DEPARTMENT OF COMPUTER SCIENCE
SERIES OF PUBLICATIONS A
REPORT A-2004-7

Learning Small Trees and Graphs that Generalize

Matti Kääriäinen

UNIVERSITY OF HELSINKI
FINLAND

DEPARTMENT OF COMPUTER SCIENCE
SERIES OF PUBLICATIONS A
REPORT A-2004-7

Learning Small Trees and Graphs that Generalize

Matti Kääriäinen

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Room 10, University Main Building, on October 22, 2004, at noon.

UNIVERSITY OF HELSINKI
FINLAND

Contact information

Postal address:
Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki
Finland

Email address: postmaster@cs.Helsinki.FI

URL: http://www.cs.Helsinki.FI/

Telephone: +358 9 1911

Telefax: +358 9 1915 1120

ISSN 1238-8645
ISBN 952-10-2050-4 (paperback)
ISBN 952-10-2051-2 (PDF)
Computing Reviews (1998) Classification: I.2.6, F.2.2, G.3
Helsinki 2004
Helsinki University Printing House

Learning Small Trees and Graphs that Generalize

Matti Kääriäinen

Department of Computer Science

P.O. Box 68, FIN-00014 University of Helsinki, Finland

Matti.Kaariainen@cs.Helsinki.FI, http://www.cs.Helsinki.FI/u/mtkaaria/

PhD Thesis, Series of Publications A, Report A-2004-7

Helsinki, September 2004, 45+49 pages

ISSN 1238-8645

ISBN 952-10-2050-4 (paperback)

ISBN 952-10-2051-2 (PDF)

Abstract

In this Thesis we study issues related to learning small tree and graph formed classifiers. First, we study reduced error pruning of decision trees and branching programs. We analyze the behavior of a reduced error pruning algorithm for decision trees under various probabilistic assumptions on the pruning data. As a result we get, e.g., new upper bounds for the probability of replacing a tree that fits random noise by a leaf. In the case of branching programs we show that the existence of an efficient approximation algorithm for reduced error pruning would imply P=NP. This indicates that reduced error pruning of branching programs is most likely impossible in practice, even though the corresponding problem for decision trees is easily solvable in linear time.

The latter part of the Thesis is concerned with generalization error analysis, more particularly on Rademacher penalization applied to small or otherwise restricted decision trees. We develop a progressive sampling method based on Rademacher penalization that yields reasonable data dependent sample complexity estimates for learning two-level decision trees. Next, we propose a new scheme for deriving generalization error bounds for prunings of induced decision trees. The method for computing these bounds efficiently relies on the reduced error pruning algorithm studied in the first part of this Thesis. Our empirical experiments indicate that the obtained training set bounds may be almost tight enough to be useful in practice.

Computing Reviews (1998) Categories and Subject Descriptors:
I.2.6 Learning—decision trees, branching programs, pruning, progressive sampling, generalization error analysis
F.2.2 Analysis of Algorithms and Problem Complexity: Nonnumerical Algorithms and Problems—branching programs, pruning
G.3 Probability and Statistics: Nonparametric statistics

General Terms:
Theory, Experimentation

Additional Key Words and Phrases:
Learning, learning theory, decision trees, decision tree pruning, branching program pruning, progressive sampling, generalization error analysis, Rademacher penalization

Acknowledgments

I am most grateful to my advisor, Professor Tapio Elomaa, for his advice and continuous support during my studies. He introduced me to machine learning and computer science research in general while I was still an undergraduate student, and his supervision and guidance has been invaluable in my studies and research ever since. Without him pushing me forward, I would probably never have finished this Thesis project.

The Department of Computer Science has provided me with great working conditions in Vallila and now in the new murky building in Kumpula. Special thanks go to the marvellous Computing Facilities staff. The computing environtment at the department has been excellent — I have had to spend almost no time in fighting with computer problems (caused by factors other than me).

I have received financial support for my PhD studies from multiple sources. The Department of Computer Science, where I have worked as a part time teacher and as a summer trainee, was my primary source of income in the beginning of my PhD studies. Working as a teaching assistant has taught me a lot and I still like to teach part time. Helsinki Graduate School in Computer Science and Engineering (HeCSE) has provided me financial support from mid 2002. I have got additional funding from the Academy of Finland and from the From Data to Knowledge (FDK) research unit. I wish to thank Professor Esko Ukkonen for this support and his guidance.

Of my collegues I wish to thank my co-students and friends Taneli Mielikäinen, Ari Rantanen, Janne Ravantti, Teemu Kivioja, Veli Mäkinen, Jussi Lindgren, and late Tuomo Malinen. You have always been ready for heated discussions about the state of affairs at our department and elsewhere. Of the more senior colleagues I wish to thank Juho Rousu, Matti Luukkainen, Jyrki Kivinen, Patrik Floréen, Floris Geerts and Bart Goethals. People in real life have also been of great help. Of them I wish to thank my parents Helena and Ilpo, my brother and friend Anssi, Jasmina the dog, all my friends, and especially my girlfriend Jessica.

Helsinki, September 2004
Matti Kääriäinen

Contents

Original Publications

This Thesis is based on the following papers, which are referred to in the text as Paper 1, Paper 2, Paper 3, and Paper 4.

1. Tapio Elomaa and Matti Kääriäinen:

   An Analysis of Reduced Error Pruning

   Journal of Artificial Intelligence Research 15 (2001), pages 163–187.

2. Richard Nock, Tapio Elomaa, and Matti Kääriäinen:

   Reduced Error Pruning of branching programs cannot be approximated to within a logarithmic factor

   Information Processing Letters 87:2 (2003), pages 73–78.

3. Tapio Elomaa and Matti Kääriäinen:

   Progressive Rademacher Sampling

   In *Proc. 18th National Conference on Artificial Intelligence, AAAI-2002* (Edmonton, Canada), pages 140–145.

4. Matti Kääriäinen and Tapio Elomaa:

   Rademacher Penalization over Decision Tree Prunings

   In LNAI 2837: *Machine Learning: ECML 2003, Proc. 14th European Conference, ECML'03* (Cavtat-Dubrovnik, Croatia), pages 193–204.

Chapter 1

Introduction

We begin with an informal motivation for the subject of the Thesis, after which the contributions of the author are briefly summarized in Section 1.2.

1.1 Motivation

We are interested in *learning small trees and graphs that generalize*. The main emphasis will be on the generalization ability of the learned classifiers — the interpretable graph structure and small size will either facilitate generalization or come as a free bonus. In this section we briefly motivate our interests, starting with smallness. The discussion will be very informal and pragmatic, thus avoiding delving on the philosophical debate around Occam's principle [9, 43]. Only minimal background on machine learning will be assumed. Concepts used here without being properly introduced will be defined in detail later. For a more thorough introduction to machine learning and related issues, see the next chapter and, e.g., [50, 58].

Small size is considered to enhance understandability. It is probably easier to understand the inner logic of a classifier — that is, a function that classifies objects based on their attributes — if its description fits on a single page than if its shortest description fills an entire library. Smallness thus has some connection to simplicity, at least on an intuitive level. As we would like the learned classifiers to not only give correct classifications but also represent some knowledge in a human understandable way, smallness is a good property to look for.

Another fact favoring smallness is that small size is beneficial from a computational point of view. With any sensible definition of smallness, small classifiers require little storage space (e.g. memory in a computer). In case of classifiers that have a tree or graph structure, small size also implies that classification of instances is fast. Thus, a small classifier is efficient with respect to both space and

time complexity, the most important complexity measures studied in computer science.

A deeper reason for being interested in small classifiers is that the set of all small classifiers (classifiers with short descriptions with respect to a fixed description method) is itself small and thus not overly complex. For example, the number of things one can say on a single page of text is relatively small, at least if compared to the multitude of different things one could communicate using an entire library full of books. In the learning framework studied in this Thesis the smallness of sets of small classifiers enables one to prove generalization error bounds. That is, one can (under certain assumptions) prove that a classifier performing well on learning data will with high probability perform well on unseen data, too, given that the classifier is selected from a small set of classifiers. Here, what actually matters is not the small size of individual classifiers but that of classes of classifiers. However, since the former implies the latter we can conclude that (again under suitable assumptions) small classifier size guarantees some generalization capability. Learning classifiers that generalize well is commonly considered one of the ultimate goals in machine learning, so even if we were not particularly interested in generalization — which we are — the connection between small size and good generalization would strongly support learning small classifiers.

Human experts commonly think that tree formed classifiers are easy to understand [14], although no experimental study supporting this seems to exist [15]. The reason graph formed classifiers are considered easy to understand is probably their visualizable discrete structure that determines the logic by which the classifiers classify examples. Any deeper analysis of understandability would, of course, require some understanding of what understanding means and so on. In this Thesis, however, understandability is used only as a motivation for our objectives and will not be a subject for any further study.

From a computer science viewpoint a more appealing property of tree and graph formed classifiers is the fact that objects like trees and graphs are well-studied discrete structures that can be efficiently represented in and handled by computers [17]. The computational problems arising in learning such classifiers have thus a combinatorial flavor and can be attacked using tools most familiar to a computer scientist. Finally, decision trees have not only been studied theoretically but are also widely used in data analysis and have been observed to perform well on a wide range of problems with practical importance [14, 57, 42, 25]. Advances in the theory of decision tree learning may thus have strong significance in applications.

Small tree and graph formed classifiers and their generalization performance is not only the cement that glues the four papers constituting this Thesis together. These topics are also central to each of the individual papers. In Papers 1 and

2 the focus is more on smallness. We analyze reduced error pruning of decision trees and graphs, respectively. Reduced error pruning is an elementary pruning algorithm, i.e., an algorithm that tries to find and delete the parts of a graph formed classifier that do not enhance its classification performance on unseen data. Thus, pruning aims at reducing the size of a classifier while maintaining or improving its accuracy — both goals well in line with our agenda.

The remaining two papers concentrate on generalization error analysis of decision trees. In Paper 3 we apply a recently introduced technique called Rademacher penalization to progressive sampling. More specifically, we use Rademacher penalties for estimating the amount of data that is needed to learn two-level decision trees that meet given generalization performance guarantees. In Paper 4 we apply the reduced error pruning algorithm for decision trees analyzed in Paper 1 to computing Rademacher penalties of the class of prunings of an induced decision tree. This technique enables us to prove tight data-dependent generalization error bounds for decision trees learned by standard two-phase decision tree learning algorithms like C4.5 [57]. Even though small size is not the main concern in Papers 3 and 4 it is an important ingredient in providing good generalization error bounds.

1.2 Main Contributions

For easy access, the main research contributions presented in this Thesis are listed below. The numbering of the list corresponds to the numbering of the papers that constitute the bulk of this Thesis.

1. The analysis of reduced error pruning (Paper 1) yields improved results on the behavior of reduced error pruning of decision trees with less imposed assumptions than those in previous studies.

2. Our hardness results on reduced error pruning of branching programs (Paper 2) show that branching program pruning — at least in the reduced error pruning sense — is probably a lot harder than pruning decision trees.

3. Applying Rademacher penalization in the context of progressive sampling (Paper 3) is to the author's knowledge the first application of data dependent generalization error bounds to sample complexity estimation. The empirical results suggest that the approach improves significantly on previous theoretically motivated sample complexity estimation methods that do not take the data distribution into account.

