

THE UNIVERSITY OF CHICAGO

ALGORITHMIC STABILITY AND ENSEMBLE-BASED LEARNING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
SAMUEL KUTIN

CHICAGO, ILLINOIS

JUNE 2002

Copyright © 2002 by Samuel Kutin
All rights reserved.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	iv
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF SYMBOLS	xii
ABSTRACT	xvi
1 INTRODUCTION	1
1.1 An algorithmic theory of learning	1
1.1.1 Overview	2
1.2 Generalization error and VC dimension	3
1.3 Algorithmic stability	6
1.3.1 Independent bounded differences	9
1.4 Ensemble-based learning	11
1.4.1 The algorithmic stability of AdaBoost	12
1.4.2 Decorrelation	13
1.4.3 Lower bounds via threshold complexity	14
1.5 Conclusions: beyond VC dimension	16
1.6 Prior publication and collaboration	17
2 PRELIMINARIES	18
2.1 Learning theory framework	18
2.2 Notation	19
2.2.1 Pairs of classifiers	22
2.3 Probably-Approximately-Correct learning	24
2.4 Approximation error and estimation error	25
2.5 Weak learners and ensemble-based learning	26
2.5.1 Base classifiers	28
2.5.2 Boosting algorithms	29
2.5.2.1 Majority-of-three	29
2.5.2.2 Boost-by-majority	30
2.5.2.3 AdaBoost	31
2.5.3 Bagging	32
2.6 Vapnik-Chervonenkis dimension	33

3	EXTENSIONS TO MCDIARMID'S INEQUALITY WHEN DIFFERENCES ARE BOUNDED WITH HIGH PROBABILITY	37
3.1	Introduction	37
3.2	Preliminaries	43
3.2.1	Notions of difference-boundedness	43
3.2.2	Martingales	45
3.2.3	The moment generating function	47
3.2.4	Bernstein's Theorem	49
3.3	Strongly difference-bounded functions	51
3.4	Weakly difference-bounded functions	57
3.5	The case when $\lambda = 0$	64
3.6	Open questions	68
4	ALMOST-EVERYWHERE ALGORITHMIC STABILITY AND GENERALIZATION ERROR	70
4.1	Introduction	70
4.2	Definitions of uniform stability	73
4.3	Definitions of strong and weak stability	75
4.4	Training stability and CV stability	80
4.4.1	CV stability and weak error stability	82
4.5	Stability and generalization error	83
4.5.1	Mean generalization error is small	85
4.5.2	Training error is difference-bounded	87
4.5.3	True error is difference-bounded	90
4.5.4	Putting it all together	90
4.6	CV stability and learnability	94
4.7	VC dimension and stability	97
4.8	Examples of stable and unstable learners	102
4.8.1	$ \mathcal{H} = 2$	102
4.8.2	$ \mathcal{H} < \infty$	105
4.8.3	Hyperplanes and maximum margin	107
4.8.4	Learning finite languages	111
4.8.4.1	The noisy setting	113
4.8.5	Overtraining	115
4.9	Open questions	116
4.9.1	The "right" definition of stability	116
4.9.2	Leave-one-out error estimates	118
4.9.3	Stability, VC dimension, and examples	119

5	CONCENTRATION BOUNDS FOR GOOD-TURING ESTIMATORS	122
5.1	Introduction	122
5.2	First proof: independent bounded differences	124
5.3	Second proof: sum of squared ranges: M_0	127
5.4	Second proof: sum of squared ranges: M_k	131
5.5	Conclusions and open questions	138
6	THE INTERACTION OF STABILITY AND WEAKNESS IN ADABOOST . .	140
6.1	Introduction	140
6.2	Stability on weighted sets	143
6.3	Weakness	145
6.3.1	Weakness of classifiers	145
6.3.2	Weakness of learning algorithms	147
6.3.3	Implications of weakness	151
6.4	The algorithmic stability of AdaBoost	153
6.4.1	Constructing the final hypothesis in AdaBoost	153
6.4.2	Bounding the behavior of AdaBoost	155
6.4.3	AdaBoost is stability-preserving	166
6.5	Conclusions and open questions	168
7	CONSTRUCTING AND COMBINING DECORRELATED CLASSIFIERS . . .	170
7.1	Introduction	170
7.2	The optimal three-classifier ensemble	175
7.2.1	A constrained minimization formulation	175
7.2.2	Analysis	176
7.2.3	Experimental results: three-classifier ensembles	180
7.2.3.1	Artificial data	180
7.2.3.2	Phoneme database	182
7.3	Connections to boosting	182
7.3.1	A reconsideration of Schapire's algorithm	183
7.3.2	Is 1/2 the optimal weighting scheme?	185
7.3.3	Decorrelation in AdaBoost	189
7.4	PairTree	193
7.5	PairBoost	197
7.5.1	The VC dimension of pairs	200
7.6	Practical considerations	203
7.6.1	Working with training data	204
7.6.2	Finding pairs of decision stumps quickly	205
7.6.2.1	Comparing two coordinates in $O(m \log m)$ time . . .	206
7.6.2.2	Additional remarks and questions	208
7.6.3	Finding pairs of perceptrons	210

