LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TR-535

ON THE SAMPLE COMPLEXITY OF PAC-LEARNING USING RANDOM AND CHOSEN EXAMPLES

Bronwyn Bonnie Eisenberg

March 1992

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

On the Sample Complexity of Pac-learning using Random and Chosen Examples

by

Bronwyn Bonnie Eisenberg

B.A., English
Princeton University
(1981)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology
May 1991

Signature of Author _____
Department of Electrical Engineering and Computer Science
March 1, 1992

Certified by _____
Ronald L. Rivest
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Abstract

Two protocols used for learning under the pac-learning model introduced by Valiant are learning from random examples and learning from membership queries. Membership queries have also been used to efficiently and exactly learn a concept class $\mathcal{C}$ that is *too difficult* to pac-learn using random examples. We ask whether using membership queries—in conjunction with or instead of random examples—can serve a new purpose: helping to reduce the total number of examples needed to pac-learn a concept class $\mathcal{C}$ already known to be pac-learnable using just random examples. We focus on concept classes that are *dense in themselves*, such as half-spaces of $\mathbf{R}^n$, rectangles in the plane, and the class $\mathcal{I} = \{[0, a] : 0 \leq a < 1\}$ of initial segments of $[0, 1]$.

The main results of this thesis are:

1. Adding the option of using membership queries cannot significantly reduce the total number of examples needed to pac-learn a dense-in-itself concept class $\mathcal{C}$; $\Omega((1/\epsilon)\ln(1/\delta))$ random examples are still required to pac-learn $\mathcal{C}$, where ϵ and δ are the usual pac-learning parameters. Interestingly, this bound holds for *any* unknown probability distribution, unlike the "standard" proof (due to Ehrenfeucht, Haussler, Kearns, and Valiant [12]), which holds only for a particular distribution constructed by an adversary.

2. Even if the unknown probability distribution is known to be "smooth", at least $(1/4\epsilon)\ln(1/\delta)$ examples are required to pac-learn $\mathcal{C}$ from random examples (only).

3. For the special case of learning half-spaces of an n-dimensional unit simplex, if the probability distribution is smooth and examples can be chosen as well as drawn randomly, then the number of examples required to pac-learn decreases significantly, to $n^2(\lg(s/\epsilon) + 4)$, and the computational complexity is $O(n^7 s/\epsilon)$.

Portions of this thesis are joint work with Ron Rivest.

Thesis Supervisor: Ronald L. Rivest

Title: Professor of Electrical Engineering and Computer Science

Keywords: Pac-learning, Queries, Drawing examples, Choosing examples

Contents

1 Introduction 4

1.1 Overview 4

1.2 Standard Machine Learning Definitions 5

1.3 Pac-learning Definition 8

1.3.1 Pac-learning error rate 8

1.3.2 Pac-learning complexity 8

1.3.3 Protocols for obtaining examples 10

1.4 New Definitions 10

1.4.1 Dense-in-itself concept classes 10

1.4.2 *Smooth* probability distributions 13

1.5 Pac-learning Background 13

1.6 The Results of this Thesis 15

2 A Lower Bound When Both Drawing and Choosing Examples **18**

2.1 A Lower Bound 18

2.2 Comparison of the Lower Bound to Previous Results 21

3 Smoothness Guarantees **23**

3.1 A Lower Bound When Drawing Examples Only 23

3.2 Upper and Lower Bounds When Both Drawing and Choosing Examples . . 25

3.2.1 Using the binary-search approach in one dimension 25

3.2.2 Generalizing the binary-search approach to n dimensions 27

4 Conclusions and Open Problems 24

5 Acknowledgments 25

Chapter 1

Introduction

1.1 Overview

In 1984, Valiant published *A Theory of the Learnable* [18]. His paper introduced a model of learning that has come to be known as *probably approximately correct* learning, or pac-learning for short. As proposed by Valiant, a good learning algorithm should, after viewing a certain number of labeled examples, output a hypothesis that with *high* probability classifies *almost* all unseen examples correctly. Valiant's model has been widely embraced, and many interesting classes have been shown to be efficiently learnable under the pac-model. (The notion of efficiency is addressed in more detail later).

In spite of the fact that the pac-model leads to efficient learning algorithms, however, the most traditional implementation of the model imposes restrictions on the way learning can be done, and these restrictions potentially drive up the number of labeled examples, in other words the amount of information, the learning algorithm needs to obtain in order to compute a hypothesis. This thesis asks the following question: can altering several of the pac-learning restrictions reduce the number of examples the learning algorithm, or equivalently the learner, needs to obtain in order to compute a hypothesis?

In the standard pac-model, the learner has no choice over what examples are obtained. Additionally, the learner knows nothing about the probability distribution under which the learning is learning: in particular, any probability distribution is possible. In Chapter 2 we consider what additional efficiency can be gained in Valiant's model if the learner has some control over what examples it receives. We prove a negative result: in spite of having some choice over what examples are obtained, the number of examples the learner must obtain in order to compute a hypothesis is within a constant factor of the number of examples the learner must obtain in the standard pac-model.

In Chapter 3, we ask whether there is an increase in efficiency if the learner knows ahead of time that certain probability distributions (for instance, extremely skewed distributions) are disallowed. Here we show a positive result: if the learner knows that certain probability

distributions are not possible, then giving the learner some control over what examples it may obtain *can* lead to a substantial decrease in the number of examples needed in order to compute a hypothesis.

In Chapter 4, we state conclusions and open problems.

We continue this chapter by presenting some standard machine learning definitions, an introduction to pac-learning, and some new definitions. In light of our definitions and the introduction to pac-learning, we then discuss the learning models and main results of this thesis.

1.2 Standard Machine Learning Definitions

We consider the problem of learning an unknown concept, called the target concept, from examples. The examples are drawn from a space X, called the *sample space*. For instance, we may have $X = \{0,1\}^n$ or $X = \mathbf{R}^n$. In this thesis we focus on continuous sample spaces, such as $\mathbf{R}^n$. We use the terms *learner* and *learning algorithm* interchangeably.

A *concept* c is a subset of the sample space X. For $x \in X$, we say that $x \in c$ is a *positive* instance (or positive example) of a concept c, and that $x \notin c$ is a *negative* instance (or negative example) of a concept c. We also use functional notation, writing $c(x) = 1$ for positive examples of a concept c and $c(x) = 0$ for negative examples; we call $c(x)$ the *label* or *classification* of x (by c). The unknown *target concept* is denoted c_*. A hypothesis concept is denoted $\hat{c}$.

For instance, if $X = \mathbf{R}^n$, an example of a concept is a half-space defined by an $n-1$-dimensional hyperplane. Such a hyperplane splits $\mathbf{R}^n$ into two half-spaces. The subset of $\mathbf{R}^n$ consisting of the points on the hyperplane and all points on one side of the hyperplane (that is, one of the half-spaces) is considered to be the concept. Each of these points is a positive example of the concept, and each point on the other side of the hyperplane is a negative example of the concept.

A concept thus is a set of points. Observe that the set, which is potentially infinite, can, for concept classes of interest here, be represented by a finite set of parameters. For instance, a half-space is a set with an infinite number of points; it can be represented, however, by a hyperplane using a finite number of real-valued parameters.

We assume that there is a probability distribution D defined on the sample space X. There are then two ways to *obtain* examples. First, the learner can *draw* examples from X according to probability distribution D. Here we say that the learner is *learning from*

random examples. In this case, the learner has no choice over what examples it obtains. Second, the learner may also, in some cases, make a membership query arbitrarily in X, in which case the learner is also *learning from membership queries.* Making a membership query is also called *choosing* an example. With a membership query, the learner can ask about any point x, even if x is in a set that D assigns zero probability.

We assume that the learner is told $c_*(x)$ for any example x obtained, so that the learner knows whether x is a positive example or a negative example of the target concept. We say that the learner *sees* examples in a set S if it either draws a random example from S or makes a membership query in S.

A concept class $\mathcal{C}$ is the set of all concepts of a particular type. For instance, the set of all half-spaces in $\mathbf{R}^n$ (each half-space defined by an $n-1$-dimensional hyperplane) is a concept class. Another concept class is the class $\mathcal{I} = \{[0, a] : 0 \leq a < 1\}$ of initial segments of $[0, 1]$. We assume that the learner knows a priori the particular concept class $\mathcal{C}$ of which the *target concept* c_* is a member. However, it is not a requirement that the hypothesis concept that the learner outputs come from this same concept class $\mathcal{C}$. In fact, our proofs do not depend on the assumption that the learner outputs some member of $\mathcal{C}$ as an approximation to c_*.