4. Rademacher penalization over decision tree prunings (Paper 4) is a conceptually new way of providing generalization error bounds for decision

trees. To the authors best knowledge, the obtained bounds are the tightest published training set bounds for decision tree prunings.

The author of this Thesis made the main contribution to papers 1, 3, and 4. Paper 2 that improves on our earlier results on branching program pruning [23], is to a large extent due to Professor Richard Nock. Even in this paper the author's contribution is substantial.

The rest of this Thesis is devoted to describing the above contributions in more detail. The next chapter presents some preliminaries necessary for understanding the chapters that follow. The main results of Papers 1–4 will be presented in Chapters 3–6, respectively, while the conclusions of this study are summarized in Chapter 7. Papers 1–4 in their original published form are included in the end of the Thesis.

Chapter 2

Preliminaries

The first section of this chapter presents the learning model of statistical learning theory that underlies the rest of this Thesis. After that, a short introduction to established generalization error analysis methods is given in Section 2.2. A special emphasis is given to Rademacher penalization. Background information on topics like decision tree learning and progressive sampling, is given in later chapters as needed.

2.1 Learning Model

We are interested in learning classifiers from examples, which is a special case of supervised learning. As our learning framework we use *statistical learning theory*. A good introduction to classical results in this field is given by Vapnik [65]. Here, we will review only the very basics. Related and sometimes quite orthogonal approaches to learning classifiers from examples include, e.g., PAC-learning [62] and its agnostic variant [35], Bayesian approaches [31], different versions of query learning [1], and a whole variety of on-line learning models [67, 36, 19, 68].

We consider the following learning situation. The *learner* (e.g. a machine learning algorithm) is presented with an ordered set of labeled examples $(x_1, y_1), \ldots, (x_n, y_n)$ called the *learning sample*. Here, the *attribute vectors* $x_i \in \mathcal{X}$ represent the attributes of the examples and the $y_i \in \mathcal{Y}$ are the corresponding *labels*. We will be interested in classification only, so $\mathcal{Y}$ is assumed to be finite. For a concrete example, suppose the learner tries to learn to classify digitalized 16×16 gray-scale images of hand-written digits from 0 to 9. In this case, the attribute space $\mathcal{X}$ might be $\{0, \ldots, 255\}^{256}$ (assuming 8 bits are used to encode the shade of gray of a pixel) and the label space $\mathcal{Y}$ would be $\{0, \ldots, 9\}$. Thus, an example (x, y) would consist of a gray-scale image x labeled with a digit $y \in \{0, \ldots, 9\}$.

The learner outputs a *classifier* (also referred to as a *hypothesis*) $f\colon \mathcal{X} \to \mathcal{Y}$ based on the learning sample $(x_1, y_1), \ldots, (x_n, y_n)$. In the hand-written digit classification problem, a classifier would simply be a function associating to each gray-scale image $x \in \mathcal{X}$ some digit $y \in \mathcal{Y}$. Usually, the learner does not consider all possible functions from $\mathcal{X}$ to $\mathcal{Y}$, but restricts itself to a *hypothesis class* F that is a subset of all functions $f\colon \mathcal{X} \to \mathcal{Y}$. The restriction to such a subset F has an important role in generalization error analysis and will be discussed in the next section. Intuitively, F can be seen to represent the prior assumptions the learner has about the learning task. That is, the learner assumes that the learning task is such that the class F contains some hypotheses f that perform well on the task. In this Thesis, the hypothesis class F will usually consist of a subset of the classifiers that can be represented by decision trees or branching programs.

So far, we have in no way restricted the process generating the learning sample or the way the learner chooses its classifier. In order to make the learning model non-vacuous, we have to at least specify some quality criteria that the classifier output by the learner should meet. For example in the handwritten digit recognition problem the classifier output by the learner should be such that it gives correct labels to all reasonably clearly written digits. This hints that in order to specify a quality criterion for the classifiers we first have to assume something about the learning sample generating process — without a definition of what a reasonably clearly written digit means, there is no way to make the intuitive quality criterion above precise. Ideally, we would like to assume as little as possible of the learning sample generating process, as the properties of this process are exactly what we want to model with the learned classifier. However, nothing can be done without prior assumptions, a fact exemplified by the various *no free lunch theorems* [69].

A natural way of measuring the performance of a classifier is to see how accurately it predicts the labels of previously unseen examples, that is, how well it generalizes. For example, in the digit recognition problem we want the learned classifier to classify correctly also hand-written digits that it has not encountered before. If we wish the learner to be able to learn a classifier that performs well on unseen examples, we have to guarantee that the learning sample and the future examples are somehow related. In *statistical learning theory* [65] one assumes that the learning examples (x_i, y_i) are chosen independently at random from a fixed but unknown distribution P over $\mathcal{X} \times \mathcal{Y}$. The learning sample is thus just a random element of $(\mathcal{X} \times \mathcal{Y})^n$ selected according to the n-fold product distribution P^n. The goal of the learner is to find a classifier $f \in F$ with small *generalization error*

$$\epsilon(f) = P(f(X) \neq Y),$$

where the random vector (X, Y) is distributed according to P. In other words, the learner is supposed to find a classifier whose probability of misclassification

on examples chosen from the same distribution as the learning examples is low. Other characteristics of the classifier that the learner could try to optimize, e.g., the size of the classifier, are ignored in this theoretical model.

Of course, the problem here is that P is not known to the learner. Otherwise, the learner could simply choose the provably optimal *Bayes-classifier* [20]

$$f_{\text{bayes}}(x) = \arg\max_{y \in \mathcal{Y}} P(y|x)$$

or the best approximation thereof as its classifier. In the learning model of statistical learning theory, the only knowledge of P available to the learner is the randomly chosen learning sample $(x_1, y_1), \ldots, (x_n, y_n)$. It is this knowledge the learner should use in finding a classifier with good generalization performance. One theoretically motivated way to find classifiers with guaranteed generalization performance is outlined in the next section.

2.2 Generalization Error Analysis

2.2.1 Main Ideas

Given that we have the sample $(x_1, y_1), \ldots, (x_n, y_n)$ at our disposal, it is natural to try to approximate the generalization error of a classifier — its true probability of misclassification — by the observable empirical rate of misclassifications on the learning sample. To this end, let us define the *empirical error* $\hat{\epsilon}_n(f)$ of a classifier f as

$$\hat{\epsilon}_n(f) = \frac{1}{n} \sum_{i=1}^{n} [\![f(x_i) \neq y_i]\!].$$

Here, the notation $[\![\cdot]\!]$ means the function taking the value 1 if the expression inside the double brackets is true and 0 otherwise. When the sample size is clear from context we often drop the subscript n from $\hat{\epsilon}_n(f)$.

Suppose that the empirical errors of all the classifiers in F can with high probability be guaranteed to be close to the corresponding generalization errors. That is, suppose

$$\sup_{f \in F} (\epsilon(f) - \hat{\epsilon}(f)) \tag{2.1}$$

is small with high probability. Then the learner can solve the learning task by picking a classifier with small empirical error, because when (2.1) is small, any hypothesis with small empirical error will have a small generalization error, too:

$$\epsilon(f) = \hat{\epsilon}(f) + \epsilon(f) - \hat{\epsilon}(f) \leq \hat{\epsilon}(f) + \sup_{f \in F} (\epsilon(f) - \hat{\epsilon}(f)).$$

This principle of selecting the classifier with minimal empirical error has been introduced by Vapnik [64]. It is called the *empirical risk minimization* (ERM) principle and will be of central importance throughout the rest of this Thesis.

We have shown that the intuitively appealing ERM principle solves the learning problem if we succeed in proving good upper bounds for (2.1). Deriving such upper bounds is a special case of *generalization error analysis*, which can be defined in mathematical terms as follows. Given a confidence parameter $\delta > 0$, find some penalty term A such that with probability at least $1 - \delta$ we have

$$\epsilon(\hat{f}) \leq \hat{\epsilon}(\hat{f}) + A, \tag{2.2}$$

where $\hat{f} \in F$ is the classifier chosen by the learning algorithm based on the learning sample $(x_1, y_1), \ldots, (x_n, y_n)$. The complexity penalty term A may depend on anything known to the learning algorithm, for example on the algorithm itself, the properties of the classifier $\hat{f}$, the hypothesis class F, the learning sample $(x_1, y_1), \ldots, (x_n, y_n)$ and of course the confidence parameter δ. Obviously, the goal is to make A as small as possible, that is, to prove tight generalization error bounds. A dual problem for generalization error analysis is *sample complexity analysis*: Given δ and some upper bound ε for A, find a lower bound for the sample size that ensures inequality (2.2) holds. We will return to a variant of the sample complexity analysis problem in Chapter 4 when discussing the results of Paper 3 on progressive sampling.

Bounding the quantity (2.1) leads to the the special case of inequality (2.2) in which A is not allowed to depend on $\hat{f}$ nor the algorithm for choosing it. Thus, A has to be a uniform bound for the difference between the generalization error and the empirical error of a hypothesis over the whole class F. As A does not depend on $\hat{f}$, the ERM hypothesis is the one that minimizes the upper bound for the generalization error. This is the principal motivation behind the ERM principle. Of course, bounding

$$\sup \left\{ \epsilon(f) - \hat{\epsilon}(f) \mid f \in F \text{ and } \epsilon(f) = \min_{g \in F} \epsilon(g) \right\}$$

directly might (and sometimes does [5]) lead to tighter bounds for ERM. Such bounds, however, have turned out to be very hard to obtain in practice so we will mostly confine ourselves to uniform bounds of the form (2.1). Bounds in which A may depend on $\hat{f}$ in some way lead to different learning principles. Thus, deriving new ways to analyze the generalization error gives as an important side product new criteria for selecting classifiers. Even generalization error bounds that are too loose to be applicable in practice may thus be useful as they may provide new insight for designing learning algorithms [12].

Machine learning literature is packed with different approaches to proving generalization error bounds. Following Langford [40], these can be roughly divided into two categories: *test set bounds* and *training set bounds*. In test set bounds, the learning algorithm is allowed to use only part of the learning sample in learning the classifier $\hat{f}$ while the rest of the sample is used in providing an unbiased test error estimate for $\epsilon(\hat{f})$. On the other hand, in training set bounds the learner may use all the data for learning purposes, which means that the performance of the learned classifier has to be evaluated on the sample that was used in choosing it. In training set bounds we thus have more data to learn from, but as there is no separate test set, the generalization error analysis is a more complicated task and the resulting bounds are therefore often not particularly tight.

To give a general picture of existing generalization error analysis techniques and to relate our work to them we will next present some examples of both test set and training set bounds. First, we will derive the basic test error bound (2.3), after which training set bounds for finite hypothesis classes and classes with finite VC dimension [10, 64] are given. Test set bounds not discussed here include, e.g., cross validation bounds and leave one out estimates [20], while some of the most important uncovered training set bounds are those based on covering numbers [2], marginals of linear classifiers [18], sparseness [26], Occam's theorem [9], PAC-Bayesian theorems [46], PAC-MDL theorems [8], the luckiness framework [60, 30] and stability [13]. In Section 2.2.3 we will finally present the basics of Rademacher penalization [37], a relatively new data-dependent generalization error analysis technique that is central to the work presented in Papers 3 and 4. The currently less practical local variations of Rademacher penalization presented in the literature [39, 4] will not be discussed further in this Thesis.

