Noise Tolerant Algorithms for Learning and Searching

by

Javed Alexander Aslam

S.M., Electrical Engineering and Computer Science
Massachusetts Institute of Technology
(1992)
B.S.E.E., Electrical and Computer Engineering
University of Notre Dame
(1987)

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1995

Signature of Author ______________________________
Department of Electrical Engineering and Computer Science
January 27, 1995

Certified by ______________________________
Ronald L. Rivest
Professor of Computer Science
Thesis Supervisor

Accepted by ______________________________
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Noise Tolerant Algorithms for Learning and Searching

by

Javed Alexander Aslam

Submitted to the Department of Electrical Engineering and Computer Science
on January 27, 1995,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

We consider the problem of developing robust algorithms which cope with noisy data. In the Probably Approximately Correct model of machine learning, we develop a general technique which allows nearly all PAC learning algorithms to be converted into highly efficient PAC learning algorithms which tolerate noise. In the field of combinatorial algorithms, we develop techniques for constructing search algorithms which tolerate linearly bounded errors and probabilistic errors.

In the field of machine learning, we derive general bounds on the complexity of learning in the recently introduced Statistical Query model and in the PAC model with noise. We do so by considering the problem of improving the accuracy of learning algorithms. In particular, we study the problem of "boosting" the accuracy of "weak" learning algorithms which fall within the Statistical Query model, and we show that it is possible to improve the accuracy of such learning algorithms to any arbitrary accuracy. We derive a number of interesting consequences from this result, and in particular, we show that nearly all PAC learning algorithms can be converted into highly efficient PAC learning algorithms which tolerate classification noise and malicious errors.

We also investigate the longstanding problem of searching in the presence of errors. We consider the problem of determining an unknown quantity x by asking "yes-no" questions, where some of the answers may be erroneous. We focus on two different models of error: the *linearly bounded* model, where for some known constant $r < 1/2$, each initial sequence of i answers is guaranteed to have no more than ri errors, and the *probabilistic* model, where errors occur randomly and independently with probability $p < 1/2$. We develop highly efficient algorithms for searching in the presence of linearly bounded errors, and we further show that searching in the presence of probabilistic errors can be efficiently reduced to searching in the presence of linearly bounded errors.

Thesis Supervisor: Ronald L. Rivest
Title: Professor of Computer Science

Acknowledgements

Thanks go first to my advisor, Ron Rivest. Ron gave me my start in research at MIT, and in the years that followed, he has always been supportive, quick to give advice, and a font of knowledge, wisdom and insight. It has been a pleasure working with Ron, both as an advisee and teaching assistant; I can't thank him enough for the experience.

Much of the research in this thesis is joint work with two colleagues, each of whom I consider good friends. The research in Part I of this thesis is largely joint work with Scott Decatur, and the research in Part II is largely joint work with Aditi Dhagat. Never has work been so enjoyable as when I've been locked in a room proving theorems with either of them.

I would also like to thank all of the friends I have made at MIT for making the time I spent there, both at and away from work, special. In this regard, thanks go especially to Margrit Betke, Scott Decatur, Aditi Dhagat, Trevor Jim and Mona Singh. Life these past years would not have been much fun without them. Special thanks also go to the heart and soul of the Theory of Computation Group at MIT, Be Hubbard, for her endless supply of warmth, cheer and chocolates.

Thanks also go to my siblings Tariq & Kim, Tahira & Tim, Khalil, Anita & Franco, Anne and Linda. I can't imagine life without them.

Finally, my greatest thanks are reserved for my parents, Muhammad and Carol Aslam. I cannot express in words what they have meant to me. I owe my parents everything, far more than I could ever repay in spirit or in kind. This thesis is dedicated to them.

Table of Contents

CHAPTER 1

Introduction

This thesis is concerned with the problems of *concept learning* and *searching*, and in particular, it deals with the problem of coping with faulty data in each of these settings. In the sections that follow, we informally introduce the topics of concept learning and searching.

1.1 Concept Learning

A *concept* is simply a rule which divides objects into two categories: *positive examples* and *negative examples*. The concept "circular region of radius 1 centered at the origin," for instance, divides all points in the plane into positive and negative examples. All points in the plane whose Euclidean distance to the origin is at most 1 are positive examples of this concept, while all points in the plane whose Euclidean distance to the origin is greater than 1 are negative examples. *Concept learning* is the problem of inferring a rule from a set of positive and negative examples of that rule.

Devising machines, algorithms and programs which can learn concepts from positive and negative examples is an important goal of artificial intelligence, and it has motivated much research in the field. In this thesis, we approach the problem of concept learning from the perspective of *computational learning theory* [1] in that we are concerned with the ability to learn concepts *efficiently*. A machine or algorithm for concept learning is said to be efficient if the quantities of resources it uses (*e.g.* time, space, examples, etc.) are bounded by polynomials in the various learning parameters.

Many models of concept learning have been developed, and we focus on one such model, the Probably Approximately Correct (PAC) model introduced by Valiant [34]. In the PAC model of learning, the unknown concept to be learned is assumed to be a member of a known *concept class*, and our goal is to develop a general algorithm which can learn any concept in the known class from examples of that concept. In the above example, for instance, the known concept class may be "circular regions in the plane," and the unknown target concept is "circular region of radius 1 centered at the origin."

A great many algorithms have been devised to "PAC-learn" various concept classes [1]; however, nearly all of these algorithms are "brittle" in the sense that they cannot tolerate noisy data. In Part I of this thesis, a model of PAC learning is studied which is both *general* in the sense that it encompasses nearly all known PAC learning algorithms and *robust* in the sense that many types of noise can be cleanly and efficiently accommodated. We extend and improve this model of learning in ways which both increase the power of the algorithm designer and decrease the complexity of the resultant robust learning algorithms. A more formal introduction to this topic and to the corresponding results contained within this thesis can be found in Chapter 2.

1.2 Searching

The problem of search is a fundamental one in computer science, and its importance need hardly be justified. Perhaps the simplest example of search is embodied in the problem of finding a particular element within a sorted collection of elements. The standard *binary search* algorithm can be used to optimally solve this problem wherein all of the requisite queries are *comparison questions*. This, however, is but one type of search. Consider also, for example, the problem of medical diagnosis. A patient is suffering from some unknown disease, and a doctor has at his disposal a number of tests, each of which rules out some number of possible maladies. By performing some number of these tests, the doctor, can, in principle, eliminate all possibilities until only one remains. This too is an example of search wherein the "elements" (possible diseases) are not ordered and the queries themselves correspond to *subset questions*. Instances of search are pervasive, and we provide these two examples merely for illustrative purposes.

The problem of searching in an optimal fashion is well understood, but not if one were to allow "faulty" data. For example, consider the aforementioned medical diagnosis example, and suppose that some tests were known to occasionally be false-positive or false-negative. It is problems such as this which motivate the work in Part II of this thesis wherein various models of searching in the presence of faulty data are studied and robust algorithms for performing search are developed. A more formal introduction to this topic and to the corresponding results contained within this thesis can be found in Chapter 7.

Part I

Learning in the Presence of Noise

CHAPTER 2

Introduction

The statistical query model of learning was created so that algorithm designers could construct noise-tolerant PAC learning algorithms in a natural way. Ideally, such a model of robust learning should restrict the algorithm designer as little as possible while maintaining the property that these new learning algorithms can be efficiently simulated in the PAC model with noise. In the following chapters, we both extend and improve the current statistical query model in ways which both increase the power of the algorithm designer and decrease the complexity of simulating these new learning algorithms. In this chapter, we summarize our results and introduce the various models of learning required for the exposition that follows.

2.1 Introduction

Since Valiant's introduction of the Probably Approximately Correct model of learning [34], PAC learning has proven to be an interesting and well studied model of machine learning. In an instance of PAC learning, a learner is given the task of determining a close approximation of an unknown $\{0, 1\}$-valued *target function* f from *labelled examples* of that function. The learner is given access to an *example oracle* and accuracy and confidence parameters. When polled, the oracle draws an instance according to a distribution D and returns the instance along with its label according to f. The error rate of an hypothesis output by the learner is the probability that an instance chosen according to D will be mislabelled by the hypothesis. The learner is required to output an hypothesis such that, with high confidence, the error rate of the hypothesis is less than the accuracy parameter. Two standard complexity measures studied in the PAC model are *sample complexity* and *time complexity*. Efficient PAC learning algorithms have been developed for many function classes [1], and PAC learning continues to be a popular model of machine learning.

The model of learning described above is often referred to as the *strong learning* model since a learning algorithm may be required to output an arbitrarily accurate hypothesis depending on the accuracy parameter supplied. An interesting variant referred to as the *weak learning* model is identical, except that there is no accuracy parameter and the output hypothesis need only have error rate slightly less than 1/2. In other words, the output of a weak learning algorithm need only perform slightly better than random guessing. A fundamental and surprising result first shown by Schapire [28, 29] and later improved upon by Freund [14, 15] states that any algorithm which efficiently weakly learns can be transformed into an algorithm which efficiently strongly learns. These results have important consequences for PAC learning, including providing upper bounds on the time and sample complexities of strong learning.

