Sample Compression, Margins and Generalization: Extensions to the Set Covering Machine

Mohak Shah

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the degree of Doctor of Philosophy in Computer Science

School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

For my parents Upendra and Rakshika Shah
and my dear sister Tamanna.

Abstract

This thesis studies the generalization behavior of algorithms in Sample Compression Settings. It extends the study of the Sample Compression framework to derive data-dependent bounds that give tighter guarantees to the algorithms where data-independent bounds such as the VC bounds are not applicable. It also studies the interplay between sparsity and the separating margin of the classifier and shows how new compression based data-dependent bounds can be obtained that can exploit these two quantities explicitly. These bounds not only provide tight generalization guarantees but by themselves present optimization problems for learning leading to novel learning algorithms.

This thesis studies the algorithms based on learning conjunctions or disjunctions of data-dependent boolean features. With the Set Covering Machine (SCM) as its basis, the thesis shows how novel learning algorithms can be designed in compression settings that can perform a non-trivial margin-sparsity trade-off to yield better classifiers. Moreover, the thesis also shows how feature-selection can be integrated with the learning process in these settings yielding algorithms that not only perform successful feature selection but also have provable theoretical guarantees.

In particular, the thesis proposes two novel learning algorithms. The first algorithm is for the SCM with data-dependent half-spaces along with a tight compression bound that can successfully perform model selection. The second algorithm aims at learning conjunctions of features called data-dependent Rays to classify gene expression data from DNA micro-arrays. The thesis shows how a PAC-Bayes approach to learning Rays' conjunctions can perform a non-trivial margin-sparsity trade-off to achieve classifiers that not only have provable theoretical guarantees but also utilize a significantly small number of attributes unlike traditional feature selection algorithms.

This thesis also proposes two new formulations for the classical SCM algorithm with data-dependent balls aimed at performing margin-sparsity trade-off by utilizing Occam's Razor and PAC-Bayes principles respectively. The thesis shows how such approaches yield more general classifiers with tight risk bounds that can potentially guide the model selection process.

CONTENTS

Acknowledgements

I would like to express my gratitude to my supervisors Mario Marchand and Marcel Turcotte for their able guidance and support throughout my Ph.D. studies. Mario Marchand was my reason and inspiration for coming to Ottawa. I have gained immensely from his expertise both during my first year as a Masters student and subsequent years during the Ph.D. I would like to thank him for his supervision, teaching, patience and understanding. I have indeed been fortunate for having him as a supervisor.

A special thank also goes to Marcel Turcotte for not only guiding me and sharing his Bioinformatics expertise but also for his generosity, patience, trust, availability and discussions both on research and personal fronts. I would indeed treasure our meetings and discussions we had in his office and hallways. I would also like to thank John Shawe-Taylor and François Laviolette for extending their Mathematics' expertise.

I would like to thank Marina Sokolova for being such a nice colleague and research collaborator throughout my stay in Ottawa.

I would also like to thank the members of my thesis defence committee who gave useful suggestions on both technical as well as stylistic aspects of this thesis. A special thank goes to Dale Schuurmans for a meticulous review of my thesis that helped improve its quality significantly. My thank also goes to Nathalie Japkowicz and John Oommen for their kind support and consideration whether it was my comprehensive examination, candidacy paper, teaching, projects or advices on the career front. I would also like to thank Herna Viktor for useful comments on my thesis. I am also thankful to Stan Szpakowicz for introducing me to the field of Natural Language Processing and sharing his expertise in my research endeavors. I am grateful to Suzanne St-Michel for making things easier for me on the administrative front and to Michel Racine and Marc Fortier for being really good friends and their willingness to help.

My heartfelt thank goes to my friends for their loyalty, trust and support, especially during tough times. Ruma, Sushil, Rajeet, Sumit and Arvind, thank you all for being there for me. I would especially like to thank Ruma for being a colleague, friend and someone I could count on during these hard PhD years.

Last but not the least, I am greatly indebted to my parents Upendra and Rakshika Shah and my sister Tamanna for their unconditional love and for believing in me throughout. I could not have done this without you.

Abbreviations

ALL:	Acute Lymphoblastic Leukaemia
AML:	Acute Myeloid Leukaemia
BuildSCM:	The Set covering machine generic algorithm
CNF:	Conjunctive Normal Form
COLT:	COmputational Learning Theory
CV:	Cross Validation
DNA:	Deoxyribonucleic acid
DNF:	Disjunctive Normal Form
ERM:	Empirical Risk Minimization
i.i.d.:	independent identically distributed
KL-divergence:	Kullback-Leibler divergence
MDL:	Minimum Description Length
NP:	Nondeterministic-Polynomial
PAC:	Probably Approximately Correct
RBF:	Radial Basis Function
RP:	Random probabilistic polynomial (time)
SCM:	Set Covering Machine
SRM:	Structural Risk Minimization
SVM:	Support Vector Machine
SV:	Support Vector
UCI:	University of California at Irvine
VC-dimension:	Vapnik-Chervonenkis dimension

LIST OF TABLES

CHAPTER 1

Introduction

Recent developments in machine learning have made it possible to obtain insights from vast amounts of data. This has resulted in algorithms that can learn and generalize well across a wide spectrum of tasks such as face recognition, speech and handwriting recognition, games, medical diagnosis and prognosis and machine translation. One of the main areas of research in machine learning has been the problem of classification where an algorithm is expected to learn so as to be able to distinguish between instances of two (or more) classes.

Learning approaches vary in the learning biases that they exploit to achieve good generalization capabilities. With respect to the classification problem, linear classifiers have emerged as important learning algorithms. These have become quite popular because of the ease of analysis and implementation. The Support Vector Machine (SVM) has appeared as one of the most prominent result from this line of research. The SVM was originally proposed by Boser, Guyon and Vapnik (1992). An SVM relies on obtaining a separating hyperplane (a linear classifier) in the feature space and it has been shown that one can expect this classifier to generalize well when the separating margin around the decision boundary of this hyperplane is maximized. However, other approaches have also been developed in parallel that focus on alternative biases for learning. One such approach has appeared in the form of algorithms aiming to represent the hypothesis (classifier) in terms of minimum possible number of training examples. The algorithms utilizing this bias of relying on a substantially small subset of training examples to represent the hypothesis and being able to reconstruct this hypothesis with the help of just these examples fall under the category of Sample Compression based algorithms. Although the notion of Sample Compression is quite old (Littlestone and Warmuth, 1986), recent developments in the field have resulted in promising algorithms that are more general and have a wider domain of application. The basic idea behind the Sample compression algorithms is to obtain learning algorithms with the property that the generated classifier (with respect to some training data) can often be reconstructed with a very small subset of training examples. Such classifiers are typically called sparse classifiers.

In a sense an SVM can be considered as sample compression algorithm: The decision

boundary of the SVM is represented in the form of an additive model. That is, if h is the decision function output by the SVM, then the output of h on some example x is given as: $h(x) = \sum_{i=1,\ldots,m;x_i \in S} \alpha_i k(x, x_i)$ where $k(.,.)$ is known as the kernel function and m the number of examples in the training set. The solution hyperplane is expected to have only a few non-zero α_i's. Hence, the solution hyperplane is essentially represented in terms of a few training examples. Moreover, running the same learning algorithm only on this subset of training examples for which the $\alpha_i \neq 0$ (instead of the whole training set) will still output the same hyperplane decision boundary.

However, the inherent limitation of this approach in exploiting the sparsity of the solution comes from the fact that the optimization for obtaining the final hyperplane is based entirely on maximizing the separating margin around it. Some attempts have been made to obtain sparse classifiers from SVM. For instance Bennett (1999) and Bi et. al. (2003) propose to minimize an ℓ_1-norm functional (instead of the traditional ℓ_2-norm) and have found that, indeed, the sparser SVM sometimes had better generalization. Therefore, from this SVM perspective, we should consider algorithms that minimizes an ℓ_β-norm functional for any $\beta \in [0, 2]$. In the $\beta = 2$ limit, we obtain the SVM with the largest possible separating margin (without considering its sparsity). In the $\beta = 0$ limit, we would obtain the sparsest SVM (without considering the magnitude of its separating margin). This parameter β would then control the margin-sparsity trade-off of the final classifier.

On the other hand, algorithms have been proposed that focus solely on the sparsity of solution and also have practical guarantees on the future performance. For instance, consider an initial approach in the form of the Standard monomial learning algorithm proposed by Valiant (1984) that aimed at learning a small conjunction or disjunction of monomials. Haussler (1988) further reduced it to the minimum set cover problem and proposed a greedy heuristic for this algorithm with a provably good worst case lower bound on the number of monomials required (Chvátal, 1979). However, not until recently, was this approach generalized for practical learning tasks when Marchand and Shawe-Taylor (2001, 2002) proposed the Set Covering Machine learning algorithm that builds data-dependent boolean features on data and then learns a (preferably) small conjunction or disjunction of these features to obtain a classifier. This approach used data-dependent balls as features and showed good preliminary results. There was one more striking feature of this approach: *pragmatic risk bounds*. The generalization bounds that can be obtained over the future performance of the classifier are not only considerably tight but are also practical enough to enable the learning

algorithm to perform model selection[1].

The motivation for this thesis comes from the above initial results. The encouraging results obtained by the simple bias that the SCM utilizes certainly warranted further research to investigate the possibility of extending this framework. This thesis aims at examining whether extending this bias of learning conjunctions or disjunctions of (possibly data-dependent) features can lead to better classifiers with provable guarantees. This thesis also examines the role of the margin of the decision surface of the classifier and its affect on the algorithm's compression abilities.

To summarize, the primary aim of this thesis is:
Investigating the generality of the sample compression algorithms by extending the SCM framework and studying the interplay between the margin of the decision surface and sparsity of the solution (and consequently a possible trade-off between the two) so as to obtain classifiers with better performance and guarantees.

We adopt the basic set covering machine framework as the basis for our learning algorithms. The primary reason for adopting SCM as basis is that it is an elegant generalization of algorithm for learning conjunctions or disjunctions of boolean features. Moreover, the SCM framework *is modular*, *robust*[2] and *stable*[3]. Consequently, our approaches automatically inherit these properties.

1.1 Contributions

In the light of the above discussion, the main contributions of this thesis can be seen under the following broad categories.

i. ***Novel learning algorithms*** that utilize the bias of learning conjunctions or disjunctions of data-dependent features. In this respect, we propose:

   (a) *Two new sample compression based learning algorithms.*
   The first is a learning algorithm for Set Covering Machines with data-dependent Half-Spaces. This algorithm not only demonstrates the generality of the approach by showing that the basic SCM algorithm is not limited to a certain set of features, but also shows that indeed alternate feature sets might be required in certain sce-

[1] Model selection refers to choosing the best classifier from among the potential ones. We discuss model selection in Chapter 2.

[2] *Robustness* refers to the ability of handle noisy data.

[3] *Stability* refers to reliable performance on unseen examples both empirically and theoretically.

narios for better performance. Half-spaces are one such set of features that in general yield sparser classifiers. The second novel learning algorithm that we propose comes in the form of learning conjunctions of linear threshold features called *Rays* built on individual attributes (unlike the features built on examples in traditional SCM). Exploiting sparsity in terms of features, as opposed to examples, in this case gives us a feature selection algorithms with quite competitive practical performance as well as, and probably more importantly, provable theoretical guarantees. We present an application of this feature selection algorithm to the gene expression data obtained from DNA microarrays to which probably the approach is probably best suited.

(b) *Two alternate formulations for the SCM algorithm.*
These resulted from our attempts to exploit the margin of the decision boundary in addition to the sparsity to obtain better classifiers. Two approaches, one motivated from the Occam's Razor principle and another based on the PAC-Bayesian principle appeared as a result.

ii. ***Tight Generalization bounds*:** The motivation behind our algorithms came from the ability to obtain tight and practical risk bounds. The most important results coming from this research are in the form of generalization error bounds that explicitly depend on *both* the sparsity of a classifier *and* the magnitude of its separating margin. We show how both sparsity and margin can be considered as different forms of data compression and exploited by performing a trade-off to yield more general classifiers (see Chapter 7, 8 and 9 in particular).

1.2 Thesis Organization

The rest of this thesis is organized as below[4]:

Chapter 2 provides an overview of machine learning.

Chapter 3 provides an introduction to the Statistical Learning Theory that forms the theoretical basis of various machine learning approaches. We study this formal framework in three parts viz. main algorithmic approaches, mathematical learning models and generalization risk bounds. Note that we focus on uniform risk bounds that provide guarantees on the generalization performance of the classifier in terms of studying the uniform convergence

[4]For the readers who have sufficient background, the thesis is organized in such a way that each chapter can be read in isolation.

of the empirical risk to true risk of all the classifiers. The various bounds presented in this chapter form the basis of the bounds that we present in subsequent chapters.

Chapter 4 gives a brief description of the general Set Covering Machine algorithm of Marchand and Shawe-Taylor (2001). We also give a formal outline of the algorithm that we will be referring to in further chapters. Along with this we describe the set of features called *data-dependent balls* introduced by Marchand and Shawe-Taylor (2001) and derive a tight sample compression bound for SCM with balls.

Chapter 5 presents our first result in the form of a new learning algorithm for Set Covering Machines with data-dependent Half-Spaces along with a sample compression bound over its generalization performance. We show empirically that exploiting alternate sets of features is not only possible but also necessary for better performance in some cases. Moreover, the proposed bound performs model selection successfully.

Chapter 6 proposes algorithms for learning conjunctions (or disjunctions) of simple threshold features called *Rays* based on Sample Compression and Occam principles. These algorithms implicitly aim at performing feature selection to find a minimum number of Rays to classify gene expression data. However, the inherent property of the algorithms to focus on sparse solutions have a limiting effect as we discuss there. This limitation is addressed with a new learning algorithm based on PAC-Bayes settings that allows us to trade-off sparsity to some extent in favor of large separating margins. We present this approach in ***Chapter 7*** along with a PAC-Bayes risk bound.

Chapter 7 also forms the groundwork and motivation for a margin-sparsity based study of the SCM framework. Consequently, the chapters that follow present two alternate learning algorithms for the SCM that perform a non-trivial margin-sparsity trade-off.

Chapter 8 proposes an alternative algorithm for the set covering machine that uses a code for message strings to represent the radii of data-dependent balls. The algorithm trades off sparsity in favor of a short code for the radii. We also present a tight risk bound and show that it can successfully perform model selection.

In ***Chapter 9*** we propose a PAC-Bayes approach to the set covering machine in an effort to address a limitation of previous approach, that of, choosing an *a priori* scale over the radii of data-dependent balls. This algorithm is also, as before, accompanied with a tight risk bound.

Finally, ***Chapter 10*** summarizes the main results of our work and provides an insight to future directions resulting from it.

CHAPTER 2

Machine Learning Overview

This Chapter presents a brief overview of Machine Learning concepts such as the notion of learning (esp. inductive inference) and the corresponding issues, types of learning problems and empirical evaluation methods along with their use in model selection.

2.1 Introduction

One of the most interesting, exciting and challenging avenues since the advent of the computer has been the notion of *learning*. It has always interested mankind whether computers can be put to tasks that require them to improve their performance with increasing experience. The efforts in this direction have opened up many a new avenues of research. Some of the initial formalizations of the definition of learning have been observed in the following form:

Definition 1. *(Simon, 1983)* **Learning** *corresponds to the changes in system that...enables [it] to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.*

Note that *learning* should not be confused with *memorizing* or *adaptation*. Memorizing corresponds to the ability of merely remembering and recalling the output (or a course of action) when a similar instance or task arrives. Adaptation refers to a temporary change in condition that can be altered in the event of future changes. On the other hand, learning has the ability to *generalize*. That is, learning can look beyond the available scenarios by exploiting the regularities in the present situations. Hence, in a sense, learning signifies intelligence. The existence of regularities in many a natural domains make learning feasible.

Learning, hence is a very wide notion. Efforts to automate learning have largely focussed on automating the process of *Inductive inference*. Inductive inference basically refers to observing a phenomenon and generalizing over it. Essentially, this is done by modeling this phenomenon and then using this model to make predictions over future phenomena.

Machine Learning aims at the practical aspects of automating this process of inductive inference. That is, here we refer the computer program as the learning entity. However,

alongside has evolved a parallel stream known as Statistical Learning Theory (also known as Computational Learning Theory). Statistical Learning Theory attempts to formalize this process so as to provide a theoretical framework for Machine learning algorithms. In this chapter, we will discuss in detail, the notion and the essential aspects and components of the Machine learning framework and the types of problems that it addresses. The next chapter provides an introduction to the Statistical Learning Theory.

Mitchell (1997) gives the following definition of *Machine*-Learning:

Definition 2. *(Mitchell, 1997) A computer program is said to* **learn** *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

In fact T can be thought of as a distribution over the space of possible tasks so that with increasing experience, on any future task drawn independently according to T, the performance evaluated according to P improves.

This improvement in the system can be brought up either by providing the system with additional information and skills or by helping the system to adapt its behavior according to the existing task. The mode in which the improvement is made helps address varying kinds of learning problems, for instance, the *classification* problem where the learning system learns from more and more examples (i.e. improvement is made by providing additional information) and the *regression* problem where the task is to find a functional that fits the given data (i.e. adapt to the existing task). We will see the broad classes of learning problems further in the chapter.

As can be seen in the definition above, we need to define and formalize three quantities for any system:

1. the learning task T; the ultimate problem on which we expect the system to perform well.

2. a performance measure P on T; method of evaluating how well the system performs on the task.

3. the mode of experience E; the formalization which would be used to provide the additional information or skills to the system so as to enable it to gain experience and hence improve from it.

These three quantities vary according to the problem and domain of application giving rise to various models of learning. For instance, the additional information can be provided

to the learner in the form of vectors leading to: *supervised learning* when the vectors are labeled with the output category and *unsupervised learning* otherwise. The availability of the labels for input examples also dictates the goals of these two learning methods. The first aims to obtain a model that can provide an output based on the observed inputs. The latter on the other hand tries to model the inputs themselves. Providing additional information in terms of examples is probably the most widely practised methodology. However, other approaches exist such as providing additional information via relations, constraints, functions and even models. Moreover, there have recently been attempts to learn from examples with and without labels together, an approach largely known as semi-supervised learning (see Zhu (2005) for a survey).

The manner in which the additional information is provided also plays an important role in learning giving rise to two main models of learning: Active learning, where an algorithm tries to learn using a minimal number of labeled examples, and; Passive learning, where an algorithm is given a fixed set of examples. It should be noted that Active learning is different from Online learning where a master algorithm uses the prediction of competing hypotheses to predict the label of a new example and then learns from its actual label.

2.2 Why is Learning Difficult?

It is obvious that learning systems can be instrumental in our understanding of many complex phenomena, discovery of knowledge, information and patterns in databases as well as developing systems that can not only understand a domain but also adapt themselves so as to perform better in the future. However, such learning is not always easy since a system is expected to generalize its limited experience from finite amount of data to an (often) infinite domain. There can be more than one possible generalization. The criterion used to favor one generalization over another is called the *bias* of the learning algorithm. See work of Gordon and Desjardins (1995) for more details on learning biases.

A simple example can give an idea of the level of difficulty of the learning problem and some of the issues involved in learning. Consider the problem of fitting a functional to a set of data points in 2-dimensional plane, known as the *regression problem* (that most closely approximates the data point and can also generalize well in the future) as shown in figure 2.1.

There are always two extremes. The first is a curve that passes through each and every given point (the blue curve in the figure) and the second is a line that approximates the

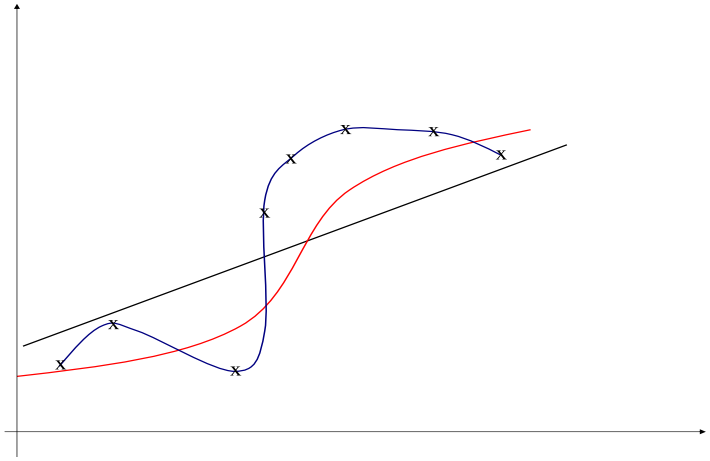

Figure 2.1: A simple regression problem

data closely (the black line in the figure). The first extreme is the most specific extreme while the second can be considered to be the most general one. In between these two extremes, there are infinite possibilities (e.g. the red curve in the figure) and several issues to consider. For instance, the choice of the first option, fitting a curve passing through every data point, might often lead to what is called *overfitting* (making the solution too specific to generalize well in the future). On the other hand, the other extreme, approximation of a line, might be a misleading approximation if the data is sparse. Many such issues exist and efforts have been made to address them. For instance, the issue of overfitting is generally addressed using approaches such as Pruning or Boosting. Pruning is defined as generalizing the final solution by removing some very specific information, for example, by removing nodes in the case of decision trees. Boosting, on the other hand, aims at finding some general rules of thumb instead of a very accurate approximation of the data and use these rules repeatedly. The underlying idea is to combine "weak" learners[1] to form an ensemble so that the performance of a single member is improved. Approaches such as *Regularization* and Bayesian formulations have also shown promise in tackling this problem. We discuss these approaches in the next Chapter. In case of sparse datasets we use some kind of smoothing or back-off strategy. We do not discuss these issues in detail here. However, we make our point about the difficulty of the learning problem. For details on these issues, please refer to the pointers in Section 2.6.

A learning framework has many components such as the instance space, hypothesis (an estimate of the function to be learned), hypothesis space, performance methods, learning algorithm and *learning bias*. Different approaches to the representation, organization and

[1]that performs slightly better than random guessing

extensions to these components have lead to various types of learning. Some of the prominent types of learning include Decision Tree Learning, Artificial Neural Networks, Bayesian Learning, Instance/Case based learning, Genetic Algorithms, Rule Induction, Analytical learning and Reinforcement learning. For details on these approaches please refer to the literature cited in Section 2.6.

Throughout this work, we will be referring to Supervised learning from examples with two possible classes, i.e. the binary classification framework. Moreover, we will focus on investigating the Sample Compression based algorithms. This class of algorithms includes the ones that use a (preferably small) subset of training examples to represent the hypothesis. We describe this framework in Section 3.7. In particular, our approaches assumes as its basis the **S**et **C**overing **M**achine algorithm proposed by Marchand and Shawe-Taylor (2001, 2002). We describe the basic Set Covering Machine (SCM) algorithm in Chapter 4.

2.3 Formalization of a Learning Problem

Let us now formalize the general settings of a learning problem. The general model of learning can be described using the following three components:

1. An instance space $\mathcal{X}$ from which random vectors $\mathbf{x} \in \mathbb{R}^n$ can be drawn independently according to some fixed but unknown distribution.

2. A label $y \in \mathcal{Y}$ for every vector $\mathbf{x}$ according to some fixed but unknown conditional distribution.

3. A learning algorithm A that can implement a set of functions f from some function class $\mathcal{F}$ over the instance space. Each such function $f \in \mathcal{F}$ is known as a classifier.

The problem of learning is that of choosing the best classifier from the given set of functions that can most closely approximate the labels of the vectors. This classifier is selected based on a training set S of m training examples drawn according to $\mathcal{X}$ with their respective labels. Each tuple of a vector $\mathbf{x}$ and its label y can be represented by $\mathbf{z} = (\mathbf{x}, y)$ which can be assumed to be drawn independently from a joint distribution D.

The choice of the best classifier is often based on the measure of *risk* which is nothing but the degree of disagreement between the label y of a vector $\mathbf{x}$ and the one assigned by the classifier $f : \mathcal{X} \to \mathcal{Y}$ that we denote by $f(\mathbf{x})$. Before defining the *risk* of a classifier, let us define the *loss function*. A loss function is simply a quantitative measure of the loss when the label y of the vector $\mathbf{x}$ is different from the label assigned by the classifier. We denote

the generic loss function by $L(y, f(\mathbf{x}))$ that outputs the loss incurred when y differs from $f(\mathbf{x})$. We can now define "the *risk* of the classifier f" as:

$$R(f) = \int L(y, f(\mathbf{x}))dD(\mathbf{x}, y) \tag{2.1}$$

where the probability measure $D(\mathbf{z}) = D(\mathbf{x}, y)$ is unknown. This risk is often referred to as the *true risk* of the classifier f. For the zero-one loss, i.e. $L(y, f(\mathbf{x})) = 1$ when $y \neq f(\mathbf{x})$ and 0 otherwise, we can write the *expected risk* as:

$$R(f) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D}\left(f(\mathbf{x}) \neq y\right) \tag{2.2}$$

The *generalization error* is a measure of the deviation of the expected risk of the classifier $f = A(S)$ learned from the overall minimum expected risk. Hence, the generalization error of algorithm A over a sample S is:

$$R(A, S) \stackrel{\text{def}}{=} R(f) - \inf_{f' \in \mathcal{F}} R(f')$$

In the most common learning principle called the *Empirical Risk Minimization*, the expected risk takes the form of a measurable quantity known as the empirical risk. It can be shown that the empirical risk converges exponentially to the true risk with respect to some sample size m (see (Herbrich, 2002) subsection A.5.2 for details) Also, this convergence rate depends on the true risk and the sample size. We show this in Section 2.5.1.

Given a training set $S = (\mathbf{z}_1, \ldots, \mathbf{z}_m)$ of m examples, the task of a learning algorithm is to construct a classifier with the smallest possible risk without any information about D. However, computing the true risk of a classifier as given above, can be quite difficult. Hence, the learner often computes the *empirical risk* $R_S(f)$ of any given classifier f according to:

$$R_S(f) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} L(y_i, f(\mathbf{x}_i)) \tag{2.3}$$

which is the risk of the classifier with respect to the training data. Here, $L(y, f(\mathbf{x}))$ is the specific risk function that outputs the loss of mislabeling an example. Note that this function can be a binary function (outputting only 1 or 0), or a continuous function depending upon the class of problems as discussed in section 2.4.

2.4 Classes of Learning Problems

Learning problems can be formulated in many ways. However, we discuss three main classes of learning problems along the lines of Vapnik (1995):

2.4.1 Classification

This problem is also known as the *Pattern Recognition* problem. We consider the two-class classification problem. Note that multi-class classification is a general case of two-class classification. Any multi-class classification problem can be broken down into multiple two-class classification problems.

In a two-class classification problem the label y is either 0 or 1, i.e. $y \in \{0, 1\}$. In this case the classifier f outputs only 0 or 1. Hence, we can consider this classifier as a set of *Indicator* functions. The classifier incurs a loss of 0 whenever the output of the classifier matches the true label y of the instance vector $\mathbf{x}$ and 1 otherwise. The later condition is known as *classification error* or *misclassification*.

Hence, the classification problem is to minimize the probability of this misclassification error over the set S of training examples while making a promising case of good generalization. The true error in this case can be represented as:

$$R(f) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D}\left(f(\mathbf{x}) \neq y\right) = \mathbf{E}_{(\mathbf{x},y)\sim D} I(f(\mathbf{x}) \neq y)$$

where $I(a) = 1$ if predicate a is true and 0 otherwise. Similarly, the empirical risk $R_S(f)$ can be shown to be:

$$R_S(f) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} I(f(\mathbf{x}_i) \neq y_i) \stackrel{\text{def}}{=} \mathbf{E}_{(\mathbf{x},y)\sim S} I(f(\mathbf{x}) \neq y)$$

Note that the loss function $L(\cdot, \cdot)$ in Equation 2.3 is replaced by indicator function for the classification problem.

2.4.2 Regression Estimation

Let the label y of every instance vector be such that $y \in \mathbb{R}$. Let $f(\mathbf{x})$ be the set of real functions containing:

$$f'(\mathbf{x}) = \int y dD(y|\mathbf{x})$$

Note that $f'(\cdot)$ is the target *regression function*. It has been shown that the regression function is the one that minimizes the squared loss. Hence the loss function is:

$$L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2 \tag{2.4}$$

The regression problem is the one of minimizing the true risk of Equation 2.1 while minimizing the loss of Equation 2.4.

2.4.3 Density Estimation

The problem of Density estimation is the one of estimating densities from a set of densities $p(\mathbf{x})$. The loss function to minimize in this case is:

$$L(p(\mathbf{x})) = -\log p(\mathbf{x}) \tag{2.5}$$

Hence the problem is to estimate the density while minimizing the true risk of Equation 2.1 with loss function of Equation 2.5.

2.5 Empirical Evaluation

So we have a learning algorithm and we have obtained a hypothesis as the output after training this learning algorithm on training data. The questions that arise now are: How good is this hypothesis? In fact, if there are competing hypotheses, how do we select the best one? And finally, how will this hypothesis perform on future unseen instances? i.e. how will this hypothesis generalize? Evaluation methods aim at answering these questions that essentially can be broken down into three broad categories:

(i) Testing: Measuring the "goodness" of hypothesis (ideally on unseen data).

(ii) Model Selection: Selecting the best hypothesis from among the potential hypotheses.

(iii) Generalization: Predicting quantitatively the future performance of hypothesis based on its performance in training and/or testing.

In order to evaluate the classifier's performance over a domain, we would ideally like to have a measure on how this classifier would perform on unseen instances. Note that these unseen instances are assumed to come from the same distribution as the training instances for the learning algorithm. One approach of obtaining such quantitative measure is via a generalization risk bound over the error of chosen hypothesis. Such risk bounds provide guarantees over the true risk of the classifier in terms of its performance on training (or test) data. See (Langford, 2005) for more details. We will detail such bounds in Chapter 3, focusing in particular, on uniform risk bounds obtained with respect to the hypothesis' performance on the training set. Here we discuss two prominent methods that enable us to evaluate the hypothesis by testing it on unseen instances. The amount of the data available generally governs the choice of an apt method for evaluation in various scenarios.

2.5.1 Holdout Method

In this method, a separate set of instances is reserved to assess the classifier's performance. This set is different from the training set for the learning algorithm. The learning algorithm takes as input a labeled set of instances for training and outputs a classifier. This classifier is then given the unlabeled set of instances from the testing set. The classifier outputs labels for each of these instances and the estimate of the empirical error is basically the fraction of test instances that the classifier misclassifies. Performance of the classifier on a separate test set is generally a good indicator of its generalization performance. Formal guarantees over the test set performance in terms of confidence intervals can be provided in this case. We show one such bound below that utilizes the Hoeffding's inequality. We establish the confidence interval within which we expect the true risk to fall with confidence $1-\delta$ for some $\delta \in (0, 1]$. However, this further goes to show the inherent limitation of this method too.

Consider an algorithm A that given some training set S outputs a classifier $h = A(S)$. We wish to estimate the true risk $R(h)$ in terms of the risk on a distinct test sample T (disjoint from training set S). We can define the empirical risk of h on test set T as:

$$R_T(h) \stackrel{\text{def}}{=} \frac{1}{m'} \sum_{i=1}^{m'} L(h(\mathbf{x}_i), y_i)$$

where $m' \stackrel{\text{def}}{=} |T|$ is the number of examples in the testing set. Note that the test set $T = \mathbf{z}_1, ..., \mathbf{z}_{m'}$ of m' samples is formed from the instantiation of the variables $\mathbf{Z}^{m'} \stackrel{\text{def}}{=} \mathbf{Z}_1, ..., \mathbf{Z}_{m'}$. Every $\mathbf{Z}_i$ is distributed according to some distribution D that generates the sample S. Each $\mathbf{z}_i$ consist of an example $\mathbf{x}_i$ and its label y_i. Each example $\mathbf{x}_i$ can hence be considered an instantiation of a variable $\mathbf{X}_i$ and its label y_i as an instantiation of variable Y_i.

Hence, over all the test sets generated from the instantiations of variables $\mathbf{Z}^{m'}$, the risk of some classifier h can be represented as:

$$R(\mathbf{Z}^{m'}, h) \stackrel{\text{def}}{=} \frac{1}{m'} \sum_{i=1}^{m'} L(h(\mathbf{X}_i), Y_i)$$

where $L()$ is again the loss function over the misclassification. Now, consider the loss function $L = L_z$ such that L_z is a Bernoulli variable, then the true risk can be expressed as:

$$R(h) = \{\Pr(L_z(h(\mathbf{X}), Y) = 1\} \stackrel{\text{def}}{=} p$$

In order to bound the true risk $R(h)$ we make use of the Hoeffding's inequality stated below:

Theorem 3. *(Hoeffding: Bernoulli case) For any sequence $Y_1, Y_2, ..., Y_{m'}$ of variables obeying a Bernoulli distribution with $\Pr(Y_i = 1) = p \ \forall i$, we have:*

$$\Pr\left[\left|\frac{1}{m'}\sum_{i=1}^{m'} Y_i - p\right| > \epsilon\right] \leq 2\exp(-2m'\epsilon^2)$$

This implies that:

$$\Pr\left[\left|\frac{1}{m'}\sum_{i=1}^{m'} Y_i - p\right| \leq \epsilon\right] \geq 1 - 2\exp(-2m'\epsilon^2)$$

Now, since $L_z(h(\mathbf{X}), Y)$ is a Bernoulli variable, we have:

$$\Pr\big[\big|R(\mathbf{Z}^{m'}, h) - R(h)\big| \leq \epsilon\big] \geq 1 - 2\exp(-2m'\epsilon^2)$$

Equating the right hand side of the above equation to $1 - \delta$, we get:

$$t_{1-\delta} = \epsilon = \sqrt{\frac{1}{2m'}\ln\left(\frac{2}{\delta}\right)}$$

Hence, for any classifier h, the generalization error $R(h)$, with probability $1 - \delta$ and test error $R_T(h)$ on some test set T, satisfies:

$$\big|R_T(h) - R(h)\big| \leq t_{1-\delta} = \sqrt{\frac{1}{2m'}\ln\left(\frac{2}{\delta}\right)} \tag{2.6}$$

Therefore, with probability $1 - \delta$:

$$R(h) \approx R_T(h) \pm t_{1-\delta}$$

Hence, it can be seen that the convergence of empirical risk to the true risk depends on the sample size m' and the true risk $R(h)$. This in turn gives us bound over the sample complexity. Rearranging Equation 2.6 and solving for sample size m', we get:

$$m' \geq \frac{1}{2\epsilon^2}\ln\left(\frac{2}{\delta}\right)$$

The sample size bound grows very quickly for small ϵ and δ. This shows that hold-out might not be a good idea if the dataset is not large enough. With a single train and test partition, too few cases in the training group can lead to learning a poor concept, while too few test cases can lead to erroneous error estimates.

