mathcal{H}} \hat{R}[h, \boldsymbol{z}]$.
2. $h^* \in \mathcal{H}$ is an hypothesis that minimises the true risk, i.e. $h^* = \operatorname{argmin}_{h \in \mathcal{H}} R[h]$.

For each of these two hypotheses we can evaluate the empirical risk $\hat{R}$ and the true risk R leading to the four quantities $\hat{R}[h_{\boldsymbol{z}}, \boldsymbol{z}]$, $\hat{R}[h^*, \boldsymbol{z}]$, $R[h_{\boldsymbol{z}}]$ and $R[h^*]$. From the definition of the empirical risk minimiser we have

$$\hat{R}[h^*, \boldsymbol{z}] - \hat{R}[h_{\boldsymbol{z}}, \boldsymbol{z}] \geq 0 \,.$$

Now consider the following chain of inequalities

$$\begin{aligned} R[h_{\boldsymbol{z}}] - R[h^*] &\leq R[h_{\boldsymbol{z}}] - R[h^*] + \left(\hat{R}[h^*, \boldsymbol{z}] - \hat{R}[h_{\boldsymbol{z}}, \boldsymbol{z}]\right) \\ &= \left|\left(R[h_{\boldsymbol{z}}] - \hat{R}[h_{\boldsymbol{z}}, \boldsymbol{z}]\right) - \left(R[h^*] - \hat{R}[h^*, \boldsymbol{z}]\right)\right| \\ &\leq \left|R[h_{\boldsymbol{z}}] - \hat{R}[h_{\boldsymbol{z}}, \boldsymbol{z}]\right| + \left|R[h^*] - \hat{R}[h^*, \boldsymbol{z}]\right| \\ &\leq 2 \cdot \sup_{h \in \mathcal{H}} \left|R[h] - \hat{R}[h, \boldsymbol{z}]\right| \,, \end{aligned}$$

where we applied the triangle inequality in the third line and upper bounded the deviations of empirical from true risk of two specific hypotheses $h_{\boldsymbol{z}}, h^* \in \mathcal{H}$ each time by the worst case deviation $\sup_{h \in \mathcal{H}} \left|R[h] - \hat{R}[h, \boldsymbol{z}]\right|$. As a consequence of this reasoning in order to bound the deviation of the true risks between the solution $h_{\boldsymbol{z}}$ of an ERM algorithm and the best hypothesis h^* we must be able to bound the deviation between empirical risk and true risk uniformly over the hypothesis space $\mathcal{H}$. In the original formulation of PAC learning (Valiant 1984) it is assumed that the "true" or "teacher" hypothesis is in fact contained in $\mathcal{H}$ and hence that the probability distribution $\mathbf{P}_{\mathsf{Z}}$ is given by $\mathbf{P}_{\mathsf{XY}} = \mathbf{P}_{\mathsf{Y}|\mathsf{X}} \mathbf{P}_{\mathsf{X}}$ with

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y) = \mathbf{I}_{h^*(x)=y} \,.$$

As a consequence, in the original PAC framework we have that $R[h^*] = 0$ and that $\hat{R}[h_{\boldsymbol{z}}, \boldsymbol{z}] = 0$, which implies that under these conditions

$$R[h_{\boldsymbol{z}}] - R[h^*] = R[h_{\boldsymbol{z}}] \leq \sup_{\left\{h \in \mathcal{H} : \hat{R}[h, \boldsymbol{z}] = 0\right\}} R[h] \,.$$

Incorporating the more general case of non-zero training error sometimes referred to as *agnostic PAC learning* (Kearns and Schapire 1994), we thus require that the learning algorithm solves the learning problem in a probably approximately correct (PAC) way in the sense of the following definition.

Definition 39 (PAC prediction error bound). Given an hypothesis space $\mathcal{H}$, a probability measure $\mathbf{P}_{\mathsf{Z}}$, a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, a learning algorithm $\mathcal{A}$, and a real number $\delta \in (0, 1]$ we call a function $\varepsilon : \mathbb{N} \times \cdots \times (0, 1] \to \mathbb{R}$, a *PAC prediction error bound* if it holds for every sample size $m > 0$ that

$$\mathbf{P}_{\mathsf{Z}^m} \left(R\left[\mathcal{A}\left(\mathbf{Z}\right)\right] > \varepsilon\left(m, \ldots, \delta\right)\right) < \delta \,.$$

As we will later see, the quantities entering a PAC bound $\varepsilon\left(m, \ldots, \delta\right)$ typically involve some complexity measure of the hypothesis space $\mathcal{H}$, the training error $\hat{R}\left[h, \boldsymbol{z}\right]$, and may even involve data-dependent quantities such as empirical margin or sparseness of the solution. In a sense, PAC results are not unlike statistical tests in the sense that the probability is at most δ that the data has misled us, that is, that the bound on the prediction error is invalid.

Let us consider an example of a PAC bound for the case that the hypothesis space $\mathcal{H}$ is of finite cardinality $|\mathcal{H}|$. Then we have the following theorem

Theorem 11 (Finite cardinality PAC bound for consistent hypotheses). *Given an input space $\mathcal{X}$ and the output space $\mathcal{Y} \in \{+1, -1\}$, for any hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ of finite cardinality $|\mathcal{H}|$, for any probability measure $\mathbf{P}_{\mathsf{Z}}$, for every sample size $m > 0$, and for all $\delta \in (0, 1]$ we have with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that for any consistent hypothesis $h \in \mathcal{H}$ the prediction error $R\left[h\right]$ is bounded by*[1]

$$R\left[h\right] < \varepsilon\left(m, |\mathcal{H}|, \delta\right)$$

with

$$\varepsilon\left(m, |\mathcal{H}|, \delta\right) = \frac{1}{m}\left(\log|\mathcal{H}| + \log\frac{1}{\delta}\right) \,.$$

Proof. Given an hypothesis $h \in \mathcal{H}$ with $R\left[h\right] > \varepsilon$ we can bound the probability of the event $\hat{R}\left[h, \boldsymbol{z}\right] = 0$ on an iid training sample $\boldsymbol{z}$ of size m for any $h \in \mathcal{H}$ by

$$\mathbf{P}_{\mathsf{Z}^m}\left(\hat{R}\left[h, \mathbf{Z}\right] = 0\right) \leq \exp\left(-\varepsilon m\right) \,,$$

which follows from the *binomial tail bound*, Theorem 58. The probability that at least one of the $|\mathcal{H}|$ hypotheses $h \in \mathcal{H}$ is consistent with the training sample $\boldsymbol{z}$ is then bounded by

$$\mathbf{P}_{\mathsf{Z}^m}\left(\bigvee_{h\in\mathcal{H}} \hat{R}\left[h, \mathbf{Z}\right] = 0\right) \leq \sum_{h\in\mathcal{H}} \mathbf{P}_{\mathsf{Z}^m}\left(\hat{R}\left[h, \mathbf{Z}\right] = 0\right) \leq |\mathcal{H}| \exp\left(-\varepsilon m\right) \,, \tag{3.1}$$

[1] Note that the statement here is phrased in the jargon of learning theory. More precisely the statement is: If it where the case that $R\left[h\right] \geq \varepsilon\left(m, |\mathcal{H}|, \delta\right)$ then the probability of oberserving $\hat{R}\left[h\right] = 0$ would be small, i.e., less than δ.

by the application of the *union bound,* Theorem 57. Setting the right-hand side of (3.1) to δ and solving for ε yields the desired result. □

This simple bound already contains the crucial ingredients for any PAC bound. In particular, the complexity of the hypothesis space $\mathcal{H}$ is indicated by its cardinality $|\mathcal{H}|$: The more complex $\mathcal{H}$ is the more examples we need to achieve the desired accuracy with high probability. Theorem 12 is a uniform convergence result because the convergence statement holds uniformly over all hypotheses $h \in \mathcal{H}$. This uniform convergence is proved by the application of a non-uniform convergence result like the binomial tail bound combined with the union bound.

Of course it may not always be the case that we can achieve an empirical risk of $\hat{R}[h, \boldsymbol{z}] = 0$. In this case we cannot apply the binomial tail bound but must instead bound the deviation of the empirical risk from the true risk. The only difference to the previous theorem, Theorem 12, is that we replace the binomial tail bound by Hoeffding´s inequality, Theorem 61.

Theorem 12 (Finite cardinality PAC bound). *Given an input space $\mathcal{X}$ and the output space $\mathcal{Y} \in \{+1, -1\}$, for any hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ of finite cardinality $|\mathcal{H}|$, for any probability measure $\mathbf{P}_{\mathsf{Z}}$, for every sample size $m > 0$, and for all $\delta \in (0, 1]$ we have with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that for any hypothesis $h \in \mathcal{H}$ the prediction error $R[h]$ is bounded by*

$$R[h] < \varepsilon\left(m, |\mathcal{H}|, \hat{R}[h, \boldsymbol{z}], \delta\right)$$

with

$$\varepsilon\left(m, |\mathcal{H}|, \hat{R}[h, \boldsymbol{z}], \delta\right) = \hat{R}[h, \boldsymbol{z}] + \sqrt{\frac{1}{2m}\left(\log|\mathcal{H}| + \log\frac{1}{\delta}\right)}\,.$$

Proof. Given an hypothesis $h \in \mathcal{H}$ with $R[h]$ we can bound the probability of the event $R[h] - \hat{R}[h, \boldsymbol{z}] > \varepsilon$ on an iid training sample $\boldsymbol{z}$ of size m by Hoeffding´s inequality, Theorem 61, that is for any $h \in \mathcal{H}$,

$$\mathbf{P}_{\mathsf{Z}^m}\left(R[h] - \hat{R}[h, \mathbf{Z}] > \varepsilon\right) \leq \exp\left(-2m\varepsilon^2\right)\,,$$

Again we apply the union bound in order to bound the probability of the event that the supremum $\sup_{h\in\mathcal{H}}\left(R[h] - \hat{R}[h, \boldsymbol{z}]\right)$ of the deviation between empirical and true risk exceeds a value ε,

$$\begin{aligned}\mathbf{P}_{\mathsf{Z}^m}\left(\sup_{h\in\mathcal{H}}\left(R[h] - \hat{R}[h, \mathbf{Z}]\right) > \varepsilon\right) &\leq \sum_{h\in\mathcal{H}} \mathbf{P}_{\mathsf{Z}^m}\left(\left|\hat{R}[h, \mathbf{Z}] - R[h]\right| > \varepsilon\right) \\ &\leq |\mathcal{H}| \exp\left(-2m\varepsilon^2\right)\,.\end{aligned}$$

Setting the right-hand side of the equation to δ and solving for ε yields the desired result. □

3.1.2. VC Bounds for Infinitely Many Hypotheses

One of the main contributions of Vapnik & Chervonenkis´ theory is the extension of uniform convergence bounds to the case of hypothesis spaces $\mathcal{H}$ of infinite cardinality $|\mathcal{H}|$. While our subsequent results will not crucially depend on the VC approach we will nonetheless give a short outline of the arguments involved. To this end we focus on the case of consistent hypotheses h, i.e. the case of $\hat{R}[h, \boldsymbol{z}] = 0$.

The problem with hypothesis spaces of infinite cardinality is that the application of the union bound would give a sum over infinitely many finite quantities and would thus yield trivial results. The key insight of VC theory is that the capacity or complexity of a hypothesis class is not determined merely by the number of hypotheses involved but by the number of hypotheses that yield different results on a finite training sample. This leads to the characterisation of hypothesis spaces $\mathcal{H}$ by a quantity known as their VC dimension. The argument can roughly be divided into three steps

First we bound the probability of a large deviation of the empirical risk $\hat{R}[h, \mathbf{Z}]$ from the true risk $R[h]$ by twice the probability of a large deviation of one empirical risk $\hat{R}[h, \mathbf{Z}_1]$ from another empirical risk $\hat{R}[h, \mathbf{Z}_2]$ on a so-called ghost-sample of the same size m. This relation is known as the basic lemma (Vapnik 1998) and serves to eliminate the true risk from the bounding problem. In the case of consistent hypotheses the basic lemma reads:

Theorem 13 (Basic lemma). *Given an input space $\mathcal{X}$, the output space $\mathcal{Y} \in \{+1, -1\}$, for any hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, for any probability measure $\mathbf{P}_{\mathsf{Z}}$, for every $\varepsilon > 0$ and every sample size $m > \frac{2}{\varepsilon}$ we have that*

$$\mathbf{P}_{\mathsf{Z}^m}\left(\exists h \in \mathcal{H} : \hat{R}[h, \mathbf{Z}] = 0 \wedge R[h] > \varepsilon\right) \leq$$
$$2 \cdot \mathbf{P}_{\mathsf{Z}^{2m}}\left(\exists h \in \mathcal{H} : \hat{R}[h, \mathbf{Z}_1] = 0 \wedge \hat{R}[h, \mathbf{Z}_2] > \frac{\varepsilon}{2}\right). \tag{3.2}$$

The intuition behind the basic lemma is that if two empirical means do not deviate much, then it is quite likely that also the mean does not deviate too much from its expectation. Proofs can be found in Vapnik (1998), Anthony and Bartlett (1999), and Herbrich (2001).

In the second step the right-hand side of the basic lemma is bounded by a symmetrisation technique based on the fact that the probability of an event on the sample space is invariant under permutations of the training sample. Let $\Upsilon(\mathbf{Z}_1\mathbf{Z}_2)$ be the event

$$\Upsilon(\mathbf{Z}_1\mathbf{Z}_2) \equiv \left(\hat{R}[h, \mathbf{Z}_1] = 0 \wedge \hat{R}[h, \mathbf{Z}_2] > \frac{\varepsilon}{2}\right).$$

Clearly, from the iid assumption we have that the probability of Υ is invariant under any permutation $\pi : \{1, \ldots, 2m\} \to \{1, \ldots, 2m\}$ of the training (and ghost) sample,

$$\mathbf{P}_{\mathsf{Z}^{2m}}(\Upsilon(\mathsf{Z}_1, \ldots, \mathsf{Z}_{2m})) = \mathbf{P}_{\mathsf{Z}^{2m}}\left(\Upsilon\left(\mathsf{Z}_{\pi(1)}, \ldots, \mathsf{Z}_{\pi(2m)}\right)\right). \tag{3.3}$$

Let Π_{2m} be the set $\Pi_{2m} \stackrel{\text{def}}{=} \{1,\ldots,2m\}^{\{1,\ldots,2m\}}$ of all permutation functions π on $\{1,\ldots,2m\}$. Then we have

$$\begin{aligned}
\mathbf{P}_{\mathsf{Z}^{2m}}\left(\Upsilon\left(\mathsf{Z}_1,\ldots,\mathsf{Z}_{2m}\right)\right) &= \frac{1}{|\Pi_{2m}|}\sum_{\pi\in\Pi_{2m}} \mathbf{P}_{\mathsf{Z}_1^m\mathsf{Z}_2^m}\left(\Upsilon\left(\mathsf{Z}_{\pi(1)},\ldots,\mathsf{Z}_{\pi(2m)}\right)\right) \\
&= \mathbf{E}_{\mathsf{Z}^{2m}}\left[\frac{1}{|\Pi_{2m}|}\sum_{\pi\in\Pi_{2m}} \mathbf{I}_{\Upsilon\left(\mathsf{Z}_{\pi(1)},\ldots,\mathsf{Z}_{\pi(2m)}\right)}\right] \\
&\leq \max_{\boldsymbol{z}\in\mathcal{Z}^m}\left(\frac{1}{|\Pi_{2m}|}\sum_{\pi\in\Pi_{2m}} \mathbf{I}_{\Upsilon\left(z_{\pi(1)},\ldots,z_{\pi(2m)}\right)}\right).
\end{aligned}$$

In effect, this technique transforms the problem of bounding the probability

$$\mathbf{P}_{\mathsf{Z}^{2m}}\left(\Upsilon\left(\mathsf{Z}_1,\ldots,\mathsf{Z}_{2m}\right)\right)$$

to the problem of counting permutations π of the double-sample with the property that the event Υ holds. In order to make an assertion that is independent of the particular $2m$ sample points we restrict ourselves to the set $\tilde{\Pi}_{2m}$ of permutations resulting from the swapping of pairs of corresponding points in the two samples. Then the fraction of permutations for which Υ holds is given by

$$\frac{1}{\left|\tilde{\Pi}_{2m}\right|}\sum_{\pi\in\tilde{\Pi}_{2m}} \mathbf{I}_{\Upsilon\left(z_{\pi(1)},\ldots,z_{\pi(2m)}\right)} = \frac{2^{m\left(1-\frac{\varepsilon}{2}\right)}}{2^m} = 2^{-\frac{m\varepsilon}{2}}$$

In the third step, we observe that in the case of binary classification for a double sample $\boldsymbol{z}$ of size $2m$ there exist at most 2^{2m} different labellings. Hence, any hypothesis space $\mathcal{H}$ is partitioned into at most 2^{2m} equivalence classes on a $2m$-sample and is thus effectively finite. In order to uniformly bound the probability on the right-hand side of (3.2) we need a bound on the number $\mathcal{N}_{\mathcal{H}}(2m)$ of equivalence classes of $\mathcal{H}$ on $2m$ inputs, a quantity known as the growth function.

Definition 40 (Growth function). Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1,-1\}$, and an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ the function $G_{\mathcal{H}} : \mathbb{N} \to \mathbb{N}$ given by

$$G_{\mathcal{H}}(m) = \max_{\boldsymbol{x}\in\mathcal{X}^m} \left|\left\{\left(h(x_1),\ldots,h(x_m)\right) \mid h \in \mathcal{H}\right\}\right|,$$

is called the *growth function* of $\mathcal{H}$.

Clearly, the growth function is bounded by $G_{\mathcal{H}}(m) \leq 2^m$. If equality holds then we say that the m-sample $\boldsymbol{x}$ is shattered according to the following definition.

Definition 41 (Shattering). Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1,-1\}$, an input sample $\boldsymbol{x}$ of size m, and an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ we say that $\boldsymbol{x}$ is *shattered* by $\mathcal{H}$ if

$$\left|\left\{\left(h(x_1),\ldots,h(x_m)\right) \mid h \in \mathcal{H}\right\}\right| = 2^m.$$

If for any sample size m there exists an input sample $\boldsymbol{x}$ such that $\mathcal{H}$ shatters $\boldsymbol{x}$ then the growth function is, for all m, given by $G_{\mathcal{H}}(m) = 2^m$. The final cornerstone of the VC theory that we will consider here is the analysis of the case that there exists a finite maximum sample size of a shattered set.

Definition 42 (VC dimension). Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, and an hypothesis space $\mathcal{H}$ we call the largest number $d \in \mathbb{N}$ such that there exists an input sample $\boldsymbol{x}$ of size $m = d$ that is shattered by $\mathcal{H}$ the VC dimension of $\mathcal{H}$ and write

$$d = \mathrm{VCdim}(\mathcal{H}) .$$

If it exists, the VC dimension provides a bound on the growth function in the sense of the following theorem (Vapnik 1998):

Theorem 14 (Bound on growth function). *Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, and an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ of VC dimension d the growth function $G_{\mathcal{H}}$ is bounded for all $m \geq d$ by*

$$G_{\mathcal{H}}(m) \leq \left(\frac{em}{d}\right)^d .$$

The growth function, and its one-number summary, the VC dimension, provide a measure for the richness, capacity or complexity of a given hypothesis space $\mathcal{H}$. Given the ingredients so far developed we can now bound the probability on the left-hand side of the basic lemma, Lemma 13, by

$$\mathbf{P}_{\mathbf{Z}^m}\left(\exists h \in \mathcal{H} : \hat{R}[h, \mathbf{Z}] = 0 \wedge R[h] > \varepsilon\right) \leq 2 \cdot \left(\frac{2em}{d}\right)^d 2^{-\frac{\varepsilon m}{2}} ,$$

where $d = \mathrm{VCdim}(\mathcal{H})$. Putting all the ingredients together then yields a VC bound for hypothesis spaces of infinite cardinality.

Theorem 15 (VC prediction error bound for consistent classifiers). *Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, for any hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ of finite VC dimension d, for any probability measure $\mathbf{P}_{\mathbf{Z}}$, for every sample size $m > 0$, and for all $\delta \in (0, 1]$ we have with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that the prediction error $R[h]$ of a consistent hypothesis h is bounded by*

$$R[h] < \varepsilon(m, d, \delta)$$

with

$$\varepsilon(m, d, \delta) = \frac{2}{m}\left(d \log_2 \frac{2em}{d} + \log_2 \frac{2}{\delta}\right) ,$$

provided that $d \leq m$.

However, VC theory not only provides upper bounds on the prediction error. In order to show that these bounds are tight in principle and thus characterise learnability in the PAC/VC sense the theory also provides lower bounds on the prediction error in the sense of the following theorem (Ehrenfeucht et al. 1989).

Theorem 16 (VC lower bound). *Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, for any hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ of finite VC dimension d there exist probability distributions $\mathbf{P}_{\mathbf{Z}}$ such that with probability of at least $\delta \in (0, 1]$ over m examples, the prediction error of a consistent hypothesis h is bounded from below by*

$$R[h] > \max\left(\frac{d-1}{32m}, \frac{\log\frac{1}{\delta}}{m}\right).$$

This lower bound effectively shows that there exist probability distributions $\mathbf{P}_{\mathbf{Z}}$ that in fact force a learning algorithm to require a large training sample in the case that the VC dimension of the hypothesis space $\mathcal{H}$ is large. Luckily, the lower bound does not hold for all distributions, indicating that the learning algorithm may be lucky and encounter a distribution under which few training examples are sufficient for learning. The bounds presented so far, however, are unable to detect such a distribution. The reason is that these bounds can effectively be evaluated before learning: No quantity from the learning process itself enters these bounds. In order to be valid *apriori* these bounds need to be *data-independent* and make worst case assumptions on the distributions to be encountered.

In order to be able to apply VC theory to the hypothesis space $\mathcal{H}_{\mathcal{K}}$ of linear classifiers it is necessary to determine the VC dimension of $\mathcal{H}_{\mathcal{K}}$ as given, for example in Anthony and Bartlett (1999).

Theorem 17 (VC dimension of linear classifiers). *The VC dimension* $\mathrm{VCdim}(\mathcal{H}_{\mathcal{K}})$ *of the hypothesis space $\mathcal{H}_{\mathcal{K}}$ of linear classifiers in $\mathcal{K} \subseteq \ell_2^n$ is n.*

As a consequence of Theorem 17 and the lower bound provided by Theorem 16 one should expect that learning in high-dimensional feature spaces $\mathcal{K}$ should be impossible: Any attempt to learn by a linear classifier in n dimensions (implying a VC dimension of $d \approx n$) in a scenario with $m < n$ or even $m \approx n$ training examples should fail in general. However, it turns out that if we incorporate certain properties of the data into the generalisation error bound *aposteriori,* i.e., we make the bound *data-dependent* then we can expect to detect benign distributions that yield more reasonable bound values indicating the possibility of successful learning.

Of course various refinements of the VC bound (Theorem 15) are conceivable. For example, using similar arguments as in the transition from Theorem 11 to Theorem 12 we can treat the case of non-zero training error, $\hat{R}[h, \boldsymbol{z}] \neq 0$. Of course the argument about counting permutations in the double-sample needs refinement, in this case. However, we will simply consider the resulting bound which can be found in similar form in Cristianini and Shawe-Taylor (2000):

Theorem 18 (VC generalisation error bound). *Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, for any hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ of finite VC dimension d, for any probability measure $\mathbf{P}_{\mathbf{Z}}$, for every sample size $m > 0$, and for all $\delta \in (0, 1]$ we have with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that the prediction error $R[h]$ of an hypothesis $h \in \mathcal{H}$ is bounded by*

$$R[h] < \varepsilon\left(m, d, \hat{R}[h, \boldsymbol{z}], \delta\right)$$

with

$$\varepsilon\left(m, d, \hat{R}\left[h, \boldsymbol{z}\right], \delta\right) = 2 \cdot \hat{R}\left[h, \boldsymbol{z}\right] + \frac{4}{m}\left(d \log_2 \frac{2em}{d} + \log_2 \frac{4}{\delta}\right),$$

provided that $d \leq m$.

Reconsider the motivation from the beginning of this section for considering uniform convergence: The goal was to bound the deviation of the risk of an ERM algorithm $\mathcal{A}_{\text{erm}}$ from the risk optimal hypothesis $h^* \in \mathcal{H}$. Now considering the bound from Theorem 18 we see that once $\mathcal{H}$ and hence $d = \text{VCdim}\left(\mathcal{H}\right)$ are fixed the only way of minimising the error bound is by finding the hypothesis h that minimises $\hat{R}\left[h, \boldsymbol{z}\right]$. The bound thus constitutes a theoretical motivation for the principle of empirical risk minimisation.

Finally, let us consider what has become known as *structural risk minimisation*. Assume that we are given a nested sequence of hypothesis spaces,

$$\mathcal{H}_1 \subset \cdots \subset \mathcal{H}_i \subset \cdots \subset \mathcal{H}_M$$

and find the empirical risk minimiser $h_i = \mathcal{A}_{\text{erm},i}\left(\boldsymbol{z}\right)$ in each $\mathcal{H}_i$. Then the resulting sequence of empirical risks will satisfy

$$\hat{R}\left[h_1, \boldsymbol{z}\right] \geq \cdots \geq \hat{R}\left[h_i, \boldsymbol{z}\right] \geq \cdots \geq \hat{R}\left[h_M, \boldsymbol{z}\right]$$

while the sequence of VC dimensions $d_i = \text{VCdim}\left(\mathcal{H}_i\right)$ will satisfy

$$d_1 \leq \cdots \leq d_i \leq \cdots \leq d_M\,.$$

By choosing the hypothesis space $\mathcal{H}_i$ for which the bound from Theorem 18 is minimised a trade-off between training error and capacity control can be achieved. However, there is a caveat w.r.t. the application of the bound from Theorem 18 to the hypothesis h that may result from such a procedure. The bound does not hold as it stands because it has really been applied M times to achieve the desired result. The phenomenon is akin to the problem of multiple testing a la Bonferroni in statistics (Holm 1979). In order to obtain a valid bound consider the following stratification lemma:

Lemma 1 (Stratification lemma). *Given* M *logical formulae* $\Upsilon_i : \mathcal{Z}^m \times (0,1] \to \{\text{true}, \text{false}\}$ *such that for all* $i \in \{1, \ldots, m\}$ *and all* $\delta \in (0,1]$ *we have*

$$\mathbf{P}_{\mathbf{Z}^m}\left(\Upsilon_i\left(\mathbf{Z}, \delta\right)\right) \geq 1 - \delta\,,$$

then for positive numbers $p_1, \ldots, p_M$ *with* $\sum_{i=1}^{M} p_i \leq 1$ *we have for all* $\delta \in (0,1]$

$$\mathbf{P}_{\mathbf{Z}^m}\left(\bigwedge_{i=1}^{M} \Upsilon_i\left(\mathbf{Z}, \delta p_i\right)\right) \geq 1 - \delta\,.$$

Proof. The proof is an application of the union bound, Theorem 57. For all $\delta \in (0,1]$ we have

$$\begin{aligned}
\mathbf{P}_{\mathbf{Z}^m}\left(\bigwedge_{i=1}^{M} \Upsilon_i\left(\mathbf{Z}, \delta \cdot p_i\right)\right) &= 1 - \mathbf{P}_{\mathbf{Z}^m}\left(\bigvee_{i=1}^{M} \neg\Upsilon_i\left(\mathbf{Z}, \delta p_i\right)\right) \\
&\geq 1 - \sum_{i=1}^{M} \mathbf{P}_{\mathbf{Z}^m}\left(\neg\Upsilon_i\left(\mathbf{Z}, \delta p_i\right)\right) \\
&> 1 - \sum_{i=1}^{M} \delta p_i \\
&= 1 - \delta\,.
\end{aligned}$$

□

Application of the stratification lemma with $p_i = \frac{1}{M}$ for all $i \in \{1, \ldots, m\}$ to the bound from Theorem 18 yields essentially the same bound with δ replaced by $\frac{\delta}{M}$ to ensure that the probability that any one of the bounds for the M hypothesis spaces $\mathcal{H}_i$ fails is at most δ.

3.1.3. PAC Margin Bounds

In Chapter 2 we encountered at several points the margin γ of a linear classifier. Not only does it determine the speed of convergence of the perceptron algorithm, but it also is the quantity maximised by the well-known support vector machine (SVM). The SVM owes its popularity by a fair amount to the fact that it has been presented as an algorithm that has a foundation in learning theory. In fact, the SVM has the reputation that it was constructed from learning theoretical principles. In this subsection we will consider a few bounds that learning theory provides in terms of the margin and that may serve as a theoretical foundation for algorithms that aim at maximising the margin. Since the margin is a quantity that is intimately related to thresholding real-valued functions let us introduce a generalised definition of the margin:

Definition 43 (Functional margin). Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, and a real-valued function class $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{X}}$ with an associated hypothesis space $\mathcal{H}_{\mathcal{F}} \subseteq \mathcal{Y}^{\mathcal{X}}$ given by $\mathcal{H}_{\mathcal{F}} = \{h_f(x) = \operatorname{sign}(f(x)) \mid f \in \mathcal{F}\}$, we define the (functional) margin of an example $z_i = (x_i, y_i) \in \mathcal{Z}$ w.r.t. a function $f \in \mathcal{F}$ as

$$\gamma(f, z_i) \stackrel{\text{def}}{=} y_i f(x_i)\,,$$

and the (functional) margin of a training sample $\boldsymbol{z}$

$$\gamma(f, \boldsymbol{z}) \stackrel{\text{def}}{=} \min_{z_i \in \boldsymbol{z}} \gamma(f, z_i)\,.$$

Also, we define the (functional) margin of a training sample $\boldsymbol{z}$ w.r.t. a function class $\mathcal{F}$ as

$$\gamma(\mathcal{F}, \boldsymbol{z}) \stackrel{\text{def}}{=} \max_{f \in \mathcal{F}} \gamma(f, \boldsymbol{z})\,.$$

In principle, in order to make assertions about the generalisation ability of a classifier h_f it is of interest to consider the whole margin distribution $\{\gamma(f, z_i) \,|\, i \in \{1, \ldots, m\}\}$. This program has in fact been carried out successfully (Shawe-Taylor and Cristianini 2000a; Shawe-Taylor and Cristianini 2000b; Shawe-Taylor and Cristianini 1998), in particular, with an emphasis on the soft-margin SVM as described in Chapter 2. We will focus here on bounds in terms of the margin $\gamma(f, \boldsymbol{z})$ on a training sample $\boldsymbol{z}$. Again our intention is to bound the probability on the right-hand side of Theorem 13 uniformly over the *effective* hypothesis space, i.e. as realised on a double-sample. Given a margin $\gamma(f, \boldsymbol{z})$ we aim at using the fact that the classification of any one of the $2m$ sample points does not change if the function value at that point is only approximated within a zone of tolerance of size $\frac{1}{2}\gamma(f, \boldsymbol{z})$. We thus aim at approximating $\mathcal{F}$ by a finite set of functions such that this condition holds.

Definition 44 (Cover and covering numbers). Given an input space $\mathcal{X}$ let $\mathcal{F}$ be a class of real-valued functions $f : \mathcal{X} \to \mathbb{R}$. A *$\gamma$-cover* of $\mathcal{F}$ w.r.t. an input sample $\boldsymbol{x}$ is a finite set $B \subset \mathcal{F}$ of functions such that for all $f \in \mathcal{F}$, there exists $g \in B$, with

$$\max_{i \in \{1,\ldots,m\}} |f(x_i) - g(x_i)| < \gamma .$$

The size of the smallest such cover is denoted by $\mathcal{N}(\mathcal{F}, \boldsymbol{x}, \gamma)$ and the *covering numbers* are given by

$$\mathcal{N}(\mathcal{F}, m, \gamma) = \max_{\boldsymbol{x} \in \mathcal{X}^m} \mathcal{N}(\mathcal{F}, \boldsymbol{x}, \gamma) .$$

Given this definition, our goal is to reformulate Theorem 15 for the case of classifiers based on thresholding real-valued functions from a function class $\mathcal{F}$ with the covering numbers of $\mathcal{F}$ playing the role of the VC dimension. To this end, we have another basic lemma (see Herbrich (2001) for a proof):

$$\begin{aligned} &\mathbf{P}_{\mathbf{Z}^m}\left(\exists f \in \mathcal{F} : \wedge\left(\gamma(f, \mathbf{Z}) > \gamma\right) \wedge \left(R[h_f] > \varepsilon\right)\right) \\ &\leq 2 \cdot \mathbf{P}_{\mathbf{Z}^{2m}}\left(\exists f \in \mathcal{F} : \left(\hat{R}[h_f, \mathbf{Z}_1] = 0\right) \wedge \left(\gamma(f, \mathbf{Z}_1) > \gamma\right) \wedge \left(\hat{R}[h_f, \mathbf{Z}_2] > \frac{\varepsilon}{2}\right)\right) \end{aligned} \tag{3.4}$$

where $\mathbf{Z}_1 = (\mathrm{Z}_1, \ldots, \mathrm{Z}_m)$ and $\mathbf{Z}_2 = (\mathrm{Z}_{m+1}, \ldots, \mathrm{Z}_{2m})$. In order to bound the right-hand side of the previous equation we consider a $\frac{\gamma}{2}$-cover B w.r.t. a double sample $\boldsymbol{z}_1\boldsymbol{z}_2$. For a function $g \in B$ within $\frac{\gamma}{2}$ of f we have on the first sample $\boldsymbol{z}_1$ that $\hat{R}[h_g, \boldsymbol{z}_1] = 0$ and $\gamma(g, \boldsymbol{z}_1) > \frac{\gamma}{2}$. On the second sample $\boldsymbol{z}_2$ we have for every $(x, y) \in \boldsymbol{z}_2$ with $h_f(x) \neq y$ that $\gamma(g, \boldsymbol{z}_2) < \frac{\gamma}{2}$. Denoting by

$$\hat{R}^{\frac{\gamma}{2}}[h_f, \boldsymbol{z}] \stackrel{\text{def}}{=} \frac{1}{m}\left|\left\{z \in \boldsymbol{z} \,\middle|\, \gamma(f, z) < \frac{\gamma}{2}\right\}\right|$$

the fraction of examples failing to achieve at least a margin of $\frac{\gamma}{2}$ we can bound the right-hand side of (3.4) by

$$\begin{aligned} &2 \cdot \mathbf{P}_{\mathbf{Z}^{2m}}\left(\exists f \in \mathcal{F} : \left(\hat{R}[h_f, \mathbf{Z}_1] = 0\right) \wedge \left(\gamma(f, \mathbf{Z}_1) > \gamma\right) \wedge \left(\hat{R}[h_f, \mathbf{Z}_2] > \frac{\varepsilon}{2}\right)\right) \\ &\leq 2\,\mathbf{P}_{\mathbf{Z}^{2m}}\left(\exists g \in B(\mathbf{Z}_1\mathbf{Z}_2) : \left(\hat{R}[h_g, \mathbf{Z}_1] = 0\right) \wedge \left(\gamma(g, \mathbf{Z}_1) > \frac{\gamma}{2}\right) \wedge \left(\hat{R}^{\frac{\gamma}{2}}[h_g, \mathbf{Z}_2] > \frac{\varepsilon}{2}\right)\right) \end{aligned}$$

Now we again use the symmetrisation argument (3.3) that leads to the counting of swapping permutations and apply the union bound, Theorem 57. This leads to

$$2 \cdot \mathbf{P}_{\mathbf{Z}_1^m \mathbf{Z}_2^m} \left(\exists g \in B(\mathbf{Z}_1, \mathbf{Z}_2) : \left(\hat{R}[h_g, \mathbf{Z}_1] = 0 \right) \wedge \left(\gamma(g, \boldsymbol{z}) > \frac{\gamma}{2} \right) \wedge \left(\hat{R}^{\frac{\gamma}{2}}[h_g, \mathbf{Z}_2] > \frac{\varepsilon}{2} \right) \right) \leq 2 \cdot \mathcal{N}\left(\mathcal{F}, 2m, \frac{\gamma}{2} \right) 2^{-\frac{\varepsilon m}{2}}$$

As a result we have the following theorem.

Theorem 19 (Covering number bound). *Consider an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, and a real-valued function class $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{X}}$ with an associated hypothesis space $\mathcal{H}_{\mathcal{F}} \subseteq \mathcal{Y}^{\mathcal{X}}$ given by $\mathcal{H}_{\mathcal{F}} = \{h_f(x) = \operatorname{sign}(f(x)) \mid f \in \mathcal{F}\}$. Given $\gamma > 0$ we have for any $\mathbf{P}_{\mathbf{Z}}$ with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z}$ of size m, for any hypothesis h_f with $f \in \mathcal{F}$ and $\gamma(f, \boldsymbol{z}) \geq \gamma$*

$$R[h_f] \leq \varepsilon(m, \mathcal{F}, \delta, \gamma) = \frac{2}{m} \left(\log_2 \mathcal{N}\left(\mathcal{F}, 2m, \frac{\gamma}{2} \right) + \log_2 \frac{2}{\delta} \right).$$

It is instructive to compare this result to Theorem 15. The expression $d \log \frac{2em}{d}$ involving the VC dimension d has been replaced by the logarithm $\log_2 \mathcal{N}\left(\mathcal{F}, 2m, \frac{\gamma}{2}\right)$ of the covering number which now serves as the capacity measure of the hypothesis space $\mathcal{H}_{\mathcal{F}}$. Again there is a caveat about the application of the bound: the value γ has to be fixed before learning in order to make the bound applicable. However, again this situation can be remedied using the stratification lemma, Lemma 1, to stratify over different values of γ thus making sure that the observed value of the margin lies close to one of the values taken into account by one of the bounds. Now, in order to make Theorem 19 applicable we need a bound on the covering number of real-valued function classes $\mathcal{F}$. To this end we use the notion of the *fat-shattering dimension,* which is a generalisation of the VC dimension for real-valued function classes $\mathcal{F}$.

Definition 45 (Fat-shattering dimension). Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, an input sample $\boldsymbol{x} \in \mathcal{X}^m$, and a real-valued function class $\mathcal{F}$ we say that $\boldsymbol{x}$ is *γ-shattered* by $\mathcal{F}$ if there exist real numbers r_i for all $i \in \{1, \ldots, m\}$, such that for every binary classification $\boldsymbol{y} \in \mathcal{Y}^m$ there exists a function $f_{\boldsymbol{y}} \in \mathcal{F}$, with

$$f_{\boldsymbol{y}}(x_i) = \begin{cases} \geq r_i + \gamma & \text{if} \quad y_i = +1 \\ < r_i - \gamma & \text{if} \quad y_i = -1 \end{cases}.$$

The *fat-shattering dimension* $\operatorname{fat}_{\mathcal{F}}(\gamma)$ at scale γ is the size of the largest γ-shattered subset of $\mathcal{X}$.

We now turn to our original objective of bounding the covering number in terms of the fat-shattering dimension (Alon et al. 1997).

Lemma 2 (Bound on covering numbers). *Consider an input space $\mathcal{X}$, let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to [a, b]$ and let $\mathbf{P}_{\mathbf{X}}$ be any distribution on $\mathcal{X}$. Given $\gamma > 0$, then with $d = \operatorname{fat}_{\mathcal{F}}\left(\frac{\gamma}{4}\right) \leq m$ we have*

$$\log_2 \mathcal{N}(\mathcal{F}, m, \gamma) \leq 1 + d \log_2 \frac{2em(b-a)}{d\gamma} \log_2 \frac{4m(b-a)^2}{\gamma^2}.$$

Ignoring the second log-factor we see that the bound on the log-covering number $\log_2 \mathcal{N}(\mathcal{F}, m, \gamma)$ in terms of the fat-shattering dimension $\text{fat}_{\mathcal{F}}\left(\frac{\gamma}{4}\right)$ of $\mathcal{F}$ is very similar to the dependence of the growth function $G_{\mathcal{H}}(m)$ on the VC dimension $\text{VCdim}(\mathcal{H})$ given in Theorem 14. The structural similarity of the resulting bounds is best seen if we combine Lemma 2 with Theorem 19, which gives the following theorem:

Theorem 20 (Fat-shattering dimension bound). *Consider an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, and a real-valued function class $\mathcal{F} \subseteq [-\varsigma, \varsigma]^{\mathcal{X}}$ with an associated hypothesis space $\mathcal{H}_{\mathcal{F}} \subseteq \mathcal{Y}^{\mathcal{X}}$ given by $\mathcal{H}_{\mathcal{F}} = \{h_f(x) = \text{sign}(f(x)) \mid f \in \mathcal{F}\}$. Given $\gamma \in (0, \varsigma)$ we have for any $\mathbf{P_Z}$ with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z}$ of size m, for any hypothesis h_f with $f \in \mathcal{F}$ and $\gamma(f, \boldsymbol{z}) \geq \gamma$*

$$R[h_f] \leq \varepsilon(m, d, \varsigma, \delta, \gamma) = \frac{2}{m}\left(d \log_2 \frac{16em\varsigma}{d\gamma} \log_2 \frac{128m\varsigma^2}{\gamma^2} + \log \frac{4}{\delta}\right),$$

provided that $d = \text{fat}_{\mathcal{F}}\left(\frac{\gamma}{8}\right) \leq m$.

Comparing this bound to Theorem 15 we see that the fat-shattering dimension at level $\frac{\gamma}{8}$ acts as an effective VC dimension. In fact, there exist cases when the VC dimension of $\mathcal{H}_{\mathcal{F}}$ is infinite implying the learning is not possible in general, while the fat-shattering dimension $\text{fat}_{\mathcal{F}}\left(\frac{\gamma}{8}\right)$ of $\mathcal{F}$ at a given level $\frac{\gamma}{8}$ is finite. However, there is no contradiction to VC theory because the resulting bound depends via the fat-shattering dimension on the observed margin. While the bound given by Theorem 20 is valid for all distributions $\mathbf{P_Z}$ it will yield non-trivial results only in the case of a large observed margin $\gamma(f, \boldsymbol{z})$ indicating a benign data distribution.

Our original goal was to provide a bound on the generalisation error for linear classifiers in terms of the observed margin. To this end let us consider a bound on the fat-shattering dimension for linear function classes $\mathcal{F}$ such as those defined in Definition 19. A proof of the following result can be found in Bartlett and Shawe-Taylor (1998).

Theorem 21 (Bound on fat-shattering dimension of $\mathcal{H}_{\mathcal{K}}$). *Suppose that $\mathcal{K} \subset \ell_2^n$ is a ball $\mathcal{K} \stackrel{\text{def}}{=} \left\{\mathbf{x} \in \ell_2^n \,\middle|\, \|\mathbf{x}\|^2 \leq \varsigma^2\right\}$ of radius ς, and consider the class of functions*

$$\mathcal{F}_{\mathcal{K}} \stackrel{\text{def}}{=} \left\{f_{\mathbf{w}}(\mathbf{x}) \mapsto \langle \mathbf{x}, \mathbf{w} \rangle \,\middle|\, \mathbf{x} \in \mathcal{K} \wedge \|\mathbf{w}\|^2 \leq 1\right\},$$

then the fat-shattering dimension of $\mathcal{F}$ is bounded by

$$\text{fat}_{\mathcal{F}}(\gamma) \leq \left(\frac{\varsigma}{\gamma}\right)^2.$$

It should be noted that the bound on the fat-shattering dimension for linear classifiers, Theorem 21, is the same as the bound on the number M of mistakes of the perceptron algorithm, Theorem 2, for $\rho^2 = 0$. Later in Chapter 6 we will explore this interesting relation further.

In summary, we have reformulated the VC bound, Theorem 15, in terms of covering numbers leading to Theorem 19. We have bounded the covering numbers in terms of the

fat-shattering dimension in Lemma 2 leading to Theorem 20, and we have bounded the fat-shattering dimension for linear functions classes in terms of the margin in Theorem 21. These steps finally lead to a bound that is applicable to linear margin classifiers with weight vectors $\mathbf{w}$ of bounded length $\|\mathbf{w}\| \leq 1$ and data of radius ς in $\mathcal{K}$. As a consequence, the functions $f_{\mathbf{w}} \in \mathcal{F}_{\mathcal{K}}$ are bounded on $\mathcal{K}$, that is we have $|f_{\mathbf{w}}(\mathbf{x})| = |\langle \mathbf{w}, \mathbf{x} \rangle| \leq \|\mathbf{w}\| \cdot \|\mathbf{x}\| \leq \varsigma$ for all $\mathbf{x} \in \mathcal{K}$.

Theorem 22 (Margin bound for linear classifiers). *Consider an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, and a real-valued function class $\mathcal{F} \subseteq [-\varsigma, \varsigma]^{\mathcal{X}}$ with an associated hypothesis space $\mathcal{H}$ given by $\mathcal{H} = \left\{ h_{\mathbf{w}}(x) = \operatorname{sign}(\langle \mathbf{x}, \mathbf{w} \rangle) \,\middle|\, \mathbf{x} \in \mathcal{K} \wedge \|\mathbf{w}\|^2 = 1 \right\}$. Given a fixed value $\gamma \in (0, \varsigma)$ we have for any $\mathbf{P}_{\mathbf{Z}}$ with support in a ball of radius ς with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z}$ of size m, for any hypothesis $h_{\mathbf{w}} \in \mathcal{H}$ with $\gamma(f_{\mathbf{w}}, \boldsymbol{z}) \geq \gamma$ that*

$$R[h_{\mathbf{w}}] \leq \varepsilon(m, \varsigma, \delta, \gamma) = \frac{2}{m} \left(\frac{64\varsigma^2}{\gamma^2} \log_2 \frac{em\gamma}{4\varsigma} \log_2 \frac{128m\varsigma^2}{\gamma^2} + \log \frac{4}{\delta} \right),$$

provided that $\frac{64\varsigma^2}{\gamma^2} < m$.

This result is a corrected version of a result to be found in Cristianini and Shawe-Taylor (2000). More elaborate results that take into account the observed margin by stratification can be found in Bartlett and Shawe-Taylor (1998) and Shawe-Taylor et al. (1998). These results can be considered the main theoretical motivation of large margin classifiers such as the support vector machine. Its main qualitative feature is the independence of the bound of the dimensionality of the feature space $\mathcal{K}$. This means that learning may take place in high-dimensional feature spaces as long as the distribution of the data is benign in the sense that the margin the classifier achieves on the data is large. The argument presented can be extended to include not only the minimal margin but the whole margin distribution (Shawe-Taylor and Cristianini 2000a; Shawe-Taylor and Cristianini 2000b; Shawe-Taylor and Cristianini 1998). Another extension incorporates the distance of a given test input from the hyperplane into a bound on the error probability for the classification of that point (Shawe-Taylor 1996). We will continue the discussion of margin bounds in Chapter 4 where completely different techniques based on the PAC-Bayesian framework will be employed to prove margin bounds for linear classifiers.

In the margin bound, Theorem 22, the margin a classifier achieves on the training sample serves as an indicator for a benign distribution. Another property of certain classifiers that can be observed after learning is the achieved sparseness of the classifier. Again the SVM may serve as an example: The solution of the SVM is usually sparse in the sense that $|\mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)| \ll m$, i.e. only a small subset of the training data is used to determine the resulting classifier. The reduction in required sample size is referred to as a *compression scheme* (Littlestone and Warmuth 1986; Floyd and Warmuth 1995) and will be treated in more detail in Chapter 5. It turns out that in practice the compression bounds yields excellent results, i.e. very low bound-values—as compared to the simple VC bound, Theorem 15, and the margin bound, Theorem 22—when applied to typical

support vector machine solutions. Interestingly, there also exist some fundamental relations between the notions of margin and sparseness that will be explored further in Chapter 6.

The observation that there exist two quantities that may indicate how benign an input distribution is—margin and sparsity—has prompted the development of the so-called *luckiness framework* (Shawe-Taylor et al. 1998) that gives sufficient conditions for the construction of bounds on the prediction error of classifiers based on luckiness functions that describe the alignment of the hypothesis space and the data distribution. Conceptually, the development of new luckiness functions may then prompt the development of new learning algorithms. In turn, the success of a particular learning algorithm may be explained in terms of a luckiness function that is optimised by the algorithm. In a sense, the luckiness framework can be considered a complicated way of defining a prior over hypotheses not unlike in the Bayesian framework that will be considered in the subsequent section.

3.2. The Bayesian Framework

3.2.1. Reasoning under Uncertainty: From Prior to Posterior

The previous learning theoretical considerations all revolved around the idea of empirical risk minimisation. We assumed that learning was based on a random sample $\boldsymbol{z}$ drawn iid from some probability distribution $\mathbf{P}_Z$. Learning was described as selecting an hypothesis h from some fixed hypothesis space $\mathcal{H}$ based on the empirical risk $\hat{R}[h, \boldsymbol{z}]$. This probabilistic framework is adequate because the randomness of the learning process is due to the random distribution of the data. The Bayesian approach to learning takes a different viewpoint. The notion of a probability distribution is extended to include not only random phenomena in the sense of the random data generation from $\mathbf{P}_Z$, but to include also the quantification of *uncertainty* or *belief* about model parameters or hypotheses about the data. This use of probability has been and is subject to a sometimes heated debate in the statistical community. Various arguments have been put forward to support the idea of representing *beliefs* in terms of probabilities (Gelman et al. 1995).

1. Physical randomness induces uncertainty. By analogy people describe uncertainty in the language of random events as is evident from the usage of words such as "probably" and "unlikely" in common speech. However, a mere analogy can hardly be a solid foundation for a method of scientific inference.

2. Decision theory formulates certain axioms of rational behaviour in the context of decision making. These axioms (see, e.g. Cox (1946)) imply that uncertainty must be represented by probabilities obeying the Kolmogorov axioms. Of course, different decision-theoretic axioms lead to different measures of uncertainty. An example of an approach different from probability is fuzzy logic (Zadeh 1992).

3. The argument about coherence of bets (Gelman et al. 1995) relates subjective probabilities of events to the betting odds someone is willing to accept w.r.t. the

occurrence of those events. If someone offers you a bet of a against the occurrence of event Υ, then your subjective probability of Υ or your belief in the occurrence of Υ can be defined as the fraction $\mathbf{P}(\Upsilon) \in [0,1]$ of a you are willing to bet in return to a. If Υ occurs you win $(1 - \mathbf{P}(\Upsilon)) \cdot a$ and if Υ does not occur you loose $\mathbf{P}(\Upsilon) \cdot a$. Unfortunately, exact odds must be determined, which might be difficult to do in situations of uncertainty, where it might be advisable not to bet at all.

While these arguments may be considered more or less convincing it is certainly the practical application of Bayesian methods that can convince people of their value. We will now proceed by considering the learning problem from a Bayesian perspective.

For simplicity consider a simple PAC learning problem with a teacher hypothesis $h^* \in \mathcal{H}$ and hence with $R[h^*] = 0$. Given the problem there is certainly no randomness in the hypothesis h^* which is fixed. Nonetheless we are uncertain about h^* and the Bayesian approach suggests that we quantify this uncertainty about h^* in terms of a probability distribution $\mathbf{P}_{\mathsf{H}}$ over $\mathcal{H}$, the so-called *prior distribution.* In order to make probabilistic statements about hypotheses h in relation to the data sample $\boldsymbol{z}$ we introduce a joint probability distribution $\mathbf{P}_{\mathsf{Z}^m\mathsf{H}}(\boldsymbol{z}, h)$ of data and hypothesis given by

$$\mathbf{P}_{\mathsf{Z}^m\mathsf{H}}(\boldsymbol{z}, h) = \mathbf{P}_{\mathsf{Z}^m|\mathsf{H}=h}(\boldsymbol{z})\, \mathbf{P}_{\mathsf{H}}(h) \ .$$

Conditioning on the data using the definition of conditional probability we obtain what is known as *Bayes´ rule* (Bayes 1763),

$$\underbrace{\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}(h)}_{\text{Posterior over } \mathcal{H}} = \frac{\overbrace{\mathbf{P}_{\mathsf{Z}^m|\mathsf{H}=h}(\boldsymbol{z})}^{\text{Likelihood of } h}\ \overbrace{\mathbf{P}_{\mathsf{H}}(h)}^{\text{Prior over } \mathcal{H}}}{\underbrace{\mathbf{E}_{\mathsf{H}}\left[\mathbf{P}_{\mathsf{Z}^m|\mathsf{H}=h}(\boldsymbol{z})\right]}_{\text{Evidence of } \mathcal{H}}} \ . \tag{3.5}$$

Bayes´ rule clearly reveals statistical inference as an *inversion* process in the sense that it aims at retrieving "causes"—parameters of the probabilistic generating process—from the "effects"—observations (Robert 1994). The result of the inversion is a *posterior distribution* over hypotheses that can be looked upon as an *update of the prior* in the light of the observations $\boldsymbol{z}$. As can be seen from Bayes´ rule (3.5) the result of Bayesian inference is based on two ingredients that must be specified in advance: *likelihood* and *prior.*

Definition 46 (Likelihood). Given a family $\mathcal{P}$ of probability distributions $\mathbf{P}_{\mathsf{Z}|\mathsf{H}=h}$ over $\mathcal{Z}$ parameterised by $h \in \mathcal{H}$ and an observation $z \in \mathcal{Z}$ we define the *likelihood* of h as any function $\mathcal{L} : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$ given by

$$\mathcal{L}[h; z] \stackrel{\text{def}}{=} c(z) \cdot \mathbf{P}_{\mathsf{Z}|\mathsf{H}=h}(z) \ ,$$

where c is an arbitrary positive real-valued function of z not explicitly depending on h.

Bayesian inference adheres to the so-called *likelihood principle.* It is only via the likelihood $\mathcal{L}[h; z]$ that the data enter the process of inference, or in other words (Robert 1994):

Likelihood Principle The information brought by an observation z about h is entirely contained in the likelihood function $\mathcal{L}[h; z]$.

Let us now consider the problem of supervised learning under the PAC assumption $h^* \in \mathcal{H}$ and hence with $R[h^*] = 0$ in more detail. First we observe that in the supervised learning scenario the data distribution $\mathbf{P}_{\mathsf{Z}^m}$ can be written as

$$\mathbf{P}_{\mathsf{Z}^m}(\boldsymbol{z}) = \mathbf{P}_{\mathsf{Y}^m|\mathsf{X}^m=\boldsymbol{x}}(\boldsymbol{y}) \cdot \mathbf{P}_{\mathsf{X}^m}(\boldsymbol{x}) . \tag{3.6}$$

As a consequence, Bayes´ rule for the supervised learning task reads

$$\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}(h) = \frac{\mathbf{P}_{\mathsf{Y}^m|\mathsf{X}^m=\boldsymbol{x},\mathsf{H}=h}(\boldsymbol{y}) \cdot \mathbf{P}_{\mathsf{H}}(h)}{\mathbf{E}_{\mathsf{H}}\left[\mathbf{P}_{\mathsf{Y}^m|\mathsf{X}^m=\boldsymbol{x},\mathsf{H}=h}(\boldsymbol{y})\right]} . \tag{3.7}$$

Assuming the existence of an hypothesis $h^* \in \mathcal{H}$ that actually labelled the data we can define the likelihood corresponding to the PAC model of learning.

Definition 47 (PAC likelihood). Given an input space $\mathcal{X}$, an output space $\mathcal{Y}$, and an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, the likelihood function

$$\mathcal{L}_{\text{pac}}(h; (x, y)) \stackrel{\text{def}}{=} \mathbf{I}_{h(x)=y} = 1 - l_{0-1}(h(x), y)$$

is called the *PAC likelihood.*

Clearly, this definition captures the idea of PAC learning with a teacher in the sense that any hypothesis h that misclassifies the example (x, y) is ruled out while the remaining hypotheses have the same likelihood value.

The Bayesian framework is often criticised for the necessity of providing an adequate prior distribution $\mathbf{P}_{\mathsf{H}}$ over hypotheses. There are two main problems frequently raised against the Bayesian framework in general and the Bayesian prior in particular:

1. If prior belief is present, how can it be quantified such that the prior correctly represents the belief and leads to adequate inference?

2. If no prior belief is present, how can a non-informative prior be formulated such that Bayesian inference is still feasible?

It is certainly a rather academic endeavour to criticise the Bayesian framework in general for the difficulties that may arise in the formulation and specification of the prior. In specific cases, however the choice of prior can in fact be rather difficult and conceptually worrying. An example is the necessity to use improper prior densities that cannot be normalised, e.g., when estimating a location parameter $\mu \in \mathbb{R}$ (Robert 1994). However, we will assume for the moment a uniform prior over $\mathcal{H}$ that reflects our ignorance about the true hypothesis h^*. For hypothesis spaces $\mathcal{H}$ of finite cardinality $|\mathcal{H}|$ this leads to the simple prior for all h

$$\mathbf{P}_{\mathsf{H}}(h) = \frac{1}{|\mathcal{H}|} .$$

In the case of non-finite cardinality $|\mathcal{H}|$ we will later define a density $\mathbf{f}_{\mathsf{H}}$ over parameter space.

3.2.2. From Posterior to Decision

While we have discussed so far the probabilistic inference process leading to a posterior distribution $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ the Bayesian framework is really a decision-theoretic tool. To this end, we must provide a decision criterion in the form of a loss function. Suppose we aim at identifying a single hypothesis $\hat{h} \in \mathcal{H}$ in an attempt to draw an inference on the true hypothesis h^* in the sense of a statistical point estimation. Then in order to make the decision which hypothesis $\hat{h}$ to choose we define a loss function $l^{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to [0, \infty)$ that captures the losses resulting from deciding for hypothesis $\hat{h} \in \mathcal{H}$ while the true hypothesis is h^*. While a given decision situation may be best described by very specific loss functions we consider here rather generic loss functions $l^{\mathcal{H}}$. Given posterior and loss we need a guiding principle in order to make our decision. While various decision-theoretic principles such as the min-max-principle have been suggested (Berger 1985; Robert 1994) we focus here on a criterion known as Laplace´s principle.

Definition 48 (Laplace´s principle). Given a posterior distribution $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ and a loss function $l^{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to [0, \infty)$ choose the hypothesis h that minimises the expected risk $\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}} \left[l^{\mathcal{H}} (\mathsf{H}, h) \right]$,

$$\hat{h} \stackrel{\text{def}}{=} \operatorname*{argmin}_{h \in \mathcal{H}} \mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}} \left[l^{\mathcal{H}} (\mathsf{H}, h) \right] .$$

The hypothesis $\hat{h}$ is called the *Bayes hypothesis* or *Bayes estimator* w.r.t. $l^{\mathcal{H}}$.

As a simple illustration of the consequences of Laplace's principle consider the following loss function.

Definition 49 (Bayesian zero-one loss). Given an hypothesis space $\mathcal{H}$ we define the Bayesian zero-one loss $l^{\mathcal{H}}_{0-1}$ as

$$l^{\mathcal{H}}_{0-1} \left(h, \hat{h} \right) = \mathsf{I}_{h \neq \hat{h}} = \begin{cases} 0 & \text{if} \quad h = \hat{h} \\ 1 & \text{if} \quad h \neq \hat{h} \end{cases} .$$

Following Laplace´s principle we have the following proposition about the Bayes estimator w.r.t. the Bayesian zero-one loss.

Theorem 23 (Maximum-aposteriori estimator). *Given a posterior* $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ *and the zero-one loss function* $l^{\mathcal{H}}_{0-1} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}^+$ *the Bayesian estimator is given by*

$$\hat{h}_{\boldsymbol{z}} = \operatorname*{argmax}_{h \in \mathcal{H}} \mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}} (h) .$$

Proof. Minimising the Bayes risk w.r.t. $l^{\mathcal{H}}_{0-1}$ gives

$$\begin{aligned}
\hat{h}_{\boldsymbol{z}} &= \operatorname*{argmin}_{h \in \mathcal{H}} \mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}} \left[l_{0-1}^{\mathcal{H}} (\mathsf{H}, h) \right] \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}} \left[1 - \mathbf{I}_{\mathsf{H}=h} \right] \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \left(-\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}} (h) \right) \\
&= \operatorname*{argmax}_{h \in \mathcal{H}} \mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}} (h) \; .
\end{aligned}$$

□

Intuitively, this result says that if we can profit only from an exact match of our learned hypothesis $\hat{h}$ with the true hypothesis h^*, then the best choice we can make is to choose the hypothesis h with the highest aposteriori probability. Note that this choice may not be unique. If we choose a uniform prior over hypothesis space and use the PAC-likelihood as defined in Definition 47 then all the hypotheses $h \in V(\boldsymbol{z})$ contained in version space are maximum-aposteriori hypotheses.

More generally, given a metric or pseudo metric $d : \mathcal{H} \times \mathcal{H} \to [0, \infty)$on the hypothesis space, we can often define a suitable loss function based on the metric. As an example, consider the estimation of a location parameter $h \in \mathbb{R}$. Based on the metric $d : \mathbb{R} \times \mathbb{R} \to [0, \infty)$ given by $d(x_1, x_2) = |x_1 - x_2|$ we can define the loss $l_2^{\mathbb{R}} : \mathbb{R} \times \mathbb{R} \to [0, \infty)$ by $l_2^{\mathbb{R}}(h, \hat{h}) \stackrel{\text{def}}{=} d^2(h, \hat{h}) = (h - \hat{h})^2$. This choice of loss then leads to the posterior expectation $\hat{h} = \mathbf{E}_{\mathsf{H}|\mathsf{Z}^m} [\mathsf{H}]$ as the risk minimal hypothesis. In fact, we will pursue these ideas further in the context of the Bayes point machine in Section 7.2.

3.2.3. Bayesian Transduction and the Predictive Distribution

In machine learning the performance of learning machines is often measured in terms of their prediction error. While the notion of inference refers to statements about non-observable entities such as an hypothesis h, *prediction* aims at making statements about future observations. If we are interested in the prediction of a new example z, then instead of a posterior $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ together with a loss $l^{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}^+$ over hypotheses we might want to consider a posterior or predictive distribution $\mathbf{P}_{\mathsf{Z}|\mathsf{Z}^m=\boldsymbol{z}}$. Again, the starting point is the joint distribution

$$\mathbf{P}_{\mathsf{ZH}} (z, h) = \mathbf{P}_{\mathsf{Z}|\mathsf{H}=h} (z) \, \mathbf{P}_{\mathsf{H}} (h)$$

of data and hypotheses. Integrating out the hypothesis we have for the *marginal* or *prior predictive distribution* of an example z,

$$\mathbf{P}_{\mathsf{Z}} (z) = \mathbf{E}_{\mathsf{H}} \left[\mathbf{P}_{\mathsf{Z}|\mathsf{H}=h} (z) \right] \; .$$

Notice, how the prior $\mathbf{P}_{\mathsf{H}}$ over the hypothesis space $\mathcal{H}$ together with the likelihood $\mathbf{P}_{\mathsf{Z}|\mathsf{H}=h} (z) = \mathcal{L}(h; z)$ implicitly defines an apriori data distribution that does not depend on any particular hypothesis h. After considering a training sample $\boldsymbol{z} \in \mathcal{Z}^m$ we obtain for the *posterior predictive distribution*

$$\mathbf{P}_{\mathsf{Z}|\mathsf{Z}^m=\boldsymbol{z}}(z) = \mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left[\mathbf{P}_{\mathsf{Z}|\mathsf{H}=h}(z)\right] . \tag{3.8}$$

In this case the process of learning is the transition or update

$$\mathbf{P}_{\mathsf{Z}} \mapsto \mathbf{P}_{\mathsf{Z}|\mathsf{Z}^m=\boldsymbol{z}}$$

of the predictive distribution. This procedure constitutes the natural Bayesian mode of prediction and has recently been rediscovered in machine learning under the label *transduction* (Gammerman et al. 1998; Graepel et al. 2000). Let us again turn to the case of supervised learning when we are interested in predicting the output y for a given input x. Then the prior predictive distribution is given by

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y) = \mathbf{E}_{\mathsf{H}}\left[\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}(y)\right] ,$$

and the posterior predictive distribution is

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{Z}^m=\boldsymbol{z}}(y) = \mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left[\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}(y)\right] . \tag{3.9}$$

Together with a loss $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ the posterior predictive distribution suggests the following classification strategy.