One criticism of the PAC model is that the data presented to the learner is assumed to be *noise-free*. In fact, most of the standard PAC learning algorithms would fail if even a small number of the labelled examples given to the learning algorithm were "noisy." Two popular noise models for both theoretical and experimental research are the *classification noise* model introduced by Angluin and Laird [2, 21] and the *malicious error* model introduced by Valiant [35] and further studied by Kearns and Li [20]. In the classification noise model, each example received by the learner is mislabelled randomly and independently with some fixed probability. In the malicious error model, an adversary is allowed, with some fixed probability, to substitute a labelled example of his choosing for the labelled example the learner would ordinarily see.

While a limited number of efficient PAC algorithms had been developed which tolerate classification noise [2, 16, 26], no general framework for efficient learning[1] in the presence of classification noise was known until Kearns introduced the Statistical Query model [19].

[1] Angluin and Laird [2] introduced a general framework for learning in the presence of classification noise. However, their methods do not yield computationally efficient algorithms in most cases.

In the SQ model, the example oracle of the standard PAC model is replaced by a statistics oracle. An SQ algorithm queries this new oracle for the values of various statistics on the distribution of labelled examples, and the oracle returns the requested statistics to within some specified additive error. Upon gathering a sufficient number of statistics, the SQ algorithm returns an hypothesis of the desired accuracy. Since calls to a statistics oracle can be *simulated* with high probability by drawing a sufficiently large sample from an example oracle, one can view this new oracle as an intermediary which effectively limits the way in which a learning algorithm can make use of labelled examples. Two standard complexity measures of SQ algorithms are *query complexity*, the maximum number of statistics required, and *tolerance*, the minimum additive error required. The time and sample complexities of simulating SQ algorithms in the PAC model are directly affected by these measures; therefore, we would like to bound these measures as closely as possible.

Kearns [19] has demonstrated two important properties of the SQ model which make it worthy of study. First, he has shown that nearly every PAC learning algorithm can be cast within the SQ model, thus demonstrating that the SQ model is quite general and imposes a rather weak restriction on learning algorithms. Second, he has shown that calls to a statistics oracle can be simulated, with high probability, by a procedure which draws a sufficiently large sample from a *classification noise oracle*. An immediate consequence of these two properties is that nearly every PAC learning algorithm can be transformed into one which tolerates classification noise.

Decatur [9] has further demonstrated that calls to a statistics oracle can be simulated, with high probability, by a procedure which draws a sufficiently large sample from a *malicious error oracle*. Thus, nearly every PAC learning algorithm can be transformed into one which tolerates malicious errors. While Kearns and Li [20] had previously demonstrated a general technique for converting a PAC learning algorithm into one which tolerates small amounts of malicious error, the results obtained by appealing to SQ are better in some interesting cases [9].

While greatly expanding the function classes known to be learnable in the presence of noise, Kearns' technique does not constitute a formal reduction from PAC learning to SQ learning. In fact, such a reduction cannot exist: while the class of parity functions is known to be PAC learnable [17], Kearns has shown that this class is provably unlearnable in the SQ model.

Kearns' technique for converting PAC algorithms to SQ algorithms consists of a few general rules, but each PAC algorithm must be examined in turn and converted to an SQ algorithm individually. Thus, one cannot derive general upper bounds on the complexity of SQ learning from upper bounds on the complexity of PAC learning, due to the dependence on the specific conversion of a PAC algorithm to an SQ algorithm. A consequence of this fact is that general upper bounds on the time and sample complexities of PAC learning in the presence of noise are not directly obtainable either.

We obtain bounds for SQ learning and PAC learning in the presence of noise by making use of the following result. We define weak SQ learning in a manner analogous to weak PAC learning, and we show that it is possible to boost the accuracy of weak SQ algorithms to obtain strong SQ algorithms. Thus, we show that weak SQ learning is equivalent to strong SQ learning. We use the technique of "boosting by majority" [15] which is nearly optimal in terms of its dependence on the accuracy parameter ϵ.

In the SQ model, as in the PAC model, this boosting result allows us to derive general upper bounds on many complexity measures of learning. Specifically, we derive simultaneous upper bounds with respect to ϵ on the number of queries, $O(\log^2 \frac{1}{\epsilon})$, the Vapnik-Chervonenkis dimension of the query space, $O(\log \frac{1}{\epsilon} \log \log \frac{1}{\epsilon})$, and the inverse of the minimum tolerance, $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$. In addition, we show that these general upper bounds are nearly optimal by describing a class of learning problems for which we simultaneously lower bound the number of queries by $\Omega(\frac{d}{\log d} \log \frac{1}{\epsilon})$ and the inverse of the minimum tolerance by $\Omega(\frac{1}{\epsilon})$. Here d is the Vapnik-Chervonenkis dimension of the function class to be learned.

The complexity of a statistical query algorithm in conjunction with the complexity of simulating SQ algorithms in the various noise models determine the complexity of the noise-tolerant PAC learning algorithms obtained. Kearns [19] has derived general bounds on the minimum complexity of SQ algorithms, and we derive some specific lower bounds as well. Our boosting result provides a general technique for constructing SQ algorithms which are nearly optimal with respect to these bounds. However, the robust PAC learning algorithms obtained by simulating even *optimal* SQ algorithms in the presence of noise are inefficient when compared to known lower bounds for PAC learning in the presence of noise [11, 20, 30]. In fact, the PAC learning algorithms obtained by simulating optimal SQ algorithms in the *absence* of noise are inefficient when compared to the tight bounds known for noise-free PAC learning [7, 11]. These shortcomings could be consequences of either inefficient simulations or a deficiency in the model itself. In this thesis, we show that both of these explanations are true, and we provide both new simulations and a variant of the SQ model which combat the current inefficiencies of PAC learning via the statistical query model.

We improve the complexity of simulating SQ algorithms in the presence of classification noise by providing a more efficient simulation. If τ_* is a lower bound on the minimum additive error requested by an SQ algorithm and $\eta_b < 1/2$ is an upper bound on the unknown noise rate, then Kearns' original simulation essentially runs $\Theta(\frac{1}{\tau_*(1-2\eta_b)^2})$ different copies of the SQ algorithm and processes the results of these runs to obtain an output. We show that this "branching factor" can be reduced to $\Theta(\frac{1}{\tau_*} \log \frac{1}{1-2\eta_b})$, thus reducing the time complexity of the simulation. We also provide a new and simpler proof that statistical queries can be estimated in the presence of classification noise, and we show that our formulation can easily be generalized to accommodate a strictly larger class of statistical queries.

We improve the complexity of simulating SQ algorithms in the absence of noise and in the presence of malicious errors by proposing a natural variant of the SQ model and providing efficient simulations for this variant. In the *relative error* SQ model, we allow SQ algorithms to submit statistical queries whose estimates are required within some specified relative error. We show that a class is learnable with relative error statistical queries if and only if it is learnable with (standard) additive error statistical queries. Thus, known learnability and hardness results for statistical query learning [6, 19] also hold in this variant.

We demonstrate general bounds on the complexity of relative error SQ learning, and we show that many learning algorithms can naturally be written as highly efficient, relative error SQ algorithms. We further provide simulations of relative error SQ algorithms in both the absence and presence of noise. These simulations in the absence of noise and in the presence of malicious errors are more efficient than the simulations of additive error SQ algorithms, and given a roughly optimal relative error SQ algorithm, these simulations yield roughly optimal PAC learning algorithms. These results hold for *all* function classes which are SQ learnable

Finally, we show that our simulations of SQ algorithms in the absence of noise, in the presence of classification noise, and in the presence malicious errors can all be modified to accommodate a strictly larger class of statistical queries. In particular, we show that our simulations can accommodate *real-valued* statistical queries. Real-valued queries allow an algorithm to query the *expected value* of a real-valued function of labelled examples. Our results on improved simulations hold for this generalization in both the absence and presence of noise.

The remainder of this work is organized as follows. In Section 2.2, we formally define the learning models of interest, and in Section 2.3, we describe PAC model boosting results which are used in later chapters. In Chapters 3 and 4, we present our additive error and relative error SQ model results, respectively. In Chapter 5, we present some extensions of our results, and we conclude with a discussion of some open questions in Chapter 6.

2.2 Learning Models

In this section, we formally define the relevant models of learning necessary for the exposition that follows. We begin by defining the weak and strong PAC learning models, followed by the classification noise and malicious error models, and finally the statistical query model.

2.2.1 The Weak and Strong PAC Learning Models

In an instance of PAC learning, a learner is given the task of determining a close approximation of an unknown $\{0,1\}$-valued target function from labelled examples of that function. The unknown target function f is assumed to be an element of a known function class $\mathcal{F}$ defined over an instance space X. The instance space X is typically either the Boolean hypercube $\{0,1\}^n$ or n-dimensional Euclidean space $\Re^n$. We use the parameter n to denote the common length of each instance $x \in X$.