Model Selection using Holdout Method

The test set performance of a hypothesis generally gives an idea of the "goodness" of the *selected* classifier. However, how can we utilize training data to select a final hypothesis that should be tested? The answer to this question lies in *model selection*, i.e. selecting the best hypothesis from among the potential candidate hypotheses. In the holdout scenario, a general approach to selecting the best hypothesis is to divide the data into three disjoint subsets (instead of two subsets as mentioned before): a training set, a validation set and a test set. The learning algorithm is trained on the training set yielding a set of candidate hypotheses (depending on say different parameter values that the algorithm takes as input). Each of these hypotheses are then tested on the validation set and the one that performs the best (i.e. makes least errors) on this validation set is then selected. Testing of this *selected* hypothesis is done in the same manner as discussed above via testing its performance on the test set.

2.5.2 Resampling

One of the most common difficulties in machine learning problems is the unavailability of enough data. Hence, as discussed above, if we use all the available data for training, it is not possible to have a good performance measure about the future performance of the classifier. Second, if we divide this already small dataset into training and testing sets, then reliable learning is not possible. Moreover, the sample size requirement grows exponentially with falling ϵ and δ. In such cases, researchers often make use of what are called the *resampling* methods.

As the name suggests, these methods are based upon the idea of being able to re-use the data for training and testing. Resampling has some advantages over single partition holdout method. Resampling allows for more accurate estimates of the error rates while training on most cases. Also, this method allows the duplication of the analysis conditions in future experiments on the same data. The most prominent resampling method is the k-fold cross validation:

- **k-fold Cross-validation method:** In k-fold cross validation, the training data are randomly divided into k (usually 10) mutually disjoint sets of approximately equal size (of at least 30 examples). The hypothesis is learned from the examples in $k-1$ sets, and is tested on the examples from the remaining set. This is repeated k times, once for each set (i.e. each set is once used as a test set). The average error rates over all k

sets is the cross-validated error rate.

- **Leave one out method:** The Leave one out estimate is a special case of k-fold cross validation such that $k = m$ where m is the number of examples in the training data. A hypothesis is learned from $m - 1$ examples and is tested on the remaining example. This is repeated m times, each time leaving out a different example. The error rate is the total number of errors on the single test case divided by m.

Model Selection using Resampling

In the resampling scenario, one of the prominent methods to perform model selection is *nested k-fold cross validation*. The idea behind the nested k-fold cross validation is to divide the dataset into k disjoint subsets just as done in the above described k-fold cross validation method. But now in addition, we perform a separate k-fold cross validation *within* the $k-1$ folds during training. The testing is as usual performed on the k^{th} fold. The rationale behind this approach is to make the algorithm totally unbiased in parameter selection.

Resampling methods can serve as good evaluation measures when the available data is limited. However, these methods also have some limitations. Resampling methods do not estimate the risk of a classifier but, rather, estimate the expected risk $E_S(A(S))$ of a learning algorithm A over samples S of size $m(1-1/k)$ for k-fold CV. Although, as mentioned above, these methods do try to take the most out of data by training on most cases but suffer from the fact that no confidence intervals are known and it is thus currently impossible to provide formal guarantees over the risk of classifier. This is in contrast with holdout methods where such guarantees and confidence intervals can be precisely stated. The sample standard deviation of k-fold CV risk R^k_{CV} over the k different groups serves at best to give a rough idea of the uncertainty of the estimate. However, training set bounds provide acceptably good guarantees over the generalization behavior of the classifier in terms of its empirical performance on training set. We will present these bounds in detail in Chapter 3. There are many variants of resampling methods. See (Weiss and Kapouleas, 1989, Mitchell, 1997, Kibler and Langley, 1988) for resampling and other evaluation methods.

As discussed before, Statistical Learning Theory provides a *formalization* to Machine Learning in the form of underlying frameworks guiding the algorithms as well as dealing with other theoretical issues such as providing mathematical guarantees over the error rate and performance of classifier. The basic idea behind the theoretical performance measure for empirical evaluation of classifier come in the form of *generalization error bounds* also known as risk bounds. The generalization error bounds basically provide upper (and possibly

lower) ranges over the empirical errors obtained on training (or test) data in which one would expect the true error to fall. These bounds can depend not only on the classifier's performance on the training or test set but also on the framework under which the learning algorithm is functioning such as complexity of the hypothesis class. The error bound that we presented in Section 2.5.1 is an example of one such bound based on the classifier's test set performance. We give more details about some prominent learning frameworks and these bounds in Chapter 3.

2.6 Bibliographical Notes

In addition to the cited literature in the text, please follow the following links for more detailed account of various topics.

For a detailed discussion on the various types of learning and issues in machine learning see (Mitchell, 1997, Duda, Hart and Stork, 2001, Guyon and Elisseeff, 2003, Witten and Frank, 2000) and the references therein. For history of Machine Learning see (Snell, 1997) and references therein. For details on Unsupervised learning see (Mitchell, 1997, Hinton and Sejnowski, 1999, Hastie, Tibshirani and Friedman, 2001). One of the main result in unsupervised learning in the field of Pattern Recognition has appeared in the form of Kohonen's Algorithm and related approaches, the main being Self Organizing Maps (SOMs). Details can be found in (Kohonen, 1982, 1989, 1995). SOMs provide a way to represent multidimensional data into a lower dimensional space. This dimensionality reduction approach is basically a data-compression scheme called Vector Quantization. See (Gersho and Gray, 1992) for details on vector quantization and data-compression.

For details on Empirical evaluation see (Mitchell, 1997, Dietterich, 1998, Provost and Fawcett, 1998, Demšar, 2006). For details on Boosting see (Meir and Rätsch, 2003, Schapire, 1990, 1999). On the lines of sample compression, approaches such as Prototype Reduction Schemes (PRS) have also shown good performance. The PRS have been explored to many tasks. See (Bezdek and Kuncheva, 2001) for PRS useful in Nearest Neighbor classification for instance. However, obtaining bounds on the generalization performance of the classifiers is quite difficult in such cases.

CHAPTER 3

Statistical Learning Theory: An Introduction

This Chapter presents an introduction to Statistical Learning Theory. In particular, we present an overview of different algorithmic approaches to learning and cover some main mathematical models of learning. For each model, we present a risk bound on the generalization ability of learning algorithms by studying the uniform convergence of expected risk. The bounds are called Uniform Risk Bounds.

3.1 Introduction

As discussed before, Machine Learning aims at automating the process of inductive inference. A *formalization* of this process is lent by Statistical Learning Theory. Design and implementation of competent and effective machine learning algorithms warrants a scrutinized detailing of various components both formally and analytically. To this end, statistical learning theory provides a framework to characterize the behavior of the learning algorithm. This is generally done by providing a mathematical model in which the learning is performed. The theory focuses on issues such as learnability of the task in the model, sample and/or time complexity of the learning algorithm and finally guarantees on the generalization performance of the algorithm. The goals of statistical learning theory can broadly be summarized as:

1. Providing machine learning with a mathematical framework;
2. Within this framework, providing guarantees for various aspects of learning such as generalization error and sample complexity;
3. Analyzing the learning problems to guide the learning process. This analysis can help in providing answers to issues like the level of difficulty that is inherent to a particular learning task and the best suited algorithm to learn this task.

Our main focus will be the mathematical models of learning. The main advantage of a mathematical learning model is that guarantees on the generalization ability of the learning

algorithm can be obtained from them. We break our study of statistical learning theory into *Learning Algorithms*, *Learning models* and *Risk Bounds*. The algorithmic approaches to learning focus on: how to select the best hypothesis in light of the available data? Two prominent approaches exist: the *empirical risk minimization* approach and the *structural risk minimization* approach. Also there are some other methods that are built on top of one or both of these two approaches, such as *Regularization*. We will discuss these briefly in this chapter.

We know that there is no way to calculate the true risk $R(f)$ for $f \in \mathcal{F}$. Hence, we need to obtain as best an estimate of true risk as possible using the empirical risk $R_S(f)$ that can be calculated on the available data. That is, we wish to investigate the (true) risk of classifiers under various models in terms of what they can achieve on the training data (e.g. empirical risk). We will show that rather than studying the generalization behavior of a learning algorithm it is sufficient to study the uniform convergence of the empirical risk to the true risk over all the classifiers. Under each of the learning models that we discuss, we will present such bounds based on the convergence of expected risks. These bounds are collectively called Uniform risk bounds.

The main mathematical learning models that we will study include: The PAC (Probably Approximately Correct) and VC models, the Occam Razor model and the Sample Compression model. We will also briefly review the PAC-Bayes framework that provides PAC guarantees to Bayesian algorithms. The bounds, especially the PAC, VC and the sample compression risk bounds, are along the lines of Marchand (2004). We further derive the Occam's Razor bound under similar settings. The PAC-Bayes bound is due to McAllester (2003). The version we present is due to Langford (2005) and Seeger (2002).

In this thesis, we find general bounds for classifiers that allow the learner to make some training error. The dependence of bounds on what can be achieved on training data is not only in terms of the training error. In coming chapters, we will see how different quantities over the training performance are added to the bound considerations. These quantities might include, for instance, the sparsity of the solution (using the minimal number of training examples to specify the classifier) and the separating margin achieved by the classifiers. In fact, studying the interplay between sparsity and the separating margin of the decision surface of classifier and exploiting this interplay to obtain better generalization will be one of the main aims of our algorithmic designs.

Moreover, we will mainly derive PAC-style bounds[1] of the following form: For some $\delta \in [0, 1]$, *with probability* $1 - \delta$*, over a random draw of some sample from a distribution* D*, the expected error rate of the classifier is bounded by some function of* δ*, the empirical risk of classifier and some other properties of the classifier computable from* S *(such as the compression set, the margin, etc.).* These bounds are also commonly known as training set bounds.

There is another class of bounds that depends on the empirical error rate of the classifier on a *separate testing set*. We gave an example of such a test set bound obtained from the Hoeffding's inequality in Section 2.5.1. These bounds depict the probability of the amount of deviation of the generalization error of the classifier from the performance of the classifier on a set of test examples. These test examples are not used in training of the learning algorithms that generated the classifier. Hence, these *test set bounds* are unbiased. However, there are factors that limit this approach, the main being the limited availability of data, reserving a separate test set from which is not possible without adversely affecting the training of the learner. Another reason supporting the training set bounds, as discussed by Langford (2005) is that many algorithms implicitly assume that the train set accuracy is very close to the true error behavior of the classifier and hence more amount of training data available can guide the process of choosing the classifier better.

3.2 Definitions

The classification problems we consider are the binary classification problems. The input space $\mathcal{X}$ consists of an arbitrary subset of $\mathbb{R}^n$ and the output space $\mathcal{Y} = \{-1, +1\}$. We denote a training set by S. The training set $S = \mathbf{z}_1, ..., \mathbf{z}_m$ of m samples is formed from the instantiation of the variables $\mathbf{Z}^m \stackrel{\text{def}}{=} \mathbf{Z}_1, ..., \mathbf{Z}_m$. Each $\mathbf{z}_i$ consist of an example $\mathbf{x}_i$ and its label y_i.

The learning algorithm A outputs a classifier $h \in \mathcal{H}$, when input some training set S. We denote by $R_{\mathbf{Z}^m}(h)$, the empirical risk $R_S(h)$ and by $\mathbf{P}_{\mathbf{Z}^m}(a)$ the probability, over one random draw of $\mathbf{Z}^m$, that predicate a is true. $R(h)$ denotes the true risk of the classifier h.

[1] See Section 3.5

3.3 Learning Algorithms

The question of choosing the best hypothesis that not only describes the observed data but can also generalize well is to the core of various algorithmic approaches to learning. Note that for any learning method, we require its performance to improve with training sample size so that the generalization error should decrease as the sample size increases. We discuss the Empirical risk minimization and the Structural risk minimization approach to achieve this. Both these approaches by themselves have been proven to be quite effective and have also paved the way for other approaches that exploit these to yield better ones. Note, however, that unlike the structural risk minimization, the empirical risk minimization approaches focus on one hypothesis class and hence are relatively restrictive.

3.3.1 Empirical Risk Minimization

Given some training set S, the empirical risk minimization (ERM) algorithm A_{erm} outputs the hypothesis that minimizes the empirical risk on the training set. That is,

$$A_{\text{erm}}(S) = h' \stackrel{\text{def}}{=} \underset{h \in \mathcal{H}}{\operatorname{argmin}}\ R_S(h)$$

3.3.2 Structural Risk Minimization

The basic idea behind a structural risk minimization algorithm (SRM) is to choose a hypothesis (or model) with the least complexity (also referred to as *size* or *capacity*) that achieves a small empirical risk. For this, $\mathcal{H}$ is represented using a sequence of hypotheses of increasing sizes $\{\mathcal{H}_d : d = 1, 2, \ldots\}$ and the structural risk minimization algorithm A_{srm} is such that:

$$A_{\text{srm}}(S) = h' \stackrel{\text{def}}{=} \underset{h \in \mathcal{H}_d, d \in \mathbb{N}}{\operatorname{argmin}}\ R_S(h) + p(d, |S|)$$

where $p(d, |S|)$ is a function that penalizes the algorithm for hypothesis spaces of increasing complexity.

Let cm be some complexity measure on hypothesis space. We would have a set of hypothesis spaces $\mathcal{H} = \{\mathcal{H}_1, \ldots, \mathcal{H}_k\}$ such that the complexity of hypothesis space $\mathcal{H}_i$ denoted as $cm_{\mathcal{H}_i}$ is greater than or equal to $cm_{\mathcal{H}_{i-1}}$. Then the structural risk minimization algorithm would be to compute a set of hypotheses minimizing the empirical risk over the hypothesis spaces $\mathcal{H} = \{\mathcal{H}_1, \ldots, \mathcal{H}_k\}$ and then to select the hypothesis space that gives the best trade-off between its complexity and the minimum empirical risk obtained over it on the training data.

3.3.3 Regularization

There are other approaches that extend the above algorithms such as regularization where one tries to minimize the regularized empirical risk. This is done by defining a *regularizing term* or a *regularizer* (typically a norm on the hypothesis $||h||$) over the hypothesis space $\mathcal{H}$ such that the algorithm outputs a hypothesis h':

$$h' = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \; R_S(h) + \lambda ||h||^2$$

Variants of this approach exist such as Normalized regularization. The details on these and other approaches can be found in (Herbrich, 2002, Bousquet, Boucheron and Lugosi, 2004, Hastie, Tibshirani and Friedman, 2001) among others.

3.4 ERM and Uniform Convergence

Recall that given some training set S formed from m examples such that each example $\mathbf{z} = (\mathbf{x}, y)$ is formed from the instantiation of variable $\mathbf{Z} = (\mathbf{X}, Y)$, the empirical risk minimization algorithm A_{erm} is such that:

$$A_{\text{erm}}(S) = h' \stackrel{\text{def}}{=} \underset{h \in \mathcal{H}}{\operatorname{argmin}} \; R_S(h)$$

since the risk of the classifier output by the algorithm A_{erm} can be minimized only by minimizing the empirical risk on training set S. We focus on algorithms that *learn by risk minimization*. Recall that the empirical risk $R_S(h)$ is defined as:

$$R_S(h) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} L(h(\mathbf{x}), y)$$

where L is some loss function over the misclassification of some example $\mathbf{x}$ by a classifier h. The true risk is just the expectation over this risk:

$$R(h) = E_{\mathbf{X},Y} L(h(\mathbf{X}), Y) = \Pr(h(\mathbf{X}) \neq Y)$$

Now, the classifier with the minimal true risk, denoted by h^*, is such that:

$$h^* \stackrel{\text{def}}{=} \underset{h \in \mathcal{H}}{\operatorname{argmin}} \; R(h)$$

Since, $R_S(h^*) \geq R_S(h')$, we can bound $R(h')$ by:

$$\begin{aligned} R(h') - R(h^*) &\leq R(h') - R(h^*) + [R_S(h^*) - R_S(h')] \\ &= |R(h') - R_S(h') + (R_S(h^*) - R(h^*))| \\ &\leq |R(h') - R_S(h')| + |R(h^*) - R_S(h^*)| \\ &\leq 2 \sup_{h \in \mathcal{H}} |R(h) - R_S(h)| \end{aligned}$$

This goes to show that it is sufficient to consider the *uniform convergence of empirical risk to expected risk* over all classifiers $h \in \mathcal{H}$ instead of focusing on the generalization error of the empirical risk minimization algorithm directly since any upper bound on the convergence is also an upper bound on the risk of ERM algorithm. We study such bounds and hence call them *Uniform Risk Bounds*.

3.5 PAC and VC Learning

The PAC Learning model was introduced by Valiant (1984) with the computational efficiency of learning algorithms as the central theme. This computational efficiency of learning algorithm appeared in the form of the notion of *feasibility of a learning problem*. A learning problem is *feasible* if there exists an algorithm that can solve it in polynomial amount of time. The PAC-model aims at successful learning of some unknown target concept, with high probability, by a hypothesis that approximates this target concept closely. Since the learning algorithm, instead of trying to exactly learn the target concept, tries to approximate it as accurately as possible with a high probability, the model is called the *Probably Approximately Correct* learning model. The original work of Valiant (1984) focused on a more specialized task of learning logical formulae assuming the existence of a hypothesis space $\mathcal{H}$ that included the target formula to be learned. The distribution of the input formulae $\mathcal{X}$, however, was unknown. This can be viewed more generally as the problem of the uniform convergence of the empirical risk to the true risk. The analysis focused on finite hypothesis space, i.e. $|\mathcal{H}| < \infty$. Various generalizations have since been proposed to tackle the issue of infinite hypothesis space based on some measure(s) related to the complexity of the hypothesis class, like its VC dimension. Note that PAC results are generally stated as the number of examples needed to achieve a risk less than ϵ with a probability $1 - \delta$, where $\epsilon, \delta \in [0, 1]$. This is equivalent to stating a risk bound in terms of ϵ as a function of the number of examples in the training set.

The VC (Vapnik-Chervonenkis) framework (Vapnik and Chervonenkis, 1971, Vapnik, 1982) also studies the convergence of expected risk using an *a-priori* measure of complexity of the hypothesis space called the Growth function. However, owing to the difficulty of computing the growth function, a quantity called VC-dimension is used to characterize it. VC dimension is also known as an integer-summary of the growth function. We will define VC dimension in Section 3.5.2.

The difference between the PAC and VC approach is that PAC learning assumes $R(h^*) = 0$, i.e. $h^*(\mathbf{x}_i) = y_i \ \forall i \in S$. Also, the PAC framework assumes $R_S(h') = 0$ for a classifier h' output by the algorithm A_{erm} on a sample S of m examples. Hence:

$$R(h') \leq \sup_{h \in \mathcal{H}: R_S(h)=0} R(h)$$

Let us now study the PAC and VC risk bounds:

3.5.1 Bounds for Finite Hypothesis Space

Let us look for the bound on $\sup_{h\in\mathcal{H}}(R(h) - R_S(h))$ when $|\mathcal{H}| < \infty$ that is when the hypothesis space is finite. Then:

$$\begin{aligned} P' &\stackrel{\text{def}}{=} \mathbf{P}_{\mathbf{Z}^m}\big(\sup_{h\in\mathcal{H}}(R(h) - R_{\mathbf{Z}^m}(h)) > \epsilon\big) \\ &= \mathbf{P}_{\mathbf{Z}^m}\big(\exists h \in \mathcal{H} : R(h) - R_{\mathbf{Z}^m}(h) > \epsilon\big) \\ &= \mathbf{P}_{\mathbf{Z}^m}\Big(\bigvee_{i=1}^{|\mathcal{H}|} R(h_i) - R_{\mathbf{Z}^m}(h_i) > \epsilon\Big) \end{aligned}$$

Now, We know that the union bound says that:

Theorem 4. *(Union Bound) Let $X_1, X_2, ...X_n \subseteq \mathcal{X}$ be a finite number of sets from superset $\mathcal{X}$, then, for any measure $\mathbf{P}_{\mathcal{X}}$:*

$$\mathbf{P}_{\mathcal{X}}(X_1 \bigcup X_2 \bigcup ... \bigcup X_n) \leq \sum_{i=1}^{n} \mathbf{P}_{\mathcal{X}}(X_i).$$

Hence:

$$\mathbf{P}_{\mathbf{Z}^m}\Big(\bigvee_{i=1}^{|\mathcal{H}|} R(h_i) - R_{\mathbf{Z}^m}(h_i) > \epsilon\Big) \leq \sum_{i=1}^{|\mathcal{H}|} \mathbf{P}_{\mathbf{Z}^m}\big(R(h_i) - R_{\mathbf{Z}^m}(h_i) > \epsilon\big) \tag{3.1}$$

Now, for any fixed hypothesis h, the true risk and the empirical risk are nothing but the expectation and mean of a random variable between 0 and 1. Hence, we can apply the Hoeffding's inequality here to obtain:

$$\mathbf{P}_{\mathbf{Z}^m}\big(R(h) - R_{\mathbf{Z}^m}(h) > \epsilon\big) \leq \exp(-2m\epsilon^2) \ \forall h \tag{3.2}$$

Combining Equations 3.1 and 3.2, we get:

$$\mathbf{P}_{\mathbf{Z}^m}\big(\sup_{h\in\mathcal{H}}(R(h) - R_{\mathbf{Z}^m}(h)) > \epsilon\big) \leq |\mathcal{H}| \exp(-2m\epsilon^2)$$

Equivalently:

$$\mathbf{P}_{\mathbf{Z}^m}\big(\forall h \in \mathcal{H} : (R(h) - R_{\mathbf{Z}^m}(h)) \leq \epsilon\big) \geq 1 - |\mathcal{H}| \exp(-2m\epsilon^2)$$

Now, $|\mathcal{H}| \exp(-2m\epsilon^2) = \delta$ when

$$\epsilon^2 = \frac{1}{2m} \ln\left(\frac{|\mathcal{H}|}{\delta}\right)$$

Hence, we proved the following theorem:

Theorem 5. *For any finite space $\mathcal{H}$ of classifiers and $\forall \delta \in (0, 1]$:*

$$\mathbf{P}_{\mathbf{Z}^m}\left(\forall h \in \mathcal{H} : R(h) \leq R_{\mathbf{Z}^m}(h) + \sqrt{\frac{1}{2m} \ln\left(\frac{|\mathcal{H}|}{\delta}\right)}\right) \geq 1 - \delta$$

The above bound can be improved for the risk of the classifier output by algorithm A_{erm} on training set S when $\exists h \in \mathcal{H} : R(h) = 0$. This is the case when the algorithm A_{erm} outputs a classifier with zero empirical risk, i.e. $R_{\mathbf{Z}^m}(h) = 0$. Hence, we bound the following probability:

$$P' \stackrel{\text{def}}{=} \mathbf{P}_{\mathbf{Z}^m}\big(\exists h \in \mathcal{H} : R(h) > \epsilon \wedge R_{\mathbf{Z}^m}(h) = 0\big)$$

Bounding this above P' by δ, we get:

$$\mathbf{P}_{\mathbf{Z}^m}\big(\forall h \in \mathcal{H} : R(h) \leq \epsilon \vee R_{\mathbf{Z}^m}(h) \neq 0\big) \geq 1 - \delta$$

This implies the following:

$$\mathbf{P}_{\mathbf{Z}^m}\big(\forall h \in \mathcal{H} : R_{\mathbf{Z}^m}(h) = 0 \Longrightarrow R(h) \leq \epsilon\big) \geq 1 - \delta$$

Hence, applying union bound, we get:

$$P' \stackrel{\text{def}}{=} \mathbf{P}_{\mathbf{Z}^m}\big(\exists h \in \mathcal{H} : R(h) > \epsilon \wedge R_{\mathbf{Z}^m}(h) = 0\big) \leq \sum_{i=1}^{|\mathcal{H}|} \mathbf{P}_{\mathbf{Z}^m}\big(R(h_i) > \epsilon \wedge R_{\mathbf{Z}^m}(h_i) = 0\big)$$

Limiting this probability now (using the union bound):

$$\mathbf{P}_{\mathbf{Z}^m}\big(\exists h \in \mathcal{H} : R(h) > \epsilon \wedge R_{\mathbf{Z}^m}(h) = 0\big) \leq \sum_{h\in\mathcal{H}} \mathbf{P}_{\mathbf{Z}^m}\big(R_{\mathbf{Z}^m}(h) = 0 \wedge R(h) > \epsilon\big)$$
$$\leq \sum_{h\in\mathcal{H}} \mathbf{P}_{\mathbf{Z}^m}\big(R_{\mathbf{Z}^m}(h) = 0 | R(h) > \epsilon\big)$$

Now, for any classifier h having $R(h) > \epsilon$, we have:

$$\mathbf{P}_{\mathbf{Z}}\big(h(\mathbf{X}) = Y\big) < 1 - \epsilon \leq e^{-\epsilon}$$

Since every example $\mathbf{z}_i$ is generated i.i.d. from distribution $\mathbf{P}_{\mathbf{Z}}$, we have, for m examples:

$$\mathbf{P}_{\mathbf{Z}^m}\big(R_{\mathbf{Z}^m}(h) = 0 | R(h) > \epsilon\big) \leq (1-\epsilon)^m \leq e^{-m\epsilon} \ \forall h$$

Hence,

$$\mathbf{P}_{\mathbf{Z}^m}\big(R(h) > \epsilon \wedge R_{\mathbf{Z}^m}(h) = 0\big) \leq (1-\epsilon)^m \leq e^{-m\epsilon} \ \forall h \tag{3.3}$$

This implies:

$$\mathbf{P}_{\mathbf{Z}^m}\big(\exists h \in \mathcal{H} : R(h) > \epsilon \wedge R_{\mathbf{Z}^m}(h) = 0\big) \leq |\mathcal{H}| e^{-m\epsilon}$$

which further implies that:

$$\mathbf{P}_{\mathbf{Z}^m}\big(\forall h \in \mathcal{H} : R_{\mathbf{Z}^m}(h) = 0 \implies R(h) \leq \epsilon\big) \geq 1 - |\mathcal{H}| e^{-m\epsilon}$$

Now equating $\delta = |\mathcal{H}| e^{-m\epsilon}$, we have:

$$\epsilon = \frac{1}{m} \ln\left(\frac{|\mathcal{H}|}{\delta}\right)$$

This gives us:

$$\mathbf{P}_{\mathbf{Z}^m}\left(\forall h \in \mathcal{H} : R_{\mathbf{Z}^m}(h) = 0 \implies R(h) \leq \frac{1}{m} \ln\left(\frac{|\mathcal{H}|}{\delta}\right)\right) \geq 1 - \delta$$

Hence, we proved the following theorem:

Theorem 6. *For any finite space $\mathcal{H}$ of classifiers and $\forall \delta \in (0, 1]$:*

$$\mathbf{P}_{\mathbf{Z}^m}\left(\forall h \in \mathcal{H} : R_{\mathbf{Z}^m}(h) = 0 \implies R(h) \leq \frac{1}{m} \ln\left(\frac{|\mathcal{H}|}{\delta}\right)\right) \geq 1 - \delta$$

Note that the bound of Theorem 6 is tighter than the bound of Theorem 5 since here we bound the generalization error by a factor of $1/m$ as opposed to the previous bound where this factor is $1/\sqrt{m}$.

3.5.2 Bound for Infinite Hypothesis Spaces

The above PAC bound over the finite hypothesis space can be extended to the case when the classifier space $\mathcal{H}$ is infinite. We first derive a bound when $\mathcal{H}$ is countably infinite[2]. For this, let us introduce a distribution P over the space of classifiers $\mathcal{H}$. This distribution is the *a priori* distribution without any reference to the training sample S. The only condition that this distribution should satisfy is:

$$\sum_{h \in \mathcal{H}} P(h) \leq 1$$

For the case when the empirical risk is zero, i.e. $\exists h \in \mathcal{H} : R_{\mathbf{Z}^m}(h) = 0$, we bound:

$$P' \stackrel{\text{def}}{=} \mathbf{P}_{\mathbf{Z}^m}\big(\exists h \in \mathcal{H} : R(h) > \epsilon \wedge R_{\mathbf{Z}^m}(h) = 0\big)$$

Utilizing Equation 3.3 and equating it to $P(h)\delta$, we have:

$$\mathbf{P}_{\mathbf{Z}^m}\big(R(h) > \epsilon \wedge R_{\mathbf{Z}^m}(h) = 0\big) \leq e^{-m\epsilon} = P(h)\delta \;\; \forall h \in \mathcal{H} \tag{3.4}$$

In this case the ϵ depends on h, and yields:

$$\epsilon(h) = \frac{1}{m} \ln \left(\frac{1}{P(h)\delta} \right)$$

Applying the union bound gives us:

$$\begin{aligned} P' &\stackrel{\text{def}}{=} \mathbf{P}_{\mathbf{Z}^m}\big(\exists h \in \mathcal{H} : R(h) > \epsilon \wedge R_{\mathbf{Z}^m}(h) = 0\big) \\ &\leq \sum_{h \in \mathcal{H}} \mathbf{P}_{\mathbf{Z}^m}\big(R(h) > \epsilon \wedge R_{\mathbf{Z}^m}(h) = 0\big) \leq \sum_{h \in \mathcal{H}} P(h)\delta \leq \delta \end{aligned}$$

Note however that for above bound to hold, $\mathcal{H}$ should be countable.

For any distribution $P(h)$ such that $P(h) > 0, \forall h \in \mathcal{H}$, we have:

$$\mathbf{P}_{\mathbf{Z}^m}\left(\exists h \in \mathcal{H} : R_{\mathbf{Z}^m}(h) = 0 \Longrightarrow R(h) \leq \frac{1}{m} \ln \left(\frac{1}{P(h)\delta} \right) \right) \geq 1 - \delta$$

Hence, we have shown the following theorem:

[2]VC bound can be used for infinite $\mathcal{H}$ as shown below.

Theorem 7. *For any measure $P(h)$ over the countably infinite space of classifiers $\mathcal{H}$ such that $P(h) > 0, \forall h \in \mathcal{H}$, and $\forall \delta \in (0, 1]$:*

$$\mathbf{P}_{\mathbf{Z}^m}\left(\exists h \in \mathcal{H} : R_{\mathbf{Z}^m}(h) = 0 \Longrightarrow R(h) \leq \frac{1}{m} \ln\left(\frac{1}{P(h)\delta}\right)\right) \geq 1 - \delta$$

Note that the Theorem 6 is a special case of Theorem 7 when the classifier space $\mathcal{H}$ is finite and $P(h) = 1/|\mathcal{H}|, \forall h \in \mathcal{H}$.

As we discussed earlier, the VC framework uses an *a-priori* measure of complexity of the hypothesis space called the Growth function.[3] However, owing to the difficulty of computing the growth function,[4] a quantity called VC-dimension is used to characterize it. In simple terms, the VC dimension of hypothesis space $\mathcal{H}$, *is defined to be the maximum number d of instances that can be labeled as positive and negative examples in all 2^d possible ways, such that each labeling is consistent with some hypothesis in $\mathcal{H}$(Vapnik and Chervonenkis, 1971, Vapnik, 1982, Cover, 1965)(Haussler, 1992).*

Using this VC dimension d, a bound of the following form on the true risk of a classifier can be obtained. The following bound is due to Vapnik (1998). The version presented is due to Herbrich (2002)[5]:

Theorem 8. *(VC bound) For any space of classifiers $\mathcal{H}$, $\forall h \in \mathcal{H}$, and $\forall \delta \in (0, 1]$:*

$$\forall 2m > d : \mathbf{P}_{\mathbf{Z}^m}\big(R(h) - R_{\mathbf{Z}^m}(h) \leq \epsilon(d)\big) \geq 1 - \delta$$

where,

$$\epsilon(d) = \sqrt{8\left(\frac{\ln\frac{4}{\delta}}{m} + \frac{d}{m}\big(\ln\frac{2m}{d} + 1\big)\right)}$$

3.6 Occam Razor Learning

The Occam approach to learning has resulted from the principle wielded by theologian William of Occam. The principle tends to simply avoid the superfluous elaborations. With respect to learning, the Occam's principle can be interpreted as (Blumer et. al., 1987): *Given two explanations of the data, all other things being equal, the simpler explanation is preferable.* Hence, the principle can be seen as the one aiming to find the simplest hypothesis consistent with the sample data (Angluin and Smith, 1983). Since the principle tends to cut off unnecessary information, it has been known as *Occam's Razor.*

[3] We do not discuss the growth function analysis here. Please refer to Herbrich (2002) for detailed analysis.

[4] Note that we need to calculate the growth function *before* learning which is not possible for an arbitrary $\mathcal{H}$ and any m.

[5] Please refer to Herbrich (2002) for details on the proof of the theorem.

The Occam model of learning, in contrast with the PAC model, makes no assumption about the distribution from which samples are drawn. The PAC model defined learning directly in terms of the predictive power of the selected hypothesis that the learner identifies from the hypothesis class. However, this criterion made an implicit assumption that the training as well as testing was performed on samples drawn from *fixed* probability distribution $\mathcal{D}$. The Occam model makes no such assumption about the distribution of instances. The labels of the examples, however, are still generated by an unknown target concept (chosen from a known class). The hypothesis that explains the *observed* data most succinctly is preferred.

The succinctness might take different forms. For instance, the succinctness might aim to find or discover a pattern in the observed data so as to output a compact representation of this data. The succinctness can also be measured in terms of the size of the representation on the hypothesis directly. Yet another way of measuring succinctness can be in terms of the cardinality of the hypothesis class used by the algorithm, since a typical hypothesis from a hypothesis class with low cardinality can be represented by a short (possibly binary) string and vice-versa. It should be noted that the last measure would not work in the cases where the hypothesis class is infinite. The measure of *VC-dimension* can be useful in such cases.

The key question that arises now is: *Is Succinctness a guarantee for good generalization?*. In order to be able to learn successfully, we indeed need such a guarantee. It has been shown that, in the restricted setting of the PAC model, Occam algorithms have predictive powers, i.e. Occam learning implies PAC learning in this restricted setting (Blumer et. al., 1987, Kearns & Vazirani, 1994). Hence, under appropriate conditions, any algorithm that can find a succinct representation of an hypothesis consistent with the training sample is automatically a PAC learning algorithm.

We now present the theorem that relates the notion of Occam learning to PAC learning. Specifically, we are interested in bounding the error ϵ of an algorithm A with a probability $1 - \delta$ with the help of an *a-priori* distribution over the hypothesis as an implicit measure of succinctness. Note that if $P_{\mathcal{H}}(h)$ is the *a priori* distribution over the hypothesis space, then $log(1/P(h))$ is basically the amount of information (in bits if $log_2()$ is used) to identify $h \in \mathcal{H}$. Hence, this distribution acts as a measure of succinctness.