Given two concepts c and c', let $c \oplus c'$ denote the set $(c - c') \cup (c' - c)$ of points that are classified differently by c and c'. This set of points on which c and c' disagree is called the *symmetric difference* of c and c'. Given two concepts c and c', we are not interested in simply the *size* of the symmetric difference. Rather, we are interested in the *probability* of the symmetric difference.

More formally, we define the *distance* d_D between concepts c and c' (with respect to probability D) as

$$d_D(c, c') = D(c \oplus c') \ ,$$

the probability that $c(x) \neq c'(x)$ for a point x randomly drawn according to D. That is, $d_D(c, c')$ is the probability that c and c' disagree on the classification of a randomly drawn point. Another way to think about this is that $d_D(c, c')$ measures the probability associated with the symmetric difference of c and c'.

Now, given a target concept c_* and another concept c, we wish to measure the distance $d_D(c, c_*)$ between the target concept and c. In this case, distance represents the error (again in a probabilistic sense) of concept c with respect to target concept c_*. More formally, we define the *error rate* of a concept c (with respect to the distribution D and the target concept c_*) to be $d_D(c, c_*)$. Thus, the *error rate* of a concept c measures the probability associated with the symmetric difference of c_* and c. We remark that potentially the

symmetric difference of the target concept and c can be a very large set and yet the error rate $d_D(c, c_*)$ can be very small, provided that there is little or no probability associated with the symmetric difference of c_* and c.

Under some probability distributions, it is in fact possible that the symmetric difference of two concepts c and c' is non-empty, and yet has zero probability. In other words, although the two concepts do disagree on some set of points, there is no probability weight on these points. In this case, although the two concepts are not the same (that is, they do not classify all points the same), the error rate $d_D(c, c')$ equals zero. Thus we note that in general d_D defines only a bounded *pseudo*metric on $\mathcal{C}$, for any probability distribution D. For d_D to be a proper metric, it must be the case that for any incorrect concept c there is a positive probability $d_D(c, c')$ of drawing a random example that demonstrates that c is incorrect. If we were to consider only those probability distributions such that $c \neq c'$ implies that $d_D(c, c') > 0$, then d_D would be a bounded *metric* on $\mathcal{C}$. (In this thesis we do not state the conditions on measurability that may be required to make our claims go through. See, for example, Ben-David et al. [6] and Blumer et al. [9] for discussion.)

Often, the hypothesis space of a concept class is uncountably infinite. For instance, the concept class $\mathcal{I}$ has an infinite hypothesis space: there are infinitely many initial segments $[0, a]$ of $[0, 1]$. The notion of the VC dimension of a hypothesis space offers a way of measuring such an infinite space. More specifically, the VC dimension is a combinatorial parameter of the hypothesis space. Haussler gives a very clear definition of VC dimension [13]:

> "The VC dimension of a hypothesis space H, denoted VCdim(H), is defined to be the maximum number d of instances that can be labeled as positive and negative examples in all 2^d possible ways, such that each labeling is consistent with some hypothesis in H [10] [19]."

In other words, the VC dimension is equal to the largest number m of points, such that for *each* element of the power set of the m points, the following holds: there exists a hypothesis in the concept class, such that each of the points in this particular element of the power set is classified as positive, and the remaining points (from the original set of m points) which are not in this element of the power set are classified as negative. For example, the VC dimension of the class $\mathcal{I}$ is 1.

1.3 Pac-learning Definition

The core idea of pac-learning is that a learning algorithm should compute with high probability a hypothesis concept that is accurate to a given error rate. That is, the learner's goal in the pac-learning model is *not* to exactly identify the target concept, but rather to compute a hypothesis concept such that with high probability the hypothesis concept agrees with the target concept on almost all examples in the sample space. Thus we have Valiant's notion of *probably approximately correct* [13] [18].

In the case of "batch" learning (used in the models of this thesis), the learner first sees all examples, and then computes the hypothesis concept. In the case of "online" learning, the learner may recompute a hypothesis concept after each example seen. In either case, it is not sufficient for an algorithm simply to compute and output a hypothesis consistent or close to consistent with all examples seen thus far; the learner should have confidence that with high probability almost all *future* examples will be classified correctly.

1.3.1 Pac-learning error rate

A pac-learning algorithm is given as part of its input an error parameter ϵ, which specifies the maximum error rate acceptable (this relates to being *approximately* correct), and a confidence parameter δ, which specifies the degree of certainty that the hypothesis will classify all unseen examples to within the specified error rate ϵ (this relates to being *probably* approximately correct).

More formally, let $\hat{c}$ represent the hypothesis concept computed by a run of a pac-learning algorithm. To pac-learn successfully, the algorithm, given input parameters ϵ and δ, must, for all probability distributions D, with probability at least $1 - \delta$ compute a hypothesis $\hat{c}$ such that the error rate $d_D(c_*, \hat{c}) < \epsilon$:

$$\Pr(d_D(c_*, \hat{c}) < \epsilon) > 1 - \delta. \tag{1.1}$$

In Valiant's original definition, $d_D(c_*, \hat{c})$ may be less than or equal to ϵ. For the sake of simplicity, we assume that the error rate must be strictly less than ϵ.

We can now define computational complexity in pac-learning.

1.3.2 Pac-learning complexity

Valiant's goal was to encourage computer scientists to study complexity in learning theory via "precise computational models of the learning phenomenon"[18]. Valiant delineates

two components of such learning models, which describe frameworks for learning target concepts: (1) a protocol for receiving information about the target concept, and (2) an algorithm for computing a hypothesis concept based on this information.

We define *sample complexity* to be the number of examples the learner needs to see to compute a hypothesis concept. Then complexity in learning theory, as proposed by Valiant, is measured as a function both of sample complexity (related to (1) above), and of computational, or time, complexity, that is, how much computing a learning algorithm must do to output a good hypothesis concept (related to (2) above). The goal in pac-learning is to use only *polynomial* sample and time complexity to learn a target concept. This thesis focuses in particular on issues related to sample complexity.

For a pac-learning algorithm to have polynomial *sample* complexity, it must have sample complexity polynomial in $1/\epsilon$, $1/\delta$ and the length of the input. For a pac-learning algorithm to have polynomial *time* complexity, it must take time polynomial in the number of examples received to compute a good hypothesis.

In sum, in order for a pac-learning algorithm to be computationally efficient as defined by the pac-learning model, two conditions need to hold:

1. The number of examples obtained, that is, the sample complexity, should be polynomial in $1/\epsilon$, $1/\delta$, and the length of the input.
2. The time taken by the algorithm to compute a good hypothesis, that is, the computational complexity, should be polynomial in the number of examples received.

Henceforth, when we refer to pac-learning, we will be referring to computationally efficient pac-learning, unless otherwise noted.

We make an assumption that ϵ is sufficiently small. It is always true that

$$1/\epsilon \geq -1/\ln(1-\epsilon).$$

For $\epsilon < .39841$, however, it is also true that

$$1/(4\epsilon) \leq -1/\ln(1-2\epsilon).$$

We thus assume ϵ is less than .39841; this assumption allows us to express our non-asymptotic lower bounds in the more standard form as a function of $1/\epsilon$. For instance, in Theorem 2 we substitute the weaker lower bound $m > (1/4\epsilon)\ln(1/\delta)$ for the stronger $m > (-1/\ln(1-2\epsilon))\ln(1/\delta)$, where m is the number of examples obtained.

1.3.3 Protocols for obtaining examples

Given the pac-learning model, there are numerous ways in which the learner can obtain information about the target concept.

In the most traditional form of pac-learning (used, for example, by Blumer et al. [8] and Valiant [18]), the protocol for receiving information is for the learner to *draw* random examples according to a *fixed but unknown* probability distribution on the sample space. (By unknown probability distribution, we mean unknown to the learner.) When the learner requests a sample point, the learner receives back an example drawn according to this unknown probability distribution. The example is labeled as either a positive or negative example of the concept. Any probability distribution on the sample space is possible. Since the complexity bounds must hold no matter what the unknown probability distribution is, this type of learning is also called distribution-free learning.

A second protocol is to obtain information through queries. A model in which the learner asks for information is called learning using queries. Although touched upon by Valiant [18], learning using queries was left largely unnoticed until Angluin published *Types of Queries For Concept Learning* [1] in 1986. In her paper, Angluin defines a wide variety of protocols for receiving information about the target concept. The protocols all have in common the fact that they allow the learner more control over what examples are seen. For instance, the learner can *choose* a particular example and ask whether it is in the target concept; this is called making a *membership query*. In general, the goal in learning by query, in contrast to pac-learning, is to achieve exact learning, that is, to find the exact target concept.