We assume that the instances are distributed according to some unknown probability distribution D on X. The learner is given access to an example oracle $EX(f, D)$ as its source of data. A call to $EX(f, D)$ returns a labelled example $\langle x, l \rangle$ where the instance $x \in X$ is drawn randomly and independently according to the unknown distribution D, and the label l is equal to $f(x)$. We often refer to a sequence of labelled examples drawn from an example oracle as a *sample*.

A learning algorithm draws a sample from $EX(f, D)$ and eventually outputs an hypothesis

h from some hypothesis class $\mathcal{H}$ defined over X. For any hypothesis h, the *error rate* of h is defined to be the probability that h mislabels an instance drawn randomly according to D. By using the notation $\Pr_D[P(x)]$ to denote the probability that a predicate P is satisfied by an instance drawn randomly according to D, we may define $\mathit{error}(h) = \Pr_D[h(x) \neq f(x)]$. We often think of $\mathcal{H}$ as a class of representations of functions in $\mathcal{F}$, and as such we define $\mathit{size}(f)$ to be the size of the smallest representation in $\mathcal{H}$ of the target function f.

The learner's goal is to output, with probability at least $1-\delta$, an hypothesis h whose error rate is at most ϵ, for the given *accuracy parameter* ϵ and *confidence parameter* δ. A learning algorithm is said to be *polynomially efficient* if its running time is polynomial in $1/\epsilon$, $1/\delta$, n and $\mathit{size}(f)$. We formally define PAC learning as follows (adapted from Kearns [19]):

Definition 1 *(Strong PAC Learning)*
Let $\mathcal{F}$ and $\mathcal{H}$ be function classes defined over X. The class $\mathcal{F}$ is said to be polynomially learnable by $\mathcal{H}$ if there exists a learning algorithm $\mathcal{A}$ and a polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that for any $f \in \mathcal{F}$, for any distribution D on X, for any accuracy parameter ϵ, $0 < \epsilon \leq 1$, and for any confidence parameter δ, $0 < \delta \leq 1$, the following holds: if $\mathcal{A}$ is given inputs ϵ and δ, and access to an example oracle $EX(f,D)$, then $\mathcal{A}$ halts in time bounded by $p(1/\epsilon, 1/\delta, n, \mathit{size}(f))$ and outputs an hypothesis $h \in \mathcal{H}$ that with probability at least $1-\delta$ satisfies $\mathit{error}(h) \leq \epsilon$.

As stated, this is often referred to as *strong learning* since the learning algorithm may be required to output an arbitrarily accurate hypothesis depending on the input parameter ϵ. A variant of strong learning called *weak learning* is identical, except that there is no accuracy parameter ϵ and the output hypothesis need only have error rate slightly less than $1/2$, *i.e.* $\mathit{error}(h) \leq \frac{1}{2} - \gamma = \frac{1}{2} - \frac{1}{p(n,\mathit{size}(f))}$ for some polynomial p. Since random guessing would produce an error rate of $1/2$, one can view the output of a weak learning algorithm as an hypothesis whose error rate is slightly better than random guessing. We refer to the output of a weak learning algorithm as a *weak hypothesis* and the output of a strong learning algorithm as a *strong hypothesis*.

2.2.2 The Classification Noise and Malicious Error Models

One criticism of the PAC model is that the data presented to the learner is required to be *noise-free*. Two popular models of noise for both experimental and theoretical purposes are the *classification noise model* introduced by Angluin and Laird [2, 21] and the *malicious error model* introduced by Valiant [35].

The Classification Noise Model

In the *classification noise model*, the example oracle $EX(f, D)$ is replaced by a noisy example oracle $EX_{\mathrm{CN}}^{\eta}(f, D)$. Each time this noisy example oracle is called, an instance $x \in X$ is drawn according to D. The oracle then outputs $\langle x, f(x)\rangle$ with probability $1 - \eta$ or $\langle x, \neg f(x)\rangle$ with probability η, randomly and independently for each instance drawn. Despite the noise in the labelled examples, the learner's goal remains to output an hypothesis h which, with probability at least $1 - \delta$, has error rate $\mathit{error}(h) = \Pr_D[h(x) \neq f(x)]$ at most ϵ.

While the learner does not typically know the exact value of the *noise rate* η, the learner is given an upper bound η_b on the noise rate, $0 \leq \eta \leq \eta_b < 1/2$, and the learner is said to be polynomially efficient if its running time is polynomial in the usual PAC learning parameters as well as $\frac{1}{1-2\eta_b}$.

The Malicious Error Model

In the *malicious error model*, the example oracle $EX(f, D)$ is replaced by a noisy example oracle $EX_{\mathrm{MAL}}^{\beta}(f, D)$. When a labelled example is requested from this oracle, with probability $1 - \beta$, an instance x is chosen according to D and $\langle x, f(x)\rangle$ is returned to the learner. With probability β, a malicious adversary selects any instance $x \in X$, selects a label $l \in \{0, 1\}$, and returns $\langle x, l\rangle$. Again, the learner's goal is to output an hypothesis h which, with probability at least $1 - \delta$, has error rate $\mathit{error}(h) = \Pr_D[h(x) \neq f(x)]$ at most ϵ.

2.2.3 The Statistical Query Model

In the statistical query model, the example oracle $EX(f, D)$ from the standard PAC model is replaced by a statistics oracle $STAT(f, D)$. An SQ algorithm queries the $STAT$ oracle for the values of various statistics on the distribution of labelled examples (*e.g.* "What is the probability that a randomly chosen labelled example $\langle x, l \rangle$ has variable $x_i = 0$ and $l = 1$?"), and the $STAT$ oracle returns the requested statistics to within some specified additive error. Formally, a statistical query is of the form $[\chi, \tau]$. Here χ is a mapping from labelled examples to $\{0, 1\}$ (*i.e.* $\chi : X \times \{0, 1\} \to \{0, 1\}$) corresponding to an indicator function for those labelled examples about which statistics are to be gathered, while τ is an additive error parameter. A call $[\chi, \tau]$ to $STAT(f, D)$ returns an *estimate* $\hat{P}_\chi$ of $P_\chi = \Pr_D[\chi(x, f(x))]$ which satisfies $|\hat{P}_\chi - P_\chi| \leq \tau$.

A call to $STAT(f, D)$ can be simulated, with high probability, by drawing a sufficiently large sample from $EX(f, D)$ and outputting the fraction of labelled examples which satisfy $\chi(x, f(x))$ as the estimate $\hat{P}_\chi$. Since the required sample size depends polynomially on $1/\tau$ and the simulation time additionally depends on the time required to evaluate χ, an SQ learning algorithm is said to be polynomially efficient if $1/\tau$, the time required to evaluate each χ, and the running time of the SQ algorithm are all bounded by polynomials in $1/\epsilon$, n and $size(f)$. We formally define polynomially efficient learning in the statistical query model as follows (adapted from Kearns [19]):

Definition 2 *(Strong SQ Learning)*
Let $\mathcal{F}$ and $\mathcal{H}$ be function classes defined over X. The class $\mathcal{F}$ is said to be polynomially learnable via statistical queries by $\mathcal{H}$ if there exists a learning algorithm $\mathcal{A}$ and polynomials $p_1(\cdot,\cdot,\cdot)$, $p_2(\cdot,\cdot,\cdot)$, and $p_3(\cdot,\cdot,\cdot)$ such that for any $f \in \mathcal{F}$, for any distribution D on X, and for any error parameter ϵ, $0 < \epsilon \leq 1$, the following holds: if $\mathcal{A}$ is given input ϵ and access to a statistics oracle $STAT(f, D)$, then (1) for every query $[\chi, \tau]$ made by $\mathcal{A}$, χ can be evaluated in time bounded by $p_1(1/\epsilon, n, size(f))$ and $1/\tau$ is bounded by $p_2(1/\epsilon, n, size(f))$, and (2) $\mathcal{A}$ halts in time bounded by $p_3(1/\epsilon, n, size(f))$ and outputs an hypothesis $h \in \mathcal{H}$ that satisfies $error(h) \leq \epsilon$.

For an SQ algorithm $\mathcal{A}$, we may further define its *query complexity* and *tolerance*. In a given instance of learning, the query complexity of $\mathcal{A}$ is the number of queries submitted by $\mathcal{A}$, and the tolerance of $\mathcal{A}$ is the smallest additive error requested by $\mathcal{A}$. We let $N_* = N_*(\epsilon, n, size(f))$ be an upper bound on the query complexity of $\mathcal{A}$, and we let $\tau_* = \tau_*(\epsilon, n, size(f))$ be a lower bound on the tolerance of $\mathcal{A}$. Note that $N_* \leq p_3(1/\epsilon, n, size(f))$ and $\tau_* \geq 1/p_2(1/\epsilon, n, size(f))$.