Definition 50 (Bayes classification strategy). Given an input space $\mathcal{X}$, an output space $\mathcal{Y}$, an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, a training sample $\boldsymbol{z}$, a posterior probability distribution $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$, and a loss function $l^{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$the Bayes classification strategy is defined as

$$\text{Bayes}(x; \boldsymbol{z}) \stackrel{\text{def}}{=} \operatorname*{argmin}_{y \in \mathcal{Y}} \mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left[l^{\mathcal{Y}}(y, \mathsf{H}(x))\right] .$$

We call $\text{Bayes}(x; \boldsymbol{z})$ a classification strategy because while it is based on an hypothesis space $\mathcal{H}$ it does not select a single classifier $h \in \mathcal{H}$ but classifies an example x on the basis of the whole hypothesis space $\mathcal{H}$. Now consider the special case of the PAC likelihood from Definition 47. Then the Bayes classification strategy is given by

$$\text{Bayes}(x; \boldsymbol{z}) = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}(\mathsf{H}(x) = y) .$$

This classification strategy can be interpreted as a voting scheme among classifiers. The prior assigns a certain apriori weight to the classifiers. Then the PAC likelihood rules out those classifiers that are not consistent with the training sample. The remaining classifiers vote on the label of the test input x with (normalised) weight proportional to their original prior probability.

While the Bayes classification strategy is deterministic it is quite natural in a probabilistic context to consider non-deterministic classification strategies as well. In particular, the *Gibbs classification strategy* will be of major importance in the context of PAC-Bayesian theory.

Definition 51 (Gibbs classification strategy). Given an input space $\mathcal{X}$, an output space $\mathcal{Y}$, an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, a training sample $\boldsymbol{z}$, and a posterior probability distribution $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ the Gibbs classification strategy is defined as

$$\text{Gibbs}\,(x;\boldsymbol{z}) \stackrel{\text{def}}{=} h\,(x)\,, \quad h \sim \mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\ .$$

The Gibbs classification strategy samples hypotheses h from the posterior $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ and uses them to classify examples x by $h\,(x)$. Although this strategy is hardly ever used in practice it is the most direct application of the posterior to the classification of examples. Both the voting strategy of Bayes and the sampling strategy of Gibbs are robust classification schemes in the sense that they take into account outputs of many classifiers in an ensemble. This plurality of opinions safeguards these classification schemes against over-fitting as will be seen in the subsequent section.

The Bayesian framework provides strategies such as the Bayes and Gibbs classification strategies from Definitions 50 and 51, respectively. These strategies are in a sense optimal under the assumptions made. As can be seen from (3.8) in the predictive setting, the Bayesian framework uses a data model of a specific form by specifying hypothesis space, prior, and likelihood. In contrast to the PAC/VC framework the Bayesian framework does not provide any distribution-free guarantees on the performance of its resulting classification strategies. In the PAC-Bayesian framework an attempt is made to remedy these deficiencies of the Bayesian framework.

3.3. The PAC-Bayesian Framework

In the previous two sections we discussed the PAC/VC model of learning and the Bayesian model of learning. The PAC/VC model is essentially a worst case model whose results hold with high probability even in the worst case scenario of very difficult to learn data distributions. The resulting learning strategies, i.e., empirical risk minimisation and structural risk minimisation are conservative in the sense that they aim at results with performance guarantees even under worst case conditions. In contrast, the Bayesian learning scenario takes less conservative approach by making explicit assumptions on the data distribution that are encoded in prior, likelihood, and loss functions. It was shown in empirical studies (Kearns et al. 1997) that the more conservative strategies of structural risk minimisation and regularisation are less prone to *over-fitting* than Bayesian methods at the cost of displaying a certain degree of *under-fitting*. Not surprisingly, however, in the case that the Bayesian assumptions held Bayesian methods showed superior performance. The PAC-Bayesian framework aims at providing performance guarantees of the PAC style for Bayesian algorithms. The first PAC-Bayesian ideas emerged in Shawe-Taylor and Williamson (1997). The ideas to be outlined here are mainly due to McAllester (1998) and McAllester (1999).

3.3.1. Gibbs Classification Strategy: Realisable Case

The basic PAC-Bayesian framework (McAllester 1998) is based on a learning scenario that can be viewed as a combination of the PAC and the Bayesian framework. Beside the

usual ingredients of a supervised learning task, we assume a prior distribution $\mathbf{P}_{\mathsf{H}}$ over the hypothesis space $\mathcal{H}$. Based on this scenario, we will present two pairs of theorems, one pair for the case of $\hat{R}[h, \boldsymbol{z}] = 0$ and one pair for the agnostic case of $\hat{R}[h, \boldsymbol{z}] \neq 0$. Each time we will consider a preliminary theorem that bounds the risk of a single classifier and then a main theorem that bounds the risk of the Gibbs classification strategy over a subset of $\mathcal{H}$ (i.e. the risk averaged over $\mathbf{P}_{\mathsf{H}}$). The first preliminary theorem is essentially a generalisation of the simple PAC bound for zero empirical risk and an hypothesis space of finite cardinality, Theorem 11.

Theorem 24 (PAC-Bayesian bound for single consistent classifiers). *For any hypothesis space $\mathcal{H}$ of finite cardinality $|\mathcal{H}|$, for any prior probability distribution $\mathbf{P}_{\mathsf{H}}$ satisfying $\mathbf{P}_{\mathsf{H}}(h) > 0$ for all $h \in \mathcal{H}$, for any probability measure $\mathbf{P}_{\mathsf{Z}}$, for every sample size $m > 0$, and for all $\delta \in (0, 1]$ we have with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that for any consistent hypothesis $h \in \mathcal{H}$ the prediction error $R[h]$ is bounded by*

$$R[h] \leq \varepsilon(m, \mathbf{P}_{\mathsf{H}}, \delta) = \frac{1}{m}\left(\log \frac{1}{\mathbf{P}_{\mathsf{H}}(h)} + \log \frac{1}{\delta}\right) .$$

Proof. The proof is equivalent to the proof of Theorem 11 with the simple union bound, Theorem 57, replaced by the stratification lemma, Lemma 1, effectively stratifying over all hypotheses $h \in \mathcal{H}$. We have from the binomial tail bound, Theorem 58, that for a given h with $R[h] > \varepsilon$

$$\mathbf{P}_{\mathsf{Z}^m}\left(\hat{R}[h, \mathbf{Z}] = 0\right) \leq \exp(-\varepsilon m) .$$

Now we apply the stratification lemma, Lemma 1, with

$$\Upsilon_h(\mathbf{Z}, \delta) \equiv \left(\hat{R}[h, \boldsymbol{z}] = 0 \ \wedge \ R[h] \leq \frac{1}{m}\log \frac{1}{\delta}\right),$$

and $p_h \equiv \mathbf{P}_{\mathsf{H}}(h)$ satisfying $\sum_{h\in\mathcal{H}} p_h = \sum_{h\in\mathcal{H}} \mathbf{P}_{\mathsf{H}}(h) = 1$, which gives

$$\mathbf{P}_{\mathsf{Z}^m}\left(\forall h \in \mathcal{H} : \hat{R}[h, \boldsymbol{z}] = 0 \ \wedge \ R[h] \leq \frac{1}{m}\log \frac{1}{\delta \cdot \mathbf{P}_{\mathsf{H}}(h)}\right) \geq 1 - \delta .$$

This concludes the proof. □

This theorem serves to illustrate the role of the prior $\mathbf{P}_{\mathsf{H}}$ in PAC-Bayesian theory. Clearly, Theorem 24 holds for all priors that fulfil the rather technical condition $\mathbf{P}_{\mathsf{H}}(h) > 0$ for all $h \in \mathcal{H}$. Thus, in contrast to any Bayesian statement whose truth (or at least precision) rests on the truth of the assumed prior, a PAC-Bayesian bound is valid independent of the validity of the prior. Although this may sound like magic it is really only the distribution of the remaining error probability δ among every single single hypothesis $h \in \mathcal{H}$. Consider two special cases of Theorem 24:

1. A uniform prior $\mathbf{P}_{\mathsf{H}}(h) = \frac{1}{|\mathcal{H}|}$ for all $h \in \mathcal{H}$. Then Theorem 24 reverts back to the simple PAC bound, Theorem 11.

2. A prior $\mathbf{P}_{\mathsf{H}}$ that is strongly peaked at a particular hypothesis $\tilde{h}$, i.e. $\mathbf{P}_{\mathsf{H}}(\tilde{h}) = 1 - \Delta$ and $\mathbf{P}_{\mathsf{H}}(h) = \frac{\Delta}{|\mathcal{H}|-1}$ for all $h \neq \tilde{h}$, where Δ is a small positive value. In this case if we find that $\hat{R}\left[\tilde{h}, \boldsymbol{z}\right] = 0$ we can make the rather strong statement that with high probability $R[\tilde{h}] < \frac{1}{m}\left(\log\frac{1}{1-\Delta} + \log\frac{1}{\delta}\right) \approx \frac{1}{m}\log\frac{1}{\delta}$ which is almost the same result as would have been obtained in the case that $\tilde{h}$ is the only hypothesis, $\mathcal{H} = \{\tilde{h}\}$. Of course, for any hypothesis $h \neq \tilde{h}$ with $\hat{R}[h, \boldsymbol{z}] = 0$ we obtain with high probability that $R[h] < \frac{1}{m}\log\frac{|\mathcal{H}|-1}{\Delta} + \log\frac{1}{\delta}$, which would yield the trivial result $R[h] \leq 1$ unless $m \geq \log\frac{|\mathcal{H}|-1}{\Delta\delta}$, requiring a huge training sample.

Let us now consider the main PAC-Bayesian theorem for the case of consistent classifiers. To this end recall that the version space $V(\boldsymbol{z})$ is the set of all classifiers $h \in \mathcal{H}$ that are consistent with the training sample $\boldsymbol{z}$. We will also consider expectations $\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}$ over subsets $U \subseteq \mathcal{H}$, where the notation $\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}$ is to be read as

$$\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[R\left[\mathsf{H}\right]\right] \stackrel{\text{def}}{=} \frac{\mathbf{E}_{\mathsf{H}}\left[\mathbf{I}_{\mathsf{H}\in U} \cdot R\left[\mathsf{H}\right]\right]}{\mathbf{P}_{\mathsf{H}}(U)}\,.$$

Theorem 25 (PAC-Bayesian subset bound for consistent classifiers). *For any hypothesis space $\mathcal{H}$, for any prior probability distribution $\mathbf{P}_{\mathsf{H}}$ over $\mathcal{H}$, for any probability measure $\mathbf{P}_{\mathsf{Z}}$, for every sample size $m > 0$, and for all $\delta \in (0,1]$ we have with probability at least $1-\delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that for any subset $U \subseteq V(\boldsymbol{z})$ of version space the average prediction error $\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[R\left[\mathsf{H}\right]\right]$ is bounded by*

$$\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[R\left[\mathsf{H}\right]\right] \leq \varepsilon_{\text{Gibbs}}\left(m, \mathbf{P}_{\mathsf{H}}, U, \delta\right) = \frac{1}{m}\left(\log\frac{1}{\mathbf{P}_{\mathsf{H}}(U)} + \log\frac{1}{\delta} + 2\log m + 1\right).$$

Let us briefly consider the quantity $\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[R\left[\mathsf{H}\right]\right]$ bounded here, i.e., the prediction error $R[h]$ averaged over the prior probability $\mathbf{P}_{\mathsf{H}}$ restricted to a subset U of version space. The corresponding classification strategy is the Gibbs classification strategy $\text{Gibbs}(x, \boldsymbol{z})$ considered in the previous section. In particular, the bound of Theorem 25 is minimised for $U = V(\boldsymbol{z})$. In this case, the quantity $\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[R\left[\mathsf{H}\right]\right]$ bounded is the expectation of the prediction error $R[h]$ w.r.t. the posterior probability distribution $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ over hypotheses, given an arbitrary prior $\mathbf{P}_{\mathsf{H}}$ and the PAC likelihood $\mathcal{L}_{\text{pac}}(h;(x,y)) \stackrel{\text{def}}{=} \mathbf{I}_{h(x)=y}$.

In order to prove Theorem 25 we apply a quantifier reversal lemma:

Lemma 3 (Quantifier reversal lemma). *Let X and Y be random variables with associated probability spaces $(\mathcal{X}, \mathfrak{X}, \mathbf{P}_{\mathsf{X}})$ and $(\mathcal{Y}, \mathfrak{Y}, \mathbf{P}_{\mathsf{Y}})$, respectively, and let $\delta \in (0,1]$. Let $\Upsilon : \mathcal{X} \times \mathcal{Y} \times (0,1] \to \{\text{true}, \text{false}\}$ be any measurable formula such that for any x and y we have $\{\delta \in (0,1] \mid \Upsilon(x,y,\delta)\} = (0, \delta_{\max}]$ for some $\delta_{\max} \in \mathbb{R}$. If for all $x \in \mathcal{X}$ and for all $\delta \in (0,1]$*

$$\mathbf{P}_{\mathsf{Y}}\left(\Upsilon\left(x, \mathsf{Y}, \delta\right)\right) \geq 1 - \delta$$

then for any $\delta \in (0,1]$ and $\beta \in (0,1)$ we have

$$\mathbf{P}_{\mathsf{Y}}\left(\forall \alpha \in (0,1] :\ \mathbf{P}_{\mathsf{X}}\left(\Upsilon\left(\mathsf{X}, \mathsf{Y}, (\alpha\beta\delta)^{\frac{1}{1-\beta}}\right)\right) \geq 1 - \alpha\right) \geq 1 - \delta\,.$$

The proof of the quantifier reversal lemma can be found in Appendix A. Note that a very similar lemma is given by Lemma 11 and proved in Chapter 4. Now let us turn to the proof of the PAC-Bayesian subset bound, Theorem 25.

Proof. From the proof of Theorem 24 we know that for all $h \in \mathcal{H}$

$$\mathbf{P}_{\mathsf{Z}^m}\left(h \in V\left(\mathbf{Z}\right) \ \wedge \ R\left[h\right] > \frac{1}{m}\log\frac{1}{\delta}\right) \leq \delta\,.$$

Let $\Upsilon\left(h, \boldsymbol{z}, \delta\right)$ be the formula

$$\Upsilon\left(h, \boldsymbol{z}, \delta\right) \ = \ \left(h \in V\left(\boldsymbol{z}\right) \ \Rightarrow \ R\left[h\right] \leq \frac{\log\frac{1}{\delta}}{m}\right).$$

Then we can rewrite the bound for all single classifiers $h \in \mathcal{H}$ and for all $\delta \in (0, 1]$ as

$$\mathbf{P}_{\mathsf{Z}^m}\left(\Upsilon\left(h, \mathbf{Z}, \delta\right)\right) \geq 1 - \delta\,.$$

From the application of the quantifier reversal lemma, Lemma 3, with $h \equiv x$ and $\boldsymbol{z} \equiv y$ we get for any $\delta \in (0, 1]$ and $\beta \in (0, 1)$,

$$\mathbf{P}_{\mathsf{Z}^m}\left(\forall\alpha \in (0,1] : \ \mathbf{P}_{\mathsf{H}}\left(\mathsf{H} \in V\left(\mathbf{Z}\right) \ \Rightarrow \ R\left[\mathsf{H}\right] \leq \frac{1}{(1-\beta)\,m}\log\frac{1}{\alpha\beta\delta}\right) \geq 1 - \alpha\right) \geq 1 - \delta\,.$$

We can now instantiate α by $\alpha = \frac{\mathbf{P}_{\mathsf{H}}(U)}{m}$ with $U \subseteq V\left(\boldsymbol{z}\right)$ for the given $\boldsymbol{z}$. Then with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ it holds that

$$\forall\frac{\mathbf{P}_{\mathsf{H}}\left(U\right)}{m} \in (0,1] : \ \mathbf{P}_{\mathsf{H}}\left(\mathsf{H} \in V\left(\mathbf{Z}\right) \ \Rightarrow \ R\left[\mathsf{H}\right] \leq \frac{\log\frac{1}{\mathbf{P}_{\mathsf{H}}(U)} + \log\frac{m}{\beta\delta}}{(1-\beta)\,m}\right) \geq 1 - \frac{\mathbf{P}_{\mathsf{H}}\left(U\right)}{m}\,.$$

Using $1 - \frac{\mathbf{P}_{\mathsf{H}}(U)}{m} \geq \left(1 - \frac{1}{m}\right)\mathbf{P}_{\mathsf{H}}\left(U\right)$ we have a bound on the prediction error for a fraction of $1 - \frac{1}{m}$ of the classifiers $h \in U$. The remaining fraction of $\frac{1}{m}$ classifiers can have at most prediction error $R\left[h\right] = 1$. We can now determine an upper bound on the average $\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[R\left[\mathsf{H}\right]\right]$ of interest and choose $\beta = \frac{1}{m}$, which leads to

$$\begin{aligned}\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[R\left[\mathsf{H}\right]\right] &\leq \left(1 - \frac{1}{m}\right)\frac{\log\frac{1}{\mathbf{P}_{\mathsf{H}}(U)} + \log\frac{1}{\beta\delta} + \log m}{(1-\beta)\,m} + \frac{1}{m} \\ &\leq \frac{1}{m}\log\frac{1}{\mathbf{P}_{\mathsf{H}}\left(U\right)} + \log\frac{1}{\delta} + 2\log m + 1\,,\end{aligned}$$

completing the proof. □

3.3.2. Gibbs Classification Strategy: Unrealisable Case

The unrealisable case is analogous to Theorem 12 in the simple PAC framework. Although we will consider only the zero-one loss the subsequent two theorems are really applicable to arbitrary bounded loss functions.

Theorem 26 (PAC-Bayesian bound for single classifiers). *For any hypothesis space $\mathcal{H}$ of finite cardinality $|\mathcal{H}|$, for any prior probability distribution $\mathbf{P}_{\mathsf{H}}$ satisfying $\mathbf{P}_{\mathsf{H}}(h) > 0$ for all $h \in \mathcal{H}$, for any probability measure $\mathbf{P}_{\mathsf{Z}}$, for every sample size $m > 0$, and for all $\delta \in (0,1]$ we have with probability at least $1-\delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that for any hypothesis $h \in \mathcal{H}$, the prediction error $R[h]$ is bounded by*

$$R[h] \leq \hat{R}[h, \boldsymbol{z}] + \sqrt{\frac{1}{2m}\left(\log \frac{1}{\mathbf{P}_{\mathsf{H}}(h)} + \log \frac{1}{\delta}\right)}\,.$$

Proof. The proof is equivalent to the proof of Theorem 12 with the simple union bound replaced by the stratification lemma, Lemma 1, again effectively stratifying over all hypotheses $h \in \mathcal{H}$. We have from Hoeffding´s inequality, Theorem 61, that for all $h \in \mathcal{H}$

$$\mathbf{P}_{\mathsf{Z}^m}\left(R[h] - \hat{R}[h, \mathbf{Z}] > \varepsilon\right) \leq \exp\left(-2m\varepsilon^2\right)\,,$$

Now we apply the stratification lemma, Lemma 1, with

$$\Upsilon_h(\mathbf{Z}, \delta) = \left(R[h] - \hat{R}[h, \mathbf{Z}] \leq \sqrt{\frac{1}{2m}\log\frac{1}{\delta}}\right)\,,$$

and $p_h = \mathbf{P}_{\mathsf{H}}(h)$ satisfying $\sum_{h\in\mathcal{H}} p_h = \sum_{h\in\mathcal{H}} \mathbf{P}_{\mathsf{H}}(h) = 1$, which gives

$$\mathbf{P}_{\mathsf{Z}^m}\left(\bigwedge_{h\in\mathcal{H}}\left(\hat{R}[h, \mathbf{Z}] - R[h]\right) \leq \sqrt{\frac{1}{2m}\log\frac{1}{\delta \cdot \mathbf{P}_{\mathsf{H}}(h)}}\right) \geq 1-\delta\,,$$

which concludes the proof. □

Again we aim at providing a bound on the average prediction error of a subset $U \subseteq \mathcal{H}$.

Theorem 27 (PAC-Bayesian subset bound). *For any hypothesis space $\mathcal{H}$, for any prior probability distribution $\mathbf{P}_{\mathsf{H}}$ over $\mathcal{H}$, for any probability measure $\mathbf{P}_{\mathsf{Z}}$, for every sample size $m > 0$, and for all $\delta \in (0,1]$ we have with probability at least $1-\delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that for any subset $U \subseteq \mathcal{H}$ of hypothesis space space, the average prediction error $\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}[R[\mathsf{H}]]$ is bounded by*

$$\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}[R[\mathsf{H}]] \leq \mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[\hat{R}[\mathsf{H}, \boldsymbol{z}]\right] + \sqrt{\frac{1}{2m}\left(\log\frac{1}{\mathbf{P}_{\mathsf{H}}(U)} + \log\frac{1}{\delta} + 2\log m\right)} + \frac{1}{m}\,.$$

Proof. The proof proceeds in analogy to the proof of Theorem 25 with the binomial tail bound, Theorem 58, replaced by Hoeffding´s inequality, Theorem 61. Let $\Upsilon(h, \boldsymbol{z}, \delta)$ be the formula

$$\Upsilon(h, \boldsymbol{z}, \delta) = \left(R[h] - \hat{R}[h, \mathbf{Z}] \leq \sqrt{\frac{1}{2m} \log \frac{1}{\delta}} \right) .$$

Then we can rewrite the bound for all single classifiers $h \in \mathcal{H}$ and for all $\delta \in (0, 1]$ as

$$\mathbf{P}_{\mathbf{Z}^m} \left(\Upsilon(h, \mathbf{Z}, \delta) \right) \geq 1 - \delta .$$

From the application of the quantifier reversal lemma, Lemma 3, with $h \equiv x$ and $\boldsymbol{z} \equiv y$ we get for any $\delta \in (0, 1]$ and $\beta \in (0, 1)$,

$$\mathbf{P}_{\mathbf{Z}^m} \left(\forall \alpha \in (0, 1] : \mathbf{P}_{\mathsf{H}} \left(R[\mathsf{H}] - \hat{R}[\mathsf{H}, \mathbf{Z}] \leq \sqrt{\frac{1}{2m(1-\beta)} \log \frac{1}{\alpha\beta\delta}} \right) \geq 1 - \alpha \right) \geq 1 - \delta .$$

Again, we instantiate α by $\alpha = \frac{\mathbf{P}_{\mathsf{H}}(U)}{m}$ with $U \subseteq \mathcal{H}$ yielding

$$\mathbf{P}_{\mathbf{Z}^m} \left(\forall \frac{\mathbf{P}_{\mathsf{H}}(U)}{m} \in (0, 1] : \mathbf{P}_{\mathsf{H}} \left(R[\mathsf{H}] - \hat{R}[\mathsf{H}, \mathbf{Z}] \leq \sqrt{\frac{\log \frac{m}{\mathbf{P}_{\mathsf{H}}(U)\beta\delta}}{2m(1-\beta)}} \right) \geq 1 - \frac{\mathbf{P}_{\mathsf{H}}(U)}{m} \right) \geq 1 - \delta .$$

And so we have a bound on the prediction error for a fraction of $1 - \frac{1}{m}$ of the classifiers $h \in U$. The remaining fraction of $\frac{1}{m}$ classifiers can have at most prediction error $R[h] = 1$. We can now determine an upper bound on the average $\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}[R[\mathsf{H}]]$ of interest and choose $\beta = \frac{1}{m}$ which leads to

$$\begin{aligned} \mathbf{E}_{\mathsf{H}|\mathsf{H}\in U} \left[R[\mathsf{H}] - \hat{R}[\mathsf{H}, \mathbf{Z}] \right] &\leq \left(1 - \frac{1}{m}\right) \sqrt{\frac{\log \frac{1}{\mathbf{P}_{\mathsf{H}}(U)} + \log \frac{1}{\beta\delta} + \log m}{2m(1-\beta)}} + \frac{1}{m} \\ &\leq \sqrt{\frac{1}{2m} \log \frac{1}{\mathbf{P}_{\mathsf{H}}(U)} + \log \frac{1}{\delta} + 2 \log m} + \frac{1}{m} . \end{aligned}$$

□

Theorem 27 constitutes a powerful result for Bayesian learning based on the Gibbs classification strategy. It motivates an algorithm that chooses a subset $U \subset \mathcal{H}$ with a low average of $\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U} \left[\hat{R}[\mathsf{H}, \mathbf{Z}] \right]$ of empirical risk and bounds the error of the Gibbs classification strategy in terms of that quantity and the apriori belief $\mathbf{P}_{\mathsf{H}}(U)$ in that subset. Again the correctness but not the quality of the bound is independent of the prior: If the prior belief was wrong, i.e. little apriori weight was assigned to subsets of low empirical risk then the bound gives trivial values unless the sample size m is very high. On the other hand, if a lot of prior weight has been given to a subset of low average empirical risk then a Gibbs classification strategy based on such a subset is guaranteed a low prediction error.

Recently, even stronger results have been obtained in the PAC-Bayesian framework (McAllester 1999). The constants in Theorems 25 and 27 have been improved upon and results for more general so-called cut-off posteriors have been obtained. For our discussion, however, the subset bounds as presented here will be sufficient.

4. PAC-Bayesian Margin Bounds

In Subsection 3.1.3 it was shown how the prediction error of a linear classifier can be successfully bounded from above in terms of the margin that classifier achieves on the training data. The basic idea of that result is to apply the union bound to a finite cover of the class of linear functions whose effective size is determined by the margin of the classifier. This coarse-grain view of the function class involved leads to Theorem 22 that qualitatively justifies algorithms that maximise the margin.

In this chapter we will present PAC-Bayesian bounds on the generalisation error of linear classifiers. The basic idea is to apply Theorem 25 to the class of linear classifiers under a suitable choice of prior and subset. While the first PAC-Bayesian results by Shawe-Taylor and Williamson (1997) succeeded in bounding the prediction error of a single Bayesian classifier, subsequent PAC-Bayesian analyses (McAllester 1998; McAllester 1999) were applicable to the Gibbs classification strategy but not to single classifiers. In a sense this chapter aims at closing that gap for the class of linear classifiers. Since we are primarily interested in the prediction error of single classifiers we first bound the prediction error of the Bayes classification strategy w.r.t. a subset $U \in V(\boldsymbol{z})$ in terms of the average prediction error on the same subset U. Next we define the notion of Bayes-admissibility that relates the predictions of a single classifier to those of the Bayes classification strategy on a certain subset $U \in V(\boldsymbol{z})$. For linear classifiers it is shown that point-symmetric sets are Bayes-admissible w.r.t. their centre of symmetry. This leads to an interesting bound on the prediction error of linear classifiers in terms of a symmetrised version space volume. Since this quantity is not very practical we derive another bound in terms of a new quantity that we term the *normalised margin*. This margin serves as a measure of the volume ratio between consistent and non-consistent classifiers. Empirical results support the view that the normalised margin is the quantity that should be minimised to obtain optimal prediction. We demonstrate in the context of the *cummerbound* how apriori or aposteriori knowledge about $\mathbf{P}_X$ can be used to further improve the tightness of the bound. Finally, we derive a margin distribution bound that is applicable also to non-consistent classifiers. Sections 4.1 and 4.2 are based on Herbrich and Graepel (2001b) and Herbrich and Graepel (2001c). The results of Section 4.4 have been presented at a NIPS workshop (see Graepel and Herbrich (2001b)).

4.1. PAC-Bayesian Bounds on Single Classifiers

The PAC-Bayesian bounds, Theorems 25 and 27, on which the subsequent results are based bound the prediction error of the Gibbs classification strategy. So the first crucial

step is to bound the prediction error of a single classifier in terms of these quantities.

4.1.1. The Bayes Gibbs Lemma and Bayes Admissibility

Theorem 25 bounds the prediction error $R[\mathrm{Gibbs}_U] \stackrel{\mathrm{def}}{=} \mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}[R[\mathsf{H}]]$ of the Gibbs classification strategy. It turns out that in order to obtain a bound for a *single* classifier h we need two steps: Find a subset $U \subset \mathcal{W}$ such that the Bayes classification strategy Bayes_U based on U has the same prediction error $R[\mathrm{Bayes}_U] \stackrel{\mathrm{def}}{=} R\left[\mathrm{sign}\left(\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}[\mathsf{H}]\right)\right]$ (in fact: yields the same classification) as h, i.e., $R[\mathrm{Bayes}_U] = R[h]$ and then bound the prediction error $R[\mathrm{Bayes}_U]$ in terms of the quantity $R[\mathrm{Gibbs}_U]$ that is bounded by Theorem 25. To this end consider the following lemma that relates the prediction errors of the Bayes and the Gibbs classification strategies Bayes_U and Gibbs_U, carried out w.r.t. a subset $U \subset \mathcal{H}$ of hypothesis space.

Lemma 4 (Bayes-Gibbs lemma). *For any two measures* $\mathbf{P}_\mathsf{H}$ *and* $\mathbf{P}_\mathsf{Z}$*, the zero-one loss* l_{0-1}*, and any subset* $U \subset \mathcal{H}$ *we have*

$$R[\mathrm{Bayes}_U] \leq 2 \cdot R[\mathrm{Gibbs}_U]$$

Proof. Recall that

$$R[\mathrm{Gibbs}_U] = \mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}[R[\mathsf{H}]]$$

and

$$R[\mathrm{Bayes}_U] = R\left[x \mapsto \operatorname*{argmax}_{y\in\mathcal{Y}} \mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[\mathbf{I}_{\mathsf{H}(x)=y}\right]\right] .$$

Note that for all $(x, y) \in \mathcal{Z}$ we have that

$$\mathbf{I}_{\mathrm{Bayes}_U(x)\neq y} = 1 \;\Rightarrow\; 2 \cdot \mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[\mathbf{I}_{\mathsf{H}(x)\neq y}\right] \geq 1 .$$

Taking expectations over $\mathbf{P}_{\mathsf{XY}}$, and exchanging expectations on the right-hand side gives

$$\begin{aligned} R[\mathrm{Bayes}_U] = \mathbf{E}_{\mathsf{XY}}\left[\mathbf{I}_{\mathrm{Bayes}_U(x)\neq y}\right] &\leq 2 \cdot \mathbf{E}_{\mathsf{XY}}\left[\mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[\mathbf{I}_{\mathsf{H}(x)\neq y}\right]\right] \\ &\leq 2 \cdot \mathbf{E}_{\mathsf{H}|\mathsf{H}\in U}\left[\mathbf{E}_{\mathsf{XY}}\left[\mathbf{I}_{\mathsf{H}(x)\neq y}\right]\right] = 2 \cdot R[\mathrm{Gibbs}_U] , \end{aligned}$$

which immediately yields the result. □

Now that the relation between the Bayes and the Gibbs classification strategies and their respective prediction errors has been given, we need to find a subset $U \subset \mathcal{H}$ such that the Bayes classification strategy based on U corresponds to the classification of a single classifier $h \in \mathcal{H}$. Formally we require Bayes admissibility in the sense of the following definition:

Definition 52 (Bayes admissibility). Given an input space $\mathcal{X}$, an output space $\mathcal{Y}$, an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^\mathcal{X}$, and a prior $\mathbf{P}_\mathsf{H}$ over $\mathcal{H}$ we call a subset $U \subseteq \mathcal{H}$ *Bayes-admissible* w.r.t. $h \in \mathcal{H}$ and $\mathbf{P}_\mathsf{H}$ if and only if for all $x \in \mathcal{X}$ we have

$$h(x) = \mathrm{Bayes}_U(x) .$$

Bayes admissibility is a rather restrictive condition and is likely to be achieved only in hypothesis spaces that display strong symmetries. Effectively we are requiring that the voting behaviour of a subset $U \subset \mathcal{H}$ can be emulated or summarised by a single hypothesis $h \in \mathcal{H}$. It turns out, however, that in the case of linear classifiers, Bayes admissibility can in fact be employed in a useful way.

4.1.2. Point-Symmetric Subsets

Let us now turn to the hypothesis space $\mathcal{H}_\mathcal{K}$ of interest (see Definition 19). Due to the normalisation condition $\|\mathbf{w}\|^2 = 1$ there exists a one-to-one correspondence between the hypotheses $h \in \mathcal{H}_\mathcal{K}$ and normalised weight vectors $\mathbf{w} \in \mathcal{W}$. Thus defining a uniform measure $\mathbf{P_W}$ over $\mathcal{W}$ induces a measure $\mathbf{P_H}$ over $\mathcal{H}$. It is now necessary to consider a subset $\mathcal{Q}(\mathbf{w}) \subset \mathcal{W}$ that is Bayes admissible w.r.t. $\mathbf{w}$ and the uniform measure. Since we are primarily interested in the situation where $\mathbf{P_W}$ is the uniform measure, $\mathbf{P_W} = \mathbf{U_W}$, we can consider the notion of Bayes-admissibility for this special case. To get a better geometrical grasp of Bayes-admissibility let us consider point symmetric convex sets in ℓ_2^n (see Definition 88 in Appendix B, Section B.2 for convexity).

Definition 53 (Point-symmetry of a convex set). A compact convex subset $\mathcal{Q} \subset \ell_2^n$ is said to be *point-symmetric* if there exists a vector $\mathbf{w} \in \mathcal{Q}$ such that for all $\mathbf{v} \in \mathcal{Q}$,

$$\mathbf{v} + 2(\mathbf{w} - \mathbf{v}) \in \mathcal{Q}.$$

The vector $\mathbf{w}$ is called the centre of symmetry of $\mathcal{Q}$.

In order to establish the relation between point-symmetry and Bayes-admissibility we define for compact convex sets $\mathcal{Q}$ and $y \in \{+1, -1\}$,

$$\mathcal{Q}_y(\mathbf{x}) \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathcal{Q} \,|\, y\langle \mathbf{v}, \mathbf{x}\rangle > 0\},$$

that is $\mathcal{Q}_y(\mathbf{x})$ is the subset of $\mathcal{Q}$ such that the weight vectors contained in $\mathcal{Q}_y(\mathbf{x})$ assign the label y to input $\mathbf{x}$. The relation between point-symmetry and Bayes-admissibility is then given by the following lemma.

Lemma 5 (Point-symmetry and Bayes-admissibility). *If a compact convex subset $\mathcal{Q} \subset \ell_2^n$ is point-symmetric w.r.t. $\mathbf{w} \in \ell_2^n$ then it is Bayes-admissible w.r.t. $\mathbf{w} \in \ell_2^n$ under the uniform measure $\mathbf{U_W}$.*

Proof. Let $\mathbf{w}$ be the centre of symmetry of $\mathcal{Q}$. Then by point-symmetry and corresponding symmetry of measurable subvolumes of $\mathcal{Q}$ the vector $\mathbf{w}$ always lies in the half of greater volume of $\mathcal{Q}$, i.e., $\mathbf{w} \in \mathcal{Q}_{\tilde{y}}(\mathbf{x})$ such that $\tilde{y} = \operatorname{argmax}_{y \in \mathcal{Y}} \operatorname{vol}(\mathcal{Q}_y(\mathbf{x}))$ unless there is a tie. The classifier $h_\mathbf{w}$ thus emulates the Bayes classification strategy. □

The relation between point-symmetry and Bayes-admissibility can be seen in Figure 4.1. Shown are lines corresponding to inputs $\mathbf{x}$ for which $\mathcal{Q}_{-1}(\mathbf{x}) = \mathcal{Q}_{+1}(\mathbf{x})$. Clearly, in the case of point-symmetry these lines intersect at the centre of symmetry and the set is thus Bayes-admissible.

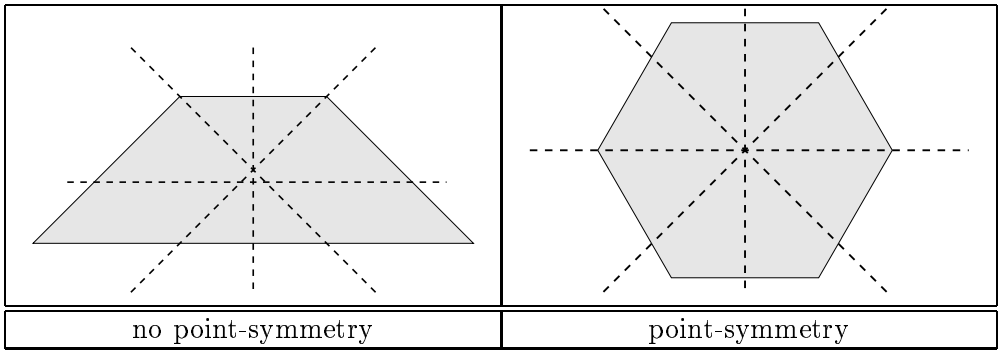


Figure 4.1.: Illustration of the relation between point-symmetry and Bayes-admissibility. In the left plot, the set $\mathcal{Q}$ is not point-symmetric. On the right, $\mathcal{Q}$ is point-symmetric, and hence Bayes-admissible w.r.t. its centre of symmetry.

In order to maximise the bound on the prediction error of a classifier $h_{\mathbf{w}}$ one should aim at considering the largest subset $\mathcal{Q} \in V(\boldsymbol{z})$ of version space that is point-symmetric w.r.t. $\mathbf{w}$. To this end consider Figure 4.2.

Clearly, the largest point-symmetric subset $\mathcal{Q}$ of version space is obtained by point-mirroring the constraints given by the training examples $\mathbf{x}_i$, thus constructing an auxiliary *mirrored* version space $V(\tilde{\boldsymbol{z}}(\mathbf{w}))$ w.r.t. any $\mathbf{w} \in V(\boldsymbol{z})$.

Definition 54 (Point-mirrored training sample). Given a training sample $\boldsymbol{z}$ we define the training sample $\tilde{\boldsymbol{z}}$ point-mirrored w.r.t. $\mathbf{w} \in \mathcal{W}$ as

$$\tilde{z}_i(\mathbf{w}) \stackrel{\text{def}}{=} (\mathbf{x}_i - 2\langle \mathbf{w}, \mathbf{x}_i \rangle \mathbf{w}, -y_i) \ .$$

Note that point-mirroring the constraints given by the training examples in weight space corresponds to mirroring the training inputs $\mathbf{x}_i$ w.r.t. the hyperplane with normal vector $\mathbf{w}$ in feature space. The subset $\mathcal{Q}(\boldsymbol{z}, \mathbf{w})$ of version space point-symmetric w.r.t. $\mathbf{w}$ is defined as the intersection of $V(\boldsymbol{z})$ and $V(\tilde{\boldsymbol{z}}(\mathbf{w}))$, i.e.,

$$\mathcal{Q}(\boldsymbol{z}, \mathbf{w}) \stackrel{\text{def}}{=} V(\boldsymbol{z}) \cap V(\tilde{\boldsymbol{z}}(\mathbf{w})) \ .$$

Note, that the intersection of two convex sets is always convex (Kolmogorov 1957). Moreover, by construction $h_{\mathbf{w}}$ agrees with the Bayesian classification strategy under a uniform measure on $\mathcal{Q}$: For every data point $\mathbf{x}$ bisecting $\mathcal{Q}(\boldsymbol{z}, \mathbf{w})$, $\mathbf{w}$ lies in the half of greater volume.

The above argument holds for any classifier $\mathbf{w} \in V(\boldsymbol{z})$ and thus allows the application of Theorem 25 to arbitrary version space members. The volume to be considered in the bound is $\mathrm{vol}(\mathcal{Q}(\boldsymbol{z}, \mathbf{w}))$. This gives the following bound.

Theorem 28 (PAC-Bayesian point-symmetric subset bound). *For the hypothesis space $\mathcal{H}_{\mathcal{K}}$ with the uniform prior U_{W} over $\mathcal{W}$, for any probability measure P_{Z}, for any sample size $m > 0$, and for all $\delta \in (0, 1]$ we have with probability at least $1 - \delta$ over the*

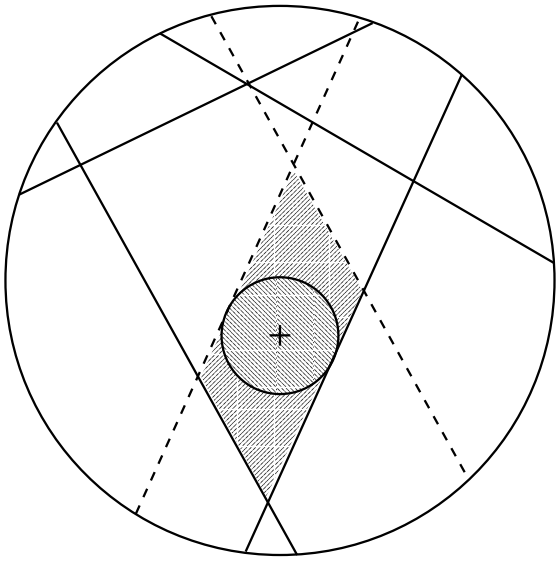


Figure 4.2.: Illustration of a point-symmetric subset of version space obtained by mirroring hyperplanes $\{\mathbf{v} \in \mathcal{W} \,|\, \langle \mathbf{v}, \mathbf{x} \rangle = 0\}$ w.r.t. the weight vector $\mathbf{w}$ of interest. The subset $\mathcal{Q}(\mathbf{w})$ thus obtained is considerable larger than the ball $\mathcal{B}_\gamma(\mathbf{w})$ with margin γ. This increase in volume of the subset considered leads to lower values of the bound on the prediction error of $h_\mathbf{w}$.

random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ *that for any consistent classifier* $h_\mathbf{w} \in \mathcal{H}_\mathcal{K}$ *the prediction error* $R[h_\mathbf{w}]$ *is bounded by*

$$R[h_\mathbf{w}] \leq \frac{2}{m}\left(\log \frac{\mathrm{vol}(\mathcal{W})}{\mathrm{vol}(\mathcal{Q}(\boldsymbol{z}, \mathbf{w}))} + \log \frac{1}{\delta} + 2\log m + 1\right),$$

provided that $\mathcal{Q}(\boldsymbol{z}, \mathbf{w}) = V(\boldsymbol{z}) \cap V(\tilde{\boldsymbol{z}}(\mathbf{w}))$.

Proof. From the combination of the PAC-Bayesian subset bound for consistent classifiers, Theorem 25, with the Bayes-Gibbs lemma, Lemma 4 we get a bound on a single classifier $h_\mathbf{w}$ provided that the subset $\mathcal{Q}(\boldsymbol{z}, \mathbf{w})$ is Bayes-admissible w.r.t. $\mathbf{w}$. Since $\mathcal{Q}(\boldsymbol{z}, \mathbf{w}) = V(\boldsymbol{z}) \cap V(\tilde{\boldsymbol{z}}(\mathbf{w}))$ we have that $\mathcal{Q}(\boldsymbol{z}, \mathbf{w})$ is point-symmetric w.r.t. $\mathbf{w}$ and hence, using Lemma 5, Bayes-admissible. □

Clearly, the bound on the prediction error is minimised by maximising the quantity $\mathrm{vol}(\mathcal{Q}(\boldsymbol{z}, \mathbf{w}))$. Given a fixed training sample $\boldsymbol{z}$ this amounts to choosing a weight vector $\mathbf{w}$ that maximises the volume of its corresponding point-symmetric subset $\mathcal{Q} \subseteq V(\boldsymbol{z})$ of version space. This suggests a new training algorithm that maximises Bayes-admissible volume. While such an approach appears to be promising from a theoretical point of view, we did not pursue this avenue of research any further. Certainly, such an algorithm might be superior to simple margin maximisers. However, it is to be expected that in the presence of feature noise this advantage may be cancelled out because the version space scenario is not valid anymore.

4.2. PAC-Bayesian Margin Bound

4.2.1. Volume Ratio and Ball Approximation

While the previous result appears to be the tightest way of bounding the prediction error of a classifier $h_{\mathbf{w}}$ in terms of its geometric properties, the result is not really practically applicable, because the volume $\mathrm{vol}\left(\mathcal{Q}\left(\boldsymbol{z}, \mathbf{w}\right)\right)$ is not easily accessible. After all it requires to calculate the volume of a rather irregular polyhedron in a possibly high dimensional space.

In order to provide a more usable bound we can employ the following statement about Bayes admissibility of balls $\mathcal{B}$ in $\mathcal{H}_{\mathcal{K}}$.

Lemma 6 (Bayes admissibility of balls in $\mathcal{H}_{\mathcal{K}}$). *For the uniform probability measure U_{W} over $\mathcal{W}$ and for every $r > 0$ the ball*

$$\mathcal{B}\left(\mathbf{w}, r\right) \stackrel{\text{def}}{=} \left\{\mathbf{v} \in \mathcal{W} \,|\, \|\mathbf{v} - \mathbf{w}\| \leq r\right\}$$

is Bayes-admissible w.r.t. its centre $\mathbf{w} \in \mathcal{W}$.

Proof. The Bayes admissibility of $\mathcal{B}\left(\mathbf{w}, r\right)$ w.r.t. its centre $\mathbf{w}$ follows from the point-symmetry of $\mathcal{B}\left(\mathbf{w}, r\right)$ (see Lemma 5). □

Please note that by considering a ball $\mathcal{Q}\left(\mathbf{w}\right) \equiv \mathcal{B}\left(\mathbf{w}\right)$ rather than just $\mathbf{w}$ we make use of the fact that $h_{\mathbf{w}}$ *summarises* all its neighbouring classifiers $h_{\mathbf{v}}$ with $\mathbf{v} \in \mathcal{B}\left(\mathbf{w}\right)$, where we have dropped the dependency on r for notational convenience. Now under a uniform prior U_{W} the *normalised* margin $\Gamma\left(\mathbf{w}, \boldsymbol{z}\right)$ quantifies the relative volume of classifiers summarised by $\mathbf{w}$ and thus allows us to bound its risk.

Definition 55 (Normalised margin). Given a training example z_i, a training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, a feature mapping $\boldsymbol{\phi}$ and a weight vector $\mathbf{w} \in \mathcal{K}$ we call

$$\Gamma\left(\mathbf{w}, z_i\right) \stackrel{\text{def}}{=} \frac{y_i \left\langle \mathbf{w}, \mathbf{x}_i \right\rangle}{\|\mathbf{w}\| \, \|\mathbf{x}_i\|} ,$$

the *normalised margin* of weight vector $\mathbf{w}$ on training example z_i, and

$$\Gamma\left(\mathbf{w}, \boldsymbol{z}\right) \stackrel{\text{def}}{=} \min_{z_i \in \boldsymbol{z}} \Gamma\left(\mathbf{w}, z_i\right) , \tag{4.1}$$

the *normalised margin* of weight vector $\mathbf{w}$ on training sample $\boldsymbol{z}$.

Note that in contrast to the classical margin $\gamma\left(\mathbf{w}, \boldsymbol{z}\right)$ (see Definition 21) this *normalised* margin is a dimensionless quantity and constitutes a measure for the relative size of the version space $V\left(\boldsymbol{z}\right)$ invariant under rescaling of both weight vectors $\mathbf{w}$ and feature vectors $\mathbf{x}_i$. It is thus meaningful without knowledge of the data radius $\varsigma\left(\boldsymbol{x}\right)$ given in Definition 22.

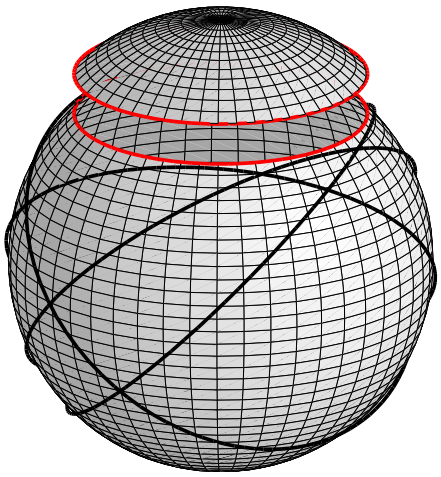


Figure 4.3.: Illustration of the volume ratio for the classifier $h_{\mathbf{w}}$ at the north pole. A sample $\boldsymbol{x}$ of four training inputs $\mathbf{x}_i$ divides the surface of the sphere into different polyhedra consistent corresponding to different labellings $\boldsymbol{y}$ of the sample. Assuming that $h_{\mathbf{w}}$ is consistent with the training sample $\boldsymbol{z}$, the top polyhedron corresponds to version space $V(\boldsymbol{z})$. The radius of the cap of the sphere is proportional to the normalised margin $\Gamma(\mathbf{w}, \boldsymbol{z})$.

4.2.2. The Margin Bound for Consistent Classifiers

The proof of the main result of this section is rather complex. Conceptually, however, it consists of the following steps that each by itself are rather simple.

1. Define a uniform prior $\mathbf{U_W}$ over the sphere $\mathcal{W}$ of all possible weight vectors of norm $\|\mathbf{w}\| = 1$.
2. Given a classifier $h_{\mathbf{w}}$ choose as the relevant subset $\mathcal{Q} \equiv U$ in the PAC-Bayesian subset bound the largest ball $\mathcal{B}(\mathbf{w})$ centred at $\mathbf{w}$ that can be inscribed in version space $V(\boldsymbol{z}) \subset \mathcal{W}$.
3. Bound the prediction error $R[h_{\mathbf{w}}]$ of the centre $\mathbf{w}$ of $\mathcal{B}(\mathbf{w})$ in terms of the prediction error $R\left[\text{Bayes}_{\mathcal{B}(\mathbf{w})}\right]$ of the Bayes classification strategy based on $\mathcal{B}(\mathbf{w})$.
4. Bound $R\left[\text{Bayes}_{\mathcal{B}(\mathbf{w})}\right]$ in terms of the average prediction error $R\left[\text{Gibbs}_{\mathcal{B}(\mathbf{w})}\right]$ of classifiers in $\mathcal{B}(\mathbf{w})$.
5. Bound $R\left[\text{Gibbs}_{\mathcal{B}(\mathbf{w})}\right]$ in terms of the volume ratio between $\mathcal{B}(\mathbf{w})$ and $\mathcal{W}$.
6. Bound the volume ratio between $\mathcal{B}(\mathbf{w})$ and $\mathcal{W}$ in terms of the normalised margin $\Gamma(\mathbf{w}, \boldsymbol{z})$ achieved on the training sample $\boldsymbol{z}$.

Let us now turn to the main result of this section, a bound in terms of the normalised margin $\Gamma(\mathbf{w}, \boldsymbol{z})$.

Theorem 29 (PAC-Bayesian margin bound). *Given a feature space $\mathcal{K} \subseteq \ell_2^n$ of dimensionality n, for all probability measures $\mathbf{P_Z}$ with probability at least $1-\delta$ over the random draw of a training sample $\boldsymbol{z}$ of size m, any classifier $h_{\mathbf{w}}$ achieving a positive margin $\Gamma(\mathbf{w}, \boldsymbol{z}) > 0$ has prediction error $R[h_{\mathbf{w}}]$ bounded from above by*

$$R[h_{\mathbf{w}}] < \frac{2}{m}\left(d \log \frac{1}{1-\sqrt{1-\Gamma^2(\mathbf{w}, \boldsymbol{z})}} + \log \frac{(me)^2}{\delta}\right),$$

where $d = \min(m, n)$.

Proof. Geometrically the hypothesis space $\mathcal{W}$ is the unit sphere in ℓ_2^n (see Figure 4.3). Let us assume that $\mathbf{P_W}$ is uniform on the unit sphere, $\mathbf{P_W} = \mathbf{U_W}$, as suggested by symmetry. Given the training sample $\boldsymbol{z}$ and a weight vector $\mathbf{w}$ all weight vectors $\mathbf{v} \in \mathcal{B}(\mathbf{w})$

$$\mathcal{B}(\mathbf{w}) \stackrel{\text{def}}{=} \left\{\mathbf{v} \in \mathcal{W} \;\middle|\; \langle \mathbf{w}, \mathbf{v} \rangle > \sqrt{1-\Gamma^2(\mathbf{w}, \boldsymbol{z})}\right\} \tag{4.2}$$

are within $V(\boldsymbol{z})$ (see Theorem 53 in Appendix A.7.1). Such a set $\mathcal{B}(\mathbf{w})$ is Bayes-admissible by Lemma 5 and hence we can use $\mathbf{P_W}(\mathcal{B}(\mathbf{w}))$ to bound the generalisation error of $h_{\mathbf{w}}$. Since $\mathbf{P_W} = \mathbf{U_W}$ is uniform, the value $-\log(\mathbf{P_W}(\mathcal{B}(\mathbf{w})))$ is simply the logarithm of the *volume ratio* between the surface of the unit sphere and the surface of all $\mathbf{v}$ satisfying (4.2). In Theorem 55 in Appendix A.7.2 it is shown that this ratio is *exactly* given by

$$\log\left(\frac{\int_0^{2\pi} \sin^{n-2}(\theta)\, d\theta}{\int_0^{\arccos\left(\sqrt{1-\Gamma^2(\mathbf{w}, \boldsymbol{z})}\right)} \sin^{n-2}(\theta)\, d\theta}\right).$$

It can be shown that this ratio is bounded from above by (see Theorem 55)

$$n \cdot \log\left(\frac{1}{1-\sqrt{1-\Gamma^2(\mathbf{w}, \boldsymbol{z})}}\right) + \log(2).$$

With $\log 2 < 1$ we obtain the desired result. Note that m points maximally span an m-dimensional space and thus we can marginalise over the remaining $n-m$ dimensions of feature space $\mathcal{K}$. This gives $d = \min(m, n)$. □

An appealing feature of equation (4.8) is that for $\Gamma^2(\mathbf{w}, \boldsymbol{z}) = 1$ the bound reduces to

$$\frac{2}{m}\left(2\log(m) + \log\left(\frac{1}{\delta}\right) + 2\right)$$

with a rapid decay to zero as m increases. In case of margins $\Gamma(\mathbf{w}, \boldsymbol{z}) > 0.91$ the troublesome situation of $d = m$, which occurs e.g. for RBF kernels, is compensated for. Furthermore, upper bounding $1/(1-\sqrt{1-\Gamma^2})$ by $\frac{2}{\Gamma^2}$ (see Lemma 22 in Appendix A) we see that Theorem 29 is an exponential improvement of Theorem 22 in terms of the attained margins. It should be noted, however, that the new bound depends on the dimensionality of the input space via $d = \min(m, n)$.

4.3. The Cummerbound

In the derivation of Theorem 29 we decided for a uniform prior $\mathbf{P_W} = \mathbf{U_W}$ over $\mathcal{W}$ on the surface of the unit sphere. In practice, however, most of the hypotheses thus considered assign *all of the training data* to the same class, in particular, if the training data occupy only a small compact area of the unit sphere. In order to remedy this weakness we take advantage of this fact by excluding those hypotheses *a priori* that assign all training data to the same class. The subsequent analysis will be based on the assumption that we know from the beginning that the training sample $\boldsymbol{x}$ is contained in a convex set $\mathcal{C}$. However, the sets we consider will effectively be the intersection $\mathcal{C} \cap \mathcal{W}$ of convex sets $\mathcal{C} \subset \mathcal{K}$ with the unit sphere $\mathcal{W} \subset \mathcal{K}$.

4.3.1. Convex Sets and the Cummerbund

The relevant set $H_{\mathcal{C}}$ of hypotheses is characterised by the condition that there exists at least one pair $(\mathbf{x}, \tilde{\mathbf{x}}) \in \mathcal{X} \times \mathcal{X}$ of points that are assigned to different classes by any classifier $h_{\mathbf{w}} \in H_{\mathcal{C}}$. We call the set of all weight vectors $\mathbf{w}$ satisfying this condition a *cummerbund* (see Figure 4.4).

Definition 56 (Cummerbund). Suppose we are given a a feature space $\mathcal{K} \subseteq \ell_2^n$ and a convex set $\mathcal{C} \subset \mathcal{K}$. Then the *cummerbund* $\operatorname{kum}(\mathcal{C})$ is defined by

$$\operatorname{kum}(\mathcal{C}) \stackrel{\text{def}}{=} \{\mathbf{w} \in \mathcal{W} \,|\, \exists \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle = 0\} ,$$

where $\mathcal{W}$ is the unit hyper-sphere in ℓ_2^n.

The cummerbund $\operatorname{kum}(\mathcal{C})$ can also be expressed using the following relationship:

$$\begin{aligned} \operatorname{kum}(\mathcal{C}) &= \mathcal{W} \setminus \overline{\operatorname{kum}(\mathcal{C})} \\ &= \mathcal{W} \setminus \{\mathbf{w} \in \mathcal{W} \,|\, \forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle \neq 0\} \\ &= \mathcal{W} \setminus \{\mathbf{w} \in \mathcal{W} \,|\, \forall \mathbf{x} \in \mathcal{C} : (\langle \mathbf{w}, \mathbf{x} \rangle < 0 \vee \langle \mathbf{w}, \mathbf{x} \rangle > 0)\} . \end{aligned} \tag{4.3}$$

The following lemma will enable us to express the cummerbund in terms of convex sets referred to as *polars* in standard terminology from convex analysis (see, e.g. Rockafellar (1970)).

Lemma 7 (Cummerbund). *For convex sets $\mathcal{C} \subset \ell_2^n$ and the unit sphere $\mathcal{W} \subset \ell_2^n$ we have*

$$\{\mathbf{w} \in \mathcal{W} \,|\, \forall \mathbf{x} \in \mathcal{C} : (\langle \mathbf{w}, \mathbf{x} \rangle < 0 \vee \langle \mathbf{w}, \mathbf{x} \rangle > 0)\} = \tag{4.4}$$

$$\{\mathbf{w} \in \mathcal{W} \,|\, (\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle < 0) \vee (\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle > 0)\} . \tag{4.5}$$

The proof of the lemma can be found in Appendix A, Section A.8. In order to apply the PAC-Bayesian analysis to the cummerbund we would like to relate its volume to known and easily accessible quantities. To this end consider the definition of the set $\operatorname{pol}(\mathcal{C})$ that can be found in Appendix B, Section B.2. Intuitively speaking the following lemma expresses the fact that the equatorial region of a sphere can be expressed as the surface of the whole sphere without its two polar regions.

Lemma 8 (Cummerbund volume). *For convex sets $\mathcal{C} \subset \mathcal{W}$ we have*

$$\mathrm{vol}\left(\mathrm{kum}\left(\mathcal{C}\right)\right) = \mathrm{vol}\left(\mathcal{W}\right) - \mathrm{vol}\left(\mathrm{pol}\left(\mathcal{C}\right)\right) - \mathrm{vol}\left(\mathrm{pol}\left(-\mathcal{C}\right)\right) ,$$

where $-\mathcal{C}$ is understood as $-\mathcal{C} \stackrel{\text{def}}{=} \{-\mathbf{x} \,|\, \mathbf{x} \in \mathcal{C}\}$.

Proof. Using the definition of the cummerbund, Definition 56, and Lemma 7 we have that

$$\mathrm{kum}\left(\mathcal{C}\right) = \mathcal{W} \setminus \left(\{\mathbf{w} \in \mathcal{W} : \left(\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle < 0\right)\} \cup \{\mathbf{w} \in \mathcal{W} \,|\, \left(\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle > 0\right)\}\right) ,$$

because $\{\mathbf{w} \in \mathcal{W} \,|\, (\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle < 0) \wedge (\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle > 0)\} = \emptyset$. Hence for the volume of the cummerbund it follows

$$\mathrm{vol}\left(\mathrm{kum}\left(\mathcal{C}\right)\right) \quad = \quad \mathrm{vol}\left(\mathcal{W}\right) - \mathrm{vol}\left(\mathrm{pol}\left(\mathcal{C}\right)\right) - \mathrm{vol}\left(\mathrm{pol}\left(-\mathcal{C}\right)\right) ,$$

because $\{\mathbf{w} \in \mathcal{W} \,|\, (\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle > 0)\} = \{\mathbf{w} \in \mathcal{W} \,|\, (\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, -\mathbf{x} \rangle < 0)\}$. Assuming continuous measures $\mathbf{P}_{\mathbf{W}}$ on $\mathcal{W}$ we will not be concerned with the difference between an open polar $\mathrm{pol}\left(\mathcal{C}\right)$ and a closed polar $\overline{\mathrm{pol}}\left(\mathcal{C}\right)$ because we have for all measurable convex sets $\mathcal{C} \subset \mathcal{W}$ that

$$\mathbf{P}_{\mathbf{W}}\left(\mathrm{pol}\left(\mathcal{C}\right)\right) = \mathbf{P}_{\mathbf{W}}\left(\overline{\mathrm{pol}}\left(\mathcal{C}\right)\right) . \tag{4.6}$$

□

Then we have the following improvement on the PAC-Bayesian margin bound, Theorem 29, when the support $\mathrm{supp}\left(\mathsf{X}\right)$ is restricted to a convex subset $\mathcal{C} \subset \ell_2^n$

Theorem 30 (General cummerbound). *Given a feature space $\mathcal{K} \subseteq \ell_2^n$, for all compact convex sets $\mathcal{C} \subset \mathcal{X}$, for all probability measures $\mathbf{P}_{\mathsf{Z}}$ with $\mathrm{supp}\left(\mathsf{X}\right) \subseteq \mathcal{C}$, with probability at least $1 - \delta$ over the random draw of a training sample $\boldsymbol{z} \in \mathcal{Z}^m$ of size m, we have that for any consistent classifier $h_{\mathbf{w}}$ the prediction error $R\left[h_{\mathbf{w}}\right]$ is bounded from above by*

$$R\left[h_{\mathbf{w}}\right] < \frac{2}{m}\left(\log \frac{\mathrm{vol}\left(\mathcal{W}\right) - \mathrm{vol}\left(\mathrm{pol}\left(\mathcal{C}\right)\right) - \mathrm{vol}\left(\mathrm{pol}\left(-\mathcal{C}\right)\right)}{\mathrm{vol}\left(\mathcal{Q}\left(\boldsymbol{z}, \mathbf{w}\right)\right)} + \log \frac{m^2 e}{\delta}\right) , \tag{4.7}$$

provided that $\mathcal{Q}\left(\boldsymbol{z}, \mathbf{w}\right) = V\left(\boldsymbol{z}\right) \cap V\left(\tilde{\boldsymbol{z}}\left(\mathbf{w}\right)\right)$.