This thesis looks at models in which information is obtained by both *drawing* and *choosing* examples.

1.4 New Definitions

1.4.1 Dense-in-itself concept classes

We are interested in the sample complexity necessary to pac-learn what we call a dense-in-itself concept class $\mathcal{C}$.

Definition 1 *A concept class $\mathcal{C}$ is called* dense in itself *if for all concepts $c \in \mathcal{C}$, for all $\gamma > 0$, and for all finite measures μ on X, there is another concept $c' \in \mathcal{C}$ (that is, $c' \neq c$) such that $\mu(c \oplus c') < \gamma$.*

A measure is similar to a probability distribution, in that it assigns weight to a sample space. However, in a probability distribution all the weight must sum to 1; in a finite measure, the weight can sum to any finite value.

The existence of dense-in-itself concept classes is easily proved using techniques by Ben-David, Benadek, and Mansour in *A Parametrization Scheme For Classifying Models of Learnability* [7]. They first define a notion $Cov_D(A, B)$, such that one set of concepts A covers another set of concepts B with respect to a given probability distribution D, if for every $\epsilon > 0$ and $b \in B$, there exists an element $a \in A$ such that $d_D(a, b) < \epsilon$. That is, for every $b \in B$, we can always find an $a \in A$ such that the probability of the symmetric difference of a and b is less than ϵ. Note that as they define it, a and b may be the same concept.

We modify their definition to handle the case when A and B are the same concept class. In particular, we adapt their notation, and say that a set C is dense in itself if $Cov'_{\forall D}(C, C)$, where $Cov'_{\forall D}(C, C)$ means that for every $\epsilon > 0$ and $c \in C$, there exists an element $c' \in C$, $c \neq c'$, such that $d_D(c, c') < \epsilon$. Observe here the extra specification that the two concepts c and c' cannot be the same. In sum, to make the analogy from their definition of $Cov_{\forall D}(A, B)$ to our definition $Cov'_{\forall D}(C, C)$ of dense-in-itself, we add two conditions: the concept classes A and B must be the same, and the two concepts a and b must be different.

Ben-David et al. also define the notion $Cov_S(A, B)$. By definition, $Cov_S(A, B)$ holds if "for every $b \in B$ there exists a sequence $\langle a_i : i \in N \rangle$ of elements of A such that $\lim_{i \to \infty} a_i = b$" [7]. That is, for every point x, there are only a finite number of concepts in the infinite a_i sequence that disagree with b on their classification of x. Observe that the notion $Cov_S(A, B)$ depends solely on the nature of the sets A and B and is independent of any probability distribution.

We define an analogous notion $Cov'_S(C, C)$, similar to $Cov_S(A, B)$, except again for the two conditions that the concept classes A and B must be the same, and that each of the concepts a_i must differ from b.

The notion $Cov'_S(C, C)$ applies to the concept classes that we consider. For instance, for the concept class of rectangles in the plane, we can think of a series of rectangles, each one containing the one before it, such that the series of rectangles converge to the last rectangle in the sequence, which is the target rectangle b. Similarly, we can think of a series of hyperplanes that are all parallel to and increasingly close to a target hyperplane (where close is measured by perpendicular distance to the hyperplane).

Ben-David et al. prove that $Cov_S(A, B)$ implies $Cov_{\forall D}(A, B)$. Their proof, essentially without modifications, also shows that $Cov'_S(C, C)$ implies $Cov'_{\forall D}(C, C)$. Thus since the

concept classes we consider have the property $Cov'_S(C,C)$, the concept classes we consider have the property $Cov'_{\forall D}(C,C)$. Recalling that if a concept class has the property $Cov'_{\forall D}(C,C)$, then the class is dense in itself, we have that the concept classes we consider are dense in themselves. Some examples of concept classes that are dense in themselves are half-spaces in n-dimensional space, rectangles in the plane, and the concept class $\mathcal{I}$.

Recall that, by definition, a dense-in-itself concept class is guaranteed to contain, for any finite measure μ and concept c, a concept $c' \in C$, $c \neq c'$, such that the measure μ of the symmetric difference of c and c' is arbitrarily small. Now the probability measure *error rate*, which is defined in Section 1.1, is a valid candidate for measure μ above, since error rate is a finite measure. Thus, substituting in the above definition error rate for μ and target concept c_* for c, we have: for any given dense-in-itself concept class C and arbitrary target concept c_*, we are guaranteed that there exists a concept c' with error rate $(d_D(c_*, c'))$ of arbitrarily small size. For example, if we are given a target concept c_* and wish to ensure that the error rate of c_* and some other concept is less than 10^{-6}, a dense-in-itself concept class guarantees us that there exists a concept c' such that $(d_D(c_*, c') < 10^{-6})$.

The following lemma provides a key property of a dense-in-itself concept class.

Lemma 1 *For every algorithm A that pac-learns a* dense-in-itself *concept class C using random examples and membership queries, and every ϵ, δ, target concept c_* in C, $\beta > 0$, and probability distribution D, there exists a concept c in $C - \{c_*\}$, such that*

$$\Pr(A(\epsilon, \delta) \textit{ sees examples in } c_* \oplus c) < \beta \ .$$

Proof: Let $S = c_* \oplus c$. Assume A runs under probability distribution D. We prove the lemma using Definition 1. Our strategy is to show that the *probability* measure

$$\Pr(A(\epsilon, \delta) \text{ sees examples in } c_* \oplus c)$$

is bounded above by the measure of the expected number of points that algorithm $A(\epsilon, \delta)$ sees in S. Then, since by definition of dense in itself we can make the measure of the expected number of points that algorithm $A(\epsilon, \delta)$ sees in S as small as we like, it follows that we can make $Pr(A(\epsilon, \delta)$ sees examples in $c_* \oplus c)$ as small as we like too.

We define $\mu(S)$ to be the expected number of points that algorithm $A(\epsilon, \delta)$ sees in S:

$$\mu(S) = \mathbf{E}(\text{number of points } A(\epsilon, \delta) \text{ sees in } S) \ . \tag{1.2}$$

Then μ is a measure defined on X. Let P_i equal the probability that $A(\epsilon, \delta)$ sees i examples in S. Then

$$\Pr(A(\epsilon, \delta) \text{ sees any examples in } S) \ = \ \sum_{i=1}^{\infty} P_i$$

$$\leq \sum_{i=1}^{\infty} iP_i$$
$$= \mu(S) .$$

Since by the definition of *dense in itself* we can find a concept c that allows us to make $\mu(S)$ as small as we like (where S varies according to which c is chosen), the lemma follows. ∎

1.4.2 *Smooth* probability distributions

In the original definition of the pac-learning model, all probability distributions are possible. One can speculate that the fact that learning must be done under such a model drives up sample complexity. For instance, the fact that the fixed but unknown probability distribution could be extremely skewed may force the algorithm to obtain far more examples than it might have to obtain if it had some information ahead of time about the probability distribution.

We therefore consider whether having some advance knowledge about the nature of the probability distribution might, in fact, lower the number of samples the algorithm needs to obtain. Thus we look at the sample complexity necessary to pac-learn a concept class $\mathcal{C}$ under a *smooth* probability distribution.

Definition 2 *A probability distribution defined on* $\mathbf{R}^n$ *is called* smooth *if there exists an* $s \geq 0$, *such that for all* $\alpha \geq 0$, *the probability associated with a region of volume* α *is* $\leq s\alpha$.

In other words, for every smooth probability distribution, there is an upper bound on the amount of probability that can be associated with any region, and this upper bound is in direct proportion to the size of the region. For instance, for the concept class $\mathcal{I}$, the value $s\alpha$ is an upper bound on the probability of a region of *length* α. Similarly, in n-dimensional space, the value $s\alpha$ is an upper bound on the probability of a region of *volume* α. Observe that under a smooth probability assumption, certain highly irregular distributions are not possible. For instance, there is a limit to how much probability weight can be found in a very small region, and there can be no probability weight assigned to a point.

1.5 Pac-learning Background

Some pac-learning history will help the reader view our results in light of past work.

We recall that pac-learning imposes two complexity requirements: (1) the sample complexity, which is the number of samples drawn, must be polynomial in the length of an

input instance and in the inverse of the error and confidence parameters, and (2) the computational complexity, which is the computational time to find a good hypothesis (with good defined using the ϵ and δ parameters), must be polynomial in the number of examples received.