2.2.2 Examples of Generalization Error Bounds

The idea behind test error bounds is the following. First divide the learning sample randomly into two parts of size $n - m$ and m, say S_1 and S_2. Then, give S_1 to the learner that selects a classifier $\hat{f}$ based on it. The generalization error of $\hat{f}$ can now be estimated by its *test set error*

$$\hat{\epsilon}_{\text{test}}(\hat{f}) = \frac{1}{m} \sum_{(x,y)\in S_2} [\![\hat{f}(x) \neq y]\!].$$

It is clear that $m\hat{\epsilon}_{\text{test}}(\hat{f})$ has binomial distribution with parameters m and $\epsilon(\hat{f})$ since it is a sum of independent Bernoulli$(\epsilon(\hat{f}))$ distributed random variables. Hence, a moment of thought (or a look at [41]) reveals that with probability at least $1 - \delta$ we have

$$\epsilon(\hat{f}) \leq \overline{\text{Bin}}\left(\hat{\epsilon}_{\text{test}}(\hat{f}), m, \delta\right),$$

where $\overline{\mathrm{Bin}}(k/m, m, \delta)$ is the *inverse binomial tail* [41] defined by

$$\overline{\mathrm{Bin}}\left(\frac{k}{m}, m, \delta\right) = \max_{p\in[0,1]}\left\{p : \sum_{i=0}^{k}\binom{m}{i}p^i(1-p)^{m-i} \geq \delta\right\}.$$

If a closed-form upper bound for $\overline{\mathrm{Bin}}(k/m, m, \delta)$ is desired, we can use the exponential moment method of Chernoff [28] to get, e.g., the well-known approximation

$$\overline{\mathrm{Bin}}\left(\frac{k}{m}, m, \delta\right) \leq \frac{k}{m} + \sqrt{\frac{\ln(\frac{2}{\delta})}{2m}}.$$

However, as computing numerical estimates for the inverse binomial tail is easy, the sometimes loose Chernoff type approximations should be used with care.

Putting the above derivations together, we get the following theorem.

Theorem 2.2.1 *Suppose $\hat{f}$ does not depend on the test sample S_2. Then, with probability at least $1-\delta$ over the choice of S_2, we have*

$$\epsilon(\hat{f}) \leq \overline{\mathrm{Bin}}(\hat{\epsilon}_{\mathit{test}}(\hat{f}), m, \delta) \leq \hat{\epsilon}_{\mathit{test}}(\hat{f}) + \sqrt{\frac{\ln(\frac{2}{\delta})}{2m}}. \tag{2.3}$$

The first inequality of Theorem 2.2.1 can be put (a bit artificially) into the form of (2.2) by picking $A = -\hat{\epsilon}(\hat{f}) + \overline{\mathrm{Bin}}(\hat{\epsilon}_{\mathrm{test}}(\hat{f}), m, \delta)$, which coincidentally shows that minimization of empirical error does not necessarily have anything to do with minimizing a test error bound, a fact supported by empirical experiments with, e.g., decision tree learning [14]. Indeed, the ERM classifier is often not the classifier with best generalization performance. In such cases, the ERM classifier is said to *overfit* the training data.

It is evident from inequality (2.3) that the test error bound for a fixed hypothesis $\hat{f} \in F$ is the tighter the larger m is — that is, the more data we have for testing purposes. However, if we have only a fixed number n of learning examples at our hands, then increasing the test set size m results in a decrease in $n-m$, the amount of data that remains for actual learning purposes. Hence, the hypothesis $\hat{f}$ has to be chosen on the basis of a smaller sample which in turn may increase the test error term in (2.3). One of the reasons for developing training set bounds is to circumvent this trade-off by allowing the use of all examples for both learning and bounding the error of the learned hypothesis.

In the proof of Theorem 2.2.1 it is essential that the classifier $\hat{f}$ whose generalization error we bound and the test sample on which the classifier is evaluated are independent. However, when we try to prove training set bounds that are based on the empirical error of the classifier, the sample used for learning and testing

is the same. This complicates things a lot, as the scaled empirical error $n\hat{\epsilon}(\hat{f})$ of the learned classifier is typically not binomially distributed even though the scaled empirical errors $n\hat{\epsilon}(f)$ for fixed $f \in F$ are. Hence, to get training set bounds we need more refined techniques than the simple ones that suffice in the case of a separate test set.

The simplest way around the above problem is to analyze the deviations of each classifier $f \in F$ separately as in the test error case and then combine these bounds using the union bound for probabilities. More specifically, as $n\hat{\epsilon}(f) \sim \text{Bin}(n, \epsilon(f))$ for every fixed $f \in F$, the inequality

$$\epsilon(f) \leq \overline{\text{Bin}}\left(\hat{\epsilon}(f), n, \delta'\right) \tag{2.4}$$

holds for any fixed $f \in F$ with probability at least $1 - \delta'$. If F is finite and we have no prior beliefs about the goodness of the classifiers $f \in F$, we can take $\delta' = \delta/|F|$. A simple application of the union bound for probabilities then gives

$$\mathbf{Pr}[\text{some } f \in F \text{ violates (2.4)}] \leq \sum_{f \in F} \mathbf{Pr}[f \text{ violates (2.4)}] \leq \sum_{f \in F} \frac{\delta}{|F|} = \delta,$$

thus establishing a bound of the form (2.1):

Theorem 2.2.2 *In case F is finite, with probability at least $1 - \delta$ it is true that*

$$\epsilon(f) \leq \overline{\text{Bin}}\left(\hat{\epsilon}(f), n, \frac{\delta}{|F|}\right) \leq \hat{\epsilon}(f) + \sqrt{\frac{\ln\left(\frac{2|F|}{\delta}\right)}{2n}} \tag{2.5}$$

for all $f \in F$.

The most important weaknesses of Theorem 2.2.2 are that it only applies to finite F, it does not take the observed learning sample into account in any way (except through the empirical errors of the classifiers) and it contains slackness due to the careless use of the union bound. These weaknesses arise from measuring the complexity of F by its cardinality alone, thus naïvely ignoring the correlations between the classifiers in F as functions on all of $\mathcal{X}$ or on the observed learning sample. Bounds based on VC dimension are a way to get rid of the finiteness assumption, but the VC bounds still suffer from the other two problems. These will be partially solved by the Rademacher penalization bounds discussed in the next subsection.

The bounds based on VC dimension as introduced by Vapnik and Chervonenkis [66] apply only to classes of binary classifiers, so let us assume throughout the rest of this subsection that $|\mathcal{Y}| = 2$, say $\mathcal{Y} = \{0, 1\}$. Under this assumption, the VC dimension of a set of classifiers F can be defined as the cardinality of the

largest set of points in $\mathcal{X}$ that can be classified in all possible ways by functions in F. Formally,

$$\text{VCdim}(F) = \max\{|A| \mid |F|A| = 2^{|A|}\},$$

where $F|A$ means the set of restrictions of functions in F to the set $A \subset \mathcal{X}$.[1] Using Sauer's lemma [59], VC dimension can be used to provide an upper bound for the *shatter coefficient* [20, 65] of a class of classifiers F — the number of different ways in which the classifiers in F can behave on a set of unlabeled examples with a given size. This way VC dimension can be connected to generalization error analysis, giving the following theorem [64].

Theorem 2.2.3 *Suppose $|\mathcal{Y}| = 2$, let F be a class of classifiers and let P be an arbitrary probability distribution on $\mathcal{X} \times \mathcal{Y}$. Suppose F has a finite VC dimension d. Then with probability at least $1 - \delta$ the inequality*

$$\epsilon(f) \leq \hat{\epsilon}(f) + 2\sqrt{\frac{d\left(\ln\left(\frac{2n}{d}\right) + 1\right) + \ln\left(\frac{9}{\delta}\right)}{n}} \tag{2.6}$$

holds for all $f \in F$.

From this theorem we see immediately that if a set of classifiers has finite VC dimension, then the empirical errors of its classifiers converge uniformly to the corresponding generalization errors independently of the choice of P. Thus, finite VC dimension is a sufficient condition for the ERM principle to work in an asymptotic sense — the generalization error of the ERM classifier will converge to $\min\{\epsilon(f) \mid f \in F\}$ as the sample size increases. The implication can also be reversed [64], so a hypothesis class is learnable using the ERM principle if and only if its VC dimension is finite. This and the fact that the convergence rate implied by inequality (2.6) is essentially the best one can prove without making further assumptions about the example generating distribution P [20] makes VC dimension a central concept in learning theory.

The VC dimension bound does not take into account the properties of P that are revealed to the learner by the learning sample. The bound is in this sense distribution independent making the bound worst-case in nature. We will next review a more recent approach called Rademacher penalization that improves on the VC dimension based bounds by using the information in the learning sample to decrease the complexity penalty term for distributions better than the worst.

[1] As a byproduct, we get a practical example of how multiple uses of a symbol (here |) may make things confusing.

2.2.3 Rademacher Penalization

Rademacher penalization was introduced to the machine learning community by Koltchinskii near the beginning of this millennium [39, 37], but the roots of the approach go back to the theory of empirical processes that matured in the 1970s. Here, we will only give the basic definition of Rademacher complexity and a generalization error bound based on it — for proofs and other details, see, e.g., [37], [6] and [63].

Let $r_1, \ldots, r_n$ be a sequence of *Rademacher random variables*, that is, symmetrical random variables that take values in $\{-1, +1\}$ and are independent of each other and the learning examples. The *Rademacher penalty* of a hypothesis class F is defined as

$$R_n(F) = \sup_{f \in F} \left| \frac{1}{n} \sum_{i=1}^{n} r_i [\![f(x_i) \neq y_i]\!] \right| . \tag{2.7}$$

Thus, $R_n(F)$ is a random variable that depends both on the learning sample and the randomness introduced by the Rademacher random variables. A moment of thought shows that the expectation of $R_n(F)$ taken over the Rademacher random variables is large if F contains classifiers that can classify the learning sample with arbitrary labels either accurately or very inaccurately. Otherwise, most of the terms in the sum cancel out each other thus making the value of $R_n(F)$ small. Hence, $R_n(F)$ has at least something to do with the intuitive concept of complexity of F.

It may seem confusing that the value of $R_n(F)$ depends on the Rademacher random variables that are auxiliary to the original learning problem. However, as a consequence of the concentration of measure phenomenon [61] the value of $R_n(F)$ is typically insensitive to the actual outcome of the Rademacher random variables. More specifically, $R_n(F)$ can be shown to be near its expectation (over the choice of the values of the Rademacher random variables or those and the learning sample) with high probability [6]. Thus we can conclude that the random value of $R_n(F)$ is large only if F is complex in the sense that it can realize almost any labeling of the randomly chosen unlabeled learning sample $(x_1, \ldots, x_n)$. As the value of $R_n(F)$ depends on the actual learning sample, $R_n(F)$ is a data dependent complexity measure which makes it potentially more accurate than data independent complexity measures like VC dimension discussed in the previous subsection.