Since calls to a statistics oracle can be simulated by a procedure which draws a sample from an example oracle, one can view the statistical query model as simply restricting the way in which PAC learning algorithms can make use of labelled examples. Kearns has shown that this restriction is rather weak in that nearly every PAC learning algorithm can be cast in the SQ model.

An important property of this model is that calls to a statistics oracle can also be simulated, with high probability, by a procedure which draws a sample from a *classification noise* oracle $EX_{\mathrm{CN}}^{\eta}(f, D)$ [19] or a *malicious error* oracle $EX_{\mathrm{MAL}}^{\beta}(f, D)$ [9]. In the former case, the sample size required is polynomial in $1/\tau$, $1/(1-2\eta_b)$ and $\log(1/\delta)$; in the latter case, the sample size required is polynomial in $1/\tau$ and $\log(1/\delta)$. While a reasonably efficient simulation of an SQ algorithm can be obtained by drawing a separate sample for each call to the statistics oracle, better bounds on the sample complexity of the simulation are obtained by drawing one large sample and estimating each statistical query using that single sample. If we let $\mathcal{Q}$ be the function space from which an SQ algorithm $\mathcal{A}$ selects its queries, then the size of the single sample required is independent of the query complexity of $\mathcal{A}$ but depends on either the size of $\mathcal{Q}$ or the Vapnik-Chervonenkis dimension[2] of $\mathcal{Q}$. $\mathcal{Q}$ is referred to as the *query space* of the SQ algorithm $\mathcal{A}$.

Kearns has shown that an SQ algorithm can be simulated in the classification noise model using a sample size which depends on $\mathcal{Q}$, τ_*, δ, ϵ and η_b. Decatur has shown that an SQ algorithm can be simulated in the malicious error model using a sample size which depends on $\mathcal{Q}$, τ_* and δ. The amount of malicious error which can be tolerated by the latter simulation depends on τ_*. Given that nearly every PAC learning algorithm can be converted to an SQ algorithm, an immediate consequence of these results is that nearly every PAC algorithm can be transformed into one which tolerates noise. The complexities of these noise-tolerant versions depend on τ_* and $\mathcal{Q}$, which themselves are a function of the *ad hoc* conversion of PAC algorithms to SQ algorithms. Thus, one cannot show general upper bounds on the complexity of these noise-tolerant versions of converted PAC algorithms.

We define weak SQ learning identically to strong SQ learning except that there is no accuracy parameter ϵ. In this case, the output hypothesis need only have error rate slightly less than $1/2$, *i.e.* $\mathit{error}(h) \leq \frac{1}{2} - \gamma = \frac{1}{2} - \frac{1}{p(n, \mathit{size}(f))}$ for some polynomial p. By showing that weak SQ learning algorithms can be "boosted" to strong SQ learning algorithms, we derive general lower bounds

[2] VC-dimension is a standard complexity measure for a space of $\{0,1\}$-valued functions.

on the tolerance of SQ learning and general upper bounds on the complexity of the requisite query space. We are then able to show general upper bounds on the complexity noise-tolerant PAC learning via the statistical query model. These results are given in Chapters 3 and 4.

2.3 Boosting in the PAC Model

In this section, we describe the PAC model boosting results on which our SQ model boosting results are based.

Schapire [28, 29] and Freund [14, 15] use similar strategies for boosting weak learning algorithms to strong learning algorithms. They both create a strong hypothesis by combining many hypotheses obtained from multiple runs of a weak learning algorithm. The boosting schemes derive their power by essentially forcing the weak learning algorithm, in later runs, to approximate the target function f with respect to *new distributions* which "heavily" weight those instances that previous hypotheses misclassify. By suitably constructing example oracles corresponding to these new distributions and properly combining the hypotheses obtained from multiple runs of the weak learning algorithm, a strong learning algorithm can be produced which uses the weak learning algorithm as a subroutine.

Freund has developed two similar methods (which we call Scheme 1 and Scheme 2) for boosting weak learning algorithms to strong learning algorithms. One is more efficient with respect to ϵ while the other is more efficient with respect to γ. Freund develops a hybrid scheme more efficient than either Scheme 1 or Scheme 2 by combining these two methods in order to capitalize on the advantages of each. We first describe the two schemes separately and then show how to combine them.

2.3.1 Boosting via Scheme 1 in the PAC Model

Scheme 1 uses a weak learning algorithm to create a set of $k_1 = \frac{1}{2\gamma^2} \ln \frac{1}{\epsilon}$ weak hypotheses and outputs the majority vote of these hypotheses as the strong hypothesis. The weak hypotheses are created by asking the weak learner to approximate f with respect to various modified distributions over the instance space X. The distribution used to generate a given weak hypothesis is based on the performance of the previously generated weak hypotheses. Hypothesis h_1 is created in the usual way by using $EX(f, D)$. For all $i \geq 1$, hypothesis h_{i+1} is created by giving the weak learner access to a *filtered* example oracle $EX(f, D_{i+1})$ defined as follows:

1. Draw a labelled example $\langle x, f(x)\rangle$ from $EX(f, D)$.
2. Compute $h_1(x), \ldots, h_i(x)$.
3. Set r to be the number of hypotheses which agree with f on x.
4. Flip a biased coin with $\Pr[\text{HEAD}] = \alpha_r^i$.
5. If HEAD, then output example $\langle x, f(x)\rangle$, otherwise go to Step 1.

When k weak hypotheses are to be generated, the set of probabilities $\{\alpha_r^i\}$ are fixed according to the following binomial distribution:

$$\alpha_r^i = \begin{cases} 0 \quad \text{if } r > \left\lfloor \frac{k}{2} \right\rfloor \\ \binom{k-i-1}{\lfloor \frac{k}{2} \rfloor - r} \left(\frac{1}{2} + \gamma\right)^{\lfloor \frac{k}{2} \rfloor - r} \left(\frac{1}{2} - \gamma\right)^{\lceil \frac{k}{2} \rceil - i - 1 + r} \\ \quad \text{if } i - \left\lceil \frac{k}{2} \right\rceil + 1 \leq r \leq \left\lfloor \frac{k}{2} \right\rfloor \\ 0 \quad \text{if } r < i - \left\lceil \frac{k}{2} \right\rceil + 1 \end{cases}$$

Freund shows that, with high probability, the majority vote of $h_1, \ldots, h_{k_1}$ has error rate no more than ϵ with respect to D if each h_j has error rate no more than $\frac{1}{2} - \gamma$ with respect to D_j.

One pitfall of this scheme is that the simulation of $EX(f, D_{i+1})$ may need to draw many examples from $EX(f, D)$ before one is output to the weak learner. Let t_i be the probability that an example drawn randomly from $EX(f, D)$ passes through the probabilistic filter which defines $EX(f, D_{i+1})$. Freund observes that if $t_i < c\epsilon^2$ for some constant c, then the majority vote of $h_1, \ldots, h_i$ is already a strong hypothesis. The boosting algorithm can estimate t_i, and if t_i is below the cutoff, the algorithm may halt and output the majority vote of the hypotheses created thus far. The boosting algorithm's time and sample complexity dependence on γ is $\tilde{\Theta}(1/\gamma^2)$, while its dependence on ϵ is $\tilde{O}(1/\epsilon^2)$.[3]

[3]For asymptotically growing functions g, $g > 1$, we define $\tilde{O}(g)$ to mean $O(g \log^c g)$ for some constant $c \geq 0$. For asymptotically shrinking functions g, $0 < g < 1$, we define $\tilde{O}(g)$ to mean $O(g \log^c(1/g))$ for some constant $c \geq 0$. We define $\tilde{\Omega}$ similarly for constants $c \leq 0$. Finally, we define $\tilde{\Theta}$ to mean both $\tilde{O}$ *and* $\tilde{\Omega}$. This asymptotic notation, read "soft-O," "soft-Omega," and "soft-Theta," is convenient for expressing bounds while ignoring lower order factors. It is somewhat different than the standard soft-order notation.

2.3.2 Boosting via Scheme 2 in the PAC Model

Scheme 2 is very similar to Scheme 1. The weak learner is again called many times to provide weak hypotheses with respect to filtered distributions. This method uses $k_2 = 2k_1 = \frac{1}{\gamma^2}\ln\frac{1}{\epsilon}$ weak hypotheses, while the filtered example oracle remains the same. The main difference is the observation that if $t_i < \frac{\epsilon(1-\epsilon)\gamma}{\ln(1/\epsilon)}$, then we may simply use a "fair coin" in place of h_{i+1} and still be guaranteed, with high probability, that the final majority of k_2 hypotheses has error rate no more than ϵ.[4] The boosting algorithm estimates t_i to see if it is below this new threshold. If so, a "fair coin" is used as hypothesis h_{i+1}, and the algorithm proceeds to find a weak hypothesis with respect to the next distribution. The boosting algorithm's time and sample complexity dependence on γ is $\tilde{O}(1/\gamma^3)$, while its dependence on ϵ is $\tilde{\Theta}(1/\epsilon)$.