The first classes shown to be pac-learnable under the distribution-free model were all classes of Boolean functions [18]. These classes—such as k-DNF and k-CNF—all have *finite* sample space, for example, $\{0,1\}^n$. In 1986, Blumer et al. extended the notion of what classes could be learned under the distribution-free pac-learning model to include classes in Euclidean n-dimensional space, such as half-spaces and hypercubes [8]. Significantly, concept classes in Euclidean n-dimensional space, in contrast to classes of Boolean functions, have *uncountably infinite* sample space: the sample space is R^n. (Observe, however, that a representation of the concept, for instance, a defining hyperplane, can be specified finitely with a finite number of real-valued parameters.) Blumer et al. gave upper and lower polynomial bounds on sample complexity for these classes, and described polynomial-time algorithms to learn certain Euclidean n-dimensional classes [8].

In fact, a wide variety of concept classes have been shown to be learnable with both polynomial sample and time complexity. Examples of such classes include monomials over n Boolean variables [18], and, for constant k, k-DNF [18], k-CNF [18], and k-decision-lists [15] [17].

To a large extent, however, the time and sample complexity requirements are separable. In 1986 Blumer et al. [8] showed that a concept class has polynomial sample complexity if the VC dimension of the family of the concept class grows polynomially, and that if the VC dimension of a family of concept classes grows faster than polynomially, then the concept class is not pac-learnable [8].

There have also been significant results in the area of computational complexity. Two of the results mentioned here are based on differing definitions of the pac-learning model: one in which the hypothesis concept must come from the same concept class as the target concept, and one in which the hypothesis concept may come from a different class than the target concept.

In 1986, Pitt and Valiant [16] showed that given a pac-learning model in which the output hypothesis concept must come from the same concept class as the target concept, the computational complexity to pac-learn certain concept classes is not polynomial. They proved that if finding a consistent hypothesis from the same class as the target concept class is NP-hard, then the given concept class is not pac-learnable unless $RP = NP$. For instance, they showed that since finding a consistent 2-term DNF hypothesis for the 2-term

DNF concept class is NP-hard, 2-term DNF is not pac-learnable, assuming $RP \neq NP$.

Kearns and Valiant proved computational complexity results for the pac-learning model in which the hypothesis concept may come from a different concept class than the target concept. They showed that even if the output hypothesis can come from a class outside the target concept class, the computational complexity to pac-learn certain concept classes is not polynomial. This result depends on certain complexity-theoretic cryptographic assumptions. Thus Kearns and Valiant showed that certain classes have polynomial sample complexity but not polynomial time complexity. Examples of such classes are Boolean formulas and finite automata [14].

Even when a class is pac-learnable, however, the traditional distribution-free model of pac-learning places certain restrictions on the ways in which a learner can learn, and these restrictions potentially drive up the sample complexity. For instance, the learner is restricted to receiving examples at *random* according to a fixed but unknown probability distribution: the learner has no choice over the particular examples seen. Additionally, the upper bounds for sample complexity must hold no matter what probability distribution the examples are drawn under: the bounds are worst case and so must hold even for extremely unusual distributions. These limitations in the traditional pac-learning model have caused researchers to look at ways of varying the model.

One of the principle means of varying the model is to allow the learner to ask queries, or questions, of various sorts. Angluin and others have shown that numerous concept classes that are not pac-learnable using the traditional model of drawing examples can be exactly learned in polynomial time under various query models. An example of such a class is the class of regular languages [1] [2]. Examples of classes that the membership-query model in particular can be used to learn are monotone DNF, monotone CNF [3] [18], read-once Boolean formulas [4], and the union of two half-spaces[5]. In sum, the extra power inherent in the query model has proven sufficient to learn classes that are not pac-learnable under the random example model.

1.6 The Results of this Thesis

This thesis examines two variations on the standard distribution-free model.

The first way we vary the model is to allow the learner to *choose* examples as well as *draw* random examples. Here, we combine the traditional pac-learning protocol of drawing random examples with the membership-query protocol of choosing examples to propose new models for learning using queries.

The second way we vary the model is to give the learner some advance knowledge about the probability distribution. We propose a model in which the learner knows *a priori* that the probability distribution is *smooth*. Since a smooth probability distribution is one that guarantees a specified upper bound on the amount of probability that can be associated with any region, certain distributions, such as extremely skewed distributions, are ruled out.

There are two key ways our new models differ from other query models. First, research on learning using queries has focused on learning classes that do *not* have polynomial computational complexity, and so are *too hard* to learn under the pac-model. This thesis looks at classes already known to be pac-learnable, and so which are already known to have polynomial sample complexity. The issue addressed then is not *whether* the given concept class has polynomial sample complexity, but rather whether using membership queries in conjunction with random examples can *reduce* the sample complexity of the concept class.

Second, most research on query models has focused on exactly identifying target concepts from classes with finite sample space. This thesis, however, looks at pac-learning classes which may have uncountably infinite sample space. So whereas membership queries are traditionally used to help *exactly identify* a concept from a *finite* class of concepts, we ask whether membership queries can help *pac-learn* (that is, approximately rather than exactly learn) a *potentially infinite*, even uncountable, concept class.

In Chapter 2, we show that under the distribution-free pac-learning model, the power to both choose and draw random examples cannot significantly reduce sample complexity. In contrast, in Chapter 3, we show that under a model in which the learner has some limited information about the nature of the probability distribution, the power to both choose and draw examples does lead to a significant reduction in sample complexity. In Chapter 4, we state conclusions and open problems.

One of our main results thus is that for traditional pac-learning models, membership queries do not help: the sample size required to pac-learn a dense-in-itself concept class if the learner can both choose and draw examples is within a constant factor of the sample size needed if the learner can just draw examples. Significantly, this lower bound is established for all probability distributions (as long as the probability distribution is unknown to the learner), not just for one worst-case probability distribution. On the other hand, we get significant results in favor of membership queries if the learner knows *a priori* that the probability distribution is smooth: if the learner can only draw examples, then the lower bound on sample complexity for pac-learning many concept classes does not improve, but if the learner can both choose and draw examples, then the sample and computational

complexity for pac-learning a hyperplane that omits a simplex is reduced significantly.

Chapter 2

A Lower Bound When Both Drawing and Choosing Examples

Up until now, when we have referred to pac-learning, we have meant the traditional model of pac-learning, in which the protocol for obtaining information about the sample space is to draw random examples. Henceforth, we consider any number of protocols to be viable ways of obtaining examples when pac-learning. In particular, we consider protocols that use random examples alone, membership queries alone, or a combination of both. We always specify the protocol being used.

2.1 A Lower Bound

Theorem 1 gives a lower bound on the number of examples required to pac-learn a dense-in-itself concept class $\mathcal{C}$ by any algorithm that can both draw random examples and make membership queries.

Theorem 1 *For every algorithm A that pac-learns a dense-in-itself concept class $\mathcal{C}$ using random examples and membership queries, and for every ϵ, $\delta > 0$, probability distribution D, and target concept $c_* \in \mathcal{C}$, the total number of examples A needs to pac-learn $\mathcal{C}$ is* $\Omega((1/\epsilon)\ln(1/\delta))$.

Note that A may be either randomized or deterministic, and it may draw examples and make membership queries in any order. The bound is on the worst-case number of examples seen, where the worst case is taken over the runs of the algorithm for the given target concept, probability distribution, ϵ, and δ. We assume in our asymptotic notation that ϵ, δ, and β (to be defined shortly) are going to zero in the limit.

Proof: We begin by assuming that a pac-learning algorithm A, an ϵ, a δ, a probability distribution D, and a target concept c_* have been given. We claim that we, as adversary, can produce a probability distribution D' and a concept c'_* that are close enough to D and

c_*, respectively, that if A does not draw $\Omega((1/\epsilon)\ln(1/\delta))$ random examples to distinguish these cases, then $\Pr(d_{D'}(c'_*, \hat{c}) \geq \epsilon) > \delta$, so that A is not a pac-learning algorithm.

Let β be an arbitrarily small positive number, and let c'_* be a concept different than c_* such that

$$\Pr(A \text{ sees examples in } c_* \oplus c'_* \mid A \text{ draws according to } D) < \beta \ .$$

The existence of c'_* is guaranteed by Lemma 1. Then

$$\Pr(A \text{ sees no examples in } c_* \oplus c'_* \mid A \text{ draws according to } D) \geq 1 - \beta \ .$$

Note that if A sees no examples in $c_* \oplus c'_*$, then A cannot distinguish between target concept c_* and target concept c'_*.