The following theorem provides a generalization error bound in terms of the Rademacher penalty, thus justifying calling $R_n(F)$ a measure of complexity of F. Unlike the VC dimension bound of Theorem 2.2.3, the next theorem applies also in case $|\mathcal{Y}| > 2$.

Theorem 2.2.4 *With probability at least $1 - \delta$ over the choice of the learning sample and the Rademacher random variables, it is true for all $f \in F$ that*

$$\epsilon(f) \leq \hat{\epsilon}(f) + 2R_n(F) + 5\sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{2.8}$$

As the Rademacher penalty does not depend on P directly, the learner has at its hands all the data it needs in evaluating the bound — the values for the Rademacher variables can be generated by flipping a fair coin. Thus, although the complexity penalty term in the bound depends on P through Rademacher complexity's dependence on the learning sample, the bound can still be evaluated without knowing P.

For an extreme example of the difference between Rademacher penalty and VC dimension as complexity measures, suppose F is the class of all functions from $\mathcal{X}$ to $\mathcal{Y}$ and P is a measure whose marginal concentrates on a single point in $\mathcal{X}$. Then $x_1 = \ldots = x_n$ and $R_n(F)$ simplifies to

$$\max\left\{\frac{1}{n}\left|\sum_{i:y_i \neq y} r_i\right| : y \in \mathcal{Y}\right\}.$$

Hence, $R_n(F)$ will be small with high probability over the choice of the Rademacher random variables as long as the learning sample is large compared to the size of $\mathcal{Y}$. The VC dimension of the class of all functions, however, is infinite, so the bound of Theorem 2.2.3 is not applicable. Such extreme distributions P may not be likely to be met in practice, but neither are the worst-case distributions for which the VC dimension based bound is tailored. It is thus plausible that Rademacher penalization may yield some improvements on real world domains, a belief supported by the results of empirical experiments summarized in Papers 3 and 4.

In order to use the bound (2.8) directly, one has to be able to evaluate $R_n(F)$ given the learning sample and a realization of the Rademacher random variables. By the definition of $R_n(F)$, this is an optimization problem, where the objective is essentially given by $\sum_{i=1}^{n} r_i [\![f(x_i) \neq y_i]\!]$ and the domain is the hypothesis class F. As shown by Koltchinskii [37] in the case $|\mathcal{Y}| = 2$, the problem can be solved by the following strategy:

1. Toss a fair coin n times to obtain a realization of the Rademacher random variable sequence $r_1, \ldots, r_n$.

2. Flip the class label y_i if $r_i = +1$ to obtain a new sequence of labels $z_1, \ldots, z_n$, where

$$z_i = \begin{cases} 1 - y_i & \text{if } r_i = +1 \\ y_i & \text{if } r_i = -1. \end{cases}$$

3. Find functions $f_1, f_2 \in F$ that minimize the empirical error with respect to the set of labels z_i and their complement labels $1 - z_i$, respectively.

4. The Rademacher penalty is given by the maximum of $|\{i : r_i = +1\}|/n - \hat{\epsilon}(f_1)$ and $|\{i : r_i = -1\}|/n - \hat{\epsilon}(f_2)$, where the empirical errors $\hat{\epsilon}(f_1)$ and $\hat{\epsilon}(f_2)$ are with respect to z_i and their complements, respectively.

The above strategy can be extended to cope with multiple classes, also, as described in Section 6.1. The hard part, here, is step 3 that requires an ERM algorithm for F. Unfortunately, in the case of many important hypothesis classes, like the class of linear classifiers, no such algorithm is known and the existence of one would violate widely believed complexity assumptions like P $\neq$ NP. Furthermore, there are no other known general methods for evaluating Rademacher penalties than the one outlined above. It is a major open question whether the Rademacher penalties or their expectations over the Rademacher random variables can, in general, be evaluated exactly or even approximately in a computationally efficient manner.

Even though evaluating Rademacher penalties for general F seems to be hard, it is not at all difficult in case an efficient ERM algorithm for F exists. We have experimented with Rademacher penalization using as our hypothesis class the class of two-leveled decision trees and the class of prunings of a given decision tree. For two-leveled decision trees, the ERM algorithm we used is a decision tree induction algorithm by Auer et al. [3]. The case of decision tree prunings is more interesting, as it turns out that reduced error pruning, the algorithm studied in Paper 1, is an ERM algorithm for the class of prunings of a decision tree. We will return in Chapters 4 and 5 to our experiments that show that Rademacher penalization can yield good sample complexity estimates and generalization error bounds in real world learning domains.

Chapter 3

Reduced Error Pruning of Decision Trees

Decision trees are usually learned using a two-phase approach consisting of a growing phase and a pruning phase. Our focus will be on pruning and more specifically on reduced error pruning, the algorithm analyzed in Paper 1. First, we will briefly introduce the basics of decision tree learning in Section 3.1. The reduced error pruning algorithm is outlined in the second section, while our results on it are summarized in the final section of this chapter.

3.1 Growing and Pruning Decision Trees

In the machine learning context decision tree is a data structure used for representing classifiers (or more general regression functions). A decision tree is a finite directed rooted tree, in which the edges go from the root toward the leaves. One usually assumes that the out-degree of all the internal nodes is at least 2 — in case the out-degree of every internal node is exactly 2, we say that the decision tree is binary. At each internal node a there is a *branching function* g_a mapping the example space $\mathcal{X}$ to a's children. The leaves of the tree are labeled with elements of $\mathcal{Y}$.

A decision tree classifies examples $x \in \mathcal{X}$ by routing them through the tree structure. Each example starts its journey from the root of the tree. Given that x has reached an internal node a with branching function g_a, x moves on to $g_a(x)$. The label of the leaf to which x finally arrives is the classification given to x. Viewed in this way a decision tree represents a function $f\colon \mathcal{X} \to \mathcal{Y}$, that is, a classifier.

The class of functions from which the branching functions are chosen is usually very restricted. A typical case is that $\mathcal{X}$ is a product space $\mathcal{X}_1 \times \cdots \times \mathcal{X}_k$, where each of the component spaces $\mathcal{X}_i$, $1 \leq i \leq k$ is either finite or $\mathbb{R}$. The

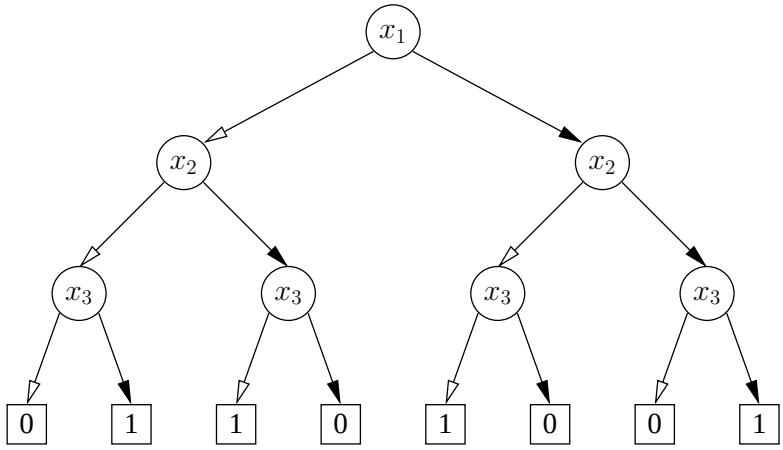

Figure 3.1: A minimal decision tree representation for the exclusive-or function of three bits. Filled arrow heads correspond to set bits.

set of branching functions might be the projections of $\mathcal{X}$ to its finite components and the threshold functions $x = (x_1, \ldots, x_k) \mapsto [\![x_i \leq \theta]\!]$, where $\mathcal{X}_i = \mathbb{R}$ and the threshold $\theta \in \mathbb{R}$ is arbitrary. Even though this class of branching functions is relatively simple, it is easily seen that the decision trees built over it are potentially extremely complex.

Figure 3.1 gives an example of a binary decision tree computing the exclusive-or function of three bits x_1, x_2 and x_3. Here, the examples are represented by binary attribute vectors $(x_1, x_2, x_3) \in \mathcal{X} = \{0, 1\}^3$ and the label space $\mathcal{Y} = \{0, 1\}$. The class of branching functions consists of the projections of $\mathcal{X}$ to its components. It is easy to verify that this is a most concise decision tree representation of the exclusive-or of three bits and that in general representing the exclusive-or of k bits requires a decision tree with at least $2^{k+1} - 1$ nodes.

Decision trees enable constructing complex classifiers from simple building blocks in a structured way. This is advantageous in many respects, the first being understandability. As the branching functions are usually simple, human experts can easily understand individual branching decisions. The tree structure provides further insight to the functioning of the classifier. For example, one can see why an example ended up in the leaf it did by backtracking its path to the root and looking at the branching functions on the way. As another example, it is commonly believed that the branching functions close to the root of the decision tree are important in classifying the examples as most of the examples have to go through these nodes on their way toward the leaves.

The structure of decision trees is central to learning them, too. Even though learning a small decision tree that has small empirical error is in general NP-

complete and inapproximable [27], there exist dozens of efficient greedy heuristics for decision tree learning that have been observed to perform relatively well on real world problems [49] and that can also be motivated theoretically in the weak learning framework [21]. These algorithms first grow a large decision tree that has small empirical error. In the second phase of the algorithms the tree is pruned in order to reduce its complexity and to improve its generalization performance.

The tree growing heuristics start from a single-node tree, which they then extend by iteratively replacing leaves of the current tree with new internal nodes. The choice of which leaf to replace, which branching function to use in the resulting new internal node, and how to label the new leaves differs from one algorithm to another (see, e.g., [49]). The common property is that all the algorithms try to greedily optimize the value of some heuristic that measures how well the partition of training data induced by the decision tree fits the labels of the data. The process of replacing leaves ends when the empirical error of the tree drops to zero or when adding new internal nodes does no longer help in reducing the value of the goodness measure.

The problem with growing decision trees is that the resulting tree is often very large and even of size linear in the number of the training examples [16, 52, 53]. The problem is especially severe on noisy learning domains on which the classes of the examples cannot be determined by a (simple) function of the attributes. Large decision trees lack all comprehensibility and (provable) generalization capability. In order to decrease the size of the trees and to improve their generalization performance the decision tree learning algorithms try to *prune* the tree. Pruning means replacing some subtrees of the original tree with leaves with the goal of reducing the size of the tree while maintaining or improving its generalization error. The pruning decisions are made based on the structure of the decision tree and on learning data, so that pruning can be viewed as learning, too.

There are lots of different pruning algorithms to choose from, most of them ad-hoc heuristics (see e.g. [57, 48, 24]) but some also with clear theoretical motivation [34, 29]. The majority of pruning algorithms makes their pruning decisions based on the same data set that was used in growing the tree (for some examples, see [57]), while some require a separate sample of pruning examples [14, 56] or work in an on-line fashion [29]. Also the goals of the algorithms vary — the focus may be on accuracy [56, 51], on small size [16, 11], on a combination of those two [34, 47], or on something completely different [44]. As the field of pruning algorithms is so diverse we will not even try to explore it here to any depth. Instead, we will go directly to reduced error pruning, the pruning algorithm analyzed in Paper 1.

3.2 Reduced Error Pruning

Reduced error pruning (REP) is an elementary pruning algorithm introduced by Quinlan [56]. The original description of the algorithm was quite loose and left much room (or need) for interpretation. As a consequence, there exists a whole family of different variants of the REP algorithm. Here, we will only consider the bottom-up version analyzed in Paper 1.