4.3.2. Cap and Cummerbund

For practical purposes we would like to use the given sample $\boldsymbol{z}$ for the determination of the factor $\mathrm{vol}\left(\mathrm{kum}\left(\mathcal{C}\right)\right)$. To achieve this we need to find a function $f : \cup_{m=1}^{\infty} \mathcal{Z}^m \to \mathbb{R}$ such that $f\left(\boldsymbol{z}\right) \geq \mathrm{vol}\left(\mathrm{kum}\left(\mathcal{C}\right)\right)$. We impose the restriction that, given a training sample $\boldsymbol{z}$, for every classifier $h_{\mathbf{w}}$ there exists at least one pair $(x_i, x_j) \in \boldsymbol{x} \times \boldsymbol{x}$, $i \neq j$ that is assigned to different classes, i.e. $\langle \mathbf{w}, \mathbf{x}_i \rangle \langle \mathbf{w}, \mathbf{x}_j \rangle \leq 0$. Then the smallest set $\mathcal{W}_{\boldsymbol{z}}$ of weight vectors satisfying this condition is given by $\mathrm{kum}\left(\mathrm{conv}\left(\boldsymbol{\phi}\left(\boldsymbol{x}\right)\right)\right)$, where $\mathrm{conv}\left(\boldsymbol{\phi}\left(\boldsymbol{x}\right)\right)$ is the *convex hull* of the set $\boldsymbol{\phi}\left(\boldsymbol{x}\right) \subset \mathcal{K}$ (see Appendix B, Section B.2 for a definition of convex

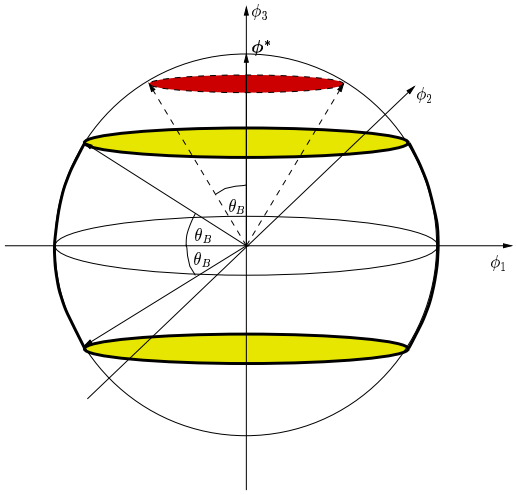


Figure 4.4.: Suppose the training data $\boldsymbol{x}$ is contained in a polar cap characterised by $\sphericalangle(\mathbf{x}, \boldsymbol{\phi}^*) \leq \theta_B$ where $\boldsymbol{\phi}^*$ is the north pole (top cap of the sphere). Those weight vectors $\mathbf{w} \in \mathcal{W}$ of hypotheses $h_{\mathbf{w}}$ assigning at least one pair of training points to different classes are contained in the *cummerbund* (tropical belt around equator).

hull). It is sufficient to have an upper bound on $\operatorname{vol}(\operatorname{kum}(\operatorname{conv}(X)))$ for all $m \in \mathbb{N}$ and all $\boldsymbol{x} \in \mathcal{X}^m$.

Possibly the simplest upper bound on $\operatorname{vol}(\operatorname{kum}(\operatorname{conv}(X)))$ can be obtained by assuming that the training sample $\boldsymbol{x}$ is contained in a cap $\mathcal{C}_{\text{cap}}$ (see Figure 4.4) with opening angle $2\theta_B$, i.e.,

$$\min_{(x_i,x_j)\in\mathcal{X}\times\mathcal{X}} \frac{|\langle \mathbf{x}_i, \mathbf{x}_j \rangle|}{\|\mathbf{x}_i\| \, \|\mathbf{x}_j\|} \geq \cos(2\theta_B) \; .$$

Just like the largest sphere inscribable in version space is characterised by the normalised margin Γ this cap can be characterised by the normalised data concentration Ξ.

Definition 57 (Normalised data concentration). Given an input sample $\boldsymbol{x}$, a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$ we call

$$\Xi(\boldsymbol{x}) \stackrel{\text{def}}{=} \max_{\boldsymbol{\phi}^* \in \mathcal{W}} \min_{x_i \in \boldsymbol{x}} \frac{\langle \mathbf{x}_i, \boldsymbol{\phi}^* \rangle}{\|\mathbf{x}_i\|}$$

the *normalised data concentration.*

We then have the following lemma on the volume of the cummerbund:

Lemma 9 (Cap cummerbund volume). *Given a feature space $\mathcal{K} \subseteq \ell_2^n$ of dimensionality n and an input sample $\boldsymbol{x} \in \mathcal{X}^m$ with normalised data concentration $\Xi(\boldsymbol{x})$ the volume of the cummerbund of the cap $\mathcal{C}_{\text{cap}}$ containing the data is bounded from above by*

$$\operatorname{vol}(\operatorname{kum}(\mathcal{C}_{\text{cap}}(\Xi(\boldsymbol{x})))) \leq 1 - \left(1 - \sqrt{1 - \Xi^2(\boldsymbol{x})}\right)^n \; .$$

The proof can be found in Appendix A, Section A.8. Combining the previous results we can finally provide a bound in terms of quantities that are easily observable, the normalised margin Γ and the concentration of the data, $\Xi(\boldsymbol{x})$.

Theorem 31 (Cummerbound). *Given a feature space $\mathcal{K} \subseteq \ell_2^n$ of dimensionality n, for all probability measures $\mathbf{P}_\mathbf{Z}$, for all $\delta \in (0,1]$ with probability at least $1-\delta$ over the random draw of a training sample $\boldsymbol{z} \in \mathcal{Z}^m$ of size m, any classifier $h_\mathbf{w}$ achieving a positive margin $\Gamma(\mathbf{w}, \boldsymbol{z}) > 0$ has prediction error $R[h_\mathbf{w}]$ bounded from above by*

$$R[h_\mathbf{w}] < \frac{2}{m}\left(\log \frac{1-\left(\frac{1}{2}\Xi^2(\boldsymbol{x})\right)^d}{\left(\frac{1}{2}\Gamma^2(\mathbf{w},\boldsymbol{z})\right)^d} + \log \frac{m(me)^2}{\delta}\right), \tag{4.8}$$

where $d = \min(m,n)$ and provided that $\Xi(\boldsymbol{x}) \geq \exp(-m)$.

Proof. The volume ratio from Theorem 30 is bounded by

$$\begin{aligned}\log\left(\frac{1-\mathrm{vol}(\mathrm{pol}(\mathcal{C}))-\mathrm{vol}(\mathrm{pol}(-\mathcal{C}))}{\mathrm{vol}(\mathcal{Q}(\boldsymbol{z},\mathbf{w}))}\right) &\leq \log\left(\frac{1-\left(1-\sqrt{1-\Xi^2(\boldsymbol{x})}\right)^d}{\left(1-\sqrt{1-\Gamma^2(\mathbf{w},\boldsymbol{z})}\right)^d}\right)\\ &\leq \log\left(\frac{1-\left(\frac{1}{2}\Xi^2(\boldsymbol{x})\right)^d}{\left(\frac{1}{2}\Gamma^2(\mathbf{w},\boldsymbol{z})\right)^d}\right),\end{aligned}$$

using Lemma 9 in the first line and twice Lemma 22 (see Appendix A, Section A.8) in the second line. Since the normalised data concentration $\Xi(\boldsymbol{x})$ enters the bound aposteriori as an observed quantity, we apply the stratification lemma, Lemma 1, with a sequence $\forall i \in \{1,\ldots,m\} \quad \Xi_i = \exp(-i)$ chosen such that for all Ξ there exists an index i such that $\Xi_i \leq \Xi$, and $p_i \equiv \frac{1}{m}$ resulting in an additional term $\frac{\log m}{m}$ in the bound. □

4.4. Margin Distribution Bound

4.4.1. Margin Distribution and Average Empirical Risk

While the previous bound nicely illustrates the idea of how to find a PAC-Bayesian bound on a single linear classifier, the pure version space scenario has its limits in practice. Also it appears to be a waste of information to consider only the minimal real-valued output in contrast to the whole margin distribution.

Building on the PAC-Bayesian margin bound, Theorem 29, let us define the normalised margin distribution.

Definition 58 (Normalised margin distribution). Given a training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, a feature mapping $\boldsymbol{\phi}$ and a weight vector $\mathbf{w} \in \mathcal{K}$, we call the sequence $\boldsymbol{\Gamma} \in \mathbb{R}^m$,

$$\boldsymbol{\Gamma} \stackrel{\text{def}}{=} \boldsymbol{\Gamma}(\mathbf{w}) \stackrel{\text{def}}{=} \left(\Gamma_{(1)},\ldots,\Gamma_{(m)}\right),$$

of normalised margins $\Gamma(\mathbf{w}, z_i)$ sorted in ascending order such that $\Gamma_{(i)} \leq \Gamma_{(j)}$ for $i < j$, the *normalised margin distribution.*

Our aim is to apply the PAC-Bayesian subset bound for the unrealisable case, Theorem 27, to a large cap $\mathcal{B}(\mathbf{w})$ around the weight vector $\mathbf{w}$ in question. To this end we need to find an upper bound on the average empirical risk $\mathbf{E}_{\mathbf{W}|\mathbf{W}\in\mathcal{B}}\left[\hat{R}[h_{\mathbf{W}}, \boldsymbol{z}]\right]$.

In the case that $h_{\mathbf{w}}$ is consistent with the training sample we know that $\Gamma_{(i)} > 0$ for all $i \in \{1, \ldots, m\}$. Corresponding to the margin distribution let us consider a sequence of caps $\mathcal{B}_{(i)}(\mathbf{w})$, with

$$\mathcal{B}_{(i)}(\mathbf{w}) \stackrel{\text{def}}{=} \left\{ \mathbf{v} \in \mathcal{W} \;\middle|\; \langle \mathbf{w}, \mathbf{v} \rangle > \sqrt{1 - \Gamma_{(i)}^2(\mathbf{w})} \right\}, \tag{4.9}$$

for $\Gamma_{(i)} < 0$ and $\mathcal{B}_{(i)}(\mathbf{w}) = \mathcal{W}$ else. Clearly, for any classifier $h_{\mathbf{v}}$ with $\mathbf{v} \in \mathcal{B}_{(i)}(\mathbf{w})$ we have

$$\hat{R}[h_{\mathbf{v}}, \boldsymbol{z}] \leq \frac{i-1}{m}, \tag{4.10}$$

by construction. In the case that $h_{\mathbf{w}}$ is inconsistent with the training sample we have $\hat{R}[h_{\mathbf{w}}, \boldsymbol{z}] > 0$. Then (4.10) holds for any classifier $h_{\mathbf{v}}$ with $\mathbf{v} \in \mathcal{B}_{(i)}(\mathbf{w})$ with i such that $\Gamma_i > 0$. In fact, in the latter case (4.10) is less tight than in the former because if $h_{\mathbf{w}}$ is consistent, then the empirical risk $\hat{R}[h_{\mathbf{v}}, \boldsymbol{z}]$ is a monotonically increasing function of the geodesic distance $\arccos(\langle \mathbf{v}, \mathbf{w} \rangle)$. In the case of $\hat{R}[h_{\mathbf{w}}, \boldsymbol{z}] > 0$ this is not in general the case because the empirical risk decreases at those training examples that were classified incorrectly by $h_{\mathbf{w}}$. Using (4.10) we can now upper-bound the average empirical risk by the following lemma.

Lemma 10 (Bound on average empirical error). *Given a ball $\mathcal{B}(\mathbf{w})$ corresponding to a normalised margin Γ according to (4.9) with $0 < \Gamma \leq \Gamma_{(i)}$, the average empirical risk of classifiers corresponding to weight vectors in $\mathcal{B}(\mathbf{w})$ is bounded by*

$$\mathbf{E}_{\mathbf{W}|\mathbf{W}\in\mathcal{B}(\mathbf{w})}\left[\hat{R}[h_{\mathbf{W}}, \boldsymbol{z}]\right] \leq \frac{i-1}{m}$$

Proof. Since $\Gamma \leq \Gamma_{(i)}$ we have $\mathcal{B}(\mathbf{w}) \subseteq \mathcal{B}_{(i)}(\mathbf{w})$. Thus it follows from (4.10) that for all $\mathbf{v} \in \mathcal{B}(\mathbf{w})$ we have

$$\hat{R}[h_{\mathbf{v}}, \boldsymbol{z}] \leq \frac{i-1}{m},$$

which must also hold for any expectation over $\mathcal{B}(\mathbf{w})$. □

Figure 4.5 illustrates this bound on the average empirical error. It is clear that the bound does not fully exploit the geometry of the model because the ball $\mathcal{B}(\mathbf{w})$ is not decomposed into error shells of decreasing empirical risk. Instead we just take the outer shell's empirical risk as an upper bound on the average.

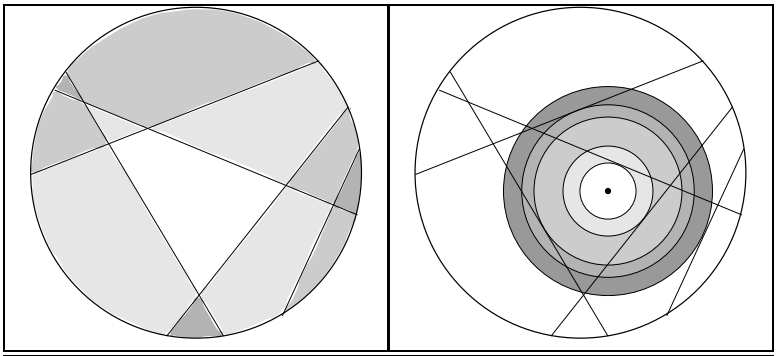


Figure 4.5.: Illustration of the approximation of regions of empirical error by spherical shells of empirical error. Shown on the left is a schematic view of the spherical weight space $\mathcal{W}$ with constraints given by great circles $\{\mathbf{v} \in \mathcal{W} \,|\, \langle \mathbf{v}, \mathbf{x}_i \rangle = 0\}$. The white area is version space $V(\boldsymbol{z})$ and the adjacent areas are shaded such that darker shade indicates higher empirical error. On the right, the approximation of regions of empirical error by shells is shown with respect to a particular weight vector $\mathbf{w} \in \mathcal{W}$ (black dot). Given a value Γ of the margin we can thus give a bound on the average empirical error within the associated ball $\mathcal{B}(\mathbf{w})$.

Since we aim at characterising a classifier $h_{\mathbf{w}}$ by its margin distribution, let us consider the following extension of the empirical risk as defined in Definition 9 to consider the normalised margin:

Definition 59 (Empirical margin risk). Given a linear classifier $h_{\mathbf{w}} \in \mathcal{H}_{\mathcal{K}}$ we define the empirical risk at normalised margin $\Gamma > 0$ as

$$\hat{R}_{\Gamma}[h_{\mathbf{w}}, \boldsymbol{z}] \stackrel{\text{def}}{=} \frac{1}{m} |\{z_i \in \boldsymbol{z} \,|\, \Gamma(\mathbf{w}, z_i) < \Gamma\}| \ .$$

Note that the fraction of training examples that fail to achieve a margin $\Gamma(\mathbf{w}, z_i) \geq \Gamma$ is the sum of the fraction of misclassified training examples, i.e., with $\Gamma(\mathbf{w}, z_i) \leq 0$, and those with $0 < \Gamma(\mathbf{w}, z_i) < \Gamma$.

4.4.2. PAC-Bayesian Margin Distribution Bound

Now we can provide a theorem based on Theorem 27 that bounds the prediction error of a linear classifier $h_{\mathbf{w}}$ in terms of its margin distribution $\boldsymbol{\Gamma}$.

Theorem 32 (PAC-Bayesian margin distribution bound). *Given a feature space $\mathcal{K} \subseteq \ell_2^n$ of dimensionality n, for all probability measures $\mathbf{P}_{\mathsf{Z}}$, for all $\delta \in (0, 1]$ with probability at least $1 - \delta$ over the random draw of a training sample $\boldsymbol{z}$ of size m, any classifier $h_{\mathbf{w}}$ with normalised margin distribution $\boldsymbol{\Gamma}$, for any $\Gamma > 0$, has prediction error*

$R[h_{\mathbf{w}}]$ bounded from above by

$$R[h_{\mathbf{w}}] < 2 \cdot \left(\hat{R}_{\Gamma}[h_{\mathbf{w}}, \boldsymbol{z}] + \sqrt{\frac{1}{2m}\left(d \log \frac{2}{\Gamma^2} + \log \frac{1}{\delta} + 2 \log m \right)} + \frac{1}{m} \right)$$

provided that $\Gamma \leq \Gamma_i$ and $d = \min(m, n)$.

Proof. The proof is a simple application of Theorem 27 with the complexity term bounded by

$$\log \frac{1}{\mathbf{P}_{\mathsf{H}}(\mathcal{B}(\mathbf{w}))} \leq d \log \frac{1}{1 - \sqrt{1 - \Gamma^2}} \leq d \log \frac{2}{\Gamma^2}$$

as in the proof of Theorem 29 and the average empirical risk bounded by

$$\mathbf{E}_{\mathbf{W}|\mathbf{W} \in \mathcal{B}} \left[\hat{R}[h_{\mathbf{W}}, \boldsymbol{z}] \right] \leq \hat{R}_{\Gamma}[h_{\mathbf{w}}, \boldsymbol{z}] \ .$$

□

It should be noted that the bound depends on a free parameter Γ that can be understood as balancing the two contributions to the bound. Increasing the value of Γ may lead to a higher empirical risk at margin Γ but leads to a decrease in the second term due to the greater volume of classifiers considered. Decreasing the value of Γ has the opposite effect. The best, i.e., lowest, bound values are thus obtained by optimising the bound over the choice of the parameter Γ. Note that to this end it is *not* necessary to stratify over the values of Γ because the PAC-Bayesian subset bound, Theorem 27 holds uniformly over all subsets of hypothesis space.

4.4.3. Discussion and Experiments

The PAC-Bayesian margin distribution bound, Theorem 32, has a very similar structure to a bound in Schapire et al. (1997) and Schapire et al. (1998) derived in the context of boosting. They consider classifiers $h_f = \operatorname{sign}(f(x))$ with $f \in \mathcal{F}$ based on the set of convex combinations

$$\mathcal{F} = \left\{ x \mapsto \sum_{h \in \mathcal{H}} \alpha_h h(x) \,\middle|\, \alpha_h \geq 0, \sum_h \alpha_h = 1 \right\}$$

of base classifiers $h : \mathcal{X} \to \mathcal{Y}$ and use the functional margin $\gamma = yf(x)$ as defined in Definition 43. Then their result states the following:

Theorem 33 (Boosting the margin). *For all probability measures $\mathbf{P}_{\mathsf{Z}}$, for all finite hypothesis spaces $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, and for all $\delta \in (0, 1]$, with probability at least $1 - \delta$ over the random choice of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$, every weighted average function $f \in \mathcal{F}$ satisfies the following bound for all $\gamma > 0$:*

$$R(h_f) \leq \hat{R}_{\gamma}[h_f, \boldsymbol{z}] + \mathcal{O}\left(\sqrt{\frac{1}{m}\left(\frac{\log m \log |\mathcal{H}|}{\gamma^2} + \log \frac{1}{\delta} \right)} \right) .$$

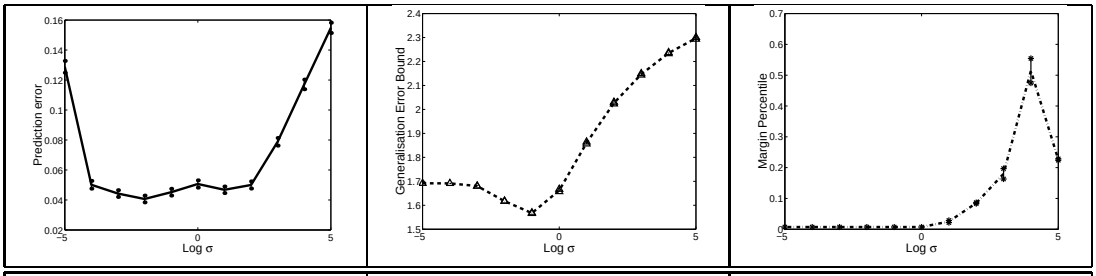


Figure 4.6.: Experimental results of training a support vector machine on the `thyroid` data set. **(a)** Prediction error as a function of $\log \sigma$ estimated on 100 different 60% (training data) : 40% (test data) splits of the whole data set. **(b)** Value of the margin distribution bound, Theorem 32, minimised over Γ as a function of $\log \sigma$. **(c)** Fraction of training examples failing to achieve a normalised margin Γ^* (which minimises the bound) as a function of $\log \sigma$.

For the case that $\mathcal{H}$ has infinite cardinality but finite VC dimension $d = \mathrm{VCdim}(\mathcal{H})$, they provide an alternative theorem with $\log m \log |\mathcal{H}|$ replaced by $d \log^2 \frac{m}{d}$. We see that the bounds have a similar structure in that they bound the prediction error of a hypothesis in terms of a complexity term that involves the inverse squared margin and the empirical fraction of examples that fail to attain that margin.

In order to illustrate the usefulness of the new bound, Theorem 32, for the purpose of model selection we performed experiments on the `thyroid` dataset from the UCI repository of machine learning (Merz and Murphy 1998). The key question is: Does the new margin percentile bound correctly mimic the behaviour of the (estimated) prediction error and can thus be used for the task of model selection?

The model selection task we considered was the selection of the kernel bandwidth σ of the RBF kernel

$$k(\vec{x}_1, \vec{x}_2) = \exp\left(-\frac{\|\vec{x}_1 - \vec{x}_2\|^2}{2\sigma^2}\right),$$

used by a support vector machine. The results are depicted in Figure 4.6. As expected the (estimated) prediction error shown in Figure 4.6 a) displays a U-shape. The regime of small values of σ corresponds to "over-fitting", the regime of large values of σ to "under-fitting". As can be seen in Figure 4.6 b), the margin distribution bound mimics the behaviour of the prediction error and displays a U-shape as well. The minimum is located at a value of σ that corresponds very well to the interval of low prediction error in Figure 4.6 a). In the regime of "over-fitting" the bound value saturates to a constant value, because the effective dimensionality for these low values of σ is effectively m and the margin does not change in this regime. In this regime the bound value is minimised for a value of $\Gamma^* = 0$ as can be seen from Figure 4.6 c): The margin distribution does not kick in yet. In the regime of "under-fitting" the bound values are slightly pessimistic but mimic the increasing prediction error for increasing values of σ. In this regime the number

of training examples failing to achieve the margin Γ^* drastically increases: Apparently, a large value of Γ decreases the complexity term by such a great amount that a large margin percentile is compensated for. In summary, these first results indicate that the new bound might be a promising candidate for model selection.

4.5. Conclusions and Future Work

4.5.1. Consequences for Support Vector Learning

Theorem 29 suggests the following learning algorithm: Given a version space $V(\boldsymbol{z})$ (through a given training set $\boldsymbol{z}$) find the classifier $\mathbf{w}$ that maximises $\Gamma(\mathbf{w}, \boldsymbol{z})$. This algorithm, however, is given by the SVM *only if* the training data in feature space $\mathcal{K}$ are normalised. We investigate the influence of such a normalisation on the prediction error in the feature space $\mathcal{K}$ of all monomials up to the p-th degree (well-known from handwritten digit recognition, see Vapnik (1995)). Since in this case $\mathcal{X} \subset \ell_2^n$ one possible kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is given by the polynomial kernel

$$\forall p \in \mathbb{N} \qquad k(\vec{x}_1, \vec{x}_2) = (\langle \vec{x}_1, \vec{x}_2 \rangle + 1)^p \ .$$

In order to normalise the feature vectors $\mathbf{x}_i$ it is sufficient to normalise the kernel. This is achieved by

$$k_{\mathrm{norm}}(\vec{x}_1, \vec{x}_2) \stackrel{\mathrm{def}}{=} \frac{k(\vec{x}_1, \vec{x}_2)}{\sqrt{k(\vec{x}_1, \vec{x}_1)\, k(\vec{x}_2, \vec{x}_2)}} ,$$

which under the normalised kernel leads to

$$\|\mathbf{x}\|^2 = k_{\mathrm{norm}}(\vec{x}, \vec{x}) = \frac{k(\vec{x}, \vec{x})}{\sqrt{k(\vec{x}, \vec{x})\, k(\vec{x}, \vec{x})}} = 1 .$$

Earlier experiments had shown that without normalisation too large values of p may lead to "over-fitting" (Vapnik 1995). We used the UCI (Blake and Merz 1998) data sets `thyroid` ($n = 5$, $m = 140$, $m_{\mathrm{test}} = 75$) and `sonar` ($n = 60$, $m = 124$, $m_{\mathrm{test}} = 60$) and plotted the estimated prediction error of SVM solutions (estimated over 100 different splits of the data set) as a function of p (see Figure 4.7). As suggested by Theorem 29 in almost all cases the normalisation improved the performance of the support vector machine solution at a statistically significant level. As a consequence, we recommend to always normalise data in feature space when training an SVM. Intuitively, it is only the *spatial direction* of both weight vector and feature vectors that determines the classification. Hence, the different lengths of feature vectors in the training sample should not enter the SVM optimisation problem.

4.5.2. Discussion

The PAC-Bayesian framework together with simple geometrical arguments yields the so far tightest margin bound for linear classifiers. The novelty of our approach to proving a PAC generalisation error bound in terms of the margin lies in the fundamentally different

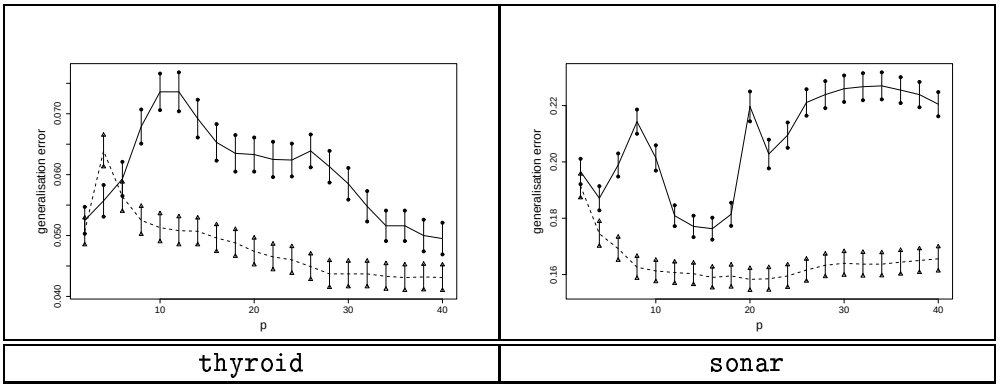


Figure 4.7.: Estimated prediction errors of classifiers learned by an SVM with (dashed line) and without (solid line) normalisation of the feature vectors $\mathbf{x}_i$. The error bars indicate one standard deviation over 100 random splits of the data sets. The left plot shows the result for the `thyroid` dataset ($n = 5$, $m = 140$, $m_{\text{test}} = 75$), the right plot for the `sonar` dataset.

reasoning applied: As pointed out in Chapter 3 classical PAC techniques use a ghost sample argument to consider equivalence classes of classifiers on a double sample and then aim at bounding the worst case number of equivalence classes in terms (or at the scale) of the margin observed (see, e.g. Anthony and Bartlett (1999) and Vapnik (1982)). Interestingly, the *pure* covering number bound (see Theorem 19) seems to be of the same order as our current result. The weakness of PAC margin bounds such as Theorem 22 mainly comes from the application of Alon's lemma (Alon et al. 1997), Lemma 2, when bounding the covering number at the observed margin scale by the fat shattering dimension.

In the current proof we avoid the usage of double samples and covering numbers at all. Instead we used pure volume ratio arguments and demonstrated that the margin $\Gamma(\mathbf{w}, \boldsymbol{z})$ has a natural interpretation of characterising a subset of classifiers in version space summarised by the classifier $h_{\mathbf{w}}$ under consideration. By instantiating a general result in the PAC-Bayesian framework with a special prior over classifiers we only needed to bound volume ratios in weight space. It is worthwhile mentioning that the quantity $\mathbf{P}_{\mathbf{W}}^{-1}(\mathcal{B}(\mathbf{w}))$ is related to the covering number of classifiers with a margin at least Γ via the packing number. Given a lower bound on the volume of a ball the number of disjunct balls $\mathcal{B}(\mathbf{w}_i)$ contained in $\mathcal{W}$ is naturally bounded as well.

An interesting question is the role of the number d in Theorem 29, i.e., the minimum of the number of dimension of feature space and the size of the training sample. Although it appears that this number limits the applicability of the current results to low dimensional feature spaces it seems possible to consider the effective dimensionality of the data in feature space. The cummerbound, Theorem 30, indicates how the particular shape of the input data in feature space could be taken into account in the PAC-Bayesian framework: By allowing more flexible shapes of the convex set C one could take advantage of the eigenvalue spectrum of the observed inner product matrix $\mathbf{K}$ (see Schölkopf et al. (1999)

for results that make use of the spectrum in a different way).

While the assumption of the existence of a version space makes the analysis more transparent the unrealisable case is clearly much closer to the practice of machine learning. The PAC-Bayesian margin distribution bound constitutes a result that gives a qualitative justification to learning algorithms such as SVM learning with soft margins. It is unclear, however, how the bound could be optimised directly by a learning algorithm because it appears to be difficult to minimise the number of examples that fail to attain a margin Γ and maximise the size of the margin Γ itself at the same time. On the other hand, the proof of the PAC-Bayesian margin distribution bound shows that the average empirical risk could, in principle, be upper-bounded much more tightly by more refined geometrical arguments (see Figure 4.5), indicating that an algorithms based on such more refined bounds might be preferable.

The proofs of the margin bounds presented reveals an important point already stressed in Shawe-Taylor et al. (1998) in their seminal work on luckiness: The margin as a characterisation of consistent classifiers is *one possible* prior belief in the data distribution underlying the training sample. Whenever this prior belief is not met by the training sample observed the margin bound will be trivial although still valid in the PAC sense. The arbitrariness of the uniform prior $\mathbf{P_W} = \mathbf{U_W}$ chosen in our proof should best be compared with the arbitrariness of a luckiness function when applying the main result of Shawe-Taylor et al. (1998). The proof technique presented can therefore be considered as conceptually similar to the luckiness framework.

5. PAC-Bayesian Sparseness Bounds

5.1. Occam´s Razor and Sparseness

It is often argued that in order to achieve good generalisation on should follow the following principle also known as *Occam´s Razor:*

> *Entia non sunt multiplicanda praeter necessitatem.*

Applied to the problem of data analysis we might rephrase it in the following way:

Principle of Occam´s razor: Select the simplest hypothesis that fits the data.

Apparently, this principle was first put forward by William of Occam (1285–1349) around the year 1320 (Mitchell 1997). Intuitively, the idea is that there exist less simple hypotheses than complex hypotheses. It is thus argued that if a simple hypothesis fits the data it is less likely to be a coincidence than if one of the many more complex hypotheses fits the data. While the principle appears to be quite generally accepted among most professionals of the inference business (e.g., scientists), its formalisation has proven difficult due to the ambiguity in representing hypotheses that makes it difficult to determine simplicity of a hypothesis in an absolute sense. Furthermore, the effectiveness of the principle in practice has been subject to debate (Domingos 1998; Webb 1996).

In this chapter we will be concerned with the concept of sparseness as an indicator for the simplicity of an hypothesis in the sense of Occam's razor. To this end we will distinguish two different types of sparseness: *feature sparseness* and *data sparseness*. Assuming a feature representation, feature sparseness simply refers to the number d of features that are actually used in classification. Defining the 0-norm $\|\mathbf{w}\|_0$ in $\mathbb{R}^n$ by

$$\|\mathbf{w}\|_0 \stackrel{\text{def}}{=} \sum_{i=1}^{n} w_i^0 \quad \text{with} \quad 0^0 \stackrel{\text{def}}{=} 0\,,$$

sparse linear classifiers $h_{\mathbf{w}}$ are characterised by weight vectors $\mathbf{w}$ with low 0-norm, $d = \|\mathbf{w}\|_0$.

Consider linear classifiers in a feature space $\mathcal{K}$ such as defined in Definition 19. Clearly, if we restrict ourselves to n features apriori we are dealing with a feature space $\mathcal{K}$ of dimensionality $\dim(\mathcal{K}) = n$. It is then easy to apply the VC bound, Theorem 15, with the VC dimension $d = \mathrm{VCdim}(\mathcal{H}_{\mathcal{K}})$ from Theorem 17 to obtain a bound on the prediction error of the corresponding linear classifier. However, we may not know in

advance which features a given learning algorithm for linear classifiers will select. In this case, we need to stratify over all the possible feature combinations a learning algorithm my select. Assuming that there are n possible features, there exist $\binom{n}{d}$ possible feature combinations with d features and $\sum_{d=1}^{n}\binom{n}{d}$ feature combinations in general. We can thus give the following aposteriori bound on the prediction error of feature-sparse linear classifier.

Theorem 34 (VC-bound for feature-sparse linear classifiers). *Given an input space $\mathcal{X} \subseteq \ell_2^n$, an output space $\mathcal{Y} = \{+1, -1\}$, for the hypothesis space $\mathcal{H}_\mathcal{K}$ of linear classifiers in $n = \dim(\mathcal{K})$ dimensions, for any probability measure $\mathbf{P}_\mathbf{Z}$, for every sample size $m > 0$, and for all $\delta \in (0, 1]$ we have with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that the prediction error $R[h_\mathbf{w}]$ of a consistent hypothesis $h_\mathbf{w}$ with $\|\mathbf{w}\|_0 = d$ is bounded by*

$$R[h] < \frac{2}{m}\left(d\log\frac{2em}{d} + n + \log\frac{2}{\delta}\right)$$

provided that $d \leq n \leq m$ and $m > \frac{2}{\varepsilon}$.

Proof. Combine the VC bound, Theorem 15, with the result that $\mathrm{VCdim}(\mathcal{H}_\mathcal{K}) = \dim(\mathcal{K})$, Theorem 17. Then apply the stratification lemma, Lemma 1, over all $\sum_{d=1}^{n}\binom{n}{d}$ possible feature combinations with

$$p_i = \frac{1}{\sum_{d=1}^{n}\binom{n}{d}}\,.$$

Finally, we use the bound

$$\sum_{d=1}^{n}\binom{n}{d} = \sum_{d=1}^{n}\binom{n}{d}1^d 1^{n-d} = (1+1)^n = 2^n < e^n\,,$$

to obtain the result of the theorem. □

While the chosen stratification scheme is rather crude here we see how the observed number of non-zero features d can reduce the bound value. To illustrate this effect consider the example of $d = 1$, $m = 1\,000\,000$ and $n = 10$. Comparing only complexity terms we have $d\log\frac{2em}{d} + n \approx 26$ versus $n\log\frac{2em}{n} \approx 132$, indicating that the aposteriori bound gives lower values.

Feature-sparse classifiers are of practical importance due to their easy interpretability. Many decision tree algorithms yield feature-sparse classifiers by considering one feature after the other until the empirical risk is reduced to zero (Mitchell 1997). However, in the remainder of this chapter, which is partly based on Graepel, Herbrich, and Shawe-Taylor (2000), we will be concerned with data-sparse classifiers in the context of compression schemes.

5.2. Compression Bounds for Single Classifiers

5.2.1. Compression and Reconstruction

In order to be able to bound the prediction error of classifiers in terms of their data-sparsity it is necessary to consider particular learning algorithms instead of just hypothesis spaces. In contrast to the previous results that constitute bounds on the prediction error uniformly over all hypotheses in $\mathcal{H}$ (PAC framework) or uniformly over all subsets of $\mathcal{H}$ (PAC-Bayesian framework) we are in the following concerned with bounds on the prediction error of classifiers that result from particular learning algorithms $\mathcal{A} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ (see Definition 7). Let us decompose a learning algorithm $\mathcal{A}$ into a compression scheme as follows (Littlestone and Warmuth 1986).

Definition 60 (Compression scheme). We define the set $I_{d,m} \subset \{1, \ldots, m\}^d$ as the set containing all index vectors of size $d \in \mathbb{N}$,

$$I_{d,m} \stackrel{\text{def}}{=} \left\{ (i_1, \ldots, i_d) \in \{1, \ldots, m\}^d \,|\, i_1 < i_2 < \cdots < i_d \right\} .$$

Given a training sample $\boldsymbol{z} \in \mathcal{Z}^m$ and an index vector $\mathbf{i} \in I_{d,m}$ let $\boldsymbol{z}_{\mathbf{i}}$ be the subsequence indexed by $\mathbf{i}$,

$$\boldsymbol{z}_{\mathbf{i}} \stackrel{\text{def}}{=} (z_{i_1}, \ldots, z_{i_d}) .$$

We call an algorithm $\mathcal{A} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ a compression scheme of size d if and only if there exists a pair $(\mathcal{C}_d, \mathcal{R}_d)$ of functions $\mathcal{C}_d : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \bigcup_{m=1}^{\infty} I_{d,m}$ (compression function) and $\mathcal{R}_d : \bigcup_{m=1}^{\infty} I_{d,m} \to \mathcal{H}$ (reconstruction function) such that we have for all training samples $\boldsymbol{z} \in \mathcal{Z}^m$,

$$\mathcal{A}(\boldsymbol{z}) = \mathcal{R}_d\left(\boldsymbol{z}_{\mathcal{C}_d(\boldsymbol{z})}\right) .$$

We call the compression scheme permutation invariant if and only if the reconstruction function $\mathcal{R}_d$ is permutation invariant for any training sample $\boldsymbol{z} \in \mathcal{Z}^m$, i.e., if for all permutation functions $\Pi : \{1, \ldots, d\}^d \to \{1, \ldots, d\}^d$

$$\mathcal{R}_d\left(\boldsymbol{z}_{\Pi(\mathbf{i})}\right) = \mathcal{R}_d\left(\boldsymbol{z}_{\mathbf{i}}\right) .$$

The definition of a compression scheme is easily illustrated by three algorithms we have been considering before, the perceptron, the SVM, and the KNN classifier.

1. The perceptron algorithm (see Chapter 2, Subsection 2.1.1) is a compression scheme that is not permutation invariant. Rerunning the perceptron algorithm on a training sample that consists only of those training examples that caused an update in the previous run leads to the same classifier as before. Permuting the order of the examples, however, may lead to a different classifier.

2. The support vector machine (see Chapter 2, Subsection 2.1.3) is a permutation invariant compression scheme. Rerunning the SVM only on the support vectors leads to the same classifier regardless of their order because the expansion coefficients in the optimal solution of the other training examples are zero and the objective function is invariant under permutation of the training examples.

3. The KNN classifier (see Chapter 2, Subsection 2.2.1) can be turned into a permutation invariant compression scheme as well: Delete those training examples that do not change the majority on any conceivable test input $x \in \mathcal{X}$ (consider Figure 2.7 for an illustration).

Based on the concept of compression let us consider PAC-style bounds on the prediction error of learning algorithms as described above.

5.2.2. The Realisable Case

Let us first consider the realisable learning scenario, i.e. for every training sample $\boldsymbol{z} \in \mathcal{Z}^m$ there exists a classifier $h \in \mathcal{H}$ such that $\hat{R}[h, \boldsymbol{z}] = 0$. Then we have the following compression bound (see Littlestone and Warmuth (1986) for early results on compression and Graepel et al. (2000) for a discussion of sparse linear classifiers).

Theorem 35 (PAC compression bound). *Let $\mathcal{A} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a compression scheme of size $d \leq \frac{m}{2}$. For any probability measure $\mathbf{P}_{\mathsf{Z}}$, and any $\delta \in (0, 1]$, with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$, if $\hat{R}[\mathcal{A}(\boldsymbol{z}), \boldsymbol{z}] = 0$ then*

$$R[\mathcal{A}(\boldsymbol{z})] \leq \frac{1}{m-d}\left(\log\binom{m}{d} d! + \log m + \log\frac{1}{\delta}\right),$$

and if $\mathcal{A}$ is a permutation invariant compression scheme, then

$$R[\mathcal{A}(\boldsymbol{z})] \leq \frac{1}{m-d}\left(\log\binom{m}{d} + \log m + \log\frac{1}{\delta}\right). \tag{5.1}$$

Proof. First we bound the probability

$$\begin{aligned}
& \mathbf{P}_{\mathsf{Z}^m}\left(\hat{R}[\mathcal{A}(\mathbf{Z}), \mathbf{Z}] = 0 \ \wedge \ R[\mathcal{A}(\mathbf{Z})] > \varepsilon\right) && (5.2)\\
\leq \ & \mathbf{P}_{\mathsf{Z}^m}\left(\exists \mathbf{i} \in I_{d,m} : \left(\hat{R}[\mathcal{R}_d(\mathbf{Z}_\mathbf{i}), \mathbf{Z}] = 0 \ \wedge \ R[\mathcal{R}_d(\mathbf{Z}_\mathbf{i})] > \varepsilon\right)\right) && (5.3)\\
\leq \ & \sum_{\mathbf{i} \in I_{d,m}} \mathbf{P}_{\mathsf{Z}^m}\left(\hat{R}[\mathcal{R}_d(\mathbf{Z}_\mathbf{i}), \mathbf{Z}] = 0 \ \wedge \ R[\mathcal{R}_d(\mathbf{Z}_\mathbf{i})] > \varepsilon\right). && (5.4)
\end{aligned}$$

The second line follows from the property $\mathcal{A}(\boldsymbol{z}) = \mathcal{R}_d\left(\boldsymbol{z}_{\mathcal{C}_d(\boldsymbol{z})}\right)$ and the fact that the event in the second line is implied by the event of the first line. The third line follows from the union bound, Theorem 57. Each summand in (5.4) is further bounded by

$$\mathbf{E}_{\mathsf{Z}^d}\left[\mathbf{P}_{\mathsf{Z}^{m-d}|\mathsf{Z}^d=\boldsymbol{z}_\mathbf{i}}\left(\hat{R}[\mathcal{R}_d(\boldsymbol{z}_\mathbf{i}), \mathbf{Z}] = 0 \ \wedge \ R[\mathcal{R}_d(\boldsymbol{z}_\mathbf{i})] > \varepsilon\right)\right] \tag{5.5}$$

where we used the fact that correct classification of the whole training sample $\boldsymbol{z}$ implies correct classification of any subset $\tilde{\boldsymbol{z}} \subseteq \boldsymbol{z}$ of it. Since the $m - d$ remaining training examples are drawn iid from $\mathbf{P}_{\mathsf{Z}}$ we can apply the binomial tail bound, Theorem 58, thus bounding the probability in (5.5) by $\exp(-(m-d)\varepsilon)$. The number $|I_{d,m}|$ of different

index vectors $\mathbf{i} \in I_{d,m}$ is given by $\binom{m}{d} d!$ for the case that $\mathcal{R}_d$ is not permutation invariant and $\binom{m}{d}$ in the case that $\mathcal{R}_d$ is permutation invariant. As a result, the probability in (5.4) is strictly less than $\binom{m}{d} d! \exp\left(-\left(m-d\right)\varepsilon\right)$ or $\binom{m}{d} \exp\left(-\left(m-d\right)\varepsilon\right)$, respectively.

Thus we have with probability at least $1-\delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}$ for all compression schemes $\mathcal{A}_i$ of size i that the proposition

$$\Upsilon_i\left(\boldsymbol{z}, m, \delta\right) \equiv \left(\hat{R}\left[\mathcal{A}_i\left(\mathbf{Z}\right), \mathbf{Z}\right] = 0 \;\Rightarrow\; R\left[\mathcal{A}_i\left(\mathbf{Z}\right)\right] \leq \frac{\log\binom{m}{i} i! + \log m + \log\frac{1}{\delta}}{m-i} \right)$$

holds true (with $\binom{m}{i} i!$ replaced by $\binom{m}{i}$ for the permutation invariant case). Finally, we apply the stratification lemma, Lemma 1, to the sequence of propositions Υ_i with $p_i = \frac{1}{m}$ for all $i \in \{1, \ldots, m\}$. □

The bound (5.1) in Theorem 35 is easily interpreted if we consider the bound on the binomial coefficient, $\binom{m}{d} < \left(\frac{em}{d}\right)^d$ and $\frac{1}{m-d} \leq \frac{2}{m}$ for $d \leq \frac{m}{2}$, thus obtaining

$$R\left[\mathcal{A}\left(\boldsymbol{z}\right)\right] \leq \frac{2}{m}\left(d \log\frac{em}{d} + \log m + \log\frac{1}{\delta}\right) . \tag{5.6}$$

This result should be compared to the simple VC bound, Theorem 15. Ignoring constants that are worse in the VC bound, these two bounds almost look alike. The (data-dependent) number of examples needed by the compression scheme replaces the VC dimension. Compression bounds can thus provide bounds on the prediction error of classifiers even if the classifier is chosen from an hypothesis space $\mathcal{H}$ of infinite VC dimension. The relation between VC bounds and compression schemes—motivated by equations such as (5.6)—is still not fully explored (Floyd and Warmuth 1995). We can observe an interesting analogy between the ghost sample argument in VC theory (see basic lemma, Lemma 13 in Section 3.1 of Chapter 3) and the use of the remaining $m-d$ examples from the sample. While the uniform convergence requirement in VC theory forces us to assume an extra ghost sample to be able to bound the true risk, the $m-d$ training examples serve the same purpose in the compression framework: To measure an empirical risk that serves to bound the true risk.

The second interesting observation about Theorem 35 is that the bound for a permutation invariant compression scheme is slightly better than its counterpart without this invariance. This difference can be understood from a coding point of view: It requires more bits to encode a sequence of indices (where order matters) as compared to a set of indices (where order does *not* matter).

In the proof of the PAC compression bound, Theorem 35, the stratification over the number d of training examples used was carried out using a uniform sequence $p_1 = \cdots = p_m = \frac{1}{m}$ of numbers indicating complete ignorance about the sparseness to be expected. In a PAC-Bayesian spirit, however, we may choose a more natural sequence of numbers that express our prior belief about the sparseness to be achieved. To this end we assume that given a training sample $\boldsymbol{z}$ the probability

$$p \equiv \mathbf{P}_{\mathsf{I}}\left(\mathsf{I} \in \mathcal{C}\left(\boldsymbol{z}\right)\right)$$

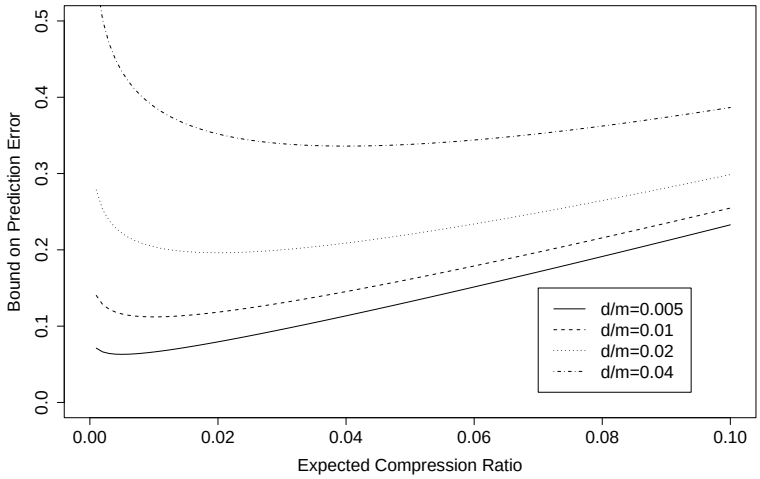


Figure 5.1.: Dependency of the PAC-Bayesian compression bound (5.7) on the expected value p and the observed value $\hat{p}$ of the compression coefficient. For increasing values $\hat{p} \stackrel{\text{def}}{=} \frac{d}{m}$ the optimal choice of the expected compression ratio p increases as indicated by the shifted minima of the family of curves.

that any given example $z_i \in \boldsymbol{z}$ will be in the compression sample $\boldsymbol{z}_{\mathcal{C}(\boldsymbol{z})}$ is constant and independent of z_i and $\boldsymbol{z}$. This induces a distribution over $d = |\mathcal{C}(\boldsymbol{z})|$ given for all $d \in \{1, \ldots, m\}$ by

$$p_d = \binom{m}{d} p^d (1-p)^{m-d} ,$$

for which we have $\sum_{i=1}^{d} p_d < 1$ as required for the stratification lemma, Lemma 1. The value p thus serves as an apriori belief about the value of the observed compression coefficient $\hat{p} \stackrel{\text{def}}{=} \frac{d}{m}$. This alternative sequence leads to the following bound for permutation invariant compression schemes,

$$R[\mathcal{A}(\boldsymbol{z})] \leq 2 \cdot \left(\hat{p} \log \frac{1}{p} + (1-\hat{p}) \log \frac{1}{1-p} + \frac{1}{m} \log \frac{1}{\delta} \right) . \tag{5.7}$$

Note that the term $\hat{p} \log \frac{1}{p} + (1-\hat{p}) \log \frac{1}{1-p}$ can be interpreted as the cross entropy between two random variables that are Bernoulli-distributed with success probabilities p and $\hat{p}$, respectively. For an illustration of how a suitably chosen value p of the expected compression ratio can decrease the bound value for a given value $\hat{p}$ of the compression ratio consider Figure 5.1.

5.2.3. The Unrealisable Case

The previous compression bound indicates an interesting relation between PAC/VC theory and data compression. Of course, data compression schemes come in two flavours,

lossy and non-lossy. Thus it comes as no surprise that we can derive bounds on the prediction error of compression schemes also for the unrealisable case with non-zero empirical risk (Graepel et al. 2000).

Theorem 36 (Lossy compression bound). *Let $\mathcal{A} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a compression scheme of size d. For any probability measure $\mathbf{P}_{\mathbf{Z}}$, and any $\delta \in (0,1]$, with probability at least $1-\delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ the prediction error of $\mathcal{A}(\boldsymbol{z})$ is bounded from above by*

$$R[\mathcal{A}(\boldsymbol{z})] \leq \frac{m}{m-d}\hat{R}[\mathcal{A}(\boldsymbol{z}), \boldsymbol{z}] + \sqrt{\frac{1}{2(m-d)}\left(\log\binom{m}{d}d! + 2\log m + \log\frac{1}{\delta}\right)},$$

and if $\mathcal{A}$ is a permutation invariant compression scheme, then by

$$R[\mathcal{A}(\boldsymbol{z})] \leq \frac{m}{m-d}\hat{R}[\mathcal{A}(\boldsymbol{z}), \boldsymbol{z}] + \sqrt{\frac{1}{2(m-d)}\left(\log\binom{m}{d} + 2\log m + \log\frac{1}{\delta}\right)}.$$

Proof. Fixing the number of training errors $q \in \{1, \ldots, m\}$ we bound—in analogy to the proof of Theorem 35—the probability

$$\mathbf{P}_{\mathbf{Z}^m}\left(\hat{R}[\mathcal{A}(\mathbf{Z}), \mathbf{Z}] \leq \frac{q}{m} \wedge R[\mathcal{A}(\mathbf{Z})] > \varepsilon\right) \tag{5.8}$$

$$\leq \sum_{\mathbf{i} \in I_{d,m}} \mathbf{P}_{\mathbf{Z}^m}\left(\hat{R}[\mathcal{R}_d(\mathbf{Z}_{\mathbf{i}}), \mathbf{Z}] \leq \frac{q}{m} \wedge R[\mathcal{R}_d(\mathbf{Z}_{\mathbf{i}})] > \varepsilon\right). \tag{5.9}$$

Again we have that $m \cdot \hat{R}[\mathcal{A}(\boldsymbol{z}), \boldsymbol{z}] \leq q$ implies $(m-d) \cdot \hat{R}[\mathcal{A}(\boldsymbol{z}), \boldsymbol{z}_{\bar{\mathbf{i}}}] \leq q$ for all $\mathbf{i} \in I_{m,d}$ and $\bar{\mathbf{i}} \stackrel{\text{def}}{=} \{1, \ldots, m\} \setminus \mathbf{i}$ leading to an upper bound,

$$\mathbf{E}_{\mathbf{Z}^d}\left[\mathbf{P}_{\mathbf{Z}^{m-d}|\mathbf{Z}^d=\boldsymbol{z}_{\mathbf{i}}}\left(\hat{R}[\mathcal{R}_d(\boldsymbol{z}_{\mathbf{i}}), \mathbf{Z}_{\bar{\mathbf{i}}}] \leq \frac{q}{m-d} \wedge R[\mathcal{R}_d(\boldsymbol{z}_{\mathbf{i}})] > \varepsilon\right)\right], \tag{5.10}$$

on the probability in (5.9). From Hoeffding´s inequality, Theorem 61, we know for a given sample $\boldsymbol{z}_{\mathbf{i}}$ that the probability in (5.10) is bounded by

$$\exp\left(-2(m-d)\left(\varepsilon - \frac{q}{m-d}\right)^2\right).$$

The number $|I_{d,m}|$ of different index vectors $\mathbf{i} \in I_{d,m}$ is again given by $\binom{m}{d}d!$ for the case that $\mathcal{R}_d$ is not permutation invariant and $\binom{m}{d}$ in the case that $\mathcal{R}_d$ is permutation invariant.

Thus we have with probability at least $1-\delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}$ for all compression schemes $\mathcal{A}$ of size i and maximal number of training errors q that the proposition

$$\Upsilon_{i,q}(\boldsymbol{z}, m, \delta) \equiv \left(\hat{R}[\mathcal{A}_i(\mathbf{Z}), \mathbf{Z}] \leq \frac{q}{m} \Rightarrow R[\mathcal{A}_i(\mathbf{Z})] \leq \frac{q}{m-i}\sqrt{\frac{\log\binom{m}{d} + \log\frac{1}{\delta}}{2(m-i)}}\right)$$

holds true (with $\binom{m}{i} i!$ replaced by $\binom{m}{i}$ for the permutation invariant case). Finally, we apply the stratification lemma, Lemma 1, to the sequence of propositions $\Upsilon_{i,q}$ with $p_{i,q} = \frac{1}{m^2}$ for all $(i,q) \in \{1,\ldots,m\}^2$. □

5.3. PAC-Bayesian Sparseness Bounds

In the proofs of the compression results, Theorem 35 and Theorem 36, we made use of the fact that $m-d$ of the m training examples had not been used for constructing the classifier and could thus be used to bound the true risk with high probability. In this section, we will make use of similar arguments in order to deal with data-dependent hypothesis spaces such as those parameterised by $\boldsymbol{\alpha}$ in kernel classifiers

$$h_{(\boldsymbol{\alpha},\boldsymbol{x})}(x) = \operatorname{sign}\left(\sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x})\right), \tag{5.11}$$

as described in Definition 19. Obviously, typical VC results cannot be applied to this type of data-dependent hypothesis class, because the hypothesis class is not fixed in advance. Hence, its complexity cannot be determined before learning. In this section we will proceed similarly to Section 3.3: First we prove what David McAllester called a PAC-Bayesian "folk" theorem (McAllester 1998), then we proceed with a PAC-Bayesian subset bound.

5.3.1. The PAC-Bayesian Folk Theorem for Data-Dependent Hypotheses

Suppose instead of a PAC-Bayesian prior $\mathbf{P_H}$ over a fixed hypothesis space we define a prior $\mathbf{P_A}$ over the sequence $\boldsymbol{\alpha}$ of expansion coefficients α_i in (5.11). Relying on a sparse representation with $\|\boldsymbol{\alpha}\|_0 < m$ we can then prove the following theorem in analogy to Theorem 24:

Theorem 37 (Data-dependent PAC-Bayesian bound for single classifiers). *Consider the hypothesis space $\mathcal{H}_\mathcal{K}$. For any prior probability distribution $\mathbf{P_A}$ on a finite subset $A \subset \mathbb{R}^m$ satisfying $\mathbf{P_A}(\boldsymbol{\alpha}) > 0$ for all $\boldsymbol{\alpha} \in A$, for any probability measure $\mathbf{P_Z}$, for every sample size $m > 0$, and for all $\delta \in (0,1]$ we have with probability at least $1-\delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that for any hypothesis $h_{(\boldsymbol{\alpha},\boldsymbol{x})} \in \mathcal{H}_\mathcal{K}$ the prediction error $R\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}\right]$ is bounded by*

$$R\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}\right] \leq \frac{1}{m - \|\boldsymbol{\alpha}\|_0}\left(\log\frac{1}{\mathbf{P_A}(\boldsymbol{\alpha})} + \log\frac{1}{\delta}\right).$$

Proof. First we show that the proposition $\Upsilon_{\boldsymbol{\alpha}}(\boldsymbol{z}, \|\boldsymbol{\alpha}\|_0)$,

$$\Upsilon_{\boldsymbol{\alpha}}(\boldsymbol{z}, \|\boldsymbol{\alpha}\|_0) \equiv \left(\hat{R}\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}, \boldsymbol{z}\right] = 0 \;\Rightarrow\; R\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}\right] \leq \frac{\log\frac{1}{\delta}}{m - \|\boldsymbol{\alpha}\|_0}\right),$$

holds for all $\boldsymbol{\alpha} \in A$ with probability at least $1 - \delta$ over the random draw of $\boldsymbol{z}$. Let $\mathbf{i}_\alpha \in I_{\|\boldsymbol{\alpha}\|_0, m}$ be the index vector with entries at which $\alpha_i \neq 0$. Then we have for all $\boldsymbol{\alpha} \in A$ that

$$\begin{aligned} & \mathbf{P}_{\mathbf{Z}^m}\left(\hat{R}\left[h_{(\boldsymbol{\alpha},\mathbf{X})}, \mathbf{Z}\right] = 0 \ \wedge \ R\left[h_{(\boldsymbol{\alpha},\mathbf{X})}\right] > \varepsilon\right) \\ \leq \ & \mathbf{P}_{\mathbf{Z}^m}\left(\hat{R}\left[h_{(\boldsymbol{\alpha},\mathbf{X}_{\mathbf{i}_\alpha})}, \mathbf{Z}_{\bar{\mathbf{i}}_\alpha}\right] = 0 \ \wedge \ R\left[h_{(\boldsymbol{\alpha},\mathbf{X}_{\mathbf{i}_\alpha})}\right] > \varepsilon\right) \\ \leq \ & \mathbf{E}_{\mathbf{Z}^d}\left[\mathbf{P}_{\mathbf{Z}^{m-d}|\mathbf{Z}^d = \boldsymbol{z}_{\bar{\mathbf{i}}_\alpha}}\left(\hat{R}\left[h_{(\boldsymbol{\alpha},\mathbf{X}_{\mathbf{i}_\alpha})}, \mathbf{Z}_{\bar{\mathbf{i}}_\alpha}\right] = 0 \ \wedge \ R\left[h_{(\boldsymbol{\alpha},\mathbf{X}_{\mathbf{i}_\alpha})}\right] > \varepsilon\right)\right] \\ < \ & (1-\varepsilon)^{m-\|\boldsymbol{\alpha}\|_0} \leq \exp\left(-\varepsilon\left(m - \|\boldsymbol{\alpha}\|_0\right)\right) . \end{aligned}$$

The key is that the classifier $h_{(\boldsymbol{\alpha},\boldsymbol{x})}$ does not change over the random draw of the $m - \|\boldsymbol{\alpha}\|_0$ samples in $\boldsymbol{z}_{\bar{\mathbf{i}}_\alpha}$ not used in its expansion. Finally, apply the stratification lemma, Lemma 1, to the proposition $\Upsilon_{\boldsymbol{\alpha}}(\boldsymbol{z}, \|\boldsymbol{\alpha}\|_0)$ with $p_{\boldsymbol{\alpha}} \equiv \mathbf{P}_{\mathbf{A}}(\boldsymbol{\alpha})$. □

Obviously, replacing the binomial tail bound with Hoeffding´s inequality allows us to derive a result for the unrealisable case with non-zero empirical risk—equivalent to Theorem 26. This bound then reads

$$R\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}\right] \leq \frac{m}{m - \|\boldsymbol{\alpha}\|_0}\hat{R}\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}\right] + \sqrt{\frac{1}{2\left(m - \|\boldsymbol{\alpha}\|_0\right)}\left(\log\frac{1}{\mathbf{P}_{\mathbf{A}}(\boldsymbol{\alpha})} + \log m + \log\frac{1}{\delta}\right)} .$$

Example 2 (1-norm soft margin perceptron). Suppose we run the dual perceptron algorithm with box-constraints $0 \leq \alpha_i \leq C$ as described in Remark 1 and obtain a classifier $h_{(\boldsymbol{\alpha},\boldsymbol{x})}$ with $\hat{R}\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}\right]$ and d non-zero coefficients α_i. Defining a prior $\mathbf{P}_{\mathbf{A}}$ over the set $\{\boldsymbol{\alpha} \in \{-C, \ldots, 0, \ldots, C\}^m\}$ as

$$\mathbf{P}_{\mathbf{A}}(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \frac{1}{m\binom{m}{d}(2C)^d} \tag{5.12}$$

we get for the bound

$$R\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}\right] \leq \frac{m}{m-d}\hat{R}\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}\right] + \sqrt{\frac{1}{2(m-d)}\left(\log\binom{m}{d} + d\log 2C + 2\log m + \log\frac{1}{\delta}\right)},$$

which yields lower values than the compression bound, Theorem 36, for non-permutation invariant compression schemes if $2C < d$ as can be seen by bounding $\log d!$ by $d\log d$ in Theorem 36 using Stirling´s formula, Theorem 63.

5.3.2. The PAC-Bayesian Subset Bound for Data-Dependent Hypotheses

Let us now consider a PAC-Bayesian subset bound like Theorem 25 for the data-dependent hypothesis space of kernel classifiers (5.11). In order to make the result more digestible we consider it for a fixed number d of non-zero coefficients.