We derive the bound on the true risk of a learning algorithm A. Recall that if $A(\mathbf{Z}^m)$ is the classifier output by the algorithm A on some instantiation of $\mathbf{Z}^m$, then we first bound P' where:

$$P' \stackrel{\text{def}}{=} \mathbf{P}_{\mathbf{Z}^m}\left(R(A(\mathbf{Z}^m)) > \epsilon\right) \tag{3.5}$$

Hence, the goal is to find the smallest value for ϵ such that $P' \leq \delta$. In that case, we will have:

$$\mathbf{P}_{\mathbf{Z}^m}\left(R(A(\mathbf{Z}^m)) \leq \epsilon\right) \geq 1 - \delta$$

Applying the union bound, we have:

$$\begin{aligned} P' &\leq \mathbf{P}_{\mathbf{Z}^m}\left(\exists h \in \mathcal{H}\colon R(h) < \epsilon\right) \\ &\leq \sum_{h \in \mathcal{H}} \mathbf{P}_{\mathbf{Z}^m}\left(R(h) < \epsilon\right) \end{aligned} \tag{3.6}$$

Now, let us say that the classifier make k errors over m training examples. Also, we allow ϵ to depend on h. Hence, we can now write the following:

$$P_{Z^m}(R(h) < \epsilon(h)) = \sum_{k=0}^{m} P_{Z^m}(R(h) < \epsilon(h), R_{Z^m}(h) = k/m) \leq \sum_{k=0}^{m} \binom{m}{k} (1 - \epsilon(h))^{m-k} \tag{3.7}$$

Hence, the Equations 3.6 and 3.7, yield:

$$P' \leq \sum_{h \in \mathcal{H}} \sum_{k=0}^{m} \binom{m}{k} (1 - \epsilon(h))^{m-k} \tag{3.8}$$

For any classifier h, we use any function $P_{\mathcal{H}}(h)$ having the following property:

$$\sum_{h \in \mathcal{H}} P_{\mathcal{H}}(h) \leq 1 \tag{3.9}$$

Hence, the function $P_{\mathcal{H}}(h)$ can be interpreted as an *a priori* distribution over $\mathcal{H}$ when it sums to one. Let ϵ be such that:

$$\binom{m}{k} \left[1 - \epsilon(h, k)\right]^{m-k} = P_{\mathcal{H}}(h) \cdot \zeta(k) \cdot \delta \tag{3.10}$$

where ζ is defined by:

$$\zeta(a) \stackrel{\text{def}}{=} \frac{6}{\pi^2}(a+1)^{-2} \tag{3.11}$$

For all non-negative integer a, it is well known that

$$\sum_{a=0}^{\infty} \zeta(a) = 1.$$

Therefore, with the following $\epsilon(h)$:

$$\epsilon(h) = \epsilon(h, k, \delta) = 1 - \exp\left(\frac{-1}{m-k}\left[\ln\binom{m}{k} + \ln\left(\frac{1}{P_{\mathcal{H}}(h)}\right) + \ln\left(\frac{1}{\zeta(k)}\right) + \ln\left(\frac{1}{\delta}\right)\right]\right) \tag{3.12}$$

we have that $P' \leq \delta$. Hence, We have the following theorem:

Theorem 9. *For any learning algorithm A that outputs a classifier $h \in \mathcal{H}$ when given a training set of m examples on which h makes $k \leq m$ errors:*

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall h \in \mathcal{H}\colon R(A(\mathbf{Z}^m)) \leq \epsilon(h, k, \delta)\right\} \geq 1 - \delta$$

where $\epsilon(h, k, \delta)$ is given by Equation 3.12.

The consistent case of the upper bound follows. When, $h \in \mathcal{H}$ makes no error on the training set, we have:

$$\epsilon(h) = 1 - \exp\left(\frac{-1}{m}\left[\ln\left(\frac{1}{P_{\mathcal{H}}(h)}\right) + \ln\left(\frac{1}{\delta}\right)\right]\right)$$

3.7 Sample Compression Learning

The notion of *Sample Compression* comes from the work of Littlestone and Warmuth (1986) in which they explored the learnability of boolean functions from samples along the lines of Data Compression. Algorithms using sample compression select a subset of samples from a training set of size m, and use these samples to represent a hypothesis. The basic idea is to have a compression scheme $\mathbf{i}$ of size $|\mathbf{i}|$ ($|\mathbf{i}| \leq m$) for a concept class that consists of two functions:

1. a compression function $\mathcal{C}$; and
2. a reconstruction function $\mathcal{R}$.

The compression function takes as input a finite sample (set of examples) and outputs a subset $\mathbf{i}$ of size at most $|\mathbf{i}|$ ($|\mathbf{i}| \leq m$) of these examples. This subset is known as the *compression set*[6]. The task of the reconstruction function is to use this compression set and (re)construct the hypothesis for the target concept to be learned.

[6]the original work by Littlestone and Warmuth (1986) call this set as a *kernel*. However, we do not use this term here to avoid confusion

The idea of sample compression stems from the observation by Littlestone and Warmuth (1986) that many learning algorithms use a fixed-size subsample of a given labeled example to specify the hypothesis. Hence, it can be viewed as a compression of the sample of size m to a (sub)sample of some fixed size. This compressed sample or the *compression set* is enough to reconstruct the labels of the original m-sample. Littlestone and Warmuth (1986) also show that if the algorithm possesses these data compression characteristics then that alone is sufficient to guarantee learnability. The theorems relating the learnability and the compression characteristics of the algorithm are basically built on top of the PAC model. Littlestone and Warmuth (1986) also give sample size bounds for these cases.

A number of algorithms can be seen as sample compression based algorithms. The classical perceptron learning rule is an example of sample compression algorithm where the compression set consists of the examples used to update the weight vector and the threshold of the separating hyperplane. The reconstruction function is nothing but the same perceptron rule applied to the compression set in the order given by the training set S. The Support Vector Machine (Vapnik, 1998, Cristianini and Shawe-Taylor, 2000), is another such example. The SVM aims at compressing the training sample into a compression set. The examples of this set are called the support vectors. The reconstruction function is again nothing but the same maximum (soft) margin algorithm applied only to these support vectors.

3.7.1 Sample Compression Risk Bound

Let us start with a formal description of the compression scheme. Recall that, a learning algorithm A is said to be a *sample-compression algorithm* iff there exists a *compression function* $\mathcal{C}$ and a *reconstruction function* $\mathcal{R}$ such that for any training sample $S = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ (where $\mathbf{z}_i \stackrel{\text{def}}{=} (\mathbf{x}_i, y_i)$ for $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$), the classifier $A(S)$ returned by A is given by:

$$A(S) = \mathcal{R}(\mathcal{C}(S)) \quad \forall S \in (\mathcal{X} \times \mathcal{Y})^m$$

For a training set S, the compression function $\mathcal{C}$ of learning algorithm A outputs a subset $\mathbf{z_i}$ of S, called the *compression set*, and an *information message* σ:

$$(\mathbf{z_i}, \sigma) = \mathcal{C}(\mathbf{z}_1, \ldots, \mathbf{z}_m)$$

The information message σ contains the additional information needed to reconstruct the classifier from the compression set $\mathbf{z_i}$. Given a training sample S, we define the compression

set $\mathbf{z_i}$ by a vector of indices $\mathbf{i}$:

$$\mathbf{i} \stackrel{\text{def}}{=} (i_1, i_2, \ldots, i_{|\mathbf{i}|}) \tag{3.13}$$

$$\text{with} \quad : \quad i_j \in \{1, \ldots, m\} \; \forall j \tag{3.14}$$

$$\text{and} \quad : \quad i_1 < i_2 < \ldots < i_{|\mathbf{i}|} \tag{3.15}$$

where $|\mathbf{i}|$ denotes the number of indices present in $\mathbf{i}$.

When given an arbitrary compression set $\mathbf{z_i}$ and an arbitrary information message σ, the *reconstruction function* $\mathcal{R}$ of a learning algorithm A must output a classifier. Note that the information message σ is chosen from the set $\mathcal{M}(\mathbf{z_i})$. This set $\mathcal{M}(\mathbf{z_i})$ consist of all the distinct messages that can be attached to the compression set $\mathbf{z_i}$. Only the existence of this reconstruction function $\mathcal{R}$ justifies the classifier returned by $A(S)$ to *always* be identified by a compression set $\mathbf{z_i}$ and an information message σ.

We present the generic risk bounds for sample compression learning algorithms in terms of the amount of data compression they can perform on the training sample. The bound is inspired by the work of Littlestone and Warmuth (1986), Floyd and Warmuth (1995), Ben-David and Litman (1998) and Graepel, Herbrich, and Shawe-Taylor (2005). However, it can be seen that this bound is significantly tighter than the ones presented in the works mentioned above. The main observation behind the bounds is that any classifier that has a small compression set of size $|\mathbf{i}|$ associated with it and has a small empirical risk with the $m-|\mathbf{i}|$ remaining examples (i.e. the training examples other than the ones in the compression set), has necessarily a small true risk.

As before, we derive the bound on the true risk of a learning algorithm A by first bounding P' where:

$$P' \stackrel{\text{def}}{=} \mathbf{P}_{\mathbf{Z}^m} \left(R(A(\mathbf{Z}^m)) > \epsilon \right) \tag{3.16}$$

Hence, the goal is to find the smallest value for ϵ such that $P' \leq \delta$. In that case, we will have:

$$\mathbf{P}_{\mathbf{Z}^m} \left(R(A(\mathbf{Z}^m)) \leq \epsilon \right) \geq 1 - \delta$$

We denote by $\mathcal{I}$ the set of all 2^m vectors of indices as defined by equations 3.13, 3.14, and 3.15. $\mathcal{M}(\mathbf{z_i})$ denotes the set of all messages σ that can be attached to compression set $\mathbf{z_i}$. The empty message is assumed to always be present in $\mathcal{M}(\mathbf{z_i})$ so $|\mathcal{M}(\mathbf{z_i})| \geq 1$ is always satisfied. Also, we use $\bar{\mathbf{i}}$ to denote the the vector of indices made of all the indices not present in $\mathbf{i}$. Finally, we will allow ϵ to depend on $\mathbf{Z_i}$ and σ as we show further. Since

$\mathbf{P}_{\mathbf{Z}^m}(\cdot) = \mathbf{E}_{\mathbf{Z_i}}\mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}(\cdot)$, we have (by the union bound):

$$\begin{aligned} P' &\leq \mathbf{P}_{\mathbf{Z}^m}\left(\exists \mathbf{i} \in \mathcal{I} \colon \exists \sigma \in \mathcal{M}(\mathbf{Z_i}) \colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \epsilon\right) \\ &\leq \sum_{\mathbf{i}\in\mathcal{I}} \mathbf{E}_{\mathbf{Z_i}}\mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}\left(\exists \sigma \in \mathcal{M}(\mathbf{Z_i}) \colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \epsilon\right) \quad (3.17) \\ &\leq \sum_{\mathbf{i}\in\mathcal{I}} \mathbf{E}_{\mathbf{Z_i}} \sum_{\sigma\in\mathcal{M}(\mathbf{Z_i})} \mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}\left(R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \epsilon\right) \quad (3.18) \end{aligned}$$

Let us now stratify $\mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}(R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \epsilon)$ in terms of the errors that $\mathcal{R}(\sigma, \mathbf{Z_i})$ can make on the *training examples that are not used for the compression set*. $\mathcal{I}_\mathbf{i}$ denotes the set of vectors of indices where each index is not present in $\mathbf{i}$. For any training sample $\mathbf{z}^m$ and a compression set $\mathbf{z_i}$, $\mathbf{R}_{\mathbf{z}_{\bar{\mathbf{i}}}}(f)$ represents the vector of indices pointing to the examples in $\mathbf{z}_{\bar{\mathbf{i}}}$ that are misclassified by f. Hence,

$$\mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}\left(R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \epsilon)\right) = \sum_{\mathbf{j}\in\mathcal{I}_\mathbf{i}} \mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}\left(R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \epsilon, \mathbf{R}_{\mathbf{Z}_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma, \mathbf{Z_i})) = \mathbf{j}\right) \quad (3.19)$$

Whenever $(\sigma, \mathbf{Z_i})$ is fixed, the classifier $\mathcal{R}(\sigma, \mathbf{Z_i})$ is fixed too. Also since each $\mathbf{Z}_i$ is i.i.d. according to the same (but unknown) distribution D, we have:

$$\mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}\left(R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \epsilon, \mathbf{R}_{\mathbf{Z}_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma, \mathbf{Z_i})) = \mathbf{j}\right) \leq (1-\epsilon)^{m-|\mathbf{i}|-|\mathbf{j}|} \quad (3.20)$$

Hence, the Equations 3.18, 3.19, and 3.20 yield:

$$\begin{aligned} P' &\leq \sum_{\mathbf{i}\in\mathcal{I}} \mathbf{E}_{\mathbf{Z_i}} \sum_{\sigma\in\mathcal{M}(\mathbf{Z_i})} \sum_{\mathbf{j}\in\mathcal{I}_\mathbf{i}} (1-\epsilon)^{m-|\mathbf{i}|-|\mathbf{j}|} \quad (3.21) \\ &= \sum_{\mathbf{i}\in\mathcal{I}} \sum_{\mathbf{j}\in\mathcal{I}_\mathbf{i}} \mathbf{E}_{\mathbf{Z_i}} \left|\mathcal{M}(\mathbf{Z_i})\right| (1-\epsilon)^{m-|\mathbf{i}|-|\mathbf{j}|} \quad (3.22) \end{aligned}$$

where $|\mathcal{M}(\mathbf{Z_i})| \geq 1$ is the maximum number of distinct messages that can be attached to compression set $\mathbf{Z_i}$. In deriving Equation 3.22 from Equation 3.21, the risk bound ϵ is allowed to depend on the compression set $\mathbf{Z_i}$ and the number of errors $|\mathbf{j}|$ made on the examples not in $\mathbf{Z_i}$. Let us first investigate the case where the bound ϵ does not depend on the message $\sigma \in \mathcal{M}(\mathbf{Z_i})$. Hence, in order that the r.h.s. of Equation 3.22 is at most δ, we need to find a suitable $\epsilon(\mathbf{Z_i}, |\mathbf{j}|)$. One possibility simply consists at solving:

$$\binom{m}{|\mathbf{i}|}\binom{m-|\mathbf{i}|}{|\mathbf{j}|} \left|\mathcal{M}(\mathbf{Z_i})\right| (1-\epsilon(\mathbf{Z_i}, |\mathbf{j}|))^{m-|\mathbf{i}|-|\mathbf{j}|} = \zeta(|\mathbf{i}|) \cdot \zeta(|\mathbf{j}|) \cdot \delta \quad (3.23)$$

where $\zeta()$ is given by Equation 3.11.
The value of $\epsilon(\mathbf{Z_i}, |\mathbf{j}|)$ satisfying Equation 3.23 is given by:

$$\epsilon(\mathbf{Z_i}, |\mathbf{j}|) = 1 - \exp\left(\frac{-1}{m - |\mathbf{i}| - |\mathbf{j}|}\left[\ln\binom{m}{|\mathbf{i}|} + \ln\binom{m - |\mathbf{i}|}{|\mathbf{j}|} + \ln\left(\frac{|\mathcal{M}(\mathbf{Z_i})|}{\zeta(|\mathbf{i}|)\zeta(|\mathbf{j}|)\delta}\right)\right]\right) \quad (3.24)$$

Consequently, we have proven the following theorem:

Theorem 10. *For any sample compression learning algorithm with a reconstruction function $\mathcal{R}$ that maps arbitrary subsets of a training set and information messages to classifiers:*

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall \mathbf{i} \in \mathcal{I}, \sigma \in \mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) \leq \epsilon(\mathbf{Z_i}, |\mathbf{j}|)\right\} \geq 1 - \delta$$

The risk bound given by Theorem 10 *does not depend* on the amount of training errors made by the classifier on the compression set. However, if the reconstruction function is constrained to be consistent with the compression set, less additional information will be needed to reconstruct the classifier. Hence, the above risk bound will generally be smaller for sample-compression learning algorithms that always return a classifier consistent with the compression set. On the other hand, however, this constraint might force the learner to output classifiers with larger compression sets.

Also note that we recover the result of Theorem 9 when the compression set $\mathbf{Z_i}$ vanishes. That is, the bound turns into an Occam's Razor bound when the classifier (represented by $A(\mathbf{Z}^m)$ in Theorem 9 unlike in terms of the reconstruction function here) can be represented solely by the distribution of messages ($P_{\mathcal{H}}(h)$ in Theorem 9). In the above theorem, note that the quantity $\frac{1}{|\mathcal{M}(\mathbf{Z_i})|}$ is equivalent to the distribution of messages $P_{\mathcal{H}}(h)$ in Theorem 9. Moreover, we use k to represent the number of errors in Theorem 9 as opposed to $|\mathbf{j}|$ here.

The risk bound ϵ can also depend on the information message σ by making explicit the dependence of ϵ on $\mathbf{Z_i}$, σ, and $\mathbf{j}$ and incorporating these in Equation 3.21:

$$P' \leq \sum_{\mathbf{i}\in\mathcal{I}}\sum_{\mathbf{j}\in\mathcal{I}_\mathbf{i}} \mathbf{E}_{\mathbf{Z_i}} \sum_{\sigma\in\mathcal{M}(\mathbf{Z_i})} [1 - \epsilon(\sigma, \mathbf{Z_i}, \mathbf{j})]^{m-|\mathbf{i}|-|\mathbf{j}|} \quad (3.25)$$

For any compression set $\mathbf{z_i}$, we use any function $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$ having the following property:

$$\sum_{\sigma\in\mathcal{M}(\mathbf{z_i})} P_{\mathcal{M}(\mathbf{z_i})}(\sigma) \leq 1 \quad (3.26)$$

Hence, the function $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$ can be interpreted to have the compression-set-dependent distribution of messages when it sums to one. Let ϵ be such that:

$$\binom{m}{|\mathbf{i}|}\binom{m-|\mathbf{i}|}{|\mathbf{j}|}\left[1-\epsilon(\sigma,\mathbf{Z_i},|\mathbf{j}|)\right]^{m-|\mathbf{i}|-|\mathbf{j}|} = P_{\mathcal{M}(\mathbf{Z_i})}(\sigma)\cdot\zeta(|\mathbf{i}|)\cdot\zeta(|\mathbf{j}|)\cdot\delta \tag{3.27}$$

where ζ is defined by Equation 3.11. This gives:

$$\begin{aligned}\epsilon(\sigma,\mathbf{Z_i},|\mathbf{j}|) \;=\; 1-\exp\left(\frac{-1}{m-|\mathbf{i}|-|\mathbf{j}|}\left[\ln\binom{m}{|\mathbf{i}|}+\ln\binom{m-|\mathbf{i}|}{|\mathbf{j}|}+\ln\left(\frac{1}{P_{\mathcal{M}(\mathbf{Z_i})}(\sigma)}\right)+\right.\right.\\ \left.\left.\ln\left(\frac{1}{\zeta(|\mathbf{i}|)\zeta(|\mathbf{j}|)\delta}\right)\right]\right)\end{aligned} \tag{3.28}$$

Therefore, again, $P' \leq \delta$. Hence, We have the following theorem:

Theorem 11. (Sample Compression Theorem) *For any sample compression learning algorithm with a reconstruction function $\mathcal{R}$ that maps arbitrary subsets of a training set and information messages to classifiers:*

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall\mathbf{i}\in\mathcal{I},\sigma\in\mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma,\mathbf{Z_i}))\leq\epsilon(\sigma,\mathbf{Z_i},|\mathbf{j}|)\right\}\geq 1-\delta$$

The Theorem 10 is basically a corollary of Theorem 11 for the case $P_{\mathcal{M}(\mathbf{Z_i})}(\sigma) = |\mathcal{M}(\mathbf{Z_i})|^{-1}\ \forall\sigma\in\mathcal{M}(\mathbf{Z_i})$.

3.8 PAC-Bayes Bound

The PAC-Bayes approach was initiated by McAllester (1999). It aims at providing PAC guarantees to "Bayesian" learning algorithms.

As we know, Bayesian algorithms are generally specified in terms of a *prior distribution* P over a space of classifiers and a *posterior distribution* Q (over the same space of classifiers). The prior distribution characterizes our prior belief about good classifiers (before the observation of the data). On the other hand the posterior distribution takes into account the additional information provided by the training data. A remarkable result that came out from this line of research, known as the "PAC-Bayes theorem", provides a tight upper bound on the risk of a stochastic classifier called the *Gibbs classifier*.

Given an input example $\mathbf{x}$, the label $G_Q(\mathbf{x})$ assigned to $\mathbf{x}$ by the Gibbs classifier is defined by the following process. We first choose a classifier h according to the posterior distribution Q and then use h to assign the label $h(\mathbf{x})$ to $\mathbf{x}$. The risk of G_Q is defined as the expected risk of classifiers drawn according to Q:

$$R(G_Q)\stackrel{\text{def}}{=}\mathbf{E}_{h\sim Q}R(h)=\mathbf{E}_{h\sim Q}\mathbf{E}_{(\mathbf{x},y)\sim D}I(f(\mathbf{x})\neq y)$$

The PAC-Bayes theorem was first proposed by McAllester (2003). The version presented here is due to Seeger (2002) and Langford (2005).

Theorem 12. (PAC-Bayes Theorem) *Given any space $\mathcal{H}$ of classifiers. For any data-independent prior distribution P over $\mathcal{H}$ and for any (possibly data-dependent) posterior distribution Q over $\mathcal{H}$, with probability at least $1-\delta$ over the random draws of training sets S of m examples:*

$$\mathrm{kl}(R_S(G_Q)\|R(G_Q)) \leq \frac{\mathrm{KL}(Q\|P) + \ln \frac{m+1}{\delta}}{m}$$

where $\mathrm{KL}(Q\|P)$ *is the Kullback-Leibler divergence between distributions Q and P:*

$$\mathrm{KL}(Q\|P) \stackrel{\mathrm{def}}{=} \mathbf{E}_{h\sim Q} \ln \frac{Q(h)}{P(h)}$$

and where $\mathrm{kl}(q\|p)$ *is the Kullback-Leibler divergence between the Bernoulli distributions with probabilities of success q and p ($p, q \in [0,1]$):*

$$\mathrm{kl}(q\|p) \stackrel{\mathrm{def}}{=} q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p}.$$

We, however, do not present the proof for the above bound. Please refer to (McAllester, 2003, Seeger, 2002, Langford, 2005) for details of the proof.

The bound given by the PAC-Bayes theorem for the risk of Gibbs classifiers can be turned into a bound for the risk of Bayes classifiers in the following way.

Given a posterior distribution Q, the Bayes classifier B_Q performs a majority vote (under measure Q) of binary classifiers in $\mathcal{H}$. When B_Q misclassifies an example $\mathbf{x}$, at least half of the binary classifiers (under measure Q) misclassifies $\mathbf{x}$. It follows that the error rate of G_Q is at least half of the error rate of B_Q. Hence $R(B_Q) \leq 2R(G_Q)$.

3.9 Concluding Remarks

In this chapter we have provided an introduction to statistical learning theory focusing on the mathematical learning models. We have presented uniform risk bounds for the generalization ability of learning algorithms in various models. We have tried to present these bounds in a general form where the learner is allowed to make some error on training data if better generalization can be obtained. These bounds form the basis of the theoretical analysis of various approaches in our work. Most of the bounds are derived from the generic bound structure presented in this chapter.

3.10 Bibliographical Notes

In addition to the references cited in the text of the PAC learning section, details on the criticisms of the PAC model and the approaches to address these criticisms can be found in (Buntine, 1990, Sarrett and Pazzani, 1992, Amsterdam, 1988, Bergadano and Saitta, 1989, Venkatesh, 1991). Refer to Kearns & Vazirani (1994) and Haussler (1992) for a detailed discussion of PAC learning. A Bayesian extension of the PAC model can be found in (Buntine, 1990). Haussler (1992) also discusses these issues in further detail. Also see the work by Dhagat and Hellerstein (1994) on PAC learning with irrelevant attributes for learning decision lists. Also see (Rivest, 1987). Another approach on obtaining a PAC-style bound on the generalization error of learning algorithms using their stability properties can be found in (Bousquet and Elisseeff, 2001). Pitt and Warmuth (1990) discuss the methods for determining the PAC learnability of concepts classes. See (Vapnik and Chervonenkis, 1971, Blumer et. al., 1989, Floyd and Warmuth, 1995, Vapnik, 1998) for details on VC dimension. See (Burges, 1998) for a study of the VC dimension of SVMs by computing VC dimension for homogenous polynomial and gaussian radial basis function kernels. Also see (Herbrich, 2002, Sontag , 1998) for survey on VC dimension.

Strong results have been obtained for Occam learning. See, for example, (Haussler, 1988) for an algorithm for learning conjunction with few relevant literals. The topic of learning in presence of many irrelevant variables has been explored by Littlestone (1988, 1989) and Blum (1992). Rivest (1987) describes an algorithm to PAC learn decision lists using Occam's Razor approach.

For further relationships between the PAC and Occam algorithms, see the work by Schapire (1990, 1992), Freund (1990, 1992) and Helmbold and Warmuth (1992). Other variants of the PAC model using the Occam approach are explored by: Angluin and Laird (1988), Kearns and Li (1993), Kearns (1990) (learning in presence of errors); Kearns and Schapire (1990) Schapire (1992) (learning probabilistic concepts); Natarajan (1993) (function learning). Evans, Rajagopalan and Vazirani (1993) give a Bayesian interpretation to the Minimum Description Length (MDL) principle using a generalized version of Occam algorithm. Another representation independent version of Occam's Razor theorem making use of Kolmogorov Complexity can be found in the paper by Li, Tromp and Vitányi (2003).

Some of the recent textbooks and surveys on statistical learning theory include: Herbrich (2002), Schölkopf and Smola (2002), Mendelson (2003, 2004), Bousquet, Boucheron and Lugosi (2004). For some interesting results in the field of sample compression, see (Marchand and Shawe-Taylor, 2001, 2002) and (Marchand et. al., 2003) for the generalization bounds

for SCM with *data-dependent* features. We present somewhat different versions of these bounds in the following chapters. See also (Floyd and Warmuth, 1995) and (Ben-David and Litman, 1998). Refer Langford (2005) for a tutorial on prediction theory and variants of various generalization error bounds based on the binomial tail inversion.

Interesting results have been obtained for PAC-Bayes bounds under various settings. See for instance (Graepel, Herbrich, and Shawe-Taylor, 2005) for PAC-Bayes compression bounds on linear classifiers. Marchand and Shah (2005) obtains PAC-Bayes bounds on conjunction (or disjunction) of linear threshold features on individual attributes. Also see (Laviolette and Marchand, 2005) for PAC-Bayes bounds on sample compressed Gibbs classifiers.

CHAPTER 4

The Set Covering Machine

This chapter presents an overview of the original Set Covering Machine algorithm proposed by Marchand and Shawe-Taylor (2001, 2002), along with the initially proposed set of features called data-dependent balls. The SCM framework forms the basis of our work in future chapters

4.1 Introduction

The algorithms and results that we present throughout this work use the generic Set Covering Machine framework as a basis. The Set Covering Machine (SCM) algorithm was proposed by Marchand and Shawe-Taylor (2001, 2002). In this chapter we provide a brief description of the generic algorithm along with a set of features called data-dependent balls introduced by Marchand and Shawe-Taylor (2001). We will also give a generalization error bound for the SCM with this set of features.

The SCM algorithm was motivated originally by the idea of learning a conjunction (or disjunction) of monomials via the Standard Monomial Learning algorithm proposed by Valiant (1984). The idea stemmed from the Combinatorial optimization approach used by Haussler (1988) to deal with the problem where it was shown that this problem could be reduced to the Minimum Set Cover problem which, although NP-hard, has a good worst-case lower bound for greedy heuristic (Chvátal, 1979). Motivated by this observation Marchand and Shawe-Taylor (2001, 2002) generalized this algorithm for learning conjunctions (or disjunctions) of Boolean attributes to the case of learning these functions over arbitrary sets of Boolean valued features. Moreover, this approach was extended so as to include the features that are *data-dependent*, i.e. constructed from data. Also, the algorithm provides some learning parameters to control the tradeoff between the accuracy and the size of the conjunction (or disjunction) so as to deal with the problems of noisy data and overfitting.

4.2 Formalization of the SCM Algorithm

Let $\mathbf{x}$ denote an arbitrary n-dimensional vector of the input space $\mathcal{X}$ which could be arbitrary subsets of $\mathbb{R}^n$. We consider binary classification problems for which the training set $S = P \cup N$ consists of a set P of positive training examples and a set N of negative training examples. A *feature* is defined as an arbitrary Boolean-valued function that maps $\mathcal{X}$ onto $\{0, 1\}$.

Let $\digamma = \{h_i(\mathbf{x})\}_{i=1}^{|\digamma|}$ be any set of features $h_i(\mathbf{x})$. The learning algorithm when given any such set $\digamma$ return a small subset $\mathcal{R} \subset \digamma$ of features. Given this subset $\mathcal{R}$ and an arbitrary input vector $\mathbf{x}$, the output $f(\mathbf{x})$ of the SCM is defined to be:

$$f(\mathbf{x}) = \begin{cases} \bigvee_{i \in \mathcal{R}} h_i(\mathbf{x}) & \text{for a disjunction} \\ \bigwedge_{i \in \mathcal{R}} h_i(\mathbf{x}) & \text{for a conjunction} \end{cases}$$

We will follow the following definition for *consistency* given by Marchand and Shawe-Taylor (2002) throughout this thesis:

Definition 13. *A function (or a feature) is said to be consistent with an example if it correctly classifies that example. Similarly, a function (or a feature) is said to be consistent with a set of examples if it correctly classifies all the examples in that set.*

To discuss both the conjunction and the disjunction cases simultaneously, let us use $\mathcal{P}$ to denote the set P in the conjunction case but the set N in the disjunction case. Similarly, $\mathcal{N}$ denotes the set N in the conjunction case but denotes the set P in the disjunction case. It then follows that f makes no error on $\mathcal{P}$ iff each $h_i \in \mathcal{R}$ makes no error on $\mathcal{P}$. Moreover, if Q_i denotes the subset of examples of $\mathcal{N}$ on which feature h_i makes no errors, then f makes no error on $\mathcal{N}$ if and only if $\bigcup_{i \in \mathcal{R}} Q_i = \mathcal{N}$. Hence, as was first observed by Haussler (1988), the problem of finding the smallest set $\mathcal{R}$ for which f makes no training errors is just the problem of finding the smallest collection of Q_is that cover all $\mathcal{N}$ (where each corresponding h_i makes no error on $\mathcal{P}$). This is the well-known *Minimum Set Cover Problem* (Garey & Johnson, 1979). The interesting fact is that, although it is NP-complete to find the smallest cover, the *set covering greedy algorithm* will always find a cover of size at most $z \ln(|\mathcal{N}|)$ when the smallest cover that exists is of size z (Chvátal, 1979, Kearns & Vazirani, 1994). Moreover this algorithm is very simple to implement and just consists of the following steps: first choose the set Q_i which covers the largest number of elements in $\mathcal{N}$, remove from $\mathcal{N}$ and each Q_j the elements that are in Q_i, then repeat this process of finding the set Q_k of largest cardinality and updating $\mathcal{N}$ and each Q_j until there are no more elements in $\mathcal{N}$.

The SCM built on the features found by the set covering greedy algorithm will make no training errors only when there exists a subset $\mathcal{E} \subset \digamma$ of features on which a conjunction (or

a disjunction) makes zero training error. However, this constraint is not really required in practice since we do want to permit the user of a learning algorithm to control the tradeoff between the accuracy achieved on the training data and the complexity (here the size) of the classifier. Indeed, a small SCM which makes a few errors on the training set might give better generalization than a larger SCM (with more features) which makes zero training errors. One way to include this flexibility into the SCM is to stop the set covering greedy algorithm when there remains a few more training examples to be covered. In this case, the SCM will contain fewer features and will make errors on those training examples that are not covered. But these examples all belong to $\mathcal{N}$ and, in general, we do need to be able to make errors on training examples of both classes. Hence, early stopping is generally not sufficient and, in addition, we need to consider features that also make some errors with $\mathcal{P}$ provided that many more examples in $\mathcal{N}$ can be covered. Hence, for a feature h, let us denote by Q_h the set of examples in $\mathcal{N}$ covered by feature h and by R_h the set of examples in $\mathcal{P}$ on which h makes an error. Given that each example in $\mathcal{P}$ misclassified by h should decrease by some fixed *penalty* p its "importance", the *usefulness* U_h of feature h is defined by the following equation:

$$U_h \stackrel{\text{def}}{=} |Q_h| - p \cdot |R_h|$$

Hence, the set covering greedy algorithm is modified in the following way. Instead of using the feature that covers the largest number of examples in $\mathcal{N}$, the feature $h \in \mathcal{F}$ that has the highest usefulness value U_h is used. We remove from $\mathcal{N}$ and each Q_g (for $g \neq h$) the elements that are in Q_h and we remove from each R_g (for $g \neq h$) the elements that are in R_h. Note that we update each such set R_g because a feature g that makes an error on an example in $\mathcal{P}$ does not increase the error of the machine if another feature h is already making an error on that example. We repeat this process of finding the feature h of largest usefulness U_h and updating $\mathcal{N}$, and each Q_g and R_g, until only a few elements remain in $\mathcal{N}$ (early stopping the greedy).

4.3 Data-dependent Balls

Marchand and Shawe-Taylor (2001, 2002) gave an implementation of the SCM algorithm with a set of features they called *data-dependent balls* and proposed a risk bound for SCM with this set of features. For the case of *data-dependent balls*, each feature is identified by a training example, called a *center* $(\mathbf{x}_c, y_c)$, and a radius ρ. Given any metric d, the output

$h(\mathbf{x})$ on any input example $\mathbf{x}$ of such a feature is given by:

$$h(\mathbf{x}) = \begin{cases} y_c & \text{if } d(\mathbf{x}, \mathbf{x}_c) \leq \rho \\ -y_c & \text{otherwise} \end{cases}$$

Marchand and Shawe-Taylor (2002) have proposed to use another training example $\mathbf{x}_b$, called a *border point*, to code for the radius so that $\rho = d(\mathbf{x}_c, \mathbf{x}_b)$.

4.3.1 Generalization Error Bound

Let us bound the generalization error of SCM with data-dependent balls. Note that data-dependent sample compression based bounds probably best reflect the nature of the learning algorithm in this case as opposed to data-independent bounds such as the VC bounds. Hence, we use the generic message dependent sample compression risk bound of Chapter 3 to bound the generalization performance of the SCM algorithm.