REP makes its pruning decisions based on a separate set of pruning examples. The overall learning strategy is thus to first split the learning sample randomly into a growing set and a pruning set. The growing set is then fed into a decision tree induction algorithm. Finally, the induced tree and the pruning set are given as input to the REP algorithm.

The intuition behind the pruning decisions of REP is the following. If a subtree does not improve the classification performance over the best single-node decision tree on pruning data, then the subtree is most likely to fit noise or other irrelevant properties of the growing set and should be removed. Otherwise, the subtree is considered to be relevant for improving classification accuracy on future data, too, and is retained. The subtrees to be removed by the above criterion can be found in linear time by a single bottom-up sweep of the tree to be pruned — for algorithmic details, see Paper 1. The result of REP is what remains after these removals.

The performance of REP on benchmark learning tasks is good but still slightly worse than the performance of the best known pruning heuristics [48]. One reason for the slightly inferior results is that as REP requires a separate pruning set, less data remains for the tree growing phase. The unpruned tree that REP starts with may thus be worse than the one that its rival pruning algorithms not requiring a separate pruning set get to work on. It has also been claimed that REP prunes too aggressively removing also relevant parts of the tree [56, 24].

The main advantage of REP is its simplicity which makes it easier to analyze than most other pruning algorithms that rely on complex heuristics and empirically tuned parameters. Our analysis of REP is a follow-up to an earlier analysis of Oates and Jensen [54]. Their intention was to use REP to explain the empirically observed phenomenon that the size of pruned decision trees tends to grow linearly in the size of the set of learning data [16, 52, 53]. In other words, the pruning phase of decision tree induction is not able to keep the complexity of the resulting classifier under control, even on domains on which the added complexity cannot yield any improvement in classification accuracy. We try to explain the same phenomenon, but using different techniques in order to make the analysis more rigorous and less dependent on unrealistic assumptions.

3.3 Analysis of Reduced Error Pruning

After clarifying the differences between the variants of REP that live on in the literature, we show that the version of REP that in our eyes seems to be the correct one solves a well-defined optimization problem, namely the following:

Reduced Error Pruning (REP)

Instance: A decision tree T and a set E of pruning examples.

Goal: Find the pruning T' of T that is the smallest among the prunings of T having the minimal empirical error on E.

The fact that REP is a solution to the above problem has been known and used before, but to our knowledge a rigorous proof has not been published previously. As a corollary we get that REP is an ERM algorithm for the class of prunings of the given tree T, a property of great importance to our work in Paper 4.

Another thing we analyze is how the pruning process of REP proceeds through the decision tree. To formulate our main result in this direction we need to define the concept of a *safe node*. First, say a node belongs to the *fringe* of a tree if it has leaves as children, or is a child of a node belonging to the fringe. The safe nodes are the fringe nodes whose parents do not belong to the fringe. Our theorem (Theorem 4) says that a sub-tree will be pruned to a leaf if and only if

- all subtrees rooted at its safe nodes are pruned and
- the majority class of pruning examples in all its safe nodes is the same.

These properties of the REP algorithm are the basis for our analysis of its behavior under various probabilistic assumptions. Following [54], we concentrate on situations in which the attributes $x \in \mathcal{X}$ of the pruning examples are independent of their class labels $y \in \mathcal{Y}$ while nothing is assumed about the tree to be pruned. Even though it is unrealistic to assume that this independence assumption holds in the whole tree, it may be used in approximating REP's behavior in subtrees that fit only noise. Given that the class is independent of the attributes, a decision tree that consists of a single leaf predicting the majority class of the pruning examples is clearly the smallest classifier with minimal generalization error. Nevertheless, we show in two different settings that REP will with high probability end up with a more complex decision tree.

In the first setting, we assume that the class labels of the n pruning examples are chosen independently of their attributes and that the probability p of the most probable class label is at least 0.5. We consider pruning a tree with k safe nodes

and assume that all safe nodes receive at least one example. With these assumptions we can prove that the probability that the whole tree is pruned to a leaf is at most

$$\left(\Phi\left(\frac{(p-1/2)\sqrt{r}}{\sqrt{p(1-p)}}\right)\right)^{k/2}, \tag{3.1}$$

where $r = 2n/k$ and Φ is the distribution function of the normal distribution. It is obvious that if we let k increase while n and p are fixed, the pruning probability drops to 0 exponentially fast. The same happens even if we let n grow with k so that r remains constant. Thus, we have shown that in such cases the pruning probability of a (sub)tree is low even if it fits pure noise.

In the previous analysis we made no assumptions on the distribution of examples to the nodes of the tree to be pruned, apart from assuming that all safe nodes receive a non-empty set of pruning examples. In the second setting, we assume again that the attributes contain no information of the class labels, but this time we also assume that the pruning examples are distributed uniformly to the safe nodes. This additional assumption enables us to prove slightly tighter upper bounds for the pruning probability of the root node and allows us take the contribution of empty safe nodes into account. The bound we get resembles the bound (3.1), but as both the bound and the analysis leading to it are less elegant, we refer the reader to the paper for details.

Using the bound (3.1) we can finally attack the original question, i.e., why REP fails to keep the size of pruned decision trees under control even in cases where the decision tree with best generalization accuracy is known to be small. We give a concrete example in which

- the class labels are independent from attributes and
- any decision tree with zero empirical error on the set of growing data has to have expected size linear in the number of growing examples.

Suppose we get increasing amounts of data to learn from and that we use a fixed proportion of the data for growing a tree and the rest for pruning it. In this example, the size of the grown tree increases at least linearly with the amount of learning data. If we hypothesize that the number of safe nodes receiving examples in the grown tree grows linearly, too, then we can apply the bound (3.1) to show that the probability of pruning the tree to a single node drops exponentially to zero. This, of course, is still far from fully understanding the linear growth of pruned decision trees, but still provides some insight to the question.

Chapter 4

The Difficulty of Branching Program Pruning

In this chapter we consider branching programs, a generalization of decision trees. After a brief introduction to the subject in Section 4.1, we present the main hardness result of Paper 2 in Section 4.2.

4.1 Branching programs and learning them

Branching programs (BPs) are a generalization of decision trees in which the graph underlying the classifier may be an arbitrary finite rooted directed acyclic graph. The root of the graph has in-degree 0, while all other nodes have in-degree at least 1. Nodes with out-degree 0 are called leaves. As in a decision tree, each non-leaf node of a BP is associated with a branching function mapping the attribute space $\mathcal{X}$ to the node's children, and the leaves are labeled with class labels $y \in \mathcal{Y}$. The classification of examples is done exactly as in the case of decision trees. For two examples of (visualizations of) branching programs, see Figure 4.1.

The advantage of branching programs over decision trees is that in branching programs substructures of a classifier can be shared. This may lead to exponential savings in the size of the classifier as explained in Paper 2. Not only can BPs represent classifiers more concisely, but the BP representations can also be learned from examples using a recently introduced boosting algorithm [45]. The algorithm and its analysis are motivated by the boosting analysis of greedy decision tree growing heuristics [21]. The bounds obtained in the case of BPs are substantially tighter than the best known bounds for growing DTs — the upper bound on the size of induced BPs with given empirical error guarantees is exponentially smaller than the one on the size of induced DTs, provided that both classes of classifiers utilize the same set of branching functions. The bounds, however, depend on a weak learning parameter measuring the power of the class of the underly-

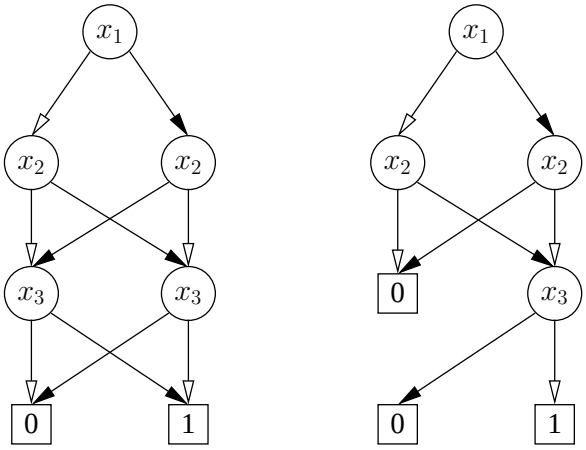

Figure 4.1: A minimal branching program counting the exclusive-or of three bits (left) and one of its prunings (right).

ing branching functions on the example distributions encountered in the induction process. Since these distributions and hence also the behavior of the weak learning parameter varies between the algorithms, a direct comparison of the bounds does not make sense. Unfortunately, no theory concerning the behavior of the weak learning parameter currently exists.

Although the theoretical results hint that branching programs might be exponentially smaller than the corresponding decision trees, this size advantage does not seem to materialize in practice — branching programs and decision trees induced from benchmark data sets seem to be approximately of the same size [22]. Thus, it seems natural to try to add a pruning phase to branching program learning. Our empirical experiments indicate that heuristic branching program pruning is indeed advantageous [22] — it reduces the size of the branching programs while maintaining their accuracy. However, as shown in Paper 2, finding even an approximately most accurate pruning of branching programs is intractable, provided that P$\neq$NP.

4.2 Hardness results

Before presenting the hardness results proved in Paper 2, let us first define the *branching program reduced error pruning* (BPREP) optimization problem formally. Analogously to decision tree pruning, we define a pruning of a branching program P to be a branching program P' that is obtainable by the following procedure:

1. Select a subset of the nodes of P.
2. Convert the selected nodes to leaves by labeling them with class $y \in \mathcal{Y}$ labels and removing all outgoing arcs attached to them.
3. Remove all parts of P that become inaccessible from the root after removing the arcs in step 2.

With this definition of branching program prunings, we can formulate the BPREP minimization problem as follows.

Branching Program Reduced Error Pruning (BPREP)

Instance: A branching program P, a set E of pruning examples with boolean attribute vectors, and a weight function $w\colon E \rightarrow [0,1]$ satisfying $\sum_{e \in E} w(e) = 1$.

Feasible solutions: Prunings P' of P.

Cost Function: The sum of weights of examples misclassified by the pruning P'.

The weight function here is included mostly for convenience, the hardness results apply to the unweighted case as well. Observe also that this formulation does not take the size of the prunings into account. This, of course, only simplifies the computational problem and makes the hardness results stronger.

Our first result is that the decision problem corresponding to the minimization problem BPREP is NP-complete. Thus, there is no hope in finding a polynomial time algorithm solving the BPREP problem exactly, unless P=NP. The NP-completeness result was proved originally in [23] where we also show that if P≠NP, the optimization version of BPREP cannot be approximated to within 1.006.

The inapproximability ratio obtained in [23] is still so close to 1 that an approximate algorithm sufficient for all practical purposes could well exist without contradicting the hardness results. After all, an increase of at most 0.6% in the empirical error of a pruning would under most conceivable circumstances be negligible compared to stochastic variation. In Paper 2 we settle the question of whether a practical BPREP algorithm exists by presenting stronger inapproximability theorem whose proof also yields a simpler proof for our original NP-completeness result.