2.3.3 Hybrid Boosting in the PAC Model

An improvement on these two boosting schemes is realized by using each in the "boosting range" for which it is most efficient. The first method is more efficient in $1/\gamma$, while the second method is more efficient in $1/\epsilon$. We therefore use the first method to boost from $\frac{1}{2} - \gamma$ to a constant, and we use the second method to boost from that constant to ϵ. Let $\mathcal{A}_{\frac{1}{4}}$ be a learning algorithm which uses Scheme 1 and makes calls to the weak learning algorithm $\mathcal{A}_{\frac{1}{2}-\gamma}$. The strong learning algorithm $\mathcal{A}_\epsilon$ uses Scheme 2 and makes calls to $\mathcal{A}_{\frac{1}{4}}$ as its "weak learner." The strong hypothesis output by such a hybrid algorithm is a depth two circuit with a majority gate at the top level. The inputs to the top level are "fair coin" hypotheses and majority gates whose inputs are weak hypotheses with respect to various distributions. The hybrid's time and sample complexity dependence on γ is $\tilde{\Theta}(1/\gamma^2)$, while its dependence on ϵ is $\tilde{\Theta}(1/\epsilon)$.

[4] A "fair coin" hypothesis ignores its input x and outputs the outcome of a fair coin flip.

Chapter 3

Learning Results in the Additive Error SQ Model

In this chapter, we derive a number of results in the additive error statistical query model. We begin by showing that it possible to boost weak learning algorithms in the SQ model, and from this we derive general bounds on learning in the SQ model. We then describe a new method for simulating SQ algorithms in the PAC model with classification noise. Finally, by combining the aforementioned results, we derive general bounds on PAC learning in the presence of classification noise which apply to all function classes known to be SQ learnable.

3.1 Boosting in the Statistical Query Model

Boosting is accomplished by forcing a weak learning algorithm to approximate the target function f with respect to modified distributions over the instance space. Specifically, the boosting methods described in the previous chapter are based on the observation that, with high probability, the majority vote of $h_1, \ldots, h_k$ has error rate at most ϵ with respect to D if each constituent h_j has error rate at most $\frac{1}{2} - \gamma$ with respect to D_j. In the PAC model, a learner interacts with the distribution over the instance space through calls to an example oracle. Therefore, boosting in the PAC model is accomplished by constructing $EX(f, D_j)$ from the original example oracle $EX(f, D)$. In the SQ model, a learner interacts with the distribution over labelled examples through calls to a statistics oracle. Therefore, boosting in the SQ model is accomplished by constructing $STAT(f, D_j)$ from the original statistics oracle $STAT(f, D)$.

In the sections that follow, we first show how to boost a weak SQ algorithm using either Scheme 1 or Scheme 2. We then show how to boost a weak SQ algorithm using the hybrid method. Although it is possible to boost in the SQ model using Schapire's method, we do not describe these results since they are somewhat weaker than those presented here.

3.1.1 Boosting via Scheme 1 in the Statistical Query Model

We can use Scheme 1 to boost weak SQ learning algorithms by simply answering statistical queries made with respect to modified distributions. Therefore, we must be able to simulate queries to $STAT(f, D_j)$ by making queries to $STAT(f, D)$. We first show how to specify the exact value of a query with respect to D_j in terms of queries with respect to D. We then determine the accuracy with which we need to make these queries with respect to D in order to obtain a sufficient accuracy with respect to D_j.

The modified distributions required for boosting are embodied in the five step description of the filtered example oracle given in Section 2.3.1. Note that Steps 2 and 3 partition the instance space into $i+1$ regions corresponding to those instances which are correctly classified by the same number of hypotheses. Let $X_r^i \subseteq X$ be the set of instances which are correctly classified by exactly r of the i hypotheses. We define the *induced distribution* D_Z on a set Z with respect to distribution D as follows: For any $Y \subseteq Z$, $D_Z[Y] = D[Y]/D[Z]$. By construction, for any given X_r^i region, the filtered example oracle uniformly scales the probability with which examples from that region are drawn. Therefore, the induced distribution on X_r^i with respect to D_{i+1} is the same as the induced distribution on X_r^i with respect to D. (This fact is used to obtain Equation 3.2 from Equation 3.1 below.)

A query $[\chi, \tau]$ to $STAT(f, D_{i+1})$ is a call for an estimate of $\Pr_{D_{i+1}}[\chi(x, f(x))]$ within additive error τ. We derive an expression for $\Pr_{D_{i+1}}[\chi(x, f(x))]$ below.

$$\begin{aligned}
\Pr_{D_{i+1}}[\chi(x, f(x))] &= \sum_{r=0}^{i} \Pr_{D_{i+1}}[\chi(x, f(x))|(x \in X_r^i)] \cdot \Pr_{D_{i+1}}[x \in X_r^i] && (3.1)\\
&= \sum_{r=0}^{i} \Pr_{D}[\chi(x, f(x))|(x \in X_r^i)] \cdot \Pr_{D_{i+1}}[x \in X_r^i] && (3.2)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{r=0}^{i} \frac{\Pr_D[\chi(x, f(x)) \wedge (x \in X_r^i)]}{\Pr_D[x \in X_r^i]} \cdot \frac{\alpha_r^i \cdot \Pr_D[x \in X_r^i]}{\sum_{j=0}^{i} \alpha_j^i \cdot \Pr_D[x \in X_j^i]} \\
&= \frac{\sum_{r=0}^{i} \alpha_r^i \cdot \Pr_D[\chi(x, f(x)) \wedge (x \in X_r^i)]}{\sum_{j=0}^{i} \alpha_j^i \cdot \Pr_D[x \in X_j^i]} \qquad (3.3)
\end{aligned}
$$

Note that the denominator of Equation 3.3 is the probability that an example drawn randomly from $EX(f, D)$ passes through the probabilistic filter which defines $EX(f, D_{i+1})$. Recall that Freund calls this probability t_i.

Ignoring the additive error parameter for the moment, the probabilities in Equation 3.3 can be stated as queries to $STAT(f, D)$ as follows

$$
STAT(f, D_{i+1})[\chi] = \frac{\sum_{j=0}^{i} \alpha_j^i \cdot STAT(f, D)[\chi \wedge \chi_j^i]}{\sum_{j=0}^{i} \alpha_j^i \cdot STAT(f, D)[\chi_j^i]} \qquad (3.4)
$$

where $\chi_j^i(x, l)$ is true if and only if $x \in X_j^i$. Note that query χ_j^i is polynomially evaluatable given $h_1, \ldots, h_i$, thus satisfying the efficiency condition given in the definition of SQ learning.

We next determine the accuracy with which we must ask these queries so that the final result is within the desired additive error τ. We make use of the following two claims.

Claim 1 *If $0 \leq a, b, c, \tau \leq 1$ and $a = b/c$, then to obtain an estimate of a within additive error τ, it is sufficient to obtain estimates of b and c within additive error $c\tau/3$.*

Proof: We must show that $(b + c\tau/3)/(c - c\tau/3) \leq a + \tau$ and $(b - c\tau/3)/(c + c\tau/3) \geq a - \tau$. The claim is proven as follows.

$$
\begin{aligned}
\frac{b + c\tau/3}{c - c\tau/3} &= \frac{a + \tau/3}{1 - \tau/3} \\
&= (a + \tau/3)\left(1 + \frac{\tau/3}{1 - \tau/3}\right) \\
&\leq (a + \tau/3)\left(1 + \frac{\tau/3}{1 - 1/3}\right) \\
&= (a + \tau/3)(1 + \tau/2) \\
&= a + a\tau/2 + \tau/3 + \tau^2/6 \\
&\leq a + \tau
\end{aligned}
$$

$$\begin{aligned}
\frac{b - c\tau/3}{c + c\tau/3} &= \frac{a - \tau/3}{1 + \tau/3} \\
&= (a - \tau/3)\left(1 - \frac{\tau/3}{1 + \tau/3}\right) \\
&\geq (a - \tau/3)(1 - \tau/3) \\
&= a - a\tau/3 - \tau/3 + \tau^2/9 \\
&\geq a - \tau
\end{aligned}$$

□

Claim 2 *If* $0 \leq s, p_i, z_i, \tau \leq 1$, $0 \leq \sum_i p_i \leq 1$ *and* $s = \sum_i p_i z_i$, *then to obtain an estimate of* s *within additive error* τ, *it is sufficient to obtain estimates of each* z_i *within additive error* τ *provided that the* p_i *coefficients are known.*

Proof: The claim follows immediately from the inequalities given below.

$$\sum_i p_i(z_i + \tau) = \sum_i p_i z_i + \tau \sum_i p_i \leq s + \tau$$

$$\sum_i p_i(z_i - \tau) = \sum_i p_i z_i - \tau \sum_i p_i \geq s - \tau$$

□

Applying Claims 1 and 2 to Equation 3.4, we find that it is sufficient to submit queries to $STAT(f, D)$ with additive error $t_i \cdot \tau/3$ in order to simulate a call to $STAT(f, D_{i+1})$ with additive error τ. There are two problems with this observation. First, if t_i is small, then we are forced to submit queries with small additive error. Second, the value t_i is unknown, and in fact, it is the value of the denominator we are attempting to estimate. We can overcome these difficulties by employing the "abort" condition of Freund which allows us to either lower bound t_i or abort the search for h_{i+1}.