7.7	Experimental results: PairTree and PairBoost	211
7.7.1	The generalization error of tall trees	215
7.7.2	Truncating trees and appending stumps	215
7.7.3	AdaBoost and PairBoost	219
7.8	Conclusions and open questions	221
8	LOWER BOUNDS ON THE CONVERGENCE TIME OF BOOSTING ALGORITHMS	224
8.1	Introduction	224
8.1.1	Training error versus generalization error	226
8.2	Weak learners	227
8.2.1	Complementation	230
8.3	Known lower bounds	231
8.3.1	Boost-by-majority	231
8.3.2	AdaBoost	235
8.3.2.1	Upper bounds for AdaBoost	238
8.4	Lower bounds for AdaBoost	239
8.4.1	ξ -distributions	241
8.4.2	A geometric distribution	244
8.5	Dynamical systems	249
8.6	Separations between AdaBoost and PairTree	253
8.6.1	AdaBoost can be slower than PairTree	253
8.6.1.1	The binary addition function	255
8.6.2	PairTree can be as slow as AdaBoost	257
8.6.3	PairTree is less expressive than AdaBoost	260
8.7	Pair-tree complexity	261
8.8	Open questions	263
8.8.1	Lower bounds for AdaBoost	263
8.8.2	Separations between AdaBoost and PairTree	264
	REFERENCES	266

ABSTRACT

We explore two themes in formal learning theory. We begin with a detailed, general study of the relationship between the generalization error and stability of learning algorithms. We then examine ensemble-based learning from the points of view of stability, decorrelation, and threshold complexity.

A central problem of learning theory is bounding generalization error. Most such bounds have been obtained through uniform convergence, via some variant of VC dimension. These analyses focus on the space of classifiers available to a learning algorithm, rather than on the algorithm itself.

Bousquet and Elisseeff (2002) have shown, using McDiarmid’s method of independent bounded differences (1989), that algorithmic stability implies good bounds on generalization error. However, their definition of stability is too restrictive to be widely applicable.

We introduce the more general notion of training stability. We show that training stability implies good bounds on generalization error even when the learner has infinite VC dimension. Our proof requires a result of independent interest: we generalize McDiarmid’s theorem to the case when differences are bounded with high probability.

In the Probably-Approximately-Correct setting of Valiant (1984), training stability is necessary and sufficient for good error performance and serves as a distribution-dependent analog of VC dimension. This enables us to emphasize the learning algorithm instead of the classifier space.

We next consider algorithms (e.g., boosting) which construct ensembles of weak classifiers. We show that AdaBoost (Freund and Schapire, 1997), a widely-used boosting algorithm, is stability-preserving.

We demonstrate that some boosting algorithms implicitly construct classifiers which are decorrelated (i.e., rarely jointly wrong). We present two new

algorithms which build ensembles of explicitly decorrelated classifier pairs. Experiments indicate that these new algorithms are sometimes competitive with AdaBoost and with bagging (Breiman, 1996). The experiments employ a fast new algorithm for finding decorrelated pairs of decision stumps.

Lastly, we use threshold complexity to show that, under certain conditions, AdaBoost takes a long time to converge. In some situations, our pair-based algorithms are exponentially faster than AdaBoost. We analyze boosting as a dynamical system over the space of sequences of weights.

CHAPTER 1

INTRODUCTION

1.1 An algorithmic theory of learning

How do people and machines learn? What makes something learnable or unlearnable? How much time, or memory, does it take to learn a particular concept? These questions have been discussed by philosophers, psychologists, linguists, neurobiologists, mathematicians, cognitive scientists, statisticians, and computer scientists.

A learning algorithm is a process which extrapolates a general principle, or *hypothesis*, from finite data. Two central problems of learning theory are:

1. How can we analyze the performance of learning algorithms?
2. How should we design new algorithms which will have good performance?