Theorem 4.2.1 *Unless NP $\subset$ DTIME($n^{\log\log n}$), BPREP is not approximable to within $\log^{1-\delta} k$ for any $\delta > 0$, where k is the number of components in the boolean attribute vectors.*

This result on the inapproximability of BPREP shows that finding accurate prunings of BPs is essentially harder than it is for decision trees. Thus, we have to pay a price for the representational efficiency of branching programs in an increase in the computational complexity of manipulating them. Another drawback in branching program representations of classifiers is that they cannot be visualized or comprehended as easily as decision trees can. Since our empirical results on growing BPs were not too promising either [22], there seems to be little reason for using them instead of decision trees.

Chapter 5

Progressive Rademacher Sampling Applied to Small Decision Trees

In this chapter we consider progressive sample complexity estimation, a dual problem for data dependent generalization error analysis. Section 5.1 introduces a variant of the estimation problem which is then solved using progressive Rademacher sampling in Section 5.2. The provided solution not only yields reasonably good results on benchmark data sets, but also has provable optimality properties.

5.1 Progressive Sampling

In generalization error analysis we are given a sample of n learning examples and a confidence parameter $\delta > 0$. Based on these, we have to produce a classifier and an upper bound on its generalization error that has to be true with probability at least $1 - \delta$. Thus, the goal is to find a classifier with as good a generalization error bound as possible, given a learning sample of a fixed size. Sometimes, however, the dual problem called *sample complexity estimation* is a more natural formulation. In traditional sample complexity estimation learning takes place according to the following protocol. The learner is first given only an *accuracy parameter* $\varepsilon > 0$ and a confidence parameter $\delta > 0$. It then has to choose a learning sample size N having the property that with probability at least $1 - \delta$, the generalization error of the classifier the learner would choose based on a randomly chosen learning sample of size N is at most ε larger than its empirical error. Finally, the learner gets a learning sample of size N which it then uses to select its final hypothesis.

The primary motivation for the criterion the sample complexity estimate N has to meet is that in the *realizable case* — the case in which the hypothesis class used by the learner is guaranteed to contain a hypothesis with perfect generalization — the criterion ensures that the generalization error of any ERM hypothesis

will be below ε with probability at least $1 - \delta$, too. The criterion is less natural in the more realistic non-realizable case in which the existence of a consistent hypothesis cannot be guaranteed. However, it still implies that the ERM hypothesis for a sample of size N has with high probability nearly optimal generalization performance.

The sample complexity estimate N may depend on the parameters ε and δ, the hypothesis class, and more generally on all kinds of background information like the properties of the learning algorithm. However, as the learner gets the learning sample only after outputting the sample size estimate N, the estimate itself cannot depend on the sample. This kind of data independent sample complexity analysis has a long history in learning theory dating back to the very first papers on PAC learning [62]. Traditionally, sample complexity bounds have relied on data independent generalization error bounds like the VC dimension based bound presented as Theorem 2.2.3. The penalty term A (cf. Section 2.2.1) in these kinds of data independent bounds depends only on the hypothesis class $\mathcal{H}$, the confidence parameter δ, and the sample size n. Thus, one can turn them into sample complexity bounds simply by solving n from the equation $A(\mathcal{H}, \delta, n) = \varepsilon$ that does not depend on the learning data in any way. As data independent sample complexity bounds can be converted into generalization error bounds in a similar fashion, the tasks of generalization error and sample complexity analysis are essentially equivalent when only data independent methods are considered.

Transforming data dependent generalization error bounds to sample complexity bounds is somewhat more challenging. This is because the penalty term A now depends on the sample and the equality $A = \varepsilon$ cannot thus be solved for the sample size n without seeing a sample of that size first. Hence, the sample complexity analysis framework has to be refined in order to make data dependent sample complexity estimation possible. One way to do this is to extend the learning model to allow *progressive sampling*, a scheme in which the learner is allowed to increase the size of the learning sample iteratively until some *stopping criterion* is met.

In practice, progressive sampling is one of the methods of estimating the amount of learning data that is needed in order to find a classifier with nearly optimal performance on the learning task. Using a minimal amount of learning data is beneficial for many reasons. For example, producing labeled learning examples is often difficult or at least expensive. Also, increasing the size of the learning data set can be computationally intolerably costly, especially if the learning algorithm has high time or space complexity. Further motivation can be found in Paper 3.

Our basic idea is to evaluate the penalty term $A = A_n$ on increasing samples of sizes $n_0, n_1, \ldots$ belonging to some *sampling schedule* $\{n_i \mid i \in \mathbb{N}\}$ until a

sample size n_k is found for which $A_{n_k} \leq \varepsilon$. The intuition is that this n_k is the smallest sample size belonging to the sampling schedule that suffices to guarantee that the empirical error of every hypothesis is within ε of its generalization error with at least probability $1 - \delta$. The matter is, however, complicated by the fact that the bounds A_{n_i} are now random variables that depend on the increasing learning samples. Thus, even if each of the generalization error bounds based on the random penalty terms A_{n_i} were true with probability at least $1 - \delta$, the bound based on the random sample size $\min\{n_i \mid A_{n_i} \leq \varepsilon\}$ might still fail with probability larger than δ. We will show how to cope with these complications using novel techniques introduced by Koltchinskii et al. [38] in a controller design context. As a result we get a progressive sampling scheme that yields nearly optimal sample size estimates in a sense described in detail in the next section.

The progressive sampling methods that have been observed to perform well in practice are typically based on learning curve extrapolation [32, 55]. As the theoretical understanding concerning the behavior of learning curves is rather limited, the methods based on learning curve assumptions are necessarily heuristic in nature. The method we propose next is not competitive with these heuristics, but clearly outperforms the earlier theoretically motivated approaches.

5.2 Progressive Rademacher Sampling

In this section, we will adapt the techniques originally developed by Koltchinskii et al. [38] for solving difficult controller design problems to fit our progressive sampling framework. These techniques rely heavily on the theory of empirical processes [63] and especially on Rademacher penalization. For a brief introduction to the latter, see Section 2.2.3.

In order to guarantee that with probability at least $1 - \delta$ the empirical errors of all hypotheses $h \in \mathcal{H}$ are within ε of their true errors, we need to have a sample of size at least

$$n^P_{\min}(\varepsilon, \delta) = \min \left\{ n \in \mathbb{N} \mid \mathbf{Pr} \left(\sup_{h \in \mathcal{H}} |\epsilon(h) - \hat{\epsilon}_n(h)| \geq \varepsilon \right) \leq \delta \right\}.$$

Here, $n^P_{\min}(\varepsilon, \delta)$ depends on P directly (through the generalization errors $\epsilon(h)$), so that it cannot be used as a sample complexity estimate — it merely represents an ideal sample size to which the sample complexity estimates produced by realizable estimation schemes can be compared.

We will restrict our attention to *geometric sampling schedules* in which $n_i = 2^i n_0$, where $n_0 = n_0(\varepsilon, \delta)$ is some fixed initial sample size. The main benefit of using a geometric schedule is that if the time complexity of the learning algorithm is super linear, the computation time lost in learning intermediate hypotheses is

dominated by the time used in learning the final hypothesis [55]. Thus, the time complexity of geometric sampling is only a constant times larger than the time complexity of the underlying learning algorithm on the final sample size.

As the sampling schedule itself is fixed in advance, the remaining thing is to derive a *stopping criterion* that determines the sample size $n_i \in \mathbb{N}$ at which to stop sampling. For the theory to go through, the stopping criterion has to be a *stopping time* [33] — informally, a random variable τ taking non-negative integral values and satisfying the additional condition that, for each $n \in \mathbb{N}$, the event $\{\tau = n\}$ depends only on information available by time n. Following [38], a stopping criterion τ is called *well-behaving* with parameters ε, δ if $\tau \geq n_0(\varepsilon, \delta)$ and

$$\mathbf{Pr}\left(\sup_{h \in \mathcal{H}} |\hat{\epsilon}_\tau(h) - \epsilon(h)| \geq \varepsilon\right) \leq \delta.$$

An immediate consequence of this definition is that if τ is well-behaving with parameters ε, δ and $\hat{h}$ is a hypothesis that minimizes empirical risk based on the sample $\{(x_i, y_i) \mid i = 1, \ldots, \tau\}$, then

$$\mathbf{Pr}\left(\epsilon(\hat{h}) \geq \inf_{h \in \mathcal{H}} \epsilon(h) + 2\varepsilon\right) \leq \delta.$$

In other words, it is enough to draw τ examples in order to find, with high probability, a hypothesis that is almost as accurate as the most accurate one in $\mathcal{H}$.

Now that we have a general definition for a stopping criterion, we can use Rademacher penalties (defined by equation (2.7) on page 13) to define the *Rademacher stopping time* $\nu(\varepsilon, \delta)$ with parameters (ε, δ) for the hypothesis class $\mathcal{H}$ as

$$\nu(\varepsilon, \delta) = \min_{i \in \mathbb{N}} \left\{ n_i = 2^i n_0(\varepsilon, \delta) \mid R_{n_i}(\mathcal{H}) < \varepsilon \right\}.$$

Here, a reasonable choice for the initial sample size $n_0(\varepsilon, \delta)$ is

$$n_0(\varepsilon, \delta) = \left\lfloor \frac{4}{\varepsilon^2} \log\left(\frac{4}{\delta}\right) \right\rfloor + 1;$$

that is, the least choice satisfying the preconditions of the next two theorems. The theorems are rather straightforward modifications of the corresponding results presented in [38].

Theorem 5.2.1 *Let*

$$n_0(\varepsilon, \delta) \geq \left\lfloor \frac{4}{\varepsilon^2} \log\left(\frac{4}{\delta}\right) \right\rfloor + 1.$$

Then, for all $\varepsilon > 0$ *and* $\delta \in (0, 1)$*,*

1. $\nu(\varepsilon, \delta)$ *is well-behaving with parameters* $(5\varepsilon, \delta)$*.*

2. *If $n^P_{min}(\varepsilon,\delta) \geq n_0(\varepsilon,\delta)$, then for all $\varepsilon > 0$ and $\delta \in (0,1/2)$, the probability that $\nu(24\varepsilon,\delta) > n^P_{min}(\varepsilon,\delta)$ is at most 3δ (for any class $\mathcal{H}$ of hypotheses and any distribution P).*

Theorem 5.2.2 *If*

$$n_0(\varepsilon,\delta) \geq \left\lfloor \frac{4}{\varepsilon^2} \log\left(\frac{4}{\delta}\right) \right\rfloor + 1$$

and $12/\varepsilon \leq n^P_{min}(\varepsilon,\delta) \leq n_0(\varepsilon,\delta)$*, then*

$$\mathbf{Pr}\left(\nu(30\varepsilon,\delta) > 2n_0(\varepsilon,\delta)\right) \leq \delta.$$

The theorems show that, under certain mild conditions, $\nu(\varepsilon,\delta)$ is well-behaving and, furthermore, that it yields sample size estimates that are comparable to the unknown optimal distribution dependent sample complexity $n^P_{\min}(\varepsilon,\delta)$. This is in clear contrast to the stopping times that one could define using the generalization error bounds based on VC dimension or other distribution independent complexity measures as those would be competitive with $n^P_{\min}(\varepsilon,\delta)$ for worst-case P only.