We now choose any point w in $c_* \oplus c'_*$. Define the probability distribution D_w to be the distribution that assigns probability 1 to point w and probability 0 to all other points.

Given probability distributions D and D_w, we make a new probability distribution D' as follows. We "skim" ϵ probability off of D, while keeping the relative distribution of points in D the same. That is, we uniformly remove ϵ probability from the probability distribution D such that the relative weight of points under D remains the same, but the total weight of points in D now adds up to $1 - \epsilon$. We assign the excess ϵ probability to w. Thus our new probability distribution D' is a linear combination of probability distribution D and probability distribution D_w:

$$D' = (1 - \epsilon)D + \epsilon D_w \ .$$

From now on, we suppose that algorithm A is drawing its random examples according to D'. When drawing an example according to D', there are two possibilities: either the example is drawn "as if according to D" (this happens with probability $1-\epsilon$), or the example is drawn "as if according to D_w" (this happens with probability ϵ). In general, we say that "A draws as if according to D" if A draws all of its random examples as if according to D.

Suppose that we run algorithm A on distribution D' and target concept c_*, that A draws all of its examples as if according to distribution D, and that A sees no examples in $c_* \oplus c'_*$. By definition, a pac-learning algorithm with high probability correctly classifies any point with weight ϵ or greater. Because A is a pac-learning algorithm and w has weight ϵ, A's output hypothesis $\hat{c}$ classifies w correctly with high probability. Thus $\hat{c}$ with high probability agrees with $c_*(w)$, and so disagrees with $c'_*(w)$.

We note that, on the one hand, c'_* is consistent with target concept c_* on all examples algorithm A sees with probability at least $1 - \beta$, yet, on the other hand, c'_* disagrees with

c_* on the classification of w. Since A is a pac-learning algorithm, therefore, when A now tries to learn c'_* instead of c_* on distribution D', we have that its output concept $\hat{c}$ satisfies

$$\Pr(\hat{c}(w) \neq c'_*(w) \mid A \text{ draws as if according to } D) \geq (1 - \delta - \beta) \ , \tag{2.1}$$

since we are assuming that A pac-learns c_* on D correctly, and with probability $1 - \beta$ the run looks the same as it would if c_* were the target concept. Observe that if A runs under probability distribution D', then saying $\hat{c}(w) \neq c'_*(w)$ is equivalent to saying $d_{D'}(\hat{c}, c'_*) \geq \epsilon$.

Let $m = m(\epsilon, \delta)$ denote the maximum possible total number of examples drawn and/or chosen by any run of A on inputs ϵ and δ, and any probability distribution and any target concept. Since random examples are independent events, if all m examples obtained are random examples, then

$$\Pr(A \text{ draws as if according to } D) = (1 - \epsilon)^m \ .$$

If less than m examples are random examples, then

$$\Pr(A \text{ draws as if according to } D) > (1 - \epsilon)^m \ .$$

Assume that m is so small that it satisfies

$$(1 - \epsilon)^m > \frac{\delta}{(1 - \delta - \beta)} \ ;$$

then

$$\Pr(A \text{ draws as if according to } D) > \frac{\delta}{(1 - \delta - \beta)} \ .$$

We show that this assumption implies that A does not pac-learn.

Assume that in a given run A receives information about m sample points, by random example according to probability distribution D', by membership query, or by both. What then is the probability that during a run of A (on input distribution D', with target concept c'_*) the following events both happen?

$E = A$ misclassifies w (and so makes error at least ϵ).

$F = A$ draws all its random examples as if according to D.

Now,

$$\Pr(EF) = \Pr(E \mid F) \Pr(F) \ .$$

Thus, since (by (2.1))

$$\Pr(E \mid F) = \Pr(A \text{ misclassifies } w \mid A \text{ draws as if according to } D) \geq 1 - \delta - \beta \ ,$$

and

$$\Pr(F) = \Pr(A \text{ draws as if according to } D) \geq (1-\epsilon)^m > \frac{\delta}{1-\delta-\beta} ,$$

we have

$$\Pr(EF) > (1-\delta-\beta) \cdot \frac{\delta}{1-\delta-\beta} = \delta .$$

Since $\Pr(E) \geq \Pr(EF) > \delta$, we have $\Pr(d_{D'}(c'_*, \hat{c}) \geq \epsilon) > \delta$, and therefore A has not pac-learned.

Solving for m in $(1-\epsilon)^m > \delta/(1-\delta-\beta)$, we have

$$m \ln(1-\epsilon) > \ln(\delta) - \ln(1-\delta-\beta),$$

so

$$\begin{aligned} m &< \frac{1}{\ln(1-\epsilon)}(\ln(\delta) - \ln(1-\delta-\beta)) , \\ &< \frac{-1}{\ln(1-\epsilon)}(\ln(\frac{1}{\delta}) + \ln(1-\delta-\beta)) . \end{aligned}$$

Thus, unless

$$m \geq \frac{-1}{\ln(1-\epsilon)}(\ln(\frac{1}{\delta}) + \ln(1-\delta-\beta)),$$

in other words, unless $m = \Omega((1/\epsilon)\ln(1/\delta))$, $\Pr(d_{D'}(c'_*, \hat{c}) \geq \epsilon) > \delta$, which proves our theorem.

∎

2.2 Comparison of the Lower Bound to Previous Results

We already know from Ehrenfeucht et al. [12] that for $\epsilon < 1/2$, an algorithm that can only draw random examples must see at least

$$\Omega\left(\frac{1}{\epsilon}\ln\frac{1}{\delta} + \frac{VCdim(C)}{\epsilon}\right)$$

examples to pac-learn, where $VCdim(C)$ is the Vapnik-Chervonenkis dimension of the class C. Since our bound is within a constant factor of their lower bound (treating $VCdim(C)$ as a constant as we let ϵ and δ go to zero), we have shown that the minimum number of examples needed by an algorithm A that can both draw random examples and make membership queries is within a constant factor of the minimum number of examples needed by an algorithm B that can only draw random examples.

We remark that, compared to the proof given by Ehrenfeucht, Haussler, Kearns, and Valiant [12], our proof has the advantage that it applies to *every* probability distribution (not just one constructed by the adversary), with the understanding of course that the learning algorithm has no prior knowledge about the actual probability distribution, and that the learning algorithm must pac-learn for every distribution. It also applies to learning from both random examples and membership queries (not just learning from random examples). On the other hand, our proof does not yield a lower bound that grows with the VC dimension of the concept class C.

Chapter 3

Smoothness Guarantees

Until now, we have assumed that any probability distribution was possible. Now suppose we have a "smoothness guarantee." Specifically, we assume that the learning algorithm has an additional input, s, and is told that any region of X having volume α, for any α, has probability at most $s\alpha$. Can this assumption improve our lower bound? Our first claim is that if the learning algorithm uses random examples only, then a smoothness guarantee does not reduce the lower bound on the number of random examples required to pac-learn.

We say $(\mathcal{C}, D)$ (where $\mathcal{C}$ is a concept class on a domain X and D is a probability distribution on X) *has all error rates* if for every target concept $c_* \in \mathcal{C}$, and for every nonnegative $\epsilon < \sup_{c \in \mathcal{C}}\{d_D(c_*, c)\}$, there is a concept $c \in \mathcal{C}$ such that $d_D(c_*, c) = \epsilon$. For instance, if D is smooth, and C is the concept class of half-spaces of $\mathbf{R}^n$, rectangles in the plane, or $\mathcal{I}$, then $(\mathcal{C}, D)$ *has all error rates.*

3.1 A Lower Bound When Drawing Examples Only

Theorem 2 *For every algorithm B that draws random examples to pac-learn $\mathcal{C}$, sufficiently small $\epsilon > 0$, $\delta > 0$, and smooth probability distribution D, if $(\mathcal{C}, D)$ has all error rates then the number of examples B needs to draw is greater than $(1/4\epsilon)\ln(1/\delta)$.*

Thus, even though an algorithm B has information that D is smooth, if $\mathcal{C}$ is a concept class such that $(\mathcal{C}, D)$ has all error rates, B still needs to draw $(1/4\epsilon)\ln(1/\delta)$ examples to pac-learn $\mathcal{C}$. Note that while the assumption that a concept class C under distribution D has all error rates may seem restrictive, as a practical matter it is quite reasonable, given the assumption of a smooth probability distribution. We leave it as an open problem to see whether this assumption actually is a restriction.