In this thesis, we give new answers to both of these questions:

1. We introduce new notions of *algorithmic stability*. We show that stable learning algorithms construct hypotheses which generalize well to new data. We give the most complete argument to date for the viability of the stability framework as a new approach to analyzing learning algorithms.
2. We analyze *ensemble-based* learning algorithms, which construct an ensemble hypothesis by combining various weaker hypotheses. We define two new ensemble-based algorithms, PairTree and PairBoost, which build collections of *decorrelated pairs* of hypotheses. We provide a theoretical and experimental analysis of these new algorithms.

For the past thirty years, the standard tool for proving bounds on the performance of learning algorithms has been uniform convergence via Vapnik-Chervonenkis (VC) dimension. The VC theoretic approach is *representational*: the emphasis is not on the learning algorithm, but on the space of hypotheses available to it. Such an approach is necessarily limited. Computer scientists do not often develop new spaces of functions; the focus of ongoing research tends to be developing new ways of traversing existing spaces and of selecting hypotheses. For this reason, our goal is to develop a non-representational, *algorithmic* theory of learning.

1.1.1 Overview

We discuss different notions of the performance of a learning algorithm in Section 1.2. We also give a brief description of the strengths and weaknesses of VC theory. We define our terms more precisely in Chapter 2.

In Section 1.3, we discuss our first major theme: algorithmic stability. We give various definitions of stability, and we explain the relationship between stability and generalization error. This relationship is the subject of Chapters 3 through 6.

In Section 1.4, we discuss our second theme: ensemble-based learning. Our primary focus is on boosting algorithms. We examine boosting through the lenses of stability, decorrelation, and threshold complexity in Chapters 6, 7, and 8. The goal of each of these analyses is an understanding of ensemble-based learning which goes beyond the representational power of the space of possible ensembles.

We state some overall conclusions in Section 1.5. Finally, in Section 1.6, we discuss the prior publication of the work in this thesis, and thank our coauthors for their contributions.

1.2 Generalization error and VC dimension

We examine the problem of learning from a computer science perspective. We work in a specific setting: learning a binary classification rule by inference. Some entity generates a collection of independent random instances, each of which comes with a binary label. These labels may be generated deterministically or probabilistically from the instances.

A learning algorithm is a process which, from this finite training set S of examples, constructs a classifier, or hypothesis, f_S : a general rule for determining labels from instances. The learning algorithm does well if this rule is a good predictor of the labels of new randomly-generated instances.

Binary classification includes a wide variety of problems in human and machine learning, such as:

- Data mining. A computer program is given a large collection of records, and constructs a rule for predicting one attribute. The data could be medical information, where the program must predict whether a person is at risk for a particular disease. Or the data could be a list of credit card purchases, where the program must determine if there is suspicious activity on the card. As data sets grow larger and larger, this problem is becoming more and more difficult.
- Handwriting and speech recognition. A computer must determine what letter or number is represented by a written symbol, or what word is represented by a sound wave. This is technically a multiclass problem, but we can build a classifier for any discrete collection of classes from a decision tree of binary classifiers.

We allow our binary classifiers to make confidence-rated predictions. This gives us another way to build multiclass classifiers: we apply a separate binary classifier for each class, and we see which classifier reports a positive answer with the highest confidence.

- Vision. Given an image, can we determine if it contains a face? Can we determine whose face it is?
- Language learning. How do children, simply by listening to their parents, learn how to determine whether or not a sentence is valid? How do bilingual children learn two languages simultaneously without confusing them?

The measure of the performance of a learning algorithm is its *true error rate*: the expected success rate of the classifier f_S on new randomly-generated examples. We use D to denote the random distribution on the space of labeled examples, and we write $\text{Err}_D(f_S)$ for the true error rate of f_S . If $c(f_S, z)$ denotes the error of f_S on example z , then $\text{Err}_D(f_S) = \mathbf{E}_z(c(f_S, z))$.

True error is the primary measure of the performance of a learning algorithm. An algorithm is good if, as the size of the training set grows, the probability that $\text{Err}_D(f_S)$ is far from optimal goes to zero. (In some settings, the “optimal” error rate is the Bayes error rate; in others, there may be some natural limitations on the space of possible classifiers which make the optimal error rate worse.)

It is often easier to analyze true error by dividing it into two components. The first is *training error*, which measures the performance of f_S on the training set S , and which we denote by $\text{Err}_S(f_S)$. If we write $S = (z_1, \dots, z_m)$, then $\text{Err}_S(f_S) = \frac{1}{m} \sum_{i=1}^m c(f_S, z_i)$.