If $t_i < c\epsilon^2$, then the majority vote of the hypotheses generated thus far is a strong hypothesis. Submit each query to $STAT(f, D)$ with additive error $\frac{c\epsilon^2}{2+3/\tau}$. Let $\hat{t}_i$ be the estimate for t_i obtained, and note that by Claim 2, $\hat{t}_i$ is within additive error $\frac{c\epsilon^2}{2+3/\tau}$ of t_i. If $\hat{t}_i < c\epsilon^2(1 - \frac{1}{2+3/\tau})$, then $t_i < c\epsilon^2$. In this case, we may halt and output the majority vote of the hypotheses created thus far. If $\hat{t}_i \geq c\epsilon^2(1 - \frac{1}{2+3/\tau})$, then $t_i \geq c\epsilon^2(1 - \frac{2}{2+3/\tau}) = c\epsilon^2(\frac{3/\tau}{2+3/\tau})$. In this case, our estimate $\hat{t}_i$ is sufficiently accurate since the additive error required by Claim 1 is $t_i \cdot \tau/3$, and

$t_i \cdot \tau/3 \geq c\epsilon^2(\frac{3/\tau}{2+3/\tau}) \cdot \tau/3 = \frac{c\epsilon^2}{2+3/\tau}$ which is the additive error used. Given that the numerator and denominator are both estimated with additive error $t_i \cdot \tau/3$, their ratio is within additive error τ by Claim 1.

We can now bound the tolerance of strong SQ learning algorithms obtained by Scheme 1 boosting. If $\tau_0 = \tau_0(n, size(f))$ is a lower bound on the tolerance of a weak SQ learning algorithm, then $\Omega(\tau_0\epsilon^2)$ is a lower bound on the tolerance of the strong SQ learning algorithm obtained by Scheme 1 boosting.

We next examine the query complexity of strong SQ learning algorithms obtained by Scheme 1 boosting. Let $N_0 = N_0(n, size(f))$ be an upper bound on the query complexity of a weak learner. In Equation 3.4, we note that $2(i+1)$ queries to $STAT(f, D)$ are required to simulate a single query to $STAT(f, D_{i+1})$. Since $k_1 = \frac{1}{2\gamma^2}\ln\frac{1}{\epsilon}$ is an upper bound on the number of weak learners run in the boosting scheme, $O(N_0{k_1}^2) = O(N_0\frac{1}{\gamma^4}\log^2\frac{1}{\epsilon})$ is an upper bound on the query complexity of the strong SQ learning algorithm obtained by Scheme 1 boosting.

We finally examine the query space complexity of strong SQ learning algorithms obtained by Scheme 1 boosting. There are two cases to consider depending on the nature of the instance space. If the instance space is *discrete*, *e.g.* the Boolean hypercube $\{0,1\}^n$, then the query space and hypothesis class used by an SQ algorithm are generally finite. In this case, we can bound the *size* of the query space used by the strong SQ learning algorithm obtained by boosting, and this result is given below. If the instance space is *continuous*, *e.g.* n-dimensional Euclidean space $\Re^n$, then the query space and hypothesis class used by an SQ algorithm are generally infinite. In this case, we can bound the VC-dimension of the query space used by the strong SQ learning algorithm obtained by boosting, and this result is given in the appendix.

Let $\mathcal{Q}_0$ and $\mathcal{H}_0$ be the finite query space and finite hypothesis class used by a weak SQ learning algorithm. The queries used by the strong SQ learning algorithm obtained by Scheme 1 boosting are of the form χ, χ_j^i and $\chi \wedge \chi_j^i$ where $\chi \in \mathcal{Q}_0$ and χ_j^i is constructed from hypotheses in $\mathcal{H}_0$. The queries χ_j^i are defined by i hypotheses and a number j, $0 \leq j \leq i$. Since the hypotheses need not be distinct, for fixed i and j, the number of unique χ_j^i queries is $\binom{|\mathcal{H}_0|+i-1}{i}$.[1] For fixed i, the number of χ_j^i queries is $(i+1) \cdot \binom{|\mathcal{H}_0|+i-1}{i}$. Since i is bounded by k_1, the total number of

[1]This expression corresponds to the number of unique arrangements of i indistinguishable balls in $|H_0|$ bins. Each unique arrangement corresponds to a unique χ_j^i in that the number of balls in bin ℓ corresponds to the number of copies of the hypothesis associated with bin ℓ used in χ_j^i.

χ_j^i queries is given by $\sum_{i=1}^{k_1}(i+1)\cdot\binom{|\mathcal{H}_0|+i-1}{i}$. Given that $\chi \in \mathcal{Q}_0$, we may bound the size of the query space used by the strong SQ learning algorithm obtained from Scheme 1 boosting as follows:

$$|\mathcal{Q}_B| = |\mathcal{Q}_0| + \sum_{i=1}^{k_1}(i+1)\cdot\binom{|\mathcal{H}_0|+i-1}{i} + |\mathcal{Q}_0|\sum_{i=1}^{k_1}(i+1)\cdot\binom{|\mathcal{H}_0|+i-1}{i}$$

In the appendix, it is shown that this expression has the following closed form:

$$|\mathcal{Q}_B| = (|\mathcal{Q}_0|+1)\binom{|\mathcal{H}_0|+k_1}{k_1} + |\mathcal{H}_0|(|\mathcal{Q}_0|+1)\binom{|\mathcal{H}_0|+k_1}{k_1-1} - 1$$

Furthermore, it is shown the $|\mathcal{Q}_B|$ is bounded above as follows:

$$|\mathcal{Q}_B| \le 2(|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^{k_1}$$

The complexity of simulating such an SQ algorithm in the various PAC models will depend on $\log|\mathcal{Q}_B|$. We note that $\log|\mathcal{Q}_B| = O(\log|\mathcal{Q}_0| + k_1\log|\mathcal{H}_0|)$.

Finally, in the appendix it is shown that the VC-dimension of the query space is bounded as follows:

$$VC(\mathcal{Q}_B) = O(\,VC(\mathcal{Q}_0) + VC(\mathcal{H}_0)\cdot k_1\log k_1)$$

Theorem 1 *Given a weak SQ learning algorithm whose query complexity is upper bounded by $N_0 = N_0(n, size(f))$, whose tolerance is lower bounded by $\tau_0 = \tau_0(n, size(f))$, whose query space and hypothesis class are $\mathcal{Q}_0$ and $\mathcal{H}_0$, respectively, and whose output hypothesis has error rate at most $\frac{1}{2}-\gamma$, then a strong SQ learning algorithm can be constructed whose query complexity is $O(N_0\frac{1}{\gamma^4}\log^2\frac{1}{\epsilon})$ and whose tolerance is $\Omega(\tau_0\epsilon^2)$. The query space complexity is given by*

$$\log|\mathcal{Q}_B| = O(\log|\mathcal{Q}_0| + \tfrac{1}{\gamma^2}\log\tfrac{1}{\epsilon}\log|\mathcal{H}_0|)$$

when $\mathcal{Q}_0$ and $\mathcal{H}_0$ are finite, or

$$VC(\mathcal{Q}_B) = O(\,VC(\mathcal{Q}_0) + VC(\mathcal{H}_0)\cdot(\tfrac{1}{\gamma^2}\log\tfrac{1}{\epsilon})\log(\tfrac{1}{\gamma^2}\log\tfrac{1}{\epsilon}))$$

when $\mathcal{Q}_0$ and $\mathcal{H}_0$ have finite VC-dimension.

3.1.2 Boosting via Scheme 2 in the Statistical Query Model

We can use Scheme 2 to boost weak SQ learning algorithms in a manner quite similar to that described above. Since the “abort” condition of Scheme 2 introduces “fair coin” hypotheses, we first rederive the probability that $\chi(x, f(x))$ is true with respect to D_{i+1} in terms of probabilities with respect to D.