Proof:

Let c be a concept such that $d_D(c_*, c) = 2\epsilon$. (We have specified in the statement of Theorem 2 that ϵ must be sufficiently small. To be more precise, if $\epsilon < 1/2(\sup_{c \in \mathcal{C}}\{d_D(c_*, c)\})$,

then by the assumption that (C, D) has all error rates, there will always be a concept c with error rate 2ϵ. Adding this constraint to our earlier constraint in Section 1.3.2 that ϵ must be less that .39841, we have $\epsilon < \min(.39841, 1/2(\sup_{c \in C}\{d_D(c_*, c)\}))$.)

Let $S = c_* \oplus c$. As before, let $\hat{c}$ be the hypothesis concept that a run of algorithm B outputs, and let c'_* be the concept that we as adversary choose.

We begin by showing that if B sees no points in S, an adversary can force B to have error rate at least ϵ. We conclude by showing that unless B draws at least $(1/4\epsilon)\ln(1/\delta)$ examples, then with probability greater than δ the algorithm will see no points in S; in other words, with probability greater than δ the algorithm will have error rate at least ϵ.

Given that B sees no points in S, we assume without loss of generality that $\hat{c}$ is more likely to agree with c_* (in the probabilistic sense) than with c on points in S. Then

$$\Pr(\hat{c}(x) = c_*(x) : x \in S) \geq 1/2 ,$$

where the probability is based on the conditional distribution induced on S by distribution D.

In this case, we set c'_* to c. (If $\hat{c}$ is more likely to agree with c, we set c'_* to c_*; the reasoning for the rest of the proof remains the same.) Thus

$$\Pr(\hat{c}(x) \neq c'_*(x) : x \in S) \geq 1/2 ,$$

where the probability again is based on the conditional distribution induced on S by distribution D.

Since d_D is a bounded pseudometric, the triangle inequality holds for d_D, giving

$$d_D(c_*, \hat{c}) + d_D(\hat{c}, c'_*) \geq d_D(c_*, c'_*) . \tag{3.1}$$

Now, by assumption,

$$d_D(\hat{c}, c'_*) \geq d_D(c_*, \hat{c}) . \tag{3.2}$$

By (3.1) and (3.2), we have

$$2d_D(\hat{c}, c'_*) \geq d_D(c_*, c'_*) , \text{ or } d_D(\hat{c}, c'_*) \geq (d_D(c_*, c'_*))/2 . \tag{3.3}$$

Also, by assumption, we have

$$d_D(c_*, c) = 2\epsilon \ . \tag{3.4}$$

Since we have set c'_* to c, (3.4) is equivalent to

$$d_D(c_*, c'_*) = 2\epsilon \ . \tag{3.5}$$

Thus, combining (3.3) and (3.5), we obtain

$$d_D(\hat{c}, c'_*) \geq 2\epsilon/2 = \epsilon \ ,$$

which shows that if B sees no points in S, B has error rate at least ϵ.

The probability that B independently draws m examples and fails to see a point in S, that is, the probability that B has error rate at least ϵ, is $(1-2\epsilon)^m$. We fail to pac-learn if the probability of error rate at least ϵ is greater than δ, that is, if $(1-2\epsilon)^m > \delta$; given our assumption that $\epsilon < .39841$, we thus have that the lower bound on m even with the smoothness guarantee is $(1/4\epsilon)\ln(1/\delta)$. ∎

3.2 Upper and Lower Bounds When Both Drawing and Choosing Examples

So far in this chapter, we have shown that a smoothness guarantee does *not* significantly reduce the lower bound on sample complexity if the learning algorithm can only draw random examples. We now show, on the other hand, that if the learning algorithm can also make membership queries, then there exists a fast pac-learning algorithm whose sample and computational complexity *upper* bounds are well below the *lower* bounds proved in the previous section. We first prove this result for a one-dimensional concept class, and then extend the result to an n-dimensional concept class. (Our generalization to n dimensions is not completely general; we prove the result only for a specific concept class. It is an open problem to prove such a result for general bounded n-dimensional domains.)

3.2.1 Using the binary-search approach in one dimension

Theorem 3 *There exists an algorithm A that pac-learns $\mathcal{I}$ using membership queries, such that for every ϵ, δ, and smooth probability distribution D with known smoothness constant s, the number of examples A needs to pac-learn $\mathcal{I}$ is $\lg(s/\epsilon)$. For deterministic algorithms $\lg(s/\epsilon)$ is also a lower bound on the sample complexity. Additionally, if we assume unit cost for each example drawn or chosen, a deterministic algorithm A runs in time $\Theta(\lg(s/\epsilon))$.*

Note that although under this model the algorithm may both draw and choose examples, the algorithm we present only chooses examples.

Proof: The proof is based on a "binary search." We call a hypothesis interval an interval defined by two points, which we call L and R. Here L is the value of the rightmost positive example seen so far, and R is the value of the leftmost negative example seen so far. Thus the interval $[L, R]$ represents the points that we don't know how to classify yet; the right endpoint of c_*, our target concept, is somewhere within this interval.

In order to guarantee that the probability $s\alpha$ associated with a region of volume α is less than ϵ, we set up the following equation, which gives us the requisite size for α:

$$\begin{aligned} s\alpha &< \epsilon, \text{ or} \\ \alpha &< \epsilon/s. \end{aligned}$$

So if A has narrowed $[L, R]$ to a size less than ϵ/s, A is guaranteed that the probability associated with $[L, R]$ is less than $\epsilon/s \cdot s$, or ϵ, and therefore can output any $\hat{c}$ with right endpoint in $[L, R]$ with total confidence ($\delta = 0$) that the error rate is less than ϵ. Algorithm A can use membership queries to do a binary search on the initial interval $[0, 1]$, in which case A needs to make only $\lg(s/\epsilon)$ membership queries in order to narrow $[L, R]$ to a size less than ϵ/s. Observe that this *upper* bound of $\lg(s/\epsilon)$ on the sample complexity when using membership queries is substantially lower than the *lower* bound of $(1/4\epsilon)\ln(1/\delta)$ on the sample complexity when using random examples only.

To see that $\lg(s/\epsilon)$ is also a lower bound on the sample complexity for deterministic algorithms, consider an adversarial argument. An adversary can always label the requested examples so that after the algorithm makes $\lg(s/2\epsilon)$ queries, an interval $[L, R]$ of size $2\epsilon/s$ remains. At this point, the adversary can always force the algorithm to output a hypothesis with error rate at least ϵ. For instance, even if the algorithm outputs the concept represented by the midpoint of the interval, the adversary can choose the concept represented by L. Then, because the interval between L and the midpoint has length ϵ/s, in the worst case the hypothesis will have error rate $s \cdot \epsilon/s = \epsilon$. Since $\lg(s/2\epsilon) = \lg(s/\epsilon) - 1$, we have that greater than $\lg(s/\epsilon) - 1$ queries are needed to ensure that in all cases the error is less than ϵ; thus, at least $\lg(s/\epsilon)$ queries are necessary.

If we assume it takes constant time for each membership query, then we also achieve the tight bound of $\Theta(\lg(s/\epsilon))$ on the computational complexity for deterministic algorithms.

∎

3.2.2 Generalizing the binary-search approach to n dimensions

So far, we have given an algorithm which pac-learns the one-dimensional class $\mathcal{I}$ under the assumption of a smooth probability distribution. In short, such an algorithm pac-learns "half-spaces" of a one-dimensional line segment. We would also like to show how to pac-learn an analogous n-dimensional class. That is, we would like to pac-learn the set of half-spaces of an n-dimensional object, where the n-dimensional object is a logical generalization of the one-dimensional line segment.

There are a number of objects that come to mind as logical generalizations of the one-dimensional line segment. Two such generalizations are the hypercube and the hypersphere. But perhaps the simplest generalization of a one-dimensional segment is the simplex. We will generalize the binary-search approach of the above section to learn half-spaces of a simplex.

A simplex is a finite region of n-space enclosed by $n+1$ hyperplanes, where each hyperplane has dimension $n-1$. Importantly, a simplex has only $n+1$ corner points, $n+1$ faces, and $\binom{n+1}{2}$ edges. Each corner point is connected to each other corner point by an edge. For instance, a simplex in one dimension is a line, which is enclosed by two points (a point has dimension zero). A simplex in two dimensions is a triangle, which is enclosed by three lines. A simplex in three dimensions is a tetrahedron, which is enclosed by four planes, and so on [11]. A *regular* simplex is one in which each edge has the same length.