Theorem 38 (PAC-Bayesian subset bound for data-dependent hypotheses). *Consider the hypothesis space $\mathcal{H}_K$. For any prior probability distribution $\mathbf{P}_{\mathbf{A}}$, for any probability measure $\mathbf{P}_{\mathbf{Z}}$, for any $d \in \{1, \ldots, m\}$, for every sample size $m > 0$, and for all $\delta \in (0, 1]$ we have with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ that for any subset $A \subseteq V(\boldsymbol{z})$ with constant sparseness $\|\boldsymbol{\alpha}\|_0 = d$, zero empirical risk $\hat{R}\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}(\boldsymbol{z})\right] = 0$ for all $\boldsymbol{\alpha} \in A$, and $\mathbf{P}_{\mathbf{A}}(A) > 0$, the average prediction error $\mathbf{E}_{\mathbf{A}|\mathbf{A}\in A}\left[R\left[h_{(\mathbf{A},\boldsymbol{x})}\right]\right]$ is bounded by*

$$\mathbf{E}_{\mathbf{A}|\mathbf{A}\in A}\left[R\left[h_{(\mathbf{A},\boldsymbol{x})}\right]\right] \leq \frac{\log \frac{1}{\mathbf{P}_{\mathbf{A}}(A)} + 2\log m + \log \frac{1}{\delta} + 1}{m - d}.$$

Proof. We decompose the expectation at some point $\varepsilon \in \mathbb{R}$ by

$$\mathbf{E}_{\mathbf{A}|\mathbf{A}\in A}\left[R\left[h_{(\mathbf{A},\boldsymbol{x})}\right]\right] \leq \varepsilon \cdot \mathbf{P}_{\mathbf{A}|\mathbf{A}\in A}\left(R\left[h_{(\mathbf{A},\boldsymbol{x})}\right] \leq \varepsilon\right) + 1 \cdot \mathbf{P}_{\mathbf{A}|\mathbf{A}\in A}\left(R\left[h_{(\mathbf{A},\boldsymbol{x})}\right] > \varepsilon\right). \quad (5.13)$$

As in the proof of Theorem 37 we have that for all $\boldsymbol{\alpha} \in A$ and for all $\delta \in (0, 1]$,

$$\mathbf{P}_{\mathbf{Z}^m|\mathbf{A}=\boldsymbol{\alpha}}\left(\Upsilon_{\boldsymbol{\alpha}}(\mathbf{Z}, d, \delta)\right) \geq 1 - \delta,$$

where the proposition $\Upsilon_{\boldsymbol{\alpha}}(\boldsymbol{z}, d, \delta)$ is given by

$$\Upsilon_{\boldsymbol{\alpha}}(\boldsymbol{z}, d, \delta) \equiv \left(\hat{R}\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}, \boldsymbol{z}\right] = 0 \;\Rightarrow\; R\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}\right] \leq \frac{\log \frac{1}{\delta}}{m - d}\right).$$

By the quantifier reversal lemma, Lemma 3, this implies that for all $\beta \in (0, 1)$ with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z}$ for all $\gamma \in (0, 1]$,

$$\begin{aligned}
\mathbf{P}_{\mathbf{A}|\mathbf{Z}^m=\boldsymbol{z}}\left(\neg\Upsilon_{\mathbf{A}}\left(\mathbf{Z}, d, (\gamma\beta\delta)^{\frac{1}{1-\beta}}\right)\right) &< \gamma \\
\mathbf{P}_{\mathbf{A}|\mathbf{Z}^m=\boldsymbol{z}}\left(\hat{R}\left[h_{(\mathbf{A},\boldsymbol{x})}, \boldsymbol{z}\right] = 0 \;\wedge\; R\left[h_{(\mathbf{A},\boldsymbol{x})}\right] > \varepsilon(\gamma, \beta)\right) &< \gamma
\end{aligned}$$

with

$$\varepsilon(\gamma, \beta) \stackrel{\text{def}}{=} \frac{\log \frac{1}{\delta\gamma\beta}}{(1 - \beta)(m - d)}.$$

Since the distribution over $\boldsymbol{\alpha}$ is by assumption a prior and thus independent of the data, we have $\mathbf{P}_{\mathbf{A}|\mathbf{Z}^m=\boldsymbol{z}} = \mathbf{P}_{\mathbf{A}}$ and hence

$$\begin{aligned}
\mathbf{P}_{\mathbf{A}|\mathbf{A}\in A}\left[R\left[h_{(\mathbf{A},\boldsymbol{x})}\right] > \varepsilon(\gamma, \beta)\right] &= \frac{\mathbf{P}_{\mathbf{A}}\left(\mathbf{A} \in A \;\wedge\; R\left[h_{(\mathbf{A},\boldsymbol{x})}\right] > \varepsilon(\gamma, \beta)\right)}{\mathbf{P}_{\mathbf{A}}(A)} \\
&\leq \frac{\gamma}{\mathbf{P}_{\mathbf{A}}(A)},
\end{aligned}$$

because by assumption $\boldsymbol{\alpha} \in A$ implies $\hat{R}\left[h_{(\boldsymbol{\alpha},\boldsymbol{x})}, \boldsymbol{z}\right] = 0$. Now choosing $\gamma = \frac{\mathbf{P}_{\mathbf{A}}(A)}{m}$ and $\beta = \frac{1}{m}$ we obtain from (5.13)

$$\begin{aligned}
\mathbf{E}_{\mathbf{A}|\mathbf{A}\in A}\left[R\left[h_{(\mathbf{A},\boldsymbol{x})}\right]\right] &\leq \varepsilon(\gamma, \beta) \cdot \left(1 - \frac{\gamma}{\mathbf{P}_{\mathbf{A}}(A)}\right) + \frac{\gamma}{\mathbf{P}_{\mathbf{A}}(A)} \\
&= \frac{\log \frac{1}{\mathbf{P}_{\mathbf{A}}(A)} + 2\log m + \log \frac{1}{\delta}}{m - d} + \frac{1}{m}
\end{aligned}$$

because by assumption we have for all $\boldsymbol{\alpha} \in A$ that $\|\boldsymbol{\alpha}\|_0 = d$. Bounding $\frac{1}{m}$ by $\frac{1}{m-d}$ completes the proof. □

Again, replacing the binomial tail bound with Hoeffding´s inequality allows us to derive a result for the unrealisable case with non-zero empirical risk—equivalent to Theorem 27.

Example 3 (1-norm soft margin permutational perceptron sampling). Continuing the discussion of Example 2 with the same prior distribution (5.12) consider the following procedure: Learn a 1-norm soft margin perceptron with box constraints $0 \leq \alpha_i \leq C$ for all $i \in \{1, \ldots, m\}$ and assume linear separability. Permute the compression sample $\boldsymbol{z}_{\mathbf{i}_{\mathrm{sv}}}$ and retrain to obtain an ensemble A of N different coefficient vectors $\boldsymbol{\alpha}_j$. Then the PAC-Bayesian subset bound for data dependent hypotheses, Theorem 38 bounds the average prediction error of the ensemble of classifiers $\left\{h_{(\boldsymbol{\alpha},\boldsymbol{x})} \,|\, \boldsymbol{\alpha} \in A\right\}$ corresponding to the ensemble A of coefficient vectors.

5.4. Conclusions

In this chapter various bounds on the prediction error of sparse classifiers were derived based on the idea of data compression. Essentially, the results rely on the fact that a data-sparse classifier $h_{(\boldsymbol{\alpha},\boldsymbol{x})}$ with d non-zero coefficients is independent of the random draw of $m - d$ training examples, which—if classified with low or zero empirical risk by $h_{(\boldsymbol{\alpha},\boldsymbol{x})}$—serve to ensure a low prediction error with high probability. In particular, the PAC-Bayesian results of McAllester (1998) were extended to data-dependent hypotheses that are represented as a linear expansion in terms of training inputs. The results are thus applicable to the class of kernel classifiers as defined in Definition 35, ranging from support vector to KNN classifiers. Empirically, the bounds given yield rather low bound values and have low constants in comparison to VC bounds or bounds based on the observed margin. In summary, they are widely applicable and rather tight.

From a Bayesian perspective the formulation of a prior over expansion coefficients $\boldsymbol{\alpha}$ that parameterise data-dependent hypotheses appears rather unusual. No contradiction, however, arises because the prior cannot be used to "cheat" by adjusting it in such a way as to manipulate the bound values. The reason is that the expansion (5.11) does not contain the labels y_i. Instead the prior serves to incorporate apriori knowledge about the representation of classifiers in terms of training inputs.

Of course, there exist many non-sparse classifiers with a low prediction error as well. It remains a challenging open question how we can formulate and prove PAC-Bayesian bounds for data-dependent hypotheses that are dense, i.e., that have few or no non-zero coefficients.

6. The Structure of Version Space

6.1. An Egalitarian Bound

The bounds on the prediction error so far made statements that held either uniformly over $\mathcal{H}$ (PAC style, e.g., Theorem 12) or uniformly over all measurable subsets of $\mathcal{H}$ (PAC-Bayesian style, e.g., Theorem 25). In this section we will be interested in what we call *egalitarian* bounds that hold for almost all hypotheses in subset of $\mathcal{H}$. The motivation for this kind of bound stems from the widespread belief that only specific classifiers have good prediction properties while all the others are necessarily worse. One example of this situation is the dominance of the large margin solution given by the support vector machine. While this solution certainly has its merits we try to demonstrate by the egalitarian bounds that in fact most of the classifiers in version space have a small prediction error.

6.1.1. Egalitarian Bound for Consistent Classifiers

The PAC-Bayesian subset bound, Theorem 25 bounds the average prediction error over a subset $U \in V(\boldsymbol{z})$ of version space in terms of the prior measure $\mathbf{P}_{\mathsf{H}}(U)$ assigned apriori to that subset. Given a bound on the expectation of a random variable we can use Markov's inequality, Theorem 59, to bound the value of the random variable with high probability. The following theorem in a sense compares the prediction error of consistent classifiers with the average prediction error of the Gibbs classification strategy based on $V(\boldsymbol{z})$.

Theorem 39 (Egalitarian Bound). *For all probability measures $\mathbf{P}_{\mathsf{Z}}$, for all $m > 0$, for all $\delta \in (0, 1]$ we have with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathsf{Z}^m$, that for all $\eta > 1$, at least a fraction of $1 - \frac{1}{\eta}$ of the classifiers in version space $V(\boldsymbol{z})$ have prediction error less than*

$$\eta \cdot \varepsilon_{\text{Gibbs}}\left(m, \mathbf{U}_{\mathsf{H}}, V(\boldsymbol{z}), \delta\right) ,$$

where $\mathbf{U}_{\mathsf{H}}$ is the uniform measure over $\mathcal{H}$.

Proof. The proof is a simple application of Markov's inequality, Theorem 59, along with the instantiation of $\mathbf{P}_{\mathsf{H}}$ by the uniform measure $\mathbf{U}_{\mathsf{H}}$. For $\boldsymbol{z}$ fixed Markov's inequality says that for all $\eta > 1$

$$\mathbf{P}_{\mathsf{H}|\mathsf{H}\in V(\boldsymbol{z})}\left(R[\mathsf{H}] \geq \eta \cdot \mathbf{E}_{\mathsf{H}|\mathsf{H}\in V(\boldsymbol{z})}[R[\mathsf{H}]]\right) < \frac{1}{\eta},$$

because the generalisation error $R : \mathcal{H} \to [0,1]$ being a functional over hypotheses is a positive random variable. From Theorem 25 we have that

$$\mathbf{P}_{\mathbf{Z}^m}\left(\mathbf{E}_{\mathsf{H}|\mathsf{H}\in V(\mathbf{Z})}\left[R\left[\mathsf{H}\right]\right] \leq \varepsilon_{\text{Gibbs}}\left(m, \mathbf{P}_{\mathsf{H}}, \mathbf{Z}, \delta\right)\right) \geq 1 - \delta\,.$$

It follows that

$$\mathbf{P}_{\mathbf{Z}^m}\left(\forall \eta > 1 : \mathbf{P}_{\mathsf{H}|\mathsf{H}\in V(\mathbf{Z})}\left(R\left[\mathsf{H}\right] < \eta \cdot \varepsilon_{\text{Gibbs}}\left(m, \mathbf{U}_{\mathsf{H}}, \mathbf{Z}, \delta\right)\right) \geq 1 - \frac{1}{\eta}\right) \geq 1 - \delta\,.$$

□

An interesting feature of this prediction error bound is that it holds true regardless of any property of the single classifiers considered. In fact, the only quantity that drives the bound on the prediction error is the volume of version space which is a *property of the model $\mathcal{H}$ and the data $\boldsymbol{z}$* but not of single classifiers h.

Consider the result of Theorem 39 with $\eta = 2$. In this case we know that with high probability ($\geq 1 - \delta$) the prediction error of at least half of the classifiers in version space $V(\boldsymbol{z})$ are bounded by at most twice the generalisation error of the Gibbs classification strategy. Remembering the fact given by Lemma 4 that the prediction error $R[\text{Bayes}_U]$ of the Bayes classification strategy is bounded by twice the prediction error $R[\text{Gibbs}_U]$ of the Gibbs classification strategy,

$$R\left[\text{Bayes}_U\right] \leq 2 \cdot R\left[\text{Gibbs}_U\right]\,,$$

we see that at least half of the classifiers in version space have a prediction error bound lower than the Bayes classification strategy!

We can also compare the result with the PAC-Bayesian bound on the prediction error of linear classifiers in terms of margins, Theorem 29,

$$\frac{2}{m}\left(d \log \frac{1}{1 - \sqrt{1 - \Gamma^2\left(\mathbf{w}, \boldsymbol{z}\right)}} + \log \frac{(me)^2}{\delta}\right) \geq 2 \cdot \varepsilon_{\text{Gibbs}}\left(m, \mathbf{U}_{\mathbf{W}}, V\left(\boldsymbol{z}\right), \delta\right)\,. \tag{6.1}$$

For normalised training inputs, $\|\mathbf{x}_i\| = 1$ for all $i \in \{1, \ldots, m\}$, the weight vector $\mathbf{w}_{\text{svm}}$ found by the support vector machine minimises the bound value on the left-hand side of (6.1). Thus we see that whenever we have a low bound on the prediction error of the SVM solution we know that at least half of the consistent classifiers have prediction error bounded by the same (or even a lower) bound value. The practical difficulty in exploiting these solutions, however, is that they keep changing over the random draw of the training sample and only the large margin classifier is able to *witness* its small generalisation error by an easy-to-determine quantity—its margin. The result suggests one should not be too dismissive of algorithms such as the perceptron learning algorithm (see Subsection 2.1.1) that merely ensure a consistent hypothesis $h \in V(\boldsymbol{z})$. It appears that the choice of the model $\mathcal{H}$ is more important than the choice of the learning procedure *within a fixed model* $\mathcal{H}$ given that a classifier with low empirical risk is found. For kernel-based classifiers this means the choice of the kernel (see also Chapter 2).

6.1.2. Agnostic Egalitarian Bound

While the simple egalitarian bound, Theorem 39, gives us a rough idea about the distribution of prediction error over classifiers in version space we can also consider a more general case. To this end we present another variant of the quantifier reversal lemma, Lemma 3. This lemma was communicated to us by David McAllester who informed us that the proof was due to Avrim Blum.

Lemma 11 (Simple quantifier reversal lemma). *Let* X *and* Y *be random variables with associated probability spaces* $(\mathcal{X}, \mathfrak{X}, \mathbf{P}_{\mathsf{X}})$ *and* $(\mathcal{Y}, \mathfrak{Y}, \mathbf{P}_{\mathsf{Y}})$*, respectively. Let* $\Upsilon : \mathcal{X} \times \mathcal{Y} \times (0,1] \to \{\text{true}, \text{false}\}$ *be any measurable formula. If for all* $x \in \mathcal{X}$ *and* $\delta \in (0,1]$*,*

$$\mathbf{P}_{\mathsf{Y}}\left(\Upsilon\left(x, \mathsf{Y}, \delta\right)\right) \geq 1 - \delta,$$

then for any $\delta > 0$ *and* $\gamma \in (0,1]$ *we have*

$$\mathbf{P}_{\mathsf{Y}}\left(\mathbf{P}_{\mathsf{X}}\left(\Upsilon\left(\mathsf{X}, \mathsf{Y}, \gamma\delta\right)\right) \geq 1 - \gamma\right) \geq 1 - \delta\,.$$

In contrast to the earlier quantifier reversal lemma, Lemma 3, the simple quantifier reversal lemma, Lemma 11, does not hold uniformly for all values of γ. Although we do not need this aspect here it is clear that we can use the union bound over a finite subset of $(0,1]$ to remedy this defect as in the following corollary.

Corollary 2 (Uniform quantifier reversal). *Given the conditions of Lemma 11 and a fixed value* $m \in \mathbb{N}$*, we have*

$$\mathbf{P}_{\mathsf{Y}}\left(\forall k \in \{1, \ldots, m\} : \mathbf{P}_{\mathsf{X}}\left(\Upsilon\left(\mathsf{X}, \mathsf{Y}, \frac{\delta \exp\left(-k\right)}{m}\right)\right) \geq 1 - \exp\left(-k\right)\right) \geq 1 - \delta\,.$$

Proof. Define a sequence $p_i \equiv \exp\left(-i\right)$ for $i \in \{1, \ldots, m\}$ and apply the stratification lemma, Lemma 1. □

Remark 2 (Alternative proof to PAC-Bayesian subset bounds). Interestingly, this corollary provides an alternative proof to Theorems 25 and 27. Simply choose

$$k = \left\lceil \log \frac{m}{\mathbf{P}_{\mathsf{H}}\left(U\right)} \right\rceil$$

such that

$$k \leq \log \frac{1}{\mathbf{P}_{\mathsf{H}}\left(U\right)} + \log m + 1\,.$$

Let us now turn back to our original goal of deriving another egalitarian bound by considering the average deviation between empirical and true risk.

Theorem 40 (Agnostic egalitarian bound). *For all hypothesis spaces* $\mathcal{H}$*, for all probability measures* $\mathbf{P}_{\mathsf{H}}$ *and* $\mathbf{P}_{\mathsf{Z}}$*, for all* $m > 0$*, for all* $\delta \in (0,1]$ *with probability at least* $1 - \delta$ *over the random draw of the training sample* $\boldsymbol{z} \in \mathcal{Z}^m$ *of size* m*, we have*

$$\mathbf{P}_{\mathsf{H}}\left(R\left[\mathsf{H}\right] - \hat{R}\left[\mathsf{H}, \mathbf{Z}\right] \leq \sqrt{\frac{1}{2m}\left(\log m + \log \frac{1}{\delta}\right)}\right) \geq \frac{m-1}{m}\,.$$

Proof. From Hoeffding´s inequality, Theorem 61, we have that for all $h \in \mathcal{H}$,

$$\mathbf{P}_{\mathbf{Z}^m}\left(R[\mathsf{H}] - \hat{R}[\mathsf{H}, \mathbf{Z}] \leq \sqrt{\frac{1}{2m}\log\frac{1}{\delta}}\right) \geq 1 - \delta\,.$$

Now applying Lemma 11 with $x \equiv h$, $y \equiv \boldsymbol{z}$, and $\gamma \equiv \frac{1}{m}$ yields the result. □

Interestingly, the result applies to all hypothesis spaces, i.e. it applies to $\mathcal{H} = \mathcal{Y}^{\mathcal{X}}$ as well. Thus even for small sample size m we have for the majority of classifiers with high probability a convergence of the empirical risk to the true risk at a fast rate $\mathcal{O}(\sqrt{\log(m)/m})$ independent of any complexity properties of the underlying hypothesis space. It follows that if we find for a given training sample $\boldsymbol{z}$ that many classifiers h have a low empirical risk then we can conclude that for most of these hypotheses the prediction error is indeed small. Of course this result does not contradict the results from VC theory, because the assertion is not made uniformly over hypothesis space, but only with high probability over $\mathbf{P}_{\mathsf{H}}$.

6.2. From Margin to Sparseness

The previous egalitarian results are general PAC-Bayesian results without any explicit reference to a particular hypothesis space or algorithm. In this section we will explore the relation between PAC bounds and mistake bounds for online learning algorithms. In particular, we will reconsider the perceptron algorithm and derive a PAC bound for the resulting classifiers from a mistake bound involving the margin a support vector machine would achieve on the same training data. We will argue that a large potential margin is sufficient to obtain good bounds on the prediction error of all the classifiers found by the perceptron on permuted training sequences $\boldsymbol{z}_{\mathsf{j}}$.

6.2.1. Online-Learning and Mistake Bounds

In order to be able to discuss the perceptron convergence theorem and the relation between mistake bounds and PAC bounds in more depth let us introduce formally the notion of an online algorithm (Littlestone 1988). To this end we need the definition of an update function $\mathcal{U}$.

Definition 61 (Update function). Consider an input space $\mathcal{X}$, a finite output space $\mathcal{Y}$ and an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$. A function $\mathcal{U} : \mathcal{Z} \times \mathcal{H} \to \mathcal{H}$ is called an *update function* on $\mathcal{H}$ and will be denoted by $\mathcal{U}(z, h)$ for a given training example z and a current hypothesis h.

Given the definition of an update function we can proceed and define an online algorithm.

Definition 62 (Online learning algorithm). Consider an input space $\mathcal{X}$, a finite output space $\mathcal{Y}$, an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, an update function $\mathcal{U} : \mathcal{Z} \times \mathcal{H} \to \mathcal{H}$

thereon, and an initial hypothesis $h_0 \in \mathcal{H}$. An *online learning algorithm* is a function $\mathcal{A} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \times \bigcup_{i=1}^{\infty} \{1, \ldots, m\}^i \times \mathcal{H} \to \mathcal{H}$ that takes a training sample $\boldsymbol{z} \in \mathcal{Z}^m$, a *training sequence* $\mathbf{j} \in \bigcup_{i=1}^{\infty} \{1, \ldots, m\}^i$, and an *initial hypothesis* $h_0 \in \mathcal{H}$, and produces the final hypothesis $\mathcal{A}_{\mathcal{U}} \stackrel{\text{def}}{=} h_{|\mathbf{j}|}$ of the $|\mathbf{j}|$-fold recursion of the update function $\mathcal{U}$,

$$h_i = \mathcal{U}\left(z_{j_i}, h_{i-1}\right) ,$$

Mistake-driven learning algorithms are a particular class of online algorithms that only change their current hypothesis if it causes an error on the current training example.

Definition 63 (Mistake-driven learning algorithm). An online algorithm $\mathcal{A}_{\mathcal{U}}$ is called *mistake-driven* if the update function satisfies for all $x \in \mathcal{X}$, for all $y \in \mathcal{Y}$, and for all $h \in \mathcal{H}$ that

$$y = h\left(x\right) \quad \Rightarrow \quad \mathcal{U}\left(\left(x, y\right), h\right) = h\,.$$

In the PAC framework we focus on the error of the final hypothesis $\mathcal{A}\left(\boldsymbol{z}\right)$ an algorithm produces after considering the whole training sample $\boldsymbol{z}$. In the analysis of online-algorithms one takes a slightly different view: The number of updates until convergence is considered the quantity of interest.

Definition 64 (Mistake bound). Consider an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ and a training sample $\boldsymbol{z}$ labelled by an hypothesis $h \in \mathcal{H}$. Denote by $\tilde{\mathbf{j}} \in \left\{1, \ldots, |\tilde{\mathbf{j}}|\right\}^{|\tilde{\mathbf{j}}|}$, with $\tilde{\mathbf{j}} \subseteq \mathbf{j}$ the sequence of mistakes, i.e., the subsequence of $\mathbf{j}$ containing the indices $i \in \{1, \ldots, m\}$ for which $h_i \neq \mathcal{U}\left(z_{j_i}, h_{i-1}\right)$. We call a function $M_{\mathcal{A}_{\mathcal{U}}} : \mathcal{Z}^m \to \mathbb{N}$ a *mistake bound* for the online algorithm $\mathcal{A}_{\mathcal{U}}$ if it bounds the number $|\tilde{\mathbf{j}}|$ of mistakes $\mathcal{A}_{\mathcal{U}}$ makes on $\boldsymbol{z} \in \mathcal{Z}^m$,

$$|\tilde{\mathbf{j}}| \leq M_{\mathcal{A}_{\mathcal{U}}} ,$$

for any ordering $\mathbf{j} \in \bigcup_{i=1}^{\infty} \{1, \ldots, m\}^i$.

In a sense, this is a very practical measure of error assuming that a learning machine is learning "on the job".

6.2.2. From Online to Batch Learning

Interestingly, we can relate any mistake bound for a mistake-driven algorithm to a PAC style bound on the prediction error:

Theorem 41 (Mistake bound to PAC bound). *Consider a mistake-driven online learning algorithm $\mathcal{A}_{\mathcal{U}}$ for $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ with a mistake bound $M_{\mathcal{A}_{\mathcal{U}}} : \mathcal{Z}^m \to \mathbb{N}$. For any probability measure $\mathbf{P}_{\mathsf{Z}}$ and any $\delta \in (0, 1]$, with probability at least $1 - \delta$ over the random draw of the training sample $\boldsymbol{z} \in \mathcal{Z}^m$ we have that the expected risk $R[\mathcal{A}_{\mathcal{U}}(\boldsymbol{z})]$ of the hypothesis $\mathcal{A}_{\mathcal{U}}(\boldsymbol{z})$ is bounded from above by*

$$R\left[\mathcal{A}_{\mathcal{U}}\left(\boldsymbol{z}\right)\right] \leq \frac{2}{m}\left(M_{\mathcal{A}}\left(\boldsymbol{z}\right)\left(1 + \log m\right) + \log m + \log\frac{1}{\delta}\right) .$$

Proof. The proof is based on the fact that a mistake-driven algorithm constitutes a (non permutation invariant) compression scheme. Assume we run $\mathcal{A}_{\mathcal{U}}$ twice on the same training sample $\boldsymbol{z}$ and training sequence $\mathbf{j}$. From the first run we obtain the sequence of mistakes $\tilde{\mathbf{j}}$. Thus we have for the compression function $\mathcal{C}_{|\tilde{\mathbf{j}}|}$,

$$\mathcal{C}_{|\tilde{\mathbf{j}}|}\left(\boldsymbol{z}_{\mathbf{j}}\right) = \tilde{\mathbf{j}} \,.$$

Running $\mathcal{A}_{\mathcal{U}}$ only on $\boldsymbol{z}_{\tilde{\mathbf{j}}}$ then leads to the same hypothesis as before,

$$\mathcal{A}_{\mathcal{U}}\left(\boldsymbol{z}, \mathbf{j}\right) = \mathcal{A}_{\mathcal{U}}\left(\boldsymbol{z}, \tilde{\mathbf{j}}\right)$$

showing that the reconstruction function $\mathcal{R}_{|\tilde{\mathbf{j}}|}$ is given by the algorithm $\mathcal{A}_{\mathcal{U}}$ itself,

$$\mathcal{R}_{|\tilde{\mathbf{j}}|}\left(\boldsymbol{z}_{\mathcal{C}_{|\tilde{\mathbf{j}}|}(\boldsymbol{z})}\right) = \mathcal{A}_{\mathcal{U}}\left(\boldsymbol{z}, \mathcal{C}_{|\tilde{\mathbf{j}}|}\left(\boldsymbol{z}_{\mathbf{j}}\right)\right) \,.$$

The compression scheme is in general not permutation invariant because $\mathcal{A}_{\mathcal{U}}$ and hence $\mathcal{R}_{|\tilde{\mathbf{j}}|}$ is not. We can thus apply Theorem 35, where we bound d from above by $M_{\mathcal{A}_{\mathcal{U}}}$ and use $\frac{1}{m-d} \leq \frac{2}{m}$ for all $d \leq \frac{m}{2}$, which gives

$$R\left[\mathcal{A}\left(\boldsymbol{z}\right)\right] \leq \frac{2}{m}\left(\log\binom{m}{M_{\mathcal{A}}} M_{\mathcal{A}}! + \log m + \log\frac{1}{\delta}\right) \,.$$

Finally, using $\log M_{\mathcal{A}_{\mathcal{U}}}! \leq M_{\mathcal{A}_{\mathcal{U}}} \log M_{\mathcal{A}_{\mathcal{U}}}$ (see Theorem 63 in Appendix B) and $\log\binom{m}{M_{\mathcal{A}_{\mathcal{U}}}} < M_{\mathcal{A}_{\mathcal{U}}} \cdot \left(\log em - \log M_{\mathcal{A}_{\mathcal{U}}}\right)$ (see Theorem 65 in Appendix B) we obtain the bound in the theorem. □

6.2.3. Novikoff's Perceptron Mistake Bound

As an application to the previous result, Theorem 41, let us reconsider the perceptron convergence theorem, in fact a special case of Theorem 2.

Theorem 42 (Perceptron Convergence). *Given a training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$ such that the data radius is $\varsigma \equiv \varsigma\left(\boldsymbol{x}\right)$ and such that there exists a weight vector $\mathbf{w}^* \in \mathcal{K}$ with margin $\gamma^* \equiv \gamma\left(\mathbf{w}^*, \boldsymbol{z}\right)$ the perceptron algorithm terminates after at most M update steps with*

$$M = \left(\frac{\varsigma}{\gamma^*}\right)^2 \,.$$

Proof. The theorem is a special case of Theorem 2 with enforced margin $\rho^2 = 0$. □

Clearly, the perceptron algorithm is a mistake-driven online learning algorithm in the sense of Definitions 62 and 63, and Theorem 42 is a mistake bound in the sense of Definition 64. Considering the bound more closely we observe that in the case of normalised training inputs, $\|\mathbf{x}_i\|^2 = 1$ for all $i \in \{1, \ldots, m\}$, the bound is minimised. We thus have the following lemma.

Lemma 12 (Normalised margin). *Given a training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$ we have that*

$$\left(\frac{\gamma(\mathbf{w},\boldsymbol{z})}{\varsigma(\boldsymbol{x})}\right)^2 \leq \Gamma^2(\mathbf{w},\boldsymbol{z}) \ .$$

Proof. From the definitions of $\gamma(\mathbf{w},\boldsymbol{z})$, $\varsigma(\boldsymbol{x})$, and $\Gamma^2(\mathbf{w},\boldsymbol{z})$ we have □

$$\begin{aligned}\left(\frac{\gamma(\mathbf{w},\boldsymbol{z})}{\varsigma(\boldsymbol{x})}\right)^2 &= \left(\frac{1}{\|\mathbf{w}\|}\cdot\frac{\min_{(x_i,y_i)\in\boldsymbol{z}} y_i\langle\mathbf{x}_i,\mathbf{w}\rangle}{\max_{x_i\in\boldsymbol{x}}\|\mathbf{x}_i\|}\right)^2 \\ &\leq \left(\min_{(x_i,y_i)\in\boldsymbol{z}}\frac{y_i\langle\mathbf{w},\mathbf{x}_i\rangle}{\|\mathbf{w}\|\,\|\mathbf{x}_i\|}\right)^2 \\ &= \Gamma^2(\mathbf{w},\boldsymbol{z}) \ .\end{aligned}$$

This result indicates again that the normalised margin $\Gamma^2(\mathbf{w},\boldsymbol{z})$ is the crucial quantity to be considered just as found before in the PAC-Bayesian margin bound, Theorem 29.

6.2.4. A Bound Based on the Potential Margin

Given a mistake bound we can obtain a corresponding PAC bound on the prediction error of the final hypothesis found by the online algorithm by Theorem 41.

Theorem 43 (Potential margin bound). *Consider the hypothesis space $\mathcal{H}_\mathcal{K}$. For any measure $\mathbf{P}_\mathsf{Z}$ over the random draw of the training sample $\boldsymbol{z}$ of size m, if there exists a linear classifier $h_{\mathbf{w}^*}\in\mathcal{H}_\mathcal{K}$ with normalised margin*

$$\Gamma^* \stackrel{\text{def}}{=} \Gamma(\mathbf{w}^*,\boldsymbol{z}) > 0 \,,$$

then for all $\delta\in(0,1]$ we have with probability at least $1-\delta$ that the prediction error $R[\mathcal{A}_{\text{perc}}(\boldsymbol{z})]$ of the classifier $\mathcal{A}_{\text{perc}}(\boldsymbol{z})$ found by the perceptron algorithm on normalised training inputs $\mathbf{x}_i$ with $\|\mathbf{x}_i\|^2=1$ for all $i\in\{1,\ldots,m\}$, is bounded by

$$R[\mathcal{A}_{\text{perc}}(\boldsymbol{z})] \leq \frac{2}{m}\left(\left(\frac{1}{\Gamma^*}\right)^2(1+\log m)+\log m+\log\frac{1}{\delta}\right) \ .$$

Proof. Apply Lemma 12 to Theorem 42 and plug the resulting mistake bound,

$$M_{\mathcal{A}_{\text{perc}}} = \frac{1}{\Gamma^2(\mathbf{w}^*,\boldsymbol{z})} \,,$$

into Theorem 41. □

The intriguing feature of this result is that the *mere existence* of a large margin classifier $h_{\mathbf{w}^*}$ is sufficient to guarantee a small prediction error for the solution $h_\mathbf{w} = \mathcal{A}_{\text{perc}}(\boldsymbol{z})$ of the dual perceptron although its attained margin $\Gamma(\mathbf{w},\boldsymbol{z})$ is likely to be much

smaller than $\Gamma(\mathbf{w}^*, \boldsymbol{z})$. It has long been argued that the attained margin $\Gamma(\mathbf{w}, \boldsymbol{z})$ *itself* is the crucial quantity controlling the generalisation error of $h_{\mathbf{w}}$. In light of this new result *if there exists a consistent classifier $h_{\mathbf{w}^*}$ with large margin we know that there also exists at least one classifier $h_{\mathbf{w}}$ with high sparsity* that can be found efficiently using the dual perceptron algorithm. In fact, whenever the SVM appears to be theoretically justified by a large observed margin, every solution found by the dual perceptron algorithm has a small guaranteed prediction error—mostly bounded more tightly than current bounds on the generalisation error of SVMs. Note that for a given training sample $\boldsymbol{z}$ it is not unlikely that by permutation of $\boldsymbol{z}$ there exist many different consistent sparse classifiers $h_{\mathbf{w}}$.

6.2.5. Impact on the Foundations of Support Vector Machines

Support vector machines owe their popularity mainly to their theoretical justification in learning theory. In particular, two arguments have been put forward to single out the solutions found by SVMs (Vapnik 1998, p. 139):

> SVMs (optimal hyperplanes) can generalise because
>
> 1. the expectation of the data compression is large.
> 2. the expectation of the margin is large.

The second reason is often justified by margin results (see Vapnik (1998) and the results in Subsection 3.1.3 and Chapter 4) that bound the generalisation of a classifier $h_{\mathbf{w}}$ in terms of its *own attained* margin $\gamma(\mathbf{w}, \boldsymbol{z})$ or $\Gamma(\mathbf{w}, \boldsymbol{z})$. Considering training inputs $\mathbf{x}_i$ normalised in feature space, $\|\mathbf{x}_i\|^2 = 1$ for all $i \in \{1, \ldots, m\}$, we can compare the PAC-Bayesian margin bound with the potential margin bound.

PAC-Bayesian Margin Bound (See Theorem 29)

$$\mathcal{O}\left(\frac{d}{m}\log\frac{1}{\Gamma^2}\right)$$

Potential Margin Bound (See Theorem 43)

$$\mathcal{O}\left(\frac{\log m}{m}\frac{1}{\Gamma^{*2}}\right)$$

- The PAC-Bayesian margin bound is a bound on the prediction error of a classifier $h_{\mathbf{w}}$ with attained margin $\Gamma \equiv \Gamma(\mathbf{w}, \boldsymbol{z})$, while the potential margin bound is a bound on the classifier learned by the perceptron algorithm if there exists a classifier $h_{\mathbf{w}^*}$ attaining a normalised margin $\Gamma^* \equiv \Gamma(\mathbf{w}^*, \boldsymbol{z})$.
- The PAC-Bayesian margin bound depends on the dimensionality $d = \min(m, n)$ of feature space while the potential margin bound is dimension free. As a consequence it can be argued that the potential margin bound defies the *curse of dimensionality.*

Digit	0	1	2	3	4	5	6	7	8	9
KP	0.2	0.2	0.4	0.4	0.4	0.4	0.4	0.5	0.6	0.7
$\|\boldsymbol{\alpha}\|_0$	740	643	1168	1512	1078	1277	823	1103	1856	1920
M	844	843	1345	1811	1222	1497	960	1323	2326	2367
SVM	0.2	0.1	0.4	0.4	0.4	0.5	0.3	0.4	0.5	0.6
$\|\boldsymbol{\alpha}\|_0$	1379	989	1958	1900	1224	2024	1527	2064	2332	2765

Table 6.1.: Results of the kernel perceptron (KP) and the SVM on NIST handwritten digit data. The kernel used was $k(\vec{x}_1, \vec{x}_2) = (\langle \vec{x}_1, \vec{x}_2 \rangle + 1)^4$ and $m = 60\,000$. For both algorithms we give the estimated error (in %) and the number $\|\boldsymbol{\alpha}\|_0$ of non-zero coefficients α_i. For the perceptron algorithm we also provide the number M of mistakes until convergence.

- The PAC-Bayesian margin bound depends logarithmically on Γ^{-2} while the potential margin bound depends linearly on Γ^{*-2}.

With regard to sparseness, it has been confirmed experimentally that SVMs find solutions that are sparse in the expansion coefficients α_i. However, there cannot exist any distribution-free guarantee that the number of support vectors will in fact be small. Consider a distribution $\mathbf{P_Z}$ concentrated for each class on one of two parallel lines with support in the unit ball. Suppose that their mutual distance is $\sqrt{2}$. Then the number of support vectors equals the size of the training sample whereas the dual perceptron never uses more than two points by Theorem 42. One could argue that it is the number of *essential support vectors* (Vapnik 1998) that characterises the data compression of an SVM (which would also have been two in our example). Their determination, however, involves a combinatorial optimisation problem and can thus hardly be performed in practical applications. In contrast, Theorem 42 gives an explicit bound on the sparseness in terms of the achievable margin Γ^*. Furthermore, experimental results on the NIST datasets show that the sparseness of solutions found by the dual perceptron algorithm is consistently (and sometimes by a factor of two) greater than that of the SVM solution (see Freund and Schapire (1999), Table 3 and Table 6.1).

6.3. Conclusions

At first glance the egalitarian bound seems to imply that we are hopeless in the search for *the* quantity controlling generalisation error (bounds) because it gives a good generalisation error bound for a huge number of consistent classifiers $h \in V(\boldsymbol{z})$ not referring to any property other than the choice of the model $\mathcal{H}$. This result, however, comes at no surprise taking into account the paradigm of PAC learning (see Definition 39). Although we are only interested in the performance of the one classifier h we learned using a fixed learning algorithm $\mathcal{A} : \cup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H}$ we claim to need guarantees on the generalisation error that hold *uniformly* over the whole hypothesis space $\mathcal{H}$ or version space $V(\boldsymbol{z})$,

respectively. This is much too demanding and can therefore only lead to bounds that indicate whether we have chosen an appropriate model or not. A more promising approach is to investigate the question of prediction error bounds for specific algorithms. In fact, the proof of Theorem 43 uses a compression bound which requires the specification of the algorithm $\mathcal{A}$ in advance, i.e. the bounds apply only to a small subset of learning algorithms (so-called *compression schemes*). A related idea is studied in Bousquet and Elisseeff (2000) where the VC dimension as a complexity measure of an hypothesis space $\mathcal{H}$ is replaced by the *robustness* of the learning algorithm $\mathcal{A}$ used. The robustness of an algorithm $\mathcal{A}$ measures by how much the training error of the learned classifier $\mathcal{A}(\boldsymbol{z})$ changes when adding one additional observation,

$$\max_{z=(x,y)} \left| \hat{R}\left[\mathcal{A}(\boldsymbol{z}), \boldsymbol{z}\right] - \hat{R}\left[\mathcal{A}(\boldsymbol{z} \cup z), \boldsymbol{z} \cup z\right] \right| .$$

According to intuition, whenever a learning algorithm is very robust we have small deviation between generalisation and training error for the classifiers learned although the VC dimension of the hypothesis class used might have been infinite.

Finally, it is worthwhile noticing that the results presented in this chapter do not deny the importance of *inductive principles*. Although we know that within a good model $\mathcal{H}$ there are many classifiers with a provable small generalisation error, there might exist procedures (the maximum margin algorithm is one such procedure) that single out classifiers with small generalisation error bounds for most random draws of the training sample $\boldsymbol{z}$. A potential candidate for formulating such inductive principles is the *luckiness framework* (Shawe-Taylor et al. 1998).

7. Bayesian Learning in Kernel Space

This chapter takes up the discussion of Bayesian learning in Section 3.2 and introduces two ways of implementing Bayesian learning w.r.t. the class of kernel classifiers. The first section treats what we call Bayesian transduction. This is really a direct application of Bayesian classification strategy as introduced in Section 3.2 to kernel classifiers. Bayesian transduction implements a voting scheme among classifiers and thus uses an ensemble of classifiers instead of picking one particular classifier. In general, the classification strategy of transduction does not correspond to any one single classifier. However, in the second section we suggest the centre of mass of version space as an approximation to the Bayes point, which is the best single classifier $h \in \mathcal{H}_K$ to approximate the classification by Bayesian transduction. The section on transduction, Section 7.1, is based on Graepel, Herbrich, and Obermayer (1999), the section on Bayes point machines, Section 7.2, is based on Herbrich, Graepel, and Campbell (1999a), Herbrich, Graepel, and Campbell (1999b), Herbrich, Graepel, and Campbell (2000), and Herbrich, Graepel, and Campbell (2001).

7.1. Bayesian Transduction

The notion of *transduction* was introduced by Vapnik (see, e.g. Vapnik (1998)) as a principle of inference that stands in contrast to *induction*. While induction aims at inferring a general dependency from specific observations, transduction aims at making specific predictions given specific observations. Vapnik argues that *when solving a given problem one should avoid solving a more general problem as an intermediate step*. The reasoning behind this principle is that in order to solve the more general task resources may be wasted or compromises may have to be made which would not have been necessary for the solution of the problem at hand. A direct application of this common-sense principle reduces the more general problem of inferring a functional dependency on the whole of input space to the problem of estimating the values of a function at given points (working sample), a paradigm referred to as *transductive inference*. More formally, given a probability measure $\mathbf{P_Z}$ on the space of data $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} = \mathcal{X} \times \{-1, +1\}$, a *training sample* $\boldsymbol{z}$ is generated i.i.d. according to $\mathbf{P_{XY}}$. Additional $\tilde{m}$ data points $\tilde{\boldsymbol{x}} = \{\tilde{x}_{m+1}, \ldots, \tilde{x}_{m+\tilde{m}}\}$ are drawn according to $\mathbf{P_X}$: the *working sample*. The goal is to label the objects of the working sample $\tilde{\boldsymbol{x}}$ using a fixed hypothesis space $\mathcal{H}$ of classifiers $h : \mathcal{X} \to \mathcal{Y}$ so as to minimise a predefined loss. In contrast, *inductive inference*, aims at choosing a *single* classifier $h \in \mathcal{H}$ best suited to capture the dependency expressed by the unknown $\mathbf{P_Z}$.

Definition 65 (Transductive classification algorithm). Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$ we call a function

$$\mathcal{A}_{\mathrm{td}} : \left(\bigcup_{m=1}^{\infty} \mathcal{Z}^m \right) \times \left(\bigcup_{\tilde{m}=1}^{\infty} \mathcal{X}^{\tilde{m}} \right) \to \mathcal{Y}^{\tilde{m}}$$

that takes a training sample $\boldsymbol{z}$ and a working sample $\tilde{\boldsymbol{x}}$ and returns a labelling $\tilde{\boldsymbol{y}}$ of $\tilde{\boldsymbol{x}}$ a *transductive classification algorithm.*

Obviously, if we have a transductive classification algorithm $\mathcal{A}_{\mathrm{td}}$ for a fixed hypothesis space $\mathcal{H}$ that assigns to each working sample $\tilde{\boldsymbol{x}}$ a sequence of labels given the training sample $\boldsymbol{z}$ and the hypothesis space $\mathcal{H}$, we can define a function $h_{\mathrm{td}} : \mathcal{X} \to \mathcal{Y}$ by

$$h_{\mathrm{td}}(x) \stackrel{\mathrm{def}}{=} \mathcal{A}_{\mathrm{td}}(\boldsymbol{z}, \{x\})$$

as a result of the transductive algorithm. There are two crucial differences to induction, however: i) $\mathcal{A}_{\mathrm{td}}$ is not restricted to select a single decision function $h \in \mathcal{H}$ for each x, ii) a transduction algorithm can give performance guarantees on particular labellings instead of functions. In practical applications this difference may be of great importance.

After all, in risk sensitive applications (medical diagnosis, financial and critical control applications) it often matters to know how *confident* we are about a given prediction. In this case a general confidence measure of the classifier w.r.t. the whole input distribution would not provide the desired warranty at all. Note that for linear classifiers some guarantee can be obtained by the margin (Shawe-Taylor 1996) which in Chapter 9 will be demonstrated to be too coarse a confidence measure. The idea of transduction was put forward in Vapnik (1982) where also first algorithmic ideas can be found. Later Wu et al. (1999) suggested an algorithm for transduction based on large margin trees and Gammerman et al. (1998) highlighted the need for confidence measures in transduction.

7.1.1. Labelling the Working Sample

In continuation of the discussion in Section 3.2 let us assume that the posterior predictive distribution over the working sample $\tilde{\boldsymbol{x}}$, see also (3.9), is given by

$$\mathbf{P}_{\tilde{\mathsf{Y}}^{\tilde{m}} | \tilde{\mathsf{X}}^{\tilde{m}} = \tilde{\boldsymbol{x}}, \mathsf{Z}^m = \boldsymbol{z}}(\tilde{\boldsymbol{y}}) = \mathbf{E}_{\mathsf{H} | \mathsf{Z}^m = \boldsymbol{z}} \left[\mathbf{P}_{\tilde{\mathsf{Y}}^{\tilde{m}} | \tilde{\mathsf{X}}^{\tilde{m}} = \tilde{\boldsymbol{x}}, \mathsf{H} = h}(\tilde{\boldsymbol{y}}) \right] .$$

Given the measure $\mathbf{P}_{\tilde{\mathsf{Y}}^{\tilde{m}} | \tilde{\mathsf{X}}^{\tilde{m}} = \tilde{\boldsymbol{x}}, \mathsf{Z}^m = \boldsymbol{z}}$ over labellings, in order to arrive at a risk minimal decision w.r.t. the labelling we need to define a loss function $l_{\mathrm{td}} : \tilde{\mathcal{Y}}^{\tilde{m}} \times \tilde{\mathcal{Y}}^{\tilde{m}} \to \mathbb{R}^+$ between labellings and minimise its expectation.

Definition 66 (Bayesian transduction). Given an input space $\mathcal{X}$, the output space $\mathcal{Y} = \{+1, -1\}$, a training sample $\boldsymbol{z} \in \mathcal{Z}^m$, a working sample $\tilde{\boldsymbol{x}} \in \mathcal{X}^{\tilde{m}}$, a predictive posterior $\mathbf{P}_{\tilde{\mathsf{Y}}^{\tilde{m}} | \tilde{\mathsf{X}}^{\tilde{m}} = \tilde{\boldsymbol{x}}, \mathsf{Z}^m = \boldsymbol{z}}$, and a loss $l_{\mathrm{td}} : \mathcal{Y}^{\tilde{m}} \times \mathcal{Y}^{\tilde{m}} \to \mathbb{R}^+$ between labellings, we define a Bayesian transductive classification algorithm by

$$\mathcal{A}_{\mathrm{btd}}(\tilde{\boldsymbol{x}}, \boldsymbol{z}) \stackrel{\mathrm{def}}{=} \operatorname*{argmin}_{\tilde{\boldsymbol{y}} \in \mathcal{Y}^{\tilde{m}}} R_{\mathrm{btd}}(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{x}}, \boldsymbol{z}) ,$$

where

$$R_{\mathrm{td}}\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{x}}, \boldsymbol{z}\right) \stackrel{\mathrm{def}}{=} \mathsf{E}_{\tilde{\mathsf{Y}}^{\tilde{m}} \mid \tilde{\mathsf{X}}^{\tilde{m}}=\tilde{\boldsymbol{x}}, \mathsf{Z}^{m}=\boldsymbol{z}}\left[l_{\mathrm{td}}\left(\tilde{\boldsymbol{y}}, \tilde{\mathbf{Y}}\right)\right] .$$

Let us consider two scenarios of Bayesian transduction, depending on two different notions of loss on labellings:

Definition 67 (Joint and additive label loss for transduction). Given the output space $\mathcal{Y} = \{+1, -1\}$ and a sequence $\tilde{\boldsymbol{y}} \in \mathcal{Y}^{\tilde{m}}$ of labels, $\tilde{\boldsymbol{y}} \stackrel{\mathrm{def}}{=} (\tilde{y}_1, \ldots \tilde{y}_{\tilde{m}})$, we define the *joint label loss,* $l_{\mathrm{td}_1} : \mathcal{Y}^{\tilde{m}} \times \mathcal{Y}^{\tilde{m}} \to \mathbb{R}^+$, by

$$l_{\mathrm{td}_1}\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{y}}'\right) \stackrel{\mathrm{def}}{=} \mathsf{I}_{\tilde{\boldsymbol{y}} \neq \tilde{\boldsymbol{y}}'}$$

and the additive label loss, $l_{\mathrm{td}_2} : \mathcal{Y}^{\tilde{m}} \times \mathcal{Y}^{\tilde{m}} \to \mathbb{R}^+$, by

$$l_{\mathrm{td}_2}\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{y}}'\right) \stackrel{\mathrm{def}}{=} \frac{1}{\tilde{m}} \sum_{i=1}^{\tilde{m}} \mathsf{I}_{\tilde{y}_i \neq \tilde{y}'_i} .$$

We aim at finding a labelling $\tilde{\boldsymbol{y}}^*$ that minimises the risk R_{td} as given in Definition 66. To this end consider the following theorem.

Theorem 44 (Transduction on joint labels). *Given a Bayesian transductive classification algorithm $\mathcal{A}_{\mathrm{btd}}$ and a joint loss function $l_{\mathrm{td}_1} : \mathcal{Y}^{\tilde{m}} \times \mathcal{Y}^{\tilde{m}} \to \mathbb{R}^+$ on labels, the labelling $\tilde{\boldsymbol{y}}^*$ returned by $\mathcal{A}_{\mathrm{btd}}\left(\tilde{\boldsymbol{x}}, \boldsymbol{z}\right)$ is given by*

$$\tilde{\boldsymbol{y}}^* = \operatorname*{argmax}_{\tilde{\boldsymbol{y}} \in \mathcal{Y}^{\tilde{m}}} \mathsf{P}_{\tilde{\mathsf{Y}}^{\tilde{m}} \mid \tilde{\mathsf{X}}^{\tilde{m}}=\tilde{\boldsymbol{x}}, \mathsf{Z}^{m}=\boldsymbol{z}}\left(\tilde{\boldsymbol{y}}\right) .$$

Proof. The risk can be written as

$$\begin{aligned} R_{\mathrm{td}_1}\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{x}}, \boldsymbol{z}\right) &= \mathsf{E}_{\tilde{\mathsf{Y}}^{\tilde{m}} \mid \tilde{\mathsf{X}}^{\tilde{m}}=\tilde{\boldsymbol{x}}, \mathsf{Z}^{m}=\boldsymbol{z}}\left[l_{\mathrm{td}_1}\left(\tilde{\boldsymbol{y}}, \tilde{\mathbf{Y}}\right)\right] \\ &= \sum_{\tilde{\boldsymbol{y}}' \in \mathcal{Y}^{\tilde{m}}} \left(1 - \mathsf{I}_{\tilde{\boldsymbol{y}}=\tilde{\boldsymbol{y}}'}\right) \mathsf{P}_{\tilde{\mathsf{Y}}^{\tilde{m}} \mid \tilde{\mathsf{X}}^{\tilde{m}}=\tilde{\boldsymbol{x}}, \mathsf{Z}^{m}=\boldsymbol{z}}\left(\tilde{\boldsymbol{y}}'\right) \\ &= 1 - \mathsf{P}_{\tilde{\mathsf{Y}}^{\tilde{m}} \mid \tilde{\mathsf{X}}^{\tilde{m}}=\tilde{\boldsymbol{x}}, \mathsf{Z}^{m}=\boldsymbol{z}}\left(\tilde{\boldsymbol{y}}\right) . \end{aligned}$$

Hence maximising the probability $\mathsf{P}_{\tilde{\mathsf{Y}}^{\tilde{m}} \mid \tilde{\mathsf{X}}^{\tilde{m}}=\tilde{\boldsymbol{x}}, \mathsf{Z}^{m}=\boldsymbol{z}}\left(\tilde{\boldsymbol{y}}\right)$ w.r.t. $\tilde{\boldsymbol{y}}$ minimises the risk. □

As a consequence, this non-label-wise loss is appropriate if the goal is to *exactly* identify a combination of labels, e.g. the combination of handwritten digits defining a postal zip code. In this case, a loss is incurred if even one of the digits receives the wrong label. Note that classical SVM transduction (see, e.g. Vapnik (1982) and Wu et al. (1999)) by maximising the margin on the combined training and working sample approximates this strategy and hence does not minimise the standard classification risk on single instances as might be intended.

Theorem 45 (Transduction on single labels). *Given a Bayesian transductive classification algorithm $\mathcal{A}_{\mathrm{btd}}$, a predictive posterior satisfying*

$$\mathbf{P}_{\tilde{\mathsf{Y}}^{\tilde{m}}|\tilde{\mathsf{X}}^{\tilde{m}}=\tilde{\boldsymbol{x}},\mathsf{Z}^m=\boldsymbol{z}}\left(\tilde{\boldsymbol{y}}\right)=\prod_{i=1}^{\tilde{m}}\mathbf{P}_{\tilde{\mathsf{Y}}|\tilde{\mathsf{X}}=\tilde{x}_i,\mathsf{Z}^m=\boldsymbol{z}}\left(\tilde{y}_i\right)$$

and a joint loss function $l_{\mathrm{td}_1}:\mathcal{Y}^{\tilde{m}}\times\mathcal{Y}^{\tilde{m}}\to\mathbb{R}^+$ on labels, the labelling returned by $\mathcal{A}_{\mathrm{btd}}\left(\tilde{\boldsymbol{x}},\boldsymbol{z}\right)$ is given by

$$\tilde{y}_i^*=\operatorname*{argmax}_{\tilde{y}\in\mathcal{Y}}\mathbf{P}_{\tilde{\mathsf{Y}}|\tilde{\mathsf{X}}=\tilde{x}_i,\mathsf{Z}^m=\boldsymbol{z}}\left(\tilde{y}\right).$$

Proof. The risk can be written as

$$\begin{aligned}
R_{\mathrm{td}_2}\left(\tilde{\boldsymbol{y}},\tilde{\boldsymbol{x}},\boldsymbol{z}\right) &= \mathbf{E}_{\tilde{\mathsf{Y}}^{\tilde{m}}|\tilde{\mathsf{X}}^{\tilde{m}}=\tilde{\boldsymbol{x}},\mathsf{Z}^m=\boldsymbol{z}}\left[l_{\mathrm{td}_2}\left(\tilde{\boldsymbol{y}},\tilde{\mathbf{Y}}\right)\right]\\
&= \frac{1}{\tilde{m}}\sum_{\tilde{\boldsymbol{y}}'\in\mathcal{Y}^{\tilde{m}}}\sum_{i=1}^{\tilde{m}}\left(1-\mathbf{I}_{\tilde{y}_i=\tilde{y}_i'}\right)\mathbf{P}_{\tilde{\mathsf{Y}}^{\tilde{m}}|\tilde{\mathsf{X}}^{\tilde{m}}=\tilde{\boldsymbol{x}},\mathsf{Z}^m=\boldsymbol{z}}\left(\tilde{\boldsymbol{y}}'\right)\\
&= 1-\frac{1}{\tilde{m}}\sum_{i=1}^{\tilde{m}}\sum_{\tilde{\boldsymbol{y}}'\in\mathcal{Y}^{\tilde{m}}}\mathbf{I}_{\tilde{y}_i=\tilde{y}_i'}\prod_{j=1}^{\tilde{m}}\mathbf{P}_{\tilde{\mathsf{Y}}|\tilde{\mathsf{X}}=\tilde{x}_j,\mathsf{Z}^m=\boldsymbol{z}}\left(\tilde{y}_j'\right)\\
&= 1-\frac{1}{\tilde{m}}\sum_{i=1}^{\tilde{m}}\mathbf{P}_{\tilde{\mathsf{Y}}|\tilde{\mathsf{X}}=\tilde{x},\mathsf{Z}^m=\boldsymbol{z}}\left(\tilde{y}_i\right).
\end{aligned}$$

Maximising each summand hence minimises the risk. □

Thus in the case of the additive loss l_{td_2} a working sample of $m>1$ point does not offer any advantages over larger working samples w.r.t. the Bayes-optimal decision. Since this corresponds to the standard classification setting we will restrict ourselves to working samples of size $m=1$, i.e. to one working point $\tilde{x}$. Transduction then corresponds to the Bayes classification strategy $\mathrm{Bayes}_{\boldsymbol{z}}$ as given in Definition 50.

7.1.2. Optimality of Bayesian Transduction

In order to see in which sense Bayesian transduction is optimal consider a probability measure $\mathbf{P}_{\mathsf{H}}$ over the space $\mathcal{Y}^{\mathcal{X}}$ of all possible mappings from $\mathcal{X}$ to $\mathcal{Y}$. Then, the *average generalisation error of a learning algorithm* $\mathcal{A}$ is defined as follows.

Definition 68 (Average Generalisation Error of Algorithms). Suppose we are given a fixed learning algorithm $\mathcal{A}:\bigcup_{m=1}^{\infty}\mathcal{Z}^m\to\mathcal{Y}^{\mathcal{X}}$. Then, for every fixed training sample size $m\in\mathbb{N}$ the *average generalisation error* $\overline{R}_m\left[\mathcal{A}\right]$ *of* $\mathcal{A}$ is defined by

$$\overline{R}_m\left[\mathcal{A}\right]=\mathbf{E}_{\mathsf{H}}\left[\mathbf{E}_{\mathsf{Z}^m|\mathsf{H}=h}\left[\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}\left[l\left(\left(\mathcal{A}\left(\mathbf{Z}\right)\right)\left(\mathsf{X}\right),\mathsf{Y}\right)\right]\right]\right]\right], \tag{7.1}$$

that is, the performance of the algorithm $\mathcal{A}$´s classification averaged over the random draw of training samples *and* target hypotheses.

The average generalisation error is the standard measure of performance of an algorithm $\mathcal{A}$ if we have little knowledge about the potential function h^* that labels all our data expressed via $\mathbf{P}_{\mathsf{H}}$. Then, the average generalisation error as given by 7.1 averages out our ignorance about the unknown h^* thus considering performance of $\mathcal{A}$ *on average.*

In the case that given a measure $\mathbf{P}_{\mathsf{H}}$, the conditional distribution of outputs y given x is governed by

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y) = \mathbf{P}_{\mathsf{H}}(\mathsf{H}(x) = y) , \tag{7.2}$$

we have that (see Definition 8)

$$\overline{R}_m[\mathcal{A}] = R_m[\mathcal{A}] .$$

In fact, for this result to hold it suffices to assume that $\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x}[l(y,\mathsf{Y})] = \mathbf{E}_{\mathsf{H}}[l(y,\mathsf{H}(x))]$, i.e. the prior correctly models the conditional distribution of the classes as far as the fixed loss is concerned. This result, however, is not too surprising taking into account that under the assumption (7.2) the measure $\mathbf{P}_{\mathsf{H}}$ fully encodes the unknown relationship between inputs x and outputs y.

Recall from Definition 50 that in order to classify a new test point x, for each class y the *Bayes classification strategy or Bayesian transduction* determines the loss incurred by each hypothesis $h \in \mathcal{H}$ applied to x and weights it according to its posterior probability $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}(h)$. The final decision is made for the class $y \in \mathcal{Y}$ that achieves the minimum expected loss, i.e.

$$\text{Bayes}(x;\boldsymbol{z}) = \operatorname*{argmin}_{y\in\mathcal{Y}} \mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}[l(\mathsf{H}(x),y)] . \tag{7.3}$$

This strategy has the following appealing property.

Theorem 46 (Optimality of the Bayes Classification Strategy). *Suppose we are given a fixed hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$. Then, for any training sample size $m \in \mathbb{N}$, for any symmetric loss $l : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}^+$, for any two measures $\mathbf{P}_{\mathsf{H}}$ and $\mathbf{P}_{\mathsf{X}}$, among all learning algorithms the Bayes classification strategy* Bayes *given by (7.3) minimises the average generalisation error $\overline{R}_m$[Bayes] under the assumption that for all h with $\mathbf{P}_{\mathsf{H}}(h) > 0$, for all $y \in \mathcal{Y}$, and for all $x \in \mathcal{X}$,*

$$\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}[l(y,\mathsf{Y})] = l(y,h(x)) . \tag{7.4}$$

Proof. Let us consider a fixed learning algorithm $\mathcal{A}$. Then it holds true that

$$\begin{aligned}
\overline{R}_m[\mathcal{A}] &= \mathbf{E}_{\mathsf{H}}\left[\mathbf{E}_{\mathsf{Z}^m|\mathsf{H}=h}\left[\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}[l((\mathcal{A}(\mathbf{Z}))(\mathsf{X}),\mathsf{Y})]\right]\right]\right] \\
&= \mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{H}}\left[\mathbf{E}_{\mathsf{Z}^m|\mathsf{H}=h}\left[\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}[l((\mathcal{A}(\mathbf{Z}))(\mathsf{X}),\mathsf{Y})]\right]\right]\right] \\
&= \mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Z}^m}\left[\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left[\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}[l((\mathcal{A}(\boldsymbol{z}))(\mathsf{X}),\mathsf{Y})]\right]\right]\right] \\
&= \mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Z}^m}\left[\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}[l((\mathcal{A}(\boldsymbol{z}))(\mathsf{X}),\mathsf{H}(\mathsf{X}))]\right]\right] ,
\end{aligned} \tag{7.5}$$

where we exchanged the order of expectations over $\mathcal{X}$ in the second line, applied the theorem of repeated integrals (see, e.g. Feller (1966)) in the third line and finally used

(7.4) in the last line. Using the symmetry of the loss function, the inner-most expression of (7.5) is minimised by the Bayes classification strategy (7.3) for any possible training sample $\boldsymbol{z}$ and any possible test input $x \in \mathcal{X}$. Hence, (7.3) minimises the whole expression which proves the theorem. □

In order to enhance the understanding of this result let us consider the simple case of $l = l_{0-1}$ and $\mathcal{Y} = \{-1, +1\}$. Then, given a particular classifier $h \in \mathcal{H}$ having non-zero prior probability $\mathbf{P}_{\mathsf{H}}(h) > 0$, by assumption (7.4) we require that the conditional distribution of classes y given x is delta peaked at $h(x)$ because

$$\begin{aligned} \mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}\left(l_{0-1}\left(y,\mathsf{Y}\right)\right) &= l_{0-1}\left(y,h\left(x\right)\right), \\ \mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}\left(-y\right) &= \mathsf{I}_{y\neq h(x)}, \\ \mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}\left(y\right) &= \mathsf{I}_{h(x)=y}. \end{aligned}$$

Now, albeit for a fixed $h \in \mathcal{H}$ drawn according to $\mathbf{P}_{\mathsf{H}}$ we *do not know* that Bayes achieves the smallest prediction error $R[\text{Bayes}]$ we can guarantee that on average over the random draw of h's the Bayes classification strategy is superior. In fact, the optimal classifier for a fixed $h^* \in \mathcal{H}$ is simply h^* itself and in general $\text{Bayes}(x;\boldsymbol{z}) \neq h^*(x)$ for at least a few $x \in \mathcal{X}$. It is worthwhile mentioning that the only information to be used in any classification strategy is the training sample $\boldsymbol{z}$ and the prior $\mathbf{P}_{\mathsf{H}}$. Hence it is impossible to *detect* which classifier $h \in \mathcal{H}$ labels a fixed m-tuple $\boldsymbol{x}$ only on the basis of the m labels $\boldsymbol{y}$ observed on the training sample. Thus, although we might be lucky in guessing h *for a fixed* $h \in \mathcal{H}$ and $\boldsymbol{z} \in \mathcal{Z}^m$ we cannot do better than the Bayes classification strategy Bayes when considering the average performance—the average being taken over the random choice of classifiers h and training samples $\boldsymbol{z}$.