The second component is *generalization error*, which is the gap

$$\text{gen}(S) = \text{Err}_D(f_S) - \text{Err}_S(f_S)$$

between training error and true error. In many contexts, there is a trade-off between these two components; the more complicated we allow our classifiers to become, the lower the training error and the higher the generalization error.

One extreme point for this trade-off is the constant algorithm, which ignores its training data and outputs a fixed classifier h . There is no reason to believe that this classifier will be correlated to the data, so $\text{Err}_S(h)$ may be large. How-

ever, by the law of large numbers, $\text{Err}_S(h)$ is likely to be a good approximation to the true error $\text{Err}_D(h)$.

At the other extreme, if a learning algorithm generates complicated classifiers, it may overfit to the training set. For example, consider the (nearest neighbor) algorithm which simply memorizes S . Given an instance, the algorithm looks it up in its memory, returning the correct label if it has seen the instance and guessing randomly otherwise. This algorithm always has $\text{Err}_S(f_S) = 0$. However, its generalization error is high; performance on the training set is not a good predictor of overall performance.

A key issue in learning theory is how to manage this trade-off between the simplicity and complexity of the space of classifiers. Grenander [Gre52] refers to this problem as the “uncertainty principle.” We need a theory which will tell us how complex we should make our space of classifiers to balance training error and generalization error.

One important measure of the complexity of a space of classifiers is its *Vapnik-Chervonenkis (VC) dimension* [VC71] (see Section 2.6 for a precise definition). The uniform convergence theorem [VC71] (see Theorem 2.26) says that, if a space \mathcal{H} of classifiers has finite VC dimension, then, as the size of the training set S approaches infinity, $\text{Err}_S(h) \rightarrow \text{Err}_D(h)$ uniformly for all $h \in \mathcal{H}$.

Vapnik and Chervonenkis [VC71] show that, if a space \mathcal{H} has finite VC dimension, then empirical risk minimization over that space (i.e., an algorithm which returns $h \in \mathcal{H}$ minimizing training error) is a good learning algorithm: $\text{Err}_D(f_S) \rightarrow \text{Err}_D(h_*)$, where h_* is the optimal classifier in \mathcal{H} . Furthermore, they prove a Chernoff-like concentration bound for $\text{Err}_D(f_S)$ about its mean.

In the closely-related Probably-Approximately-Correct (PAC) setting of Valiant [Val84] (see Section 2.3), the finiteness of the VC dimension of \mathcal{H} is necessary and sufficient for good bounds on empirical risk minimization over \mathcal{H} . A proof of this statement appears in Blumer, et al. [BEHW89].

However, there are many learning algorithms which do not perform straightforward empirical risk minimization. For example:

- Regularization, structural risk minimization, Occam learning, or minimum description length learning. These algorithms work over a space which may be of infinite VC dimension, and minimize some combination of the training error of h and the complexity of h . One can force such an algorithm into the VC framework: first argue that the output has some maximum complexity, then prove that the space of possible outputs behaves as if it has finite VC dimension, and finally conclude that uniform convergence applies. But VC analysis may not be the right tool to use.
- Gradient descent, multi-layer perceptrons trained via backpropagation, or boosting (see Section 2.5.2). These algorithms often work over a space whose VC dimension is finite, and one can prove bounds on generalization error using uniform convergence. However, these bounds tend to be worse than the observed behavior of the algorithms. Since we are not performing empirical risk minimization, VC analysis is not sharp enough; we need a theory which takes into account the way in which algorithms search the space of available classifiers.

Our goal is to prove bounds on the performance of learning algorithms without referring to VC dimension. We prove these bounds using stability, which is a property not of a space of classifiers but of a learning algorithm.

1.3 Algorithmic stability

Our first major theme is *algorithmic stability*. An algorithm is stable if a small change in the training set effects at most a small change in the output of the algorithm. This notion was first introduced by Devroye and Wagner [DW79]. We cannot hope to deduce bounds on true error through stability; the constant algorithm of Section 1.2 is stable, but performs poorly. We can, however, show that stable algorithms have low generalization error.

Recently, Bousquet and Elisseeff [BE02] introduced the notion of *uniform hypothesis stability*: for any training set S , changing any example in S to any other

possible example effects at most a small change in f_S . (The change in f_S is measured in the L_∞ norm: it is the maximum change in $c(f_S, z)$ over all examples z .) As an example, they prove that regularization is uniformly hypothesis stable.

Bousquet and Elisseeff show that uniform hypothesis stability implies that the mean generalization error approaches zero. They also prove Chernoff-like concentration bounds: $\text{gen}(S)$ is close to its mean with exponentially high probability.