Given a compression set $\mathbf{z_i}$ of size $|\mathbf{i}|$, if the classifier makes $|\mathbf{j}|$ errors on the training set S of m examples, the sample compression theorem (Theorem 11) states the following:

Sample Compression Theorem. *For any sample compression learning algorithm with a reconstruction function $\mathcal{R}$ that maps arbitrary subsets of a training set and information messages to classifiers:*

$$\mathbf{P}_{\mathbf{Z}^m} \{\forall \mathbf{i} \in \mathcal{I}, \sigma \in \mathcal{M}(\mathbf{Z_i}) \colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) \leq \epsilon(\sigma, \mathbf{Z_i}, |\mathbf{j}|)\} \geq 1 - \delta$$

where

$$\epsilon(\sigma, \mathbf{Z_i}, |\mathbf{j}|) = 1 - \exp\left(\frac{-1}{m - |\mathbf{i}| - |\mathbf{j}|}\left[\ln\binom{m}{|\mathbf{i}|} + \ln\binom{m - |\mathbf{i}|}{|\mathbf{j}|} + \ln\left(\frac{1}{P_{\mathcal{M}(\mathbf{Z_i})}(\sigma)}\right) + \ln\left(\frac{1}{\zeta(|\mathbf{i}|)\zeta(|\mathbf{j}|)\delta}\right)\right]\right)$$

and

$$\zeta(a) \stackrel{\text{def}}{=} \frac{6}{\pi^2}(a+1)^{-2}$$

In the case of data-dependent balls the compression set consists of examples denoting the centers and borders of the balls. Now, given a compression set $\mathbf{z_i}$, we need to specify the examples in $\mathbf{z_i}$ that are used for a border point without being used as a center. As explained by Marchand and Shawe-Taylor (2002), no additional amount of information is required to pair each center with its border point whenever the reconstruction function $\mathcal{R}$ is constrained to produce a classifier that always correctly classifies the compression set. Furthermore, as

argued by Marchand and Shawe-Taylor (2002), we can limit ourselves to the case where each border point is a positive example. In that case, each message $\sigma \in \mathcal{M}(\mathbf{z_i})$ just needs to specify the positive examples that are a border point without being a center. Let $n(\mathbf{z_i})$ and $p(\mathbf{z_i})$ be, respectively, the number of negative and the number of positive examples in compression set $\mathbf{z_i}$. Let $b(\sigma)$ be the number of border point examples specified in message σ and let $\zeta(a)$ be defined by Equation 3.11. We can then use:

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \zeta(b(\sigma)) \cdot \binom{p(\mathbf{z_i})}{b(\sigma)}^{-1}$$

since, in that case, we have for any compression set $\mathbf{z_i}$:

$$\sum_{\sigma \in \mathcal{M}(\mathbf{z_i})} P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \sum_{b=0}^{p(\mathbf{z_i})} \zeta(b) \sum_{\sigma : b(\sigma)=b} \binom{p(\mathbf{z_i})}{b(\sigma)}^{-1} \leq 1$$

This distribution of messages along with the *Sample Compression Theorem* provides a bound on generalization error. Note that by $\binom{a}{b}^{-1}$, we mean $\frac{1}{\binom{a}{b}}$.

4.4 The SCM Algorithm: Formal Outline

We conclude this chapter with the formal description of the SCM learning algorithm. The penalty p and the early stopping point s are the two model-selection parameters that give the user the ability to control the proper tradeoff between the training accuracy and the size of the function. Their values could be determined either by using k-fold cross-validation, or by computing the bound on the generalization error based on what has been achieved on the training data. See (Marchand and Shawe-Taylor, 2001, 2002) for the implementation of SCM with data-dependent features called generalized balls and the results of model selection using the generalization bound obtained with respect to this set of features. We propose an alternate set of features and generalization error bound in Chapter 5. Note that the learning algorithm reduces to the two-step algorithm of Valiant (1984) and Haussler (1988) when both s and p are infinite and when the set of features consists of the set of input attributes and their negations.

The pseudocode on the next page gives the outline of the SCM algorithm of any arbitrary set of, possibly data-dependent, features. The algorithm is called **BuildSCM** and we will be referring to this base learning algorithm in the coming chapters by this name.

Algorithm BuildSCM$(T, P, N, p, s, \mathcal{F})$

Input: A machine type T (which is either "conjunction" or "disjunction"), a set P of positive training examples, a set N of negative training examples, a penalty value p, a stopping point s, and a set $\mathcal{F} = \{h_i(\mathbf{x})\}_{i=1}^{|\mathcal{F}|}$ of Boolean-valued features.

Output: A conjunction (or disjunction) $f(\mathbf{x})$ of a subset $\mathcal{R} \subseteq \mathcal{F}$ of features.

Initialization: $\mathcal{R} = \emptyset$.

1. **If** ($T =$ "conjunction") let $\mathcal{P} \leftarrow P$ and $\mathcal{N} \leftarrow N$.
   Else let $\mathcal{P} \leftarrow N$ and let $\mathcal{N} \leftarrow P$.

2. For each $h_i \in \mathcal{F}$, let Q_i be the subset of $\mathcal{N}$ covered by h_i and let R_i be the subset of $\mathcal{P}$ covered by h_i (*i.e.* examples in $\mathcal{P}$ incorrectly classified by h_i).

3. Let h_k be a feature with the largest value of $|Q_k| - p \cdot |R_k|$. **If** ($|Q_k| = 0$) then go to step 7 (cannot cover the remaining examples in $\mathcal{N}$).

4. Let $\mathcal{R} \leftarrow \mathcal{R} \cup \{h_k\}$. Let $\mathcal{N} \leftarrow \mathcal{N} - Q_k$ and let $\mathcal{P} \leftarrow \mathcal{P} - R_k$.

5. For all i do: $Q_i \leftarrow Q_i - Q_k$ and $R_i \leftarrow R_i - R_k$.

6. **If** ($\mathcal{N} = \emptyset$ or $|\mathcal{R}| \geq s$) then go to step 7 (no more examples to cover or early stopping).
   Else go to step 3.

7. Return $f(\mathbf{x})$ where:

$$f(\mathbf{x}) = \begin{cases} \bigvee_{i \in \mathcal{R}} h_i(\mathbf{x}) & \text{for a disjunction} \\ \bigwedge_{i \in \mathcal{R}} h_i(\mathbf{x}) & \text{for a conjunction} \end{cases}$$

CHAPTER 5

The SCM with Data-Dependent Halfspaces

In this chapter we present an alternate set of features for the Set Covering Machine called the data-dependent Half-spaces and also present a tight sample compression risk bound. We show that this set of features is indeed sometimes better than data-dependent balls and yields sparser classifiers with better classification accuracies. These results in part appeared in:

M. Marchand, M. Shah, J. Shawe-Taylor and M. Sokolova. The Set Covering Machine with Data-dependent Half-Spaces. *Proceedings of the Twentieth International Conference on Machine Learning* (ICML'2003), 520–527, Morgan Kaufmann, San Fransisco, CA, 2003.

5.1 Introduction

We discussed the sample compression based SCM framework in Chapter 4 along with the initial set of features called data-dependent balls proposed by Marchand and Shawe-Taylor (2001, 2002). The SCM with data-dependent balls showed encouraging results especially in terms of sparsity. The SCM aims at maximizing sparsity (i.e. using minimum possible number of examples to represent the classifier) in contrast to the Support Vector Machine (SVM) that aims at maximizing the (soft-) margin of the separating hyperplane (in the feature space). Hence, the SCM can be the algorithm of choice when the objective is to find a sparse classifier. Recall that for the set of features known as *data-dependent balls*, Marchand and Shawe-Taylor (2001, 2002) have shown that good generalization is expected when a SCM with a small number of balls and errors can be found on the training data. Furthermore, on some real world data sets, they have found that the SCM achieves a much higher level of sparsity than an SVM with roughly the same generalization error.

The question however is whether the SCM framework is general enough. Can we move beyond the data-dependent balls as the features? Can an alternate set of features provide yet sparser and general solutions? This is precisely our focus in this chapter.

In this chapter, we introduce a new set of features for the SCM that we call *data-dependent half-spaces*. Since our goal is to construct sparse classifiers, we want to avoid using $O(d)$ examples to construct each half-space in a d-dimensional input space (like many

computational geometric algorithms). Rather, we want to use $O(1)$ examples for each half-space. In fact, we will see that by using only three examples per half-space, we need very few of these half-spaces to achieve a generalization as good as (and sometimes better than) the SVM on many "natural" data sets. Moreover, the level of sparsity achieved by the SCM is always substantially superior (sometimes by a factor of at least 50) than the one achieved by the SVM.

Finally, by extending the sample compression technique of Littlestone and Warmuth (1986), we bound the generalization error of the SCM with data-dependent half-spaces in terms of the number of errors and the number of half-spaces it achieves on the training data. Throughout the chapter, we use the term **BuildSCM** to denote the general SCM algorithm introduced in Chapter 4. We also adhere to the same notation for the various parameters such as the penalty and size and the utility function, as in Chapter 4.

5.2 Data-Dependent Half-Spaces

With the use of kernels, each input vector $\mathbf{x}$ is implicitly mapped into a high-dimensional vector $\boldsymbol{\phi}(\mathbf{x})$ such that $\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}') = k(\mathbf{x}, \mathbf{x}')$ (the kernel trick). We consider the case where each feature is a half-space constructed from a set of 3 points $\{\boldsymbol{\phi}_a, \boldsymbol{\phi}_b, \boldsymbol{\phi}_c\}$ where $\boldsymbol{\phi}_a$ is the image of a positive example $\mathbf{x}_a$, $\boldsymbol{\phi}_b$ is the image of a negative example $\mathbf{x}_b$, and $\boldsymbol{\phi}_c$ is the image of a $\mathcal{P}$-example $\mathbf{x}_c$. We described the $\mathcal{P}$-examples in Chapter 4. Simply stated, $\mathcal{P}$ denotes the set P in the conjunction case while it denotes the set N in the disjunction case. The weight vector $\mathbf{w}$ of such an half-space $h^c_{a,b}$ is defined by $\mathbf{w} \stackrel{\text{def}}{=} \boldsymbol{\phi}_a - \boldsymbol{\phi}_b$ and its threshold t is identified by $t \stackrel{\text{def}}{=} \mathbf{w} \cdot \boldsymbol{\phi}_c - \epsilon$, where ϵ is a small positive real number in the case of a conjunction but a small negative number in the case of a disjunction. Hence,

$$\begin{aligned} h^c_{a,b}(\mathbf{x}) &\stackrel{\text{def}}{=} \operatorname{sgn}\{\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) - t\} \\ &= \operatorname{sgn}\{k(\mathbf{x}_a, \mathbf{x}) - k(\mathbf{x}_b, \mathbf{x}) - t\} \end{aligned}$$

where

$$t = k(\mathbf{x}_a, \mathbf{x}_c) - k(\mathbf{x}_b, \mathbf{x}_c) - \epsilon.$$

When the penalty parameter p is set to ∞, **BuildSCM** tries to cover with half-spaces the examples of $\mathcal{N}$ without making any error on the examples of $\mathcal{P}$. In that case, $\boldsymbol{\phi}_c$ is the image of the example $\mathbf{x}_c \in \mathcal{P}$ that gives the smallest value of $\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_c)$ in the case of a conjunction (but the largest value of $\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_c)$ in the case of a disjunction). Note that, in contrast with

data-dependent balls (Marchand and Shawe-Taylor, 2002), we are not guaranteed to always be able to cover all $\mathcal{N}$ with such half-spaces. When training a SCM with finite p, any $x_c \in \mathcal{P}$ might give the best threshold for a given $(\mathbf{x}_a, \mathbf{x}_b)$ pair. Hence, to find the half-space that maximizes the utility function U_h (described in Chapter 4), we need to compute U_h for every triple $(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$.

Note that this set of features (in the linear kernel case $k(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$) was already proposed by Hinton & Revow (1996) for decision tree learning but no analysis of their learning method has been given.

5.3 Bound on the Generalization Error

We now bound the generalization performance of the SCM algorithm when it uses the data-dependent halfspaces as the features. We derive a risk bound that depends on the number of examples in the *compression set* and the size of the information message needed to reconstruct the final classifier from this compression set. Our bound is built on the generic sample compression bound that we obtained in Section 3.7 (Theorem 11). We re-state the generic bound first and then adapt it to the case of SCM with data-dependent Half-spaces:

Sample Compression Theorem. *For any sample compression learning algorithm with a reconstruction function $\mathcal{R}$ that maps arbitrary subsets of a training set and information messages to classifiers:*

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall \mathbf{i} \in \mathcal{I}, \sigma \in \mathcal{M}(\mathbf{Z}_\mathbf{i})\colon R(\mathcal{R}(\sigma, \mathbf{Z}_\mathbf{i})) \leq \epsilon(\sigma, \mathbf{Z}_\mathbf{i}, |\mathbf{j}|)\right\} \geq 1 - \delta$$

where

$$\epsilon(\sigma, \mathbf{Z}_\mathbf{i}, |\mathbf{j}|) = 1 - \exp\left(\frac{-1}{m - |\mathbf{i}| - |\mathbf{j}|}\left[\ln\binom{m}{|\mathbf{i}|} + \ln\binom{m - |\mathbf{i}|}{|\mathbf{j}|} + \ln\left(\frac{1}{P_{\mathcal{M}(\mathbf{Z}_\mathbf{i})}(\sigma)}\right) + \ln\left(\frac{1}{\zeta(|\mathbf{i}|)\zeta(|\mathbf{j}|)\delta}\right)\right]\right) \tag{5.1}$$

and ζ is defined by Equation 3.11.

Now, we need to specify the distribution of messages for the Half-spaces. This is the term $P_{\mathcal{M}(\mathbf{Z}_\mathbf{i})}(\sigma)$ in Equation 5.1.

5.3.1 Bound for SCM with Halfspaces

In order to obtain the distribution of messages $P_{\mathcal{M}(\mathbf{z}_\mathbf{i})}(\sigma)$ for data-dependent halfspaces, we need to construct a message string σ that identifies, from $\mathbf{z}_\mathbf{i}$, the weight vector and threshold

of each half-space. We know that each weight vector $\mathbf{w}$ is specified by a pair $(\mathbf{x}_a, \mathbf{x}_b)$ of examples, taken from $\mathbf{z_i}$. Note that the examples $\mathbf{x}_a$ and $\mathbf{x}_b$ having opposite class labels.

We denote by $p(\mathbf{z_i})$ and $n(\mathbf{z_i})$, the number of positive and the number of negative examples in the compression set $\mathbf{z_i}$ respectively. The set of positive examples in the compression set is denoted by $P(\mathbf{z_i})$ and the set of negative examples in the compression set is denoted by $N(\mathbf{z_i})$. Now, each example in the compression set $\mathbf{z_i}$ can participate in multiple weight vectors. However, under the constraint that each example in $\mathbf{z_i}$ is correctly classified, each pair $(\mathbf{x}_a, \mathbf{x}_b) \in P(\mathbf{z_i}) \times N(\mathbf{z_i})$ can provide at most one weight vector. Also, the pairs of examples identifying the weight vector cannot be used in reverse. That is, the pair $(\mathbf{x}_j, \mathbf{x}_i)$ cannot be used to represent a weight vector when the pair $(\mathbf{x}_i, \mathbf{x}_j)$ is already used. Hence, in order to identify $r(\mathbf{z_i})$ weight vectors we need two things: First, we need a string that specifies the number of weight vectors, i.e. $r(\mathbf{z_i})$ pairs among all the possible $p(\mathbf{z_i})n(\mathbf{z_i})$ pairs, and second, a string that each specifies the order of each pair $(\mathbf{x}_i, \mathbf{x}_j)$, if we use $\mathbf{w} = \boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j)$.

Let us denote by σ_1 the string that specifies the number of weight vectors $r(\mathbf{z_i})$ in the compression set $\mathbf{z_i}$. We assign a uniform probability over the number of weight vectors in the compression set. Hence, the string σ_1 can take a value between 0 and $p(\mathbf{z_i})n(\mathbf{z_i})$ with equal probability. Hence:

$$P_1(\sigma_1) = \frac{1}{p(\mathbf{z_i})n(\mathbf{z_i}) + 1} \quad \forall \sigma_1$$

Now, we denote by σ_2 the string used to specify each of the $r(\mathbf{z_i})$ weight vectors in accordance with our scheme. Again, we assign an equal probability over each possible choice of the pair of examples forming the weight vector. This yields:

$$P_2(\sigma_2|\sigma_1) = \binom{p(\mathbf{z_i})n(\mathbf{z_i})}{r(\mathbf{z_i})}^{-1} \quad \forall \sigma_2$$

We do not need any additional message string to identify the threshold point $\mathbf{x}_c \in \mathbf{z_i}$ for each weight vector $\mathbf{w}$. We can just perform the following procedure for the task: Let P and N denote the set of positive and negative examples in the compression set $\mathbf{z_i}$. We start with $P' = P, N' = N$ and repeat the following steps from the first half-space until we discover all the half-spaces. Let $\mathbf{w} = \boldsymbol{\phi}(\mathbf{x}_a) - \boldsymbol{\phi}(\mathbf{x}_b)$ be the weight vector of the current half-space. If $\mathbf{x}_a \in P$, then for the example identifying the threshold point $\mathbf{x}_c$, we choose $\mathbf{x}_c = \underset{\mathbf{x} \in N'}{\operatorname{argmax}}\ \mathbf{w} \cdot \mathbf{x}$ and we remove $\mathbf{x}_a$ from P' (to find the thresholds of the other weight vectors). Otherwise, if $\mathbf{x}_a \in N$, then, for the example identifying the threshold point $\mathbf{x}_c$, we choose $\mathbf{x}_c = \underset{\mathbf{x} \in P'}{\operatorname{argmax}}\ \mathbf{w} \cdot \mathbf{x}$ and we remove $\mathbf{x}_a$ from N'.

Hence, we get the following distribution for the message strings:

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \frac{1}{p(\mathbf{z_i})n(\mathbf{z_i}) + 1} \cdot \binom{p(\mathbf{z_i})n(\mathbf{z_i})}{r(\mathbf{z_i})}^{-1} \quad \forall \sigma \tag{5.2}$$

The bound over the SCM with data-dependent halfspaces can be obtained now by replacing the value of $P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$ in Equation 5.1 by the one obtained from Equation 5.2 and subsequently substituting Equation 5.1 in the *Sample Compression Theorem*.

5.4 Empirical Results on Natural data

We have compared the practical performance of the SCM with the Support Vector Machine (SVM) equipped with a Gaussian kernel (also called the Radial Basis Function kernel) of variance $1/\gamma$ and soft margin parameter C. Each SCM algorithm used the L_2 metric since this is the metric present in the argument of the RBF kernel.

The data sets used and the results obtained are reported in Tables 5.1 and 5.2. Each data set was randomly split in two parts. About half of the examples was used for training and the remaining set of examples was used for testing. The corresponding values for these numbers of examples are given in the "train" and "test" columns of each table. The learning parameters of all algorithms were determined from the training set *only*. The parameters C and γ for the SVM were determined by the 5-fold cross validation (CV) method performed on the training set. The parameters that gave the smallest 5-fold CV error were then used to train the SVM on the whole training set and the resulting classifier was then run on the testing set. Exactly the same method (with the same 5-fold split) was used to determine the learning parameters of both SCM with balls and SCM with half-spaces. These results are provided in Table 5.1 (SVM errs and SCM-cv, "errs" column under SCM Half-spaces (CV)) and Table 5.2. In addition to this, we have compared this 5-fold CV model selection method with a model selection method that uses the risk bound of the *Sample Compression Theorem* with Equation 5.2 to select the best SCM classifier obtained from the *same* possible choices of the learning parameters that we have used for the 5-fold CV method. The SCM that minimizes the risk bound (computed from the training set) was then run on the testing set. Similar model selection from bound is performed on SCM with balls (with message distribution given in Chapter 4. These results are provided in Table 5.1 (SCM-b and "errs" column under SCM Half-spaces (Bound)) and Table 5.2.

The "SVs" column of the SVM results refers to the number of support vectors present in the final classifier. The "errs" column, for all learning algorithms, refers to the number of

Table 5.1: Results of SVM and SCM with Balls on UCI Datasets.

Data Set			SVM results				SCM-cv		SCM-b	
Name	train	test	C	γ	SVs	errs	s	errs	s	errs
breastw	343	340	1	0.1	38	15	2	11	1	12
bupa	170	175	2	3.0	169	66	2	71	2	70
credit	353	300	100	0.25	282	51	12	65	1	57
glass	107	107	10	3.0	51	29	4	20	4	19
haberman	144	150	2	0.5	81	39	2	41	1	39
heart	150	147	1	3.0	64	26	1	28	1	23
pima	400	368	0.5	0.02	241	96	1	108	1	105
USvotes	235	200	1	0.02	53	13	8	26	3	19

Table 5.2: Results for SCM with Half-Spaces on UCI Datasets.

Data Set			SCM Half-spaces(CV)				SCM Half-spaces(Bound)			
Name	train	test	T	P	s	errs	T	P	s	errs
breastw	343	340	d	0.5	1	14	d	0.5	1	14
bupa	170	175	c	0.5	1	51	c	0.5	1	51
credit	353	300	c	2	1	44	c	1	1	47
glass	107	107	c	3.5	4	21	c	1.5	2	20
haberman	144	150	c	1.5	1	42	c	1.5	1	42
heart	150	147	d	2	1	29	d	1	1	29
pima	400	368	c	1.5	9	103	c	1	7	103
USvotes	235	200	c	10	3	21	c	1	1	10

classification errors obtained on the testing set. The "s" column in each table corresponds to the size of the classifier (i.e. number of balls in case of Table 5.1 and number of halfspaces in Table 5.2. Moreover, the columns "T" and "P" in Table 5.2 refer to the machine type (conjunction or disjunction) and the penalty values respectively.

The most striking feature in Table 5.2 is the level of sparsity achieved by the SCM in comparison with the SVM. This difference is huge. In particular, the SCMs with half-spaces never contained more than 4 half-spaces (*i.e.* a compression set of at most 12 points), except for pima (9 half-spaces, or 27 points). The other important feature is that SCMs with half-spaces often provide better generalization than SCMs with balls and SVMs. The difference is substantial on the Credit data set. Hence it is quite clear that data-dependent half-spaces provide an alternative to data-dependent balls for the set of features used by the SCM.

Moreover, the bound that we have obtained is also successful at guiding the model selection. Note how we find better results for model selection from the bound in Table 5.2, especially on USvotes. However, the price to pay for this better performance is the extra computation time[1] since triples of points needs to be examined to find a half-space but only pairs of points need to be considered for balls (Marchand and Shawe-Taylor, 2001).

5.5 Time Complexity Analysis

Let us consider the running time for this algorithm in the conjunction case: For fixed p, we consider every weight vector consisting of a positive-negative example pair. For each weight vector, we sort the examples according to their inner products (possibly kernelized) with the weight vector in $O(m \log m)$ time. Moreover, computing the utility of each potential half-space resulting from a weight vector takes $O(m)$ time. Taking into account all the possible pairs of examples that can become potential weight vectors, it takes $O(m^3 \log m)$ time to find the best half-space. We then remove the examples covered by this half-space and repeat the algorithm. It is well known that greedy algorithms of this kind have the following guarantee: if there exist r half-spaces that covers all the m examples, the greedy algorithm will find at most $r \ln(m)$ half-spaces. Since we almost always have $r \in O(1)$, the running time of the whole algorithm will almost always be $\in O(m^3 \log^2(m))$. However, as mentioned above, the running time is within acceptable limits in practice.

5.6 Conclusion and Outlook

We have introduced a new set of features for the SCM, called data-dependent half-spaces, and have shown that it can provide a favorable alternative to data-dependent balls on some "natural" data sets. Compared with SVMs, our learning algorithm for SCMs with half-spaces produces substantially sparser classifiers (often by a factor of 50) with comparable, and sometimes better, generalization.

By extending the sample compression technique of Littlestone and Warmuth (1986), we have bound the generalization error of the SCM with data-dependent half-spaces in terms of the number of errors and the number of half-spaces it achieves on the training data. Our bound indicates that good generalization error is expected whenever a SCM, with a small

[1]Note that this is still within acceptable bounds. E.g., it takes less than 20 seconds on a 1.6 GHz PC to train once the SCM with half-spaces on the BreastW data set.

number of half-spaces, makes few training errors. Moreover, our bound is tight enough to perform model selection, hence, providing an alternative to the costly cross validation method. Note, however, that our bound applies only to the case of symmetric loss. It would be interesting to generalize this bound to the case of asymmetric loss (which frequently occurs in practice).

CHAPTER 6

Learning the Conjunction of Rays

This chapter proposes two "purist" approaches of learning conjunction or disjunction of simple threshold features called Rays, viz. a Sample Compression based approach and an Occam Razor based approach. We also present corresponding risk bounds in each case. We show how the approaches can perform an implicit feature selection to represent classifier with a very small number of attributes in the case of DNA micro-array data. This chapter also shows how these approaches are limited in their classification and generalization performance by focusing solely on sparsity. The next chapter presents an improved version for learning from Rays.

6.1 Introduction

We demonstrated in the last chapter how an alternate set of features can give better generalization as well as sparser solutions in the SCM scenario. Let us see if we can take this a step further, to the case of high dimensional data. More specifically, we examine how can we adapt the SCM for classifying gene-expression data obtained from DNA micro-array experiments. This is a very important problem since it demands addressing not only the issues such as reliable feature selection and classification accuracy with respect to Machine Learning, but also has significant biological relevance. If one can find a classifier that depends on a small number of genes and that can accurately predict if a DNA micro-array sample originates from cancer tissue or normal tissue, then there is hope that these genes, used by the classifier, may be playing a crucial role in the development of cancer and hence may be of relevance for future therapies. This, in particular, seems important to devise a test for the prognosis of patients. The basic idea of our approach stems from being able to utilize sparsity when obtaining a conjunction of features built on data. So far, we have reviewed features that were built on examples (i.e. utilized all the attributes). However, if the features are built on individual attributes instead, *can we build classifiers that are sparse*? Moreover, *can these sparse classifiers demonstrate good generalization*? Since in this case a sparse classifier would mean one that utilizes a very small subset of attributes to represent the hypothesis, hence performing an implicit feature selection. We will see in this chapter

the answer to the first question. It is indeed possible to obtain sparse classifiers utilizing very few attributes to designate the classifier. We will extend the approaches presented here with regard to better generalization in the next chapter.

An important challenge in the problem of classification of high-dimensional data is to design a learning algorithm that can often construct an accurate classifier that depends on the smallest possible number of attributes. The problem of finding such a small subset of attributes is often referred to as the problem of *feature selection*. Reducing the dependence of the classifier on a small subset of attributes has obvious advantages. First, this helps in ensuring that the classifier and hence its decision making ability does not depend on the noisy and/or irrelevant attributes. Second, a sufficiently small subset might help in identifying the precise classifying behavioral traits. Let us start by briefly reviewing common approaches adopted for dealing with the feature selection problem.

There are a lot of feature selection methods used for finding a small subset of attributes so as to be able to classify high-dimensional data. The standard methods can often be characterized as either "filters" or "wrappers".

The filter based approaches are based on the idea of removing the most irrelevant attributes from the data prior to learning. A filter is an algorithm used to "filter out" these irrelevant attributes before using a base learning algorithm, such as the support vector machine (SVM). Learning algorithms such as the SVM, which was not designed to perform well in the presence of many irrelevant attributes, have an added advantage with such an approach.

A wrapper, on the other hand, is used in conjunction with the base learning algorithm: typically removing recursively the attributes that have received a small "weight" by the classifier obtained from the base learner. That is, a base learning algorithm is used to learn from the original dataset and weights are assigned in accordance to the importance of the attributes. Then a pre-decided fraction of attributes with weights below a certain threshold are removed from the data. This two-step process is then repeated until the target number of attributes is reached.

An example of such wrapper based approach would be the *recursive feature elimination method* that was used by Guyon et al. (2002) in conjunction with the SVM for classification of micro-array data. For the same task, Furey et al. (2000) have used a filter which consists of ranking the attributes (gene expressions) as function of the difference between the positive-example mean and the negative-example mean.

Both the filter based and wrapper based approaches have sometimes produced good

empirical results. However, there is an intrinsic problem with these approaches. None of these two methods have any theoretical justification associated with them. Analytically, it cannot be shown whether either method will perform well and why. Nor can the bounds be defined over the generalization performance of such approaches. Also, there are some other issues. One of the most important issues is deciding what fraction of the attributes in the original dataset are optimal for the best classification results and more importantly future generalization. Most approaches use a hit-and-trial strategy in the sense that a near exhaustive search over the possible subsets of the attributes is performed after which the subset giving the least empirical error is chosen. However, this criterion is not theoretically justified. Also, it cannot be guaranteed that this selected subset of the attributes is optimal. Of course there are other methods such as mixture of the above two strategies, knowledge based methods and so on. But these method too suffer from the limitations discussed above.

What we really need is a learning algorithm that has provably good guarantees in the presence of many irrelevant attributes. One of the first learning algorithms proposed by the COLT community has such a guarantee for the class of conjunctions: if there exists a conjunction, that depends on r out of the n input attributes and that correctly classifies a training set of m examples, then the greedy covering algorithm of Haussler (1988) will find a conjunction of at most $r \ln m$ attributes that makes no training errors. Note the absence of dependence on the number n of input attributes. In contrast, the mistake-bound of the Winnow algorithm (Littlestone, 1988) has a logarithmic dependence on n and will build a classifier on all the n attributes.

Our present work is motivated by this theoretical result of being able to find a small subset of attributes whose conjunction can perform well on the data. Another factor inspiring our current line of research is the fact that simple conjunctions of gene expression levels seems an interesting learning bias for the classification of DNA micro-array data.

In this chapter, we examine two simple strategies to building small conjunctions of simple threshold functions, called *rays*, defined on single real-valued attributes. *Rays* can also act as an alternate set of features in the SCM framework and are apt in the present context since unlike data-dependent balls and half-spaces, these are built on single attributes. The first approach that we discuss is an Occam's razor based approach while the second is a pure sample compression based approach. Both approaches have commonalities since they aim at finding the simplest representation of classifier. The former selects the classifier with a minimum number of bits while the later tries to find the one that is sparsest in terms of the number of features used. We also provide generalization risk bounds for both the

learning algorithms. We will see that such "purist" strategies (focussing purely either on most succinct representation or the sparsest solution) are often not enough to provide good generalization. Hence, in the next chapter, we will extend the sample compression approach presented here and show that learning algorithms that can perform a non-trivial trade-off between the sparsity of the solution and the separating margin of the decision surface can in fact provide better generalization, both theoretically and empirically.

6.2 Definitions

The input space $\mathcal{X}$ consists of all n-dimensional vectors $\mathbf{x} = (x_1, \ldots, x_n)$ where each real-valued component $x_i \in [A_i, B_i]$ for $i = 1, \ldots n$. Hence, A_i and B_i are, respectively, the *a priori* lower and upper bounds on values for x_i. The output space $\mathcal{Y}$ is the set of classification labels that can be assigned to any input vector $\mathbf{x} \in \mathcal{X}$. We focus here on binary classification problems. Thus $\mathcal{Y} = \{0, 1\}$. Each example $\mathbf{z} = (\mathbf{x}, y)$ is an input vector $\mathbf{x}$ with its classification label $y \in \mathcal{Y}$.

We focus on learning algorithms that construct a *conjunction of rays* from a training set. Each *ray* is just a threshold classifier defined on a single attribute (component) x_k. More formally, a ray is identified by an *attribute index* $k \in \{1, \ldots, n\}$, a *threshold value* $t \in \mathbb{R}$, and a *direction* $d \in \{-1, +1\}$ (that specifies whether class 1 is on the largest or smallest values of x_k). Given any input example $\mathbf{x}$, the output $r^k_{td}(\mathbf{x})$ of a ray is defined as:

$$r^k_{td}(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if} \quad (x_k - t)d > 0 \\ 0 & \text{if} \quad (x_k - t)d \leq 0 \end{cases}$$

To specify a *conjunction of rays* we need first to list all the attributes whose rays are present in the conjunction. For this purpose, we use a vector $\mathbf{k} \stackrel{\text{def}}{=} (k_1, \ldots, k_{|\mathbf{k}|})$ of attribute indices $k_j \in \{1, \ldots, n\}$ such that $k_1 < k_2 < \ldots < k_{|\mathbf{k}|}$ where $|\mathbf{k}|$ is the number of indices present in $\mathbf{k}$ (and thus the number of rays in the conjunction) [1].

To complete the specification of a conjunction of rays, we need a vector $\mathbf{t} = (t_{k_1}, t_{k_2}, \ldots, t_{k_{|\mathbf{k}|}})$ of threshold values and a vector $\mathbf{d} = (d_{k_1}, d_{k_2}, \ldots, d_{k_{|\mathbf{k}|}})$ of directions where $k_j \in \{1, \ldots, n\}$ for $j \in \{1, \ldots, |\mathbf{k}|\}$. On any input example $\mathbf{x}$, the output $C^{\mathbf{k}}_{\mathbf{td}}(\mathbf{x})$ of a conjunction of rays is given by:

$$C^{\mathbf{k}}_{\mathbf{td}}(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if} \quad r^j_{t_j d_j}(\mathbf{x}) = 1 \quad \forall j \in \mathbf{k} \\ 0 & \text{if} \quad \exists j \in \mathbf{k} : r^j_{t_j d_j}(\mathbf{x}) = 0 \end{cases}$$

[1] Although it is possible to use up to two rays on any attribute, we limit ourselves here to the case where each attribute can be used for only one ray.

Finally, any algorithm that builds a conjunction can be used to build a disjunction just by exchanging the role of the positive and negative labeled examples. In order to keep our description simple, we describe here only the case of a conjunction. However, the case of disjunction follows analogically.

6.3 An Occam's Razor Approach

Our first approach towards learning the conjunction (or disjunction) of *Rays* is the Occam's Razor approach. Basically, we wish to obtain a hypothesis that can be coded using the least number of bits. We first propose an Occam's Razor risk bound which will ultimately guide the learning algorithm.

To obtain the tightest possible risk bound, we fully exploit the fact that the distribution of classification errors is a binomial. The binomial tail distribution $\mathrm{Bin}(\frac{\kappa}{m}, r)$ associated with a classifier of (true) risk r is defined as the probability that this classifier makes at most κ errors on a test set of m examples:

$$\mathrm{Bin}\left(\frac{\kappa}{m}, r\right) \stackrel{\text{def}}{=} \sum_{i=0}^{\kappa} \binom{m}{i} r^i (1-r)^{m-i}$$

Following Langford (2005) and Blum and Langford (2003), we now define the *binomial tail inversion* $\overline{\mathrm{Bin}}\left(\frac{\kappa}{m}, \delta\right)$ as the largest risk value that a classifier can have while still having a probability of at least δ of observing at most κ errors out of m examples:

$$\overline{\mathrm{Bin}}\left(\frac{\kappa}{m}, \delta\right) \stackrel{\text{def}}{=} \sup\left\{r : \mathrm{Bin}\left(\frac{\kappa}{m}, r\right) \geq \delta\right\}$$

From this definition, it follows that $\overline{\mathrm{Bin}}\left(R_S(f), \delta\right)$ is the *smallest* upper bound, which holds with probability at least $1-\delta$, on the true risk of any classifier f with an observed empirical risk $R_S(f)$ on a test set of m examples:

$$\forall f: \quad \Pr_{S\sim D^m}\left(R(f) \leq \overline{\mathrm{Bin}}\big(R_S(f), \delta\big)\right) \geq 1-\delta$$

Any bound expressed in terms of the binomial tail inversion can be turned into a more conventional and looser bound by inverting a standard approximation of the binomial tail such as those obtained from the inequalities of Chernoff and Hoeffding.