7.1.3. Transduction by Maximising Volume

In order to do transduction based on the hypothesis space $\mathcal{H}_{\mathcal{K}}$ of linear classifiers we define a uniform prior $\mathbf{P}_{\mathsf{W}} \equiv \mathbf{U}_{\mathsf{W}}$ over weight space $\mathcal{W}$ and assume the existence of a version space $V(\boldsymbol{z}) \subset \mathcal{W}$. In this case, the posterior $\mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=\boldsymbol{z}}$ is given by

$$\mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=\boldsymbol{z}} \propto \begin{cases} 1 & \text{for} \quad \mathbf{W} \in V(\boldsymbol{z}) \\ 0 & \text{otherwise} \end{cases}.$$

Consider Figure 7.1 for an illustration of the above transductive principles in the context of linear classifiers. Transduction then proceeds as follows:

1. Obtain a sequence $\boldsymbol{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_N)$ of N weight vectors $\mathbf{w} \in V(\boldsymbol{z})$ uniformly from version space, e.g., by any of the methods to be described in Chapter 8.

2. Estimate the posterior label probabilities $p^y \stackrel{\text{def}}{=} \mathbf{P}_{\tilde{\mathsf{Y}}|\tilde{\mathsf{X}}=\tilde{x},\mathsf{Z}^m=\boldsymbol{z}}(y)$ by

$$\hat{p}^y = \frac{1}{N}\left|\{\mathbf{w}_i \in \boldsymbol{w} \,|\, h_{\mathbf{w}_i}(\tilde{x}) = y\}\right|.$$

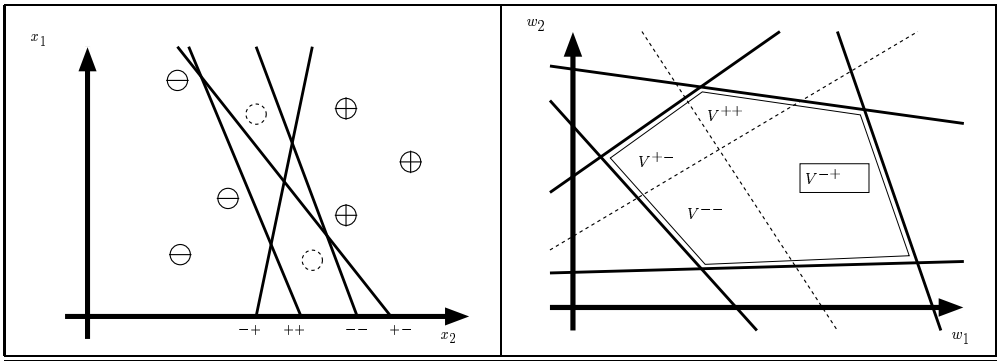


Figure 7.1.: Schematic view of data space (left) and weight space (right) for a classification toy example. Using the duality given by $\langle \mathbf{w}, \mathbf{x} \rangle = 0$ data points on the left correspond to hyperplanes on the right, while hyperplanes on the left can be thought of as points on the right. The two dashed points on the left can be labelled jointly in four different ways as indicated by the four decision boundaries. In weight space each of these four labellings corresponds to a subset $V^{y_1 y_2}$ of version space. The volumes of these subsets determine the optimal labelling for a given loss.

3. Determine the optimal label by

$$\tilde{y}^* = \operatorname*{argmax}_{y \in \mathcal{Y}} p^y$$

with the attached confidence value $c_{\text{td}} \in [0, 1]$

$$c_{\text{td}} = 2 \cdot p^{\tilde{y}^*} - 1 \,.$$

Disregarding the issue of mixing time (Cornfeld et al. 1982) and the dependence of samples we assume for the stopping criterion that $\hat{p}^+$ is the sum of N independently drawn Bernoulli variables with success probability p^+. Then $N\hat{p}^+$ is distributed Binomial (N, p^+) with expectation $\mathbf{E}\left[N\hat{p}^+\right] = Np^+$ and variance $\text{Var}\left(N\hat{p}^+\right) = Np^+\left(1 - p^+\right)$. From Chebyshev´s inequality, Theorem 60, we have that

$$\mathbf{P}_{\mathsf{X}}\left(\left|\hat{p}^+ - p^+\right| > \varepsilon\right) < \frac{p^+\left(1 - p^+\right)}{N\varepsilon^2} \stackrel{\text{def}}{=} \eta \,. \tag{7.6}$$

Thus if we want the deviation ε from the true label probability to be less than $\varepsilon < 0.05$ with probability at least $1 - \eta = 0.99$ we need approximately $N \approx 10000$ samples.

7.1.4. Discussion of Bayesian Transduction

Bayesian transduction as described above is effectively a fully Bayesian procedure for carrying out prediction on a working sample given a prior over a fixed hypothesis space.

1. In the special case of the hypothesis class $\mathcal{H}_\mathcal{K}$ of linear classifiers, a uniform prior $\mathbf{P}_\mathbf{W} \equiv \mathbf{U}_\mathbf{W}$ over weight space $\mathcal{W}$, and using the PAC-likelihood, Definition 47 it can be carried out exactly—up to appropriate sampling mechanisms—as described above.

2. As discussed in Subsection 7.1.2, Bayesian transduction can be considered optimal in the sense of the average generalisation error $\overline{R}_m[\mathcal{A}]$.

3. While it is interesting from a theoretical point of view to be able to carry out transduction in this way it is not suitable for large scale classification tasks in practice, because it takes $\mathcal{O}\left(m^2 N\right)$ kernel evaluations for a single classification. Also the memory requirements for storing the N weight vectors in their dual representation may be up to $\mathcal{O}(mN)$.

4. One great advantage of Bayesian transduction is the confidence value c_{td} that is effectively the margin by which the voting for the different classes is decided. We will demonstrate in Chapter 9 that c_{td} can in fact be considered a valid confidence measure.

7.2. Bayes Point Machines

7.2.1. The Bayes Point

Albeit the Bayes classification strategy is *on average* the optimal strategy when given limited amount of training data $\boldsymbol{z}$, it is computationally very demanding as it requires the evaluation of $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left(l\left(\mathsf{H}\left(x\right), y\right)\right)$ for each possible y at each new test point x. The problem lies in the fact that the Bayes classification strategy does not correspond to any one single classifier $h \in \mathcal{H}$. One way to tackle this problem is to require the classifier $\mathcal{A}\left(\boldsymbol{z}\right)$ learned from any training sample $\boldsymbol{z}$ to lie within a *fixed* hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^\mathcal{X}$ containing functions $h \in \mathcal{H}$ whose evaluation at a particular test point x can be carried out efficiently. Thus, if it is additionally required to limit the possible solution of a learning algorithm to a given hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^\mathcal{X}$, we can in general only hope to approximate Bayes.

Definition 69 (Bayes Point Algorithm). Suppose we are given an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^\mathcal{X}$ and a loss $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. Then, for any two measures $\mathbf{P}_\mathsf{X}$ and $\mathbf{P}_\mathsf{H}$, the *Bayes point algorithm* $\mathcal{A}_{\mathrm{bp}}$ is given by

$$\mathcal{A}_{\mathrm{bp}}\left(\boldsymbol{z}\right) \stackrel{\mathrm{def}}{=} \operatorname*{argmin}_{h \in \mathcal{H}} \ \mathbf{E}_\mathsf{X}\left[\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left[l\left(h\left(\mathsf{X}\right), \mathsf{H}\left(\mathsf{X}\right)\right)\right]\right] ,$$

that is, for each training sample $\boldsymbol{z} \in \mathcal{Z}^m$ the Bayes point algorithm chooses the classifier $\mathcal{A}_{\mathrm{bp}}\left(\boldsymbol{z}\right) \in \mathcal{H}$ that mimics best the Bayes classification strategy (7.3) on average over randomly drawn test points. The classifier $\mathcal{A}_{\mathrm{bp}}\left(\boldsymbol{z}\right)$ is called the *Bayes point*, a name to be found in Ruján (1997).

Assuming the correctness of the model given by (7.4) we furthermore remark that the Bayes point algorithm $\mathcal{A}_{\mathrm{bp}}$ is the best approximation to the Bayes classification strategy (7.3) in terms of the average generalisation error, i.e. measuring the distance of the learning algorithm $\mathcal{A}$ for $\mathcal{H}$ using the distance $\|\mathcal{A} - \mathrm{Bayes}\| = \overline{R}_m[\mathcal{A}] - \overline{R}_m[\mathrm{Bayes}]$. In this sense, for a fixed training sample $\boldsymbol{z}$ we can view the *Bayes point* $\mathcal{A}_{\mathrm{bp}}$ as a projection of Bayes into the hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$.

The difficulty with the Bayes point algorithm, however, is the need to know the input distribution $\mathbf{P}_{\mathbf{X}}$ for the determination of the hypothesis learned from $\boldsymbol{z}$. This somehow limits the applicability of the algorithm as opposed to the Bayes classification strategy which requires only broad prior knowledge about the underlying relationship expressed via some prior belief $\mathbf{P}_{\mathbf{H}}$.

7.2.2. The Centre of Mass

The Bayes point $\mathcal{A}_{\mathrm{bp}}$ can be approximated by the centre of mass $\mathcal{A}_{\mathrm{cm}}$, an observation first put forward by Ruján (1997), who also develops algorithmic ideas to determine $\mathcal{A}_{\mathrm{cm}}$. In order to approximate the Bayes point in the hypothesis space $\mathcal{H}_{\mathcal{K}}$ of linear classifiers we again define a prior $\mathbf{P}_{\mathbf{W}}$ over weight space $\mathcal{W}$. If we assume that the input distribution is spherical in the feature space $\mathcal{K}$,

$$\mathbf{f}_{\mathbf{X}}(\mathbf{x}) = \mathbf{f}_{\mathbf{X}}(\|\mathbf{x}\|) , \tag{7.7}$$

then we find that the classifier $\mathcal{A}_{\mathrm{cm}}$ corresponding to the centre of mass of the posterior,

$$\mathbf{w}_{\mathrm{cm}} = \frac{\mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}[\mathbf{W}]}{\left\|\mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}[\mathbf{W}]\right\|} \tag{7.8}$$

is a very good approximation to the Bayes point $\mathcal{A}_{\mathrm{bp}}$ and its error rate converges towards that of $\mathcal{A}_{\mathrm{bp}}$ if the posterior belief $\mathbf{P}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}$ becomes sharply peaked in the limit of $m \to \infty$ (for a similar result see Watkin (1993)).

Theorem 47 (Optimality of the Centre of Mass). *Consider the hypothesis space $\mathcal{H}_{\mathcal{K}}$. Then, for all $m \in \mathbb{N}$, if $\mathbf{P}_{\mathbf{X}}$ has a spherically symmetric density as given in (7.7) and the prior belief is correct, that is we have i.e. (7.4) is valid, the average prediction error $\overline{R}_m[\mathcal{A}_{\mathrm{cm}}]$ of the classifier $\mathcal{A}_{\mathrm{cm}} \in \mathcal{W}$ corresponding to the centre of mass weight vector $\mathbf{w}_{\mathrm{cm}}$ as given by (7.8) satisfies*

$$\overline{R}_m[\mathcal{A}_{\mathrm{cm}}] - \overline{R}_m[\mathcal{A}_{\mathrm{bp}}] \leq \mathbf{E}_{\mathbf{Z}^m}[\kappa(\varepsilon(\mathbf{Z}))] ,$$

where we define the function $\kappa : [-1, 1] \to [0, 1]$ by

$$\kappa(\varepsilon) = \begin{cases} \frac{1}{\pi}\arccos(\varepsilon) - \frac{1}{2}(1-\varepsilon) & \text{if} \quad \varepsilon \geq \sqrt{1 - \frac{4}{\pi^2}} \\ 0.11 & \text{otherwise} \end{cases} ,$$

and

$$\varepsilon(\boldsymbol{z}) = \min_{\mathbf{w}:\mathbf{P}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}(\mathbf{w})>0} \langle \mathbf{w}_{\mathrm{cm}}, \mathbf{w} \rangle .$$

The proof of Theorem 47 is given in Appendix A. The interesting fact to note about this result is that $\lim_{\varepsilon \to 1} \kappa(\varepsilon) = 0$ and thus we have

$$\lim_{m \to \infty} \mathbf{E}_{\mathbf{Z}^m} \left[\kappa \left(\varepsilon \left(\mathbf{Z} \right) \right) \right] = 0 \,,$$

because for increasing training sample size the posterior is sharply peaked at the weight vector labelling the data. This result is a slight generalisation of the result given in Watkin (1993) which only proved this to be true for the uniform prior $\mathbf{P}_{\mathbf{W}} = \mathbf{U}_{\mathbf{W}}$. This shows that for increasing training sample size the centre of mass (under the posterior $\mathbf{P}_{\mathbf{W}|\mathbf{Z}^m = \boldsymbol{z}}$) is a good approximation to the optimal projection of the Bayes classification strategy—the *Bayes point*. Henceforth, any algorithm which aims at returning the centre of mass under the posterior $\mathbf{P}_{\mathbf{W}|\mathbf{Z}^m = \boldsymbol{z}}$ is called a *Bayes point machine*. Note that in the case of the PAC likelihood as defined in Definition 47 the centre of mass under the posterior $\mathbf{P}_{\mathbf{W}|\mathbf{Z}^m = \boldsymbol{z}}$ coincides with the centre of mass of version space.

7.2.3. The Centre of Mass in Kernel Space

We aim at using the centre of mass classifier $\mathcal{A}_{\mathrm{cm}}$ in kernel space similarly to the kernel perceptron and the support vector machine. To this end we restrict ourselves to weight vectors $\mathbf{w}_{\mathrm{cm}}$ that can be expanded in terms of input vectors $\mathbf{x}_i$,

$$\mathbf{w}_{\mathrm{cm}} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i \,. \tag{7.9}$$

For the kernel perceptron this property simply followed from the update rule of the perceptron (see Subsection 2.1.1). For the SVM the KKT stationarity conditions provided the necessary expansion (see Subsection 2.1.3). For more general cost functions the expansion follows from the representer theorem, Theorem 9. In fact, for a uniform prior, $\mathbf{P}_{\mathbf{W}} = \mathbf{U}_{\mathbf{W}}$, and a likelihood $\mathbf{P}_{\mathbf{Y}|\mathbf{X}=x,\mathbf{W}=\mathbf{w}}(y) = f\left(y h_{\mathbf{w}}(x)\right)$ that depends on the weight vector $\mathbf{w}$ via the classification $h_{\mathbf{w}}(x) \in \mathcal{Y}$ only, it can be shown that the centre of mass weight vector $\mathbf{w}_{\mathrm{cm}}$ admits a representation of the form (7.9). Denoting by $\mathcal{W}_{\boldsymbol{x}}$ the intersection $\mathcal{W}_{\boldsymbol{x}} = \mathcal{W} \cap \mathcal{L}_{\boldsymbol{x}}$ of $\mathcal{W}$ and the linear span $\mathcal{L}_{\boldsymbol{x}}$ of the training data this can be demonstrated by showing that the weight vector $\mathbf{w}_{\mathrm{cm}}$ that minimises

$$\mathbf{E}_{\mathbf{V}|\mathbf{V} \in \mathcal{W}, \mathbf{Z}^m = \boldsymbol{z}} \left[\|\mathbf{w} - \mathbf{V}\|^2 \right]$$

also minimises

$$\mathbf{E}_{\mathbf{V}|\mathbf{V} \in \mathcal{W}_{\boldsymbol{x}}, \mathbf{Z}^m = \boldsymbol{z}} \left[\|\mathbf{w} - \mathbf{V}\|^2 \right] \,,$$

The complete argument can be found in Herbrich et al. (2001). As a consequence of the representation (7.9) we can provide a version of the Bayes point machine in kernel space by uniformly sample from version space. The Bayes point machine algorithm then proceeds as follows:

1. Obtain a sequence $\boldsymbol{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_N)$ of N weight vectors $\mathbf{w}_i \in V(\boldsymbol{z})$ uniformly from version space, e.g., by any of the methods to be described in Chapter 8.

2. Approximate the centre of mass $\mathbf{w}_{\text{cm}}$ by

$$\mathbf{w}_{\text{cm}} \approx \frac{\sum_{i=1}^{N} \mathbf{w}_i}{\left\| \sum_{i=1}^{N} \mathbf{w}_i \right\|} .$$

Of course, it might be more efficient to use the samples $\mathbf{w}_i$ online to calculate $\mathbf{w}_{\text{cm}}$ and to discard them immediately. An appropriate update formula with derivation can be found in Herbrich, Graepel, and Campbell (2001).

Recently other efficient methods to estimate the Bayes point directly (Rychetsky et al. 2000; Minka 2001) have been presented. The main idea in Rychetsky et al. (2000) is to work out all corners $\mathbf{w}_i$ of version space and average over them in order to approximate the centre of mass of version space. Note that there are exactly m corners because the ith corner $\mathbf{w}_i$ satisfies $\langle \mathbf{x}_j, \mathbf{w}_i \rangle = 0$ for all $j \neq i$ and $y_i \langle \mathbf{x}_i, \mathbf{w}_i \rangle > 0$. If $\mathbf{X} = (y_1 \mathbf{x}_1, \ldots, y_m \mathbf{x}_m)$ is the $N \times m$ matrix of mapped training points $\boldsymbol{x} = (x_1, \ldots, x_m)$ flipped to their correct side and we use the dual representation (7.9) for $\mathbf{w}$ this simplifies to

$$\mathbf{X}'\mathbf{w}_i = \mathbf{X}'\mathbf{X}\boldsymbol{\alpha}_i = \mathbf{G}\boldsymbol{\alpha}_i = (0, \ldots, 0, y_i, 0 \ldots, 0)' =: y_i \mathbf{e}_i$$

where the r.h.s. is the ith unit vector multiplied by y_i. As a consequence, the expansion coefficients $\boldsymbol{\alpha}_i$ of the ith corner $\mathbf{w}_i$ can easily be computed as $\boldsymbol{\alpha}_i = y_i \mathbf{G}^{-1} \mathbf{e}_i$ and are then normalised such that $\|\mathbf{w}_i\| = 1$. The difficulty with this approach, however, is the fact that the inversion of the $m \times m$ Gram matrix $\mathbf{G}$ is $\mathcal{O}\left(m^3\right)$ and is thus as computationally complex as naive support vector learning while not enjoying the anytime property of a sampling scheme.

The algorithm presented in Minka (2001), Chapter 5, (also see Opper and Winther (1999) for an equivalent method) uses the idea of approximating the posterior measure $\mathsf{P}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}$ by a product of Gaussian densities so that the centre of mass can be computed analytically. Although the approximation of the cut-off posterior over $\mathsf{P}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}$ resulting from the delta-peaked likelihood given in Definition 47 by Gaussian measures seems very crude at first glance, Minka could show that his method compares favourably to the results presented here.

7.2.4. Discussion of Bayes Point Machines

Bayes point machines are an attempt of approximating a fully Bayesian inference of labels based on an hypothesis space $\mathcal{H}$—as done in Bayesian transduction—by a single classifier $h \in \mathcal{H}$.

1. The centre of mass classifier is appealing from an intuitive point of view and has empirically been shown to have excellent generalisation ability (Herbrich et al. 2001). A rudimentary theoretical justification was given as in terms of an approximation scheme: Centre of mass approximates Bayes point approximates Bayesian classification strategy.

2. To date there exists no full theoretical justification for its use. We were not able to show that the centre of mass summarises in general a higher volume $\mathcal{Q}(\boldsymbol{z}) \subset V(\boldsymbol{z})$ in the sense of the PAC-Bayesian theorem, Theorem 28, nor that it is a good approximator of the Bayesian classification strategy for arbitrary input distributions. The further theoretical justification of the centre of mass is certainly an interesting avenue for further research.

3. As before, the restriction to version space is compensated for by adjusting the kernel as in the approach outlined in Remark 1 for the perceptron algorithm.

4. While non-sampling approaches are conceivable (Rychetsky et al. 2000; Minka 2001), our sampling approach has the merit that computational complexity can be fine-tuned by the number of sampled weight vectors.

5. Further empirical evidence for the performance of the Bayes point approach can be found in Chapter 9.

8. Sampling Schemes in Kernel Space

The development of kernel machines has been driven by essentially two communities: The PAC/VC learning theory school of thought developed the support vector machine (Vapnik 1995) while the Bayesian machine learning community introduced, e.g., Gaussian processes for classification (Barber and Williams 1997; Opper and Winther 2000b) or the relevance vector machine (Tipping 2001). Both schemes essentially work in the same function space that is characterised by kernels and covariance functions, respectively. While the formal similarity of the two methods is striking the underlying paradigms of inference are very different. This rather ideological clash can be viewed as a continuation in machine learning of the by now classical disagreement between Bayesian and frequentist statistics (Aitchison 1964). With regard to algorithms the two schools of thought appear to favour two different methods of learning and predicting: the support vector community—as a consequence of the formulation of the support vector machine as a quadratic programme—focuses on learning as optimisation while the Bayesian community favours sampling schemes based on the Bayesian posterior. Of course there exists a strong relationship between the two ideas, in particular with the Bayesian maximum a posteriori (MAP) estimator being the solution of an optimisation problem.

In practice, optimisation based algorithms have the advantage of a unique, deterministic solution and the availability of the cost function as an indicator of the quality of the solution. In contrast, Bayesian algorithms based on sampling and voting are more flexible and enjoy the so-called "anytime" property, providing a relatively good solution at any point in time. Often, however, they suffer from the computational costs of sampling the Bayesian posterior.

In this chapter we present three sampling schemes suitable for learning linear classifiers in kernel space: kernel billiard sampler, permutational perceptron sampler, and the kernel Gibbs sampler. In contrast to the Gaussian process viewpoint we do not define a Gaussian prior on the length $\|\mathbf{w}\|$ of the weight vector. Instead, we only consider weight vectors of length $\|\mathbf{w}\| = 1$ on the unit sphere $\mathbf{w} \in \mathcal{W}$ because it is only the spatial direction of the weight vector that matters for classification. It is then natural to define a uniform prior on the resulting ball–shaped hypothesis space.

In the simplest case, we use the PAC-likelihood, Definition 47, and obtain a posterior that is constant and non-zero within version space and vanishes elsewhere. Both the kernel billiard sampler and the permutational perceptron sampler are suitable for this kind of posterior, the permutational perceptron sampler being more efficient for large scale data sets. A more flexible if less efficient sampling method is provided by the kernel Gibbs sampler that can be used to sample from a piecewise constant posterior as

is desirable under a classification noise model.

The section on the kernel billiard sampler, Section 8.2, is based on Herbrich, Graepel, and Campbell (1999a), Graepel, Herbrich, and Obermayer (2000), Herbrich, Graepel, and Campbell (2000), and Herbrich, Graepel, and Campbell (2001). The section on the kernel Gibbs sampler is based on Graepel and Herbrich (2001a) and Herbrich, Graepel, and Williamson (2001). Finally, the section on permutational Gibbs sampling is based on Herbrich and Graepel (2001a) and Herbrich, Graepel, and Campbell (2001).

8.1. Sampling Hypothesis Space

8.1.1. Evaluating Posterior Expectations

As discussed in Section 3.2 the first step in Bayesian inference is to obtain a posterior probability measure $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$. Subsequently, one often aims at evaluating the expectation $\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left[f\left(\mathsf{H}\right)\right]$ of given functions f under the posterior. Often the functional form of $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ is not amenable to subsequent computations. In particular the sum or integral given by $\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left[f\left(\mathsf{H}\right)\right]$ may be difficult to evaluate. There are two ways to overcome this difficulty:

1. Functional approximation of the Bayesian posterior $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ such that an evaluation of $\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left[f\left(\mathsf{H}\right)\right]$ becomes feasible. Popular techniques include the Laplace approximation for peaked posteriors (Gelman et al. 1995) as well as mean field approximations (Opper and Winther 2000b) and related techniques (Minka 2001).

2. Sampling form the Bayesian posterior $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ and approximating the expectation $\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left[f\left(\mathsf{H}\right)\right]$ by a sum over samples (Neal 1993).

Often approximation techniques are efficient and yield good results. In fact, very few models are known in which Bayesian inference can be carried out exactly (Gelman et al. 1995) and it appears that this may not be necessary at all to obtain good results. Sampling methods—also known as Monte-Carlo methods due to the random element of sampling—have the advantage that once the samples have been obtained the evaluation of any expectation is straightforward: The expectation is simply replaced by the mean over the samples,

$$\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}\left[f\left(\mathsf{H}\right)\right] \approx \frac{1}{N}\sum_{i=1}^{N} f\left(h_i\right) \qquad h_i \sim \mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}} \ .$$

An additional advantage is that the degree of approximation can be tuned and weighted against computational complexity by adjusting the number N of samples.

8.1.2. Sampling Linear Classifiers

In our case, we focus on the hypothesis space $\mathcal{H}_{\mathcal{K}}$ of hypotheses characterised by weight vectors $\mathbf{w}$ of unit length $\|\mathbf{w}\|^2 = 1$ on the unit hyper-sphere, $\mathbf{w} \in \mathcal{W} \subset \mathcal{K}$. We can then

write the posterior $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}}$ as $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}} \equiv \mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=\boldsymbol{z}}$. Two expectations to be evaluated are, e.g.,

1. The expectation of labels, $f(\mathbf{w}) = \operatorname{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$, for the case of Bayesian transduction, aiming at a prediction of the label y of the test input $\mathbf{x}$.

2. The expectation of weight vectors, $f(\mathbf{w}) = \mathbf{w}$, to obtain the centre of mass $\mathbf{w}_{\mathrm{cm}}$ of the posterior as used in Bayes point machines.

Unfortunately, in the case of linear classifiers—in particular taking into account kernel classifiers—the space $\mathcal{W}$ to be sampled can be of rather high dimensionality n. In fact, using kernels corresponding to feature spaces $\mathcal{K}$ of infinite dimensionality, the effective dimensionality might be as high as m, the number of training examples. As a consequence, sampling even a simple piecewise constant posterior constitutes a challenge.

The simplest idea that comes to mind is called *rejection sampling*. The recipe is simple: Sample uniformly from the whole space $\mathcal{W}$ and reject examples if they do not fall into a region of non-zero posterior probability. While it is conceptually simple rejection sampling is bound to fail under these circumstances because of the *curse of dimensionality*. The ratio $\frac{N_a}{N_r}$ of the number N_a of accepted samples and N_r of rejected samples will decrease exponentially in the number n of dimensions. This leads to a prohibitive computational complexity of rejection sampling. The curse of dimensionality leads to the idea of a more goal-oriented sampling strategy.

8.1.3. Markov Chain Monte Carlo Methods

Markov Chain Monte Carlo (MCMC) methods (Neal 1993) aim at approximating the mean $\frac{1}{N}\sum_{i=1}^{N} f(\mathbf{w}_i)$ based on an iid sample

$$\boldsymbol{w}_{\mathrm{iid}} \stackrel{\mathrm{def}}{=} (\mathbf{w}_1, \ldots, \mathbf{w}_N) \qquad \mathbf{w}_i \sim \mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=\boldsymbol{z}}$$

by the mean based on a subsequence $\boldsymbol{w}_{\mathrm{mc}}$ of the realisation of a *Markov chain* $(\mathbf{W}_0, \ldots,)$, which is fully characterised by a starting point $\mathbf{w}_0$ and transition probabilities $\mathbf{P}_{\mathsf{W}|\tilde{\mathsf{W}}}$. The sample $\boldsymbol{w}_{\mathrm{mc}}$ is obtained by choosing an initial weight vector $\mathbf{w}_0$ and then sampling iteratively from

$$\mathbf{w}_{i+1} \sim \mathbf{P}_{\mathsf{W}|\tilde{\mathsf{W}}=\mathbf{w}_i}.$$

We call the Markov chain *ergodic* with respect to the distribution $\mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=\boldsymbol{z}}$ if the limiting distribution of this sampling process is given by $\mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=\boldsymbol{z}}$ regardless of the starting point $\mathbf{w}_0$ (Cornfeld et al. 1982). However, mere ergodicity might not be enough in practice, because the *mixing time* of the Markov chain determines the number of samples needed for an accurate approximation of the expectation of interest. Roughly speaking the mixing time is given by the number of steps it takes until the limiting distribution is well approximated by the Markov chain.

8.2. The Kernel Billiard Sampler

In this section we present the kernel billiard sampler as a method for obtaining samples of version space when assuming a PAC likelihood and a uniform prior $\mathbf{P_W} = \mathbf{U_W}$ over weight vectors of unit length. The billiard sampler was first introduced in Ruján (1997) in the primal formulation. The kernel version was independently introduced in Ruján and Marchand (2000) and Herbrich, Graepel, and Campbell (1999a). The key idea is to use a deterministic dynamical system such as a billiard to obtain a sequence $\boldsymbol{w}$ of pseudo-random samples from a convex polyhedron. Under the assumption that the billiard is ergodic w.r.t. the uniform distribution $\mathbf{U_W}$ (see, e.g., Cornfeld et al. (1982) for details on ergodic theory) sampling from the location of the billiard ball provides an efficient way of obtaining samples from a uniform density. Describing any single state by a position/direction pair $(\mathbf{b}, \mathbf{v}) \in \mathcal{W} \times \mathcal{W}$ the resulting trajectory can be considered as a Markov chain in the sense that the subsequent state $(\mathbf{b}, \mathbf{v})_{i+1}$ solely depends on the present state $(\mathbf{b}, \mathbf{v})_i$ and the training sample. Since the system is deterministic it cannot be considered a MCMC approach, however.

Each position $\mathbf{b}$ of the billiard ball can be expressed as a linear combination of the mapped input points, i.e.

$$\mathbf{b} = \sum_{i=1}^{m} \gamma_i \mathbf{x}_i \,, \qquad \boldsymbol{\gamma} \in \mathbb{R}^m \,.$$

Without loss of generality we can make the following ansatz for the direction vector $\mathbf{v}$ of the billiard ball

$$\mathbf{v} = \sum_{i=1}^{m} \beta_i \mathbf{x}_i \,, \qquad \boldsymbol{\beta} \in \mathbb{R}^m \,.$$

Recall from Subsection 2.3.2 that inner products and norms in feature space $\mathcal{K}$ can be expressed as

$$\langle \mathbf{b}, \mathbf{v} \rangle = \sum_{i=1}^{m} \sum_{j=1}^{m} \beta_i \gamma_j k(x_i, x_j) \,, \qquad \|\mathbf{b}\|^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} \gamma_i \gamma_j k(x_i, x_j) \,, \tag{8.1}$$

where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a Mercer kernel satisfying the conditions given in Theorem 8. Before generating a billiard trajectory in version space $V(\boldsymbol{z})$ we first run any learning algorithm to find an initial starting point $\mathbf{b}_0$ inside the version space (e.g. support vector learning or the kernel perceptron (see Algorithm 2). Then the kernel billiard algorithm consists of three steps (see also Figure 8.1):

1. Determine the closest boundary in direction $\mathbf{v}_i$ starting from current position $\mathbf{b}_i$.

   Since it is computationally very demanding to calculate the flight time of the billiard ball *on* geodesics of the hyper-sphere $\mathcal{W}$ (see also Neal (1997b)) we make use of the fact that the shortest distance in Euclidean space (if it exists) is also the shortest distance on the hyper-sphere $\mathcal{W}$. Thus, we have for the flight time τ_j of the billiard ball at position $\mathbf{b}_i$ in direction $\mathbf{v}_i$ to the hyperplane with normal vector $y_j \mathbf{x}_j$

   $$\tau_j = -\frac{\langle \mathbf{b}_i, \mathbf{x}_j \rangle}{\langle \mathbf{v}_i, \mathbf{x}_j \rangle} \,. \tag{8.2}$$

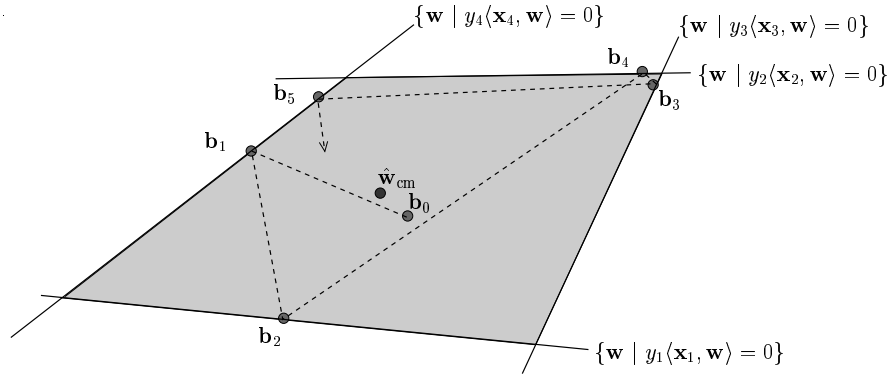


Figure 8.1.: Schematic view of the kernel billiard algorithm. Starting at $\mathbf{b}_0 \in V(\boldsymbol{z})$ a trajectory of billiard bounces $\mathbf{b}_1, \ldots, \mathbf{b}_5, \ldots$ is calculated. This sequence of samples can, for example, be used to find an approximation $\hat{\mathbf{w}}_{\text{cm}}$ to the centre of mass of version space, as used in Bayes point machines.

After calculating all m flight times, we look for the smallest positive, i.e.

$$c = \text{argmin}_{j:\tau_j>0}\tau_j \,.$$

Determining the closest bounding hyperplane in Euclidean space rather than on geodesics causes problems if the surface of the hyper-sphere $\mathcal{W}$ is almost orthogonal to the direction vector $\mathbf{v}_i$, in which case $\tau_c \to \infty$. If this happens we randomly generate a new direction vector $\mathbf{v}_i$ pointing *towards* the version space $V(\boldsymbol{z})$. Assuming that the last bounce took place at the hyperplane with normal $y_{c'}\mathbf{x}_{c'}$ this condition can easily be checked by

$$y_{c'}\left\langle \mathbf{v}_i, \mathbf{x}_{c'}\right\rangle > 0\,. \tag{8.3}$$

Please note that since the samples are taking from the bouncing points the above procedure of dealing with the curvature of the hyper-sphere does not constitute an approximation but is exact. An alternative method of dealing with the problem of the curvature of the hyper-sphere $\mathcal{W}$ can be found in Minka (2001), Section 5.8.

2. Update the billiard ball's position to $\mathbf{b}_{i+1}$ and the new direction vector to $\mathbf{v}_{i+1}$.

   The new point $\mathbf{b}_{i+1}$ and the new direction $\mathbf{v}_{i+1}$ are calculated from

$$\mathbf{b}_{i+1} = \mathbf{b}_i + \tau_c\mathbf{v}_i\,, \tag{8.4}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i - 2\frac{\langle \mathbf{v}_i, \mathbf{x}_c\rangle}{\|\mathbf{x}_c\|^2}\mathbf{x}_c\,. \tag{8.5}$$

   Finally we normalise the position vector $\mathbf{b}_{i+1}$ and the direction vector $\mathbf{v}_{i+1}$ by (8.1).

3. Store the sampling point $\mathbf{b}_{i+1}$ and repeat.

Pseudocode for the kernel billiard sample can be found in Appendix C.3.

8.2.1. Discussion of the Billiard Approach

While the kernel billiard sampler is a clever method to efficiently sample version space it has several drawbacks.

1. Samples can only be obtained from a uniform measure $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=\boldsymbol{z}} = \mathbf{U}_{\mathsf{H}|\mathsf{H}\in V}$ within version space $\mathrm{V}(\boldsymbol{z})$ or—more generally—within a given convex polyhedron. As discussed earlier, this simple setting is not always adequate in practice. However, the simple 2-norm soft margin modification of the kernel discussed in Remark 1 allows for a quite general application of the kernel billiard sampler.

2. The samples are not actually uniform over version space $V \equiv V(\boldsymbol{z})$ but over its surface $\partial V = \partial V(\boldsymbol{z})$ if only the hit points $\mathbf{b}_i$ are taken into account. On the one hand, this is not really problematic since the number of dimensions of V and ∂V only differ by one, $\dim(V) = \dim(\partial V) + 1$, and for sufficiently large $\dim(V)$ this amounts to $\frac{\dim(\partial V)}{\dim(V)} \approx 1$ which in turn means $\mathrm{vol}(\partial V) \approx \mathrm{vol}(V)$ under the uniform measure $\mathbf{U}_{\mathsf{H}}$. Thus, sampling the surface ∂V of version space is often sufficient because the volume in high-dimensional spaces is concentrated on the surface of a body[1]. On the other hand, the problem can be solved by uniformly sampling on each line segment $\{\lambda \mathbf{b}_i + (1-\lambda)\,\mathbf{b}_{i+1} \,|\, 0 \leq \lambda \leq 1\}$.

3. The kernel billiard sampler has memory requirements of order $\mathcal{O}\left(m^2 + Nm\right)$, where m^2 is required for keeping the kernel matrix in memory, and Nm to store the sample points in their dual representation. While this is acceptable for small scale problems it leads to difficulties in larger applications. Of course, if the samples are used online to calculate the average of some quantity, the memory requirements are reduced to $\mathcal{O}\left(m^2\right)$.

4. The computational complexity of the kernel billiard sampler is of order $\mathcal{O}\left(\mathcal{N}m^2\right)$ because each sample point requires the evaluation of m inner products in an m-dimensional space. Again this computational burden may be critical for large scale applications.

5. The questions of ergodicity and mixing time of the kernel billiard sampler are difficult to evaluate and lead deeply into the theory of ergodic systems (Cornfeld et al. 1982). While the experimental results of Chapter 9 are encouraging, this aspect of the algorithm certainly deserves further attention.

[1] The general effect is easily illustrated by the hypercube $\mathcal{C}_n(d)$ of size d with volume $\mathrm{vol}(\mathcal{C}_n(d)) = d^n$. If we remove a cube $\mathcal{C}_n(1-\Delta)$, with $0 < \Delta \leq 1$ from the unit hypercube $\mathcal{C}_n(1)$ we have for the volume difference $\mathrm{vol}(\mathcal{C}_n(1) \setminus \mathcal{C}_n(1-\Delta)) = \mathrm{vol}(\mathcal{C}_n(1)) - \mathrm{vol}(\mathcal{C}_n(1-\Delta)) = 1 - (1-\Delta)^n$. In the limit of $n \to \infty$ we have thus $\lim_{n\to\infty} \mathrm{vol}(\mathcal{C}_n(1) \setminus \mathcal{C}_n(1-\Delta)) = 1$, i.e., the volume does not even change in the limit.

8.3. The Kernel Gibbs Sampler

The development of the kernel Gibbs sampler addresses the first two points in the above discussion of the kernel billiard sampler. First, we aim at a more flexible sampling scheme that is able not only to sample from a uniform density over a convex polyhedron, but over a piecewise constant density. Secondly, we would like to have a sampler at our disposal that is based on a truly random Markov chain instead of a deterministic but chaotic dynamical system. Thirdly, we prefer to sample the whole area V of interest instead of just ∂V, a point that may make a difference in low dimensional spaces. The kernel Gibbs sampler is inspired by an MCMC sampling technique known as Gibbs sampling.

8.3.1. Gibbs Sampling

We will consider Gibbs sampling as an MCMC sampling method for sampling from probability distributions $\mathbf{P}_{\mathbf{W}}$ with densities $\mathbf{f}_{\mathbf{W}}$ in high-dimensional spaces $\mathcal{W} \subseteq \mathbb{R}^n$. The basic idea is to construct a Markov chain with initial state $\mathbf{w}_0$ by selecting at each step a dimension $j \in \{1, \ldots, n\}$ and update the corresponding component of $\mathbf{w}$ only. We denote the new component by $\tilde{w}_j$ and the vector of remaining components by $\mathbf{w}^{(j)} \in \mathbb{R}^{n-1}$. The new component $\tilde{w}_j$ is drawn from the conditional probability density $\mathbf{f}_{\mathsf{W}_j|\mathbf{W}^{(j)}=\mathbf{w}^{(j)}}(w_j)$ given the fixed values of the remaining components $\mathbf{w}^{(j)}$ from the previous step. The conditional density is given by

$$\mathbf{f}_{\mathsf{W}_j|\mathbf{W}^{(j)}=\mathbf{w}^{(j)}}(w_j) = \frac{\mathbf{f}_{\mathbf{W}}(\mathbf{w})}{\mathbf{P}_{\mathbf{W}^{(j)}}\left(\mathbf{w}^{(j)}\right)},$$

and the feasibility of Gibbs sampling crucially depends on the question if this probability distribution can be derived and if it can be efficiently sampled. If feasible Gibbs sampling constitutes a very practical and conceptually simple sampling scheme for which there exist ergodicity results and bounds on the mixing time for a number of cases (Neal 1993).

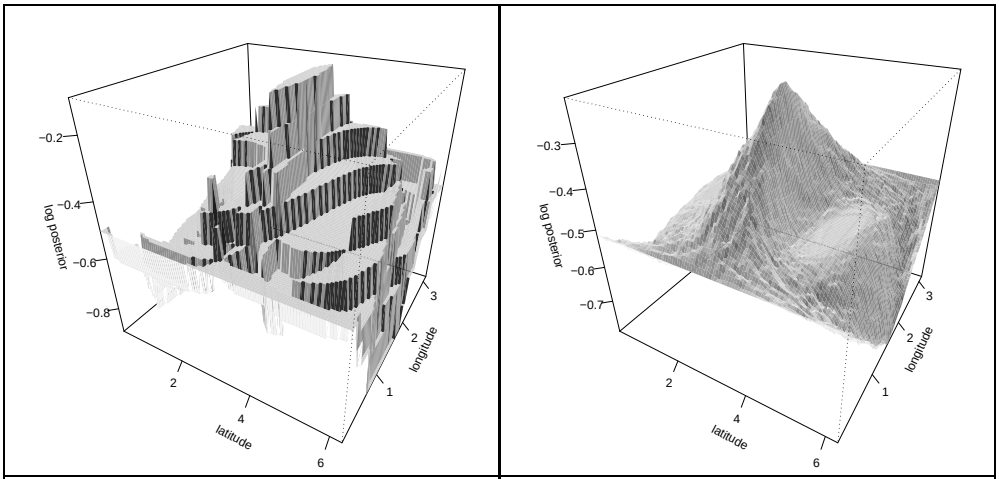
In the form introduced above Gibbs sampling is not easily applicable to kernel spaces $\mathcal{K}$ because its formulation refers explicitly to distributions on the vector representation of $\mathbf{w}$. This problem will be overcome by the kernel Gibbs sampler.

8.3.2. Bayesian Treatment of Label Noise

The sampling approach—in contrast to the billiard approach—allows us to sample from more complex densities than just a uniform density in version space. In order to see why this may be desirable in classification learning reconsider Bayes´ rule (3.7) resulting in a posterior

$$\mathbf{f}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}(\mathbf{w}) \propto \mathbf{P}_{\mathsf{Y}^m|\mathsf{X}^m=\boldsymbol{x},\mathbf{W}=\mathbf{w}}(\boldsymbol{y}) \cdot \mathbf{f}_{\mathbf{W}}(\mathbf{w}) .$$

If we assume a uniform prior density $\mathbf{f}_{\mathbf{W}}(\mathbf{w})$ over $\mathcal{W}$ and the PAC likelihood, Definition 47, then we have a uniform posterior density $\mathbf{f}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}} = \mathbf{U}_{\mathbf{W}|\mathbf{W}\in V(\boldsymbol{z})}$ over version space. This scenario corresponds to the zero noise case under the assumption that a teacher classifier $h^*_{\mathbf{w}}$ labels the data.



$m = 10$ | $m = 1000$

Figure 8.2.: Illustration of the shape of the posterior probability density $\mathbf{f}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}$ under label noise for a toy example. Shown is the (log) posterior density on the surface of sphere $\mathcal{W} \subset \ell_2^3$ resulting from a label noise model with a flip rate of $\theta = \frac{1}{5}$ for two different values of the sample size $m = 10$ and $m = 1000$. The log posterior is plotted over the longitude and latitude, and for small sample size is multi-modal due to the label noise. The weight vector $\mathbf{w}^*$ corresponding to the classifier $h_{\mathbf{w}^*}$ labelling the data (before label noise) has spherical coordinates $\left(\frac{\pi}{2}, \pi\right)$.

A more complex noise model results if we assume that the teacher $h^*_{\mathbf{w}}$ still labels the data, but that the labels are corrupted by noise in the sense of a likelihood

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathbf{W}=\mathbf{w}}(y) = \begin{cases} \theta & \text{if} \quad h_{\mathbf{w}}(x) \neq y \\ 1-\theta & \text{otherwise} \end{cases} , \tag{8.6}$$

where $0 \leq \theta \leq 1$ specifies the assumed level of *label noise*. This label noise is conceptually different from what is often referred to as *feature noise* that affects input vectors $\vec{x} \in \mathcal{X}$ and may result from imprecise measurements of real-valued features. Soft-margin variants of SVM and perceptron learning discussed in Section 2.1 specifically address feature noise by penalising examples for violating classification constraints. In contrast, label noise resulting from random flips of labels requires a different treatment as expressed in the likelihood (8.6). For the whole sample $\boldsymbol{z}$ the corresponding likelihood is given by

$$\mathcal{L}[\mathbf{w}; \boldsymbol{z}] = \theta^{m\hat{R}[h,\boldsymbol{z}]} (1-\theta)^{m\left(1-\hat{R}[h,\boldsymbol{z}]\right)} . \tag{8.7}$$

The resulting posterior $\mathbf{P}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}$ is piecewise constant and it is this type of probability distribution for which the kernel Gibbs sampler is suitable. In order to get an idea of what the posterior looks like consider Figure 8.2. For small sample size, $m = 10$, the posterior is multi-modal due to the label noise. For more samples, $m = 1000$, the posterior is smoothed out.

8.3.3. Kernel Gibbs Sampling

In order to give a simple explanation of how the kernel Gibbs sampler works, consider the zero-noise case, $\theta = 0$, in which we only sample from a uniform density in version space. The sampling strategy starts at an initial point $\mathbf{w}_0$ and samples a random direction $\mathbf{v}$. The subsequent point $\mathbf{w}_{j+1}$ is sampled uniformly on the intersection of version space $V(\boldsymbol{z})$ with the line characterised by the direction $\mathbf{v}$ and passing through the previous sample point $\mathbf{w}_i$. More precisely and taking into account the curvature of the sphere we proceed for a given value of the noise level θ and an arbitrary starting point $\mathbf{w}_0 \in \mathcal{W}$ as follows (see also Figure 8.3):

1. Choose a random starting point $\mathbf{w}_0 \in \mathcal{W}$. Set $j = 0$.
2. Choose a random direction $\mathbf{v} \in \mathcal{W}$ in the tangent space

$$\mathbf{v} \in \{\tilde{\mathbf{v}} \in \mathcal{W} \mid \langle \tilde{\mathbf{v}}, \mathbf{w}_j \rangle = 0\} . \tag{8.8}$$

   .

3. Calculate the $2m$ angular distances ζ_i from the current position $\mathbf{w}_j$ by

$$\zeta_i = \arccos\left(\frac{\langle \mathbf{v}, \mathbf{x}_i \rangle}{\|\mathbf{x}_i\|}\right) , \quad \zeta_{m+i} \equiv \zeta_i + \pi$$

4. Sort the ζ_i in ascending order (resulting in a permutation $\pi : \{1, \ldots, 2m\} \to \{1, \ldots, 2m\}$ such that $\zeta_{\pi(i)} \leq \zeta_{\pi(i)+1}$ for all $i \in \{1, \ldots, 2m-1\}$).

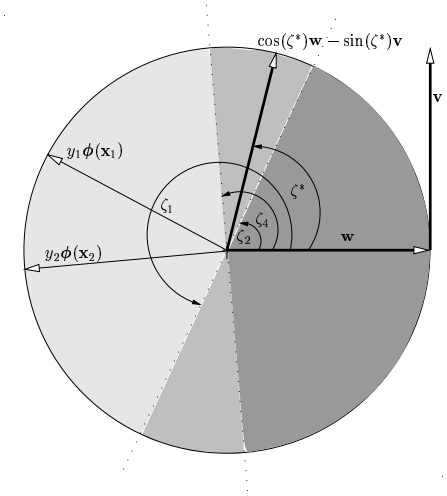


Figure 8.3.: Schematic view of the kernel Gibbs sampling procedure. Two data points $y_1\mathbf{x}_1$ and $y_2\mathbf{x}_2$ divide the space of normalised weight vectors $\mathbf{w} \in \mathcal{W}$ into four equivalence classes with different posterior density indicated by the gray shading. In each iteration, starting from $\mathbf{w}_j$ a random direction $\mathbf{v}$ with $\mathbf{v} \perp \mathbf{w}_j$ is generated. We sample ζ^* from the piecewise constant density on the great circle determined by the plane defined by $\mathbf{w}_j$ and $\mathbf{v}$. In order to obtain ζ^* we calculate the $2m$ angles ζ_i where the training samples intersect with the circle and keep track of the number $m \cdot e_i$ of training errors for each region i.

5. Calculate the empirical risks e_i of the $2m$ intervals $\left[\zeta_{\pi(i)}, \zeta_{\pi(i+1)}\right]$ by evaluating

$$e_i = R_{\text{emp}}\left[h_{\mathbf{m}_i}, \boldsymbol{z}\right], \quad \mathbf{m}_i = \cos\left(\frac{\zeta_{\pi(i+1)} - \zeta_{\pi(i)}}{2}\right)\mathbf{w}_j - \sin\left(\frac{\zeta_{\pi(i+1)} - \zeta_{\pi(i)}}{2}\right)\mathbf{v},$$

with $\zeta_{\pi(2m+1)} \equiv \zeta_{\pi(1)}$.

6. Sample an angle ζ^* using the piecewise constant density

$$\mathbf{f}(\zeta) \propto \sum_{i=1}^{2m} \mathbf{I}_{\zeta_{\pi(i)} \leq \zeta \leq \zeta_{\pi(i+1)}} \theta^{me_i} (1-\theta)^{m(1-e_i)} . \tag{8.9}$$

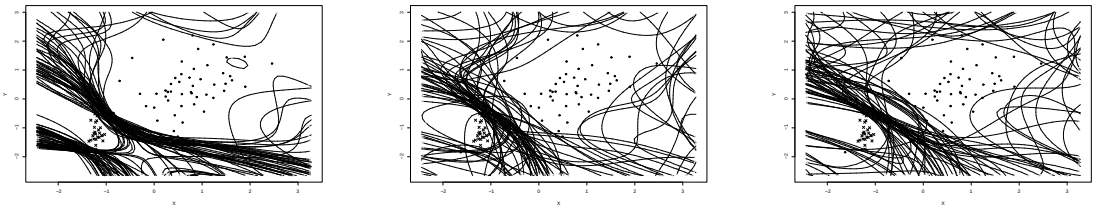
7. Calculate a new sample $\mathbf{w}_{j+1}$ by $\mathbf{w}_{j+1} = \cos(\zeta^*)\mathbf{w}_j - \sin(\zeta^*)\mathbf{v}$.

8. Set $j \leftarrow j+1$ and go back to step 2.

Pseudocode for the kernel Gibbs sampler can be found in Appendix C.4 where we use the expansions

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i , \quad \mathbf{v} = \sum_{i=1}^{m} \beta_i \mathbf{x}_i , \quad \mathbf{m} = \sum_{i=1}^{m} \gamma_i \mathbf{x}_i .$$

In Figure 8.4 we show an application of the kernel Gibbs sampler to some toy data in $\mathbb{R}^2$. As can be seen from these plots, increasing the noise level θ leads to more diverse classifiers on the training sample $\boldsymbol{z}$ including classifiers with non-zero empirical risk.



$\theta = 0.0$ $\theta = 0.1$ $\theta = 0.2$

Figure 8.4.: Illustration of the effect of the label noise level θ on a sample of non-linear decision boundaries. Artificial data of two classes ("x" and "dot") where generated in input space $\mathcal{X} = \mathbb{R}^2$. A collection of 50 samples $\mathbf{w}_j$ were drawn using the kernel Gibbs sampler with an RBF kernel $k(\vec{x}_1, \vec{x}_2) = \exp\left(-\frac{1}{2}\|\vec{x}_1 - \vec{x}_2\|\right)$ and three different noise levels $\theta = \{0.0, 0.1, 0.2\}$. Shown are the resulting decision boundaries in $\mathcal{X}$. The higher the noise level θ the more likely we find decision boundaries with non-zero empirical risk.

8.3.4. Discussion of the Kernel Gibbs Sampler

The kernel Gibbs sampler is a very flexible sampling tool for linear classifiers in kernel spaces.

1. It is suitable for sampling based on a uniform prior $\mathbf{P_W} = \mathbf{U_W}$ over $\mathcal{W}$ and a likelihood that leads to a piecewise constant posterior density $\mathbf{f_W}$ over $\mathcal{W}$. Other functional forms of the density appear to be more difficult to address in this framework because the resulting density on the great circle described by $\mathbf{w}$ and $\mathbf{v}$ is then more difficult to determine and hence more difficult to sample from.

2. The memory requirements of the kernel Gibbs sampler are essentially the same as those for the kernel billiard sampler, i.e. of order $\mathcal{O}\left(m^2 + Nm\right)$.

3. The computational complexity of the kernel Gibbs sampler is of order $\mathcal{O}\left(2Nm^2\right)$ w.r.t. to the kernel operations, including the calculation of the empirical risk of m classifiers for determining the piecewise constant density. An extra computational effort of $\mathcal{O}\left(2m \log 2m\right)$ is required for sorting the angles ζ_i. This, however, could be done in a more efficient but less transparent way by traversing the great circle of interest and adding and subtracting empirical errors when crossing constraints given by training examples.

4. Again the questions of ergodicity and mixing time of the kernel Gibbs sampler are difficult to answer. The experimental comparison with the permutational Gibbs sampler in the subsequent section, Section 8.4, is of interest in this regard because the permutational perceptron sampler employs a completely different method as compared to the kernel Gibbs sampler. From a theoretical point of view it seems

promising to consider corresponding analyses for standard Gibbs sampling (Neal 1993) or slice sampling (Neal 1997a).

8.4. The Permutational Perceptron Sampler

The previous pseudo Markov chain and MCMC methods cannot avoid a rather high computational complexity as a result of the fact that they need to consider all the constraints given by the training examples for every sample. Also, the sequential nature of these algorithms makes their performance depending on their mixing properties. Figuratively speaking, the "ball" may get stuck in a corner of the sampling space and thus lead to a considerable bias in the resulting sample. Both these problems are addressed by a heuristic that we refer to as the permutational perceptron sampler. The idea originally goes back to a theoretical consideration by Watkin (1993): Obtain different samples from version space by running a perceptron on different permutations of the training sample.

8.4.1. Sampling Version Space with Many Training Patterns

The classical perceptron learning algorithm offers the possibility to obtain many different classifiers in version space simply by learning on different permutations of the training sample. Given a permutation $\pi : \{1, \ldots, m\} \to \{1, \ldots, m\}$ the perceptron algorithm works as follows:

1. Start with $\mathbf{w}_0 = \mathbf{0}$ and $t = 0$.
2. For all $i \in \{1, \ldots, m\}$, if $y_{\pi(i)} \left\langle \mathbf{x}_{\pi(i)}, \mathbf{w}_t \right\rangle < 0$ then $\mathbf{w}_{t+1} = \mathbf{w}_t + y_{\pi(i)} \mathbf{x}_{\pi(i)}$ and $t \leftarrow t + 1$.
3. Stop, if for all $i \in \{1, \ldots, m\}$ classification is correct, i.e., $y_{\pi(i)} \left\langle \mathbf{x}_{\pi(i)}, \mathbf{w}_t \right\rangle \geq 0$.

The perceptron convergence theorem, Theorem 42, bounds the number of update steps until convergence for every permutation Π of the training sample in terms of the maximum margin $\gamma^* (\mathbf{w}^*, \boldsymbol{z})$ achievable on the training sample. Interestingly, this quantity influences the permutational perceptron sampler in two ways. On the one hand, higher sparseness as indicated by a large margin reduces the number of *different* classifiers that can be obtained by permuting the training sample. On the other hand, high sparseness makes the algorithm computationally more efficient.

Algorithmically, we can benefit from this sparseness by the following "trick": Since

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i$$

all we need to store is the m-dimensional vector $\boldsymbol{\alpha}$. Furthermore, we keep track of the m-dimensional vector $\mathbf{o}$ of real valued outputs

$$o_i = y_i \left\langle \mathbf{x}_i, \mathbf{w}_t \right\rangle = \sum_{j=1}^{m} \alpha_j k \left(x_i, x_j \right)$$

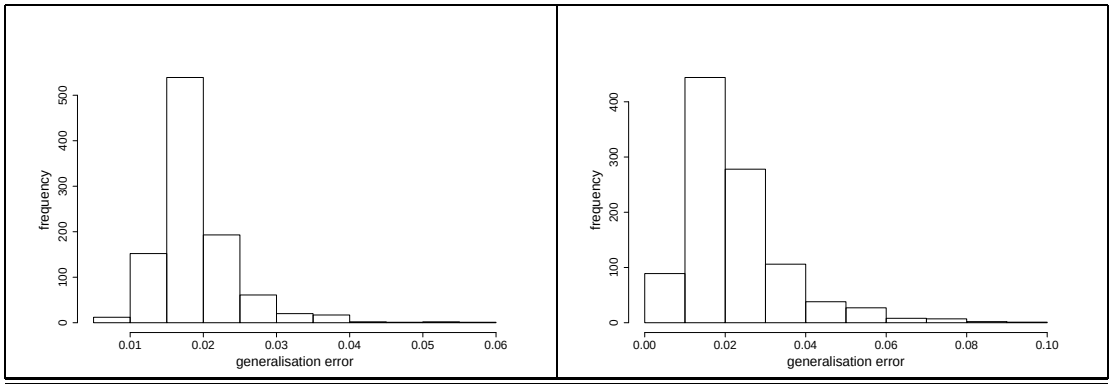


Figure 8.5.: Histograms of generalisation errors estimated on a test sample using **(a)** the kernel Gibbs sampler. **(b)** the permutational perceptron sampler.

of the current solution at the ith training point. By definition, in the beginning $\boldsymbol{\alpha} = \mathbf{o} = \mathbf{0}$. Now, if $o_i < 0$ we update α_i by $\alpha_i + y_i$ and update $\mathbf{o}$ by $o_j \leftarrow o_j + y_i k(x_i, x_j)$ which requires only m kernel calculations. Pseudocode for the permutational perceptron sampler in the dual representation and applying the aforementioned shortcut can be found in Appendix C.5.

8.4.2. Empirical Evidence for Uniform Sampling

It is not at all clear if the above heuristics leads to a *uniform* sampling of version space. In fact, geometrically, the distribution of solutions obtained are hard to characterise. One way to think about the resulting classifiers in their dual representation is to consider a fixed level of sparseness $\|\boldsymbol{\alpha}\|_0$. Since the perceptron algorithm produces only integer coefficients α_i these can be thought of as lying on a lattice in version space.

In order to investigate the usefulness of the permutational perceptron sampler experimentally, we compared the distribution of prediction errors of samples obtained by perceptron learning on permuted training sets with samples obtained by the kernel Gibbs sampler at noise level $\theta = 0$. For computational reasons, we only used 188 training patterns and 453 test patterns of the classes "1" and "2" from the MNIST data set (LeCun 1998). In Figure 8.5 we plotted the distribution of prediction errors for (a) the kernel Gibbs sampler and (b) the permutational perceptron sampler over 1000 random samples using the kernel

$$k(\vec{x}_1, \vec{x}_2) = (\langle \vec{x}_1, \vec{x}_2 \rangle + 1)^5 \ , \tag{8.10}$$

that had shown excellent generalisation performance when using the support vector machine. Using a quantile-quantile (QQ) plot technique we can compare both distributions in one graph (see Figure 8.6). These plots suggest that by simple permutation of the training sample we are able to obtain a sample of classifiers exhibiting a similar distribution of prediction errors as with time-consuming Gibbs sampling.

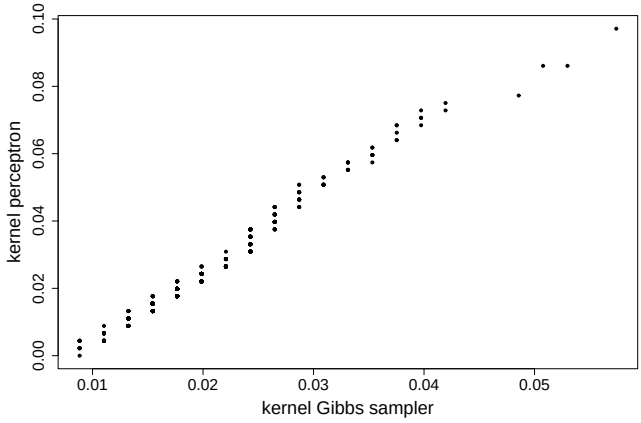


Figure 8.6.: QQ plot of distributions (a) and (b) from Figure 8.5. The straight line indicates that both distributions have a similar decay rate.

8.4.3. Discussion of the Permutational Perceptron Sampler

The permutational perceptron sampler is a fast heuristic for sampling kernel classifiers from version space.

1. It is not clear what kind of distribution of samples exactly the permutational perceptron sampler returns. All we know is that the distribution of resulting generalisation errors does not deviate very much from the corresponding distribution of the kernel Gibbs sampler.

2. The memory requirement of the permutational perceptron sampler is $\mathcal{O}(2m + Nm)$ unless the samples are used online and discarded at once, in which case we have $\mathcal{O}(2m)$ only.

3. Interestingly, the number of kernel evaluations is not more than $\mathcal{O}(d(\boldsymbol{\alpha})mN)$ where $d = \|\boldsymbol{\alpha}\|_0$ is data-dependent. As a consequence, the computational requirement of this algorithm is no more than the computational requirement for N evaluations of the margin $\gamma(\mathbf{w}, \boldsymbol{z})$ of a classifier!

4. The limitation to sampling from version space can in principle be overcome by stopping the perceptron at solutions with non-zero training error. In comparison to the kernel Gibbs sampler, however, it might prove difficult to exactly control the sampling w.r.t. a certain non-constant posterior.

8.5. Conclusions

In this chapter we have presented three different schemes for sampling kernel classifiers under a uniform prior over $\mathcal{W}$. As will be seen in the subsequent chapter, these sampling schemes prove useful in obtaining good classifiers in a Bayesian framework.

The kernel billiard sampler is based on a deterministic but chaotic dynamical system and is restricted to version space. This restriction, however is not as severe as it appears because feature noise can be taken care of by an appropriate change of kernel as discussed in Remark 1. The kernel Gibbs sampler is able to sample in the presence of label noise at the cost of higher computational costs. It is therefore mainly useful as an analytic tool to study the properties of kernels in a Bayesian framework. Finally, the permutational perceptron sampler is an efficient heuristic that makes use of the high efficiency of the perceptron algorithm due to data sparseness as discussed in Chapters 5 and 6. Future work may aim at developing efficient sampling schemes for noise models other than the one presented here.

9. Numerical Experiments for Classification by Sampling

This chapter serves to illustrate how the sampling schemes presented in the previous chapter, Chapter 8, can be applied to the Bayesian classification algorithms in kernel space presented in Chapter 7. The experiments are intended not so much as a comprehensive study but rather as an illustration of the specific properties of the algorithms under consideration. As a consequence, results are presented only for a selection of benchmark problems with an emphasis on conceptual issues rather than experimental exhaustion.