However, uniform hypothesis stability is too restrictive a definition to be of general use:

- Suppose that f_S is always a binary classifier; for any z , we have $c(f_S, z) = 0$ or 1. So there is no such thing as a “small change in f_S ,” if $c(f_S, z)$ changes for any z , then the L_∞ change in f_S is 1. Hence, in this setting, only the constant algorithm is uniformly hypothesis stable.
- Consider an algorithm which minimizes empirical risk over a finite set \mathcal{H} of classifiers. If there is an optimal classifier $h_* \in \mathcal{H}$, then, as the size of S approaches infinity, $f_S = h_*$ for most training sets S . So, for most training sets, the algorithm is stable. But there are bad, unlikely training sets for which the algorithm is not stable.
- Consider an algorithm which learns a one-dimensional threshold: the input is a collection of points x_i , each labeled 1 if $x_i \geq \theta$ and -1 if $x_i < \theta$. The algorithm guesses θ to be x_{\min} , the minimum x_i labeled 1. We are performing empirical risk minimization over a space of VC dimension 1.

This algorithm will almost never be stable. Unless x_{\min} happens to equal θ (which happens with probability zero), there is always some way to change an element of the training set and change the output f_S : simply replace some element with a number between x_{\min} and θ . So, for almost any S , we can come up with a change that effects a large change in f_S . However, for most random changes, the new point will not lie in $[\theta, x_{\min}]$, and there will be no change in f_S .

Kearns and Ron [KR99] consider the weaker notion of *error stability*: an algorithm is error stable if a small change in S usually effects a small change in $\text{Err}_D(f_S)$. Each of the algorithms described above is error stable. However, so is the overfitting algorithm of Section 1.2. As we discuss in Section 4.8.5, we cannot use error stability to bound generalization error without making some additional assumption, such as finite VC dimension.

In Section 4.3, we introduce the notion of *strong hypothesis stability*: For most training sets S , the algorithm is stable at S , which means that changing any example in S to any other possible example effects at most a small change in $c(f_S, z)$ (for the worst-case z). We prove in Section 4.5 that strong hypothesis stability implies Chernoff-like bounds on generalization error.

Strong hypothesis stability allows for the possibility of unstable but unlikely training sets. We show in Section 4.8.2 that empirical risk minimization over a finite set \mathcal{H} (when there is a unique optimal $h_* \in \mathcal{H}$) is strongly hypothesis stable. However, the above example with VC dimension 1 is not strongly hypothesis stable; the algorithm is unstable at almost every training set S .

In Section 4.3, we also introduce *weak hypothesis stability*: If we randomly select a training set S , and randomly change one example in S to another randomly-generated example, then, most of the time, the resulting change in $c(f_S, z)$ is small (for the worst-case z). We prove in Section 4.5 that weak hypothesis stability implies Chernoff-like bounds on generalization error.

Our example with VC dimension 1 is weakly hypothesis stable. However, as we will see in Section 4.8, it is possible to construct learners with finite VC dimension which are not weakly hypothesis stable. We thus need an even more general notion of stability.

In Section 4.4, we introduce cross-validation (CV) stability: If we randomly select a training set S , and randomly change one example in S to another randomly-generated example z , then, most of the time, the change to $c(f_S, z)$ is small. We show in Section 4.5 that, for empirical risk minimization, CV stability also implies Chernoff-like bounds on generalization error.

Also in Section 4.4, we introduce overlap stability: If we randomly select a training set S , and randomly change one example in S to another randomly-generated example z , then, most of the time, the change in the average error rate on the rest of S is small. We show in Section 4.5 that overlap stability, together with CV stability, implies Chernoff-like bounds on generalization error.

We use the term training stability to refer to the combination of CV stability and overlap stability. This is the weakest notion of stability for which we prove good bounds on generalization error.

In Section 4.8, we give several examples of algorithms which are stable and unstable in these various senses.

Note that our definitions of stability are “distribution-dependent,” meaning that the stability of an algorithm depends on the distribution on examples. In contrast, VC dimension is a distribution-free notion. We argue that CV stability can be viewed as a distribution-dependent analog to VC dimension; in the PAC setting, CV stability is necessary and sufficient for good bounds on generalization error.

This dependence on distribution may seem to be a limitation on stability, since the bounds obtained through VC theory are stronger. Training stability does not give us uniform convergence. However, we are able to use training stability to analyze hypothesis classes for which there is no uniform convergence. We can prove distribution-dependent bounds on generalization error for algorithms for which no distribution-free bounds can be shown. As an example, we discuss in what sense finite language learning is training stable in Section 4.8.4.