Our starting point is the Occam's razor bound of Langford (2005) which is a tighter version of the bound proposed by Blumer et. al. (1987). It is also more general in the sense that it applies to any prior distribution P over any countable class of classifiers.

Theorem 14 (Langford (2005)). *For any prior distribution P over any countable class $\mathcal{F}$ of classifiers, and for any $\delta \in (0, 1]$, we have:*

$$\Pr_{S\sim D^m}\left\{\forall f \in \mathcal{F}\colon R(f) \leq \overline{\mathrm{Bin}}\big(R_S(f), P(f)\delta\big)\right\} \geq 1 - \delta$$

The proof directly follows from a straightforward union bound argument and from the fact that $\sum_{f\in\mathcal{F}} P(f) \leq 1$. To apply this bound for conjunctions of rays we thus need to choose a suitable prior P for this class.

In our case, we have seen that ray conjunctions are specified in terms of a mixture of discrete parameters $\mathbf{k}$ and $\mathbf{d}$ and continuous parameters $\mathbf{t}$. We will see below that we will use a finite-precision bit string σ to specify the set of threshold values $\mathbf{t}$. Let us denote by $P(\mathbf{k}, \mathbf{d}, \sigma)$ the prior probability assigned to the conjunction $C^{\mathbf{k}}_{\sigma\mathbf{d}}$ described by $(\mathbf{k}, \mathbf{d}, \sigma)$. We choose a prior of the following form:

$$P(\mathbf{k}, \mathbf{d}, \sigma) = \frac{1}{\binom{n}{|\mathbf{k}|}} p(|\mathbf{k}|) \frac{1}{2^{|\mathbf{k}|}} g_{\mathbf{k},\mathbf{d}}(\sigma)$$

where $g_{\mathbf{k},\mathbf{d}}(\sigma)$ is the prior probability assigned to string σ given that we have chosen $\mathbf{k}$ and $\mathbf{d}$. Let $\mathcal{M}(\mathbf{k}, \mathbf{d})$ be the set of all message strings that we can use given that we have chosen $\mathbf{k}$ and $\mathbf{d}$. If $\mathcal{I}$ denotes the set of all 2^n possible attribute index vectors and $\mathcal{D}_{\mathbf{k}}$ denotes the set of all $2^{|\mathbf{k}|}$ binary direction vectors $\mathbf{d}$ of dimension $|\mathbf{k}|$, we have that:

$$\sum_{\mathbf{k}\in\mathcal{I}} \sum_{\mathbf{d}\in\mathcal{D}_{\mathbf{k}}} \sum_{\sigma\in\mathcal{M}(\mathbf{k},\mathbf{d})} P(\mathbf{k}, \mathbf{d}, \sigma) \leq 1$$

whenever $\sum_{d=0}^{n} p(d) \leq 1$ and $\sum_{\sigma\in\mathcal{M}(\mathbf{k},\mathbf{d})} g_{\mathbf{k},\mathbf{d}}(\sigma) \leq 1\ \forall \mathbf{k}, \mathbf{d}$.

The reasons motivating this choice for the prior are the following. The first two factors come from the belief that the final classifier, constructed from the group of attributes specified by $\mathbf{k}$, should depend only on the number $|\mathbf{k}|$ of attributes in this group. If we have complete ignorance about the number of rays the final classifier is likely to have, we should choose $p(d) = 1/(n + 1)$ for $d \in \{0, 1, \ldots, n\}$. However, we should choose a p that decreases as we increase d if we have reasons to believe that the number of rays of the final classifier will be much smaller than n. Since this is usually our case, we propose to use:

$$p(|\mathbf{k}|) = \frac{6}{\pi^2}(|\mathbf{k}| + 1)^{-2}$$

The third factor of $P(\mathbf{k}, \mathbf{d}, \sigma)$ gives equal prior probabilities for each of the two possible values of direction d_j.

To specify the distribution of strings $g_{\mathbf{k},\mathbf{d}}(\sigma)$, consider the problem of coding a threshold value $t \in [a, b] \subset [A, B]$ where $[A, B]$ is some predefined interval in which we are permitted to

choose t and where $[a, b]$ is an interval of "equally good" threshold values.[2] We propose the following diadic coding scheme for the identification of a threshold value that belongs to that interval. Let l be the number of bits that we use for the code. We adopt the convention that a code of $l = 0$ bits specifies the threshold value $(A+B)/2$. A code of $l = 1$ bit either specifies the value $(3A+B)/4$ (when the bit is 0) or the value $(A+3B)/4$ (when the bit is 1). A code of $l = 2$ specifies one of the following values: $(7A+B)/8, (5A+3B)/8, (3A+5B)/8, (A+7B)/8$. Hence, a code of l bits specifies one value among the set Λ_l of threshold values:

$$\Lambda_l \stackrel{\text{def}}{=} \left\{ \left[1 - \frac{2j-1}{2^{l+1}} \right] A + \frac{2j-1}{2^{l+1}} B \right\}_{j=1}^{2^l}$$

Given an interval $[a, b] \subset [A, B]$ of threshold values, we take the smallest number l of bits such that there exists a threshold value in Λ_l that falls in the interval $[a, b]$. In that way, we will need at most $\lfloor \log_2((B-A)/(b-a)) \rfloor$ bits to obtain a threshold value that falls in $[a, b]$.

Hence, to specify the threshold for each ray, we need to specify the number l of bits and a l-bit string s that identifies one of the threshold values in Λ_l. We also need to specify an interval $[A, B]$ of permitted values for t. For this we propose the following scheme. We start with an interval $[A^*, B^*]$ specified by the nature of the attribute (before examining any value that this attribute takes on the training set). Typically A^* and B^* would respectively be the smallest and the largest value that this attribute can possibly have from the attribute's definition. Let $C^* = (A^* + B^*)/2$. We then compute the smallest value A' and the largest value B' that this attribute takes on the training set. Then we find the largest integer κ such that $2^{-\kappa}(C^* - A^*) \geq (C^* - A')$. Then we choose A such that $C^* - A = 2^{-\kappa}(C^* - A^*)$ for that value of κ. Similarly, we find the largest integer κ' such that $2^{-\kappa'}(B^* - C^*) \geq (B' - C^*)$. Then we choose B such that $B - C^* = 2^{-\kappa'}(B^* - C^*)$ for that value of κ'. After choosing, in this way, $[A_i, B_i]$ for each attribute i we perform a first run of the learning algorithm (described below). The we rerun the algorithm by halving again each interval $[A_i, B_i]$ and repeat until the risk bound (given by our next theorem) of the classifier becomes very large.

Therefore, the message string σ that we use for any choice of $\mathbf{k}$ consists of that pair of numbers κ_i and κ_i' that we have just defined and the pair (l_i, s_i) of numbers needed to identify the threshold for each attribute $i \in \mathbf{k}$. The risk bound does not depend on how we actually code σ (for some receiver). It only depends on the a priori probabilities we assign

[2] By a "good" threshold value, we mean a threshold value for a ray that would cover many negative examples and very few positive examples (see the learning algorithm).

to each possible realization of σ. We choose the following distribution:

$$g_{\mathbf{k},\mathbf{d}}(\sigma) \stackrel{\text{def}}{=} g_{\mathbf{k},\mathbf{d}}(\kappa_1, \kappa'_1, l_1, s_1, \ldots, \kappa_{|\mathbf{k}|}, \kappa'_{|\mathbf{k}|}, l_{|\mathbf{k}|}, s_{|\mathbf{k}|}) = \prod_{i \in \mathbf{k}} \zeta(\kappa_i)\zeta(\kappa'_i)\zeta(l_i) \cdot 2^{-l_i} \quad (6.1)$$

$$\text{where} \quad : \quad \zeta(a) \stackrel{\text{def}}{=} \frac{6}{\pi^2}(a+1)^{-2} \quad \forall a \in \mathbb{N}$$

The sum over all the possible realizations of σ gives 1 since $\sum_{i=1}^{\infty} i^{-2} = \pi^2/6$. Note that by giving equal a priori probability to each of the 2^{l_i} strings s_i of length l_i, we give no preference to any threshold value in Λ_{l_i} once we have chosen an interval $[A_i, B_i]$ that we believe is appropriate.

Alternatively, for homogeneous systems where each attribute has the same definition, we could use the same interval value $[A, B]$ for each attribute. In that case, $g_{\mathbf{k},\mathbf{d}}(\sigma)$ would be defined with only one κ and one κ' instead $|\mathbf{k}|$ pairs of parameters κ_i, κ'_i.

The distribution ζ that we have chosen for each string length l_i has the advantage of decreasing slowly so that the risk bound does not deteriorate to rapidly as l_i increases. Other choices are clearly possible.

With this choice of prior, we have the following theorem:

Theorem 15. *Given all our previous definitions and for any $\delta \in (0, 1]$, we have:*

$$\Pr_{S \sim D^m}\left\{\forall \mathbf{k}, \mathbf{d}, \sigma \colon R(C^{\mathbf{k}}_{\sigma\mathbf{d}}) \leq \overline{\text{Bin}}\left(R_S(C^{\mathbf{k}}_{\sigma\mathbf{d}}), \frac{p(|\mathbf{k}|)g_{\mathbf{k},\mathbf{d}}(\sigma)\delta}{\binom{n}{|\mathbf{k}|}2^{|\mathbf{k}|}}\right)\right\} \geq 1 - \delta$$

Finally, we emphasize that the risk bound of Theorem 15, used in conjunction with the distribution of messages given by $g_{\mathbf{k},\mathbf{d}}(\sigma)$, provides a guide for choosing the appropriate tradeoff between sparsity (the number of rays) and the code (the length of the message string). Indeed, the risk bound for a conjunction with a decision surface having a small coding string (small l_is) may be smaller than the risk bound of a sparser conjunction having a large coding string (larger l_is).

6.3.1 The Occam's Razor Learning Algorithm

Ideally, we would like to find a conjunction of rays that minimizes the risk bound of Theorem 15 with the distribution given by $g_{\mathbf{k},\mathbf{d}}(\sigma)$. Unfortunately, this cannot be done efficiently in all cases since this problem is at least as hard as the (NP-complete) minimum set cover problem (Marchand and Shawe-Taylor, 2002). The simple set covering greedy heuristic has, however, a good guarantee in the case where there exists a small conjunction that makes zero training errors. It simply consists of using a ray that covers the largest number of negative examples (without making any errors on the positives), remove these negative covered

examples and repeat until all the negative examples are covered.[3] Here, however, we want to modify this heuristic by incorporating the possibility of making training errors if the final classifier is much smaller. In addition, we want to give preference to rays having a threshold coded with a small number of bits. Hence, we modify the greedy heuristic in the following way.

Let N be the set of negative examples and P be the set of positive examples. We start with $N' = N$ and $P' = P$. Let Q_i be the subset of N' covered by ray i, let R_i be the subset of P' covered by ray i, and let l_i be the number of bits used to code the threshold of ray i. We choose the ray i that maximizes the *utility* U_i defined as:

$$U_i \stackrel{\text{def}}{=} \frac{|Q_i|}{N'} - p\frac{|R_i|}{P} - \eta \cdot l_i$$

where p is the *penalty* suffered by covering (and hence, misclassifying) a positive example and η is the cost of using l_i bits for ray i. Once we have found a ray maximizing U_i, we update $N' = N' - Q_i$ and $P' = P' - R_i$ and repeat to find the next ray until either $N' = \emptyset$ or the maximum number v of rays has been reached (early stopping the greedy). Hence, p, η, and v are the three "learning parameters" that our heuristic uses to generate a set of classifiers. At the end, we use the bound of Theorem 15 to select the best classifier. Another alternative is to determine the best values for p, η, and v by cross-validation.

For each choice of (p, η, v), the learner examines, for each ray, all the threshold values that can be coded with l bits with our scheme. Since it takes $O(\log((B-A)/(b-a)))$ bits to obtain a threshold value that falls in $[a, b]$ and since there exists intervals $[a, b]$ such that $(B-A)/(b-a) \leq m$, this means examining $O(m)$ threshold values and computing U_i for each possibility.

6.4 A Sample Compression Approach

Let us now consider an alternate strategy of learning the conjunction of *Rays* focusing solely on obtaining the sparsest possible solution (the one with the minimum number of features). As before, we start by deriving a risk bound for this case and then proceed to the learning algorithm. We derive the bound on the lines of the sample compression bound obtained for the SCM Half-spaces in Chapter 5.

In sample compression settings for Rays, the message string still specifies the attributes (and directions) as before. However, the thresholds are now specified by training examples.

[3] A ray covers an example iff it assigns -1 to that example.

Hence, if we have $|\mathbf{k}|$ attributes where $\mathbf{k}$ is the set of thresholds, the compression set consists of $|\mathbf{k}|$ training examples (one per threshold).

Now, recall the general sample compression bound of Theorem 11 (Section 3.7):

Sample Compression Theorem. *For any sample compression learning algorithm with a reconstruction function $\mathcal{R}$ that maps arbitrary subsets of a training set and information messages to classifiers:*

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall \mathbf{i} \in \mathcal{I}, \sigma \in \mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) \leq \epsilon(\sigma, \mathbf{Z_i}, |\mathbf{j}|)\right\} \geq 1 - \delta$$

where

$$\epsilon(\sigma, \mathbf{Z_i}, |\mathbf{j}|) = 1 - \exp\left(\frac{-1}{m - |\mathbf{i}| - |\mathbf{j}|}\left[\ln\binom{m}{|\mathbf{i}|} + \ln\binom{m - |\mathbf{i}|}{|\mathbf{j}|} + \ln\left(\frac{1}{P_{\mathcal{M}(\mathbf{Z_i})}(\sigma)}\right) + \ln\left(\frac{1}{\zeta(|\mathbf{i}|)\zeta(|\mathbf{j}|)\delta}\right)\right]\right) \tag{6.2}$$

and ζ is defined by Equation 3.11.

Now, we need to specify the distribution of messages for the conjunction of Rays. This is the term $P_{\mathcal{M}(\mathbf{Z_i})}(\sigma)$ in Equation 6.2. Note that in order to specify a conjunction of Rays, the compression set consists of one example per Ray. For each ray we have one attribute and and a corresponding threshold value determined by the numerical value that this attribute takes on the training example.

The learner first chooses a compression set. Then, for each example in the compression set, the learner chooses an attribute. The determination of this attribute implicitly determines the threshold since the threshold is nothing but the attribute value for corresponding example. Finally, the learner chooses a direction for each attribute (and hence each example). Note that the learner does not need to select a direction if we fix it to be positive. This reduces the running time by half and also yields a tighter bound. However, on some datasets, the empirical results show that the choice of negative direction yields better classification and hence the choice of the direction is crucial by the learner. Moreover, one direction might be preferable to other in some cases.

Now, let the subset of attributes that specifies the rays in our compression set $\mathbf{i}$ be $\mathbf{k}$. Moreover, since there is one ray corresponding to each example in the compression set, we have $|\mathbf{i}| = |\mathbf{k}|$. Now, we assign equal probability to each possible set $|\mathbf{k}|$ of attributes (and hence thresholds) that can be selected from n attributes. Moreover, we assign equal probability over the direction that each ray can have $(+1, -1)$. Finally, taking into account the ordering of the attributes that are chosen, we get the following distribution of messages:

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \binom{n}{|\mathbf{k}|}^{-1} \cdot 2^{-|\mathbf{k}|} \cdot \frac{1}{|\mathbf{k}|!} \ \forall \sigma \tag{6.3}$$

Equation 6.3 along with the *Sample Compression Theorem* completes the bound for the conjunction of Rays.

6.4.1 A Greedy Learning Algorithm

Our learning algorithm for the sample compression approach to learning the conjunction (and disjunction) of rays is an adapted version of the generic Set Covering Machine algorithm of Chapter 4. It consists of choosing the ray i with the largest *utility* U_i where:

$$U_i = |Q_i| - p|R_i| \tag{6.4}$$

where Q_i is the set of negative examples covered (classified as 0) by feature i, R_i is the set of positive examples misclassified by this feature, and p is a learning parameter that gives a penalty p for each misclassified positive example. Once the feature with the largest U_i is found, we remove Q_i and P_i from the training set S and then repeat (on the remaining examples) until either no more negative examples are present or that a maximum number s of features has been reached.

6.5 Results for Classification of DNA Micro-Arrays

We now test the performance of the learning algorithm for conjunction of Rays that we designed on real-world datasets. The *colon tumor* data set (Alon et al., 1999) provides the expression levels of 40 tumor and 22 normal colon tissues measured for 6500 human genes. The *ALL/AML* data set (Golub et al., 1999) provides the expression levels of 7129 human genes for 47 samples of patients with acute lymphoblastic leukemia (ALL) and 25 samples of patients with acute myeloid leukemia (AML). The *B_MD* and *C_MD* data sets (Pomeroy et al., 2002) are micro-array samples containing the expression levels of 6817 human genes. Data set *B_MD* contains 25 classic and 9 desmoplastic medulloblastomas whereas data set *C_MD* contains 39 medulloblastomas survivors and 21 treatment failures (non-survivors). The *Lung* dataset consists of gene expression levels of 919 genes of 52 patients with 39 Adenocarcinoma and 13 Squamous Cell Cancer (Garber et al., 2001). This data has some

missing values which were replaced by zeros. Finally, the *BreastER* dataset is the Breast Tumor data of West et al. (2001) used with Estrogen Receptor status to label the various samples. The data consists of expression levels of 7130 genes of 49 patients with 25 positive Estrogen Receptor samples and 24 negative Estrogen Receptor samples.

Data Set		**SVM**	**SVM+gs**		**SVM_rfe**	
Name	**#exs**	**Errors**	**Errors**	**Size**	**Errors**	**Size**
Colon	62	12	11	256	12	128
B_MD	34	12	6	32	9	64
C_MD	60	29	21	1024	27	7129
ALL/AML	72	18	10	64	18	256
Lung	52	8	6	64	7	32
BreastER	49	14	10	256	10	256

Table 6.1: Results of SVM on DNA Micro-array Datasets.

We have compared our learning algorithm with a linear-kernel soft-margin SVM trained both on all the attributes (gene expressions) and on a subset of attributes chosen by the filter method of Golub et al. (1999). The filter method of Golub et al. (1999) consists of ranking the attributes as function of the difference between the positive-example mean and the negative-example mean and then use only the first ℓ attributes. The resulting learning algorithm, named *SVM+gs* in Table 6.1, is basically the one used by Furey et al. (2000) for the same task. Guyon et al. (2002) claimed obtaining better results with the recursive feature elimination method but, as pointed out by Ambroise and McLachlan (2002), their work contained a methodological flaw. The details on the inherent bias in the recursive feature elimination method can be found in Ambroise and McLachlan (2002). We use the SVM recursive feature elimination algorithm (SVM-rfe) with this bias removed and present these results on the above datasets as well for comparison in Table 6.1.

Each algorithm was tested with the 5-fold cross validation (CV) method. Each of the five training sets and testing sets was the same for all algorithms. The learning parameters of all algorithms and the gene subsets (for SVM+gs and SVM-rfe) were chosen from the training sets *only*. This was done by performing a second (nested) 5-fold CV on each training set. Please refer to Section 2.5 for more details on nested k-fold cross validation method for empirical evaluation of the classifiers.

For the gene subset selection procedure of SVM+gs, we have considered the first $\ell = 2^i$ genes (for $i = 0, 1, \ldots, 12$) ranked according to the criterion of Golub et al. (1999) and have

Data Set		Rays:Occam		
Name	**#exs**	**Errors**	**Size**	**Bits**
Colon	62	22	2	6
B_MD	34	15	1	3
C_MD	60	27	2	4
ALL/AML	72	27	2	6
Lung	52	20	2	5
BreastER	49	25	3	2

Table 6.2: Results of Occam's Razor Approach on DNA Micro-array Datasets.

Data Set		Rays:SC	
Name	**#exs**	**Errors**	**Size**
Colon	62	16	1
B_MD	34	15	1
C_MD	60	29	1
ALL/AML	72	26	1
Lung	52	16	1
BreastER	49	19	1

Table 6.3: Results of Sample Compression Approach on DNA Micro-array Datasets.

chosen the i value that gave the smallest 5-fold CV error on the training set.

Table 6.2 gives the result for the Occam's Razor approach to learning the conjunction of Rays. The "Bits" column signifies the optimal number of bits used to encode the classifier as discussed in Section 6.3. Table 6.3 gives the result for the Sample Compression approach.

For each algorithm, the "Errors" columns of Tables 6.1, 6.2 and 6.3 contain the 5-fold CV error expressed as the sum of errors over the five testing sets and the "Size" columns contain the number of attributes used by the classifier averaged over the five testing sets.

6.6 Conclusion and Outlook

The results in Tables 6.1, 6.2 and 6.3 clearly show that even though the conjunction of Rays are better than the SVMs in terms of the size of final classifier, the SVM beats them considerably in terms of the classification accuracy. Hence, it is clear that the Rays conjunctions in their present form are not powerful enough when it comes to classifying very high

dimensional data. Let us analyze the reasons behind such results.

Our present algorithms focus specifically on obtaining sparse solutions. The risk bound(s) is (are) dominated by the sparsity term(s), i.e. the bounds are minimized when the learner is able to find a classifier with a (very) small number of features. The empirical results hence demonstrates precisely this trait. Often we are able to find just one attribute that accounts for a large number of data points. However, focusing solely on sparsity makes the algorithm costly in terms of its generalization performance. We have thus found empirically that sparse solutions do not always generalize better than large-margin SVM solutions. Perhaps when learning a sparse SCM we should also pay attention to its separating margin. That is, among sparse solutions, we should prefer the ones for which the resulting classifier has a large margin. We did pay attention to the margin in the Occam's razor approach but the risk bound depended on the margin only indirectly via a message string. In the next chapter, we provide a risk bound that directly depends on the margin (and sparsity) of the classifier and present a new learning algorithm that empirically performs better.

CHAPTER 7

PAC-Bayes Learning of Conjunctions of Rays

In the last chapter we presented two approaches for learning conjunctions or disjunctions of Rays both of which achieved considerably sparse solutions, but were constrained by focusing on sparsity alone. This chapter, hence, presents a PAC-Bayes approach for the same where the learning algorithm can sacrifice some sparsity in favor of large margins so as to yield better classifiers. As a result, we propose a feature selection algorithm with provable guarantees over the generalization behavior of the classifier. These results appeared in part in:

M. Marchand and M. Shah. PAC-Bayes Learning of Conjunctions and Classification of Gene-Expression Data. In *Advances in Neural Information Processing Systems 17* (Proceedings of NIPS 2004), 881–888, MIT-Press, Cambridge, MA, USA, 2005.

7.1 Introduction

As described in Chapter 6, our aim is to develop an algorithm that has provably good guarantees in presence of many irrelevant attributes. In this regard, we presented algorithms based on obtaining a conjunction (or disjunction) of the set of simple threshold features built on attributes that we called Rays. However, the algorithms described had limitations in the sense that they focused solely on achieving sparse solutions. As a result, they tended to disregard a solution that had more features but had better generalization. In this chapter, we examine the possibility of sacrificing some sparsity in favor of a more general solution.

The generality of solutions (classifiers) have long been studied in terms of the separating margins achieved around the decision boundaries of the classifiers, especially in the case of the SVM, it has been shown that the optimal solution is the one that maximizes the margin around the separating hyperplane in the feature space (see for instance (Shawe-Taylor and Cristianini, 2004)). Hence, the most obvious approach in our case is to examine whether we can achieve this better generality of classifiers (in terms of large margins) by sacrificing sparsity to a certain extent, i.e. whether we can perform this non-trivial margin-sparsity trade-off and if yes, can this approach deliver better classifiers that still utilize a reasonably less number of features while providing a better generalization.

In this regard, we try to introduce a margin around the decision boundary of each feature (*Ray* in our case) and present a risk bound based on the PAC-Bayes theory that has this ability to perform a non-trivial margin-sparsity trade-off.

This chapter is organized as follows: We propose a PAC-Bayes risk bound which is minimized for classifiers achieving a non-trivial tradeoff between sparsity (the number of rays used) and the magnitude of the separating margin of each ray. Based on this, we propose a "soft greedy" learning algorithm for building small conjunctions of simple threshold functions, called *rays* that we introduced in the last chapter, defined on single real-valued attributes. Finally, we test the proposed soft greedy algorithm on DNA micro-array data sets and analyze these results.

7.2 Definitions

The input space $\mathcal{X}$ consists of all n-dimensional vectors $\mathbf{x} = (x_1, \ldots, x_n)$ where each real-valued component $x_k \in [A_k, B_k]$ for $k = 1, \ldots n$. Hence, A_k and B_k are, respectively, the *a priori* lower and upper bounds on values for x_k. That is, A_k and B_k are nothing but the minimum and the maximum values that the attribute x_k can take and that these values (bounds) are given *a priori*. The output space $\mathcal{Y}$ is the set of classification labels that can be assigned to any input vector $\mathbf{x} \in \mathcal{X}$.

We, as in the previous chapter, focus here on binary classification problems i.e. $\mathcal{Y} = \{0, 1\}$. Each example $\mathbf{z} = (\mathbf{x}, y)$ is an input vector $\mathbf{x}$ with its classification label $y \in \mathcal{Y}$. The expected and empirical risk have the same definitions as described in Section 6.2. Also, we stick to the same definition of the conjunction of Rays as in the last chapter, i.e. given any input example $\mathbf{x}$, the output $r^k_{td}(\mathbf{x})$ of a ray is defined as:

$$r^k_{td}(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if} \quad (x_k - t)d > 0 \\ 0 & \text{if} \quad (x_k - t)d \leq 0 \end{cases}$$

where $k \in \{1, \ldots, n\}$ is the *attribute index*, $t \in [A_k, B_k]$ is the *threshold value*, and $d \in \{-1, +1\}$ is the *direction* (that specifies whether class 1 is on the largest or smallest values of x_k).

On the similar notes, on any input example $\mathbf{x}$, the output $C^{\mathbf{k}}_{\mathbf{td}}(\mathbf{x})$ of a conjunction of rays is given by:

$$C^{\mathbf{k}}_{\mathbf{td}}(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if} \quad r^j_{t_j d_j}(\mathbf{x}) = 1 \quad \forall j \in \mathbf{k} \\ 0 & \text{if} \quad \exists j \in \mathbf{k} : r^j_{t_j d_j}(\mathbf{x}) = 0 \end{cases}$$

where the vector $\mathbf{k} \stackrel{\text{def}}{=} (k_1, \ldots, k_{|\mathbf{k}|})$ is the vector of attribute indices $k_j \in \{1, \ldots, n\}$ such that $k_1 < k_2 < \ldots < k_{|\mathbf{k}|}$ where $|\mathbf{k}|$ is the number of indices present in $\mathbf{k}$, $\mathbf{t} = (t_{k_1}, t_{k_2}, \ldots, t_{k_{|\mathbf{k}|}})$ is the vector of threshold values and $\mathbf{d} = (d_{k_1}, d_{k_2}, \ldots, d_{k_{|\mathbf{k}|}})$ is the vector of directions where $k_j \in \{1, \ldots, n\}$ for $j \in \{1, \ldots, |\mathbf{k}|\}$.

Finally, just as we noted previously, any algorithm that builds a conjunction can be used to build a disjunction just by exchanging the role of the positive and negative labeled examples. We describe here only the case of a conjunction. The disjunction case follows analogically.

7.3 A PAC-Bayes Risk Bound

The PAC-Bayes approach was initiated by McAllester (1999). It aims at providing PAC guarantees to "Bayesian" learning algorithms. We gave a brief description of the PAC-Bayes framework in Section 3.8.

We re-state this PAC-Bayes bound over the risk of Gibbs classifier. Given an input example $\mathbf{x}$, the label $G_Q(\mathbf{x})$ assigned to $\mathbf{x}$ by the Gibbs classifier is defined by the following process. We first choose a classifier h according to the posterior distribution Q and then use h to assign the label $h(\mathbf{x})$ to $\mathbf{x}$. The risk of G_Q is defined as the expected risk of classifiers drawn according to Q:

$$R(G_Q) \stackrel{\text{def}}{=} \mathbf{E}_{h\sim Q} R(h) = \mathbf{E}_{h\sim Q} \mathbf{E}_{(\mathbf{x},y)\sim D} I(f(\mathbf{x}) \neq y)$$

Recall the PAC-Bayes bound of Theorem 12 (Section 3.8).

PAC-Bayes Theorem. *Given any space $\mathcal{H}$ of classifiers. For any data-independent prior distribution P over $\mathcal{H}$ and for any (possibly data-dependent) posterior distribution Q over $\mathcal{H}$, with probability at least $1 - \delta$ over the random draws of training sets S of m examples:*

$$\text{kl}(R_S(G_Q) \| R(G_Q)) \leq \frac{\text{KL}(Q\|P) + \ln \frac{m+1}{\delta}}{m}$$

where $\text{KL}(Q\|P)$ is the Kullback-Leibler divergence between distributions[1] Q and P:

$$\text{KL}(Q\|P) \stackrel{\text{def}}{=} \mathbf{E}_{h\sim Q} \ln \frac{Q(h)}{P(h)}$$

[1]Here $Q(h)$ denotes the probability density function associated to Q, evaluated at h.

and where $\mathrm{kl}(q\|p)$ *is the Kullback-Leibler divergence between the Bernoulli distributions with probabilities of success* q *and* p*:*

$$\mathrm{kl}(q\|p) \stackrel{\text{def}}{=} q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p}$$

Also, as before, the bound given by the PAC-Bayes theorem for the risk of Gibbs classifiers can be turned into a bound for the risk of Bayes classifiers. It follows that the error rate of G_Q is at least half of the error rate of Bayes classifier B_Q. Hence $R(B_Q) \leq 2R(G_Q)$.

In our case, we have seen that ray conjunctions are specified in terms of a mixture of discrete parameters $\mathbf{k}$ and $\mathbf{d}$ (the attribute vector and the direction vector respectively) and continuous parameters $\mathbf{t}$ (the threshold vector). If we denote by $P_{\mathbf{k},\mathbf{d}}(\mathbf{t})$ the probability density function associated with a prior P over the class of ray conjunctions, we consider here priors of the form:

$$P_{\mathbf{k},\mathbf{d}}(\mathbf{t}) = \frac{1}{\binom{n}{|\mathbf{k}|}} p(|\mathbf{k}|) \frac{1}{2^{|\mathbf{k}|}} \prod_{j \in \mathbf{k}} \frac{1}{B_j - A_j} \quad ; \quad \forall t_j \in [A_j, B_j]$$

If $\mathcal{K}$ denotes the set of all 2^n possible attribute index vectors and $\mathcal{D}_{\mathbf{k}}$ denotes the set of all $2^{|\mathbf{k}|}$ binary direction vectors $\mathbf{d}$ of dimension $|\mathbf{k}|$, we have that:

$$\sum_{\mathbf{k} \in \mathcal{K}} \sum_{\mathbf{d} \in \mathcal{D}_{\mathbf{k}}} \prod_{j \in \mathbf{k}} \int_{A_j}^{B_j} dt_j P_{\mathbf{k},\mathbf{d}}(\mathbf{t}) = 1$$

whenever $\sum_{e=0}^{n} p(e) = 1$.

The reasons motivating this choice for the prior are the following. The first two factors come from the belief that the final classifier, constructed from the group of attributes specified by $\mathbf{k}$, should depend only on the number $|\mathbf{k}|$ of attributes in this group. If we have complete ignorance about the number of rays the final classifier is likely to have, we should choose $p(e) = 1/(n+1)$ for $e \in \{0, 1, \ldots, n\}$. However, we should choose a p that decreases as we increase e if we have reasons to believe that the number of rays of the final classifier will be much smaller than n. The third factor of $P_{\mathbf{k},\mathbf{d}}(\mathbf{t})$ gives equal prior probabilities for each of the two possible values of direction d_j. Finally, for each ray, every possible threshold value t should have the same prior probability of being chosen if we do not have any prior knowledge that would favor some values over the others. Since each attribute value x_k is constrained, a priori, to be in $[A_k, B_k]$, we have chosen a uniform probability density on $[A_k, B_k]$ for each t_k such that $k \in \mathbf{k}$. This explains the last factors of $P_{\mathbf{k},\mathbf{d}}(\mathbf{t})$.

Given a training set S, the learner will choose an attribute group $\mathbf{k}$ and a direction vector $\mathbf{d}$. For each attribute $x_k \in [A_k, B_k] : k \in \mathbf{k}$, a margin interval $[a_k, b_k] \subseteq [A_k, B_k]$ will also be

chosen by the learner. A deterministic ray-conjunction classifier is then specified by choosing the thresholds values $t_k \in [a_k, b_k]$. It is tempting at this point to choose $t_k = (a_k + b_k)/2 \; \forall k \in \mathbf{k}$ (*i.e.*, in the middle of each interval). However, we will see shortly that the PAC-Bayes theorem offers a better guarantee for another type of deterministic classifier.

The Gibbs classifier is defined with a posterior distribution Q having all its weight on the same $\mathbf{k}$ and $\mathbf{d}$ as chosen by the learner but where each t_k is uniformly chosen in $[a_k, b_k]$. The KL divergence between this posterior Q and the prior P is then given by:

$$\begin{aligned} KL(Q\|P) &= \prod_{j \in \mathbf{k}} \int_{a_j}^{b_j} \frac{dt_j}{b_j - a_j} \ln \left(\frac{\prod_{k \in \mathbf{k}} (b_k - a_k)^{-1}}{P_{\mathbf{k},\mathbf{d}}(\mathbf{t})} \right) \\ &= \ln \binom{n}{|\mathbf{k}|} + \ln \left(\frac{1}{p(|\mathbf{k}|)} \right) + |\mathbf{k}| \ln(2) + \sum_{k \in \mathbf{k}} \ln \left(\frac{B_k - A_k}{b_k - a_k} \right) \end{aligned}$$

Hence, we see that the KL divergence between the "continuous components" of Q and P (given by the last term) vanishes when $[a_k, b_k] = [A_k, B_k] \; \forall k \in \mathbf{k}$. Furthermore, the KL divergence between the "discrete components" of Q and P is small for small values of $|\mathbf{k}|$ (whenever $p(|\mathbf{k}|)$ is not too small). *Hence, this KL divergence between our choices for Q and P exhibits a tradeoff between margins (large values of $b_k - a_k$) and sparsity (small value of $|\mathbf{k}|$) for Gibbs classifiers.* According to the PAC-Bayes theorem, the Gibbs classifier with the smallest guarantee of risk $R(G_Q)$ should minimize a non trivial combination of $KL(Q\|P)$ (margins-sparsity tradeoff) and empirical risk $R_S(G_Q)$.