The section on Bayesian transduction, Section 9.1, is based on Graepel, Herbrich, and Obermayer (1999) and Graepel, Herbrich, and Obermayer (2000). The experimental section on Bayes point machines, Section 9.2, is based on Herbrich and Graepel (2001a) and Herbrich, Graepel, and Campbell (2001). Finally, the section on the distribution of prediction errors and margins, Section 9.3, is based on Herbrich, Graepel, and Williamson (2001).

9.1. Experimental Results for Bayesian Transduction

We focused on the confidence c_{td} Bayesian transduction provides together with the prediction $\tilde{y}$ of the label. If the confidence c_{td} reflects reliability of a label estimate at a given test point then rejecting those test points whose predictions carry low confidence should lead to a reduction in generalisation error on the remaining test points. In the experiments we varied a rejection threshold Θ between $[0,1]$ thus obtaining for each Θ a rejection rate together with an estimate of the generalisation error at non-rejected points. Both these curves were linked by their common Θ-axis resulting in a plot prediction error versus rejection.

We used the UCI (Blake and Merz 1998) data sets `thyroid` and `heart` because they are medical applications for which the confidence of single predictions is particularly important. Also a high rejection rate due to too conservative a confidence measure may incur considerable costs. We trained a support vector machine using RBF kernels $k(\vec{x}_1, \vec{x}_2) = \exp(-\left\|\vec{x}_1 - \vec{x}_2\right\|^2 / 2\sigma^2)$ with σ chosen such as to insure the existence of a version space. We used 100 different training samples obtained by random 60%:40% splits of the whole data set. The margin $\gamma(\mathbf{w}, z_i)$ of each test point was calculated as a confidence measure of SVM classifications. For comparison we determined the labels $\tilde{y}$ and resulting confidences c_{td} using the Bayesian transduction algorithm based

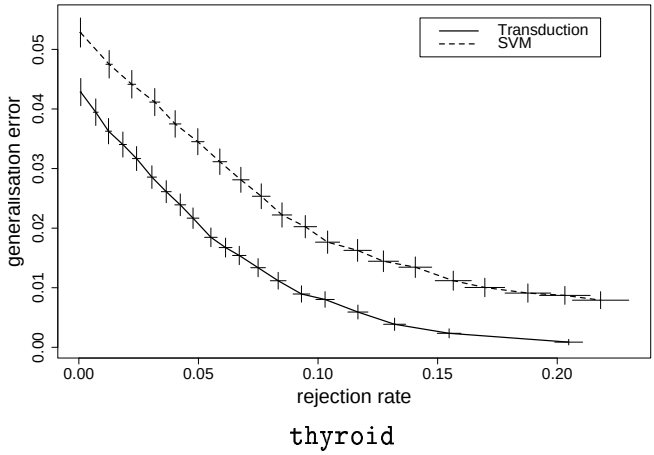


Figure 9.1.: Estimated prediction error error as a function of the rejection rate Θ for Bayesian transduction and SVMs for the `thyroid` data set ($\sigma = 3$) The error bars in both directions indicate one standard deviation of the estimated means. The upper curve depicts the result for the SVM algorithm; the lower curve is the result obtained by Bayesian transduction.

on the kernel billiard sampler presented in Section 8.2 with the same value of the kernel parameter. Since the rejection for the Bayesian transduction was in both cases higher than for SVMs at the same level Θ we determined $\Theta_{\max}$ which achieves the same rejection rate for the SVM confidence measures as Bayesian Transduction achieves at $\Theta = 1$ (`thyroid`: $\Theta_{\max} = 2.15$, `heart`: $\Theta_{\max} = 1.54$). The results for the two data sets are depicted in Figures 9.1 and 9.2.

In the `thyroid` example, Figure 9.1, one can see that c_{td} is indeed an appropriate indicator of confidence: at a rejection rate of approximately 20% the generalisation error approaches zero at minimal variance. For any desired generalisation error Bayesian Transduction needs to reject significantly less examples of the test sample as compared to SVM classifiers, e.g. 4% less at 2.3% generalisation error. The results of the `heart` data set, Figure 9.2, show even more pronounced characteristics w.r.t. to the rejection rate. Note that those confidence measures considered cannot capture the effects of noise in the data which leads to a prediction error of 16.4% even at maximal rejection $\Theta = 1$ corresponding to the Bayes error under the given function class.

9.2. Experimental Results for the Bayes Point Machine

Many experimental results about the Bayes point machine based on the kernel billiard sampler are provided in Herbrich, Graepel, and Campbell (2001). In this section we will present some results for the Bayes point machine based on the permutational perceptron sampler and the kernel Gibbs sampler. In Subsection 9.2.1 we address the problem of

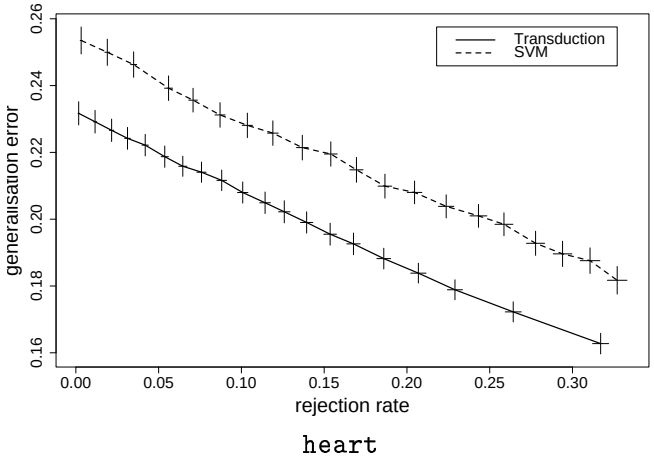


Figure 9.2.: Estimated prediction error as a function of the rejection rate Θ for Bayesian Transduction and SVMs for the `heart` data set ($\sigma = 10$). The error bars in both directions indicate one standard deviation of the estimated means. The upper curve depicts the result for the SVM algorithm; the lower curve is the result obtained by Bayesian transduction.

estimating the centre of mass for large-scale classification problems involving up to 60 000 training examples in up to 784 dimensions by the permutational perceptron sampler. In Subsection 9.2.2 we experimentally explore the performance of the centre of mass in the presence of label noise when estimated based on the kernel Gibbs sampler applied to a piecewise constant posterior.

9.2.1. Handwritten Digit Recognition

For both tasks we now consider our inputs are $n_\mathrm{p} \times n_\mathrm{p}$ pixel grey value images which were transformed into n_p^2-dimensional vectors by concatenation of the rows. The grey values were taken from the set $\{0, \ldots, 255\}$. All images were labelled by one of the ten classes "0" to "9". For each of the ten classes $y = \{0, \ldots, 9\}$ we ran the perceptron algorithm $N = 10$ times each time labelling all training points of class y by $+1$ and the remaining training points by -1. On a Pentium III 500 MHz with 128 MB memory each learning trial took $10 - 20$ minutes (MNIST) or $1 - 2$ minutes (USPS), respectively. We made use of the fact that $\approx 40\%$ of the grey values of each image are 0 since they encode background. Therefore, we encoded each image as an index-value list which allows much faster computation of the inner products $\langle \vec{x}_i, \vec{x}_j \rangle$ and speeds up the algorithm by a factor of 2–3.

For the classification of a test image x we calculated the real-valued output of all 100

different classifiers by

$$f_i(x) = \frac{\langle \mathbf{x}, \mathbf{w}_i \rangle}{\|\mathbf{w}_i\| \, \|\mathbf{x}\|} = \frac{\sum_{j=1}^{m} (\boldsymbol{\alpha}_i)_j \, k(x_j, x)}{\sqrt{\sum_{j=1}^{m} \sum_{l=1}^{m} (\boldsymbol{\alpha}_i)_j (\boldsymbol{\alpha}_i)_l \, k(x_j, x_l)} \sqrt{k(x,x)}} ,$$

where we used the inhomogeneous polynomial kernel k given by

$$k(\vec{x}_1, \vec{x}_2) = (\langle \vec{x}_1, \vec{x}_2 \rangle + 1)^5 . \tag{9.1}$$

For notational simplicity we assume that the first N classifiers are classifiers for the class "0", the next N for class "1" and so on. The symbol $(\boldsymbol{\alpha}_i)_j$ refers to the expansion coefficient corresponding to the i-th classifier and the j-th training example. Now, for each of the ten classes we calculated the real-valued decision of the Bayes point estimate $\widehat{\mathbf{w}}_{\mathrm{cm},y}$ by

$$f_{\mathrm{bp},y}(x) = \langle \mathbf{x}, \widehat{\mathbf{w}}_{\mathrm{cm},y} \rangle = \frac{1}{N} \sum_{i=1}^{N} \langle \mathbf{x}, \mathbf{w}_{i+yN} \rangle ,$$

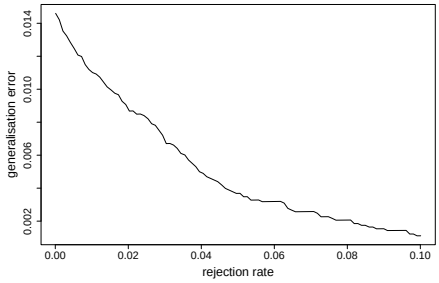
where the label y ranges from $\{0, \ldots, 9\}$. In a Bayesian spirit, the final decision was carried out by

$$h_{\mathrm{bp}}(x) = \mathrm{argmax}_{y \in \{0,\ldots,9\}} \, f_{\mathrm{bp},y}(x) .$$

Note that $f_{\mathrm{bp},y}(x)$ can be interpreted as an (unnormalised) approximation of the posterior probability that x is of class y when restricted to the hypothesis space $\mathcal{H}_{\mathcal{K}}$ (see Platt (2000)). In order to test the dependence of the prediction error on the quantity $\max_y f_{\mathrm{bp},y}(x)$ we fixed a certain rejection rate $\Theta \in [0,1]$ and rejected the set of $\Theta \cdot 10\,000$ test points with the smallest value of $\max_y f_{\mathrm{bp},y}(x)$.

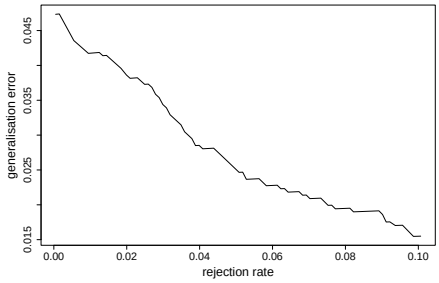
MNIST Handwritten Digits In the first of our large scale experiment we used the full MNIST dataset with 60 000 training examples and 10 000 test examples of 28×28 grey value images of handwritten digits (LeCun 1998). The plot resulting from learning only $N = 10$ consistent classifiers per class and rejection based on the real-valued output of the single Bayes points is depicted in Figure 9.3. As can be seen from this plot, even without rejection the Bayes point has excellent generalisation performance when compared to support vector machines which achieve a generalisation error of 1.4%. The commonly known result of 1.1% with the kernel (9.1) and a polynomial degree of four could not be reproduced and is thus considered invalid (personal communication with P. Haffner). Note also that the best results with support vector machines were obtained when using a soft margin. Furthermore, rejection based on the real-valued output $f_{\mathrm{bp}}(x)$ turns out to reduce the prediction error to 0.1%. One should bear in mind that the learning time for this simple algorithm was comparable to that of support vector machines which need ≈ 8 hours per digit (Platt 1998, p. 201, Table 12.2).

USPS Handwritten Digits In the second of our large scale experiments we used the USPS dataset with 7 291 training examples and 2 007 test examples of 16×16 grey value images of handwritten digits. The resulting plot of the generalisation error when



rejection rate	generalisation error
0%	1.46%
1%	1.10%
2%	0.87%
3%	0.67%
4%	0.49%
5%	0.37%
6%	0.32%
7%	0.26%
8%	0.21%
9%	0.14%
10%	0.11%

Figure 9.3.: Prediction error as a function of the rejection rate Θ for the MNIST data set. The support vector machine achieved 1.4% without rejection as compared to 1.46% for the Bayes point machine. Note that by rejection based on the real-valued output the prediction error could be reduced to 0.1% indicating that this measure is related to the probability of misclassification of single test points.



rejection rate	generalisation error
0%	4.73%
1%	4.17%
2%	3.81%
3%	3.39%
4%	2.80%
5%	2.47%
6%	2.28%
7%	2.09%
8%	1.95%
9%	1.75%
10%	1.61%

Figure 9.4.: Prediction error as a function of the rejection rate for the USPS data set. The hard margin support vector machine achieved 4.6% without rejection as compared to 4.73% for the Bayes point machine.

rejecting test examples based on the real-valued outputs of the single Bayes points is shown in Figure 9.4. Again, the resulting classifier has a prediction error performance comparable to that of support vector machines whose best results are 4.5% when using a soft margin and 4.6% in the hard margin scenario. In Figure 9.5 we plotted the 25 most commonly used images $x_i \in \boldsymbol{x}$ with non-zero coefficients $(\boldsymbol{\alpha}_j)_i$ across the 100 different classifiers learned. Though no margin maximisation was performed it turns out that in accordance with the "support vector philosophy" these are the hard patterns in the datasets with respect to classification. Moreover, as can be seen from examples 1, 6, and 8 there is clearly noise in the dataset which could potentially be taken into account using the techniques outlined in Remark 1 at no extra computational costs.

9.2.2. The BPM with a Classification Noise Model

We investigated the generalisation performance of the Bayes point machine in the case of label noise. In $\mathbb{R}^3$ we generated 100 random training and test sets of size $m_{\text{train}} = 100$ and $m_{\text{test}} = 1000$, respectively. For each normalised point $\vec{x} \in \mathbb{R}^3$ the longitude and latitude were sampled from a $\texttt{Beta}(5,5)$ and $\texttt{Beta}(0.1,0.1)$ distribution, respectively. The classes y were obtained by randomly flipping the classes assigned by the classifier $h_{\mathbf{w}^*}$ with $\mathbf{w}^*$ located at spherical coordinates $\left(\frac{\pi}{2}, \pi\right)$ (see also Figure 8.2) with a true label flip rate of $\theta^* = 5\%$. In Figure 9.6 we plotted the estimated prediction error for a BPM (trained using 100 samples $\mathbf{w}_j$ from the kernel Gibbs sampler) and 2-norm soft-margin SVM at different label noise levels θ and margin slack penalisation λ, respectively. Clearly, the BPM with the correct noise model outperformed the SVM irrespective of the chosen level of regularisation. Interestingly, the BPM appears to be quite "robust" w.r.t. the choice of the label noise parameter q. However, these are only preliminary results and it would be interesting to explore classification in the presence of label noise further.

9.3. Distribution of Prediction Errors and Margins

Based on the MNIST dataset (LeCun 1998) for images of "1" and "2" we generated well-balanced training and test samples of size 118 and 453, respectively. In order to explore the structure of version space we were interested in the distribution of prediction errors (estimated on the given test sample) and its relation to the attained normalised margin $\Gamma(\mathbf{w}, \boldsymbol{z})$.

9.3.1. Distribution of Prediction Errors

In Figure 9.7 we plotted the distribution of generalisation errors for $N = 10000$ samples $\mathbf{w}$ using different degrees of the polynomial kernel

$$k(\vec{x}_1, \vec{x}_2) = (\langle \vec{x}_1, \vec{x}_2 \rangle + 1)^p \,, \tag{9.2}$$

which produced excellent classifiers when used in SVM learning ($p_{\text{opt}} = 5$). The samples were obtained using the kernel Gibbs sampler described in Section 8.3. In order to reduce

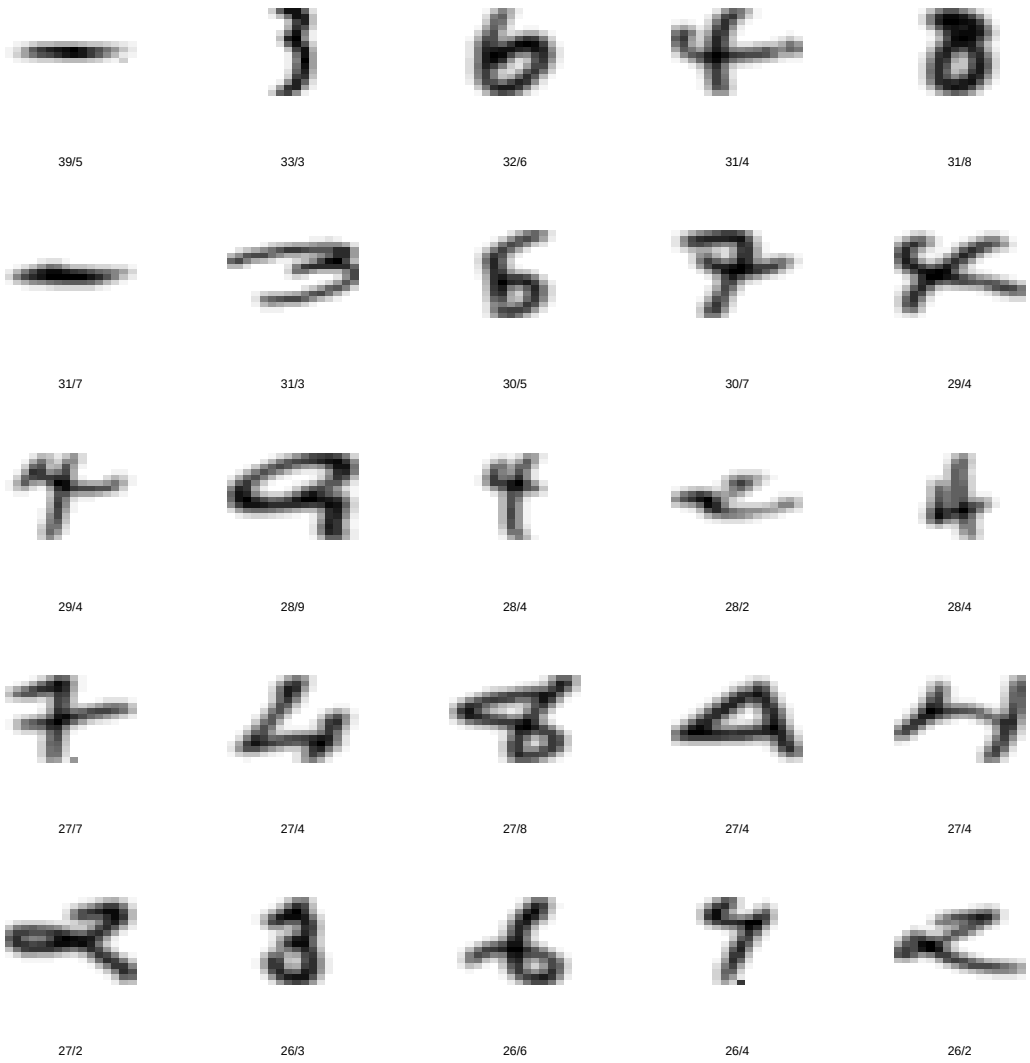


Figure 9.5.: Shown are the 25 most commonly used examples $x_i \in \boldsymbol{x}$ (non-zero coefficients $(\boldsymbol{\alpha}_j)_i$ for many $j \in \{1, \ldots, 100\}$) from the USPS dataset across the 100 different classifiers learned using the perceptron learning algorithm. The two numbers below each digit give the number of classifiers they appeared in and the true class $y \in \{0, \ldots, 9\}$ in the training sample. Interestingly, in accordance with the philosophy behind support vectors these are the "hardest" patterns with respect to the classification task although no explicit margin maximisation was performed.

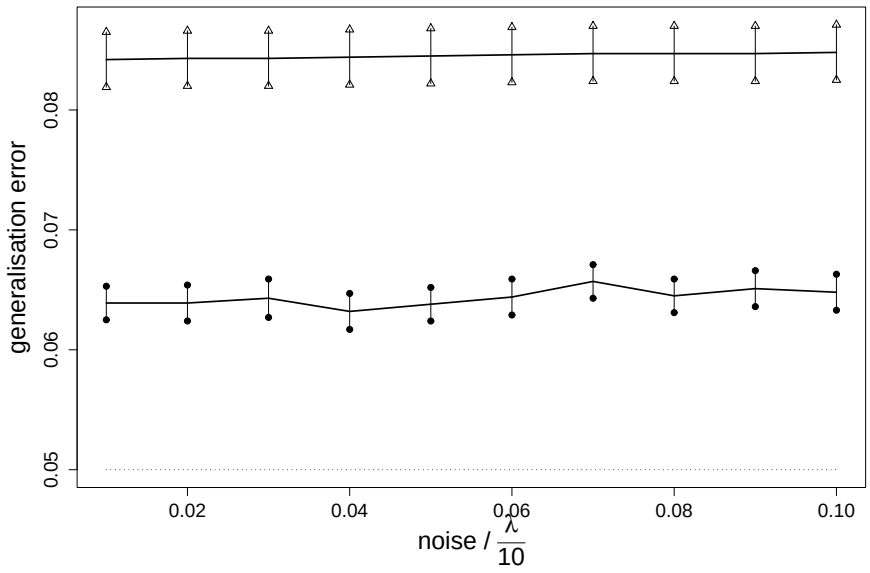


Figure 9.6.: Comparison of BPMs and SVMs on data contaminated by label noise. Shown are the estimated prediction errors of Bayes point machines (circled error-bars) and soft-margin SVMs (triangled error-bars) as functions of the assumed noise level θ and margin slack penalisation λ, respectively. The dataset consisted of $m = 100$ observations with a label noise of 5% (dotted line) and we used $k(\vec{x}_1, \vec{x}_2) = \langle \vec{x}_1, \vec{x}_2 \rangle + \lambda \cdot \mathsf{I}_{\vec{x}_1 = \vec{x}_2}$. Note that the abscissa is jointly used for θ and λ.

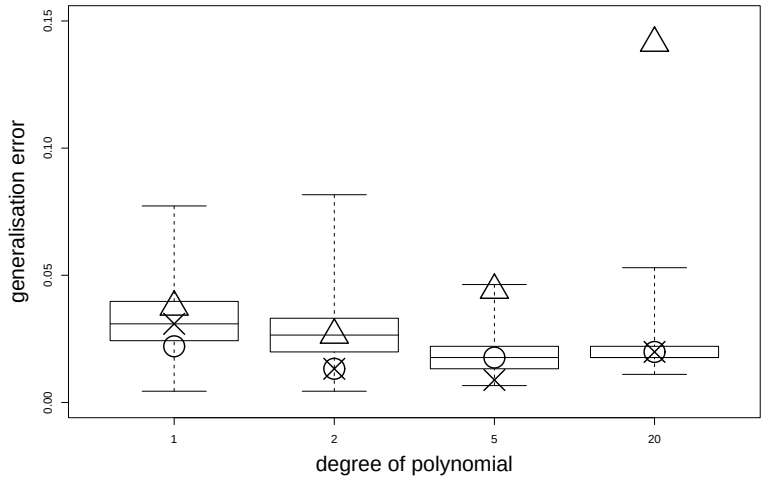


Figure 9.7.: Box plots of distributions of generalisation errors for $l = 1000$ samples $\mathbf{w}$ using different degrees in the polynomial kernel (9.2). The $\triangle$, $\times$ and $\circ$ depict the generalisation errors of the SVM solution ($\triangle$), the SVM solution when normalising in feature space $\mathcal{K}$ ($\times$) and the Bayes point machine solution ($\circ$), respectively.

dependencies between successive samples $\mathbf{w}$ of the Markov chain we used only one in ten samples thus effectively having $N_{\text{eff}} = 1000$ samples.

For any value of p considered there are at least 50% of consistent classifiers whose generalisation error is less than the one found by the SVM ($\triangle$). This empirical result can be seen as further support for the egalitarian bound, Theorem 39. Surprisingly, with increasing polynomial degree p the variance of the distribution keeps decreasing while only a small increase of its mean can be observed beyond degree 5. Furthermore, using the Bayes point machine algorithm that returns the "centre of mass" of version space $V(\boldsymbol{z})$ (see Section 7.2) or the SVM on the normalised training sample in feature space $\mathcal{K}$ we seem to be able to find classifiers always within the best 50% ($\circ$ and $\times$). Both these algorithms aim at finding a solution at the "centre" of version space $V(\boldsymbol{z})$ in the sense of $\Gamma(\mathbf{w}, \boldsymbol{z})$ .Moreover, the SVM applied to data that were not normalised in $\mathcal{K}$ appears to have considerable higher prediction error. This observation further supports the conjecture that the normalised margin $\Gamma(\mathbf{w}, \boldsymbol{z})$ as defined in Definition 55 is in fact the quantity that should be maximised. The support vector machine without normalisation maximises the standard margin $\gamma(\mathbf{w}, \boldsymbol{z})$ and may thus lead to classifiers at the very edge of version space. The deterioration of the SVM with increasing degree of the polynomial kernel shows how this effect increases with increasing dimensionality of the feature space.

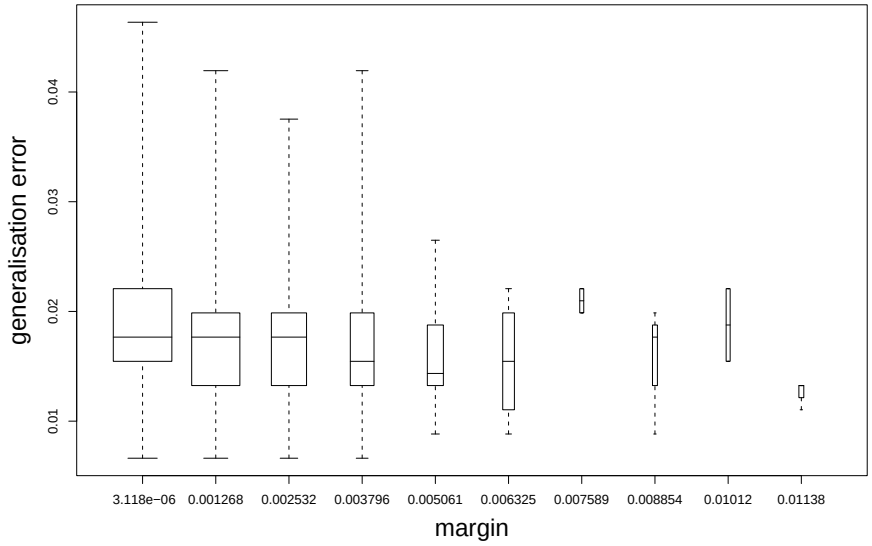


Figure 9.8.: Box plots of distributions of prediction errors for different attained margins $\Gamma(\mathbf{w}, \boldsymbol{z})$ when using a polynomial kernel of degree $p = 5$. The width of each box plot is proportional to the number of samples on which it is based.

9.3.2. Prediction Error versus Margin

In Figure 9.8 we additionally provide the distributions of prediction errors for given attained normalised margins $\Gamma(\mathbf{w}, \boldsymbol{z})$. As expected, *almost all of the classifiers $h_{\mathbf{w}}$ with a large margin $\Gamma(\mathbf{w}, \boldsymbol{z})$ do have a small prediction error $R[h_{\mathbf{w}}]$*. The plot also clarifies that large margins are only a *sufficient (probabilistic) condition* for good generalisation ability and that there exist many consistent classifiers with low prediction error despite of their small margins. This is again in accordance with the egalitarian bound, Theorem 39, in particular when keeping in mind that in high-dimensional feature spaces $\mathcal{K}$ the uniform measure over volumes is concentrated near the edges. Hence, most of the classifiers in version space $V(\boldsymbol{z})$ *do have* a small margin (see width of the box plots) albeit exhibiting good generalisation.

9.4. Conclusions

The three sampling methods presented in Chapter 8 provide practical means for the application of Bayesian learning in kernel spaces as well as for exploring the distribution of prediction error of classifiers sampled from $\mathcal{H}_{\mathcal{K}}$. This chapter was meant to give a flavour of the results that can be obtained using these sampling schemes in kernel space.

It was shown that Bayesian transduction can be carried out in kernel space yielding impressive results not so much with regard to standard classification, but in particular when used for the rejection of test examples. While this is not a really practical method

for large scale problems it constitutes true Bayesian inference and as such indicates the power of these methods.

When computational efficiency is at stake—both at training and test time—the large scale Bayes point machine based on permutational perceptron sampling appears as a good candidate. Again, the existence of a valid rejection criterion for test points whose classification is uncertain gives this method an edge over the more traditional SVM approach. Refining the Bayes point machine by the incorporation of label noise into the likelihood based on the kernel Gibbs sampler shows some promise, but it is clear that these results are not conclusive and that further experiments would be required to confirm the conjecture that an appropriate noise model can indeed lead to better generalisation.

Finally, we used the kernel Gibbs sampler to explore the error structure of version space. Interestingly, it turned out how crucial an initial normalisation of the data in feature space $\mathcal{K}$ is for successful generalisation. In fact, the take home message of this chapter for practitioners is: Normalise your data in feature space!

10. Machine Learning and Go

The material presented so far was concerned with theory and algorithms of machine learning. One lesson learned was that the correct choice of the model or—in the language of learning theory—the correct choice of the hypothesis space $\mathcal{H}$ is of crucial importance for successful learning. In practice, the question of model selection is intimately related to the choice of representation. Sometimes it is easy to find a good representation, in particular, if the objects to be classified can be represented in terms of natural features that arise from meaningful measurements on the objects of interest. In other domains, in particular, when the objects to be represented are rather complex, it can be very difficult to find a good representation for learning. Symbolic or structured objects are particularly difficult to deal with due to the combinatorial explosion of the number of possible interactions between the parts of objects. Such objects are often best represented in terms of graphs. However, learning on a graph representation is often computationally very complex.

This chapter, which is based on Graepel, Goutrie, Krüger, and Herbrich (2001), is intended as an application of kernel classifiers to objects that are naturally represented in terms of graphs. We consider the game of Go from the point of view of machine learning and as a well-defined domain for learning on graph representations. We discuss the representation of both board positions and candidate moves and introduce the common fate graph (CFG) as an adequate representation of board positions for learning. Single candidate moves are represented as feature vectors with features given by subgraphs relative to the given move in the CFG. Using this representation we train a support vector machine (SVM) and a kernel perceptron to discriminate good moves from bad moves on a collection of life-and-death problems and on 9×9 game records. We thus obtain kernel machines that solve Go problems and play 9×9 Go.

More generally, we argue that kernel classifiers can be applied successfully to a certain class of problems involving structured or symbolic objects—such as board positions in the game of Go. However, many other applications involving the classification of graphs are conceivable and the application to the game of Go should be regarded as an illustration of this approach.

10.1. The Game of Go

Go (Chinese: Wei-Qi, Korean: Baduk) is an ancient oriental game about surrounding territory that originated in China over 4000 years ago. The game has elegantly simple rules that lead to intricate tactical and strategic challenges. Its complexity by far exceeds

that of chess, an observation well supported by the fact that the human world champion of chess found a worthy challenger in the computer program `Deep Blue`, while numerous attempts at reproducing such a result for the game of Go have been unsuccessful: So far no computer program is a serious challenger even for the average Go amateur. As a consequence, Go appears to be an interesting testing ground for machine learning (Burmeister and Wiles 1994). In particular, we consider the problem of *representation* because an adequate representation of the board position is an essential prerequisite for the application of machine learning to the game of Go.

A particularly elegant statement of the rules of Go is due to Tromp and Taylor[1] and is only slightly paraphrased for our purpose:

1. Go is played on a square grid of points, by two players called Black and White.
2. Each point on the grid may be coloured *black*, *white* or *empty*. A point P is said to reach a colour C, if there exists a path of (vertically or horizontally) adjacent points of P's colour from P to a point of colour C. Clearing a colour is the process of emptying all points of that colour that do not reach *empty*.
3. Starting with an empty grid, the players alternate turns, starting with Black. A turn is either a pass; or a move that does not repeat an earlier grid colouring. A move consists of colouring an *empty* point one's own colour; then clearing the opponent colour, and then clearing one's own colour.
4. The game ends after two consecutive passes. A player's score is the number of points of her colour, plus the number of empty points that reach only her colour. The player with the higher score at the end of the game is the winner. Equal scores result in a tie.

10.2. Previous Work on Computer Go

While the majority of computer Go programs work in the fashion of rule-based expert systems (Fotland 1993), several attempts have been made to apply machine learning techniques to Go. Two basic learning tasks can be identified:

1. Learning an evaluation function for board positions
2. Learning an evaluation function for moves in given positions

The first task was tackled in the framework of reinforcement learning in Schraudolph et al. (1994) who learned a pointwise evaluation function by the application of the reinforcement learning algorithm TD (λ) (see Sutton and Barto (1998) for a general introduction to reinforcement learning) with a multi-layer perceptron (MLP) as an approximator for the value function. The second task found an application to *tsume* Go in Sasaki and Sawada (1998) who trained a multi-layer perceptron to find problem-solving moves in a

[1] See `http://www.cwi.nl/~tromp/go.html` for a detailed description.

supervised mode. Considering the complexity of the task it appears to be necessary to eventually combine both these learning tasks into a composite system.

All the known approaches suffer from a rather naive representation of board positions and candidate moves. The board is commonly represented as a simple two-dimensional array and single points or candidate moves are characterised by the pattern of stones in their surroundings. These representations obviously do not take into account the specific structure imposed by the rules of the game. We introduce a new representation for both board positions and candidate moves based on what we call a *common fate graph* (CFG). Our CFG representation builds on ideas first presented by Markus Enzensberger in the context of his Go program `Neurogo II`[2]. While our discussion is focused on the game of Go, our considerations about representation should be of interest with regard to all those domains where the standard feature vector representation does not adequately capture the structure of the learning problem. A natural problem domain that shares this characteristics is the classification of organic chemical compounds represented as graphs of atoms (nodes) and bonds (edges) (Geibel and Wysotzki 1998). The common fate graph (CFG) representation is introduced in Section 10.3 and complemented with the relative subgraph feature (RSF) representation. We demonstrate the effectiveness of the new representation in Section 10.4. We train both a kernel perceptron and a support vector machine on a data-base of life-and-death problems and on 9×9 game records: The resulting classifiers are able to discriminate between good and bad local moves, solve Go problems, and play.

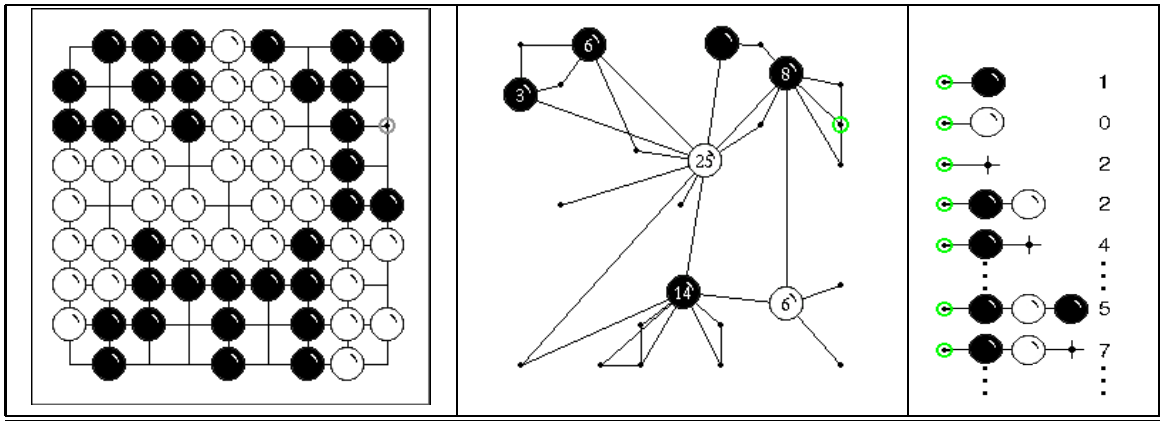
10.3. Representation

As discussed earlier an adequate representation of the learning problem at hand is an essential prerequisite for successful learning. As was pointed out in Section 2.2 one could even go as far as saying that an adequate representation should render the actual learning task trivial. More realistically, we aim at finding a representation that captures the structure of board positions by mapping similar positions (in the sense of the learning problem) to similar representations. Another desirable feature of a representation is a reduction in complexity: Only those features relevant to the learning problem at hand should be retained.

10.3.1. Common Fate Graph

The value of a given Go position is invariant under rotation and mirroring of the board. Also the rules of the game refer essentially only to the local neighbourhood structure of the game. A board position is thus adequately represented by its *full graph representation* (FGR), a graph with the structure of a square grid. In fact, the visual structure of a classical Go board supports this view (see Figure 10.1 (a)). More formally, let us define the full graph representation as an undirected tricoloured connected graph in the sense of the following definition:

[2] "The Integration of A Priori Knowledge into a Go Playing Neural Network" available via `http://www.cgl.ucsf.edu/home/pett/go/Programs/NeuroGo-PS.html`



(a: FGR) (b: CFG) (c: RSF)

Figure 10.1.: Illustration of the feature extraction process: **(a)** board position (FGR), **(b)** corresponding common fate graph (CFG), and **(c)** a selection of extracted relative subgraph features (RSF) w.r.t. the node marked by a gray circle in (a) and (b).

Definition 70 (Full graph representation). We define a full graph representation (FGR) as an undirected tricoloured connected graph given by the triple

$$G_{\text{fgr}} \stackrel{\text{def}}{=} (P, E, c) \ .$$

The set of nodes is given by

$$P \stackrel{\text{def}}{=} \left\{p_1, \ldots, p_{|P|}\right\} \ ,$$

and each node p_i is assigned one of three colours by the colouring function c,

$$c : P \to \{black, white, empty\} \ .$$

The set E,

$$E \subseteq \{(p_i, p_j) \,|\, i, j \in \{1, \ldots |P|\}\} \ ,$$

is a binary, symmetric "edge" relation. It is symmetric in the sense that $(p_i, p_j) \in E \Rightarrow (p_j, p_i) \in E$ for all $i, j \in \{1, \ldots |P|\}$ and non-reflexive, i.e. $(p_i, p_i) \notin E$ for all $i \in \{1, \ldots |P|\}$. Finally, we assume connectedness of the graph in the sense that for any two nodes $p_i, p_j \in P$ there exists a sequence $\tilde{\boldsymbol{p}} \in \bigcup_{i=0}^{|P|} P^i$ such that $(p_i, \tilde{p}_1)\,, \left(\tilde{p}_{|\tilde{\boldsymbol{p}}|}, p_j\right) \in E$ and for all $k \in \{1, \ldots, |\tilde{\boldsymbol{p}}| - 1\}$ we have $(\tilde{p}_k, \tilde{p}_{k+1}) \in E$, i.e. there exists a path between p_i and p_j via edges of the graph G_{fgr}. We denote by $\mathcal{G}_{\text{fgr}}$ the set of all G_{fgr}.

The nodes $p_i \in P$ represent the points of the Go board. The topology of the Go board is determined by the edge relation E. Being a neighbourhood relation it is symmetric and non-reflexive. The actual position is encoded by the colouring function c that determines which points on the board are *empty*, which ones are occupied by *white* and which ones are occupied by *black*. Note that up to this point the full graph representation is suitable for versions of the game of Go that are played on boards of arbitrary topology.

However, the rules of the game provide us with more structural information. In particular, we observe that *black* or *white* points that belong to the same chain (i.e. points of the same colour that see each other) will always have a *common fate*: either all of them remain on the board or all of them are cleared. In any case we can represent them in a single node. We also reduce the number of edges by requiring that any two nodes may be connected by only a single edge representing their neighbourhood relation. The resulting *reduced graph representation* will be called a *common fate graph* (CFG) and will serve as the basic representation for our experiments. More formally, let us define a graph contraction that transforms the FGR to the CFG.

Definition 71 (Common fate graph). We define the common fate graph G_{cfg} as the graph resulting from the application of the following procedure to the full graph representation G_{fgr}:

1. While there exist two nodes $p_i, p_j \in P$, that are neighbours, $(p_i, p_j) \in E$, of the same non-empty colour, $c(p_i) = c(p_j) \neq \mathit{empty}$ perform the following transformations:
2. $E \mapsto E \setminus \{(p_i, p_j), (p_j, p_i)\}$ to remove the edge between node p_i and p_j.
3. $E \mapsto E \cup \{(p_i, p_k), (p_k, p_i) \mid (p_j, p_k) \in E \wedge k \neq i\}$ to connect the remaining node p_i to those nodes p_k formerly connected to p_j.
4. $E \mapsto E \setminus \{(p_j, p_k), (p_k, p_j) \mid (p_j, p_k) \in E\}$ to remove any of the edges involving the node p_j to be removed.
5. $P \mapsto P \setminus \{p_j\}$ to remove the node p_j.

We denote by $\mathcal{G}_{\text{cfg}}$ the set of all G_{cfg}.

The result of such a transformation is shown in Figure 10.1 (b). Clearly, the complexity of the representation has been greatly reduced while retaining essential structural information in the representation.

In how far is the CFG a suitable representation for learning in the game of Go? Go players' intuition is often driven by a certain kind of aesthetics that refers to the local structure of positions and is called *good or bad shape*. As an example consider the two white groups in Figure 10.1 (a) and (b). Although they look quite distinct to the layman in Figure 10.1 (a) they share the property of *being alive,* because they both have *two eyes* (i.e. two isolated internal empty points called *liberties*). This essential property of stability is preserved in the CFG: a typical living group may be represented by a coloured node with two dangling empty points (see Figure 10.1 (b)), irrespective of the particular bulk structure of the group. The abstraction implemented by the CFG lies in the fact that only chains of stones are collapsed into joint nodes and that the structure of *liberties* is preserved. Of course, some information like, e.g. the number of points and their structure within a node is lost. This information, however could be retained by allowing for additional node labels.

10.3.2. Relative Subgraph Features

Unfortunately, almost all algorithms that aim at learning on graph representations directly suffer from severe scalability problems (see Geibel and Wysotzki (1998)). Many practically applicable learning algorithms operate on object representations known as feature vectors $\vec{x} \in \mathbb{R}^n$. We would thus like to extract feature vectors $\mathbf{x}$ from G_{cfg} for learning. Both learning tasks mentioned in the introduction can be formulated in terms of mappings from single nodes to real values: the position evaluation function can be (approximately) decomposed into a sum of node-wise evaluation functions (Schraudolph et al. 1994): For each node we estimate the probability that the node will be part of *black* or *white* territory at the end of the game. From these probabilities the expectation of the final score can be calculated. Also the move evaluation function is naturally formulated as a function from (empty) nodes to real numbers that indicate how desirable a move at that particular node is for a player. In both cases we would like to find a mapping $\boldsymbol{\phi} : \mathcal{G}_{\text{cfg}} \times P \to \mathbb{R}^n$ that maps a particular node $p \in P$ to a feature vector $\mathbf{x} \in \mathbb{R}^n$ given the context provided by the graph G_{cfg}, an idea inspired by Geibel and Wysotzki (1998) who apply context dependent classification to the mutagenicity of chemical compounds. In the following we define what we call relative subgraph features. These features are similar to the bag-of-words representation in text classification (Joachims 1998) because they are based on the number of a set of predetermined components appearing in a given object. Thus in the bag-of-words representation a feature is given by the number of times a certain word appears in a text while in the relative subgraph representation a graph is characterised by the number of times a certain subgraph appears.

Definition 72 (Relative subgraph features). Given a common fate graph $G_{\text{cfg}} = (P, E, c) \in \mathcal{G}_{\text{cfg}}$ and a root node $p \in P$ we define a relative subgraph feature $\phi_i : \mathcal{G}_{\text{cfg}} \times P \to \mathbb{N}$ corresponding to a connected subgraph $\tilde{G}_i \equiv \left(\tilde{P}_i, \tilde{E}_i, \tilde{c}_i\right)$ satisfying $p \in \tilde{P}_i$ in terms of the number n_i of times the subgraph $\tilde{G}_i$ can be found in G_{cfg},

$$n_i \stackrel{\text{def}}{=} \left| \left\{ a : \tilde{P}_i \to P \;\middle|\; \forall p_j \in \tilde{P}_i : c\left(p_j\right) = c\left(a\left(p_j\right)\right) \wedge \forall \left(p_j, p_k\right) \in \tilde{E}_i : \left(a\left(p_j\right), a\left(p_k\right)\right) \in E \right\} \right| .$$

The feature ϕ_i is then given by $\phi_i\left(p\right) \stackrel{\text{def}}{=} n_i$ and can be normalised such that $\|\mathbf{x}\|^2 = 1$.

Clearly, finding and counting subgraphs $\tilde{G}_i$ of G_{cfg} becomes quickly infeasible with increasing subgraph complexity as measured, for example by $|\tilde{P}_i|$ and $|\tilde{E}_i|$. We therefore restrict ourselves to connected subgraphs $\tilde{G}_i$ of the shape of chains without branches or loops. In practice, we limit the features to a local context $|\tilde{P}_i| \leq s$, which—given the other two constraints—also limits the number n of distinguishable features. It should be noted, however, that even with a relatively small number of s, e.g. $s = 6$, a considerable range on the board is achieved due to the compact CFG representation. Also, the relative subgraph features extracted from the CFG can be partly interpreted in terms of Go terminology. As an example, consider the feature in Figure 10.1 (c) row 3. This feature effectively counts the *liberties* of the point relative to which it is extracted (marked by gray ring), a number that constitutes an important feature for human Go players and is a measure of the tactical safety of a chain of stones.

10.4. Experiments

In our experiments we focused on learning the distinction between "good" and "bad" moves. However, we are really interested in the real-valued output of the classifier which enables us to order and select candidate moves according to their quality. Given the relative subgraph features extracted from the CFG any learning machine that takes feature vectors as inputs and provides a real-valued output is suitable for the task. In order to take into account the relatively high number of features we choose the hypothesis class $\mathcal{H}_{\mathcal{K}}$ of binary kernel classifiers for learning (see Definition 35). Recall that the predictions $\hat{y}$ of these classifiers are given by

$$\hat{y}(\mathbf{x}) \stackrel{\text{def}}{=} \operatorname{sign}(f(\mathbf{x})) = \operatorname{sign}\left(\sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x})\right),$$

and that the real-valued output of the function $f(\mathbf{x})$ may serve as a measure of confidence for the prediction. We decided to use an RBF kernel with diagonal penalty term of the form

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2 / \sigma^2\right) + \lambda \mathbf{I}_{\mathbf{x}_1 = \mathbf{x}_2},$$

where σ^2 controls the width of the kernel, and the second term makes the kernel matrix more well-conditioned and ensures linear separability of the training sample for high enough values of λ (see Remark 1). Two different learning algorithms for the expansion coefficients α_i were employed:

1. A simple kernel perceptron as described in Subsection 2.1.1. The kernel perceptron is a fast learning machine and produces sparse solutions that make the evaluation of the resulting classifier very efficient.

2. A soft margin support vector machine as described in Section 2.1.3. The support vector machine has emerged as a quasi-standard in learning kernel classifiers and gives robust, large-margin solutions.

10.4.1. Learning Solutions to Life-and-Death Problems

An interesting challenge in the game of Go—referred to as *tsume Go*—is to kill opponent groups and to save one's own groups from dying. Our *tsume* Go study was based on a database of 40 000 computer-generated Go problems with solution by Wolf (1994). In order to calibrate the parameters of both the representation and the learning schemes, we created a simple task: The training sample consisted of $m = 2\,700$ moves, where we took the best move provided by Wolf's problem solver GoTools (Wolf 1994) to be "good" and the worst one to be "bad". The length s of the subgraph chains was chosen to be $s = 6$ resulting in $n \approx 400$ features. Using a held-out data set of size $m_{\text{heldout}} \approx 3000$ we systematically scanned parameter space. For the SVM we set $\lambda = 0$ (relying on C for regularisation) and found the optimal parameter values on the held-out sample to be $\sigma = 1$ and $C = 10$. For the kernel perceptron we used the same value of σ and

test \ train	white	black	w&b	#probs	#moves	%good
white	65.8/65.3	48.0/48.8	63.8/62.6	1455	10.4	13.4
black	57.7/57.3	66.4/64.5	65.9/65.9	1711	9.8	23.0
w&b	61.5/61.0	58.0/57.3	64.9/64.4	3166	10.0	18.0
# pts	2718	2682	5400			

Table 10.1.: Results of the selection of *tsume* Go moves by an SVM (left) and a kernel perceptron (right). Shown is the percentage of successful determination of a problem-solving move. The numbers should be compared to the last column that indicates the average percentage of moves that solve the problem.

found $\lambda = 0.05$ to be optimal. Both solutions had a level of sparseness $\|\boldsymbol{\alpha}\|_0 / m$ of approximately 50% and lead to a success rate of up to 85% for discriminating between the best and the worst move in a given problem.

Given these parameters we turned to the more interesting task of picking the correct move among all the possible moves in a problem. Using essentially the same training paradigm as above we applied the resulting classifier to all the possible moves in a problem and used the real-valued output $f(\mathbf{x})$ for each legal move $\mathbf{x}$ for ranking. We counted a success when the move ranked highest by the classifier was indeed one of the winner moves. The results of these experiments are given in Table 10.1. Focusing on the full training and test sets (w&b) the success rate is about 65% with on average 18% of the moves considered solved the problem correctly. Although we were not able to obtain the data-base used in Sasaki and Sawada (1998) due to copyright problems, our result of a 65% success rate compares favourably with the 50% reported in Sasaki and Sawada (1998) on similar problems using a naive local context representation and a multi-layer perceptron with back-propagation.

In the problems used, *white* was the attacker and *black* the defender. The sections of the table refer to subsets of the whole sample containing only *black,* only *white*, or *black* and *white* moves. It turns out that in accordance with the Go proverb "Your opponent's move is your own move" the quality of certain moves does not depend on whether they are played for the purpose of attack or defence.

It should be noted that the performance of the kernel perceptron is absolutely comparable to that of the SVM—despite the much simpler training paradigm that requires much less computational resources than the SVM. We conjecture that in this special task the representation is so crucial that the difference between the learning algorithms becomes negligible. More generally, this might indicate that for complex problems in symbolic domains the impact of the choice of representation outweighs the choice of the learning algorithm.

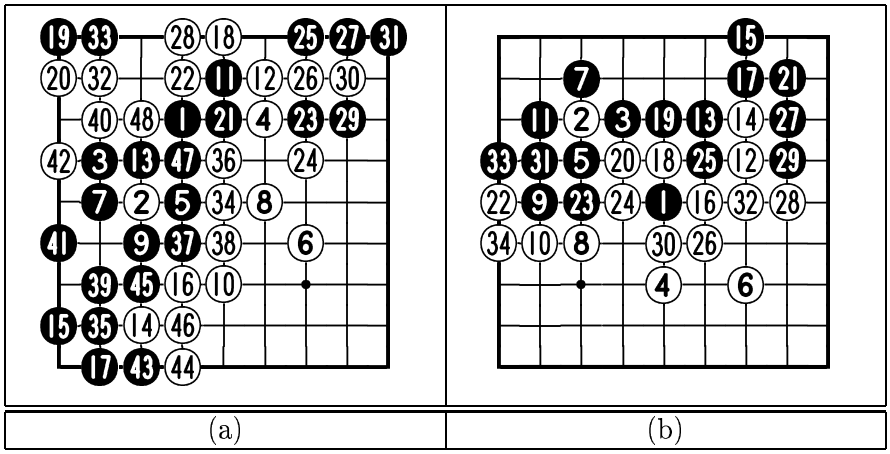


Figure 10.2.: Two examples of games played by a support vector machine (Black) against Gnu-Go (White). The SVM was trained on 9×9 game records from good amateur games. Despite the fact that the SVM eventually looses both games it finds a number of remarkable moves, creates stable groups, and surrounds territory.

10.4.2. Learning Game Play

The proof of the pudding is in the eating, and accordingly the test of the Go machine is in the playing. We applied the same learning paradigm as used above, but this time to a collection of 9×9 Go game records collected from the Internet Go Server (IGS) and pre-processed and archived by Nicol Schraudolph. We selected game records that were produced by players with a ranking of at least *amateur shodan*, assuming that their moves could be viewed as optimal from the point of view of the learning machine. For each of the $m \approx 2\,500$ moves played we randomly generated an arbitrary counterpart as a "bad" move. While this procedure does not guarantee to obtain a "bad" move it is quite likely that a random choice results in a "bad" move, in particular, because in the game of Go at any given turn many moves are possible until the late endgame.

We provide two samples of the SVMs play against Gnu-Go[3] in Figure 10.2. Connoisseurs of the game will appreciate that the machine plays amazingly coherent considering that it takes into account only local shape and has no concept of territory and urgency. Surprisingly, the program even creates situations of *atari* (in moves 9 and 45 of game (a) and moves 7, 25, and 29 of game (b)) and is able to capture attacked opponent stones in move 13 of game (a) and in move 11 of game (b). Both games are eventually lost, which is hardly surprising considering that not even the goal of the game was coded into the learning machine.

[3]Publicly available via `http://www.gnu.org/software/gnugo/`

10.5. Conclusions and Future Work

We presented Go as a successful application of learning based on feature vectors that are extracted from a graph representation. While the approach appears to be particularly suitable for the game of Go it seems that other domains could benefit from these ideas as well. In its present form, the system could serve as an intelligent move generator thus cutting down the width of the search tree in system based on an evaluation function and search of the game tree. In addition, the same representation that serves now to learn move evaluations could be used to learn position evaluation functions decomposed into a sum of the nodes of the CFG. Both the move and position evaluation combined with search would certainly lead to progress in playing strength. Eventually, it seems plausible that the idea of abstraction as represented by the CFG must be carried out over a hierarchy of representations mimicking the way human players think about the game of Go.

Conclusion and Outlook

The goal of this thesis was to document my contributions to the learning theory of kernel classifiers, the development of learning algorithms for pattern classification based on kernels, and the application of such algorithms to the game of Go. In the course of the thesis various pairs of opposites, or rather of complementary approaches have been mentioned. These pairs of concepts play an interesting role in the development of machine learning. They are often subject to discussion, sometimes they appear to be the seeds of certain (ideologic) schools of thought, and hopefully they will lead to progress both in theory and in practice.

Learning versus Programming

Machine learning is based on the idea that machines improve their performance by experience. Instead of feeding exact instructions into a machine, the machine learning approach suggests to demonstrate the correct behaviour to the machine and let it learn from examples. Of course, this approach still requires programming—but on a different level. The machine is no longer instructed on how to perform, but it is instructed on how to learn how to perform. Machine learning can thus be considered a course in meta-programming or rather in self-programming for machines. Possibly the most interesting applications of machine learning involve tasks that cannot be reasonably solved in any other way.

One such candidate application—if I may call a game an application—is the game of Go. Humans have acquired their excellent skills in playing the game of Go on two time scales: On a cultural scale where strategy and tactics evolved together with the thousands of years old tradition of the game. And on the personal scale of a lifetime—in the case of professionals—devoting a considerable fraction of their lifetime to the study and play of the game. How can a machine compete with humans under these circumstances? At the moment it cannot (see Chapter 10). Any attempt at programming a computer to play the game of Go well has failed in the sense that the average amateur player is not challenged at all by the current performance of computers. I conjecture that this will change only, if computers are taught to learn the game from the available resources in three modes: Playing against themselves, playing against others, and analysing the thousands of available game records from professional players.

While the discussion of the game of Go may appear to be rather esoteric to some it should be noted that the above statements about the game of Go apply to other—probably more important—issues of human culture, in particular, to language.

Induction versus Transduction

Machine learning is often described as an attempt to perform induction, that is, to infer general rules from particular observations. However, we need to distinguish between two different goals based on given observations: to find a good description of the general rule or to make good predictions. In machine learning given a training sample of observations the goal of induction is to select an appropriate hypothesis while the goal of transduction is to make a good prediction on the working sample. Of course, also induction is related to prediction: Often an hypothesis is considered good if its prediction error is small. From the point of view of learning theory the distinction between induction and transduction is given by the distinct quantities of interest: The prediction error of a distinct hypothesis versus the prediction error of a transduction scheme on a given working sample.

As has been argued in Section 7.1 the Bayesian framework for prediction is essentially a transduction scheme. Bayesian transduction in kernel space and Gaussian process algorithms (Opper and Winther 2000b) are instantiations of this principle and it will be a challenging task to characterise such schemes further in the PAC-Bayesian framework.

Example-based versus Hypothesis-based learning

The field of learning based on kernels has received a lot of attention during the past years—as can be seen from the large amount of publications and other activities in the field. I view the success of kernel-based learning as a consequence of the fact that kernel methods share aspects of both example-based and hypothesis-based learning (see Chapter 2). While the kernel acts as a similarity measure in feature space in the sense of the *principle of similarity* as formulated in the context of example-based learning in Chapter 2 most kernel algorithms aim at *empirical risk minimisation* in the spirit of hypothesis-based learning. The flexibility of a data-dependent hypothesis space together with the inductive principle of empirical risk minimisation appear to be a very successful combination (at least with regard to the number of papers published).

Sparseness versus Margin

The notions of sparseness and margin have played an important role both in learning theory and in the development of learning algorithms (see Chapters 5 and 4). Interestingly, the data-sparseness considered here is a property that is related to example-based learning while the notion of a margin is related to hypotheses that are expressed as thresholded real-valued functions. For kernel classifiers, sparseness and margin thus provide two complementary views on the learning process. This is reflected in the two different types of bounds on the prediction error: in terms of sparseness and in terms of margin. Analysing and possibly exploiting these complementary measures of effective complexity is an interesting avenue for further research, in particular considering that both sparseness (in the sense of compression) and margin (in the sense of approximation) play major roles in coding theory. First results have been obtained in the algorithmic luckiness framework (Herbrich and Williamson 2001).

PAC learning versus Bayesian learning

Two different schools of thought appear to be dominating the machine learning community: Those favouring the PAC learning approach and those working in the Bayesian framework. The two frameworks are difficult to compare because they essentially aim at answering different questions. While the theory of PAC learning attempts to give conditions for the feasibility of learning, the Bayesian framework often serves as a developer´s kit for learning algorithms. From a historic perspective it is interesting to see that the PAC community is rooted in frequentist statistics and that the clash between frequentist and Bayesian statistics appears to find a continuation in machine learning.

The PAC-Bayesian framework as described in Section 3.3 aims at providing PAC-type bounds on the prediction error of Bayesian predictors. People working in the PAC framework would probably argue that they provide a solid foundation for Bayesian algorithms, while Bayesians might reply that the PAC-Bayesian framework proves what they had known already, anyway. In any case, the PAC-Bayesian framework constitutes an interesting step in reconciling the different view points and it is certainly a very interesting—but politically dangerous—research project to try to provide frequentist justifications for Bayesian procedures of inference.

Apriori versus Aposteriori Bounds

Learning theory in the PAC/VC style as described in Section 3.1 traditionally provides apriori bounds on the prediction error that do not depend on the observed sample but can be evaluated before learning. The essential quantity that enters these bounds is then a complexity measure of the hypothesis space from which the hypothesis to be learned is chosen. Recent work including the luckiness framework (Shawe-Taylor et al. 1998) and the PAC-Bayesian framework (McAllester 1998; McAllester 1999) goes one step further (see also Chapters 5 and 4): After observing the training sample the alignment of training sample and hypothesis space is measured and enters the bounds on the prediction error. Both the observed margin and the sparseness of a classifier are such measures that indicate, how well the hypothesis space (or kernel) chosen matches the data. Future research may aim at identifying other such measures of complexity and at exploiting these measures for constructing learning algorithms with good generalisation abilities.

Hypothesis Space versus Learning Algorithm

PAC/VC theory typically provides bounds on the prediction error of hypotheses chosen from a given hypothesis space (see Section 3.1). It turns out, however, that the actual prediction error of a given learning algorithm is better described by taking into account the learning algorithm itself instead of just the hypothesis space on which it is based. Examples of this phenomenon are both, the support vector machine and the perceptron. Both algorithms are based on the hypothesis space of linear classifiers that is easily characterised in the VC framework by its VC dimension. However, good bounds on the prediction error of classifiers resulting from both of these algorithms take into account the particular structure of the algorithm: the large margin of the support vector machine

solution (see Chapter 4) and the sparse representation of the classifier resulting from the perceptron (see Chapter 5). Another step in this direction are bounds in terms of algorithmic stability (Bousquet and Elisseeff 2001).

Online versus Batch Learning

A common distinction between learning algorithms is that of online versus batch learning. Again, the perceptron and the support vector machine provide an excellent pair of examples. Online algorithms are often analysed in the mistake bound framework (see Subsection 6.2.1), that bounds the number of mistakes an online learning algorithm makes in the process of learning. In contrast, PAC/VC theory provides bounds on the prediction error of classifiers resulting effectively from a batch learning procedure. Interestingly, any online algorithm can be provided with a finite training sample and thus be considered a batch learning algorithm. As a consequence there exists an fascinating relation between mistake bounds and PAC bounds: mistake bounds can be converted into PAC bounds (see Chapter 6 for details). This relationship between online and batch algorithms has certainly not been fully explored, yet, and even more interesting results can be expected.

Sampling versus Optimisation

Bayesian machine learning algorithms are often based on sampling from the Bayesian posterior (see Chapter 8) while algorithms rooted in statistical learning theory such as the support vector machine are based on optimisation. While optimisation problems in neural networks are often multi-modal and difficult to solve the optimisation problems associated with more modern algorithms such as SVMs and Gaussian processes are convex and often more well-conditioned, contributing to the popularity of these algorithms. While both types of algorithm have their merits—as discussed in Chapter 8—it is interesting to observe a certain convergence of sampling and optimisation. As an example, consider the permutational perceptron sampler (see Section 8.4) that employs multiple runs of an optimisation algorithm—the perceptron—to obtain samples from version space. Vice versa, sampling schemes such as importance sampling (Neal 1993) may be viewed as being inspired from optimisation. Considering the complementary merits of sampling and optimisation this convergence of methods certainly constitutes an interesting area of future work.

Geometry versus Structure

Finally, I would like to mention what I think is the most dominating watershed in the field of machine learning. A lot of modern research in machine learning focuses on data that are given in terms of real-valued feature vectors. In agreement with multivariate statistics, a data sample is often viewed as a cloud of points in a high-dimensional space. Consequently, the classification problem is reduced to the problem of partitioning a vector space. Many important real-world problems, however, do not admit such a vector representation easily (see Goldfarb et al. (1995)). In fact, one can argue, that once a

vector representation has been chosen, the outcome of the learning endeavour—success or failure—is almost fully determined and the classification task is rendered impossible or trivial. Problems of this kind include the classification of organic molecule and of positions in the game of Go, the analysis of complex visual scenes, and probably many tasks related to language processing and understanding. For most of these problems it may be necessary to go back to the stage at which the representation of objects is designed. The development of kernels for objects with a relational structure such as strings and graphs (see, e.g. Haussler (1999)) can be considered a step towards the integration of structural and geometric approaches in machine learning: The kernel provides an implicit spatial embedding allowing for the application of geometric classification techniques.

A. Proofs and Derivations

This chapter provides proofs that are of interest in the context of this work but were relegated to the appendix for better readability of the main text.