1.3.1 Independent bounded differences

Our proof that training stability gives good bounds on generalization error follows the same general approach as Bousquet and Elisseeff’s proof for uniform hypothesis stability [BE02]. However, when we try to generalize their argument, several technical problems arise.

Bousquet and Elisseeff use McDiarmid’s “method of independent bounded differences” [McD89]. This method is a generalization of Chernoff’s inequality. We say that a multivariate, real-valued function is *uniformly difference-bounded* if any one-coordinate change to the input effects a small change in the output. McDiarmid [McD89, McD98] proves a large-deviation inequality: any uniformly difference-bounded function is tightly concentrated about its mean.

McDiarmid’s theorem has been widely used in combinatorics and in learning theory. Bousquet and Elisseeff [BE02] prove that, if a learning algorithm is uniformly hypothesis stable, then $\text{gen}(S)$ is uniformly difference-bounded.

If we make the weaker assumption that an algorithm is strongly hypothesis stable, we conclude that $\text{gen}(S)$ is *strongly difference-bounded*: for most inputs, any one-coordinate change effects a small change in the output. If we assume weak hypothesis stability or training stability, we conclude that $\text{gen}(S)$ is *weakly difference-bounded*: most of the time, a random one-coordinate change to a random input effects a small change in the output.

In Chapter 3, we generalize McDiarmid’s theorem. We prove that any strongly or weakly difference-bounded random variable is tightly concentrated about its mean. This is the key step in our proof that stability implies good bounds on generalization error.

The generalized versions of McDiarmid’s theorem are of independent interest. As an example, we apply our version of McDiarmid’s theorem to another problem in inferential learning: estimating population statistics.

Given a collection of samples drawn from an unknown distribution, how can we estimate the probability of obtaining a particular item from a new random draw? What is the probability that a new sample will be one we have not yet seen? McAllester and Schapire [MS00, MS01] give concentration bounds on these random variables using McDiarmid’s method of independent bounded differences. Their results imply good bounds on the accuracy of the Good-Turing estimators [Goo53] for these random variables.

In Chapter 5, we first use our extension to McDiarmid’s theorem to give a simpler proof of a slightly-worse concentration bound for M_k , the total probabil-

ity associated to items which occur exactly k times in the sample. We then give a second proof, using another theorem of McDiarmid [McD98]. This second proof is longer, but improves the best previously-known bounds for $k > 0$.

1.4 Ensemble-based learning

Our second major theme is ensemble-based learning: constructing a more complex classifier out of an ensemble of simpler classifiers. An ensemble-based protocol is a means of amplifying the accuracy of a learning algorithm: we employ a weak learning algorithm as a subroutine, and each call to the weak learner returns a classifier. We then combine these classifiers into an ensemble, which should have better performance than the individual constituent classifiers.

The key questions in ensemble-based learning are:

- How do we make calls to the weaker algorithm? Do we give it access to random examples, or do we modify or weight the examples in some way?
- How do we construct the ensemble?
- How can we analyze the error rate of the strong learner given properties of the weak learner?

One mechanism for constructing ensembles is *boosting*. In 1990, Schapire [Sch90] first introduced boosting, and showed that, given a weak learner whose true error rate is bounded away from 50% (i.e., a weak learner slightly better than random guessing), one can construct a strong learner with any desired level of accuracy. We describe Schapire's algorithm in Section 2.5.2.1.

In 1995, Freund and Schapire [FS97] introduced AdaBoost, a widely-used boosting algorithm which constructs a linear combination of the classifiers returned by the weak learner. The original analysis of AdaBoost relies on VC theory: if the space of classifiers used by the weak learner has finite VC dimension, then the space available to AdaBoost (after any fixed number of rounds) also has finite VC dimension. We describe AdaBoost in detail in Section 2.5.2.3.

Many have observed that the actual performance of AdaBoost is better than the VC theoretic analysis would suggest, and have searched for alternative explanations for AdaBoost's success. Such analyses, including the theory of margins [SFBL98], have generally relied on extensions to VC theory.

The problem with the VC analysis is that AdaBoost does not perform empirical risk minimization over the space of linear combinations of base classifiers. (Mason, et al. [MBBF99] have shown that AdaBoost can be viewed as performing gradient descent.) We attempt to analyze AdaBoost by considering in detail exactly how AdaBoost traverses the space of linear combinations.