Since the posterior Q is identified by an attribute group vector $\mathbf{k}$, a direction vector $\mathbf{d}$, and intervals $[a_k, b_k] \; \forall k \in \mathbf{k}$, we will refer to the Gibbs classifier G_Q by $G^{\mathbf{kd}}_{\mathbf{ab}}$ where $\mathbf{a}$ and $\mathbf{b}$ are the vectors formed by the unions of a_ks and b_ks respectively. We can obtain a closed-form expression for $R_S(G^{\mathbf{kd}}_{\mathbf{ab}})$ by first considering the risk $R_{(\mathbf{x},y)}(G^{\mathbf{kd}}_{\mathbf{ab}})$ on a single example $(\mathbf{x}, y)$ since $R_S(G^{\mathbf{kd}}_{\mathbf{ab}}) = \mathbf{E}_{(\mathbf{x},y)\sim S} R_{(\mathbf{x},y)}(G^{\mathbf{kd}}_{\mathbf{ab}})$. From our definition for Q, we find that:

$$R_{(\mathbf{x},y)}(G^{\mathbf{kd}}_{\mathbf{ab}}) = (1 - 2y) \left[\prod_{k \in \mathbf{k}} \sigma^{d_k}_{a_k,b_k}(x_k) - y \right] \tag{7.1}$$

where we have used the following piece-wise linear functions:

$$\sigma^{+}_{a,b}(x) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if} \quad x < a \\ \frac{x-a}{b-a} & \text{if} \quad a \le x \le b \\ 1 & \text{if} \quad b < x \end{cases} \quad ; \quad \sigma^{-}_{a,b}(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if} \quad x < a \\ \frac{b-x}{b-a} & \text{if} \quad a \le x \le b \\ 0 & \text{if} \quad b < x \end{cases} \tag{7.2}$$

Hence we notice that $R_{(\mathbf{x},1)}(G^{\mathbf{kd}}_{\mathbf{ab}}) = 1$ (and $R_{(\mathbf{x},0)}(G^{\mathbf{kd}}_{\mathbf{ab}}) = 0$) whenever there exist $k \in \mathbf{k} \; : \; \sigma^{d_k}_{a_k,b_k}(x_k) = 0$. This occurs iff there exists a ray which outputs 0 on $\mathbf{x}$. We can

also verify that the expression for $R_{(\mathbf{x},y)}(C^{\mathbf{k}}_{\mathbf{td}})$ is identical to the expression for $R_{(\mathbf{x},y)}(G^{\mathbf{kd}}_{\mathbf{ab}})$ except that the piece-wise linear functions $\sigma^{d_k}_{a_k,b_k}(x_k)$ are replaced by the indicator functions $I((x_k - t_k)d_k > 0)$.

The PAC-Bayes theorem provides a risk bound for the Gibbs classifier $G^{\mathbf{kd}}_{\mathbf{ab}}$. Since the Bayes classifier $B^{\mathbf{kd}}_{\mathbf{ab}}$ just performs a majority vote under the same posterior distribution as the one used by $G^{\mathbf{kd}}_{\mathbf{ab}}$, we have that $B^{\mathbf{kd}}_{\mathbf{ab}}(\mathbf{x}) = 1$ iff the probability that $G^{\mathbf{kd}}_{\mathbf{ab}}$ classifies $\mathbf{x}$ as positive exceeds 1/2. Hence, it follows that:

$$B^{\mathbf{kd}}_{\mathbf{ab}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \prod_{k\in\mathbf{k}} \sigma^{d_k}_{a_k,b_k}(x_k) > 1/2 \\ 0 & \text{if } \prod_{k\in\mathbf{k}} \sigma^{d_k}_{a_k,b_k}(x_k) \leq 1/2 \end{cases} \tag{7.3}$$

Note that $B^{\mathbf{kd}}_{\mathbf{ab}}$ has an *hyperbolic* decision surface. Consequently, $B^{\mathbf{kd}}_{\mathbf{ab}}$ is not representable as a conjunction of rays. There is, however, no computational difficulty at obtaining the output of $B^{\mathbf{kd}}_{\mathbf{ab}}(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$.

From the relation between $B^{\mathbf{kd}}_{\mathbf{ab}}$ and $G^{\mathbf{kd}}_{\mathbf{ab}}$, it also follows that $R_{(\mathbf{x},y)}(B^{\mathbf{kd}}_{\mathbf{ab}}) \leq 2R_{(\mathbf{x},y)}(G^{\mathbf{kd}}_{\mathbf{ab}})$ for any $(\mathbf{x}, y)$. Consequently, $R(B^{\mathbf{kd}}_{\mathbf{ab}}) \leq 2R(G^{\mathbf{kd}}_{\mathbf{ab}})$. Hence, we have our main theorem:

Theorem 16. *Given all our previous definitions, for any $\delta \in (0, 1]$, and for any p satisfying $\sum_{e=0}^{n} p(e) = 1$, we have:*

$$\Pr_{S\sim D^m}\left(\forall \mathbf{k},\mathbf{d},\mathbf{a},\mathbf{b}\colon R(G^{\mathbf{kd}}_{\mathbf{ab}}) \leq \sup\left\{\epsilon\colon \mathrm{kl}(R_S(G^{\mathbf{kd}}_{\mathbf{ab}})\|\epsilon) \leq \frac{1}{m}\left[\ln\binom{n}{|\mathbf{k}|} + \right.\right.\right.$$
$$\left.\left.\left. + \ |\mathbf{k}|\ln(2) + \ln\left(\frac{1}{p(|\mathbf{k}|)}\right) + \sum_{k\in\mathbf{k}} \ln\left(\frac{B_k - A_k}{b_k - a_k}\right) + \ln\frac{m+1}{\delta}\right]\right\}\right) \geq 1 - \delta$$

Furthermore: $R(B^{\mathbf{kd}}_{\mathbf{ab}}) \leq 2R(G^{\mathbf{kd}}_{\mathbf{ab}}) \quad \forall \mathbf{k},\mathbf{d},\mathbf{a},\mathbf{b}$.

7.4 A Soft Greedy Learning Algorithm

Theorem 16 suggests that the learner should try to find the Bayes classifier $B^{\mathbf{kd}}_{\mathbf{ab}}$ that uses a small number of attributes (*i.e.*, a small $|\mathbf{k}|$), each with a large separating margin $(b_k - a_k)$, while keeping the empirical Gibbs risk $R_S(G^{\mathbf{kd}}_{\mathbf{ab}})$ at a low value. To achieve this goal, we have adapted the greedy algorithm for the set covering machine (SCM) proposed by Marchand and Shawe-Taylor (2002) and described briefly in chapter 4. It consists of choosing the feature (here a ray) i with the largest *utility* U_i where:

$$U_i = |Q_i| - p|R_i| \tag{7.4}$$

where Q_i is the set of negative examples covered (classified as 0) by feature i, R_i is the set of positive examples misclassified by this feature, and p is a learning parameter that gives a penalty p for each misclassified positive example. Once the feature with the largest U_i is found, we remove Q_i and P_i from the training set S and then repeat (on the remaining examples) until either no more negative examples are present or that a maximum number s of features has been reached.

In our case, however, we need to keep the Gibbs risk on S low instead of the risk of a deterministic classifier. Since the Gibbs risk is a "soft measure" that uses the piece-wise linear functions $\sigma^d_{a,b}$ instead of the "hard" indicator functions, we cannot make use of the hard utility function of Equation 7.4. Instead, we need a "softer" version of this utility function U_i. Indeed, a negative example that falls in the linear region of a $\sigma^d_{a,b}$ is in fact partly covered and vice versa for the positive example.

Following this observation, let $\mathbf{k}'$ be the vector of indices of the attributes that we have used so far for the construction of the classifier. Let us first define the *covering value* $\mathcal{C}(G^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}})$ of $G^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}}$ by the "amount" of negative examples assigned to class 0 by $G^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}}$:

$$\mathcal{C}(G^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}}) \stackrel{\text{def}}{=} \sum_{(\mathbf{x},y)\in S} (1-y)\left[1 - \prod_{j\in\mathbf{k}'} \sigma^{d_j}_{a_j,b_j}(x_j)\right]$$

We also define the *positive-side error* $\mathcal{E}(G^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}})$ of $G^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}}$ as the "amount" of positive examples assigned to class 0 :

$$\mathcal{E}(G^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}}) \stackrel{\text{def}}{=} \sum_{(\mathbf{x},y)\in S} y\left[1 - \prod_{j\in\mathbf{k}'} \sigma^{d_j}_{a_j,b_j}(x_j)\right]$$

We now want to add another ray on another attribute, call it i, to obtain a new vector $\mathbf{k}''$ containing this new attribute in addition to those present in $\mathbf{k}'$. Hence, we now introduce the *covering contribution* of ray i as:

$$\mathcal{C}^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}}(i) \stackrel{\text{def}}{=} \mathcal{C}(G^{\mathbf{k}''\mathbf{d}'}_{\mathbf{a}'\mathbf{b}'}) - \mathcal{C}(G^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}}) = \sum_{(\mathbf{x},y)\in S} (1-y)\left[1 - \sigma^{d_i}_{a_i,b_i}(x_i)\right] \prod_{j\in\mathbf{k}'} \sigma^{d_j}_{a_j,b_j}(x_j)$$

and the *positive-side error contribution* of ray i as:

$$\mathcal{E}^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}}(i) \stackrel{\text{def}}{=} \mathcal{E}(G^{\mathbf{k}''\mathbf{d}'}_{\mathbf{a}'\mathbf{b}'}) - \mathcal{E}(G^{\mathbf{k}'\mathbf{d}}_{\mathbf{ab}}) = \sum_{(\mathbf{x},y)\in S} y\left[1 - \sigma^{d_i}_{a_i,b_i}(x_i)\right] \prod_{j\in\mathbf{k}'} \sigma^{d_j}_{a_j,b_j}(x_j)$$

Typically, the covering contribution of ray i should increase its "utility" and its positive-side error should decrease it. Moreover, we want to decrease the "utility" of ray i by an

amount which would become large whenever it has a small separating margin. Our expression for $KL(Q\|P)$ suggests that this amount should be proportional to $\ln((B_i - A_i)/(b_i - a_i))$.

We define the *utility* $U_{\mathbf{ab}}^{\mathbf{k'd}}(i)$ *of adding ray* i *to* $G_{\mathbf{ab}}^{\mathbf{k'd}}$ as:[2]

$$U_{\mathbf{ab}}^{\mathbf{k'd}}(i) \stackrel{\text{def}}{=} \mathcal{C}_{\mathbf{ab}}^{\mathbf{k'd}}(i) - p\mathcal{E}_{\mathbf{ab}}^{\mathbf{k'd}}(i) - \eta \ln \frac{B_i - A_i}{b_i - a_i}$$

where parameter p represents the *penalty* of misclassifying a positive example and η is another parameter that controls the importance of having a large margin. These learning parameters can be chosen by cross-validation. For fixed values of these parameters, the "soft greedy" algorithm simply consists of adding, to the current Gibbs classifier, a ray with maximum added utility until either the maximum number s of rays has been reached or that all the negative examples have been (totally) covered. It is understood that, during this soft greedy algorithm, we can remove an example $(\mathbf{x}, y)$ from S whenever it is totally covered. This occurs whenever $\prod_{j \in \mathbf{k'}} \sigma_{a_j, b_j}^{d_j}(x_j) = 0$.

7.4.1 Time Complexity Analysis

We analyze the time complexity of this algorithm now for fixed p and η. For each potential ray (and hence for each attribute effectively), we first sort the m examples with respect to their values for the attribute under consideration. This takes $O(m \log m)$ time. Then, we examine each potential a_i value. Corresponding to each a_i, we examine all the potential b_i values (all the values greater than a_i). This gives us a time complexity of $O(m^2)$. Now if k is the largest number of examples falling into the margin, calculating the covering and error contributions and then finding the best ray over an attribute takes $O(km^2)$ time. Moreover, we allow $k \in O(m)$ giving us a time complexity of $O(m^3)$. Finally, we do this over all the attributes. Hence, the overall time complexity of the algorithm is $O(nm^3)$. However, note that generally $n >> m$, and hence dominates the complexity factor. More so, since once the best ray is found, we remove the examples covered by this ray from the training set and repeat the algorithm. Now, we know that greedy algorithms of this kind have the following guarantee: if there exist r rays that covers all the m examples, the greedy algorithm will find at most $r \ln(m)$ rays. Since we almost always have $r \in O(1)$, the running time of the whole algorithm will almost always be $\in O(nm^3 \log(m))$. The good news is that $n >> m$ and our bound is roughly linear in terms of n.

[2]Note that this utility function is different and more stable than the one reported in (Marchand and Shah, 2005)

7.4.2 Fixed-Margin Heuristic

In order to show why we prefer a uniformly distributed threshold as opposed to the one fixed at the middle of the interval $[a_i, b_i]$ for each feature i, we use an alternate algorithm with, what we call the fixed margin heuristic. The algorithm is similar to the one described above but with an additional parameter γ. The parameter decides a fixed margin boundary around the threshold, i.e. γ decides the length of the interval $[a_i, b_i]$. The algorithm still chooses the attribute vector $\mathbf{k}$, the direction vector $\mathbf{d}$ and the vectors $\mathbf{a}$ and $\mathbf{b}$. However, the a_i's and b_i's for each feature i are chosen such that, $|b_i - a_i| = 2\gamma$. The threshold t_i is then fixed in the middle of this interval, that is $t_i = \frac{(a_i+b_i)}{2}$. Hence, for each feature i, the interval $[a_i, b_i] = [t_i - \gamma, t_i + \gamma]$ when $d_i = -1$ and $[a_i, b_i] = [t_i + \gamma, t_i - \gamma]$ when $d_i = +1$. This algorithm, as can obviously be seen, takes less time than the original one. The time complexity of this algorithm is roughly $O(nm^2 \log(m))$ for fixed p and γ.

7.5 Results for Classification of DNA Micro-Arrays

Let us now test the performance of this new soft greedy learning algorithm on real-world datasets. We have tested the soft greedy learning algorithm on the same DNA micro-array data sets as described in Section 6.5.

Also, we compare our soft greedy learning algorithm with a linear-kernel soft-margin SVM trained both on all the attributes (gene expressions) and on a subset of attributes chosen by the filter method of Golub et al. (1999) as well as the recursive feature elimination algorithm. See Section 6.5 for details.

Again, each algorithm was tested with the 5-fold cross validation (CV) method. Each of the five training sets and testing sets was the same for all algorithms. The learning parameters of all algorithms and the gene subsets (for SVM+gs and SVM-rfe) were chosen from the training sets *only*. This was done by performing a second (nested) 5-fold CV on each training set.

Tables 7.2 and 7.3 give the results of the PAC-Bayes approach to learning the conjunction of Rays discussed in Section 7.3. The results in Table 7.2 are for the PAC-Bayes learning algorithm with the fixed margin heuristic. We discussed this in Section 7.4.2.

For each algorithm, the "Errors" columns of Tables 7.1, 7.2 and 7.3 contain the 5-fold CV error expressed as the sum of errors over the five testing sets and the "Size" columns contain the number of attributes used by the classifier averaged over the five testing sets. The "G-err" and "B-err" columns of Table 7.3 refer to the error rates of Gibbs and Bayes

Data Set		SVM	SVM+gs		SVM_rfe	
Name	**#exs**	**Errors**	**Errors**	**Size**	**Errors**	**Size**
Colon	62	12	11	256	12	128
B_MD	34	12	6	32	9	64
C_MD	60	29	21	1024	27	7129
ALL/AML	72	18	10	64	18	256
Lung	52	8	6	64	7	32
BreastER	49	14	10	256	10	256

Table 7.1: Results of SVM on DNA Micro-array Datasets.

Data Set		Soft Greedy(fixed margin)		
Name	**#exs**	**size**	**Errors**	**Bound**
Colon	62	1	14	34
B_MD	34	1	7	20
C_MD	60	3	28	48
ALL/AML	72	2	21	46
Lung	52	2	9	29
BreastER	49	3	11	31

Table 7.2: Results of the PAC-Bayes Approach (with Fixed-Margin Heuristic) on DNA Micro-array Datasets.

Classifiers respectively. The "Ratio" column of Table 7.3 refers to the average value of $(b_k - a_k)/(B_k - A_k)$ obtained for the rays used by classifiers and the "bound" columns of Tables 7.2 and 7.3 refer to the average risk bound of Theorem 16 multiplied by the total number of examples.

7.6 Conclusion and Outlook

We analyze the results on some important grounds. As we mentioned in the last chapter, the feature selection method generally produce good empirical results with algorithms such as the support vector machine that are not specifically designed to perform well in presence of a high number of irrelevant attributes. This can be seen in the results of the *SVM+gs* in Table 7.1. The gene selection filter indeed improves the result of SVM on the DNA micro array datasets. So is the case for the Recursive Feature Elimination algorithm in conjunction

Data Set		Soft Greedy				
Name	**#exs**	**ratio**	**size**	**G-errs**	**B-errs**	**Bound**
Colon	62	0.42	1	12	11	34
B_MD	34	0.10	1	6	6	20
C_MD	60	0.077	3	26	22	45
ALL/AML	72	0.002	2	19	18	44
Lung	52	0.12	2	8	7	27
BreastER	49	0.09	3	10	9	29

Table 7.3: Results of the PAC-Bayes Approach on DNA micro-array Datasets.

with SVM (SVM-rfe) in Table 7.1.

In each of the proposed algorithms for learning the Rays' conjunction so far, the number of genes selected are significantly better than SVM. However, the two "purist" approaches do not give as good results as the PAC-Bayes approach. Reasons for such performance can be seen in terms of the quantities that these two approaches optimize: The Sample compression approach tries to minimize the number of genes used but does not take into account the separating margin and hence compromises accuracy. On the other hand, the Occam's Razor approach tries to find a classifier that depends on margin only indirectly.

The PAC-Bayes approach can perform significant margin-sparsity tradeoff by focusing explicitly on both margin and sparsity and gives better results. This approach hence finds a balance between these two quantities and thus outputs a very good accuracy. Moreover, fixing the threshold in the middle of the interval $[a_i, b_i]$ for each *Ray* i generally limits the choice of the classifiers. Hence, the approach with the threshold uniformly distributed in the margin interval is preferred.

In the context of the PAC-Bayes approach note that the Bayes classifier no longer has a piecewise linear decision surface. For the PAC-Bayes approach, we expect the Bayes classifier to generally perform better than the Gibbs classifier. This is aptly reflected in the empirical results as well. However, there is no means to prove that this will always be the case. The Bayes error rate is slightly better than the Gibbs error rate. Finally, the error rates of Bayes, SVM-rfe and SVM+gs are competitive but the number of genes selected by the soft greedy algorithm is always *much* smaller.

It should be noted that there are other possible utility functions possible as well for various learning approaches. We have tried some of these and the reported results are for the ones that were found best (and discussed in the description of the corresponding learning

algorithms).

Finally, as opposed to the SVM+gs and SVM-rfe methods, we have a theoretical explanation of the performance of the soft greedy algorithm and its future generalization on the data. Note that the PAC-Bayes bound that we proposed in this chapter holds for all *a priori* minimum and maximum attribute values A_i and B_i but not uniformly for all A_i and B_i. This is a problem worth investigating further in the future.

CHAPTER 8

Margin-Sparsity Tradeoff and Data-Compression

In this chapter we propose a new algorithm for SCM with data-dependent balls that utilizes two complementary sources of information to represent the hypothesis: the compression set, and the message strings. We propose a new representation for the balls' radii in terms of a code that is small when large margins around the decision boundary can be achieved. Hence, the algorithm can trade-off sparsity in favor of a smaller code. We also present a risk bound that allows the algorithm to perform this trade-off explicitly. These results appeared in part in:

F. Laviolette, M. Marchand and M. Shah. Margin-Sparsity Trade-off for the Set Covering Machine. In *Proceedings of the 16th European Conference on Machine Learning* (ECML '05), Springer LNAI vol. 3720, 206–217, 2005.

8.1 Introduction

In quest of obtaining better classifiers by performing a non-trivial margin-sparsity trade-off, we now propose an alternative algorithm for set covering machine that utilizes two complementary sources of information viz. *the compression set* and *the message string*. This approach is close to the Occam Razor approach that we applied for learning conjunction of *Rays* but provide a tighter bound in this case and moreover this bound can also be effectively utilized for model selection. Our work is inspired by the PAC-MDL approach proposed by Blum and Langford (2003).

Blum and Langford (2003) have derived a PAC-MDL risk bound for classifiers that unifies most of the standard risk bounds in one common framework. This PAC-MDL bound is stated in a non-standard "transductive" setting where, given a training set of m labeled examples, the goal of the learner is to construct a small message string that can be used by a receiver to predict the labels of an unlabelled set of $m + n$ examples that contains the m training examples (without their labels). Consequently, one important drawback of the PAC-MDL bound is that it does not capture the sample-compression bound (Littlestone and Warmuth, 1986) very well. Indeed, in the PAC-MDL transductive setting, the learner has to build a message string that specifies a compression subset among $m + n$ examples whereas, in the

usual "inductive" setting,[1] the learner just needs to specify the compression subset among m training examples. Because of this, the PAC-MDL bound is usually substantially larger than the sample-compression bound.

In this chapter, we propose a tight data-compression risk bound that, although less universal than the PAC-MDL bound, unifies the Occam's razor bound (Blumer et. al., 1987) and the sample-compression bound in the usual inductive setting. This bound, which is a tighter version of the sample-compression bound of Littlestone and Warmuth (1986), is expressed in terms of a small subset of the training set (the compression set) and a message string of additional information needed to identify a classifier. The bound reduces to the tightest version of the Occam's razor bound (Langford, 2005) when no compression set is used and also reduces to the tightest version of the sample-compression bound (Langford, 2005) when no message string of additional information is used. We illustrate, on the set covering machine (SCM) (Marchand and Shawe-Taylor, 2002), how the learner can tradeoff these two complementary sources of information (the compression set and the message string) to obtain classifiers having a smaller risk.

In particular, we present a new algorithm for the SCM where the learner uses a code for distances that becomes small when there exists large margins of "equally good" positions for the decision surface of a classifier. Hence, with this new algorithm, the learner can tradeoff sparsity with the margin. The *sparsity* is nothing but the inverse of the compression set size while *margin* denotes the inverse of the message length. We show, on natural data sets, that this new SCM algorithm compares favorably to the previous SCM algorithm of Marchand and Shawe-Taylor (2002) and we also show that the data-compression risk bound is an effective guide for choosing the proper margin-sparsity tradeoff of a classifier. In this chapter, we make use of the SCM with data-dependent balls as described in Chapter 4.

8.2 A Data-Compression Risk Bound

We derive the data-compression risk bound using the sample compression framework. However, we follow the *Binomial tail inversion* version of the bounds (Langford, 2005, Blum and Langford, 2003) somewhat similar to what we did in Chapter 6.

In addition to the notation used so far, we will use $\bar{\mathbf{i}}$ to denote the set of indices not present in $\mathbf{i}$. Hence, we have $S = \mathbf{z}_{\mathbf{i}} \cup \mathbf{z}_{\bar{\mathbf{i}}}$ for any vector $\mathbf{i} \in \mathcal{I}$ where $\mathcal{I}$ denotes the set of the 2^m possible realizations of $\mathbf{i}$.

[1] In this setting, the task of the learner is to find a classifier with the smallest true risk.

In contrast to the perceptron learning rule and the SVM where the final classifier can be reconstructed solely from a compression set (Graepel, Herbrich and Shawe-Taylor, 2000, Graepel, Herbrich, and Williamson, 2001), the reconstruction function for SCMs needs both a compression set and a message string. Later, we will see how the learner can tradeoff the compression set size with the length of the message string to obtain a classifier with a smaller risk bound and, hopefully, a smaller true risk.

We seek a tight risk bound for arbitrary reconstruction functions that holds uniformly for all compression sets and message strings. For this, we adopt the PAC setting where each example $\mathbf{z}$ is drawn according to a fixed, but unknown, probability distribution D on $\mathcal{X} \times \mathcal{Y}$. The notations for the true and the empirical risks remain the same.

Recall from Chapter 6 that $\overline{\text{Bin}}\left(R_S(f), \delta\right)$ is the *smallest* upper bound, which holds with probability at least $1 - \delta$, on the true risk of any classifier f with an observed empirical risk $R_S(f)$ on a test set of m examples:

$$\mathbf{P}_{\mathbf{Z}^m}\left\{ R(f) \leq \overline{\text{Bin}}\Big(R_{\mathbf{Z}^m}(f), \delta\Big) \right\} \geq 1 - \delta \quad \forall f \tag{8.1}$$

Note that the quantifier $\forall f$ appears *outside* the probability $\mathbf{P}_{\mathbf{Z}^m}\{\cdot\}$ because the bound $\overline{\text{Bin}}\left(R_S(f), \delta\right)$ does not hold *simultaneously* (and uniformly) for all classifiers f member of some predefined class $\mathcal{F}$. In contrast, the proposed risk bound of Theorem 17 holds uniformly for all compression sets and message strings.

The proposed risk bound is a generalization of the sample-compression risk bound of Langford (2005) to the case where part of the data-compression information is given by a message string. It also has the property to reduce to the Occam's razor bound when the sample compression set vanishes. The idea of using a message string as an additional source of information was also used by Littlestone and Warmuth (1986) and Ben-David and Litman (1998) to obtain a sample-compression bound looser than the bound presented here. Moreover, the proposed bound applies to any compression set-dependent distribution of messages $P_{\mathcal{M}(\mathbf{z_i})}$ satisfying:

$$\sum_{\sigma \in \mathcal{M}(\mathbf{z_i})} P_{\mathcal{M}(\mathbf{z_i})}(\sigma) \leq 1 \quad \forall \mathbf{z_i} \tag{8.2}$$

and any prior distribution $P_{\mathcal{I}}$ of vectors of indices satisfying:

$$\sum_{\mathbf{i} \in \mathcal{I}} P_{\mathcal{I}}(\mathbf{i}) \leq 1 \tag{8.3}$$

Theorem 17. [2] *For any reconstruction function $\mathcal{R}$ that maps arbitrary subsets of a training set and message strings to classifiers, for any prior distribution $P_{\mathcal{I}}$ of vectors of indices, for*

[2] Note that a specialized version of this result appear in Theorem 15

any compression set-dependent distribution of messages $P_{\mathcal{M}(\mathbf{z_i})}$*, and for any* $\delta \in (0,1[$*, we have:*

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall \mathbf{i} \in \mathcal{I}, \forall \sigma \in \mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) \leq \overline{\mathrm{Bin}}\Big(R_{\mathbf{Z}_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma, \mathbf{Z_i})), P_{\mathcal{I}}(\mathbf{i})P_{\mathcal{M}(\mathbf{z_i})}(\sigma)\delta\Big)\right\} \geq 1-\delta$$

where, for any training set $\mathbf{z}^m$*,* $R_{\mathbf{z}_{\bar{\mathbf{i}}}}(f)$ *denotes the empirical risk of classifier* f *on the examples of* $\mathbf{z}^m$ *that do not belong to the compression set* $\mathbf{z_i}$*.*

Proof. Consider:

$$P' \stackrel{\text{def}}{=} \mathbf{P}_{\mathbf{Z}^m}\left\{\exists \mathbf{i} \in \mathcal{I}\colon \exists \sigma \in \mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \overline{\mathrm{Bin}}\Big(R_{\mathbf{Z}_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma, \mathbf{Z_i})), P_{\mathcal{I}}(\mathbf{i})P_{\mathcal{M}(\mathbf{z_i})}(\sigma)\delta\Big)\right\}$$

To prove the theorem, we show that $P' \leq \delta$. Since $\mathbf{P}_{\mathbf{Z}^m}(\cdot) = \mathbf{E}_{\mathbf{Z_i}}\mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}(\cdot)$, the union bound and Equations 8.1, 8.2, and 8.3 imply that we have:

$$\begin{aligned} P' &\leq \sum_{\mathbf{i}\in\mathcal{I}} \mathbf{E}_{\mathbf{Z_i}} \sum_{\sigma\in\mathcal{M}(\mathbf{Z_i})} \mathbf{P}_{\mathbf{Z}_{\bar{\mathbf{i}}}|\mathbf{Z_i}}\left\{R(\mathcal{R}(\sigma, \mathbf{Z_i})) > \overline{\mathrm{Bin}}\Big(R_{\mathbf{Z}_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma, \mathbf{Z_i})), P_{\mathcal{I}}(\mathbf{i})P_{\mathcal{M}(\mathbf{z_i})}(\sigma)\delta\Big)\right\} \\ &\leq \sum_{\mathbf{i}\in\mathcal{I}} \mathbf{E}_{\mathbf{Z_i}} \sum_{\sigma\in\mathcal{M}(\mathbf{Z_i})} P_{\mathcal{I}}(\mathbf{i})P_{\mathcal{M}(\mathbf{z_i})}(\sigma)\delta \\ &\leq \delta \end{aligned}$$

□

The risk bound of Theorem 17 appears to be as tight as it possibly can. Indeed, the proof of Theorem 17 contains three inequalities. The last two inequalities come from Equations 8.1, 8.2, and 8.3 and cannot be improved. The first inequality comes from the application of the union bound for all the possible choices of a compression subset of the training set and is unavoidable for statistically independent training examples.

It is important to note that, once $P_{\mathcal{I}}$ and $P_{\mathcal{M}(\mathbf{z_i})}$ are specified, the risk bound of Theorem 17 for classifier $\mathcal{R}(\mathbf{z_i}, \sigma)$ depends on its empirical risk *and* on the product $P_{\mathcal{I}}(\mathbf{i})P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$. However, $\ln\left(\frac{1}{P_{\mathcal{I}}(\mathbf{i})P_{\mathcal{M}(\mathbf{z_i})}(\sigma)}\right)$ is just the amount of information needed to specify a classifier $\mathcal{R}(\mathbf{z_i}, \sigma)$ once we are given a training set and the priors $P_{\mathcal{I}}$ and $P_{\mathcal{M}(\mathbf{z_i})}$. The $\ln(1/P_{\mathcal{I}}(\mathbf{i}))$ term is the information content of the vector of indices $\mathbf{i}$ that specifies the compression set and the $\ln(1/P_{\mathcal{M}(\mathbf{z_i})}(\sigma))$ term is the information content of the message string σ. Consequently the bound of Theorem 17 specifies quantitatively how much training error learning algorithms should trade-off with the amount of information needed to specify a classifier by $\mathbf{i}$ and σ.

Any bound expressed in terms of the binomial tail inversion can be turned into a more conventional and looser bound by inverting a standard approximation of the binomial tail

such as those obtained from the inequalities of Chernoff and Hoeffding. Here, we make use of the following approximations (provided here without proof) for the binomial tail inversion:[3]

Lemma 18. *For any integer $m \geq 1$ and $k \in \{0, \ldots, m\}$, we have:*

$$\overline{\mathrm{Bin}}\left(\frac{k}{m}, \delta\right) \leq 1 - \exp\left(\frac{-1}{m-k}\left[\ln\binom{m}{k} + \ln\left(\frac{1}{\delta}\right)\right]\right) \tag{8.4}$$

$$\leq \frac{1}{m-k}\left[\ln\binom{m}{k} + \ln\left(\frac{1}{\delta}\right)\right] \tag{8.5}$$

Therefore, these approximations enable us to rewrite the bound of Theorem 17 into the following looser (but somewhat clearer and more conventional) form:

Corollary 19. *For any reconstruction function $\mathcal{R}$ that maps arbitrary subsets of a training set and message strings to classifiers, for any prior distribution $P_{\mathcal{I}}$ of vectors of indices, for any compression set-dependent distribution of messages $P_{\mathcal{M}(\mathbf{z_i})}$, and for any $\delta \in (0, 1]$, we have:*

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall \mathbf{i} \in \mathcal{I}, \forall \sigma \in \mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) \leq \right.$$
$$\left. 1 - \exp\left(\frac{-1}{m-d-k}\left[\ln\binom{m-d}{k} + \ln\left(\frac{1}{P_{\mathcal{I}}(\mathbf{i})P_{\mathcal{M}(\mathbf{z_i})}(\sigma)\delta}\right)\right]\right)\right\} \geq 1 - \delta \tag{8.6}$$

and, consequently:

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall \mathbf{i} \in \mathcal{I}, \forall \sigma \in \mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) \leq \right.$$
$$\left. \frac{1}{m-d-k}\left[\ln\binom{m-d}{k} + \ln\left(\frac{1}{P_{\mathcal{I}}(\mathbf{i})P_{\mathcal{M}(\mathbf{z_i})}(\sigma)\delta}\right)\right]\right\} \geq 1 - \delta \tag{8.7}$$

where $d \stackrel{\text{def}}{=} |\mathbf{i}|$ is the sample compression set size of classifier $\mathcal{R}(\sigma, \mathbf{Z_i})$ and $k \stackrel{\text{def}}{=} |\bar{\mathbf{i}}| R_{\mathbf{z}_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma, \mathbf{Z_i}))$ is the number of training errors that this classifier makes on the examples that are not in the compression set.

It is now quite clear from Corollary 19 that the risk bound of classifier $\mathcal{R}(\sigma, \mathbf{Z_i})$ is small when its compression set size d and its number k of training errors are both much smaller than the number m of training examples. These are uniform bounds over a set of data-dependent classifiers defined by the reconstruction function $\mathcal{R}$. In contrast, VC bounds (Vapnik, 1998) and Rademacher bounds (Mendelson, 2002)) are uniform bounds over a set of functions

[3]The proof for Lemma 18 can easily be obtained along the lines of Langford (2005).

defined *without reference to the training data*. Hence, these latter bounds do not apply naturally to our case.

The bound of Equation 8.6 is very similar to (and slightly tighter than) the recent bound of Marchand and Sokolova (2005). Moreover, it is also a direct improvement on the bound of Theorem 11 because of the more efficient treatment of training errors.

The looser bound of Equation 8.7 is similar to the bounds of Littlestone and Warmuth (1986) and Floyd and Warmuth (1995) when the set $\mathcal{M}$ of all possible messages is independent of the compression set $\mathbf{z_i}$ and when we choose:

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = 1/|\mathcal{M}| \quad \forall \sigma \in \mathcal{M} \tag{8.8}$$

$$P_{\mathcal{I}}(\mathbf{i}) = \binom{m}{|\mathbf{i}|}^{-1} (m+1)^{-1} \quad \forall \mathbf{i} \in \mathcal{I} \tag{8.9}$$

But other choices that give better bounds are clearly possible. For example, in the following sections we will use:

$$P_{\mathcal{I}}(\mathbf{i}) = \binom{m}{|\mathbf{i}|}^{-1} \zeta(|\mathbf{i}|) \text{ with } \zeta(a) \stackrel{\text{def}}{=} \frac{6}{\pi^2}(a+1)^{-2} \quad \forall a \in \mathbb{N} \tag{8.10}$$

which satisfies the constraint of Equation 8.3 since $\sum_{i=1}^{\infty} i^{-2} = \pi^2/6$. This choice for $P_{\mathcal{I}}$ has the advantage that the risk bounds do not deteriorate too rapidly when $|\mathbf{i}|$ increases.