A.1. The Perceptron Convergence Theorem

Theorem 48 (Margin perceptron convergence). *Given a training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, and a feature mapping $\boldsymbol{\phi}$ such that the data radius is $\varsigma \equiv \varsigma(\boldsymbol{x})$ and such that there exists a weight vector $\mathbf{w}^* \in \mathcal{K}$ with margin $\gamma^* \equiv \gamma(\mathbf{w}^*, \boldsymbol{z}) > 0$ the perceptron algorithm with enforced margin ρ^2 terminates after at most M update steps with*

$$M = \frac{2\rho^2 + \varsigma^2}{\gamma^{*2}},$$

and the margin $\gamma(\mathbf{w}_M, \boldsymbol{z})$ of the solution $\mathbf{w}_M$ on the training sample is at least

$$\gamma(\mathbf{w}_M, \boldsymbol{z}) \geq \frac{\rho^2 \gamma^*}{(2\rho^2 + \varsigma^2)}.$$

Proof. Using the Cauchy-Schwarz inequality $\langle \mathbf{w}_M, \mathbf{w}^* \rangle \leq \|\mathbf{w}_M\| \cdot \|\mathbf{w}^*\|$, Theorem 62, and repeated application of the update rule $\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{x}_{i(t-1)}$ for all $t \in \{1, \ldots, M\}$ we have that

$$\begin{aligned}
\|\mathbf{w}_M\|^2 \cdot \|\mathbf{w}^*\|^2 &\geq |\langle \mathbf{w}_M, \mathbf{w}^* \rangle|^2 \\
&= \left| \left\langle \sum_{t=1}^{M} y_{i(t)} \mathbf{x}_{i(t)}, \mathbf{w}^* \right\rangle \right|^2 \\
&= \left| \sum_{t=1}^{M} y_{i(t)} \left\langle \mathbf{x}_{i(t)}, \mathbf{w}^* \right\rangle \right|^2 \\
&\geq M^2 \gamma^{*2} \|\mathbf{w}^*\|^2,
\end{aligned}$$

where we used the existence of a large margin weight vector $\mathbf{w}^*$ with margin $\gamma^* \equiv \gamma(\mathbf{w}^*, \boldsymbol{z})$. This gives the first intermediate result

$$M^2 \gamma^{*2} \leq \|\mathbf{w}_M\|^2.$$

Again repeatedly using the update rule $\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{x}_{i(t-1)}$ and the update condition $y_{i(t-1)} \langle \mathbf{x}_{i(t-1)}, \mathbf{w}_{t-1} \rangle \leq \rho^2$ for all $t \in \{1, \ldots, M\}$ we get

$$\begin{aligned} \|\mathbf{w}_t\|^2 &= \left\|\mathbf{w}_{t-1} + y_{i(t-1)}\mathbf{x}_{i(t-1)}\right\|^2 \\ &= \|\mathbf{w}_{t-1}\|^2 + 2\left\langle y_{i(t-1)}\mathbf{x}_{i(t-1)}, \mathbf{w}_{t-1}\right\rangle + \left\|y_{i(t-1)}\mathbf{x}_{i(t-1)}\right\|^2 \\ &\leq t\left(2\rho^2 + \varsigma^2\right) \end{aligned}$$

which for $t = M$ gives the second intermediate result

$$\|\mathbf{w}_M\|^2 \leq M\left(2\rho^2 + \varsigma^2\right) .$$

Combining the two intermediate results we obtain

$$M^2\gamma^{*2} \leq \|\mathbf{w}_M\|^2 \leq M\left(2\rho^2 + \varsigma^2\right)$$

which gives the result of the theorem,

$$M \leq \frac{2\rho^2 + \varsigma^2}{\gamma^{*2}} . \tag{A.1}$$

Now we can also find a lower bound on the margin $\gamma(\mathbf{w}_M, \boldsymbol{z})$ of $\mathbf{w}_M$ on the training sample given by

$$\begin{aligned} \gamma(\mathbf{w}_M, \boldsymbol{z}) &= \min_{z_i \in \boldsymbol{z}} \frac{y_i \langle \mathbf{x}_i, \mathbf{w}_M \rangle}{\|\mathbf{w}_M\|} \\ &\geq \frac{\rho^2}{\sqrt{M\left(2\rho^2 + \varsigma^2\right)}} \\ &\geq \frac{\rho^2}{2\rho^2 + \varsigma^2}\gamma^* , \end{aligned}$$

where the last line follows from (A.1). □

A.2. The Dual Formulation of the Maximum Margin Hyperplane Quadratic Programme

Theorem 49 (Dual of maximum margin hyperplane quadratic programme). *Given a linearly separable training sample $\boldsymbol{z}$, a feature space $\mathcal{K}$, a feature mapping $\boldsymbol{\phi}$ and a solution $\boldsymbol{\alpha}^*$ of the quadratic programme*

$$\begin{aligned} \text{minimise}_{\boldsymbol{\alpha}} \quad & W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i\alpha_i = 0 \\ \forall i \in \{1, \ldots, m\} : \quad & \alpha_i \geq 0 \end{aligned}$$

then the weight vector $\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i$ yields the maximum margin hyperplane with margin $\gamma = \|\mathbf{w}^*\|^{-1}$.

Proof. Consider the primal Lagrangian

$$L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left[y_i \left(\langle \mathbf{x}_i, \mathbf{w}\rangle + w_0\right) - 1\right], \tag{A.2}$$

with Lagrangian multipliers $\alpha_i \geq 0$. Differentiation of L w.r.t. $\mathbf{w}$ and w_0 and imposing stationarity gives

$$\nabla_{\mathbf{w}} L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = \mathbf{0},$$

and

$$\frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial w_0} = \sum_{i=1}^{m} \alpha_i y_i = 0.$$

Re-substituting these results into the primal Lagrangian yields

$$\begin{aligned} L(\mathbf{w}, w_0, \boldsymbol{\alpha}) &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left[y_i \left(\langle \mathbf{x}_i, \mathbf{w}\rangle + w_0\right) - 1\right] \\ &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \end{aligned}$$

Thus we have derived the equivalent Wolf dual and have together with Theorem 3 the proof of the theorem. □

A.3. Dual Formulation of the 1-Norm Soft Margin Hyperplane

Theorem 50 (1-norm soft margin dual). *Given a training sample $\boldsymbol{z}$ (not necessarily linearly separable), a feature space $\mathcal{K}$, a feature mapping $\boldsymbol{\phi}$ and a solution $\boldsymbol{\alpha}^*$ of the quadratic programme*

$$\begin{aligned} \text{minimise}_{\boldsymbol{\alpha}} \quad & W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i \alpha_i = 0 \\ \forall i \in \{1, \ldots, m\} \quad & C \geq \alpha_i \geq 0, \end{aligned}$$

then the weight vector $\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i$ together with w_0^* chosen such that $\forall i \in \mathbf{i}_{\text{sv}}(\boldsymbol{\alpha}^*)$ we have $y_i\left(\langle \mathbf{x}_i, \mathbf{w}^*\rangle + w_0^*\right) = 1$, yields the 1-norm soft–margin hyperplane.

Proof. The Lagrangian for the 1-norm soft margin QP is given by

$$L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i \left[y_i \left(\langle \mathbf{x}_i, \mathbf{w}\rangle + w_0\right) - 1 + \xi_i\right] - \sum_{i=1}^{m} r_i \xi_i,$$

with $\alpha_i \geq 0$ as usual and Lagrange multipliers $r_i \geq 0$ for positive margin slacks ξ_i. Differentiation w.r.t. $\mathbf{w}$, w_0, and $\boldsymbol{\xi}$, and imposing stationarity gives

$$\begin{aligned}
\nabla_{\mathbf{w}} L\left(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}\right) &= \mathbf{w} - \sum_{i=1}^{m} y_i \alpha_i \mathbf{x}_i = \mathbf{0}\,, \\
\frac{\partial}{\partial w_0} L\left(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}\right) &= \sum_{i=1}^{m} y_i \alpha_i = 0\,, \\
\frac{\partial}{\partial \xi_i} L\left(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}\right) &= C - \alpha_i - r_i = 0\,.
\end{aligned}$$

Re-substituting these results into the Lagrangian $L\left(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}\right)$ to obtain the Wolf dual gives

$$W\left(\boldsymbol{\alpha}\right) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle\,,$$

which is the same function as for the maximum margin hyperplane dual. Combining the condition (for all $i \in \{1, \ldots, m\}$) $C - \alpha_i - r_i = 0$ with $r_i \geq 0$ gives $\alpha_i \leq C$, which, together with $\alpha_i \geq 0$, constitute the box constraints. Using the Karush-Kuhn-Tucker (KKT) complementarity condition (Kuhn and Tucker 1951) $r_i \xi_i = 0$ we have that $\xi_i \neq 0 \Rightarrow \alpha_i = C$. We can therefore write the KKT complementarity conditions for all $i \in \{1, \ldots, m\}$ as

$$\begin{aligned}
\alpha_i \left[y_i \left(\left\langle \mathbf{x}_i, \mathbf{w} \right\rangle + w_0\right) - 1 + \xi_i\right] &= 0 \\
\xi_i \left(\alpha_i - C\right) &= 0\,.
\end{aligned}$$

The value of w_0^* is found by observing that $C > \alpha_i^* > 0$ implies both $\xi_i^* = 0$ and $y_i \left(\left\langle \mathbf{x}_i, \mathbf{w}^* \right\rangle + w_0^*\right) - 1 + \xi_i^* = 0$ which together imply

$$y_i \left(\left\langle \mathbf{x}_i, \mathbf{w}^* \right\rangle + w_0^*\right) = 1$$

as given in the theorem. □

A.4. Dual Formulation of the 2-Norm Soft Margin Hyperplane

Theorem 51 (2-norm soft margin dual). *Given a training sample $\boldsymbol{z}$ (not necessarily linearly separable), a feature space $\mathcal{K}$, a feature mapping $\boldsymbol{\phi}$ and a solution $\boldsymbol{\alpha}^*$ of the quadratic programme*

$$\begin{aligned}
&\text{minimise}_{\boldsymbol{\alpha}} && W\left(\boldsymbol{\alpha}\right) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \left(\left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle + \frac{1}{C} \mathbf{I}_{i=j}\right) \\
&\text{subject to} && \sum_{i=1}^{m} y_i \alpha_i = 0 \\
&\qquad \forall i \in \{1, \ldots, m\} && \alpha_i \geq 0\,,
\end{aligned}$$

then the weight vector $\mathbf{w}^* = \sum_{i=1}^m \alpha_i^* y_i \mathbf{x}_i$ *together with* w_0^* *chosen such that* $\forall i \in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)$ *we have* $y_i(\langle \mathbf{x}_i, \mathbf{w}^* \rangle + w_0^*) = 1 - \frac{\alpha_i^*}{C}$*, yields the 2-norm soft-margin hyperplane with margin*

$$\gamma = \left(\|\boldsymbol{\alpha}^*\|_1 - \frac{1}{C} \|\boldsymbol{\alpha}^*\|_2^2 \right)^{-\frac{1}{2}} .$$

Proof. The Lagrangian for the 2-norm soft margin QP is given by

$$L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{2} \sum_{i=1}^m \xi_i^2 - \sum_{i=1}^m \alpha_i \left[y_i (\langle \mathbf{x}_i, \mathbf{w} \rangle + w_0) - 1 + \xi_i \right] ,$$

with $\alpha_i \geq 0$ as usual. Lagrange multipliers r_i to enforce the non-negativity of the ξ_i can be omitted because the objective function $L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha})$ is quadratic in the ξ_i. As a consequence, the value of $L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha})$ decreases if a negative ξ_i is increased towards zero, which automatically causes the solution to satisfy the non-negativity constraints for the slack variables ξ_i. Differentiation w.r.t. $\mathbf{w}$, w_0, and $\boldsymbol{\xi}$, and imposing stationarity gives

$$\begin{aligned}
\nabla_{\mathbf{w}} L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) &= \mathbf{w} - \sum_{i=1}^m y_i \alpha_i \mathbf{x}_i = \mathbf{0} , \\
\frac{\partial}{\partial w_0} L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) &= \sum_{i=1}^m y_i \alpha_i = 0 , \\
\frac{\partial}{\partial \xi_i} L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) &= C\boldsymbol{\xi} - \boldsymbol{\alpha} = \mathbf{0} .
\end{aligned}$$

Re-substituting these results into the Lagrangian L to obtain the Wolf dual W gives

$$\begin{aligned}
W(\boldsymbol{\alpha}) &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \frac{1}{2C} \langle \boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle - \frac{1}{C} \langle \boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \\
&= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \frac{1}{2C} \langle \boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \\
&= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \left(\langle \mathbf{x}_i, \mathbf{x}_j \rangle + \frac{1}{C} \mathbf{I}_{i=j} \right) .
\end{aligned}$$

The associated KKT complementarity conditions are given for all $i \in \{1, \ldots, m\}$ by

$$\alpha_i \left[y_i (\langle \mathbf{x}_i, \mathbf{w} \rangle + w_0) - 1 + \xi_i \right] = 0 .$$

Thus for determining w_0^* we use that $\xi_i^* = \frac{\alpha_i^*}{C}$ and thus for support vectors $i \in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)$

that $y_i\left(\langle \mathbf{x}_i, \mathbf{w}^*\rangle + w_0^*\right) = 1 - \frac{\alpha_i^*}{C}$. Finally, the margin $\gamma^* = \|\mathbf{w}^*\|^{-1}$ is calculated by

$$\begin{aligned}
\|\mathbf{w}^*\|^2 &= \sum_{i\in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \sum_{j\in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \alpha_i^* \alpha_j^* y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \\
&= \sum_{i\in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \alpha_i^* y_i \sum_{j\in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \alpha_j^* y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \\
&= \sum_{i\in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \alpha_i^* \left(1 - \xi_i^* - y_i w_0^*\right) \\
&= \sum_{i\in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \alpha_i^* - \sum_{i\in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \alpha_i^* \xi_i^* \\
&= \sum_{i\in \mathbf{i}_{\mathrm{sv}}(\boldsymbol{\alpha}^*)} \alpha_i^* - \frac{1}{C}\langle \boldsymbol{\alpha}^*, \boldsymbol{\alpha}^*\rangle
\end{aligned}$$

□

A.5. The Representer Theorem

Theorem 52 (Representer Theorem). *Suppose we are given a fixed mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{K} \subseteq \ell_2^n$, a training sample $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{Z}^m$, a cost function $c : \mathcal{X}^m \times \mathcal{Y}^m \times \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ strictly monotonically decreasing in its third argument and the class of linear functions in $\mathcal{K}$ as given by Definition 19. Then any $\mathbf{w}_{\boldsymbol{z}} \in \mathcal{W}$ defined by*

$$\mathbf{w}_{\boldsymbol{z}} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \; c\left(\boldsymbol{x}, \boldsymbol{y}, \left(\langle \mathbf{x}_1, \mathbf{w}\rangle, \ldots, \langle \mathbf{x}_m, \mathbf{w}\rangle\right)\right) \tag{A.3}$$

admits a representation of the form

$$\exists \boldsymbol{\alpha} \in \mathbb{R}^m : \qquad \mathbf{w}_{\boldsymbol{z}} = \sum_{i=1}^m \alpha_i \mathbf{x}_i \,. \tag{A.4}$$

Proof. Given $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{Z}^m$ we consider the projection $\mathbf{P}_{\boldsymbol{x}} : \mathcal{W} \to \mathcal{K}$ that maps all vectors of unit length into the linear span of the points $\{\boldsymbol{\phi}(x_1), \ldots, \boldsymbol{\phi}(x_m)\} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and the projection $\mathbf{P}_{\boldsymbol{x}}^{\perp} : \mathcal{W} \to \mathcal{K}$ that maps into the complement of the linear span of $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$. Thus, for any vector $\mathbf{w} \in \mathcal{K}$ we have $\mathbf{w} = \mathbf{P}_{\boldsymbol{x}}(\mathbf{w}) + \mathbf{P}_{\boldsymbol{x}}^{\perp}(\mathbf{w})$ which immediately implies that for all $(x_i, y_i) \in \boldsymbol{z}$

$$\langle \mathbf{x}_i, \mathbf{w}\rangle = \left\langle \mathbf{x}_i, \mathbf{P}_{\boldsymbol{x}}(\mathbf{w}) + \mathbf{P}_{\boldsymbol{x}}^{\perp}(\mathbf{w})\right\rangle = \langle \mathbf{x}_i, \mathbf{P}_{\boldsymbol{x}}(\mathbf{w})\rangle + \left\langle \mathbf{x}_i, \mathbf{P}_{\boldsymbol{x}}^{\perp}(\mathbf{w})\right\rangle = \langle \mathbf{x}_i, \mathbf{P}_{\boldsymbol{x}}(\mathbf{w})\rangle \,.$$

Suppose $\mathbf{w}_{\boldsymbol{z}}$ is a minimiser of (A.3) but $\mathbf{P}_{\boldsymbol{z}}(\mathbf{w}_{\boldsymbol{z}}) \neq \mathbf{w}_{\boldsymbol{z}}$, i.e. $\|\mathbf{P}_{\boldsymbol{x}}(\mathbf{w}_{\boldsymbol{z}})\| < \|\mathbf{w}_{\boldsymbol{z}}\| = 1$. Then

$$\begin{aligned}
& c\left(\boldsymbol{x}, \boldsymbol{y}, \left(\langle \mathbf{x}_1, \mathbf{w}_{\boldsymbol{z}}\rangle, \ldots, \langle \mathbf{x}_m, \mathbf{w}_{\boldsymbol{z}}\rangle\right)\right) \\
= \; & c\left(\boldsymbol{x}, \boldsymbol{y}, \left(\langle \mathbf{x}_1, \mathbf{P}_{\boldsymbol{x}}(\mathbf{w}_{\boldsymbol{z}})\rangle, \ldots, \langle \mathbf{x}_m, \mathbf{P}_{\boldsymbol{x}}(\mathbf{w}_{\boldsymbol{z}})\rangle\right)\right) \\
\geq \; & c\left(\boldsymbol{x}, \boldsymbol{y}, \|\mathbf{P}_{\boldsymbol{x}}(\mathbf{w}_{\boldsymbol{z}})\|^{-1}\left(\langle \mathbf{x}_1, \mathbf{P}_{\boldsymbol{x}}(\mathbf{w}_{\boldsymbol{z}})\rangle, \ldots, \langle \mathbf{x}_m, \mathbf{P}_{\boldsymbol{x}}(\mathbf{w}_{\boldsymbol{z}})\rangle\right)\right) \\
= \; & c\left(\boldsymbol{x}, \boldsymbol{y}, \left(\left\langle \mathbf{x}_1, \frac{\mathbf{P}_{\boldsymbol{x}}(\mathbf{w}_{\boldsymbol{z}})}{\|\mathbf{P}_{\boldsymbol{x}}(\mathbf{w}_{\boldsymbol{z}})\|}\right\rangle, \ldots, \left\langle \mathbf{x}_m, \frac{\mathbf{P}_{\boldsymbol{x}}(\mathbf{w}_{\boldsymbol{z}})}{\|\mathbf{P}_{\boldsymbol{x}}(\mathbf{w}_{\boldsymbol{z}})\|}\right\rangle\right)\right) \,,
\end{aligned}$$

where the second line follows from the assumption that c is strictly monotonically decreasing in the third argument. We see that $\mathbf{w}_{\boldsymbol{z}}$ cannot be the minimiser of (A.3) and by contradiction it follows that the minimiser must admit the representation (A.4). □

A.6. The Reversal of Quantifiers

A.6.1. The Uniform Quantifier Reversal Lemma

In order to prove the quantifier reversal lemma we need another lemma:

Lemma 13 (Antimonotonicity lemma). *Let f be any measurable function $f : \mathcal{Y} \to (0,1)$ and let g be any measurable anti-monotone function $g : (0,1) \to \mathbb{R}$, i.e., $\forall x, y \in (0,1)\ x \geq y \Rightarrow g(x) \leq g(y)$. Then if*

$$\mathbf{P}_{\mathsf{Y}}\left(f\left(\mathsf{Y}\right) \geq \delta\right) \geq 1 - \delta$$

then

$$\mathbf{E}_{\mathsf{Y}}\left[g\left(f\left(\mathsf{Y}\right)\right)\right] \leq \int_0^1 g\left(z\right)\, dz\,.$$

Proof. Without loss of generality we assume that singleton sets $U \in \mathcal{Y}, |U| = 1$ have zero measure $\mathbf{P}_{\mathsf{Y}}(U) = 0$. This may always be achieved by introducing an auxiliary random variable $\tilde{\mathsf{Y}} \sim \text{Uniform}\,(0,1)$ with an associated probability space $\left(\tilde{\mathcal{Y}}, \tilde{\mathcal{Y}}, \mathbf{P}_{\tilde{\mathsf{Y}}}\right)$ and considering the product space $\left(\mathcal{Y} \times \tilde{\mathcal{Y}}, \mathcal{Y} \times \tilde{\mathcal{Y}}, \mathbf{P}_{\mathsf{Y}\tilde{\mathsf{Y}}}\right)$ instead of $(\mathcal{Y}, \mathcal{Y}, \mathbf{P}_{\mathsf{Y}})$. We define a new random variable $\mathsf{Y}_f = f\left(\mathsf{Y}\right)$ and rewrite the given condition $\mathbf{P}_{\mathsf{Y}}\left(f\left(\mathsf{Y}\right) \geq \delta\right) \geq 1 - \delta$ in terms of the cumulative distribution function $\mathbf{F}_{\mathsf{Y}_f}$ of Y_f.

$$\begin{array}{rcl} \mathbf{P}_{\mathsf{Y}}\left(f\left(\mathsf{Y}\right) \geq \delta\right) & \geq & 1 - \delta \\ & \Leftrightarrow & \\ \mathbf{F}_{\mathsf{Y}_f}^{-1}\left(\delta\right) & \geq & \delta \end{array}$$

Now we have for the expectation

$$\begin{aligned} \mathbf{E}_{\mathsf{Y}}\left[g\left(f\left(\mathsf{Y}\right)\right)\right] &= \mathbf{E}_{\mathsf{Y}_f}\left[g\left(\mathsf{Y}_f\right)\right] \\ &= \lim_{n\to\infty} \sum_{i=1}^{n-1} g\left(\frac{i}{n}\right)\left(\mathbf{F}_{\mathsf{Y}_f}\left(\frac{i}{n}\right) - \mathbf{F}_{\mathsf{Y}_f}\left(\frac{i-1}{n}\right)\right) \\ &= \lim_{n\to\infty} \sum_{i=1}^{n-1} g\left(\mathbf{F}_{\mathsf{Y}_f}^{-1}\left(\frac{i}{n}\right)\right)\left(\mathbf{F}_{\mathsf{Y}_f}\left(\mathbf{F}_{\mathsf{Y}_f}^{-1}\left(\frac{i}{n}\right)\right) - \mathbf{F}_{\mathsf{Y}_f}\left(\mathbf{F}_{\mathsf{Y}_f}^{-1}\left(\frac{i-1}{n}\right)\right)\right) \\ &\leq \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n-1} g\left(\frac{i}{n}\right) \\ &= \int_0^1 g\left(z\right)\, dz \end{aligned}$$

where we used $\mathbf{F}_{\mathsf{Y}_f}^{-1}\left(\delta\right) \geq \delta$ for all $\delta \in (0,1]$. □

Based on the previous lemma we prove the quantifier reversal lemma, which is repeated here for convenience.

Lemma 14 (Quantifier reversal lemma). *Let* X *and* Y *be random variables with associated probability spaces* $(\mathcal{X}, \mathfrak{X}, \mathbf{P}_{\mathsf{X}})$ *and* $(\mathcal{Y}, \mathfrak{Y}, \mathbf{P}_{\mathsf{Y}})$*, respectively, and let* $\delta \in (0,1]$*. Let* $\Upsilon : \mathcal{X} \times \mathcal{Y} \times (0,1] \to \{\text{true}, \text{false}\}$ *be any measurable formula such that for any* x *and* y *we have* $\{\delta \in (0,1] \,|\, \Upsilon(x,y,\delta)\} = (0, \delta_{\max}]$ *for some* $\delta_{\max} \in \mathbb{R}$*. If for all* $x \in \mathcal{X}$ *and for all* $\delta \in (0,1]$

$$\mathbf{P}_{\mathsf{Y}}(\Upsilon(x, \mathsf{Y}, \delta)) \geq 1 - \delta$$

then for any $\delta \in (0,1]$ *and* $\beta \in (0,1)$ *we have*

$$\mathbf{P}_{\mathsf{Y}}\left(\forall \gamma \in (0,1] : \mathbf{P}_{\mathsf{X}}\left(\Upsilon\left(\mathsf{X}, \mathsf{Y}, (\gamma\beta\delta)^{\frac{1}{1-\beta}}\right)\right) \geq 1 - \gamma\right) \geq 1 - \delta\,.$$

Proof. First we define an auxiliary function $f : \mathcal{X} \times \mathcal{Y} \to \{0, \delta_{\max}\}$ as follows

$$f(x,y) = \begin{cases} 0 & \text{if} \quad \{\delta \in (0,1] : \Upsilon(\mathsf{X}, \mathsf{Y}, \delta)\} = \emptyset \\ \delta_{\max} & \text{otherwise} \end{cases}.$$

Then for $\delta \in (0,1]$ we can write

$$\Upsilon(x,y,\delta) \;\Leftrightarrow\; f(x,y) \geq \delta\,.$$

Now assume that

$$\forall x \in \mathcal{X} : \forall \delta > 0 : \mathbf{P}_{\mathsf{Y}}(f(x, \mathsf{Y}) \geq \delta) \geq 1 - \delta\,.$$

For $\beta \in (0,1)$ we have from the previous lemma with $g : (0,1) \to \mathbb{R}$ given by $g(\delta) = \delta^{\beta-1}$, anti-monotonic, and $f : \mathcal{Y} \to \mathbb{R}$ given by $f(\cdot) \equiv f(x, \cdot)$ that

$$\mathbf{E}_{\mathsf{Y}}\left[f^{\beta-1}(x, \mathsf{Y})\right] \leq \int_0^1 z^{\beta-1}\,dz = \frac{1}{\beta} z^{\beta}\Big|_{z=1} = \frac{1}{\beta}\,.$$

Now we can take the expectation over X to obtain

$$\mathbf{E}_{\mathsf{Y}}\mathbf{E}_{\mathsf{X}}\left[f^{\beta-1}(\mathsf{X}, \mathsf{Y})\right] \leq \frac{1}{\beta}\,.$$

Applying Markov´s inequality, Theorem 59, w.r.t. Y we have

$$\mathbf{P}_{\mathsf{Y}}\left(\mathbf{E}_{\mathsf{X}}\left[f^{\beta-1}(\mathsf{X}, \mathsf{Y})\right] \leq \frac{1}{\beta\delta}\right) \geq 1 - \delta\,.$$

Applying Markov´s inequality, Theorem 59, w.r.t. X we have

$$\begin{aligned}
\mathbf{P}_{\mathsf{Y}}\left(\forall \gamma > 0 : \mathbf{P}_{\mathsf{X}}\left(f^{\beta-1}(\mathsf{X}, \mathsf{Y}) \leq \frac{1}{\gamma\beta\delta}\right) \geq 1 - \gamma\right) &\geq 1 - \delta \\
\mathbf{P}_{\mathsf{Y}}\left(\forall \gamma > 0 : \mathbf{P}_{\mathsf{X}}\left(f(\mathsf{X}, \mathsf{Y}) \geq (\gamma\beta\delta)^{\frac{1}{1-\beta}}\right) \geq 1 - \gamma\right) &\geq 1 - \delta \\
\mathbf{P}_{\mathsf{Y}}\left(\forall \gamma > 0 : \mathbf{P}_{\mathsf{X}}\left(\Upsilon\left(\mathsf{X}, \mathsf{Y}, (\gamma\beta\delta)^{\frac{1}{1-\beta}}\right)\right) \geq 1 - \gamma\right) &\geq 1 - \delta\,,
\end{aligned}$$

which concludes the proof. □

A.6.2. The Simple Quantifier Reversal Lemma

Lemma 15 (Simple quantifier reversal). *Let* X *and* Y *be random variables with associated probability spaces* $(\mathcal{X}, \mathfrak{X}, \mathbf{P}_{\mathsf{X}})$ *and* $(\mathcal{Y}, \mathfrak{Y}, \mathbf{P}_{\mathsf{Y}})$*, respectively. Let* $\Upsilon : \mathcal{X} \times \mathcal{Y} \times (0,1] \to \{\text{true}, \text{false}\}$ *be any measurable formula. If for all* $x \in \mathcal{X}$ *and* $\delta \in (0,1]$*,*

$$\mathbf{P}_{\mathsf{Y}}\left(\Upsilon\left(x, \mathsf{Y}, \delta\right)\right) \geq 1 - \delta, \tag{A.5}$$

then for any $\delta > 0$ *and* $\gamma \in (0,1)$ *we have*

$$\mathbf{P}_{\mathsf{Y}}\left(\mathbf{P}_{\mathsf{X}}\left(\Upsilon\left(\mathsf{X}, \mathsf{Y}, \gamma\delta\right)\right) \geq 1 - \gamma\right) \geq 1 - \delta\,.$$

Proof. The condition (A.5) is equivalent to the proposition that for all $x \in \mathcal{X}$ and $\beta \in (0,1]$,

$$\mathbf{E}_{\mathsf{Y}}\left[\mathbf{I}_{\Upsilon(x,\mathsf{Y},\beta)}\right] \geq 1 - \beta\,.$$

Since the condition is true for all $x \in \mathcal{X}$ we also have

$$\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Y}}\left[1 - \mathbf{I}_{\Upsilon(\mathsf{X},\mathsf{Y},\beta)}\right]\right] < \beta\,,$$

which is equivalent to

$$\mathbf{E}_{\mathsf{Y}}\left[\mathbf{E}_{\mathsf{X}}\left[1 - \mathbf{I}_{\Upsilon(\mathsf{X},\mathsf{Y},\beta)}\right]\right] < \beta\,.$$

Applying Markov´s inequality, Theorem 59, we have for all $\gamma > 0$,

$$\mathbf{P}_{\mathsf{Y}}\left(\mathbf{E}_{\mathsf{X}}\left[1 - \mathbf{I}_{\Upsilon(\mathsf{X},\mathsf{Y},\beta)}\right] \geq \gamma\right) < \frac{\beta}{\gamma}\,,$$

which can also be written as

$$\mathbf{P}_{\mathsf{Y}}\left(\mathbf{E}_{\mathsf{X}}\left[\mathbf{I}_{\Upsilon(\mathsf{X},\mathsf{Y},\beta)}\right] \geq 1 - \gamma\right) \geq 1 - \frac{\beta}{\gamma}\,.$$

Reverting back to probabilities we get

$$\mathbf{P}_{\mathsf{Y}}\left(\mathbf{P}_{\mathsf{X}}\left(\Upsilon\left(\mathsf{X}, \mathsf{Y}, \beta\right)\right) \geq 1 - \gamma\right) \geq 1 - \frac{\beta}{\gamma}\,,$$

which—choosing $\beta = \gamma\delta$—gives the result,

$$\mathbf{P}_{\mathsf{Y}}\left(\mathbf{P}_{\mathsf{X}}\left(\Upsilon\left(\mathsf{X}, \mathsf{Y}, \gamma\delta\right)\right) \geq 1 - \gamma\right) \geq 1 - \delta\,.$$

□

A.7. PAC-Bayesian Margin Bound

The appendix contains the lemmata and theorems necessary to prove the PAC-Bayesian margin bound, Theorem 29. Due to the length of the single proofs we have split them into separate sections.

A.7.1. Balls in Version Space

In this section we prove that the ball

$$\mathcal{B}(\mathbf{w}) = \left\{ \mathbf{v} \in \mathcal{W} \;\middle|\; \langle \mathbf{w}, \mathbf{v} \rangle > \sqrt{1 - \Gamma^2(\mathbf{w}, \boldsymbol{z})} \right\}$$

around a linear classifier having normal $\mathbf{w}$ of unit length only contains classifiers within version space $V(\boldsymbol{z})$. Here, $\Gamma(\mathbf{w}, \boldsymbol{z})$ is the margin of the hyperplane $\mathbf{w}$ on a set of points *normalised by the length* $\|\mathbf{x}_i\|$ *of the* $\mathbf{x}_i$ (see Definition 55). In order to prove this result we need the following lemma.

Lemma 16. *Suppose* $\mathcal{K} \subseteq \ell_2^n$ *is a fixed feature space. Assume we are given two points* $\mathbf{w} \in \mathcal{W}$ *and* $\mathbf{x} \in \mathcal{K}$ *such that* $\langle \mathbf{w}, \mathbf{x} \rangle = \gamma > 0$*. Then for all* $\mathbf{v} \in \mathcal{W}$ *with*

$$\langle \mathbf{w}, \mathbf{v} \rangle > \sqrt{1 - \frac{\gamma^2}{\|\mathbf{x}\|^2}} \tag{A.6}$$

it follows that $\langle \mathbf{v}, \mathbf{x} \rangle > 0$.

Proof. Since we only evaluate the inner product of any admissible $\mathbf{v}$ with $\mathbf{w}$ and $\mathbf{x}$, we can make the following ansatz,

$$\mathbf{v} = \lambda \frac{\mathbf{x}}{\|\mathbf{x}\|} + \tau \left(\mathbf{w} - \gamma \frac{\mathbf{x}}{\|\mathbf{x}\|^2} \right).$$

Note that the vectors $\frac{\mathbf{x}}{\|\mathbf{x}\|}$ and $\mathbf{w} - \gamma \frac{\mathbf{x}}{\|\mathbf{x}\|^2}$ are orthogonal by construction. Furthermore, the squared length of $\mathbf{w} - \gamma \frac{\mathbf{x}}{\|\mathbf{x}\|^2}$ is given by $1 - \gamma^2 / \|\mathbf{x}\|^2$. Therefore, the unit norm constraint on $\mathbf{v}$ implies that

$$\tau^2 = \frac{1 - \lambda^2}{1 - \frac{\gamma^2}{\|\mathbf{x}\|^2}}.$$

Furthermore, assumption (A.6) becomes

$$\begin{aligned}
\left\langle \lambda \frac{\mathbf{x}}{\|\mathbf{x}\|} + \tau \left(\mathbf{w} - \gamma \frac{\mathbf{x}}{\|\mathbf{x}\|^2} \right), \mathbf{w} \right\rangle &> \sqrt{1 - \frac{\gamma^2}{\|\mathbf{x}\|^2}} \\
\lambda \frac{\gamma}{\|\mathbf{x}\|} \pm \sqrt{\frac{1 - \lambda^2}{1 - \frac{\gamma^2}{\|\mathbf{x}\|^2}}} \left(1 - \frac{\gamma^2}{\|\mathbf{x}\|^2} \right) &> \sqrt{1 - \frac{\gamma^2}{\|\mathbf{x}\|^2}} \\
\underbrace{\lambda \frac{\gamma}{\|\mathbf{x}\|} - \sqrt{1 - \frac{\gamma^2}{\|\mathbf{x}\|^2}} \left(1 \pm \sqrt{1 - \lambda^2} \right)}_{f(\lambda)} &> 0.
\end{aligned}$$

In order to solve for λ we consider the l.h.s. as a function of λ and determine the range of values where $f(\lambda)$ is positive. A straightforward calculation reveals that $[0, \lambda_{\max}]$ with

$$\lambda_{\max} = \frac{2\gamma}{\|\mathbf{x}\|} \sqrt{1 - \frac{\gamma^2}{\|\mathbf{x}\|^2}},$$

is the only range where $f(\lambda)$ is positive. As a consequence, the assumption $\langle \mathbf{w}, \mathbf{v} \rangle > \sqrt{1 - \gamma^2 / \|\mathbf{x}\|^2}$ is equivalent to

$$0 < \lambda \|\mathbf{x}\| < 2\gamma \sqrt{1 - \frac{\gamma^2}{\|\mathbf{x}\|}} .$$

Finally, the inner product of any $\mathbf{v}$ with $\mathbf{x}$ is given by

$$\begin{aligned} \langle \mathbf{v}, \mathbf{x} \rangle &= \left\langle \lambda \frac{\mathbf{x}}{\|\mathbf{x}\|} + \tau \left(\mathbf{w} - \gamma \frac{\mathbf{x}}{\|\mathbf{x}\|} \right), \mathbf{x} \right\rangle \\ &= \lambda \|\mathbf{x}\| + \tau (\gamma - \gamma) > 0 , \end{aligned}$$

where the last inequality follows from the previous consideration. The lemma is proven. □

Theorem 53. *Suppose $\mathcal{K} \subseteq \ell_2^n$ is a fixed feature space. Given a training sample $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}) \in (\mathcal{X} \times \{-1, +1\})^m$ and $\mathbf{w} \in \mathcal{W}$ such that $\Gamma(\mathbf{w}, \boldsymbol{z}) > 0$, for all $\mathbf{v} \in \mathcal{W}$ such that $\langle \mathbf{w}, \mathbf{v} \rangle > \sqrt{1 - \Gamma^2(\mathbf{w}, \boldsymbol{z})}$ we have for all $i = \{1, \ldots, m\}$,*

$$y_i \langle \mathbf{v}, \mathbf{x}_i \rangle > 0 .$$

Proof. According to Lemma 16 we know that all $\mathbf{v} \in B_i$ with

$$B_i = \left\{ \mathbf{v} \in \mathcal{W} \;\middle|\; \langle \mathbf{w}, \mathbf{v} \rangle > \sqrt{1 - \frac{\left(y_i \langle \mathbf{x}_i, \mathbf{w} \rangle \right)^2}{\|\mathbf{x}_i\|^2}} \right\} ,$$

parameterise classifiers consistent with the ith training example (x_i, y_i). Clearly, the intersection of all B_i gives the classifiers $\mathbf{w}$ which jointly satisfy the constraints $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0$. Noticing that the size of B_i depends inversely on $y_i \langle \mathbf{x}_i, \mathbf{w} \rangle$ we see that all $\mathbf{v}$ with $\langle \mathbf{w}, \mathbf{v} \rangle > \Gamma(\mathbf{w}, \boldsymbol{z})$ jointly classify all points $\mathbf{x}_i$ correctly. □

A.7.2. Volume Ratio Theorem

In this section we explicitly derive the volume ratio between the largest ball inscribable in version space and the whole parameter space for the special case of linear classifiers in $\mathbb{R}^n$. Given a point $\mathbf{w} \in \mathcal{W}$ and a positive number $\gamma > 0$ we can characterise the ball of radius γ in the parameter space by

$$\begin{aligned} \mathcal{B}_\gamma(\mathbf{w}) &= \left\{ \mathbf{v} \in \mathcal{W} \;\middle|\; \|\mathbf{w} - \mathbf{v}\|^2 < \gamma^2 \right\} \\ &= \left\{ \mathbf{v} \in \mathcal{W} \;\middle|\; \langle \mathbf{w}, \mathbf{v} \rangle > 1 - \gamma^2/2 \right\} . \end{aligned}$$

The following theorem gives the exact value of the *volume ratio* $\frac{\mathrm{vol}(\mathcal{W})}{\mathrm{vol}(\mathcal{B}_\gamma(\mathbf{w}))}$ where $\mathbf{w} \in \mathcal{W}$ can be chosen arbitrarily (due to the symmetry of the sphere).

Theorem 54. *Suppose we are given a fixed feature space $\mathcal{K} \subseteq \ell_2^n$. Then the fraction of the surface* $\mathrm{vol}\,(\mathcal{W})$ *of the unit sphere to the surface* $\mathrm{vol}\,(\mathcal{B}_\gamma\,(\mathbf{w}))$ *with Euclidean distance less than γ from any point $\mathbf{v} \in \mathcal{W}$ is given by*

$$\frac{\mathrm{vol}\,(\mathcal{W})}{\mathrm{vol}\,(\mathcal{B}_\gamma\,(\mathbf{w}))} = \frac{\int_0^\pi \sin^{n-2}(\theta)\; d\theta}{\int_0^{\arccos\left(1-\frac{\gamma^2}{2}\right)} \sin^{n-2}(\theta)\; d\theta}\,. \tag{A.7}$$

Proof. Consider spherical coordinates such that every $\mathbf{w} \in \mathcal{W}$ is expressed via $n-2$ angles $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{n-2})$ ranging from 0 to π, and one angle $0 \leq \varphi \leq 2\pi$. Choose $\mathbf{w} = (1, 0, \ldots, 0)'$ and the coordinate θ such that $w_1 = \cos(\theta)$. Then the intersection $\mathcal{W}_{\theta_0}^{n-1} \equiv \mathcal{W} \cap \{\mathbf{w}\,|\theta = \theta_0\}$ is an $(n-1)$-dimensional unit hyper-sphere of radius $r_{n-1} = \sin(\theta)$. The surface area $\mathrm{vol}\left(\mathcal{W}_{\theta_0}^{n-1}\right)$ of $\mathcal{W}_{\theta_0}^{n-1}$ is given by

$$\mathrm{vol}\left(\mathcal{W}_{\theta_0}^{n-1}\right) = c \cdot \sin^{n-2}(\theta)\;,$$

which gives for the ratio

$$\frac{\mathrm{vol}\,(\mathcal{W})}{\mathrm{vol}\,(\mathcal{B}_\gamma\,(\mathbf{w}))} = \frac{\int_0^\pi \mathrm{vol}\left(\mathcal{W}_{\theta_0}^{n-1}\right)\; d\theta_0}{\int_0^{\arccos\left(1-\frac{\gamma^2}{2}\right)} \mathrm{vol}\left(\mathcal{W}_{\theta_0}^{n-1}\right)\; d\theta_0}\,,$$

from which the result follows. □

A.7.3. A Bound on the Volume Ratio

In this section we present a practically useful upper bound for the expression (A.7). In order to check the usefulness of this expression we have compared the exact value of (A.7) with the upper bound and found that in the interesting regime of large margins the bound seems to be within a factor of 2 from the exact value (see Figure A.1).

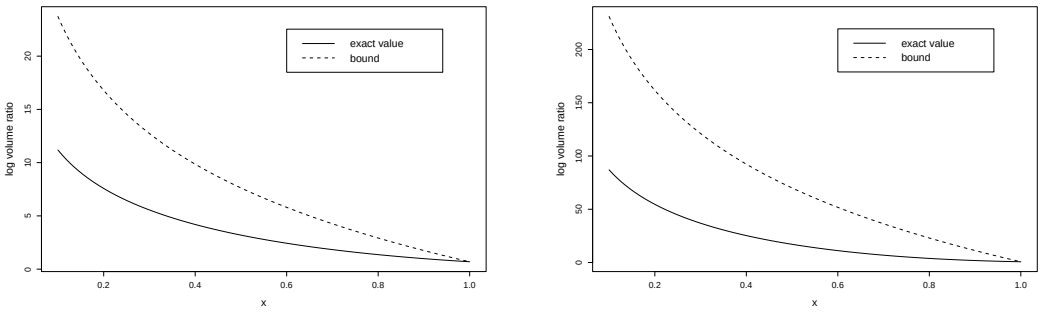
Theorem 55. *For all $j \in \mathbb{N}$ and all $0 < x \leq \frac{1}{2}$*

$$\log \frac{\int_0^\pi \sin^{2j+1}(\theta)\; d\theta}{\int_0^{\Psi(x)} \sin^{2j+1}(\theta)\; d\theta} \leq \log\left(\frac{1}{2x}\right)^{2j+1} + \log 2\,, \tag{A.8}$$

where $\Psi(x) = \arccos(1-2x)$.

Proof. From Bois (1961) we know that for all $j \in \mathbb{N}$

$$\int \sin^{2j+1}(\theta)\, d\theta = -\frac{\cos(\theta)}{2j+1} \sum_{i=0}^{j} \sin^{2i}(\theta)\, B_{j,i}\,, \tag{A.9}$$



(a) (b)

Figure A.1.: Comparison of the bound (A.8) (dashed line) with the exact value (A.7) (solid line) over the whole range of possible values of x for **(a)** $n = 10$ and **(b)** $n = 100$. Interestingly, in the relevant regime of large values of x the bound seems to be very tight regardless of the number n of dimensions.

where

$$B_{j,i} \stackrel{\text{def}}{=} \frac{2\,(i+1)\cdot 2\,(i+2)\cdot\cdots\cdot 2j}{(2i+1)\cdot(2i+3)\cdot\cdots\cdot(2j-1)} \tag{A.10}$$

$$= \frac{2\cdot 4\cdots 2j}{1\cdot 3\cdots(2j-1)}\cdot\frac{1\cdot 3\cdots(2i-1)}{2\cdot 4\cdots(2i)} \tag{A.11}$$

$$= \frac{4^j\,(j!)^2\,(2i)!}{(2j)!\,(i!)^2\,4^i} = \frac{4^j}{4^i}\frac{\binom{2i}{i}}{\binom{2j}{j}}\,. \tag{A.12}$$

Let us introduce the abbreviation

$$S\,(j,x) \stackrel{\text{def}}{=} \int_0^{\arccos(1-2x)} \sin^{2j+1}(\theta)\; d\theta\,.$$

Then the numerator of (A.8) is given by $S\,(j,1)$ whereas the denominator of (A.8) is simply $S\,(j,x)$. From (A.9) we see

$$S\,(j,x) = -\frac{\cos(\theta)}{2j+1}\sum_{i=0}^{j}\sin^{2i}(\theta)\,B_{j,i}\Big|_0^{\arccos(1-2x)}$$

$$= \frac{4^j}{(2j+1)\binom{2j}{j}}\left(1+(2x-1)\sum_{i=0}^{j}\binom{2i}{i}x^i\,(1-x)^i\right)\,.$$

where we have used (A.12) and

$$\sin^{2i}(\theta) = \left(\sin^2(\theta)\right)^i = \left(1-\cos^2(\theta)\right)^i$$

$$= \left(1-(1-2x)^2\right)^i = \left(4x-4x^2\right)^i\,.$$

For the fraction we obtain

$$\log\left(\frac{S(j,1)}{S(j,x)}\right) = \log\left(\frac{2}{2x+(2x-1)\sum_{i=1}^{j}\binom{2i}{i}x^i(1-x)^i}\right).$$

In Lemma 21 we show that for any $j \in \mathbb{N}+$ and $0 \leq x < \frac{1}{2}$

$$\sum_{i=1}^{j}\binom{2i}{i}x^i(1-x)^i \leq \frac{2x\left((2x)^{2j}-1\right)}{2x-1}.$$

Inserted into the last expression we obtain

$$\begin{aligned}\log\frac{S(j,1)}{S(j,x)} &\leq \log\frac{2}{2x+(2x-1)\frac{2x\left((2x)^{2j}-1\right)}{(2x-1)}}\\ &= \log\frac{2}{(2x)^{2j+1}}\\ &= -(2j+1)\log 2x + \log 2\,,\end{aligned}$$

which proves the theorem. Note that in the case of $x = \frac{1}{2}$ the problem reduces to

$$\log\frac{S(j,1)}{S\left(j,\frac{1}{2}\right)} = \underbrace{-(2j+1)\log 2x}_{0} + \log 2\,.$$

□

A.7.4. Bollmann´s Lemma

In the course of the proof of Theorem (55) we needed a tight upper bound on the growth of $\sum_{i=1}^{j}\binom{2i}{i}x^i(1-x)^i$ as a function of x. In the following we present a series of lemmata resulting in a reasonably accurate upper bound that we call *Bollmann lemma* (Lemma 21), in honour of Peter Bollmann who proved it.

Lemma 17. *For all* $i \in \mathbb{N}^+$

$$\binom{2(i+1)}{i+1} = \binom{2i}{i}\left(4-\frac{2}{i+1}\right).$$

Proof. A straightforward calculation shows that

$$\begin{aligned}\binom{2(i+1)}{i+1} &= \frac{2(i+1)(2i+1)}{(i+1)(i+1)}\binom{2i}{i}\\ &= \binom{2i}{i}\frac{4i+2}{i+1}\\ &= \binom{2i}{i}\left(4-\frac{2}{i+1}\right).\end{aligned}$$

The lemma is proven. □

Lemma 18. *For all* $i \in \mathbb{N}^+$ *and* $j \in \mathbb{N}^+$

$$\binom{2(j+1)}{j+1}\binom{2i}{i} \leq 2\binom{2(i+j)}{i+j}.$$

Proof. We prove the lemma by induction over i. For $i = 1$ it follows that

$$\binom{2(j+1)}{j+1}\binom{2}{1} = 2\binom{2(j+1)}{j+1}.$$

Assume the assertion is true for $i \in \mathbb{N}^+$. Then

$$\begin{aligned}
\binom{2(j+1)}{j+1}\binom{2(i+1)}{i+1} &= \binom{2(j+1)}{j+1}\binom{2i}{i}\left(4 - \frac{2}{i+1}\right) \\
&\leq 2\binom{2(i+j)}{i+j}\left(4 - \frac{2}{i+1}\right) \\
&\leq 2\binom{2(i+j)}{i+j}\left(4 - \frac{2}{i+j+1}\right) \\
&= 2\binom{2(i+j+1)}{i+j+1},
\end{aligned}$$

where we used Lemma 17 in the first and last line. □

Lemma 19. *For all* $0 \leq x < \frac{1}{2}$

$$\sum_{i=1}^{\infty}\binom{2i}{i}x^i(1-x)^i = \frac{2x}{1-2x}.$$

Proof. This can be seen by considering

$$\begin{aligned}
\arcsin(u) &= u + \sum_{i=1}^{\infty}\binom{2i}{i}\frac{1}{4^i}\frac{u^{2i+1}}{2i+1}, \\
\frac{d\arcsin(u)}{du} &= 1 + \sum_{i=1}^{\infty}\binom{2i}{i}\frac{1}{4^i}u^{2i} \\
&= \frac{1}{\sqrt{1-u^2}}.
\end{aligned}$$

Using $u = 2\sqrt{x(1-x)}$ we obtain the result, i.e.

$$\begin{aligned}
\sum_{i=1}^{\infty}\binom{2i}{i}\frac{1}{4^i}\left(2\sqrt{x(1-x)}\right)^{2i} &= \frac{1}{\sqrt{1-4x(1-x)}} - 1 \\
\sum_{i=1}^{\infty}\binom{2i}{i}x^i(1-x)^i &= \frac{1-\sqrt{1-4x(1-x)}}{\sqrt{1-4x(1-x)}} \\
&= \frac{1-\sqrt{(1-2x)^2}}{\sqrt{(1-2x)^2}}.
\end{aligned}$$

The lemma is proven. □

Lemma 20. *For all* $0 \leq x < \frac{1}{2}$ *and* $j \in \mathbb{N}^+$

$$4x^2 \sum_{i=1}^{\infty} \binom{2(i+j)}{i+j} x^{i+j} (1-x)^{i+j} \leq \sum_{i=1}^{\infty} \binom{2(i+j+1)}{i+j+1} x^{i+j+1} (1-x)^{i+j+1} .$$

Proof. By Lemma 18 we have

$$\sum_{i=1}^{\infty} \binom{2i}{i} \binom{2(j+1)}{j+1} x^{i+j} (1-x)^{i+j} \leq \sum_{i=1}^{\infty} 2 \binom{2(i+j)}{i+j} x^{i+j} (1-x)^{i+j} .$$

As $0 < 1 - x \leq 1 + 2x$ it follows that

$$\begin{aligned} & (1-x) \binom{2(j+1)}{j+1} x^j (1-x)^j \sum_{i=1}^{\infty} \binom{2i}{i} x^i (1-x)^i \\ \leq \; & 2(1+2x) \sum_{i=1}^{\infty} \binom{2(i+j)}{i+j} x^{i+j} (1-x)^{i+j} (1-x) \binom{2(j+1)}{j+1} x^j (1-x)^j \frac{2x}{1-2x} \\ \leq \; & 2(1+2x) \sum_{i=1}^{\infty} \binom{2(i+j)}{i+j} x^{i+j+1} (1-x)^{i+j+1} . \end{aligned}$$

Multiplying both sides by $\frac{1-2x}{2}$ (which positive is by assumption) yields

$$\binom{2(j+1)}{j+1} x^{j+1} (1-x)^{j+1} \leq (1-4x^2) \sum_{i=1}^{\infty} \binom{2(i+j)}{i+j} x^{i+j} (1-x)^{i+j} .$$

Rearranging terms gives

$$\begin{aligned} & 4x^2 \sum_{i=1}^{\infty} \binom{2(i+j)}{i+j} x^{i+j} (1-x)^{i+j} \\ \leq \; & \sum_{i=1}^{\infty} \binom{2(i+j)}{i+j} x^{i+j} (1-x)^{i+j} - \binom{2(j+1)}{j+1} x^{j+1} (1-x)^{j+1} \\ = \; & \sum_{i=1}^{\infty} \binom{2(i+j+1)}{i+j+1} x^{i+j} (1-x)^{i+j} . \end{aligned}$$

The lemma is proven. □

Lemma 21. *[Bollmann's Lemma]For any* $j \in \mathbb{N}^+$ *and* $0 < x < \frac{1}{2}$

$$\sum_{i=1}^{j} \binom{2i}{i} x^i (1-x)^i \leq \frac{2x\left((2x)^{2j} - 1\right)}{2x - 1} .$$

Proof. The assertion can be transformed into

$$\begin{aligned}\sum_{i=1}^{j}\binom{2i}{i}x^i\left(1-x\right)^i &\leq \frac{2x\left(\left(2x\right)^{2j}-1\right)}{2x-1}\\ &= \frac{2x\left(1-\left(2x\right)^{2j}\right)}{1-2x}\\ &\leq \frac{2x}{1-2x}-\frac{\left(2x\right)^{2j+1}}{1-2x}\\ &\leq \sum_{i=1}^{\infty}\binom{2i}{i}x^i\left(1-x\right)^i-\frac{\left(2x\right)^{2j+1}}{1-2x},\end{aligned}$$

which is equivalent to

$$\left(2x\right)^{2j+1}\leq\left(1-2x\right)\sum_{i=1}^{\infty}\binom{2\left(i+j\right)}{i+j}x^{i+j}\left(1-x\right)^{i+j}.$$

We prove this by induction over j. For $j=1$ we have

$$\begin{aligned}\binom{2}{1}x\left(1-x\right) &= 2x-2x^2\leq 2x+4x^2=\frac{8x^3-2x}{2x-1}\\ &= \frac{2x\left(\left(2x\right)^2-1\right)}{2x-1}.\end{aligned}$$

Assume the assertion is true for j. Then

$$\begin{aligned}\left(2x\right)^{2(j+1)+1} &= 4x^2\left(2x\right)^{2j+1}\\ &\leq 4x^2\left(\left(1-2x\right)\sum_{i=1}^{\infty}\binom{2\left(i+j\right)}{i+j}x^{i+j}\left(1-x\right)^{i+j}\right)\\ &\leq \left(1-2x\right)\sum_{i=1}^{\infty}\binom{2\left(i+j+1\right)}{i+j+1}x^{i+j+1}\left(1-x\right)^{i+j+1},\end{aligned}$$

where the second line was assumed to be true and the third line follows from Lemma 20. □

A.7.5. Volume Ratio Bound Simplification

Lemma 22 (Volume ratio bound simplification). *For all* $0<x\leq 1$ *we have that*

$$1-\sqrt{1-x^2}\geq\frac{x^2}{2}$$

Proof. We have the following equivalences for $0 < x \leq 1$:

$$\begin{aligned} 1 - \sqrt{1 - x^2} &\geq \frac{x^2}{2} \\ \sqrt{1 - x^2} &\leq 1 - \frac{x^2}{2} \\ 1 - x^2 &\leq 1 - x^2 + \frac{x^4}{4} \\ 0 &\leq \frac{x^4}{4} . \end{aligned}$$

□

A.8. The Cummerbund

Lemma 23 (Cummerbund). *For convex sets $\mathcal{C} \subset \ell_2^n$ we have*

$$\mathcal{W} \setminus \{\mathbf{w} \in \mathcal{W} \,|\, \forall \mathbf{x} \in \mathcal{C} : (\langle \mathbf{w}, \mathbf{x} \rangle < 0 \vee \langle \mathbf{w}, \mathbf{x} \rangle > 0)\} = \tag{A.13}$$

$$\mathcal{W} \setminus \{\mathbf{w} \in \mathcal{W} \,|\, (\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle < 0) \vee (\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle > 0)\} . \tag{A.14}$$

Proof. We need to show that

$$(\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle \neq 0) \Leftrightarrow (\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle < 0) \vee (\forall \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle > 0)$$

which is equivalent to

$$(\exists \mathbf{x} \in \mathcal{C} : \langle \mathbf{w}, \mathbf{x} \rangle = 0) \Leftrightarrow (\exists \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C} : (\langle \mathbf{w}, \mathbf{x}_1 \rangle \geq 0) \wedge (\langle \mathbf{w}, \mathbf{x}_2 \rangle \leq 0)) . \tag{A.15}$$

- "⇒": This is trivially true by noticing that $\mathbf{x} \in \mathcal{C}$ such that $\langle \mathbf{w}, \mathbf{x} \rangle = 0$ implies $\langle \mathbf{w}, \mathbf{x} \rangle \geq 0$ and $\langle \mathbf{w}, \mathbf{x} \rangle \leq 0$.
- "⇐": Suppose the r.h.s. of (A.15) is true and let $c_1 = \langle \mathbf{w}, \mathbf{x}_1 \rangle \geq 0$ and $c_2 = \langle \mathbf{w}, \mathbf{x}_2 \rangle \leq 0$. If $c_1 = 0$ and $c_2 = 0$ then both $\mathbf{x}_1$ and $\mathbf{x}_2$ satisfy the left-hand side of (A.15). Therefore we can assume $c_1 - c_2 > 0$ and $c_1 - c_2 \geq c_1$. Let λ be given by

  $$\lambda = \frac{c_1}{c_1 - c_2} = \frac{-c_1}{c_2 - c_1} \in [0, 1] .$$

  By convexity of $\mathcal{C}$ the point $\mathbf{x} = (1 - \lambda)\, \mathbf{x}_1 + \lambda \mathbf{x}_2 \in \mathcal{C}$. Furthermore, by definition

  $$\begin{aligned} \langle \mathbf{w}, \mathbf{x} \rangle &= \langle \mathbf{w}, (1 - \lambda)\, \mathbf{x}_1 + \lambda \mathbf{x}_2 \rangle \\ &= (1 - \lambda) \langle \mathbf{w}, \mathbf{x}_1 \rangle + \lambda \langle \mathbf{w}, \mathbf{x}_2 \rangle \\ &= (1 - \lambda)\, c_1 + \lambda c_2 \\ &= \lambda\, (c_2 - c_1) + c_1 \\ &= 0 . \end{aligned}$$

The equivalence is established. □

Lemma 24 (Cap cummerbund volume). *Given the normalised data concentration Ξ the volume of the cummerbund of the cap $\mathcal{C}_{\text{cap}}$ containing the data is bounded by*

$$\text{vol}\left(\text{kum}\left(\mathcal{C}_{\text{cap}}\left(\Xi\right)\right)\right) \leq 1 - \left(1 - \sqrt{1-\Xi^2}\right)^n .$$

Proof. The volume of the cummerbund is by definition invariant under any rotation of $\mathcal{C}$. This allows us to choose $\boldsymbol{\phi}^* = (0, 0, \ldots, 1)'$. Thus the convex set $\mathcal{C}_{\text{cap}}\left(\Xi\right)$ is given by

$$\mathcal{C}_{\text{cap}}\left(\Xi\right) = \left\{\mathbf{x} \in \mathcal{K} : \|\mathbf{x}\|^2 = 1 \wedge x_n \geq \Xi\right\} .$$

Now, the cummerbund $\text{kum}\left(\mathcal{C}_{\text{cap}}\left(\Xi\right)\right)$ contains all weight vectors $\mathbf{w}$ of unit length with the property that there exists a vector $\mathbf{x} \in \mathcal{C}_{\text{cap}}\left(\Xi\right)$ with $\langle \mathbf{w}, \mathbf{x} \rangle = 0$. Using the definitions $\mathbf{x} = (\widetilde{\mathbf{x}}', x_n)'$ and $\mathbf{w} = (\widetilde{\mathbf{w}}', w_n)'$ we have

$$\|\widetilde{\mathbf{w}}\|^2 = 1 - w_n^2 ,$$

and

$$\forall \mathbf{x} \in \mathcal{C}_{\text{cap}}\left(\Xi\right) : \quad \|\widetilde{\mathbf{x}}\|^2 = 1 - x_n^2 \leq 1 - \Xi^2 .$$

Furthermore, by the Cauchy-Schwarz inequality applied to $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{w}}$ it follows that

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x} \rangle &= \langle \widetilde{\mathbf{w}}, \widetilde{\mathbf{x}} \rangle + w_n x_n = 0 \\ -w_n x_n &= \langle \widetilde{\mathbf{w}}, \widetilde{\mathbf{x}} \rangle \\ w_n^2 x_n^2 &= \left(\langle \widetilde{\mathbf{w}}, \widetilde{\mathbf{x}} \rangle\right)^2 \leq \|\widetilde{\mathbf{w}}\|^2 \|\widetilde{\mathbf{x}}\|^2 . \end{aligned}$$

Note that for each admissible vector $\widetilde{\mathbf{w}}$ there exists an admissible vector $\widetilde{\mathbf{x}}$ which makes the inequality an equality and thus tight. This implies

$$\begin{aligned} w_n^2 &\leq \frac{\|\widetilde{\mathbf{w}}\|^2 \|\widetilde{\mathbf{x}}\|^2}{x_n^2} \leq \frac{\|\widetilde{\mathbf{w}}\|^2 \left(1 - \Xi^2\right)}{\Xi^2} = \frac{\left(1 - w_n^2\right)\left(1 - \Xi^2\right)}{\Xi^2} \\ w_n^2 \Xi^2 &\leq 1 - \Xi^2 - w_n^2 + w_n^2 \Xi^2 \\ w_n^2 &\leq 1 - \Xi^2 . \end{aligned}$$

Note that this corresponds to a belt of width $2\sqrt{1-\Xi^2}$ (see Figure 4.4) around the sphere $\mathcal{W}$. As a consequence, $\text{vol}\left(\text{kum}\left(\mathcal{C}_{\text{cap}}\left(\Xi\right)\right)\right)$ can be expressed by

$$\begin{aligned} \text{vol}\left(\text{kum}\left(\mathcal{C}_{\text{cap}}\left(\Xi\right)\right)\right) &= \text{vol}\left(\mathcal{W}\right) - 2 \cdot \text{vol}\left(\mathcal{C}_{\text{cap}}\left(\sqrt{1-\Xi^2}\right)\right) \\ &= 1 - 2 \cdot \text{vol}\left(\mathcal{C}_{\text{cap}}\left(\sqrt{1-\Xi^2}\right)\right) . \end{aligned}$$

According to Theorem 55 the last term is lower bounded by

$$\text{vol}\left(\mathcal{C}_{\text{cap}}\left(\sqrt{1-\Xi^2}\right)\right) \geq \frac{1}{2}\left(1 - \sqrt{1-\Xi^2}\right)^n ,$$

which yields

$$\text{vol}\left(\text{kum}\left(\mathcal{C}_{\text{cap}}\left(\Xi\right)\right)\right) \leq 1 - \left(1 - \sqrt{1-\Xi^2}\right)^n .$$

□

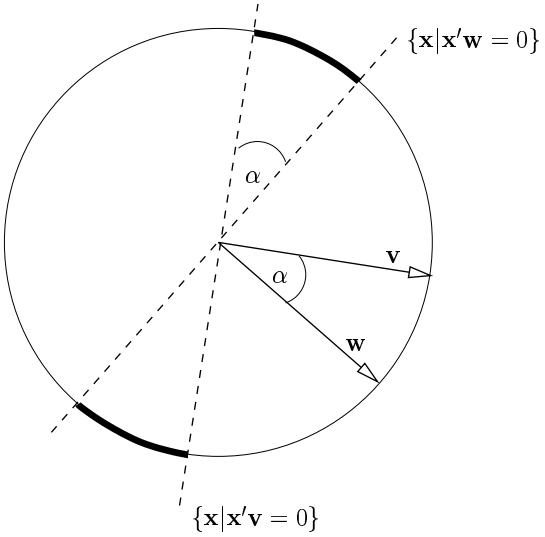


Figure A.2.: The fraction of points on the circle that are classified differently by $\mathbf{w}$ and $\mathbf{v}$ is depicted by the solid black arc. Note that this fraction is in general given by $\frac{2\alpha}{2\pi} = \frac{\alpha}{\pi} = \frac{\arccos(\langle \mathbf{w}, \mathbf{v} \rangle)}{\pi}$.