1.4.1 The algorithmic stability of AdaBoost

In Chapter 6, we combine our two main themes. We prove that AdaBoost is *stability-preserving* in the following sense: if a weak learner is uniformly hypothesis stable, then AdaBoost applied to that weak learner is strongly hypothesis stable.

We also require that the weak learner be *weak*: roughly speaking, we require that $\text{Err}_S(f_S)$ be bounded away from zero as the size of S approaches infinity. In particular, if the Bayes error rate is nonzero, then any algorithm is weak. We discuss several definitions of weakness in Section 6.3. Our analysis is unusual; many analyses require an assumption about the *strength* of the weak learner, rather than about its weakness.

Our original goal was to give generalization error bounds for AdaBoost using stability which are tighter than those obtained using VC theory. However, the bounds we prove in Chapter 6 are significantly worse than the VC theoretic bounds. One could argue that our approach suggests that AdaBoost does not really preserve stability.

1.4.2 Decorrelation

In Chapter 7, we approach ensemble-based learning from another perspective: decorrelation. Two binary classifiers are *decorrelated* if the probability that both are wrong is smaller than the product of their individual error rates. We show in Section 7.3 that Schapire’s original boosting algorithm [Sch90] and AdaBoost [FS97] implicitly construct decorrelated pairs of classifiers.

As we stated earlier, one central problem of learning theory is how we can construct new learning algorithms. We approach this problem using decorrelation: how can we build an ensemble from decorrelated pairs? We give two answers to this question. In Section 7.7, we describe experiments comparing our new algorithms to AdaBoost and to *bagging*, an ensemble-based algorithm developed by Breiman in 1996 [Bre96a].

Our first pair-based algorithm is called PairTree (see Section 7.4). It is often possible to construct a pair of classifiers which are never jointly wrong on the training set. We can build a sequence (or “tree”) of such pairs; the output of the ensemble on a point x is the output of the first pair in the sequence which agree on x . Since no pair can agree on the wrong answer, this approach reduces training error to zero.

Our experiments indicate that PairTree is erratic; it performs much worse than standard algorithms on most data sets we considered. On some data sets, PairTree gets stuck—it is not possible to find pairs which are never jointly wrong. On other data sets, PairTree does too good a job of fitting to the data, constructing a tall tree of pairs. We observe that there is a direct correlation between the height of the tree and its generalization error.

Our second pair-based algorithm is called PairBoost (see Section 7.5). We construct a linear combination of pairs, using the same approach as in AdaBoost. We view a pair as a classifier which outputs a binary label (when the two classifiers in the pair agree) or “I don’t know” (when the two classifiers disagree), and we use a technique of Schapire and Singer [SS99] for applying AdaBoost to three-valued classifiers.

On the data sets we consider, PairBoost is competitive with AdaBoost and with bagging, but slightly worse than AdaBoost. (PairBoost is also slower than AdaBoost.) The gap between PairBoost and AdaBoost is larger than VC theory predicts; these experiments thus provide further evidence for one of our main points, that VC theory is not the right tool for analyzing the generalization error of boosting.

One interpretation of the gap between PairBoost and AdaBoost is that boosting works better with simple base classifiers than with complex base classifiers. It may therefore be profitable to combine AdaBoost with a regularized learner. This observation further justifies our work described in Section 1.4.1: regularization is uniformly hypothesis stable, and in Chapter 6 we consider AdaBoost applied to a uniformly hypothesis stable learner.

It is worth noting that finding pairs of decorrelated classifiers is potentially very slow. If our base classifier space consists of decision stumps (see Section 2.5.1) over \mathbb{R}^k , and we have m training points, then a naïve algorithm for finding the optimal pair takes time $\Theta(k^2 m^2)$.

In Section 7.6, we give a fast algorithm which finds decorrelated pairs of decision stumps in time $O(k^2 m \log m)$. The algorithm minimizes individual error rate over all pairs which have joint error rate 0 (or, more generally, minimizes any linear combination of individual error rate and joint error rate). We also give a gradient descent algorithm for finding decorrelated pairs of perceptrons.

1.4.3 Lower bounds via threshold complexity

Up to now, we have focused on upper bounds for learning algorithms: guarantees that the training error or generalization error will be small. In Chapter 8, we turn to lower bounds—specifically, to guarantees that, under certain conditions, the training error of an algorithm will be large. We phrase these bounds in terms of convergence time: we argue that the algorithm must run for a long time to reduce training error to zero (i.e., to converge to the target function).