When i hypotheses have been generated, let w be the number of weak hypotheses and let $i - w$ be the number of "fair coin" hypotheses. The weak hypotheses $h_1, \ldots, h_w$ partition the instance space X into $w + 1$ regions corresponding to those instances which are correctly classified by the same number of weak hypotheses. Let $X_r^w \subseteq X$ be the set of instances which are correctly classified by exactly r of the w weak hypotheses. Consider the probability that an instance $x \in X_r^w$ passes through the probabilistic filter which defines $EX(f, D_{i+1})$. If none of the "fair coin" hypotheses agree with f, then this probability is α_r^i. If j of the "fair coin" hypotheses agree with f, the this probability is α_{r+j}^i. The total probability is thus $\lambda_r^w = \sum_{j=0}^{i-w} \alpha_{r+j}^i \beta_j^{i-w}$ where $\beta_j^{i-w} = \binom{i-w}{j} / 2^{i-w}$ is the probability that exactly j of the "fair coin" hypotheses agree with f. The following filtered example oracle is equivalent to $EX(f, D_{i+1})$:

1. Draw a labelled example $\langle x, f(x) \rangle$ from $EX(f, D)$.
2. Compute $h_1(x), \ldots, h_w(x)$.
3. Set r to be the number of hypotheses which agree with f on x.
4. Flip a biased coin with $\Pr[\text{HEAD}] = \lambda_r^w$.
5. If HEAD, then output example $\langle x, f(x) \rangle$, otherwise go to Step 1.

We may now derive an expression for $\Pr_{D_{i+1}}[\chi(x, f(x))]$ as before.

$$
\begin{aligned}
\Pr_{D_{i+1}}[\chi(x, f(x))] &= \sum_{r=0}^{w} \Pr_{D_{i+1}}[\chi(x, f(x)) | (x \in X_r^w)] \cdot \Pr_{D_{i+1}}[x \in X_r^w] \\
&= \sum_{r=0}^{w} \Pr_D[\chi(x, f(x)) | (x \in X_r^w)] \cdot \Pr_{D_{i+1}}[x \in X_r^w] \\
&= \sum_{r=0}^{w} \frac{\Pr_D[\chi(x, f(x)) \wedge (x \in X_r^w)]}{\Pr_D[x \in X_r^w]} \cdot \frac{\lambda_r^w \cdot \Pr_D[x \in X_r^w]}{\sum_{j=0}^{w} \lambda_j^w \cdot \Pr_D[x \in X_j^w]} \\
&= \frac{\sum_{r=0}^{w} \lambda_r^w \cdot \Pr_D[\chi(x, f(x)) \wedge (x \in X_r^w)]}{\sum_{j=0}^{w} \lambda_j^w \cdot \Pr_D[x \in X_j^w]} \qquad (3.5) \\
STAT(f, D_{i+1})[\chi] &= \frac{\sum_{j=0}^{w} \lambda_j^w \cdot STAT(f, D)[\chi \wedge \chi_j^w]}{\sum_{j=0}^{w} \lambda_j^w \cdot STAT(f, D)[\chi_j^w]} \qquad (3.6)
\end{aligned}
$$

Note that the denominators of Equations 3.5 and 3.6 again correspond to the probability t_i. Also note that $\sum_{r=0}^{w} \lambda_r^w = \sum_{r=0}^{w} \sum_{j=0}^{i-w} \alpha_{r+j}^i \beta_j^{i-w} \leq 1$ since the unique terms of the latter sum are all contained in the product $(\sum_{r=0}^{i} \alpha_r^i)(\sum_{j=0}^{i-w} \beta_j^{i-w}) = 1$.

Applying Claims 1 and 2 to Equation 3.6, we again find that it is sufficient to submit queries to $STAT(f, D)$ with additive error $t_i \cdot \tau/3$ in order to simulate a call to $STAT(f, D_{i+1})$ with additive error τ. Again, there are two problems with this observation. First, if t_i is small, then we are forced to submit queries with small additive error. Second, the value t_i is unknown, and in fact, it is the value of the denominator we are attempting to estimate. We can overcome these difficulties by employing the "abort" condition of Freund which allows us to either lower bound t_i or use a "fair coin" in place of h_{i+1}.

If $t_i < \epsilon(1-\epsilon)\gamma/\ln(1/\epsilon)$, then a "fair coin" can be used in place of h_{i+1}. Submit each query to $STAT(f, D)$ with additive error $\frac{\epsilon(1-\epsilon)\gamma/\ln(1/\epsilon)}{2+3/\tau}$. Let $\hat{t}_i$ be the estimate for the t_i obtained, and note that by Claim 2, $\hat{t}_i$ is within additive error $\frac{\epsilon(1-\epsilon)\gamma/\ln(1/\epsilon)}{2+3/\tau}$ of t_i. If $\hat{t}_i < \frac{\epsilon(1-\epsilon)\gamma}{\ln(1/\epsilon)}(1 - \frac{1}{2+3/\tau})$, then $t_i < \epsilon(1-\epsilon)\gamma/\ln(1/\epsilon)$. In this case, we may use a "fair coin" in place h_{i+1} and proceed to the next distribution. If $\hat{t}_i \geq \frac{\epsilon(1-\epsilon)\gamma}{\ln(1/\epsilon)}(1 - \frac{1}{2+3/\tau})$, then $t_i \geq \frac{\epsilon(1-\epsilon)\gamma}{\ln(1/\epsilon)}(1 - \frac{2}{2+3/\tau}) = \frac{\epsilon(1-\epsilon)\gamma}{\ln(1/\epsilon)}(\frac{3/\tau}{2+3/\tau})$. In this case, our estimate $\hat{t}_i$ is sufficiently accurate since the additive error required by Claim 1 is $t_i \cdot \tau/3$, and $t_i \cdot \tau/3 \geq \frac{\epsilon(1-\epsilon)\gamma}{\ln(1/\epsilon)}(\frac{3/\tau}{2+3/\tau}) \cdot \tau/3 = \frac{\epsilon(1-\epsilon)\gamma/\ln(1/\epsilon)}{2+3/\tau}$ which is the additive error used. Given that the numerator and denominator are both estimated with additive error $t_i \cdot \tau/3$, their ratio is within additive error τ by Claim 1.

We can now bound the tolerance of strong SQ learning algorithms obtained by Scheme 2 boosting. If $\tau_0 = \tau_0(n, \mathit{size}(f))$ is a lower bound on the tolerance of a weak SQ learning algorithm, then $\Omega(\tau_0 \epsilon\gamma/\log(1/\epsilon))$ is a lower bound on the tolerance of the strong SQ learning algorithm obtained by Scheme 2 boosting.

We next examine the query complexity of strong SQ learning algorithms obtained by Scheme 2 boosting. Let $N_0 = N_0(n, \mathit{size}(f))$ be an upper bound on the query complexity of a weak learner. In Equation 3.6, we note that $2(w+1) \leq 2(i+1)$ queries to $STAT(f, D)$ are required to simulate a single query to $STAT(f, D_{i+1})$. Since $k_2 = \frac{1}{\gamma^2} \ln \frac{1}{\epsilon}$ is an upper bound on the number of weak learners run in the boosting scheme, $O(N_0 {k_2}^2) = O(N_0 \frac{1}{\gamma^4} \log^2 \frac{1}{\epsilon})$ is an upper bound on the query complexity of the strong SQ learning algorithm obtained by Scheme 2 boosting.

We finally note that the query space complexity results for Scheme 2 boosting are identical to those for Scheme 1 boosting when k_1 is replace by k_2.

Theorem 2 *Given a weak SQ learning algorithm whose query complexity is upper bounded by $N_0 = N_0(n, size(f))$, whose tolerance is lower bounded by $\tau_0 = \tau_0(n, size(f))$, whose query space and hypothesis class are $\mathcal{Q}_0$ and $\mathcal{H}_0$, respectively, and whose output hypothesis has error rate at most $\frac{1}{2} - \gamma$, then a strong SQ learning algorithm can be constructed whose query complexity is $O(N_0 \frac{1}{\gamma^4} \log^2 \frac{1}{\epsilon})$ and whose tolerance is $\Omega(\tau_0 \epsilon\gamma / \log(1/\epsilon))$. The query space complexity is given by*

$$\log |\mathcal{Q}_B| = O(\log |\mathcal{Q}_0| + \tfrac{1}{\gamma^2} \log \tfrac{1}{\epsilon} \log |\mathcal{H}_0|)$$

when $\mathcal{Q}_0$ and $\mathcal{H}_0$ are finite, or

$$VC(\mathcal{Q}_B) = O(VC(\mathcal{Q}_0) + VC(\mathcal{H}_0) \cdot (\tfrac{1}{\gamma^2} \log \tfrac{1}{\epsilon}) \log(\tfrac{1}{\gamma^2} \log \tfrac{1}{\epsilon}))$$

when $\mathcal{Q}_0$ and $\mathcal{H}_0$ have finite VC-dimension.

3.1.3 Hybrid Boosting in the Statistical Query Model

We obtain a more efficient boosting scheme in the SQ model by combining the two previously described methods. As in the PAC model, we use Scheme 1 to boost from $\frac{1}{2} - \gamma$ to $\frac{1}{4}$ and Scheme 2 to boost from $\frac{1}{4}$ to ϵ. By combining the results of Theorem 1 and Theorem 2, we immediately obtain an upper bound on the query complexity of the hybrid boosting scheme and a lower bound on the tolerance of the hybrid boosting scheme. An upper bound on the query space complexity of the hybrid boosting scheme is given in the appendix. We thus obtain the following improved boosting result.