To test the performance of progressive Rademacher sampling in practice, we conducted experiments on some benchmark data sets from the UCI machine learning repository [7]. As our learning algorithm we used T2 [3], an ERM algorithm for learning two-level decision trees with discrete and continuous attributes. Thus, the set of hypotheses $\mathcal{H}$ consists in this case of all two-level decision trees learnable by T2. Even though this hypothesis class is relatively simple, its good performance on real world learning tasks shows that it is both non-trivial and interesting from a practical point of view. The fact that T2 is an ERM algorithm for this hypothesis class enables us to compute the associated Rademacher penalties and hence also the stopping times $\nu(\varepsilon,\delta)$ efficiently by simply following the strategy described in Section 2.2.3.

In the experiments we compared the sample complexity estimates provided by Rademacher penalization to those obtainable using the bounds based on VC dimension. The results show that the sample complexity estimates obtainable using progressive Rademacher penalization can be substantially smaller than the estimates based on data independent VC dimension bounds. For example, on the Adult (Census) domain the estimate given by the proposed method is of the order 60,000 while the VC dimension based estimate is approximately 2,623,000. On the other hand, the optimal sample size for the more complex problem of learning decision trees using C4.5 as determined empirically by Provost et al. [55] using a heuristic learning curve sampling method [52] is around 8,000. Thus, progressive

Rademacher sampling does not seem to be competitive with the heuristic stopping time determination methods used in practice, whereas it seems to clearly outperform other theoretically sound sample complexity estimation schemes. For more details on the experiments including some plots on the Adult data set, see Paper 3.

Chapter 6

Generalization Error Bounds for Decision Tree Prunings Using Rademacher Penalties

In this chapter we show how Rademacher penalization can be used to derive tight data dependent generalization error bounds for decision tree prunings. Before presenting the main idea of Paper 4 in Section 6.2 we first describe how one can compute Rademacher penalties in multiclass settings. The presented ideas result in a generalization error analysis scheme that yields non-trivial error bounds for general decision tree prunings with little computational overhead.

6.1 Evaluating Rademacher Penalties in a Multiclass Setting

Multiclass learning tasks — tasks in which the label set $\mathcal{Y}$ has more than two elements — are common in practice. Still, the standard forms of traditional generalization error bounds like the VC bounds are applicable only in two-class settings. Unlike them, the bounds based on Rademacher penalization remain true as long as the *loss function* according to which the cost of (mis)classifications is measured is bounded [37]. In our case, the loss function is the obviously bounded 0-1 loss $\ell\colon \mathcal{Y} \times \mathcal{Y} \to \{0,1\}$ given by $\ell(y,y') = [\![y \neq y']\!]$. Thus, the error bounds based on Rademacher penalties are applicable irrespective of the cardinality of $\mathcal{Y}$ — provided that we can evaluate them efficiently.

In Paper 4, we present a derivation that leads to the following strategy for computing the values of Rademacher penalties in multiclass settings. Instead of flipping class labels that was enough in the two-class case, we now replace a class label $y \in \mathcal{Y}$ by its *complement class* $\bar{y} \in \bar{\mathcal{Y}} = \{\bar{z} = \mathcal{Y} \setminus \{z\} \mid z \in \mathcal{Y}\}$

that represents all the classes but y. Accordingly, an example with class $\bar{y}$ is considered correctly classified if its classification is in $\bar{y}$ and misclassified only if the classification is y. The classifications themselves are still required to belong to $\mathcal{Y}$.

With this notation, the computation of the Rademacher penalty $R_n(F)$ for a class of multi-class classifiers F proceeds as follows.

1. Toss a fair coin n times to obtain a realization of the Rademacher random variable sequence $r_1, \ldots, r_n$.

2. Change the label y_i to $\bar{y}_i$ if and only if $r_i = +1$ to obtain a new sequence of labels $z_1, \ldots, z_n$.

3. Find functions $h_1, h_2 \in F$ that minimize the empirical error with respect to the set of labels z_i and $\bar{z}_i$, respectively. Here, we follow the convention that $\bar{\bar{z}} = z$ for all $z \in \mathcal{Y} \cup \bar{\mathcal{Y}}$.

4. The Rademacher penalty is given by the maximum of $|\{i : r_i = +1\}|/n - \hat{\epsilon}(h_1)$ and $|\{i : r_i = -1\}|/n - \hat{\epsilon}(h_2)$, where the empirical errors $\hat{\epsilon}(h_1)$ and $\hat{\epsilon}(h_2)$ are with respect to the labels z_i and $\bar{z}_i$, respectively.

This strategy obviously bears a strong resemblance to the two-class version introduced by Koltchinskii [37] and reviewed in Section 2.2.3. However, the optimization problem in step 3 is now a bit more involved as the optimization algorithm has to cope with complement classes, too. Fortunately, both of the ERM algorithms that we have considered in this Thesis can easily be extended to handle this more general setting. For details, see Paper 4.

6.2 Rademacher Penalization over Decision Tree Prunings

The class of all decision trees has infinite VC-dimension already if $\mathcal{X} = \mathbb{R}$ and the set of branching functions is $\{x \mapsto [\![x \leq \theta]\!] \mid \theta \in \mathbb{R}\}$. Thus, decision trees are not learnable in the PAC learning model [10]. This means in particular that there is no hope in proving data independent generalization error bounds for them. We will, nevertheless, present a data dependent generalization error analysis methodology that enables us to prove non-trivial bounds for unrestricted decision trees learned by two-phase decision tree learning algorithms. Thus, by making use of the information in the learning sample we will achieve something that would be provably impossible otherwise.

The decision tree learning strategy we consider here is the one outlined in Section 3.2. In particular, in addition to the growing set there is a separate set

of pruning examples. The growing set is used in inducing a decision tree that is then pruned using the pruning data. The key idea in our approach is to view the pruning phase as empirical risk minimization in the class of prunings of the induced decision tree. As the tree induction phase is not restricted in any way, the tree to be pruned and hence also its prunings may be arbitrary. Still, we are able to provide non-trivial error bounds for this data dependent class of prunings on real world learning domains.

It is interesting to relate the proposed approach to the dichotomy of generalization error bounds suggested by Langford [40] and briefly discussed in Section 2.2.1. Our bound resembles test set bounds in that part of the data is put aside in the tree growing phase. Still, this set of data is not reserved for testing purposes only as the pruning phase of decision tree learning is proper learning, too. Our approach uses the pruning set as a test set for the tree growing phase — thus enabling us to prove generalization error bounds for unrestricted decision trees — and as a standard training set for the pruning phase.

In summary, the strategy we propose in Paper 4 is the following:

1. Split the learning sample randomly into a growing set and a pruning set.

2. Choose one of the available decision tree growing heuristics (or even better, invent one of your own) and induce a decision tree by applying it to the set of growing examples.

3. Prune the tree built in the previous step by feeding it and the pruning set to REP (or your own favorite pruning algorithm — this time deviating from REP is never advantageous, though, if error bounds are the only concern). The obtained pruning is the final hypothesis.

4. Evaluate the error bound based on the Rademacher penalty corresponding to the class of prunings F of the decision tree induced in step 1.

Evaluating the error bound in step 4 can be done efficiently as REP is a linear time ERM algorithm for the class of prunings of a decision tree. Even multiclass problems present no problem by the arguments given in the previous section. For example, if one used REP for pruning anyway, the proposed approach would provide error bounds for the final hypothesis for the price of two additional runs of the REP algorithm. Most importantly, the proposed bounds are training set bounds in that no learning data has to be reserved for testing purposes.

Of course, there is no magic in splitting the learning data into separate growing and pruning sets — we are still unable to provide non-trivial generalization error bounds for general decision trees on all learning domains. The bounds we provide are potentially informative only in case the decision tree induction heuristic

is successful, i.e., the induced decision tree is both sufficiently small and contains prunings with low generalization error. Otherwise, the hypothesis class we work in the pruning phase would either be too complex or its hypotheses would all be inaccurate. The conditions under which heuristic decision tree growing succeeds may not be easily characterizable, but empirical experiments have undeniably shown that decision tree learning performs well on a wide variety of real world learning domains.

The empirical success of decision tree induction is only a necessary but not a sufficient condition for our generalization error analysis approach to work well. It could still be the case that the complexity penalty term in Rademacher penalization would overestimate the true complexity of the class of prunings of a decision tree, thus resulting in loose and uninteresting generalization error bounds. Fortunately, our experiments on UCI benchmark data sets indicate that our bounds are tighter than the previous data independent bounds for decision tree prunings and that they may give information that could be of some value in practice. Still, the bounds leave room for further improvement as they are on all domains significantly above the test error bounds.

Chapter 7

Conclusions

The results in this Thesis lie somewhere between theoretical and practical machine learning. Our primary goal is not to advance the theory itself but only to bring it closer to practice. An ideal end product of such research would thus be a theoretically sound machine learning method that works well on real world problems. Of course, we did not quite reach this ambitious goal – the methods we proposed do have some theoretical performance guarantees, but they do not perform as well as the best ad-hoc heuristics for the tasks in question.

Our research on reduced error pruning yielded two-fold results. For decision trees, we extended the results of previous analyses and provided more rigorous derivations for some existing results. In particular, the fact that reduced error pruning is an ERM algorithm for the class of prunings of a decision tree turned out to be of major importance in our later work. On the other hand, the branching program version of the reduced error pruning problem turned out to be intractable, at least if P$\neq$NP. This hardness result further supports our belief that using branching programs in machine learning is a bad idea.

The first of our generalization error analysis related results is on sample complexity estimation. We highlighted the connection of progressive sampling and data dependent generalization error analysis, leading to a new data dependent sample complexity estimation scheme. The proposed progressive Rademacher sampling methodology is in theory nearly optimal among a wide class of sample complexity estimates. It also seems to work well in practice – at least better than the previously introduced theoretically sound methods.

The final result of this Thesis concerns generalization error bounds for prunings of an induced decision tree. These bounds are again based on Rademacher penalization. This time, however, the hypothesis class is determined by an induced decision tree and is thus data dependent. The bounds obtained are applicable to all decision tree learning algorithms that use a separate set of pruning data. To evaluate the bounds, we use the reduced error pruning algorithm for decision trees that

was studied earlier in this Thesis. Hence, the bound can be evaluated in linear time. Our empirical experiments suggest that the proposed methodology gives non-trivial training set bounds that clearly outperform the earlier bounds based on data independent complexity measures.

As future work, it would be interesting to explore further possibilities of tightening generalization error bounds in the statistical learning framework. Besides that, it might be fruitful to try to apply techniques similar to the ones used in this Thesis to other problems in data analysis and computer science in general. For example, the methods used in this Thesis can be used to derive generalization error bounds and sample complexity estimates in semi-supervised and active learning settings. Methods like Rademacher penalization might also give better sample complexity bounds for sampling based randomized algorithms, e.g., clustering algorithms based on random sampling. Thus, there still seems to be a lot of work to be done.

References

[1] Dana Angluin. Queries revisited. *Theoretical Computer Science*, 313(2):175–194, 2004.

[2] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, UK, 1999.

[3] Peter Auer, Robert C. Holte, and Wolfgang Maass. Theory and application of agnostic PAC-learning with small decision trees. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 21–29, San Francisco, CA, 1995. Morgan Kaufmann.