When Freund introduced the Boost-by-majority algorithm [Fre95] (see Section 2.5.2.2), he also proved a lower bound on convergence time. The proof relies on the computational complexity of threshold gates. Freund first notes that his algorithm outputs an unweighted linear threshold of base classifiers; i.e., after T rounds, the output is 1 if $\sum_{t=1}^T h_t > \theta$ for some threshold θ . He then argues that, for some target functions and some weak learners, the target function cannot be written in this form unless T is large. He concludes that it takes many rounds for the output of the boosting algorithm to be equal to the target function. We describe this reasoning in detail in Section 8.3.1.

Freund’s argument does not apply directly to AdaBoost. AdaBoost builds a *weighted* linear threshold of base classifiers. In many cases, the target function has a succinct representation as a weighted threshold, but AdaBoost still takes a long time to converge. This is another manifestation of our overall goal: the threshold analysis applies to the space of classifiers available to AdaBoost, but we want an analysis which uses properties of the learning algorithm itself.

In Section 8.4, we give new lower bounds for AdaBoost. We provide examples where AdaBoost requires a long time to converge to the solution. We show that, for particular choices of target function, weak learner, and distribution on the space of instances, AdaBoost constructs an unweighted linear threshold. In these situations, Freund’s argument applies.

In Section 8.6.1, we show an exponential separation between PairTree and AdaBoost: in some situations, AdaBoost takes T rounds to converge, while PairTree requires only $O(\log T)$ rounds. This proves that the lower bounds are not based on the inherent unlearnability of the target function, but on the details of how AdaBoost attempts to learn the target. In Section 8.6.2, we show a situation where we can prove strong lower bounds for both algorithms.

In Section 8.7, we frame our results as complexity-theoretic statements by comparing the optimal representation of a function as a sequence of pairs and as an unweighted linear combination of base classifiers. There are cases where the sequence of pairs can be exponentially smaller than the unweighted linear combination, and cases where both representations must be large. There are also

simple unweighted linear combinations which cannot be written as a sequence of pairs.

Several authors have pointed out that AdaBoost is a dynamical system in the space of linear combinations of classifiers. One can ask whether AdaBoost approaches a fixed point (i.e., approaches some limiting function) or exhibits some other behavior.

In Section 8.5, we describe AdaBoost as a dynamical system in the space of sequences of weights on the training set. In this space there is not generally a fixed point, but there may still be periodic behavior, which corresponds to a fixed point in the space of linear combinations of classifiers. We present a theoretical and experimental analysis of this dynamical system in a limited setting (in which we can consider the space of sorted sequences of weights). We give new evidence for the view that the dynamical systems approach is a useful way of analyzing AdaBoost.

1.5 Conclusions: beyond VC dimension

The overall goal of this work is to analyze complex learning algorithms without using uniform convergence or VC theoretic techniques. If we want distribution-free convergence bounds on empirical risk minimization, then VC theory tells a complete story. But if we want distribution-dependent bounds, or if we want to consider other types of algorithms, we need to go beyond VC dimension.

We show that various algorithms are almost-everywhere stable, and that our notions of stability are sufficient to obtain good bounds on generalization error. One open question is whether we can use CV stability to give generalization error bounds for empirical risk minimization over a space of finite VC dimension; this would enable us to fold VC theory into stability theory.

We discuss ensemble-based learning from the points of view of stability, decorrelation, and threshold complexity. In all cases, we focus not on a space of

functions, but on a learning algorithm itself. We believe that this approach is an important tool in the study of inferential learning.

Of course, our work leads to many open questions, both large and small. We close each chapter with a list of questions for future research.

1.6 Prior publication and collaboration

Chapter 3 appeared as a University of Chicago technical report [Kut02b]. A preliminary version was part of a 2001 technical report [KN01] with Partha Niyogi.

Chapter 4 is joint work with Partha Niyogi, and appeared as a University of Chicago technical report [KN02]. A preliminary version was part of a 2001 technical report [KN01].

Chapter 5 will appear as a University of Chicago technical report [Kut02a].

Chapter 6 is joint work with Partha Niyogi. It appeared as a 2001 University of Chicago technical report [KN01], which also contains preliminary versions of Chapters 3 and 4.

Chapter 7 is joint work with Partha Niyogi and Olivier Siohan, and will appear as a University of Chicago technical report [KNS02]. The work is an extension of earlier work by Partha Niyogi, Jean-Benoit Pierrot, and Olivier Siohan [NPS00], which consists roughly of Sections 7.1 through 7.4.

Chapter 8 will appear as a University of Chicago technical report [Kut02c].

Journal versions of all six of these chapters are in preparation.