A simplex thus has certain properties that make it easier to use as the generalization of a line segment than a hypercube or a hypersphere. For instance, observe that the number of faces, corner points, and edges of a simplex is linear in the dimension. This contrasts sharply with a hypercube, for example, in which the number of edges and corner points is exponential in the dimension. Our algorithm capitalizes—in fact, depends—on these properties of the simplex. We leave it as an open problem to pac-learn the set of hyperplanes which intersect a hypercube or a hypersphere.

Let n represent the dimension of the concept class.

Theorem 4 *Let $\mathcal{S}$ be the class of hyperplanes which intersect a regular simplex. There exists an algorithm A that pac-learns $\mathcal{S}$ using membership queries, such that for every ϵ, δ, and smooth probability distribution D with known smoothness constant s, the number of examples A needs to pac-learn $\mathcal{S}$ is $n^2(\lg(s/\epsilon)+4)$. Additionally, if we assume unit cost for each example drawn or chosen, A runs in time $O(n^7 s/\epsilon)$.*

Proof:

We consider a regular simplex in which each edge has length 1, and we assume that the coordinates of the simplex corners are given to the learning algorithm. The domain X of $\mathcal{S}$ consists of exactly those points on the boundary of the simplex and within the simplex. Given a target hyperplane that intersects a simplex, we consider the target hyperplane and all domain points on one side of the hyperplane to be positive examples of the concept, and all domain points on the other side of the hyperplane to be negative examples of the concept. Thus concept class $\mathcal{S}$ is represented by the set of hyperplanes of $n-1$-dimensions that intersect the given simplex of n dimensions.

We begin our analysis by giving the formulas for computing the height and volume of a regular simplex whose edges have length 1.

The recurrence formula for the height of such an n-dimensional simplex is:

$$H_1 = 1;$$

$$H_{n+1} = \sqrt{1 - (H_n \cdot n/(n+1))^2}.$$

This evaluates to

$$H_n = \sqrt{(n+1)/2n}.$$

The volume V_n of an n-dimensional simplex of edge-length 1 is equal to the base of the simplex (B_n) times the height of the simplex (H_n) divided by the dimension n:

$$V_n = B_n \cdot H_n/n.$$

Since the base B_n of an n-dimensional simplex is equal to the volume V_{n-1} of the corresponding $n-1$-dimensional simplex, the recurrence formula for V_n is:

$$V_n = V_{n-1} \cdot H_n/n.$$

This evaluates to

$$V_n = \sqrt{(n+1)/2^n}/n!.$$

The surface area S_n of an n-dimensional simplex is equal to the number of faces of an n-dimensional simplex $(n+1)$ times the volume of an $n-1$-dimensional simplex (V_{n-1}):

$$S_n = (n+1) \cdot V_{n-1}$$

$$= (n+1) \cdot \sqrt{n/2^{n-1}}/(n-1)!.$$

For any target hyperplane which cuts the simplex, the learning algorithm A should output a hypothesis hyperplane such that the error rate of the hypothesis is less than ϵ.

Consider first the classification of each of the simplex corner points. In the trivial case where all $n+1$ corner points are labeled positive, all points inside the simplex are also positive. In this case any hypothesis which classifies all simplex points as positive suffices. (Similarly if all $n+1$ corner points are labeled negative.) Otherwise, there is at least one positive corner point and one negative corner point.

Let k be the total number of positive corner points, and let l be the total number of negative corner points. Then $k+l = n+1 =$ the total number of corner points. Since each corner point is connected to each other corner point, the total number of "mixed" edges, that is, edges such that one endpoint is positive and one is negative, is kl.

The separating target hyperplane must cut each of these kl mixed edges. In fact, the set of simplex edges that the target hyperplane cuts is precisely this set of kl mixed edges. (All other edges have both endpoints positive or both endpoints negative, and clearly the separating hyperplane cannot cut these edges.)

By definition, the length of each edge of the simplex is 1. We wish to find a much smaller interval, which we call a cut-interval, on each mixed edge. A cut-interval is a sub-interval of a mixed edge that still retains the property that one endpoint is positive and one endpoint is negative. Since a mixed edge is one dimensional, we can apply the binary-search techniques of the previous section to find a cut-interval of a specified size on the mixed edge.

Note we only need n points to determine an $n-1$-dimensional hyperplane. By doing a binary search along each of the kl mixed edges, we find kl cut-intervals (rather than points). Since kl is always at least n, these kl cut-intervals define a constrained n-dimensional *region* within which the hyperplane must lie. (By definition of a simplex, and the fact that each of the kl cut-intervals comes from a different edge of the simplex, we are guaranteed that the kl regions will indeed define an n-dimensional region.)

The overall strategy of the algorithm, then, is to find a cut-interval of a predetermined size, which we call *width*, on each of the kl mixed edges, and then output any hypothesis consistent with each of the cut-interval constraints.

We present the algorithm, give a derivation for the value of the key variable *width*, prove the correctness of the algorithm, and finally give the sample complexity (number

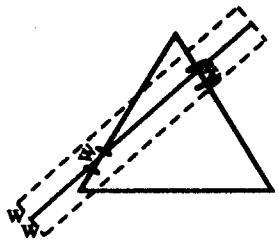

Figure 3.1: Simplex with pancake region within the dashed lines; $w = width$.

of membership queries needed) and computational complexity (time to find a hypothesis consistent with the constraints) analysis.

3.2.2.1 The algorithm

The algorithm begins by making $n + 1$ membership queries to obtain the classification of each corner point.

For each of the resulting kl mixed edges, the algorithm does a binary search using membership queries to find a cut-interval of size $width$ ($width$ is precisely defined shortly). Since the length of each simplex edge is 1, the maximum number of queries necessary to do each binary search is $\lg(1/width)$.

Each of the two endpoints of each cut-interval determines a constraint on where the target hyperplane can lie, so there are $2kl$ constraints total. Any hypothesis consistent with the constraints is output.

3.2.2.2 Derivation of *width*

In order to analyze the sample and computational complexity, we need to determine the size of $width$ necessary to achieve an error rate less than ϵ.

We form a "pancake" as follows: consider the $n-1$-dimensional target hyperplane and a one-dimensional line perpendicular to this hyperplane. If we move the target hyperplane along the perpendicular a distance of $width$ in each of the two directions away from the target hyperplane, then we trace out a pancake-like region with total width $2 \cdot width$. (See Figure 3.1.)

We claim that the only points on which the target hyperplane and any hypothesis consistent with all kl constraints can disagree must lie in this n-dimensional pancake that

encloses the target hyperplane. We thus wish to choose a value of *width* so that the entire pancake has total probability less than ϵ.

The volume of our pancake is clearly less than or equal to the width of the pancake times the volume of the maximum cross section of the pancake. The volume of the maximum cross section of the pancake in turn is less than or equal to the volume of the maximum cross section of the simplex, which in turn is less than or equal to the surface area S_n of the simplex. (We conjectured that the maximum cross section of a simplex is at most equal to the surface area of a *single* face of the simplex, but could not prove this conjecture.) Thus the volume of the pancake must be less than or equal to the width of the pancake, which is $2 \cdot width$, times the surface area of the simplex, which is S_n.

We wish that the total volume of the pancake should have probability weight less than ϵ. So given smoothness constant s, it is necessary to have

$$s \cdot (2 \cdot width \cdot S_n) < \epsilon. \tag{3.6}$$

This is equivalent to

$$s \cdot 2 \cdot width \cdot (n+1) \cdot V_{n-1} < \epsilon,$$

and to

$$s \cdot 2 \cdot width \cdot (n+1) \cdot \sqrt{n/2^{n-1}}/(n-1)! < \epsilon,$$

yielding (with rearrangements of terms)

$$width < \epsilon/s \cdot 1/2 \cdot (\sqrt{2^{n-1}/n} \cdot (n-1)!/(n+1)). \tag{3.7}$$

Any value of *width* which satisfies (3.7) will also satisfy (3.6), and any value of *width* which satisfies (3.6) will ensure that the total volume of the pancake has probability weight less than ϵ. So we choose *width* equal to one half the right-hand side of (3.7) –

$$\begin{aligned} width &= 1/2(\epsilon/s \cdot 1/2 \cdot (\sqrt{2^{n-1}/n} \cdot (n-1)!/(n+1)) \\ &= \epsilon/s \cdot 1/4 \cdot (\sqrt{2^{n-1}/n} \cdot (n-1)!/(n+1)). \end{aligned}$$

– and then analyze the resulting number of membership queries required given this choice of *width*.