[4] Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Localized Rademacher averages. In Jyrki Kivinen and Robert H. Sloan, editors, *Proceedings of the 15th Annual Conference on Computational Learning Theory*, volume 2375 of *Lecture Notes in Computer Science*, pages 44–58, Berlin, Heidelberg, 2002. Springer Verlag.

[5] Peter L. Bartlett and Shahar Mendelson. Empirical risk minimization. Submitted for journal publication.

[6] Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[7] Cathrine L. Blake and Christopher J. Merz. *UCI Repository of Machine Learning Databases*. University of California, Department of Information and Computer Science, Irvine, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[8] Avrim Blum and John Langford. PAC-MDL bounds. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Learning Theory and Kernel Machines*, volume 2777 of *Lecture Notes in Artificial Intelligence*, pages 344–357, Berlin, Heidelberg, 2003. Springer Verlag.

[9] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, 1987.

[10] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):829–965, 1989.

[11] Marco Bohanec and Ivan Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15(3):223–250, 1994.

[12] Olivier Bousquet. New approaches to statistical learning theory. *Annals of the Institute of Statistical Mathematics*, 55(2):371–389, 2003.

[13] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[14] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[15] Leonard A. Breslow and David W. Aha. Simplifying decision trees: a survey. *Knowledge Engineering Review*, 12(1):1–40, 1997.

[16] Jason Catlett. Overpruning large decision trees. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 764–769, San Mateo, CA, 1991. Morgan Kaufmann.

[17] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[18] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[19] A. Philip Dawid. Prequential analysis, stochastic complexity and Bayesian inference. *Bayesian Statistics*, 109:109–125, 1992.

[20] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, NY, 1996.

[21] Tom Dietterich, Michael Kearns, and Yishay Mansour. Applying the weak learning framework to understand and improve C4.5. In Lorenza Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 96–104, San Francisco, CA, 1996. Morgan Kaufmann.

[22] Tapio Elomaa and Matti Kääriäinen. On the practice of branching program boosting. In Luc De Raedt and Peter Flach, editors, *Proceedings of the Twelfth European Conference on Machine Learning*, volume 2167 of *Lecture Notes in Artificial Intelligence*, pages 133–144, Berlin, Heidelberg, 2001. Springer-Verlag.

[23] Tapio Elomaa and Matti Kääriäinen. The difficulty of reduced error pruning of leveled branching programs. *Annals of Mathematics and Artificial Intelligence*, 41(1):111–124, 2004.

[24] Floriana Esposito, Donato Malerba, and Giovanni Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.

[25] Robert B. Evans and Douglas Fisher. Using decision tree induction to minimize process delays in the printing industry. In Willi Klösgen and Jan M. Żytkow, editors, *Handbook of Data Mining and Knowledge Discovery*, pages 874–881. Oxford University Press, 2002.

[26] Thore Graepel, Ralf Herbrich, and Robert C. Williamson. From margin to sparsity. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 210–216, 2001.

[27] Michelangelo Grigni, Vincent Mirelli, and Christos H. Papadimitriou. On the difficulty of designing good classifiers. *SIAM Journal on Computing*, 30(1):318–323, 2000.

[28] Torben Hagerup and Christine Rüb. A guided tour of Chernoff bounds. *Information Processing Letters*, 33(6):305–308, 1990.

[29] David P. Helmbold and Robert E. Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(1):51–68, 1997.

[30] Ralf Herbrich and Robert C. Williamson. Algorithmic luckiness. *Journal of Machine Learning Research*, 3:175–212, 2002.

[31] Edwin T. Jaynes and G. Larry Bretthorst (editor). *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[32] George H. John and Pat Langley. Static versus dynamic sampling for data mining. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 367–370. AAAI Press, 1996.

[33] Olav Kallenberg. *Foundations of Modern Probability*. Springer Verlag, New York, NY, 2002. Second edition.

[34] Michael Kearns and Yishay Mansour. A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. In Jude Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 269–277, San Francisco, CA, 1998. Morgan Kaufmann.

[35] Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2/3):115–141, 1994.

[36] Jyrki Kivinen. Online learning of linear classifiers. In Shahar Mendelson and Alex J. Smola, editors, *Advanced Lectures on Machine Learning: Machine Learning Summer School 2002*, volume 2600 of *Lecture Notes in Artificial Intelligence*, pages 235–257. Springer Verlag, Heidelberg, 2003.

[37] Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001.

[38] Vladimir Koltchinskii, C. T. Abdallah, M. Ariola, P. Dorato, and D. Panchenko. Improved sample complexity estimates for statistical learning control of uncertain systems. *IEEE Transactions on Automatic Control*, 45(12):2383–2388, 2000.

[39] Vladimir Koltchinskii and Dmitriy Panchenko. Rademacher processes and bounding the risk of function learning. In Evarist Gine, David M. Mason, and Jon A. Wellner, editors, *High-Dimensional Probability II*, pages 443–459. Birkhäuser, Basel, 2000.

[40] John Langford. Combining training set and test set bounds. In Claude Sammut and Achim G. Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 331–338, San Francisco, CA, 2002. Morgan Kaufmann.

[41] John Langford. Basic sample complexity. Available electronically at `http://www.cs.cmu.edu/~jcl/research/tutorial/tutorial.ps`, 2003.

[42] Pat Langley and Herbert A. Simon. Applications of machine learning and rule induction. *Communications of the ACM*, 38(11):54–64, 1995.

[43] Ming Li and Paul Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Verlag, New York, NY, 1997. Second Edition.

[44] Ian MacNaughton (director), Graham Chapman, John Cleese, Terry Gilliam, Eric Idle, Terry Jones, and Michael Palin. Monty Python's And Now for Something Completely Different, 1971. For availability, see `http://us.imdb.com/title/tt0066765/`.

[45] Yishay Mansour and David McAllester. Boosting using branching programs. In Nicolò Cesa-Bianchi and Sally Goldman, editors, *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 220–224, San Francisco, CA, 2000. Morgan Kaufmann.

[46] David A. McAllester. PAC-Bayesian stochastic model selection. *Machine Learning*, 51(1):5–21, 2003.

[47] Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. MDL-based decision tree pruning. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 216–221, Montreal, Canada, 1995. AAAI Press.

[48] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4(2):227–243, 1989.

[49] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3(4):319–342, 1989.

[50] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[51] Tim Niblett and Ivan Bratko. Learning decision rules in noisy domains. In Max A. Bramer, editor, *Research and Development in Expert Systems III*, pages 25–34, Cambridge, UK, 1986. Cambridge University Press.

[52] Tim Oates and David Jensen. The effects of training set size on decision tree complexity. In Douglas H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 254–261, San Francisco, CA, 1997. Morgan Kaufmann.

[53] Tim Oates and David Jensen. Large datasets lead to overly complex models: An explanation and a solution. In Rakesh Agrawal, Paul Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 294–298, Menlo Park, CA, 1998. AAAI Press.

[54] Tim Oates and David Jensen. Toward a theoretical understanding of why and when decision tree pruning algorithms fail. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 372–378, Menlo Park, CA/Cambridge, MA, 1999. AAAI Press/MIT Press.

[55] Foster Provost, David Jensen, and Tim Oates. Efficient progressive sampling. In Kyuseok Shim, editor, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 23–32, New York, NY, 1999. ACM Press.

[56] J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–248, 1987.

[57] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[58] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[59] Nicolas Sauer. On the densities of families of sets. *Journal of Combinatorial Theory - Series A*, 13:145–147, 1972.

[60] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

[61] Michél Talagrand. A new look at independence. *Annals of Probability*, 24(1):1–34, 1996.

[62] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[63] Aad W. van der Vaart and Jon A. Wellner. *Weak Convergence and Empirical Processes With Applications to Statistics*. Springer Verlag, New York, NY, 2000.

[64] Vladimir N. Vapnik. *Estimation of Dependencies based on Empirical Data*. Springer Verlag, New York, NY, 1982.

[65] Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, NY, 1998.

[66] Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

[67] Vladimir Vovk. Aggregating strategies. In Mark A. Fulk, editor, *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.

[68] Vladimir Vovk. On-line compression modelling. A description of the approach and a list of working papers is available at `http://www.vovk.net/kp/index.html`, 2004.

[69] David H. Wolpert. On the connection between in-sample testing and generalization error. *Complex Systems*, 6(1):47–94, 1992.

TIETOJENKÄSITTELYTIETEEN LAITOS
PL 68 (Gustaf Hällströmin katu 2 b)
00014 Helsingin yliopisto

DEPARTMENT OF COMPUTER SCIENCE
P.O. Box 68 (Gustaf Hällströmin katu 2 b)
FIN-00014 University of Helsinki, FINLAND

JULKAISUSARJA **A**

SERIES OF PUBLICATIONS **A**

Reports may be ordered from: Kumpula Science Library, P.O. Box 64, FIN-00014 University of Helsinki, FINLAND.

A-2002-2 V. Ollikainen: Simulation techniques for disease gene localization in isolated populations. 149+5 pp. (Ph.D. thesis)

A-2002-3 J. Vilo: Discovery from biosequences. 149 pp. (Ph.D. thesis)

A-2003-1 J. Lindström: Optimistic concurrency control methods for real-time database systems. 111 pp. (Ph.D. thesis)

A-2003-2 H. Helin: Supporting nomadic agent-based applications in the FIPA agent architecture. 200+17 pp. (Ph.D. thesis)

A-2003-3 S. Campadello: Middleware infrastructure for distributed mobile applications. 164 pp. (Ph.D. thesis)

A-2003-4 J. Taina: Design and analysis of a distributed database architecture for IN/GSM data. 130 pp. (Ph.D. thesis)

A-2003-5 J. Kurhila: Considering individual differences in computer-supported special and elementary education. 135 pp. (Ph.D. thesis)

A-2003-6 V. Mäkinen: Parameterized approximate string matching and local-similarity-based point-pattern matching. 144 pp. (Ph.D. thesis)

A-2003-7 M. Luukkainen: A process algebraic reduction strategy for automata theoretic verification of untimed and timed concurrent systems. 141 pp. (Ph.D. thesis)

A-2003-8 J. Manner: Provision of quality of service in IP-based mobile access networks. 191 pp. (Ph.D. thesis)

A-2004-1 M. Koivisto: Sum-product algorithms for the analysis of genetic risks. 155 pp. (Ph.D. thesis)

A-2004-2 A. Gurtov: Efficient data transport in wireless overlay networks. 141 pp. (Ph.D. thesis)

A-2004-3 K. Vasko: Computational methods and models for paleoecology. 176 pp. (Ph.D. thesis)

A-2004-4 P. Sevon: Algorithms for Association-Based Gene Mapping. 101 pp. (Ph.D. thesis)

A-2004-5 J. Viljamaa: Applying Formal Concept Analysis to Extract Framework Reuse Interface Specifications from Source Code. 206 pp. (Ph.D. thesis)

A-2004-6 J. Ravantti: Computational Methods for Reconstructing Macromolecular Complexes from Cryo-Electron Microscopy Images. 100 pp. (Ph.D. thesis)

A-2004-7 M. Kääriäinen: Learning small trees and graphs that generalize. 45+49 pp. (Ph.D. thesis)

ISSN 1238-8645
ISBN 952-10-2050-4 (paperback)
ISBN 952-10-2051-2 (PDF)
Helsinki 2004
Helsinki University Printing House