In the next section, we show how we can apply the risk bounds of Theorem 17 and Corollary 19 to the SCM. For this task, we will provide choices for the distribution of messages $P_{\mathcal{M}(\mathbf{z_i})}$ which are more appropriate than the simplest choice given by Equation 8.8. Indeed, we feel that it is important to allow the set of messages to depend on the sample compression $\mathbf{z_i}$ since it is conceivable that for some $\mathbf{z_i}$, very little extra information may be needed to identify the classifier whereas for some other $\mathbf{z_i}$, more information may be needed. Without such a dependency on $\mathbf{z_i}$, the set of possible messages $\mathcal{M}$ would be unnecessarily large and would loosen the risk bound. But, more importantly, the risk bound would not depend on the particular message σ used. However, we feel that it is important for learning algorithms to be able to trade-off the complexity (or information content) of $\mathbf{i}$ with the complexity of σ. Hence, a good risk bound should somehow indicate what the proper trade-off should be.

8.3 Application to the Set Covering Machine

Recall that the task of the SCM is to construct the smallest possible conjunction of (Boolean-valued) features. We discuss here only the conjunction case. The disjunction case is treated similarly just by exchanging the role of the positive with the negative examples.

For the case of *data-dependent balls*, each feature is identified by a training example, called a *center* $(\mathbf{x}_c, y_c)$, and a radius ρ. Given any metric d, the output $h(\mathbf{x})$ on any input example $\mathbf{x}$ of such a feature is given by:

$$h(\mathbf{x}) = \begin{cases} y_c & \text{if } d(\mathbf{x}, \mathbf{x}_c) \leq \rho \\ -y_c & \text{otherwise} \end{cases}$$

8.3.1 Coding Each Radius with a Training Example

The original formulation of the SCM uses another training example $\mathbf{x}_b$, called a *border point*, to code for the radius so that $\rho = d(\mathbf{x}_c, \mathbf{x}_b)$. In this case, recall that:

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \zeta(b(\sigma)) \cdot \binom{p(\mathbf{z_i})}{b(\sigma)}^{-1} \tag{8.11}$$

since, in that case, we have for any compression set $\mathbf{z_i}$:

$$\sum_{\sigma \in \mathcal{M}(\mathbf{z_i})} P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \sum_{b=0}^{p(\mathbf{z_i})} \zeta(b) \sum_{\sigma : b(\sigma) = b} \binom{p(\mathbf{z_i})}{b(\sigma)}^{-1} \leq 1$$

With this distribution $P_{\mathcal{M}(\mathbf{z_i})}$, the risk bound of Theorem 17 is tighter than the *Sample compression Bound* of Chapter 4. This is because of the more efficient treatment of the training errors made by using the binomial tail inversion.

8.3.2 Coding Each Radius with a Small Message String

Another alternative is to code each radius value by a message string having the fewest number of bits. In this case, no border points are used and the compression set only consists of ball centers. Consequently, the risk bounds of Theorem 17 and Corollary 19 will be smaller for classifiers described by this method provided that we do not use too many bits to code each radius. We expect that this will be the case whenever there exists a large interval $[r_1, r_2]$ (*i.e.*, a margin) of radius values such that no training examples are present between the two concentric spheres, centered on $\mathbf{x}_c$, with radius r_1 and r_2. The best radius value in that case will be the one that has the shortest code. A similar idea was applied by von Luxburg, Bousquet, and Schölkopf (2004) for coding the maximum-margin hyperplane solution for support vector machines.

Hence, consider the problem of coding a radius value $r \in [r_1, r_2] \subset [0, R]$ where R is some predefined value that cannot be exceeded and where $[r_1, r_2]$ is an interval of "equally good"

radius values[4]. We propose the following diadic coding scheme for the identification of a radius value that belongs to that interval. Let l be the number of bits that we use for the code. We adopt the convention that a code of $l = 0$ bits specifies the radius value $R/2$. A code of $l = 1$ bit either specifies the value $R/4$ (when the bit is 0) or the value $3R/4$ (when the bit is 1). A code of $l = 2$ specifies one of the following values: $R/8, 3R/8, 5R/8, 7R/8$. Hence, a code of l bits specifies one value among the set Λ_l of radius values:

$$\Lambda_l \stackrel{\text{def}}{=} \left\{ \frac{2j-1}{2^{l+1}} R \right\}_{j=1}^{2^l}$$

Given an interval $[r_1, r_2] \subset [0, R]$ of radius values, we take the smallest number l of bits such that there exists a radius value in Λ_l that falls in the interval $[r_1, r_2]$. In this way, we will need at most $\lfloor \log_2(R/(r_2 - r_1)) \rfloor$ bits to obtain a radius value that falls in $[r_1, r_2]$.

Hence, to specify the radius for each center of a compression set, we need to specify the number l of bits and a l-bit string s that identifies one of the radius values in Λ_l. Therefore, the message string σ sent to the reconstruction function $\mathcal{R}$, for a compression set $\mathbf{z_i}$, consists of the set of pairs (l_i, s_i) of numbers needed to identify the radius of each center $i \in \mathbf{i}$. The risk bound does not depend on how we actually code σ (for some receiver). It only depends on the a priori probabilities assigned to each possible realization of σ. We choose the following distribution:

$$\begin{aligned} P_{\mathcal{M}(\mathbf{z_i})}(\sigma) &\stackrel{\text{def}}{=} P_{\mathcal{M}(\mathbf{z_i})}(l_1, s_1, \ldots, l_{|\mathbf{i}|}, s_{|\mathbf{i}|}) \\ &= \prod_{i \in \mathbf{i}} \zeta(l_i) \cdot 2^{-l_i} \end{aligned} \tag{8.12}$$

where $\zeta(l_i)$ is given by Equation 8.10.

Note that by giving equal *a priori* probability to each of the 2^{l_i} strings s_i of length l_i, we give no preference to any radius value in Λ_{l_i} once we have chosen a scale R that we believe is appropriate. The distribution ζ that we have chosen for each string length l_i has the advantage of decreasing slowly so that the risk bound does not deteriorate to rapidly as l_i increases. Other choices are clearly possible.

By comparing the risk bounds of Corollary 19 for the two possible choices we have for coding each radius (either with an example or with a message string), we notice that it should be preferable to code explicitly a radius value with a string whenever we use a number l of bits less than $\log_2 m$ (roughly). Hence, this will be the case whenever there exists an interval $[r_1, r_2]$ of "good" radius values such that $(r_2 - r_1)/R \gtrapprox 1/m$.

[4]By a "good" radius value, we mean a radius value for a ball that would cover many negative examples and very few positive examples (see the learning algorithm).

Finally, we emphasize that the risk bounds of Theorem 17 and Corollary 19, used in conjunction with the distribution of messages given by Equation 8.12, provides a guide for choosing the appropriate tradeoff between sparsity (the inverse of the size of the compression set) and margin (the inverse of the expected length of the message string). Indeed, the risk bound for an SCM with a decision surface having a large margin of separation (small l_is) may be smaller than the risk bound of a sparser SCM having a smaller margin (large l_is).

The Learning Algorithm

Ideally, we would like to find a conjunction of balls that minimizes the risk bound of Theorem 17 with the distribution given by Equation 8.12. Unfortunately, this cannot be done efficiently in all cases since this problem is at least as hard as the (NP-complete) minimum set cover problem (Marchand and Shawe-Taylor, 2002) as discussed before. However, we can make use of the *set covering greedy heuristic*.

We say that a ball *covers* an example iff it assigns `-1` to that example. The set covering greedy heuristic simply consists of using a ball that covers the largest number of negative examples (without making any errors on the positives), remove these negative covered examples and repeat until all the negative examples are covered. Marchand and Shawe-Taylor (2002) have modified this heuristic by incorporating the possibility of making training errors if the final classifier is much smaller as we showed in Chapter 4. Recall that it works as follows. Let N be the set of negative examples and P be the set of positive examples. We start with $N' = N$ and $P' = P$. Let Q_i be the subset of N' covered by ball i and let R_i be the subset of P' covered by ball i. We choose the ball i that maximizes the *utility* U_i defined as:

$$U_i \stackrel{\text{def}}{=} |Q_i| - p \cdot |R_i| \tag{8.13}$$

where p is the *penalty* suffered by covering (and hence, misclassifying) a positive example. Once we have found a ball maximizing U_i, we update $N' = N' - Q_i$ and $P' = P' - R_i$ and repeat to find the next ball until either $N' = \emptyset$ or the maximum number v of balls has been reached (early stopping the greedy).

Here we first modify this heuristic by allowing a maximum number of bits l^* that can be used for coding the radius of each ball. Classifiers obtained with a small value of l^* will, on average, have a large separating margin. Moreover, for this new learning algorithm, the distribution of messages given by Equation 8.12 is defined for a fixed value of R (the "predefined radius value that cannot be exceeded"). Hence, in this case, R should be chosen

from the *definition* of each input attribute *without observing the data*. Consequently, this will generally force *each ball* of the classifier to use a large number of bits for its radius value; otherwise the final classifier is likely to make numerous training errors. We have therefore used the following scheme to choose R *from the training data*. We first choose a value R^* from the definition of each input attribute (without observing the data). This could be $R^* = \sqrt{n}$ for the case of n $\{0,1\}$-valued attributes. Then, we consider t equally-spaced values for R in the interval $]0, R^*]$. The message string σ described in Section 8.3.2 is then just preceded by the index to one of these t possible values. The value of R referred to by this index will then be used for *every ball* of the classifier. For this extra part of the message, we have assigned equal probability to each of the t possible values for R. With this scheme, we only need to multiply $P_{\mathcal{M}(\mathbf{Z_i})}(\sigma)$ of Equation 8.12 by $1/t$. Nevertheless, this introduces one more adjustable parameter in the learning algorithm: the value of R.[5] Therefore, p, v, l^*, and R are the "learning parameters" that our heuristic uses to generate a set of classifiers. At the end, we can use the bound of Theorem 17 to select the best classifier. Another alternative is to determine the best parameter values by cross-validation.

8.4 Empirical Results on Natural Data

We have compared the new learning algorithm (called here SCM2), that codes each ball radius with a message string, with the old algorithm (called here SCM1), that codes each radius with a training example. Both of these algorithms were also compared with the support vector machine (SVM) equipped with a RBF kernel of variance $1/\gamma$ and a soft margin parameter C. Each SCM algorithm used the L_2 metric since this is the metric present in the argument of the RBF kernel.

Each algorithm was tested on the UCI data sets of Tables 8.1, 8.2 and 8.3. Each data set was randomly split in two parts. About half of the examples were used for training and the remaining examples were used for testing. The corresponding values for these numbers of examples are given in the "train" and "test" columns of Tables 8.1, 8.2 and 8.3. The learning parameters of all algorithms were determined from the training set *only*. The parameters C and γ for the SVM were determined by the 5-fold cross validation (CV) method performed on the training set. The parameters that gave the smallest 5-fold CV error were then used to train the SVM on the whole training set and the resulting classifier was then run on the testing set. Exactly the same method (with the same 5-fold split) was used to determine the

[5] We have used $t \approxeq 30$ different values of R in our experiments.

learning parameters of both SCM1 and SCM2. These results are referred to (in Tables 8.2 and 8.3) as SCM1-cv and SCM2-cv. In addition to this, we have compared this 5-fold CV model selection method with a model selection method that uses the risk bound 8.6 of Corollary 19 to select the best SCM classifier obtained from the *same* possible choices of the learning parameters that we have used for the 5-fold CV method[6]. The SCM that minimizes the risk bound (computed from the training set) was then run on the testing set. These results are referred to (in Tables 8.2 and 8.3) as SCM1-b and SCM2-b. For SCM1, the risk bound was used in conjunction with the distribution of messages given by Equation 8.11. For SCM2, the risk bound was used in conjunction with the distribution of messages given by Equation 8.12.

Table 8.1: SVM Results on UCI Datasets.

Data Set			**SVM results**			
Name	**train**	**test**	**C**	γ	**SVs**	**errs**
breastw	343	340	1	0.1	38	15
bupa	170	175	2	3.0	169	66
credit	353	300	100	0.25	282	51
Glass	107	107	10	3.0	51	29
haberman	144	150	2	0.5	81	39
Heart	150	147	1	3.0	64	26
pima	400	368	0.5	0.02	241	96
USvotes	235	200	1	0.02	53	13

The "SVs" column of the SVM results refers to the number of support vectors present in the final classifier. The "errs" column, for all learning algorithms, refers to the number of classification errors obtained on the testing set. Finally, the "b" and "l^*" columns of the SCM results refer, respectively, to the number of balls and the maximum number of bits used by the final classifier.

8.5 Conclusion and Outlook

In this Chapter, we have proposed a new representation for the SCM that uses two distinct sources of information to represent a conjunction of data-dependent balls: a *compression set* to specify the center of each ball and a *message string* to encode the radius value of

[6]It consists of an *exhaustive* list of possible values for (p, l^*, v).

Table 8.2: SCM1 Results on UCI Datasets.

Data Set			SCM1-cv		SCM1-b	
Name	**train**	**test**	**b**	**errs**	**b**	**errs**
breastw	343	340	2	11	1	12
bupa	170	175	2	71	2	70
credit	353	300	12	65	1	57
Glass	107	107	4	20	4	19
haberman	144	150	2	41	1	39
Heart	150	147	1	28	1	23
pima	400	368	1	108	1	105
USvotes	235	200	8	26	3	19

Table 8.3: SCM2 Results on UCI Datasets.

Data Set			SCM2-cv			SCM2-b		
Name	**train**	**test**	**b**	l^*	**errs**	**b**	l^*	**errs**
breastw	343	340	1	3	12	1	1	12
bupa	170	175	2	7	69	11	7	67
credit	353	300	11	6	49	8	5	46
Glass	107	107	7	6	19	3	5	18
haberman	144	150	8	2	36	2	2	37
Heart	150	147	1	2	24	1	2	23
pima	400	368	4	1	107	13	5	103
USvotes	235	200	7	3	19	4	2	15

each ball. Moreover, we have proposed a general data-compression risk bound that depends explicitly on these two information sources. This bound therefore exhibits a non trivial trade-off between sparsity (the inverse of the compression set size) and the margin (the inverse of the message length) that classifiers should attempt to optimize on the training data. We have also proposed a new learning algorithm for the SCM where the learner can control the amount of trade-off between the sparsity of the classifier and the magnitude of its separating margin. Compared to the algorithm of Marchand and Shawe-Taylor (2002), our experiments on natural data sets indicate that this new learning algorithm generally produces classifiers having a larger separating margin at the expenses of having more balls. The generalization error of classifiers produced by the new algorithm was generally slightly better. Finally, the

proposed data-compression risk bound seems to be an effective guide for choosing the proper margin-sparsity trade-off of a classifier.

Note, however, the SCM2 learning algorithm depends on the *a priori* distance scale R which in most cases, if not all, is unknown and this dependency limits the algorithm to some extent. In the next Chapter, we investigate another approach based on the PAC-Bayes theory to propose a learning algorithm for SCM that overcomes this limitation while still performing a non-trivial margin-sparsity trade-off.

CHAPTER 9

A PAC-Bayes approach to the Set Covering Machine

The margin-sparsity trade-off based algorithm for the Set Covering Machine with data-dependent balls that we proposed in the last Chapter suffered from the issue of choosing an a priori distance scale R for the balls' radii. In this chapter, we propose an alternative algorithm for the same based on the PAC-Bayes theory making the risk bound depend more explicitly on margin and sparsity. The results appeared in part in:

F. Laviolette, M. Marchand and M. Shah. A PAC-Bayes Approach to the Set Covering Machine. In *Advances in Neural Information Processing Systems 18,* (Proceedings of NIPS 2005), 731–738, MIT-Press, Cambridge, MA, USA.

9.1 Introduction

In the last chapter, we investigated if better SCM's could be found by optimizing a non-trivial function that depends on both the sparsity of the classifier and the magnitude of its separating margin. Our main result was a general data-compression risk bound that applies to any algorithm producing classifiers represented by two complementary sources of information: a subset of the training set, called the *compression set*, and a *message string* that code the additional information needed to identify a classifier from the compression set. Based on this, we proposed a new algorithm for the SCM where the information string was used to encode radius values for data-dependent balls and, consequently, the location of the decision surface of the classifier. Since a small message string is sufficient when large regions of equally good radius values exist for balls, the data compression risk bound applied to this version of the SCM exhibits, indirectly, a non trivial-margin sparsity trade-off. However, the proposed algorithm for the SCM suffered from the fact that the radius values, used in the final classifier, depends on a *a priori* chosen distance scale R.

In this chapter, we use a new PAC-Bayes approach, that applies to the sample-compression setting, and present a new learning algorithm for the SCM that does not suffer from the previous scaling problem. Moreover, we propose a risk bound that depends more explicitly on the margin and which is also minimized by classifiers achieving a non trivial margin-sparsity

trade-off.

We will follow the same notational conventions as before throughout this chapter.

9.2 A PAC-Bayes Risk Bound

Recall that the PAC-Bayes approach aims at providing PAC guarantees to "Bayesian" learning algorithms. These algorithms are specified in terms of a *prior distribution* P over a space of classifiers that characterizes our prior belief about good classifiers (before the observation of the data) and a *posterior distribution* Q (over the same space of classifiers) that takes into account the additional information provided by the training data. The "PAC-Bayes theorem" described in Chapter 3, provides a tight upper bound on the risk of a stochastic classifier called the *Gibbs classifier* (see Langford (2005) for a survey).

However, for all these versions of the PAC-Bayes theorem, the prior P must be defined without reference to the training data. Consequently, these theorems cannot be applied to the sample-compression setting where classifiers are partly described by a subset of the training data (as for the case of the SCM).

Again recall that in the sample compression setting, each classifier is described by a subset $S_{\mathbf{i}}$ of the training data, called the *compression set*, and a *message string* σ that represents the additional information needed to obtain a classifier. In other words, in this setting, there exists a reconstruction function $\mathcal{R}$ that outputs a classifier $\mathcal{R}(\sigma, S_{\mathbf{i}})$ when given an arbitrary compression set $S_{\mathbf{i}}$ and a message string σ.

Given a training set S, the compression set $S_{\mathbf{i}} \subseteq S$ is defined by a vector of indices $\mathbf{i} \stackrel{\text{def}}{=} (i_1, \ldots, i_{|\mathbf{i}|})$ that points to individual examples in S. For the case of a conjunction of balls, each $j \in \mathbf{i}$ will point to a training example that is used for a ball center and the message string σ will be the vector $\boldsymbol{\rho}$ of radius values (defined above) that are used for the balls. Hence, given $S_{\mathbf{i}}$ and $\boldsymbol{\rho}$, the classifier obtained from $\mathcal{R}(\boldsymbol{\rho}, S_{\mathbf{i}})$ is just the conjunction $C_{\mathbf{i},\boldsymbol{\rho}}$ defined previously.[1]

Recently, Laviolette and Marchand (2005) have extended the PAC-Bayes theorem to the sample-compression setting. Their proposed risk bound depends on a data-independent prior P and a data-dependent posterior Q that are both defined on $\mathcal{I} \times \mathcal{M}$ where $\mathcal{I}$ denotes the set of the 2^m possible index vectors $\mathbf{i}$ and $\mathcal{M}$ denotes, in our case, the set of possible radius vectors $\boldsymbol{\rho}$. The posterior Q is used by a stochastic classifier, called the *sample-compressed*

[1] We assume that the examples in $S_{\mathbf{i}}$ are ordered as in S so that the kth radius value in $\boldsymbol{\rho}$ is assigned to the kth example in $S_{\mathbf{i}}$.

Gibbs classifier G_Q, defined as follows. Given a training set S and given a new (testing) input example $\mathbf{x}$, a sample-compressed Gibbs classifier G_Q chooses randomly $(\mathbf{i}, \boldsymbol{\rho})$ according to Q to obtain classifier $\mathcal{R}(\boldsymbol{\rho}, S_{\mathbf{i}})$ which is then used to determine the class label of $\mathbf{x}$.

In this chapter, we focus on the case where, given any training set S, the learner returns a Gibbs classifier defined with a posterior distribution Q having all its weight on a single vector $\mathbf{i}$. Hence, a single compression set $S_{\mathbf{i}}$ will be used for the final classifier. However, the radius ρ_i for each $i \in \mathbf{i}$ will be chosen stochastically according to the posterior Q. Hence we consider posteriors Q such that $Q(\mathbf{i}', \boldsymbol{\rho}) = I(\mathbf{i} = \mathbf{i}')Q_{\mathbf{i}}(\boldsymbol{\rho})$ where $\mathbf{i}$ is the vector of indices chosen by the learner. Hence, given a training set S, the true risk $R(G_{Q_{\mathbf{i}}})$ of $G_{Q_{\mathbf{i}}}$ and its empirical risk $R_S(G_{Q_{\mathbf{i}}})$ are defined by

$$R(G_{Q_{\mathbf{i}}}) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\boldsymbol{\rho}\sim Q_{\mathbf{i}}} R(\mathcal{R}(\boldsymbol{\rho}, S_{\mathbf{i}})) \quad ; \quad R_S(G_{Q_{\mathbf{i}}}) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\boldsymbol{\rho}\sim Q_{\mathbf{i}}} R_{S_{\bar{\mathbf{i}}}}(\mathcal{R}(\boldsymbol{\rho}, S_{\mathbf{i}})) \ ,$$

where $\bar{\mathbf{i}}$ denotes the set of indices not present in $\mathbf{i}$. Thus, $\bar{\mathbf{i}} \cap \mathbf{i} = \emptyset$ and $\mathbf{i} \cup \bar{\mathbf{i}} = (1, \ldots, m)$.

In contrast with the posterior Q, the prior P assigns a non zero weight to several vectors $\mathbf{i}$. Let $P_{\mathcal{I}}(\mathbf{i})$ denote the prior probability P assigned to vector $\mathbf{i}$ and let $P_{\mathbf{i}}(\boldsymbol{\rho})$ denote the probability density function associated with prior P given $\mathbf{i}$. The risk bound depends on the Kullback-Leibler divergence $\mathrm{KL}(Q\|P)$ between the posterior Q and the prior P which, in our case, gives

$$\mathrm{KL}(Q_{\mathbf{i}}\|P) = \mathop{\mathbf{E}}_{\boldsymbol{\rho}\sim Q_{\mathbf{i}}} \ln \frac{Q_{\mathbf{i}}(\boldsymbol{\rho})}{P_{\mathcal{I}}(\mathbf{i})P_{\mathbf{i}}(\boldsymbol{\rho})} \ .$$

For these classes of posteriors Q and priors P, the PAC-Bayes theorem of Laviolette and Marchand (2005) reduces to the following simpler version.

Theorem 20 (Laviolette and Marchand (2005)). *Given all our previous definitions, for any prior P and for any $\delta \in (0, 1]$*

$$\Pr_{S\sim D^m} \left(\forall\, Q_{\mathbf{i}} : \ \mathrm{kl}(R_S(G_{Q_{\mathbf{i}}})\|R(G_{Q_{\mathbf{i}}})) \leq \tfrac{1}{m-|\mathbf{i}|} \left[\mathrm{KL}(Q_{\mathbf{i}}\,\|P) + \ln \tfrac{m+1}{\delta}\right] \right) \quad \geq \quad 1 \ - \ \delta \quad ,$$

where

$$\mathrm{kl}(q\|p) \stackrel{\text{def}}{=} q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p} \quad \textit{for } q < p \ .$$

To obtain a bound for $R(G_{Q_{\mathbf{i}}})$ we need to specify $Q_{\mathbf{i}}(\boldsymbol{\rho})$, $P_{\mathcal{I}}(\mathbf{i})$, and $P_{\mathbf{i}}(\boldsymbol{\rho})$.

Since all vectors $\mathbf{i}$ having the same size $|\mathbf{i}|$ are, *a priori*, equally "good", we choose

$$P_{\mathcal{I}}(\mathbf{i}) = \frac{1}{\binom{m}{|\mathbf{i}|}} p(|\mathbf{i}|)$$

for any $p(\cdot)$ such that $\sum_{d=0}^{m} p(d) = 1$. We could choose $p(d) = 1/(m+1)$ for $d \in \{0, 1, \ldots, m\}$ if we have complete ignorance about the size $|\mathbf{i}|$ of the final classifier. But since the risk bound

will deteriorate for large $|\mathbf{i}|$, it is generally preferable to choose, for $p(d)$, a slowly decreasing function of d.

For the specification of $P_{\mathbf{i}}(\boldsymbol{\rho})$, we assume that each radius value, in some predefined interval $[0, R]$, is equally likely to be chosen for each ρ_i such that $i \in \mathbf{i}$. Here R is some "large" distance specified *a priori*. For $Q_{\mathbf{i}}(\boldsymbol{\rho})$, a margin interval $[a_i, b_i] \subseteq [0, R]$ of equally good radius values is chosen by the learner for each $i \in \mathbf{i}$. Hence, we choose

$$P_{\mathbf{i}}(\boldsymbol{\rho}) = \prod_{i \in \mathbf{i}} \frac{1}{R} = \left(\frac{1}{R}\right)^{|\mathbf{i}|} \quad ; \quad Q_{\mathbf{i}}(\boldsymbol{\rho}) = \prod_{i \in \mathbf{i}} \frac{1}{b_i - a_i} \ .$$

Therefore, the Gibbs classifier returned by the learner will draw each radius ρ_i uniformly in $[a_i, b_i]$. A deterministic classifier is then specified by fixing each radius values $\rho_i \in [a_i, b_i]$. It is tempting at this point to choose $\rho_i = (a_i + b_i)/2 \ \forall i \in \mathbf{i}$ (*i.e.*, in the middle of each interval). However, we will see shortly that the PAC-Bayes theorem offers a better guarantee for another type of deterministic classifier.

Consequently, with these choices for $Q_{\mathbf{i}}(\boldsymbol{\rho})$, $P_{\mathcal{I}}(\mathbf{i})$, and $P_{\mathbf{i}}(\boldsymbol{\rho})$, the KL divergence between $Q_{\mathbf{i}}$ and P is given by

$$KL(Q_{\mathbf{i}} \| P) = \ln \binom{m}{|\mathbf{i}|} + \ln \left(\frac{1}{p(|\mathbf{i}|)}\right) + \sum_{i \in \mathbf{i}} \ln \left(\frac{R}{b_i - a_i}\right) \ .$$

Notice that the KL divergence is small for small values of $|\mathbf{i}|$ (whenever $p(|\mathbf{i}|)$ is not too small) and for large margin values $(b_i - a_i)$. Hence, the KL divergence term in Theorem 20 favors both sparsity (small $|\mathbf{i}|$) and large margins. Hence, in practice, the minimum might occur for some $G_{Q_{\mathbf{i}}}$ that sacrifices sparsity whenever larger margins can be found.

Since the posterior Q is identified by $\mathbf{i}$ and by the intervals $[a_i, b_i] \ \forall i \in \mathbf{i}$, we will now refer to the Gibbs classifier $G_{Q_{\mathbf{i}}}$ by $G^{\mathbf{i}}_{\mathbf{ab}}$ where $\mathbf{a}$ and $\mathbf{b}$ are the vectors formed by the unions of a_is and b_is respectively. To obtain a risk bound for $G^{\mathbf{i}}_{\mathbf{ab}}$, we need to find a closed-form expression for $R_S(G^{\mathbf{i}}_{\mathbf{ab}})$. For this task, let $U[a, b]$ denote the uniform distribution over $[a, b]$ and let $\sigma^i_{a,b}(\mathbf{x})$ be the probability that a ball with center $\mathbf{x}_i$ assigns to $\mathbf{x}$ the class label y_i when its radius ρ is drawn according to $U[a, b]$:

$$\sigma^i_{a,b}(\mathbf{x}) \stackrel{\text{def}}{=} \Pr_{\rho \sim U[a,b]} \left(h_{i,\rho}(\mathbf{x}) = y_i\right) = \begin{cases} 1 & \text{if} \quad d(\mathbf{x}, \mathbf{x}_i) \leq a \\ \frac{b - d(\mathbf{x}, \mathbf{x}_i)}{b - a} & \text{if} \quad a \leq d(\mathbf{x}, \mathbf{x}_i) \leq b \\ 0 & \text{if} \quad d(\mathbf{x}, \mathbf{x}_i) \geq b_i \ . \end{cases}$$

Therefore,

$$\zeta^i_{a,b}(\mathbf{x}) \stackrel{\text{def}}{=} \Pr_{\rho \sim U[a,b]} \left(h_{i,\rho}(\mathbf{x}) = 1\right) = \begin{cases} \sigma^i_{a,b}(\mathbf{x}) & \text{if} \quad y_i = 1 \\ 1 - \sigma^i_{a,b}(\mathbf{x}) & \text{if} \quad y_i = 0 \ . \end{cases}$$

Now let $G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x})$ denote the probability that $C_{\mathbf{i},\boldsymbol{\rho}}(\mathbf{x}) = 1$ when each $\rho_i \in \boldsymbol{\rho}$ are drawn according to $U[a_i, b_i]$. We then have

$$G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}) \;=\; \prod_{i\in\mathbf{i}} \zeta^i_{a_i,b_i}(\mathbf{x}) \;.$$

Consequently, the risk $R_{(\mathbf{x},y)}(G^{\mathbf{i}}_{\mathbf{ab}})$ on a single example $(\mathbf{x}, y)$ is given by $G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x})$ if $y = 0$ and by $1 - G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x})$ otherwise. Therefore

$$R_{(\mathbf{x},y)}(G^{\mathbf{i}}_{\mathbf{ab}}) \;=\; y(1 - G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x})) + (1-y)G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}) \;=\; (1-2y)(G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}) - y) \;.$$

Hence, the empirical risk $R_S(G^{\mathbf{i}}_{\mathbf{ab}})$ of the Gibbs classifier $G^{\mathbf{i}}_{\mathbf{ab}}$ is given by

$$R_S(G^{\mathbf{i}}_{\mathbf{ab}}) \;=\; \frac{1}{m - |\mathbf{i}|} \sum_{j\in\bar{\mathbf{i}}} (1-2y_j)(G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}_j) - y_j) \;.$$

From this expression we see that $R_S(G^{\mathbf{i}}_{\mathbf{ab}})$ is small when $G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}_j) \to y_j \;\forall j \in \bar{\mathbf{i}}$. Training points where $G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}_j) \approx 1/2$ should therefore be avoided.

The PAC-Bayes theorem below provides a risk bound for the Gibbs classifier $G^{\mathbf{i}}_{\mathbf{ab}}$. Since the Bayes classifier $B^{\mathbf{i}}_{\mathbf{ab}}$ just performs a majority vote under the same posterior distribution as the one used by $G^{\mathbf{i}}_{\mathbf{ab}}$, we have that $B^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}) = 1$ iff $G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}) > 1/2$. From the above definitions, note that the decision surface of the Bayes classifier, given by $G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}) = 1/2$, differs from the decision surface of classifier $C_{\mathbf{i}\boldsymbol{\rho}}$ when $\rho_i = (a_i + b_i)/2 \;\forall i \in \mathbf{i}$. In fact there does not exists any classifier $C_{\mathbf{i}\boldsymbol{\rho}}$ that has the same decision surface as Bayes classifier $B^{\mathbf{i}}_{\mathbf{ab}}$. From the relation between $B^{\mathbf{i}}_{\mathbf{ab}}$ and $G^{\mathbf{i}}_{\mathbf{ab}}$, it also follows that $R_{(\mathbf{x},y)}(B^{\mathbf{i}}_{\mathbf{ab}}) \leq 2R_{(\mathbf{x},y)}(G^{\mathbf{i}}_{\mathbf{ab}})$ for any $(\mathbf{x}, y)$. Consequently, $R(B^{\mathbf{i}}_{\mathbf{ab}}) \leq 2R(G^{\mathbf{i}}_{\mathbf{ab}})$. Hence, we have the following theorem.

Theorem 21. *Given all our previous definitions, for any $\delta \in (0, 1]$, for any p satisfying $\sum_{d=0}^{m} p(d) = 1$, and for any fixed distance value R, we have:*

$$\Pr_{S\sim D^m}\left(\forall \mathbf{i}, \mathbf{a}, \mathbf{b}\colon R(G^{\mathbf{i}}_{\mathbf{ab}}) \leq \sup\left\{\epsilon\colon \mathrm{kl}(R_S(G^{\mathbf{i}}_{\mathbf{ab}})\|\epsilon) \leq \frac{1}{m-|\mathbf{i}|}\left[\ln\binom{m}{|\mathbf{i}|} + \right.\right.\right.$$
$$\left.\left.\left. + \ln\left(\frac{1}{p(|\mathbf{i}|)}\right) + \sum_{i\in\mathbf{i}} \ln\left(\frac{R}{b_i - a_i}\right) + \ln\frac{m+1}{\delta}\right]\right\}\right) \geq 1 - \delta \;.$$

Furthermore: $R(B^{\mathbf{i}}_{\mathbf{ab}}) \leq 2R(G^{\mathbf{i}}_{\mathbf{ab}}) \quad \forall \mathbf{i}, , \mathbf{a}, \mathbf{b}$.

Recall that the KL divergence is small for small values of $|\mathbf{i}|$ (whenever $p(|\mathbf{i}|)$ is not too small) and for large margin values $(b_i - a_i)$. Furthermore, the Gibbs empirical risk $R_S(G^{\mathbf{i}}_{\mathbf{ab}})$ is small when the training points are located far away from the Bayes decision surface $G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}) = 1/2$ (with $G^{\mathbf{i}}_{\mathbf{ab}}(\mathbf{x}_j) \to y_j \;\forall j \in \bar{\mathbf{i}}$). *Consequently, the Gibbs classifier with the smallest guarantee of risk should perform a non trivial margin-sparsity tradeoff.*

9.3 A Soft Greedy Learning Algorithm

Theorem 21 suggests that the learner should try to find the Bayes classifier $B^{\mathbf{i}}_{\mathbf{ab}}$ that uses a small number of balls (*i.e.*, a small $|\mathbf{i}|$), each with a large separating margin $(b_i - a_i)$, while keeping the empirical Gibbs risk $R_S(G^{\mathbf{i}}_{\mathbf{ab}})$ at a low value. To achieve this goal, we have adapted the greedy algorithm for the set covering machine (SCM).