The number of membership queries necessary to narrow a cut-interval to size *width* is $\lg(1/width)$. Here, we have

$$\lg(1/width) = \lg(s/\epsilon) + \lg 4 + \lg(\sqrt{n/2^{n-1}} \cdot (n+1)/(n-1)!).$$

Since

$$\lg(\sqrt{n/2^{n-1}} \cdot (n+1)/(n-1)!) < 1.6$$

for all n, (and in fact is less than 0 for $n > 3$), and

$$\lg 4 = 2,$$

we have that $\lg(s/\epsilon) + 4$ is an upper bound on $\lg(1/width)$:

$$\lg(1/width) < \lg(s/\epsilon) + 4$$

So the number of membership queries necessary on each mixed edge is at most $\lg(s/\epsilon)+4$. Thus, we must do a binary search making at most $\lg(s/\epsilon) + 4$ membership queries on each of the kl mixed edges. Since $kl \leq n^2$ ($1 \leq k \leq n$, $1 \leq l \leq n$, thus $kl \leq n^2$), the total sample complexity is at most $n^2(\lg(s/\epsilon) + 4)$.

3.2.2.3 Correctness of the algorithm

We claim that the maximum error rate of any hypothesis hyperplane consistent with the $2kl$ constraints is less than ϵ. We prove this by showing that all the points on which the target and hypothesis hyperplanes disagree—that is, the symmetric difference of the target and hypothesis concepts—must lie in the n-dimensional pancake region of width $2 \cdot width$ that surrounds the target hyperplane. Since this pancake region has total probability weight less than ϵ, the error rate of any consistent hypothesis must be less than ϵ.

We first show that each cut-interval endpoint lies within the pancake region, and then show that all points in the symmetric difference of the target and hypothesis concepts lie within the pancake.

The total length of each cut-interval is *width*. This implies that the projection of each cut-interval endpoint onto the line perpendicular to the target hyperplane must also have distance to the target hyperplane less than or equal to *width*. Since the pancake is created by moving *width* along the perpendicular to the target hyperplane in each of the two directions away from the target hyperplane, each endpoint thus must lie within the pancake.

We now establish that *any* point (not just the endpoints) on which the target and hypothesis concepts disagree lies within the specified pancake region.

Consider the convex hull created by connecting the $2kl$ cut-interval endpoints. (We remark that not all of the $\binom{2kl}{2}$ edges formed by connecting endpoints define the convex hull—some of these edges lie inside the convex hull.) The target and hypothesis hyperplanes lie entirely within this convex hull, since each point where the target and hypothesis

hyperplanes intersect an edge of the simplex is within a cut-interval. Thus, any point in the symmetric difference of the hypothesis and target concepts also lies within the convex hull. Now we establish that every point within the convex hull lies within the pancake region.

As already stated, the distance from the target hyperplane to the projection of each of the $2kl$ cut-interval endpoints onto the target hyperplane perpendicular must be less than or equal to *width*. Since any point within the convex hull is a convex linear combination of cut-interval endpoints, we know any point in the convex hull region must also have distance to the target concept less than or equal to *width*. But then every point in the convex hull, which in particular includes all points in the symmetric difference of the target and hypothesis concepts, must be contained within the pancake region. Thus, since the pancake has total probability at most ϵ, the hypothesis error rate must be less than ϵ.

3.2.2.4 Complexity of the algorithm

The computational complexity is equal to the time necessary to compute a hypothesis consistent with the $2kl$ constraints. We can solve this constraint problem using linear programming in time polynomial in the number of constraints, $2kl$, and the precision of each constraint, $\lg(1/width)$. The worst-case bound for solving the constraint problem with $2kl$ constraints and bit precision $\lg(1/width)$ requires time $O((2kl)^{3.5} \cdot \lg(1/width))$. Since $2kl$ is $O(n^2)$, and $\lg(1/width)$ is $O(s/\epsilon)$, we can find a hypothesis consistent with all $2kl$ constraints in time $O(n^7 s/\epsilon)$.

■

hull—some of these edges lie inside the convex hull.) The target and hypothesis hyperplanes lie entirely within this convex hull, since each point where the target and hypothesis

Chapter 4

Conclusions and Open Problems

We have shown that membership queries do not help us to pac-learn a wide variety of classes. We have also shown that *a priori* knowledge of a smooth probability distribution does not help us to pac-learn many concept classes, when we are only allowed to draw random examples. A smooth probability distribution, however, does help us to pac-learn certain of these classes when we can also make membership queries. In particular, we have shown a significantly reduced upper bound on the number of samples needed to learn $\mathcal{I}$, which is a concept class in one dimension, and $\mathcal{S}$, which is a concept class in n dimensions. We hope to generalize this upper bound to handle many other classes.

The following open problems arise from this research:

1. Improve the lower bound of Theorem 1 to depend upon $VCdim(\mathcal{C})$.

2. Generalize the n-dimensional approach of Theorem 3 to handle $n-1$-dimensional hyperplanes intersecting arbitrary bounded convex n-dimensional domains (e.g. hyperspheres or hypercubes).

3. In regards to solving 2 above, we hypothesize that it is possible to make a reduction from our problem to the problem solved by Maass and Turan [15] of exactly learning a target hyperplane in a boolean-valued n-dimensional sample space. In making the reduction we would capitalize on the fact that we are working with a smooth probability distribution and so simulate a boolean-valued domain in our continuous space. We would choose a small enough value for the distance between the boolean-valued points, so that if an algorithm exactly learned the target hyperplane, the error rate in the continuous space would be less than ϵ. This reduction is not worked out, but if correct, would give good bounds on learning half-spaces in n-dimensional space.

Chapter 5

Acknowledgments

I first of all thank Ron Rivest. He has been a superb and supportive advisor.

I thank Avrim Blum, Mic Grigni, Alex Ishii, Joe Kilian, James Park, Rob Schapire, and Robert Solovay for their helpful discussions and comments.

I thank Jonathan Amsterdam, Aditi Dhagat, Sally Goldman, Michael Kearns, Norman Margolus, and Rob Schapire for proof-reading my paper and giving me advice on my COLT talk.

Thanks especially to Tandy Warnow for always being such a nurturing and supportive friend-teacher, and to Phil Rogaway for always giving me such good advice.

Special thanks to Be Hubbard both for all her help to me personally and for helping to create and maintain the warm, supportive, thriving theory community.

I also thank all my other wonderful friends not already mentioned above, including: Wendy Brunzie, Carolyn Cox, Ann Feehan, Shalom Franck, Debra Goldentyer, Deborah Greif, Agiua Heath, Maya Kopell, Suzanne Lawrence, Mojdeh Mohteshemi, Sharon Port, Karen Saracik, Mark Schaeffer, Michal Shimshoni, Mona Singh, Julie Sweedler, Margaret and Mark Tuttle, Debbie Utley, Debra Waller, Jill Werman, Su-Ming Wu, and Guillermo Zemba.

Most of all, I thank my wonderful parents for all their love and support.

Bibliography

[1] Dana Angluin. Types of queries for concept learning. Technical Report YALEU/DCS/TR-479, Yale University Department of Computer Science, June 1986.

[2] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, November 1987.

[3] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.

[4] Dana Angluin, Lisa Hellerstein, and Marek Karpinski. Learning read-once formulas with queries. Technical report, University of California, Report No. UCB/CSD 89/528, 1989.

[5] Eric B. Baum. On learning a union of half spaces. *Journal of Complexity*, 6(1):67–101, March 1990.

[6] Shai Ben-David and Gyora M. Benedek. Measurability constraints on pac learnability. Technical report, Technion, Haifa, 1991.

[7] Shai Ben-David, Gyora M. Benedek, and Yishay Mansour. A parametrization scheme for classifying models of learnability. In *Proceedings of COLT '89*, pages 285–302. Morgan Kaufmann, 1989.

[8] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 273–282, Berkeley, California, May 1986.

[9] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[10] Thomas M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications to pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14(3):326–334, 1965.

[11] H. S. M. Coxeter. *Regular Polytopes.* Dover Publications, Inc., New York, second edition, 1973.

[12] Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–251, September 1989.

[13] David Haussler. Probably approximately correct learning. Technical Report UCSC-CRL-90-16, University of California Santa Cruz, May 1990.

[14] Michael Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 433–444, Seattle, Washington, May 1989.

[15] Wolfgang Maass and Györgi Turán. How fast can a threshold gate learn? In Drastal, Hanson, and Rivest, editors, *Computational Learning Theory and Natural Learning Systems: Constraints and Prospects.* MIT Press. (To appear.).

[16] Leonard Pitt. Recent results in computational learning theory, 1990.

[17] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. Technical report, Harvard University Aiken Computation Laboratory, July 1986.

[18] Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

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

[20] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, New York, 1982.