A.9. The Centre of Mass in Kernel Space

A.9.1. Convergence of Centre of Mass to the Bayes Point

In this section we present the proof of Theorem 47. We start with a simple lemma.

Lemma 25 (Prediction error for spherical input distributions). *Consider the hypothesis space $\mathcal{H}_{\mathcal{K}}$. Furthermore let us assume that P_{X} has a spherical density,*

$$\mathsf{f}_{\mathsf{X}}(\mathbf{x}) \stackrel{\text{def}}{=} \mathsf{f}_{\mathsf{X}}(\|\mathbf{x}\|) .$$

Then, for all $\mathbf{w}$ with $\|\mathbf{w}\| = 1$, and all $\mathbf{v}$ with $\|\mathbf{v}\| = 1$, we have that

$$\mathsf{E}_{\mathsf{X}}\left[l_{0-1}\left(\operatorname{sign}\left(\langle \mathsf{X}, \mathbf{w} \rangle\right), \operatorname{sign}\left(\langle \mathsf{X}, \mathbf{v} \rangle\right)\right)\right] = \frac{1}{\pi} \arccos\left(\langle \mathbf{w}, \mathbf{v} \rangle\right) .$$

Proof. For a fixed value $r \in \mathbb{R}^+$ let us consider all $\mathbf{x}$ such that $\|\mathbf{x}\|^2 = r$. Given two weight vectors $\mathbf{w} \in \mathcal{K}$ and $\mathbf{v} \in \mathcal{K}$ we consider the projection $\mathbf{P}_{\mathbf{w},\mathbf{v}} : \mathcal{K} \to \mathcal{K}$ into the linear space spanned by $\mathbf{w}$ and $\mathbf{v}$ and its complement $\mathbf{P}^{\perp}_{\mathbf{w},\mathbf{v}}$, i.e. we represent all $\mathbf{x} \in \mathcal{K}$ as

$$\mathbf{x} = \mathbf{P}_{\mathbf{w},\mathbf{v}}(\mathbf{x}) + \mathbf{P}^{\perp}_{\mathbf{w},\mathbf{v}}(\mathbf{x}) .$$

Then for all $\mathbf{w}$ (and $\mathbf{v}$) we have

$$\operatorname{sign}\left(\langle \mathbf{x}, \mathbf{w} \rangle\right) = \operatorname{sign}\left(\langle \mathbf{P}_{\mathbf{w},\mathbf{v}}(\mathbf{x}), \mathbf{w} \rangle\right) .$$

Hence for any value of $r \in \mathbb{R}^+$ the fraction of inputs that are misclassified (see Figure A.2) is given by

$$\mathbf{E}_{\mathbf{X}|\,\|\mathbf{X}\|=r}\left[l_{0-1}\left(\operatorname{sign}\left(\langle \mathbf{X}, \mathbf{w}\rangle\right), \operatorname{sign}\left(\langle \mathbf{X}, \mathbf{v}\rangle\right)\right)\right] = \frac{1}{\pi}\arccos\left(\langle \mathbf{w}, \mathbf{v}\rangle\right) .$$

Thus integrating over shells of radius r results in

$$\mathbf{E}_{\mathbf{X}}\left[l_{0-1}\left(\operatorname{sign}\left(\langle \mathbf{X}, \mathbf{w}\rangle\right), \operatorname{sign}\left(\langle \mathbf{X}, \mathbf{v}\rangle\right)\right)\right] = \frac{1}{\pi}\arccos\left(\langle \mathbf{w}, \mathbf{v}\rangle\right) .$$

□

According to Definition 69 and the previous lemma, in order to find the Bayes point $\mathbf{w}_{\mathrm{bp}}$ for a given training sample $\boldsymbol{z}$ we need to find the vector $\mathbf{v}$ that minimises the following function

$$\begin{aligned}\mathbf{E}_{\mathbf{X}}\left[\mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}\left[l_{0-1}\left(h_{\mathbf{v}}\left(\mathbf{X}\right), h_{\mathbf{W}}\left(\mathbf{X}\right)\right)\right]\right] &= \mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}\left[\mathbf{E}_{\mathbf{X}}\left[l_{0-1}\left(h_{\mathbf{v}}\left(\mathbf{X}\right), h_{\mathbf{W}}\left(\mathbf{X}\right)\right)\right]\right] \\ &= \mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}\left[\frac{1}{\pi}\arccos\left(\langle \mathbf{v}, \mathbf{W}\rangle\right)\right]\end{aligned}$$

subject to the constraint $\|\mathbf{v}\| = 1$. Hence we have to determine the saddle point of the following Lagrangian

$$L_{\mathrm{exact}}\left(\mathbf{v}, \alpha\right) = \mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}\left[\frac{1}{\pi}\arccos\left(\langle \mathbf{v}, \mathbf{W}\rangle\right)\right] + \alpha\left(\|\mathbf{v}\|^2 - 1\right) ,$$

w.r.t. $\mathbf{v}$ and α. The difficulty with this expression, however, is that the stationarity conditions,

$$\nabla_{\mathbf{v}} L_{\mathrm{exact}}\left(\mathbf{v}, \alpha\right)|_{\mathbf{v}_{\mathrm{bp}}} = -\mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}\left[\left(1 - \left(\langle \mathbf{v}_{\mathrm{bp}}, \mathbf{W}\rangle\right)^2\right)^{-\frac{1}{2}} \mathbf{W}\right] + 2\alpha\mathbf{v}_{\mathrm{bp}} = \mathbf{0} ,$$

lead to coupled fix-point equations for the components of $\mathbf{v}_{\mathrm{bp}}$,

$$2\alpha\mathbf{v}_{\mathrm{bp}} = \mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}\left[\left(1 - \left(\langle \mathbf{v}_{\mathrm{bp}}, \mathbf{W}\rangle\right)^2\right)^{-\frac{1}{2}} \mathbf{W}\right] .$$

Nevertheless, we can find a good proxy for $\frac{1}{\pi}\arccos\left(\langle \mathbf{w}, \mathbf{v}\rangle\right)$ by $\frac{1}{2}\left(1 - \langle \mathbf{w}, \mathbf{v}\rangle\right)$ (see Figure A.3). This is made more precise in the following lemma.

Lemma 26 (Quality of Euclidean distance proxy). *Consider the hypothesis space $\mathcal{H}_{\mathcal{K}}$. Furthermore let us assume that $\mathbf{P}_{\mathbf{X}}$ has a spherical density, $\mathbf{f}_{\mathbf{X}}\left(\mathbf{x}\right) \stackrel{\mathrm{def}}{=} \mathbf{f}_{\mathbf{X}}\left(\|\mathbf{x}\|\right)$. Then for any vector $\mathbf{v} \in \mathcal{K}$ of unit length, $\|\mathbf{v}\| = 1$, satisfying*

$$\min_{\mathbf{w}:\mathbf{P}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}(\mathbf{w})>0} \langle \mathbf{v}, \mathbf{w}\rangle > \varepsilon . \tag{A.16}$$

we have that

$$\mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}\left[\mathbf{E}_{\mathbf{X}}\left[l_{0-1}\left(h_{\mathbf{v}}\left(\mathbf{X}\right), h_{\mathbf{W}}\left(\mathbf{X}\right)\right)\right]\right] \leq \mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}\left[\frac{1}{4}\left\|\mathbf{W} - \mathbf{v}\right\|^2\right] + \kappa\left(\varepsilon\right) ,$$

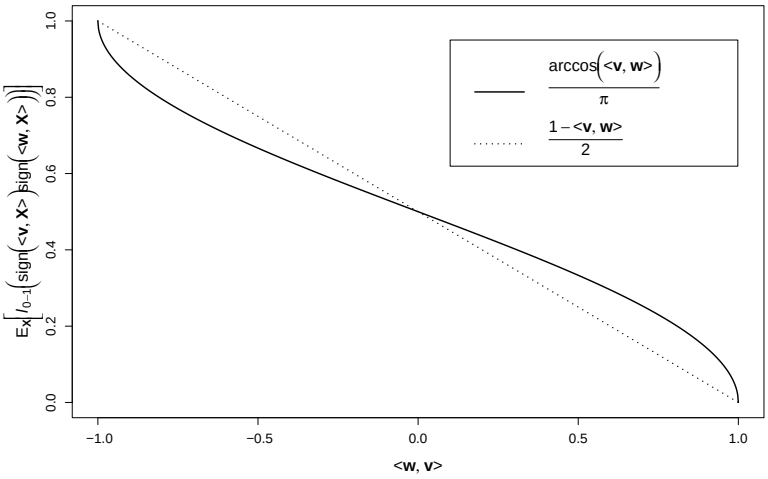


Figure A.3.: Plot of $\frac{1}{\pi}\arccos(x)$ and $\frac{1}{2}(1-x)$ as functions of $x \equiv \langle \mathbf{v}, \mathbf{w} \rangle$. It can be seen that shifting up the graph of $\frac{1}{2}(1-x)$ by 0.11 leads to an upper bound for $\frac{1}{\pi}\arccos(\langle \mathbf{v}, \mathbf{w} \rangle)$.

where

$$\kappa(\varepsilon) = \begin{cases} \frac{1}{\pi}\arccos(\varepsilon) - \frac{1}{2}(1-\varepsilon) & \text{if} \quad \varepsilon \geq \sqrt{1-\frac{4}{\pi^2}} \\ 0.11 & \text{otherwise} \end{cases} ,$$

Proof. Using Lemma 25 we only need to show that under the assumption (A.16) it holds true that

$$\frac{1}{\pi}\arccos(\langle \mathbf{v}, \mathbf{w} \rangle) \leq \frac{1}{4}\|\mathbf{v}-\mathbf{w}\|^2 + \kappa(\varepsilon) .$$

At first we notice that for normalised $\mathbf{v}$ and $\mathbf{w}$ we have

$$\frac{1}{4}\|\mathbf{v}-\mathbf{w}\|^2 = \frac{1}{2}(1-\langle \mathbf{v}, \mathbf{w} \rangle) .$$

We determine the extrema of the function $f : [-1,1] \to [0,1]$ given by

$$f(x) = \frac{1}{\pi}\arccos(x) - \frac{1}{2}(1-x) .$$

Differentiation w.r.t. x gives $f'(x) = -\frac{1}{\pi\sqrt{1-x^2}} + \frac{1}{2}$ which has zeros at $x^* = \pm\sqrt{1-\frac{4}{\pi^2}}$. For the value of $f(x^*)$ we have $0.10 < |f(x^*)| < 0.11$. Hence, whenever $|\varepsilon| > \sqrt{1-\frac{4}{\pi^2}} > 0.77$ we can directly use $f(x)$ which itself is in the worst case upper bounded by 0.11. Noticing that that $f(x)$ is antisymmetric proves the lemma. □

If we replace $\frac{1}{\pi}\arccos(\langle \mathbf{w}, \mathbf{v} \rangle)$ by $\frac{1}{4}\|\mathbf{w}-\mathbf{v}\|^2$ on the basis of the previous lemma we obtain the simpler problem of determining the saddle point of the Lagrangian

$$L_{\text{approx}}(\mathbf{v}, \alpha) = \mathbf{E}_{\mathbf{W}|\mathbf{Z}^m=\boldsymbol{z}}\left[\frac{1}{4}\|\mathbf{W}-\mathbf{v}\|^2\right] + \alpha\left(\|\mathbf{v}\|^2 - 1\right) . \tag{A.17}$$

Then the stationarity conditions,

$$\nabla_{\mathbf{v}} L_{\text{approx}}(\mathbf{v}, \alpha)|_{\mathbf{v}_{\text{cm}}} = \mathbf{E}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}\left[-\frac{1}{2}\mathbf{W}\right] + 2\alpha\mathbf{v}_{\text{cm}} = \mathbf{0}\,,$$

lead to

$$2\alpha\mathbf{v}_{\text{cm}} = \mathbf{E}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}\left[\frac{1}{2}\mathbf{W}\right]\,.$$

The value of α is determined by multiplying the whole expression by $\mathbf{v}_{\text{cm}}$ and applying the constraint $\mathbf{v}'_{\text{cm}}\mathbf{v}_{\text{cm}} = 1$, i.e.

$$\alpha = \frac{1}{4}\mathbf{E}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}\left[\langle \mathbf{W}, \mathbf{v}_{\text{cm}}\rangle\right]\,.$$

Re-substituting this expression into (A.17) results in

$$\mathbf{v}_{\text{cm}} = \frac{1}{4\alpha}\mathbf{E}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}\left[\mathbf{W}\right] = \frac{\mathbf{E}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}\left[\mathbf{W}\right]}{\left\langle \mathbf{E}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}\left[\mathbf{W}\right], \mathbf{v}_{\text{cm}}\right\rangle}\,.$$

The only solution to this equation is given by

$$\mathbf{v}_{\text{cm}} = \frac{\mathbf{E}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}\left[\mathbf{W}\right]}{\left\|\mathbf{E}_{\mathbf{W}|\mathsf{Z}^m=\boldsymbol{z}}\left[\mathbf{W}\right]\right\|}\,.$$

B. Definitions and Useful Results

B.1. Basic Definitions and Notation of Probability Theory

Since a great amount of work in this thesis is based on probability theory this section provides some basic definitions and notation.

Definition 73 (Set and indicator function). A set is a collection of objects. Sets are denoted by upper-case Roman or calligraphic letters, i.e., for example by X or $\mathcal{X}$. Elements of sets are denoted by lower-case Roman letters, e.g., x. For any set $\mathcal{X}$ and subset $X \subseteq \mathcal{X}$ the indicator function $\mathbb{I}_X : \mathcal{X} \to \{0, 1\}$ is defined for all $x \in \mathcal{X}$ as

$$\mathbb{I}_X(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if} \quad x \in X \\ 0 & \text{if} \quad x \notin X \end{cases} .$$

Definition 74 (Logical formula). We define a logical formula Υ on a set $\mathcal{X}$ as a function

$$\Upsilon : \mathcal{X} \to \{\text{true}, \text{false}\}$$

and use the short–hand notation

$$\mathbb{I}_{\Upsilon(x)} \stackrel{\text{def}}{=} \mathbb{I}_{\{x \in \mathcal{X} | \Upsilon(x) = \text{true}\}} .$$

Definition 75 (σ-Algebra). Given a set $\mathcal{X}$ we call $\mathfrak{X}$ a σ-algebra over $\mathcal{X}$ if

1. $\forall X \in \mathfrak{X} : \; X \subseteq \mathcal{X}$
2. $\forall X \in \mathfrak{X} : \; \mathcal{X} \setminus X \in \mathfrak{X}$
3. $\forall \{X_1, \ldots, X_i, \ldots\} \subseteq \mathfrak{X}$ countable, $\bigcup_{i=1}^{\infty} X_i \in \mathfrak{X}$ and $\bigcap_{i=1}^{\infty} X_i \in \mathfrak{X}$

Definition 76 (Borel sets). The Borel sets $\mathcal{B}_n$ are the smallest σ-algebra over $\mathbb{R}^n$ that contains all open intervals

$$\{(x_1, \ldots, x_n) \in \mathbb{R}^n \,|\forall i \in \{1, \ldots, n\} : x_i \in (a_i, b_i)\}$$

for all $a_i, b_i \in \mathbb{R}^n$.

Definition 77 (Measure and sample space). Given a set $\mathcal{X}$ and a σ–algebra $\mathfrak{X}$ over $\mathcal{X}$ we call the tuple $(\mathcal{X}, \mathfrak{X})$ a *measure space* and $\mathcal{X}$ the corresponding *sample space.*

Definition 78 (Probability measure and probability space). Given a measure space $(\mathcal{X}, \mathfrak{X})$ we define a *probability measure* $\mathbf{P}_{\mathsf{X}}$ on $\mathcal{X}$ as a function $\mathbf{P}_{\mathsf{X}} : \mathfrak{X} \to \mathbb{R}$ satisfying Kolmogorov´s axioms

1. $\mathbf{P}_{\mathsf{X}}(X) \geq 0$ for all $X \in \mathfrak{X}$
2. $\mathbf{P}_{\mathsf{X}}(\mathcal{X}) = 1$
3. $\mathbf{P}_{\mathsf{X}}(\bigcup_{i=1}^{\infty} X_i) = \sum_{i=1}^{\infty} \mathbf{P}_{\mathsf{X}}(X_i)$ for all $\{X_1, \ldots, X_i, \ldots\} \subseteq \mathfrak{X}$ countable and satisfying $\forall i, j \in X_i \cap X_j = \emptyset$.

Sets $X \in \mathfrak{X}$ are also called *events* and the triple $(\mathcal{X}, \mathfrak{X}, \mathbf{P}_{\mathsf{X}})$ is called a *probability space.*

Given a logical formula $\Upsilon : \mathcal{X} \to \{\text{true}, \text{false}\}$ over $\mathcal{X}$ we use the short–hand notation

$$\mathbf{P}_{\mathsf{X}}(\Upsilon(\mathsf{X})) \stackrel{\text{def}}{=} \mathbf{P}_{\mathsf{X}}(\{x \in \mathcal{X} \,|\, \Upsilon(x) = \text{true}\}) \,.$$

Definition 79 (Conditional probability measure). Given a probability space $(\mathcal{X}, \mathfrak{X}, \mathbf{P}_{\mathsf{X}})$ we define for all events $Y \in \mathfrak{X}$ with $\mathbf{P}_{\mathsf{X}}(Y) \neq 0$ and for all events $X \in \mathfrak{X}$ the conditional probability measure

$$\mathbf{P}_{\mathsf{X}|\mathsf{X}\in X}(X) \stackrel{\text{def}}{=} \frac{\mathbf{P}_{\mathsf{X}}(X \cap Y)}{\mathbf{P}_{\mathsf{X}}(Y)} \,.$$

Definition 80 (Measurability and random variable). Given a measure space $(\mathcal{X}, \mathfrak{X})$, a function $f : \mathcal{X} \to \mathbb{R}$ is called $\mathfrak{X}$–*measurable* if and only if for all $y \in \mathbb{R}$,

$$\{x \in \mathcal{X} \,|\, f(x) \leq y\} \in \mathfrak{X} \,.$$

Such a function is also called a *random variable.*

Definition 81 (Distribution function and density). For a random variable X we define the distribution function as

$$\mathbf{F}_{\mathsf{X}}(x) \stackrel{\text{def}}{=} \mathbf{P}_{\mathsf{X}}(\mathsf{X} \leq x) \,,$$

and its density if $\mathbf{F}$ is differentiable as

$$\mathbf{f}_{\mathsf{X}}(x) \stackrel{\text{def}}{=} \frac{d\mathbf{F}_{\mathsf{X}}(x)}{dx} \,.$$

Definition 82 (Support). Given a random variable X we call the smallest interval $[a, b]$ with

$$\mathbf{P}_{\mathsf{X}}\left(\mathsf{X} \in [a, b]\right) = 1$$

the *support* $\operatorname{supp}\mathsf{X}$ of X.

Definition 83 (Expectation value). The *expectation value* $\mathbf{E}_{\mathsf{X}}\left[\mathsf{X}\right]$ of a random variable X is defined as

$$\mathbf{E}_{\mathsf{X}}\left[\mathsf{X}\right] \stackrel{\text{def}}{=} \lim_{n\to\infty} \sum_{i=-n}^{+n} \frac{i}{n}\left(\mathbf{F}_{\mathsf{X}}\left(\frac{i}{n}\right) - \mathbf{F}_{\mathsf{X}}\left(\frac{i-1}{n}\right)\right) = \int_{-\infty}^{+\infty} x\, d\mathbf{F}_{\mathsf{X}}\left(x\right) ,$$

provided that the limit converges.

Definition 84 (Variance). The *variance* $\operatorname{Var}\left(\mathsf{X}\right)$ of a random variable X is defined as

$$\operatorname{Var}\left(\mathsf{X}\right) \stackrel{\text{def}}{=} \mathbf{E}_{\mathsf{X}}\left[\left(\mathsf{X} - \mathbf{E}_{\mathsf{X}}\left[\mathsf{X}\right]\right)^2\right] ,$$

provided that both X and $\left(\mathsf{X} - \mathbf{E}_{\mathsf{X}}\left[\mathsf{X}\right]\right)^2$ have an expectation value.

Definition 85 (Product space). Given two measure spaces $(\mathcal{X}, \mathfrak{X})$ and $(\mathcal{Y}, \mathfrak{Y})$ the product measure space is defined as $(\mathcal{X} \times \mathcal{Y}, \mathfrak{X} \times \mathfrak{Y})$ and given probability measures $\mathbf{P}_{\mathsf{X}}$ and $\mathbf{P}_{\mathsf{Y}}$ the product probability space is defined as $(\mathcal{X} \times \mathcal{Y}, \mathfrak{X} \times \mathfrak{Y}, \mathbf{P}_{\mathsf{XY}})$.

Definition 86 (Marginal and conditional probability measure). Given the joint probability space $(\mathcal{X} \times \mathcal{Y}, \mathfrak{X} \times \mathfrak{Y}, \mathbf{P}_{\mathsf{XY}})$, then for all $X \in \mathfrak{X}$ the *marginal probability measure* $\mathbf{P}_{\mathsf{X}}$ is defined by

$$\mathbf{P}_{\mathsf{X}}\left(X\right) \stackrel{\text{def}}{=} \mathbf{P}_{\mathsf{XY}}\left(X \times \mathcal{Y}\right) ,$$

and, given $Y \in \mathfrak{Y}$ with $\mathbf{P}_{\mathsf{Y}}\left(Y\right) \neq 0$, the *conditional probability measure* $\mathbf{P}_{\mathsf{X}|\mathsf{Y}=Y}$ is defined by

$$\mathbf{P}_{\mathsf{X}|\mathsf{Y}=Y}\left(X\right) \stackrel{\text{def}}{=} \frac{\mathbf{P}_{\mathsf{XY}}\left(X \times Y\right)}{\mathbf{P}_{\mathsf{Y}}\left(Y\right)} ,$$

and vice versa.

Definition 87 (Independence). Two random variables X and Y are called *independent* w.r.t. the product measure $\mathbf{P}_{\mathsf{XY}}$, if and only if for all $X \in \mathfrak{X}$ and $Y \in \mathfrak{Y}$,

$$\mathbf{P}_{\mathsf{XY}}\left(X \times Y\right) = \mathbf{P}_{\mathsf{X}}\left(X\right) \cdot \mathbf{P}_{\mathsf{Y}}\left(Y\right) .$$

B.2. Some Basic Definitions from Convex Analysis

These basic definitions are used in Section A.8 and can be found, for example, in Rockafellar (1970).

Definition 88 (Convex set). A set $\mathcal{C} \subset \ell_2^n$ is called *convex* if and only if for all $\lambda \in [0,1]$ and all $\mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{C}$ we have

$$\lambda \mathbf{x} + (1-\lambda)\,\tilde{\mathbf{x}} \in \mathcal{C}\,.$$

Definition 89 (Polar). Suppose we are given a convex set $\mathcal{C} \subset \ell_2^n$ and a feature space $\mathcal{K} \subseteq \ell_2^n$ such that $\dim(\mathcal{K}) = n$. Then the *closed polar* $\overline{\mathrm{pol}}\,(\mathcal{C})$ *of* $\mathcal{C}$ is defined by

$$\overline{\mathrm{pol}}\,(\mathcal{C}) = \{\mathbf{w} \in \mathcal{W} \,|\forall \mathbf{x} \in \mathcal{C} :\, \langle \mathbf{w}, \mathbf{x}\rangle \leq 0\}\,.$$

Accordingly, the *open polar* $\mathrm{pol}\,(C)$ is defined by

$$\mathrm{pol}\,(\mathcal{C}) = \{\mathbf{w} \in \mathcal{W} \,|\forall \mathbf{x} \in \mathcal{C} :\, \langle \mathbf{w}, \mathbf{x}\rangle < 0\}\,.$$

Definition 90 (Convex hull). Suppose we are given a set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \in \ell_2^n$. The *convex hull* $\mathrm{conv}\,(X)$ of X is defined as the intersection of all convex sets containing X, i.e.

$$\mathrm{conv}\,(X) = \bigcap_{\mathcal{C}_i \in \ell_2^n : X \subseteq \mathcal{C}_i} \mathcal{C}_i\,.$$

B.3. Useful Results

Theorem 56 (Euler´s inequality). *For all* $x > 0$ *and* $a \in \mathbb{R}$, *if* $a \neq 0$

$$\left(1 + \frac{a}{x}\right)^x < e^a\,.$$

Proof. First we show that for all $y \in \mathbb{R}$ we have $1 + y \leq \exp(y)$ with equality if and only if $y = 0$. To this end consider the function $f(y) = 1 + y - \exp(y)$. The first and second derivative of this function are $\frac{df(y)}{dy} = 1 - \exp(y)$ and $\frac{d^2 f(y)}{dy^2} = -\exp(y)$, respectively. Hence f has a maximum at $y_{\max} = 0$. This implies that $\forall y\, f(y) \leq f(0) = 0$ which is equivalent to $1 + y \leq \exp(y)$. Now choosing $y = a/x$ we have $1 + \frac{a}{x} < \exp\left(\frac{a}{x}\right)$ since by assumption $a \neq 0$. For $x > 0$ we then have $\left(1 + \frac{a}{x}\right)^x < \left(\exp\left(\frac{a}{x}\right)\right)^x = \exp(a)$ as stated by the theorem. □

Theorem 57 (Union bound). *Given a probability space* $(\mathcal{X}, \mathfrak{X}, \mathsf{P}_\mathsf{X})$ *and a finite number of events* $X_1, \ldots, X_n \in \mathfrak{X}$ *then we have that*

$$\mathsf{P}_\mathsf{X}\left(\bigcup_{i=1}^n X_i\right) \leq \sum_{i=1}^n \mathsf{P}_\mathsf{X}(X_i)\,.$$

Proof. Consider any two sets $A, B \in \mathcal{X}$. From the axioms of probability we have that

$$\begin{aligned} \mathbf{P}_{\mathsf{X}}(A \cup B) &= \mathbf{P}_{\mathsf{X}}(A) + \mathbf{P}_{\mathsf{X}}(B) - \mathbf{P}_{\mathsf{X}}(A \cap B) \\ &\leq \mathbf{P}_{\mathsf{X}}(A) + \mathbf{P}_{\mathsf{X}}(B) \ . \end{aligned}$$

The theorem follows from repeated application of this relation. □

Theorem 58 (Binomial tail bound). *Let* $\mathsf{X}_1, \ldots, \mathsf{X}_n$ *be independent random variables distributed* Bernoulli (μ). *Then we have that*

$$\mathbf{P}_{\mathsf{X}^n}\left(\sum_{i=1}^{n} \mathsf{X}_i = 0\right) \leq \exp(-n\mu) \ .$$

Proof. From the independence of the X_i we have that

$$\mathbf{P}_{\mathsf{X}^n}\left(\sum_{i=1}^{n} \mathsf{X}_i = 0\right) = (1-\mu)^n \leq \exp(-n\mu) \ ,$$

where the second step follows from Euler´s inequality. □

Theorem 59 (Markov´s inequality). *For any random variable* X *with* $\mathbf{F}_{\mathsf{X}}(0) = 0$ *we have for all* $\lambda > 0$ *that*

$$\mathbf{P}_{\mathsf{X}}(\mathsf{X} > \lambda \cdot \mathbf{E}_{\mathsf{X}}[\mathsf{X}]) < \frac{1}{\lambda}.$$

Theorem 60 (Chebyshev´s inequality). *If* $\mathbf{E}_{\mathsf{X}}\left[\mathsf{X}^2\right]$ *exists then we have for all* $\varepsilon > 0$,

$$\mathbf{P}_{\mathsf{X}}(|\mathsf{X}| > \varepsilon) < \frac{\mathbf{E}_{\mathsf{X}}\left[\mathsf{X}^2\right]}{\varepsilon^2},$$

and in particular

$$\mathbf{P}_{\mathsf{X}}(|\mathsf{X} - \mathbf{E}_{\mathsf{X}}[\mathsf{X}]| > \varepsilon) < \frac{\mathrm{Var}(\mathsf{X})}{\varepsilon^2}.$$

Theorem 61 (Hoeffding´s inequality (Hoeffding 1963)). *Given* n *independent bounded random variables* $\mathsf{X}_1, \ldots, \mathsf{X}_n$ *such that for* $a, b \in \mathbb{R}$, $a < b$ $\forall i$ $\mathbf{P}_{\mathsf{X}_i}(\mathsf{X}_i \in [a,b]) = 1$, *then we have for all* $\varepsilon > 0$

$$\mathbf{P}_{\mathsf{X}^n}\left(\frac{1}{n}\sum_{i=1}^{n} \mathsf{X}_i - \mathbf{E}_{\mathsf{X}}[\mathsf{X}] > \varepsilon\right) < \exp\left(-\frac{2n\varepsilon^2}{(b-a)^2}\right) .$$

Theorem 62 (Cauchy-Schwarz inequality). *For any two elements* $\mathbf{x}, \tilde{\mathbf{x}} \in \ell_2^n$ *we have*

$$|\langle \mathbf{x}, \tilde{\mathbf{x}} \rangle| \leq \|\mathbf{x}\| \cdot \|\tilde{\mathbf{x}}\| \,.$$

Theorem 63 (Simple Stirling´s approximation). *For all* $n \in \mathbb{N}$ *we have*

$$n\left(\log\left(n\right) - 1\right) < \log\left(n!\right) < n \log\left(n\right)$$

Theorem 64 (Binomial theorem). *For all* $a \in \mathbb{R}$ *and* $d \in \mathbb{N}$

$$(1+a)^d = \sum_{i=0}^{d} \binom{d}{i} a^i \,.$$

Theorem 65 (Bound on the binomial coefficient). *For all* $m, d \in \mathbb{N}$ *with* $m \geq d$ *we have*

$$\log \binom{m}{d} \leq d \log \frac{em}{d}$$

Proof. We first show that under the conditions of the theorem

$$\sum_{i=1}^{d} \binom{m}{i} < \left(\frac{em}{d}\right)^d \,. \tag{B.1}$$

Using $\left(\frac{m}{d}\right)^d \left(\frac{d}{m}\right)^i \geq 1$ for $0 \leq i \leq d$ and $m \geq d$, we obtain

$$\begin{aligned} \sum_{i=1}^{d} \binom{m}{i} &\leq \left(\frac{m}{d}\right)^d \sum_{i=1}^{d} \binom{m}{i} \left(\frac{d}{m}\right)^i \\ &= \left(\frac{m}{d}\right)^d \left(1 + \frac{d}{m}\right)^m \\ &< \left(\frac{em}{d}\right)^d \,, \end{aligned}$$

where the equality follows from the binomial theorem, Theorem 64, and the last inequality is a consequence of Euler´s inequality Theorem 56. Since all the summands in (B.1) are positive the inequality must hold also for the summand $i = d$, which—taking logarithms on both sides—proves the theorem. □

C. Pseudocode

C.1. Perceptron Algorithm

C.1.1. Primal Perceptron Learning

Algorithm 1 Primal Perceptron Learning

Require: A training sample $\boldsymbol{z}$
Require: feature mapping $\boldsymbol{\phi}(x) : \mathcal{X} \to \mathcal{K}$
Ensure: Linear separability of $\boldsymbol{z}$ in $\mathcal{K}$
  $\mathbf{w} = \mathbf{0}$
  repeat
    for $i = 1, \ldots, m$ **do**
      if $y_i \langle \mathbf{x}_i, \mathbf{w} \rangle \leq 0$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_i y_i$
      end if
    end for
  until the **if** branch was never entered within the **for** loop

C.1.2. Dual Perceptron Learning

Algorithm 2 Dual (Kernel) Perceptron Learning

Require: A training sample $\boldsymbol{z}$
Require: A Mercer kernel k corresponding to a feature space $\mathcal{K}$.
Ensure: Linear separability of $\boldsymbol{z}$ in $\mathcal{K}$
  $\boldsymbol{\alpha} = \mathbf{0}$
  repeat
    for $i = 1, \ldots, m$ **do**
      if $y_i \sum_{j=1}^{m} \alpha_j k(x_i, x_j) \leq 0$ **then**
        $\alpha_i \leftarrow \alpha_i + y_i$
      end if
    end for
  until the **if** branch was never entered within the **for** loop

C.2. Support Vector Machine

Algorithm 3 Support Vector Machine (1-Norm Soft Margin) without additive bias.

Require: A QP solver $\mathcal{QP}\left(\mathbf{H}, \mathbf{c}, \mathbf{A}_1, \mathbf{A}_2, \mathbf{b}_1, \mathbf{b}_2, \mathrm{l}, \mathbf{u}\right) = \mathbf{x}^*$ where $\mathbf{x}^*$ is the solution to minimise$_{\mathbf{x}\in\mathbb{R}^n}$ $\frac{1}{2}\mathbf{x}'\mathbf{H}\mathbf{x} + \mathbf{c}'\mathbf{x}$ subject to $\mathbf{A}_1\mathbf{x} \leq \mathbf{b}_1,\ \mathbf{A}_2\mathbf{x} = \mathbf{b}_2,\ \mathrm{l} \leq \mathbf{x} \leq \mathbf{u}$.
Require: A training sample $\boldsymbol{z}$
Require: A Mercer kernel k corresponding to a feature space $\mathcal{K}$
$\mathbf{H} \leftarrow \mathbf{YKY}$
$\mathbf{c} \leftarrow -\mathbf{1}_m$
$\mathrm{l} \leftarrow \mathbf{0}_m$
$\mathbf{u} \leftarrow \frac{C}{m}\mathbf{1}_m$
$\boldsymbol{\alpha}^* \leftarrow \mathcal{QP}\left(\mathbf{H}, \mathbf{c}, \mathrm{l}, \mathbf{u}\right)$

C.3. Kernel Billiard Sampler

Algorithm 4 Kernel billiard sampler

Require: A training sample $\boldsymbol{z}$
Require: A Mercer kernel k corresponding to a feature space $\mathcal{K}$.
Ensure: Linear separability of $\boldsymbol{z}$ in $\mathcal{K}$
Require: Number of samples N
Require: Maximum flight time $\tau_{\max} \in \mathbb{R}^+$
Ensure: $y_i \sum_j \gamma_j k(x_i, x_j) > 0$ for all $i = 1, \ldots, m$
  $\boldsymbol{\beta}$ = random; normalise $\boldsymbol{\beta}$ using equation (8.1)
  for $j \leftarrow 1, \ldots, N$ **do**
    repeat
      for $i \leftarrow 1, \ldots, m$ **do**
        $d_i \leftarrow y_i \sum_j \gamma_j k(x_j, x_i)$
        $\nu_i \leftarrow y_i \sum_j \beta_j k(x_j, x_i)$
        $\tau_i \leftarrow -d_i/\nu_i$
      end for
      $c' \leftarrow \min_{i:\tau_i>0} \tau_i$
      if $\tau_{c'} \geq \tau_{\max}$ **then**
        $\boldsymbol{\beta} \leftarrow$ random, but satisfies Equation (8.3)
        normalise $\boldsymbol{\beta}$ using Equation (8.1)
      else
        $c \leftarrow c'$
      end if
    until $\tau_{c'} < \tau_{\max}$
    $\boldsymbol{\gamma}' \leftarrow \boldsymbol{\gamma} + \tau_c \boldsymbol{\beta}$; normalise $\boldsymbol{\gamma}'$ using equation (8.1)
    $\beta_c \leftarrow \beta_c - 2\nu_c y_c / k(x_c, x_c)$; normalise $\boldsymbol{\beta}$ using Equation (8.1)
    $\boldsymbol{\alpha}_j \leftarrow \boldsymbol{\gamma}$
  end for

C.4. Kernel Gibbs Sampler

Algorithm 5 Kernel Gibbs sampler

Require: A training sample $\boldsymbol{z}$
Require: A Mercer kernel k corresponding to a feature space $\mathcal{K}$
Require: Number of samples N
Require: Noise level θ
  $\boldsymbol{\alpha} \leftarrow$ random; normalise $\boldsymbol{\alpha}$ using equation (8.1)
  for $j \leftarrow 1, \ldots, N$ **do**
    $\boldsymbol{\beta} \leftarrow$ random;
    $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \sum_{i,j=1}^{m} \alpha_i \beta_j k(x_i, x_j) \boldsymbol{\alpha}$; normalise $\boldsymbol{\beta}$ using Equation (8.1)
    for $i \leftarrow 1, \ldots, m$ **do**
      $\zeta_i \leftarrow \arccos\left(\frac{\sum_j \beta_j k(x_j, x_i)}{\sqrt{k(x_i, x_i)}}\right)$
      $\zeta_{m+i} \leftarrow \zeta_i + \pi$
    end for
    Sort the ζ_i in ascending order such that $\zeta_{\Pi(i)} \leq \zeta_{\Pi(i)+1}$for all $i \in \{1, \ldots, 2m-1\}$
    for $i \leftarrow 1, \ldots, m$ **do**
      $\boldsymbol{\gamma} \leftarrow \cos\left(\frac{\zeta_{\Pi(i+1)} - \zeta_{\Pi(i)}}{2}\right) \boldsymbol{\alpha} - \sin\left(\frac{\zeta_{\Pi(i+1)} - \zeta_{\Pi(i)}}{2}\right) \boldsymbol{\beta}$
      $e_i \leftarrow \hat{R}[\boldsymbol{\gamma}]$
    end for
    Sample an angle ζ^* from $\mathbf{f}(\zeta) \propto \sum_{i=1}^{2m} \mathbf{I}_{\zeta_{\Pi(i)} \leq \zeta \leq \zeta_{\Pi(i+1)}} \theta^{m e_i} (1-\theta)^{m(1-e_i)}$
    $\boldsymbol{\alpha} \leftarrow \cos(\zeta^*) \boldsymbol{\alpha} - \sin(\zeta^*) \boldsymbol{\beta}$
    $\boldsymbol{\alpha}_j \leftarrow \boldsymbol{\alpha}$
  end for

C.5. Permutational Perceptron Sampler

Algorithm 6 Permutational perceptron sampler

Require: A training sample $\boldsymbol{z}$
Require: A Mercer kernel k corresponding to a feature space $\mathcal{K}$
Ensure: Linear separability of $\boldsymbol{z}$ in $\mathcal{K}$
Require: Number of samples N
  for $j \leftarrow 1, \ldots, N$ **do**
    Generate a random permutation $\Pi : \{1, \ldots, m\} \to \{1, \ldots, m\}$
    $\boldsymbol{\alpha} \leftarrow \mathbf{0}$
    $\mathbf{o} \leftarrow \mathbf{0}$
    repeat
      for $i \leftarrow 1, \ldots, m$ **do**
        if $y_{\Pi(i)} o_{\Pi(i)} \leq 0$ **then**
          $\alpha_i \leftarrow \alpha_i + y_{\Pi(i)}$
          for $l \leftarrow 1, \ldots, m$ **do**
            $o_{\Pi(l)} \leftarrow o_{\Pi(l)} + y_{\Pi(i)} k\left(x_{\Pi(i)}, x_{\Pi(l)}\right)$
          end for
        end if
      end for
    until the if branch was never entered within the **for** loop
    $\boldsymbol{\alpha}_j \leftarrow \boldsymbol{\alpha}$
  end for

References

Aitchison, J. (1964). Bayesian tolerance regions (with discussion). *Journal of the Royal Statistical Society – Series B 26*, 161–175.

Aizerman, M. A., É.. M. Braverman, and L. I. Rozonoér (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control 25*, 821–837.

Allwein, E. L., R. E. Schapire, and Y. Singer (2000). Reducing multiclass to binary: a unifying approach for margin classifiers. In P. Langley (Ed.), *Proceedings of the International Conference on Machine Learning*, San Francisco, California, pp. 9–16. Morgan Kaufmann Publishers.

Alon, N., S. Ben-David, N. Cesa-Bianchi, and D. Haussler (1997). Scale–sensitive Dimensions, Uniform Convergence, and Learnability. *Journal of the ACM 44*(4), 615–631.

Anthony, M. and P. Bartlett (1999). *A Theory of Learning in Artificial Neural Networks*. Cambridge University Press.

Barabino, N., M. Pallavicini, A. Petrolini, M. Pontil, and A. Verri (1999). Support vector machines vs multi-layer perceptrons in particle identification. In M. Verleysen (Ed.), *Proceedings ESANN*, Brussels, pp. 257–262. D Facto.

Barber, D. and C. K. I. Williams (1997). Gaussian processes for Bayesian classification via hybrid Monte Carlo. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing Systems 9*, pp. 340–346. MIT Press.

Bartlett, P. and J. Shawe-Taylor (1998). Generalization performance of support vector machines and other pattern classifiers. In *Advances in Kernel Methods — Support Vector Learning*, pp. 43–54. MIT Press.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philiosophical Transactions of the Royal Socienty 53*, 370–418.

Berger, J. O. (1985). *Statistical Decision theory and Bayesian Analysis*. New York: Springer.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press.

Blake, C. and C. Merz (1998). UCI repository of machine learning databases.

Blumer, A., A. Ehrenfeucht, D. Haussler, and M. Warmuth (1989). Learnability and the Vapnik-Chervonenkis Dimension. *Journal of the ACM 36*(4), 929–965.

Bois, G. P. (1961). *Tables of Indefinite Integrals*. Dover Publications.

Boser, B. E., I. M. Guyon, and V. N. Vapnik (1992, July). A training algorithm for optimal margin classifiers. In D. Haussler (Ed.), *Proceedings of the Annual Conference on Computational Learning Theory*, Pittsburgh, PA, pp. 144–152. ACM Press.

Bousquet, O. and A. Elisseeff (2000). Stability and generalization. Technical report, Centre de Mathematiques Appliquees.

Bousquet, O. and A. Elisseeff (2001). Algorithmic stability and generalization performance. In T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems 13*, pp. 196–202. MIT Press.

Brown, M. P. S., W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, M. Ares, and D. Haussler (2000). Knowledge-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Sciences 97*(1), 262–267.

Burmeister, J. and J. Wiles (1994). The challenge of go as a domain for ai research: A comparison between go and chess. In *Proceedings of the 3rd Australian and New Zealand Conference on Intelligent Information Systems*.

Cornfeld, I., S. Fomin, and Y. Sinai (1982). *Ergodic Theory*. Springer.

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

Cox, R. (1946). Probability, frequency, and reasonable expectations. *American Journal of Physics 14*, 1–13.

Cristianini, N., C. Campbell, and J. Shawe-Taylor (1999). Dynamically adapting kernels in support vector machines. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems 11*, pp. 204–210. Cambridge, MA: MIT Press.

Cristianini, N. and J. Shawe-Taylor (2000). *An Introduction to Support Vector Machines*. Cambridge, UK: Cambridge University Press.

Cristianini, N., J. Shawe-Taylor, and H. Lodhi (2001). Latent semantic kernels. In *Proceedings of the International Conference on Machine Learning*, San Francisco. Morgan Kaufmann Publishers.

Debnath, L. and P. Mikusinski (1998). *Hilbert Spaces with Applications*. Academic Press.

Devroye, L. and L. Györfi (1985). *Nonparametric Density Estimation: The L_1 View*. John Wiley and Sons.

Devroye, L., L. Györfi, and G. Lugosi (1996). *A Probabilistic Theory of Pattern Recognition*. Number 31 in Applications of mathematics. New York: Springer.

Dietterich, T. G. and G. Bakiri (1995). Solving multi-class learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research 2*, 263–286.

Domingos, P. (1998). Occam's two razors: The sharp and the blunt. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 37–43.

Drucker, H., C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik (1997). Support vector regression machines. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing Systems 9*, Cambridge, MA, pp. 155–161. MIT Press.

Duda, R. O. and P. E. Hart (1973). *Pattern Classification and Scene Analysis*. New York: John Wiley and Sons.

Duda, R. O., P. E. Hart, and D. G. Stork (2001). *Pattern Classification*. New York: John Wiley and Sons. Second edition.

Ehrenfeucht, A., D. Haussler, M. Kearns, and L. Valiant (1989). A general lower bound on the number of examples needed for learning. *Information and Computation 82*, 247–261.

Feller, W. (1950). *An Introduction To Probability Theory and Its Application*, Volume 1. New York: John Wiley and Sons.

Feller, W. (1966). *An Introduction To Probability Theory and Its Application*, Volume 2. New York: John Wiley and Sons.

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

Fix, E. and J. Hodges (1951a). Discriminatory analysis. nonparametric discrimination: Consistency properties. Technical Report 21-49-004, USAF School of Aviation Medicine, Randolph Field, TX, U.S.

Fix, E. and J. Hodges (1951b). Discriminatory analysis: small sample performance. Technical Report 21-49-004, USAF School of Aviation Medicine, Randolph Field, TX, U.S.

Floyd, S. and M. Warmuth (1995). Sample compression, learnability, and the Vapnik Chervonenkis dimension. *Machine Learning 27*, 1–36.

Fotland, D. (1993). Knowledge representation in The Many Faces of Go.

Fraser, D. (1957). *Nonparametric Methods in Statistics*. New York: John Wiley and Sons.

Freund, Y. and R. E. Schapire (1999). Large margin classification using the perceptron algorithm. *Machine Learning 37*(3), 277–296.

Frieß, T.-T., N. Cristianini, and C. Campbell. (1998). The kernel adatron algorithm: A fast and simple learning procedure for support vector machines. In J. Shavlik (Ed.), *Proceedings of the International Conference on Machine Learning*, pp. 188–196. Morgan Kaufmann Publishers.

Gammerman, A., V. Vovk, and V. Vapnik (1998). Learning by transduction. In *Proceedings of Uncertainty in AI*, Madison, Wisconsin, pp. 148–155.

Geibel, P. and F. Wysotzki (1998). Learning relational concepts with decision trees. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pp. 1141–1144. Morgan Kaufmann Publishers.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin (1995). *Bayesian Data Analysis*. London: Chapman and Hall.

Gibbs, M. and D. J. C. Mackay (1998). Variational Gaussian process classifiers. Technical report, Cavendish Laboratory, Cambridge, UK.

Girolami, M. (2001). Mercer kernel based clustering in feature space. *IEEE Transactions on Neural Networks 13*(4), 780–784. To appear.

Goldfarb, L., J. Abela, V. C. Bhavsar, and V. N. Kamat (1995). Can a vector space based learning model discover inductive class generalization in a symbolic environment: A new approach to pattern recognition. *Pattern Recognition Letters 16*, 719–726.

Graepel, T., M. Goutrie, M. Krüger, and R. Herbrich (2001). Learning on graphs in the game of Go. In *Proceedings of the International Conference on Neural Networks*.

Graepel, T. and R. Herbrich (2001a). The kernel Gibbs sampler. In T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems 13*, Cambridge, MA, pp. 514–520. MIT Press.

Graepel, T. and R. Herbrich (2001b). A PAC-Bayesian margin distribution bound for kernel classifiers. presented at the NIPS workshop on "New directions in kernel-based learning methods".

Graepel, T., R. Herbrich, P. Bollmann-Sdorra, and K. Obermayer (1999). Classification on pairwise proximity data. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems 11*, Cambridge, MA, pp. 438–444. MIT Press.

Graepel, T., R. Herbrich, and K. Obermayer (1999). Bayesian transductive classification by maximizing volume in version space. In *Proceedings of Learning 1999 Conference*.

Graepel, T., R. Herbrich, and K. Obermayer (2000). Bayesian Transduction. In S. A. Solla, T. K. Leen, and K.-R. Müller (Eds.), *Advances in Neural Information Processing Systems 12*, Cambridge, MA, pp. 456–462. MIT Press.

Graepel, T., R. Herbrich, B. Schölkopf, A. J. Smola, P. Bartlett, K. Müller, K. Obermayer, and R. C. Williamson (1999). Classification on proximity data with LP-machines. In *Ninth International Conference on Artificial Neural Networks*, Conference Publications No. 470, London, pp. 304–309. IEE.

Graepel, T., R. Herbrich, and J. Shawe-Taylor (2000). Generalisation error bounds for sparse linear classifiers. In *Proceedings of the Annual Conference on Computational Learning Theory*, pp. 298–303.

Graepel, T. and K. Obermayer (1998). Fuzzy topographic kernel clustering. In W. Brauer (Ed.), *Proceedings of the 5th GI Workshop Fuzzy Neuro Systems '98*, pp. 90–97.

Guyon, I. and D. Storck (2000). Linear discriminant and support vector classifiers. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans (Eds.), *Advances in Large Margin Classifiers*, Cambridge, MA, pp. 147–169. MIT Press.

Guyon, I., J. Weston, S. Barnhill, and V. Vapnik (2002). Gene selection for cancer classification using support vector machines. *Machine Learning 46*(1-3), 389–422.

Haussler, D. (1999). Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99-10, Computer Science Department, University of California at Santa Cruz.

Haykin, S. (1994). *Neural Networks : A Comprehensive Foundation*. New York: Macmillan.

Heckerman, D. (1996). A tutorial on learning with Bayesian networks.

Hellman, M. (1970). The nearest neighbor classification rule with a reject option. *Transactions on Systems, Man and Cybernetics 2*, 179–185.

Herbrich, R. (2001). *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press.

Herbrich, R. and T. Graepel (2001a). Large scale Bayes point machines. In T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems 13*, Cambridge, MA, pp. 528–534. MIT Press.

Herbrich, R. and T. Graepel (2001b). A PAC-Bayesian margin bound for linear classifiers. submitted to IEEE Transactions on Information Theory.

Herbrich, R. and T. Graepel (2001c). A PAC-Bayesian margin bound for linear classifiers: Why SVMs work. In T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems 13*, Cambridge, MA, pp. 224–230. MIT Press.

Herbrich, R., T. Graepel, and C. Campbell (1999a). Bayes point machines: Estimating the Bayes point in kernel space. In *Proceedings of IJCAI Workshop Support Vector Machines*, pp. 23–27.

Herbrich, R., T. Graepel, and C. Campbell (1999b). Bayesian learning in reproducing kernel Hilbert spaces. Technical report, Technical University of Berlin. TR 99-11.

Herbrich, R., T. Graepel, and C. Campbell (2000). Robust Bayes point machines. In *Proceedings of ESANN 2000*, pp. 49–54.

Herbrich, R., T. Graepel, and C. Campbell (2001). Bayes point machines. *Journal of Machine Learning Research 1*, 245–279.

Herbrich, R., T. Graepel, and K. Obermayer (2000). Large margin rank boundaries for ordinal regression. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans (Eds.), *Advances in Large Margin Classifiers*, Cambridge, MA, pp. 115–132. MIT Press.

Herbrich, R., T. Graepel, and R. C. Williamson (2001). The structure of version space. submitted to Journal of Machine Learning Research.

Herbrich, R. and R. C. Williamson (2001). Algorithmic luckiness. submitted to NIPS*2001.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58*, 13–30.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics 6*, 65–70.

Jaakkola, T. S., M. Diekhans, and D. Haussler (1999). Using the fisher kernel method to detect remote protein homologies. In *Proccedings of the International Conference on Intelligence Systems for Molecular Biology*, pp. 149–158. AAAI Press.

Jaakkola, T. S. and D. Haussler (1999). Exploiting generative models in discriminative classifiers. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems 11*, Cambridge, MA, pp. 487–493. MIT Press.

Jain, A. K. and R. C. Dubes (1988). *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice Hall.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, Berlin, pp. 137–142. Springer.

Jolliffe, I. T. (1986). *Principal Component Analysis*. New York: Springer.

Kearns, M., Y. Mansour, A. Y. Ng, and D. Ron (1997). An experimental and theoretical comparison of model selection methods. *Machine Learning 27*, 7–50.

Kearns, M. J. and R. E. Schapire (1994). Efficient distribution–free learning of probabilistic concepts. *Journal of Computer and System Sciences 48*(3), 464–497.

Kearns, M. J. and U. V. Vazirani (1994). *An Introduction to Computational Learning Theory*. Cambridge, Massachusetts: MIT Press.

Kimeldorf, G. S. and G. Wahba (1970). A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics 41*, 495–502.

Kolmogorov, A. N. (1957). On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk SSSR 114*, 953–956.

König, H. (1986). *Eigenvalue Distribution of Compact Operators*. Basel: Birkhäuser.

Kuhn, H. W. and A. W. Tucker (1951). Nonlinear programming. In *Proc. 2^{nd} Berkeley Symposium on Mathematical Statistics and Probabilistics*, Berkeley, pp. 481–492. University of California Press.

LeCun, Y. (1998). MNIST handwritten digit database. Available as http://www.research.att.com/~yann/ocr/mnist/.

Lindsey, J. K. (1996). *Parametric Statistical Inference*. Clarendon Press.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-treshold algorithm. *Machine Learning 2*, 285–318.

Littlestone, N. and M. Warmuth (1986). Relating data compression and learnability. Technical report, University of California Santa Cruz.

Lodhi, H., J. Shawe-Taylor, N. Cristianini, and C. Watkins (2001). Text classification using kernels. In T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems 13*, Cambridge, MA, pp. 563–569. MIT Press.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology 58*, 1246–1266.

McAllester, D. A. (1998). Some PAC Bayesian theorems. In *Proceedings of the Annual Conference on Computational Learning Theory*, Madison, Wisconsin, pp. 230–234. ACM Press.

McAllester, D. A. (1999). PAC-Bayesian model averaging. In *Proceedings of the Annual Conference on Computational Learning Theory*, Santa Cruz, USA, pp. 164–170.

Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London A 209*, 415–446.

Merz, C. J. and P. M. Murphy (1998). UCI repository of machine learning databases. [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Michie, D., D. H. Spiegelhalter, and C. C. Taylor (1994). *Machine Learning, Neural and Statistical Classification*. Series in Artificial Intelligence. Ellis Horwood.

Mika, S., G. Rätsch, and K.-R. Müller (2001). A mathematical programming approach to the kernel fisher algorithm. In T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.), *Advances in Neural Information Processing Systems 13*, pp. 591–597. MIT Press.

Minka, T. (2001). *Expectation Propagation for approximative Bayesian inference*. Ph. D. thesis, MIT Media Labs, Cambridge, USA.

Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw-Hill.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical report, Dept. of Computer Science, University of Toronto. CRG-TR-93-1.

Neal, R. M. (1997a). Markov chain Monte Carlo method based on 'slicing' the density function. Technical report, Department of Statistics, University of Toronto. TR–9722.

Neal, R. M. (1997b). Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report Technical Report 9702, Dept. of Statistics.

Opper, M. and O. Winther (1999). Mean field methods for classification with Gaussian processes. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems 11*, Cambridge, MA, pp. 309–315. MIT Press.

Opper, M. and O. Winther (2000a). Gaussian processes and SVM: mean field and leave-one-out. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans (Eds.), *Advances in Large Margin Classifiers*, Cambridge, MA, pp. 311–326. MIT Press.

Opper, M. and O. Winther (2000b). Gaussian processes for classification: Mean field algorithms. *Neural Computation 12*(11), 2655–2684.

Platt, J. (2000). Probabilities for SV machines. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans (Eds.), *Advances in Large Margin Classifiers*, Cambridge, MA, pp. 61–73. MIT Press.

Platt, J. C. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research.

Poggio, T. and F. Girosi (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science 247*, 978–982.

Popper, K. R. (1980). *The Logic of Scientific Discovery*. London: Hutchinson. Originally published in 1934.

Popper, K. R. (1981). *Realism and the Aim of Science*. Totowa, New Jersey: Rowman and Littlefield.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press.

Robert, C. P. (1994). *The Bayesian choice: A decision theoretic motivation*. Ney York: Springer.

Rockafellar, R. T. (1970). *Convex Analysis*, Volume 28 of *Princeton Mathematics Series*. Princeton University Press.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review 65*(6), 386–408.

Rosenblatt, F. (1962). *Principles of neurodynamics: Perceptron and Theory of Brain Mechanisms*. Washington D.C.: Spartan–Books.

Ruján, P. (1997). Playing billiards in version space. *Neural Computation 9*, 99–122.

Ruján, P. and M. Marchand (2000). Computing the Bayes kernel classifier. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans (Eds.), *Advances in Large Margin Classifiers*, Cambridge, MA, pp. 329–347. MIT Press.

Rychetsky, M., J. Shawe-Taylor, and M. Glesner (2000). Direct Bayes point machines. In *Proceedings of the International Conference on Machine Learning*.

Saitoh, S. (1988). *Theory of Reproducing Kernels and its Applications*. Harlow, England: Longman Scientific & Technical.

Sasaki, N. and Y. Sawada (1998). Neural networks for tsume-go problems. In *Proceedings of the Fifth International Conference on Neural Information Processing*, pp. 1141–1144.

Schapire, R., Y. Freund, P. L. Bartlett, and W. S. Lee (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics 26*, 1651–1686.

Schapire, R. E., Y. Freund, P. Bartlett, and W. S. Lee (1997). Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the International Conference on Machine Learning*.

Schölkopf, B. (2000). The kernel trick for distances. TR MSR 2000 - 51, Microsoft Research, Redmond, WA. Published in: T. K. Leen, T. G. Dietterich and V. Tresp (eds.), *Advances in Neural Information Processing Systems 13*, MIT Press, 2001.

Schölkopf, B., R. Herbrich, and A. J. Smola (2001). A generalized representer theorem. In *Proceedings of the Annual Conference on Computational Learning Theory*.

Schölkopf, B., S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola (1999). Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks 10*(5), 1000–1017.

Schölkopf, B., J. Shawe-Taylor, A. J. Smola, and R. C. Williamson (1999). Generalization bounds via eigenvalues of the Gram matrix. Technical Report 99-035, NeuroCOLT. Available from http://www.kernel-machines.org.

Schölkopf, B., P. Simard, A. Smola, and V. Vapnik (1998). Prior knowledge in support vector kernels. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems 10*, Cambridge, MA, pp. 640–646. MIT Press.

Schölkopf, B., A. Smola, and K.-R. Müller (1996). Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max-Planck-Institut für biologische Kybernetik.

Schölkopf, B., A. Smola, and K.-R. Müller (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation 10*, 1299–1319.

Schölkopf, B., A. Smola, R. C. Williamson, and P. L. Bartlett (2000). New support vector algorithms. *Neural Computation 12*, 1207–1245.

Schölkopf, B., R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt (2000). Support vector method for novelty detection. In S. A. Solla, T. K. Leen, and K.-R. Müller (Eds.), *Advances in Neural Information Processing Systems 12*, Cambridge, MA, pp. 582–588. MIT Press.

Schraudolph, N. N., P. Dayan, and T. J. Sejnowski (1994). Temporal difference learning of position evaluation in the game of go. In J. D. Cowan, G. Tesauro, and J. Al-

spector (Eds.), *Advances in Neural Information Processing Systems*, Volume 6, pp. 817–824. Morgan Kaufmann Publishers, Inc.

Shawe-Taylor, J. (1996). Confidence estimates of classification accuracy on new examples. Technical report, Royal Holloway, University of London. NC2–TR–1996–054.

Shawe-Taylor, J., P. L. Bartlett, R. C. Williamson, and M. Anthony (1998). Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory 44*(5), 1926–1940.

Shawe-Taylor, J. and N. Cristianini (1998). Margin distribution bounds on generalization. NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK.

Shawe-Taylor, J. and N. Cristianini (2000a). Margin distribution and soft margin. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans (Eds.), *Advances in Large Margin Classifiers*, Cambridge, MA, pp. 349–358. MIT Press.

Shawe-Taylor, J. and N. Cristianini (2000b). On the generalisation of soft margin algorithms. *IEEE Transactions on Information Theory*. Submitted.

Shawe-Taylor, J. and R. C. Williamson (1997). A PAC analysis of a Bayesian estimator. Technical report, Royal Holloway, University of London. NC2–TR–1997–013.

Silverman, B. W. (1986). *Density estimation for statistical and data analysis*. Monographs on statistics and applied probability. London: Chapman and Hall.

Smola, A., T. Frieß, and B. Schölkopf (1999). Semiparametric support vector and linear programming machines. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems 11*, Cambridge, MA, pp. 585–591. MIT Press.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Tipping, M. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research 1*, 211–244.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM 27*(11), 1134–1142.

Vanderbei, R. J. (1994). LOQO: An interior point code for quadratic programming. TR SOR-94-15, Statistics and Operations Research, Princeton Univ., NJ.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. New York: Springer.

Vapnik, V. (1998). *Statistical Learning Theory*. New York: John Wiley and Sons.

Vapnik, V. and A. Chervonenkis (1974). *Theory of Pattern Recognition [in Russian]*. Moscow: Nauka. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979).

Vapnik, V. N. (1982). *Estimation of Dependences Based on Empirical Data*. Berlin: Springer.

Vapnik, V. N. and A. Y. Chervonenkis (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications 16*(2), 264–281.

Vidyasagar, M. (1997). *A Theory of Learning and Generalization*. New York: Springer.

Wahba, G. (1990). *Spline Models for Observational Data*, Volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Philadelphia: SIAM.

Watkin, T. (1993). Optimal learning with a neural network. *Europhysics Letters 21*, 871.

Watkins, C. (2000). Dynamic alignment kernels. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans (Eds.), *Advances in Large Margin Classifiers*, Cambridge, MA, pp. 39–50. MIT Press.

Webb, G. I. (1996). Further experimental evidence against the utility of occam's razor. *Journal of Artificial Intelligence Research 4*, 397–417.

Weston, J. and C. Watkins (1999). Multi-class support vector machines. In M. Verleysen (Ed.), *Proceedings ESANN*, Brussels. D Facto.

Williams, C. K. I. and C. E. Rasmussen (1996). Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8*, Cambridge, MA, pp. 514–520. MIT Press.

Wolf, T. (1994). The program GoTools and its computer-generated Tsume Go database. In *Proceedings of the 1st Game Programming Workshop*.

Wu, D., K. P. Bennett, N. Cristianini, and J. Shawe-Taylor (1999). Large margin trees for induction and transduction. In *Proceedings of the International Conference on Machine Learning*.

Zadeh, L. (1992). *An Introduction to fuzzy logic applications in intelligent systems*. Boston: Kluwer Academic.