In our case, however, we need to keep the Gibbs risk on S low instead of the risk of a deterministic classifier. Since the Gibbs risk is a "soft measure" that uses the piece-wise linear functions $\sigma^i_{a,b}$ instead of "hard" indicator functions, we need a "softer" version of the utility function U_i. Indeed, a negative example that falls in the linear region of a $\sigma^i_{a,b}$ is in fact partly covered. Following this observation, let $\mathbf{k}$ be the vector of indices of the examples that we have used as ball centers so far for the construction of the classifier. Let us first define the *covering value* $\mathcal{C}(G^{\mathbf{k}}_{\mathbf{ab}})$ of $G^{\mathbf{k}}_{\mathbf{ab}}$ by the "amount" of negative examples assigned to class 0 by $G^{\mathbf{k}}_{\mathbf{ab}}$:

$$\mathcal{C}(G^{\mathbf{k}}_{\mathbf{ab}}) \stackrel{\text{def}}{=} \sum_{j \in \overline{\mathbf{k}}} (1 - y_j) \left[1 - G^{\mathbf{k}}_{\mathbf{ab}}(\mathbf{x}_j)\right] \ .$$

We also define the *positive-side error* $\mathcal{E}(G^{\mathbf{k}}_{\mathbf{ab}})$ of $G^{\mathbf{k}}_{\mathbf{ab}}$ as the "amount" of positive examples assigned to class 0 :

$$\mathcal{E}(G^{\mathbf{k}}_{\mathbf{ab}}) \stackrel{\text{def}}{=} \sum_{j \in \overline{\mathbf{k}}} y_j \left[1 - G^{\mathbf{k}}_{\mathbf{ab}}(\mathbf{x}_j)\right] \ .$$

We now want to add another ball, centered on an example with index i, to obtain a new vector $\mathbf{k}'$ containing this new index in addition to those present in $\mathbf{k}$. Hence, we now introduce the *covering contribution* of ball i (centered on $\mathbf{x}_i$) as

$$\begin{aligned} \mathcal{C}^{\mathbf{k}}_{\mathbf{ab}}(i) &\stackrel{\text{def}}{=} \mathcal{C}(G^{\mathbf{k}'}_{\mathbf{a'b'}}) - \mathcal{C}(G^{\mathbf{k}}_{\mathbf{ab}}) \\ &= (1 - y_i) \left[1 - \zeta^i_{a_i,b_i}(\mathbf{x}_i)\, G^{\mathbf{k}}_{\mathbf{ab}}(\mathbf{x}_i)\right] + \sum_{j \in \overline{\mathbf{k}'}} (1 - y_j) \left[1 - \zeta^i_{a_i,b_i}(\mathbf{x}_j)\right] G^{\mathbf{k}}_{\mathbf{ab}}(\mathbf{x}_j) \ , \end{aligned}$$

and the *positive-side error contribution* of ball i as

$$\begin{aligned} \mathcal{E}^{\mathbf{k}}_{\mathbf{ab}}(i) &\stackrel{\text{def}}{=} \mathcal{E}(G^{\mathbf{k}'}_{\mathbf{a'b'}}) - \mathcal{E}(G^{\mathbf{k}}_{\mathbf{ab}}) \\ &= y_i \left[1 - \zeta^i_{a_i,b_i}(\mathbf{x}_i)\, G^{\mathbf{k}}_{\mathbf{ab}}(\mathbf{x}_i)\right] + \sum_{j \in \overline{\mathbf{k}'}} y_j \left[1 - \zeta^i_{a_i,b_i}(\mathbf{x}_j)\right] G^{\mathbf{k}}_{\mathbf{ab}}(\mathbf{x}_j) \ . \end{aligned}$$

Typically, the covering contribution of ball i should increase its "utility" and its positive-side error should decrease it. Hence, we define the *utility* $U^{\mathbf{k}}_{\mathbf{ab}}(i)$ *of adding ball* i *to* $G^{\mathbf{k}}_{\mathbf{ab}}$ as

$$U^{\mathbf{k}}_{\mathbf{ab}}(i) \stackrel{\text{def}}{=} \mathcal{C}^{\mathbf{k}}_{\mathbf{ab}}(i) - p\mathcal{E}^{\mathbf{k}}_{\mathbf{ab}}(i) \ ,$$

where parameter p represents the *penalty* of misclassifying a positive example. For a fixed value of p, the "soft greedy" algorithm simply consists of adding, to the current Gibbs classifier, a ball with maximum added utility until either the maximum number of possible features (balls) has been reached or that all the negative examples have been (totally) covered. It is understood that, during this soft greedy algorithm, we can remove an example $(\mathbf{x}_j, y_j)$ from S whenever it is totally covered. This occurs whenever $G^{\mathbf{k}}_{\mathbf{ab}}(\mathbf{x}_j) = 0$.

The term $\sum_{i \in \mathbf{i}} \ln(R/(b_i - a_i))$, present in the risk bound of Theorem 21, favors "soft balls" having large margins $b_i - a_i$. Hence, we introduce a *margin parameter* $\gamma \geq 0$ that we use as follows. At each greedy step, we first search among balls having $b_i - a_i = \gamma$. Once such a ball, of center $\mathbf{x}_i$, having maximum utility has been found, we try to increase further its utility be searching among all possible values of a_i and $b_i > a_i$ while keeping its center $\mathbf{x}_i$ fixed.[2] Both p and γ will be chosen by cross validation on the training set.

9.3.1 Time Complexity Analysis

We conclude this section with an analysis of the running time of this soft greedy learning algorithm for fixed p and γ. For each potential ball center, we first sort the $m - 1$ other examples with respect to their distances from the center in $O(m \log m)$ time. Then, for this center $\mathbf{x}_i$, the set of a_i values that we examine are those specified by the distances (from $\mathbf{x}_i$) of the $m - 1$ sorted examples.[3] Since the examples are sorted, it takes time $\in O(km)$ to compute the covering contributions and the positive-side error *for all* the $m - 1$ values of a_i. Here k is the largest number of examples falling into the margin. We are always using small enough γ values to have $k \in O(\log m)$ since, otherwise, the results are terrible. It therefore takes time $\in O(m \log m)$ to compute the utility values of all the $m - 1$ different balls of a given center. This gives a time $\in O(m^2 \log m)$ to compute the utilities for all the possible m centers. Once a ball with a largest utility value has been chosen, we then try to increase further its utility by searching among $O(m^2)$ pair values for (a_i, b_i). We then remove the examples covered by this ball and repeat the algorithm on the remaining examples. It is well known that greedy algorithms of this kind have the following guarantee: if there exist r balls that covers all the m examples, the greedy algorithm will find at most $r \ln(m)$ balls. Since we almost always have $r \in O(1)$, the running time of the whole algorithm will almost always be $\in O(m^2 \log^2(m))$.

[2] The possible values for a_i and b_i are defined by the location of the training points.

[3] Recall that for each value of a_i, the value of b_i is set to $a_i + \gamma$ at this stage.

9.4 Empirical Results on Natural Data

Table 9.1: SVM and SCM Results on UCI Datasets.

Data Set			SVM results				SCM	
Name	train	test	C	γ	SVs	errs	b	errs
breastw	343	340	1	5	38	15	2	12
bupa	170	175	2	0.17	169	66	2	62
credit	353	300	100	2	282	51	12	58
glass	107	107	10	0.17	51	29	4	22
haberman	144	150	2	1	81	39	2	39
heart	150	147	1	0.17	64	26	1	23
pima	400	368	0.5	0.02	241	96	1	108
USvotes	235	200	1	25	53	13	8	27

We have compared the new PAC-Bayes learning algorithm (called here SCM-PB), with the old algorithm (called here SCM). Both of these algorithms were also compared with the SVM equipped with a RBF kernel of variance γ and a soft margin parameter C. Each SCM algorithm used the L_2 metric since this is the metric present in the argument of the RBF kernel. However, in contrast with Laviolette, Marchand and Shah (2005), each SCM was constrained to use only balls having centers of the same class (negative for conjunctions and positive for disjunctions).

Each algorithm was tested on the UCI data sets of Tables 9.1 and 9.2. Each data set was randomly split in two parts. About half of the examples was used for training and the remaining set of examples was used for testing. The corresponding values for these numbers of examples are given in the "train" and "test" columns of Tables 9.1 and 9.2. The learning parameters of all algorithms were determined from the training set *only*. The parameters C and γ for the SVM were determined by the 5-fold cross validation (CV) method performed on the training set. The parameters that gave the smallest 5-fold CV error were then used to train the SVM on the whole training set and the resulting classifier was then run on the testing set. Exactly the same method (with the same 5-fold split) was used to determine the learning parameters of both SCM and SCM-PB.

The SVM results are reported in Table 9.1 where the "SVs" column refers to the number of support vectors present in the final classifier. The "errs" column refers to the number of classification errors obtained on the testing set. This notation is used also for all the SCM results reported in Tables 9.1 and 9.2. In addition to this, the "b" and "γ" columns in

Table 9.2: PAC-Bayes-SCM Results on UCI Datasets.

Data Set			SCM-PB		
Name	train	test	b	γ	errs
breastw	343	340	4	0.08	10
bupa	170	175	6	0.1	67
credit	353	300	11	0.09	55
glass	107	107	16	0.04	19
haberman	144	150	1	0.2	38
heart	150	147	1	0	28
pima	400	368	7	0.1	105
USvotes	235	200	18	0.14	12

Table 9.1 refer, respectively, to the number of balls and the margin parameter (divided by the average distance between the positive and the negative examples). The results reported for SCM-PB (in Table 9.1) refer to the Bayes classifier only. The results for the Gibbs classifier are similar. We observe that, except for bupa and heart, the generalization error of SCM-PB was always smaller than SCM. However, the only significant difference occurs on USvotes. We also observe that SCM-PB generally sacrifices sparsity (compared to SCM) to obtain some margin $\gamma > 0$.

CHAPTER 10

Conclusions and Future Work

The initial efforts in this work started with the intention to investigate the extensibility of the basic Set Covering Machine framework and to see if a simple learning bias of *learning conjunctions or disjunctions of (possibly data-dependent) features* can lead to classifiers that are "general",[1] "practical"[2] and "have provable performance guarantees". To this end, we have been able indeed to find evidence that demonstrate these. We have seen extensions to the SCM framework across different sets of features (*Half-Spaces* and *Rays*) and theoretical frameworks that can lead to better performance practically and tight guarantees analytically.

In quantitative terms, we have seen for instance, that sometimes features such as data-dependent Half-Spaces can lead to classifiers that yield better classification accuracy as well as sparser solutions (as compared to initially proposed SCM with data-dependent balls). Moreover, the resource requirements of the algorithm are within pragmatic bounds. Also, as we saw, tight sample compression based guarantees can be provided over generalization performance of the classifier in terms of the data-compression it achieves.

We also demonstrated how the SCM framework can be adapted to high dimensional domains, especially to the gene expression data originating from DNA micro-arrays. The aim was to obtain classifier(s) with high accuracy while making use of a small number of attributes. Our approach of learning conjunctions or disjunctions of simple threshold features called *Rays* that were built on single real valued attributes in conjunction with a PAC-Bayes learning settings yielded a learning algorithm that not only gave us accuracies competitive with the state-of-the-art approaches but also use a minimum number of genes to do so (generally better by a factor of 100).

In addition to these novel algorithms based on the basic SCM framework, we were also able to improve upon the initial Set Covering Machine algorithm itself and provided two alternate algorithms that give better classifiers and tight guarantees.

Hence, we did manage to address our initial concerns that lead us to this research but more important results coming out from this work should be considered in rather general

[1] applicable across various domains

[2] require reasonable amount of resources(time and sample size)

sense of sample compression framework(s). This is because these results are not restricted to the domain of the SCM but rather have significant implications in our overall understanding of the sample compression theory as well as the Machine Learning technology. In this respect, the most important results from this thesis come as follows:

i. A novel feature selection algorithm with provable theoretical guarantee. The algorithm for learning a conjunction or disjunction of *Rays* not only performs feature selection but also comes with a tight PAC-Bayes risk bound.[3] An important implication resulting from this work can be seen in the sense that instead of adapting the current machine learning algorithms to work in high-dimensional domains by coupling them with feature selectors (such as filters or wrappers), we can obtain better learning algorithms if we adopt a more direct approach by designing algorithms that perform such feature selection as a part of the learning process. Although it is difficult to come up with generic algorithms, such algorithms can certainly be thought of in terms of the intended application domain(s). Such task-oriented learning algorithms that integrate feature selection with learning can not only provide good solutions but can also be theoretically justifiable unlike most of the current approaches, lending them credibility.

ii. Another important result coming out from this work is in the form of margin-sparsity trade-off bounds for sample compression based algorithms. It has been believed widely for quite some time now that algorithms should perform a non-trivial margin-sparsity trade-off to output better classifiers. We propose generalization bounds that explicitly depend on *both* the sparsity of a classifier *and* the magnitude of its separating margin. Moreover, we show how both sparsity and margin can be considered as different forms of data compression and exploited by performing a trade-off to yield more general classifiers in Chapters 7, 8 and 9.

We have also shown how sample compression based bounds can provide tighter guarantees to learning algorithms as opposed to data-independent bounds such as VC bounds. This should be further viewed in the light of the results of Floyd and Warmuth (1995) and the recent results of Kuzmin and Warmuth (2005) showing that concept classes, known as maximum classes, of VC dimension d, have a sample compression scheme of size d. For these classes, we can thus use the tighter sample-compression bounds instead of VC bounds. If this can be proven to be true for "any" concept class of VC dimension d (an open problem currently), the sample compression bounds can provide tighter guarantees for learning

[3]It remains to be seen if the bound can guide the model selection process.

algorithms for these classes.

10.1 Future Work

The directions worth investigating in future come from both the novel results as well as limitations of this work:

Just as every approach comes with an inherent cost, our approaches are not free of limitations. However, the existence of such limitations are a must to motivate further research to obtain yet better solutions. Let us discuss some of the main issues in this work. The first is probably the time complexity of the SCM with Half-space learning approach. Unlike the initially proposed SCM algorithms with data-dependent balls as features that requires pairs of examples to be examined to find the best ball in each greedy step, half-spaces require triplets of examples to be examined resulting in an algorithm that is costly in terms of computation. Efforts towards optimizing the algorithm time-wise are hence required. This can not only result in a faster algorithm but also open up doors for further research along the lines that we have done so far with the SCM framework.

Another such issue can be seen in terms of the algorithm for learning conjunction of *Rays*. We indeed have obtained good results for the DNA micro-array datasets and that too with good risk bounds, it however remains to be seen if this approach will work successfully over other domains. As we discussed earlier how task-specific feature selections algorithms can be designed, we can also investigate if this Rays' approach can be further generalized across various domains. An obvious domain that comes to mind is that of text categorization. We have examined the performance of this learning algorithm over the text data obtained from Electronic-Negotiations (Shah, Sokolova and Szpakowicz, 2004, Sokolova et. al., 2005) and during our preliminary experiments we found that the algorithm performs acceptably and yields competitive results with search heuristic based algorithms such as best-first feature selection. However, further empirical evidence is required to conclude this initial observation. Moreover, further investigations are also required for the Occam Razor approach to Rays as given in Chapter 6. This approach can be improved on the lines of the Occam Razor based approach that we proposed for the SCM with balls in Chapter 8, for instance, by putting a threshold on the length of the message strings.

In view of the results that we have obtained in this thesis, the most logical extensions and directions for future work can be viewed as below:

i. *Utilize the risk bounds for model selection*: The first and foremost task ahead would

be to investigate if the risk bounds that we propose can successfully perform model selection. We have seen such instances in the case of SCM with Half-spaces (Chapter 5) as well as for our first margin-sparsity trade-off based approach for the SCM with data-dependent balls (Chapter 8). Moreover, the original algorithm proposed for the SCM also showed encouraging results where the sample compression based bound was utilized for this task. Being able to perform model selection from bound helps us avoid the costly k-fold cross validation based model selection. This can also open doors to other algorithms based on bound minimization.

ii. *Investigate more specific, possibly data-dependent, bounds*: It is widely known that algorithm specific bounds are generally tighter and can lead to more pragmatic limits on the generalization error of classifiers. Hence, we should investigate if we can make our risk bounds more specific, probably exploiting the algorithm's dependency on the training data for instance, to achieve tighter guarantees over the future performance of classifier. Another interesting direction would be to investigate asymmetric loss bounds.

iii. *Extending SCM framework to high dimensional and/or large datasets*: This is another interesting domain worth looking into. Features such as *Rays* have demonstrated competitive performance in high dimensional domains. However, further research is needed in the associated directions to make the overall SCM framework applicable to high dimensional domains as well as very large datasets. These may include optimizing the algorithm for present set of features to make it faster for larger datasets, designing new features to do so, and so on.

iv. *Investigating richer hypothesis classes*: With respect to the class of sample compression framework, it might be worthwhile to look beyond the present hypothesis class of learning conjunction or disjunction of features. Richer classes, such as decision lists or even decision trees, may lead to yet better classifiers. This would however involve addressing corresponding issues too, for instance, the time complexity of the resulting algorithm(s).

With respect to overall Machine Learning technology, the insights obtained from this work can be exploited on a rather general extent to come up with novel learning algorithms for existing frameworks, not to mention the new approaches possible. One such scenario can be seen in terms of the guarantees for decision trees based on the sample compression and Occam's razor settings. Another can be a possible incorporation of explicit margin-sparsity

trade-off capability in existing machine learning algorithms. Even trade-off's in other terms can be thought of for the algorithms that have a bias towards other measures of complexity than sparsity and margin of a classifier.

BIBLIOGRAPHY

Alon U., N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA*, 96:6745–6750.

Ambroise C. and G. J. McLachlan (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *Proc. Natl. Acad. Sci. USA*, 99:6562–6566.

Amsterdam J. (1988). The valiant learning model: Extensions and assessment. Master's thesis, MIT, Department of Electrical Engineering and Computer Science.

Angluin S. and P. Laird (1988). Learning from noisy examples. *Machine Learning*, 2(4): 343–370.

Angluin D. and C. H. Smith (1983). Inductive Inference: Theory and methods. *Comput. Surv.*, 15(3):327–369.

Anthony M. and N. Biggs (1992). *Computational Learning Theory.* Cambridge Tracts in Theoretical Computer Science (30). Cambridge University Press.

Ben-David S. and A. Litman (1998). Combinatorial variability of Vapnik-Chervonenkis classes. *Discrete Applied Mathematics*, 86:3–25.

Bergadano F. and L. Saitta (1989). On the error probability of boolean concept descriptions. In *Proceedings of the 1989 European Working Session on Learning*, 25–35.

Bennett K. P. (1999). Combining support vector and mathematical programming methods for classifications. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods–Support Vector Learning*, MIT Press, Cambridge MA, 307-326.

Bezdek J. C. and L. I. Kuncheva (2001). Nearest Prototype Classifier Designs: An Experimental Study. *Int'l. J. Intelligent Systems*, 16(12):1445–1473.

Bi J., K. P. Bennett, M. Embrechts, K. M. Breneman and M. Song (2003). Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3: 1229-1245.

Blum A. (1992). Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386.

Blum A. (2002). Machine Learning Theory. *Lecture Notes*, Carnegie Mellon University.

Blum A. and J. Langford (2003). PAC-MDL bounds. In *Proceedings of 16th Annual Conference on Learning Theory, COLT 2003,*, *Lecture Notes in Artificial Intelligence*, vol. 2777, 344–357. Springer, Berlin.

Blum A. and R. L. Rivest (1988). Training a three-neuron neural net is NP-Complete. In *Proceedings of the 1988 Workshop on Computational Learning Theory* 9–18, San Mateo, CA.

Blumer A., A. Ehrenfeucht, D. Haussler, and M. K. Warmuth (1987). Occam's Razor. *Information Processing Letters*, 24:377–380.

Blumer A., A. Ehrenfeucht, D. Haussler, and M. K. Warmuth (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965.

Board R. and L. Pitt (1992). On the necessity of the Occam algorithms. *Theoretical Computer Science*, 100:157–184.

Boser B. E., I. M. Guyon and V. N. Vapnik (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 144–152, ACM Press.

Bousquet O. and A. Elisseeff (2001). Algorithmic Stability and Generalization Performance. In *Advances in Neural Information Processing Systems 13*, 196-202, MIT Press.

Bousquet O., S. Boucheron and G. Lugosi (2004). Introduction to Statistical Learning Theory. Advanced Lectures on Machine Learning Lecture Notes in Artificial Intelligence 3176, 169-207. (Eds.) Bousquet, O., U. von Luxburg and G. Rtsch, Springer, Heidelberg, Germany.

Buntine W. (1990). *A Theory of Learning Classification Rules*. PhD thesis, University of Technology, Sydney.

Burges C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, Kluwer Academic Publishers.

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

Chvátal V. (1979). A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4:233–235.

Cooper G. F., C. F. Aliferis, R. Ambrosino, J. Aronis, B. G. Buchanan, R. Caruana, M. J. Fine, C. Glymour, G. Gordon, B. H. Hanusa, J. Janosky, C. Meek, T. Mitchell, T. Richardson and P. Spirtes (1997). An evaluation of machine-learning methods for predicting pneumonia mortality. *Artificial Intelligence in Medicine*, 9:107–138.

Cover T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. on Electronic Computers*, EC-14: 326–334.

Demšar J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30.

Dhagat A. and L. Hellerstein (1994). PAC learning with irrelevant attributes. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 64–74. IEEE Computer Society Press, Los Alamitos, CA.

Dietterich T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1924.

Duda R. O., P. E. Hart, and D. G. Stork (2001). *Pattern Classification, 2nd ed.*, New York: John Wiley and Sons.

Evans W., S. Rajagopalan and U. Vazirani (1993). Choosing a reliable hypothesis. In *Proceedings of the 6th Workshop on Computational Learning Theory*, 269–276. ACM Press, New York, NY.

Floyd S., & M. Warmuth (1995). Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21:269–304.

Freund Y. (1990). Boosting a weak learning algorithm by majority. In *Proceedings of the 3rd Workshop on Computational Learning Theory*, 202–216. Morgan Kaufmann, San Mateo, CA.

Freund Y. (1992). An improved boosting algorithm and its implications on learning complexity. In *Proceedings of the 5th Workshop on Computational Learning Theory*, 391-398. ACM Press, New York, NY.

Furey T. S., N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16:906–914.

Garber M. E., O. G. Troyanskaya, K. Schluens, S. Petersen, Z. Thaesler, M. Pacyna-Gengelbach, M. van de Rijn, G. D. Rosen, C. M. Perou, R. I. Whyte, R. B. Altman, P. O. Brown, D. Botstein and I. Petersen (2001). Diversity of gene expression in adenocarcinoma of the lung. *Proc. Natl. Acad. Sci. USA*, 98(24):13784–13789.

Garey M. R. and D. S. Johnson (1979). *Computers and intractability, a guide to the theory of np-completeness*. New York, NY: Freeman.

Gersho, A. and R. M. Gray (1992). *Vector quantization and signal compression*. Kluwer Academic Publishers, Boston.

Graepel T., R. Herbrich, and J. Shawe-Taylor (2000). Generalisation error bounds for sparse linear classifiers. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, 298–303.

Graepel T., R. Herbrich, and J. Shawe-Taylor (2005). Pac-bayesian compression bounds on the prediction errors of learning algorithms for classification. *Machine Learning* 59(1-2): 55–76.

Graepel T., R. Herbrich, and R. C. Williamson (2001). From margin to sparsity. In *Advances in Neural Information Processing Systems 13*, 210–216.

Golub T. R., D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield and E. S. Lander (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537.

Gordon D. F. and M. Desjardins (1995). Evaluation and selection of biases in machine learning. *Machine Learning*, 20(1-2):5–22.

Guyon I. and A. Elisseeff, editors (2003). *Journal of Machine Learning Research Special Issue on Variable and Feature Selection*, 3.

Guyon I., J. Weston, S. Barnhill, and V. Vapnik (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422.

Hastie T., R. Tibshirani and J. Friedman (2001). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer-Verlag.

Haussler D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36:177–221.

Haussler D. (1992). Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications. *Information and Computation*, 100:78–150.

Haussler D. (1989). Learning conjunctive concepts in structural domains *Machine Learning* 4:7–40.

Haussler D. and E. Welzl (1987). Epsilon-nets and simplex range queries. *Disc. Comput. Geometry*, 2:127–151.

Helmbold D. P. and M. K. Warmuth (1992). Some weak learning results. In *Proceedings of the 5th Workshop on Computational Learning Theory*, 399–412. ACM Press, New York, NY.

Herbrich R. *Learning Kernel Classifiers.* MIT Press, Cambridge, Massachusetts, 2002.

Herbrich R. and R. C. Williamson (2003). Algorithmic Luckiness. *Journal of Machine Learning Research*, 3(2):175–212.

Hinton G. and M. Revow (1996). Using pairs of data-points to define splits for decision trees. *Advances in Neural Information Processing Systems 8*, 507–513.

Hinton G. and T. J. Sejnowski, editors (1999). *Unsupervised Learning, Foundation of Neural Computation*, MIT Press.

Kearns M. (1990). *The Computational Complexity of Machine Learning.* MIT Press, Cambridge, MA.

Kearns M. and M. Li (1993). Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837.

Kearns M., M. Li, L. Pitt and L. Valiant (1987). On the learnability of boolean formulae. In *19th ACM Symposium on Theory of Computing*, 285–295, New York.

Kearns M. J. and R. E. Schapire (1990). Efficient distribution-free learning of probabilistic concepts. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, 382–391. IEEE Computer Society Press, Los Alamitos, CA.

Kearns M. J. and U.V. Vazirani (1994). *An introduction to computational learning theory.* Cambridge, Massachusetts: MIT Press.

Kibler D. and P. Langley (1988). Machine learning as an experimental science. In *Proceedings of the Third European Working Session on Learning.*

Kohonen T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

Kohonen T. (1989). *Self-organization and associative memory*, 3rd ed. Springer, Berlin.

Kohonen T. (1995). *Self-organization maps.* Springer, Berlin.

Kuzmin D. and M. Warmuth (2005). Unlabeled Compression Schemes for Maximum Classes. In *Proceedings of the eighteenth Annual Conference on Learning Theory (COLT 2005)*, Springer, LNCS 3559, 591–605.

Langford J. (2005). Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 3:273-306.

Langley P. and H. Simon (1995). Applications of machine learning and rule induction. *Communications of the ACM*, 38(11):55–64.

Laviolette F. and M. Marchand (2005). PAC-Bayes Risk Bounds for Sample-Compressed Gibbs Classifiers. In *Proceedings of the Twenty Second International Conference on Machine Learning* (ICML'05), 481–488, ACM Press.

Laviolette F., M. Marchand and M. Shah (2005). Margin-Sparsity Trade-off for the Set Covering Machine. In *Proceedings of the 16th European Conference on Machine Learning* (ECML '05), Springer LNAI vol. 3720, 206–217.

Laviolette F., M. Marchand and M. Shah (2006). A PAC-Bayes Approach to the Set Covering Machine. In *Advances in Neural Information Processing Systems 18*, 731–738, MIT-Press, Cambridge, MA, USA.

Lee K. (1989). *Automatic speech recognition: The development of the Sphinx system*. Kluwer Academic Publishers, Boston.

Li M., J. Trom and P. Vitányi (2003). Sharpening Occam's Razor. *Information Processing Letters*, 85(5):267–274.

Littlestone N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318.

Littlestone N. (1989). *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, University of California Santa Cruz.

Littlestone N. and M. Warmuth (1986). Relating data compression and learnability. Technical report, University of California Santa Cruz, Santa Cruz, CA, 1986.

Marchand M. (2004). *Lecture notes of course IFT-65764, Apprentissage Automatique*. Université Laval, Canada.

Marchand M., M. Shah, J. Shawe-Taylor and M. Sokolova (2003). The Set Covering Machine with Data-dependent Half-Spaces. *Proceedings of the Twentieth International Conference on Machine Learning (ICML'2003)*, 520–527, Morgan Kaufmann, San Fransisco, CA.

Marchand M. and M. Shah (2005). PAC-Bayes Learning of Conjunctions and Classification of Gene-Expression Data. In *Advances in Neural Information Processing Systems 17*, 881–888, MIT-Press, Cambridge, MA, USA.

Marchand M. and J. Shawe-Taylor (2001). Learning with the set covering machine. *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, 345–352.

Marchand M. and J. Shawe-Taylor (2002). The set covering machine. *Journal of Machine Learning Research*, 3:723–746.

Marchand M. and M. Sokolova (2005). Learning with decision lists of data-dependent Features. *Journal of Machine Learning Research*, 6:427–451.

McAllester D. (1999). Some PAC-Bayesian theorems. *Machine Learning*, 37:355–363.

McAllester D. (2003). PAC-Bayesian stochastic model selection. *Machine Learning*, 51:5–21.

Meir R. and G. Rätsch (2003). An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning: Machine Learning Summer School 2002, Canberra, Australia, February 11-22, 2002. Revised Lectures*, LNCS 2600, 118–183, Springer.

Mendelson S. (2002). Rademacher averages and phase transitions in Glivenko-Cantelli class. *IEEE Transactions on Information Theory*, 48:251–263.

Mendelson S. (2003). A few notes on Statistical Learning Theory. In S. Mendelson and A. J. Smola, editors. *Advanced Lectures on Machine Learning: Machine Learning Summer School 2002, Canberra, Australia, February 11-22, 2002. Revised Lectures*, LNCS 2600, 1–40, Springer.

Mendelson S. (2004). Geometric Parameters in learning theory. In *GAFA lecture notes*, LNM 1850, 193–236.

Mitchell T. (1980). The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers University, New Brunswick, NJ.

Mitchell T. (1997). *Machine Learning*. WCB/McGraw-Hill.

Natarajan B. K. (1993). Occam's Razor for functions. In *Proceedings of the 6th Workshop on Computational Learning Theory*, 370–376. ACM Press, New York, NY.

Pajor A. (1985) *Sous-espaces l_1^n des espaces de Banach. (French) [l_1^n-subspaces of Banach spaces]*. Hermann, Paris.

Pearl J. (1978). On the connection between the complexity and credibility of inferred models. *Journal of General Systems*, 4:255–264.

Pitt L. (1989). Inductive Inference, DFAs, and Computational Complexity. *Proc. AII-89 Workshop on Analogical and Inductive Inference; Lecture Notes in Artificial Intelligence 397*, 18–44, Springer-Verlag.

Pitt L. and L. Valiant (1988). Computational limitations on learning from examples. *Journal of the ACM*, 35:965–984.

Pitt L. and M. Warmuth (1990). Prediction preserving reducibility. *J. Comp. Sys. Sci.*, 41(3):430–467. Special Issue of the *Third Annual Conference of Structure in Complexity Theory* (Washington, DC., June 1988)

Pollard D. (1984). *Convergence of Stochastic Processes*. Springer.

Pomerlaeu D. A. (1989). *ALVINN: An autonomous land vehicle in a neural network*. Technical Report CMU-CS-89-107. Carnegie Mellon University, Pittsburgh, PA.

Pomeroy S. L., P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. Kim, L. C. Goumnerova, P. M. Black, C. Lau, J. C. Allen, D. Zagzag, J. M. Olson, T. Curran, C. Wetmore, J. A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. N. Louis, J. P. Mesirov, E. S. Lander, and T. R. Golub (2002). Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436–442.

Provost F. J. and T. Fawcett (1998). Robust classification systems for imprecise environments. In *Proceedings of the 16th National Conference on Artificial Intelligence*, 706–713.

Rivest R. L. (1987). Learning decision lists *Machine Learning*, 2:229–246.

Rosenblatt F. (1957). *The perceptron: A perceiving and recognizing automaton (project PARA)*. Technical Report 85-460-1, Cornell Aeronautical Laboratory.

Rosenblatt F. (1959). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Rumelhart D., B. Widrow and M. Lehr (1994). The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–92.

Sarrett W. and M. Pazzani (1992). Average case analysis of empirical and explanation-based learning algorithms. *Machine Learning*, 9(4):349–372.

Saunders C., O. Stitson, J. Weston, L. Bottou, B. Schölkopf and A. Smola (1998). *Support vector machine reference manual* (Technical Report CSD-TR-98-03). Department of Computer Science, Royal Holloway, University of London, London, UK.

Schapire R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Schapire R. E. (1992). *The Design and Analysis of Efficient Learning Algorithms*. MIT Press, Cambridge, MA.

Schapire R. E. (1999). A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 1401–1406.

Seeger M. (2002). PAC-Bayesian generalization bounds for gaussian processes. *Journal of Machine Learning Research*, 3:233–269.

Shah M., M. Sokolova and S. Szpakowicz (2004). The Role of Domain Specific Knowledge in Classifying the Language of E-negotiations. In *Proceedings of the 3rd International Conference of Natural Language Processing* (ICON 2004), 99–108, Allied, India.

Shawe-Taylor J., M. Anthony and N. Biggs (1989). *Bounding sample size with the Vapnik-Chervonenkis dimension.* Technical Report CSD-TR-618, University of London, Royal Holloway and New Bedford college.

Shawe-Taylor J. and N. Cristianini (2004). *Kernel Methods for Pattern Analysis.* Cambridge University Press, Cambridge, U.K..

Simon, Herbert A (1983). Why should machines learn. In Michalski, S. Ryszard; Carbonell, G. Jaime; and Mitchell, Tom M., editors, *Machine Learning, An Artificial Intelligence Approach (volume I).* Morgan Kaufmann.

Schölkopf B. and A. Smola (2002). *Learning with Kernels.* The MIT Press.

Snell N. D. (1997). *Machine Learning Why and How Explaind.* Lecture Notes,University of Sunderland.

Sokolova M., V. Nastase, S. Szpakowicz and M. Shah (2005). Analysis and Models of Language in Electronic Negotiations. in M. Draminski, P. Grzegorzewski, K. Trojanowski, S. Zadrozny, editors, *Issues in Intelligent Systems: Models and Techniques.* Akademicka Oficyna Wydawnicza EXIT, Warszawa, 197–211.

Sontag E.D. (1998). VC dimension of neural networks. In C. M. Bishop (Ed.), *Neural Networks and Machine Learning*, 69–94, Springer, Berlin.

Talagrand M. (1987). The Gilvenko-Cantelli Problem. *The Annals of Probability*, 15:837–870.

Valiant L. G. (1984). A theory of the learnable. *Communications of the Association of Computing Machinery*, 27:1134–1142.

Valiant L. G. (1985). Learning disjunctions of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, 560–566, Los Angeles.

Vapnik V. N. (1982). *Estimation of Dependences Based on Empirical data*. Springer-Verlag, New York.

Vapnik V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

Vapnik V. N. (1998). *Statistical Learning Theory*. Wiley, New York, NY.

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):264–280.

Venkatesh S. (1991). On learning binary weights for majority functions. In *Computational Learning Theory: Proceedings of the Fourth Annual Workshop*, 257-299.

von Luxburg U., O. Bousquet, and B. Schölkopf (2004). A compression approach to support vector model selection. *Journal of Machine Learning Research*, 5:293–323.

Waibel A., T. Hanazawa, G. Hinton, K. Shikano and K. Lang (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339.

Weiss S. M. and I. Kapouleas (1989). An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, 781–787.

West M., C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. A. Olson Jr, J. R. Marks and J. R. Nevins (2001). Predicting the clinical status of human breast cancer by using gene expression profiles. *Proc. Natl. Acad. Sci. USA*, 98(20):11462–11467.

Witten I. and E. Frank (2000). *Data Mining*. Morgan-Kaufmann.

Zhu X. (2005). Semi-Supervised Learning Literature Survey. Technical Report, Computer Sciences TR 1530, University of Wisconsin-Madison.

Typeset in Computer Modern by TEXand LATEX 2ε