Theorem 3 *Given a weak SQ learning algorithm whose query complexity is upper bounded by $N_0 = N_0(n, size(f))$, whose tolerance is lower bounded by $\tau_0 = \tau_0(n, size(f))$, whose query space and hypothesis class are $\mathcal{Q}_0$ and $\mathcal{H}_0$, respectively, and whose output hypothesis has error rate at most $\frac{1}{2} - \gamma$, then a strong SQ learning algorithm can be constructed whose query complexity is $O(N_0 \frac{1}{\gamma^4} \log^2 \frac{1}{\epsilon})$ and whose tolerance is $\Omega(\tau_0 \epsilon / \log(1/\epsilon))$. The query space complexity is given by*

$$\log |\mathcal{Q}_{HB}| = O(\log |\mathcal{Q}_0| + \tfrac{1}{\gamma^2} \log \tfrac{1}{\epsilon} \log |\mathcal{H}_0|)$$

when $\mathcal{Q}_0$ and $\mathcal{H}_0$ are finite, or

$$VC(\mathcal{Q}_{HB}) = O(\,VC(\mathcal{Q}_0) + VC(\mathcal{H}_0) \cdot (\tfrac{1}{\gamma^2} \log \tfrac{1}{\epsilon}) \log(\tfrac{1}{\gamma^2} \log \tfrac{1}{\epsilon}))$$

when $\mathcal{Q}_0$ and $\mathcal{H}_0$ have finite VC-dimension.

Note that the tolerance of the strong SQ learning algorithm constructed has no dependence on γ in this hybrid boosting scheme.

3.2 General Bounds on Learning in the Statistical Query Model

In this section, we derive general upper bounds on the complexity of statistical query learning. These results are obtained by applying the boosting results of the previous section. We further show that our general upper bounds are nearly optimal by demonstrating the existence of a function class whose minimum learning complexity nearly matches our general upper bounds.

3.2.1 General Upper Bounds on Learning in the SQ Model

Just as the sample complexity of boosting in the PAC model yields general upper bounds on the sample complexity of strong PAC learning, the query, query space and tolerance complexities of boosting in the SQ model yield general bounds on the query, query space and tolerance complexities of strong SQ learning.

We can convert any strong SQ learning algorithm into a weak SQ learning algorithm by "hardwiring" the accuracy parameter ϵ to a constant. We can then boost this learning algorithm, via Scheme 2 for instance, to obtain a strong SQ learning algorithm whose dependence on ϵ is nearly optimal.

Theorem 4 *If the class $\mathcal{F}$ is strongly SQ learnable, then $\mathcal{F}$ is strongly SQ learnable by an algorithm whose query complexity is $O(N_0 \log^2 \frac{1}{\epsilon})$, whose tolerance is $\Omega(\tau_0 \epsilon / \log(1/\epsilon))$, and whose query space complexity is $O(p_3(n) \log \frac{1}{\epsilon})$ when the query space is finite or $O(p_4(n) \log \frac{1}{\epsilon} \log\log \frac{1}{\epsilon})$ when the query space has finite VC-dimension, where $N_0 = p_1(n, size(f))$, $\tau_0 = 1/p_2(n, size(f))$ and p_1, p_2, p_3 and p_4 are polynomials.*

While we have focused primarily on the query, query space and tolerance complexities of SQ learning, we note that our boosting results can also be applied to bound the time, space and hypothesis size complexities of SQ learning. It is easily shown that, with respect to ϵ, these complexities are bounded by $O(\log^2 \frac{1}{\epsilon})$, $O(\log \frac{1}{\epsilon})$ and $O(\log \frac{1}{\epsilon})$, respectively.

For any function class of VC-dimension d, Kearns [19] has shown that learning in the SQ model requires $\Omega(d/\log d)$ queries each with additive error $O(\epsilon)$. Whereas Kearns simultaneously lower bounds the query complexity and upper bounds the tolerance, we have simultaneously upper bounded the query complexity and lower bounded the tolerance. Note that the tolerance we give in Theorem 4 is optimal to within a logarithmic factor. While Kearns' general lower bound leaves open the possibility that there may exist a general upper bound on the query complexity which is independent of ϵ, we show that this is not the case by demonstrating a specific learning problem which requires $\Omega(\frac{d}{\log d} \log \frac{1}{\epsilon})$ queries each with additive error $O(\epsilon)$ in the SQ model. Thus, with respect to ϵ, our general upper bound on query complexity is within a $\log \frac{1}{\epsilon}$ factor of the best possible general upper bound.

3.2.2 A Specific Lower Bound for Learning in the SQ Model

In this section, we describe a function class whose minimum learning complexity nearly matches our general upper bounds. We begin by introducing a game on which our learning problem is based.

Consider the following two player game parameterized by t, d and N where $t \leq d \leq N$. The *adversary* chooses a set[2] $S \subseteq [N]$ of size d, and the goal of the *player* is to output a set $T \subseteq [N]$ such that $|S \triangle T| \leq t$. The player is allowed to ask *queries* of the form $Q \subseteq [N]$ to which the adversary returns $|Q \cap S|$.

Lemma 1 *For any $d \geq 4$, $t \leq d/4$ and $N = \Omega(d^{1+\alpha})$ for some $\alpha > 0$, the player requires $\Omega(\frac{d}{\log d} \log N)$ queries of the oracle, in the worst case.*

Proof: Any legal adversary must return responses to the given queries which are consistent with some set $S \subseteq [N]$ of size d. We construct an adaptive, malicious adversary which works as follows. Let $\mathcal{S}_0 \subset 2^{[N]}$ be the set of all $\binom{N}{d}$ subsets of size d. When the player presents the first query $Q_1 \subseteq [N]$, the adversary calculates the value of $|S \cap Q_1|$ for every $S \in \mathcal{S}_0$ and partitions the set $\mathcal{S}_0$ into $d+1$ sets $\mathcal{S}_0^0, \mathcal{S}_0^1, \ldots, \mathcal{S}_0^d$ where each subset $S \in \mathcal{S}_0^i$ has $|S \cap Q_1| = i$.

[2] We use the standard combinatorial notation $[N] = \{1, \ldots, N\}$.

For $i = \arg\max_j \{|\mathcal{S}_0^j|\}$, the adversary returns the value i and lets $\mathcal{S}_1 = \mathcal{S}_0^i$. In general, $\mathcal{S}_k$ is the set of remaining subsets which are consistent with the responses given to the first k queries, and the adversary answers each query so as to maximize the remaining number of subsets. Note that $|\mathcal{S}_k| \geq |\mathcal{S}_0|/(d+1)^k = \binom{N}{d}/(d+1)^k$.

For any $\mathcal{S} \subseteq 2^{[N]}$, we define $\mathit{width}(\mathcal{S}) = \max_{S_i, S_j \in \mathcal{S}} \{|S_i \triangle S_j|\}$. Note that if $\mathit{width}(\mathcal{S}_k) > 2t$, then there exist at least two sets $S_1, S_2 \in \mathcal{S}_k$ such that $|S_1 \triangle S_2| > 2t$. This implies that there cannot exist a set T which satisfies both $|S_1 \triangle T| \leq t$ and $|S_2 \triangle T| \leq t$ (since $\triangle$ is a metric over the space of sets which satisfies the triangle inequality property). If the player were to stop and output a set T at this point, then the malicious adversary could always force the player to lose. We now bound $\mathit{width}(\mathcal{S}_k)$ as a function of $|\mathcal{S}_k|$. This, combined with our bound on $|\mathcal{S}_k|$ as a function of k, will effectively bound the minimum number of queries required by the player. We make use of the following inequalities:[3]

$$\left(\frac{n}{r}\right)^r \leq \binom{n}{r} \leq \left(\!\!\binom{n}{r}\!\!\right) \leq \left(\frac{en}{r}\right)^r$$

For any $\mathcal{S} \subseteq 2^{[N]}$ of width at most w, one can easily show that $|\mathcal{S}| \leq \left(\!\binom{N}{w}\!\right)$. Thus, if $|\mathcal{S}_k| > \left(\!\binom{N}{2t}\!\right)$, then $\mathit{width}(\mathcal{S}_k) > 2t$. We now note that any k which satisfies the following inequality will guarantee that $\mathit{width}(\mathcal{S}_k) > 2t$:

$$\left(\!\!\binom{N}{2t}\!\!\right) \leq \left(\!\!\binom{N}{d/2}\!\!\right) \leq \left(\frac{eN}{d/2}\right)^{d/2} < \frac{\left(\frac{N}{d}\right)^d}{(d+1)^k} \leq \frac{\binom{N}{d}}{(d+1)^k} \leq |\mathcal{S}_k|$$

Solving the third inequality for $(d+1)^k$, we obtain:

$$(d+1)^k < \left(\frac{N}{d}\right)^d \left(\frac{d/2}{eN}\right)^{d/2} = \left(\frac{N}{2ed}\right)^{d/2}$$

Thus, a lower bound on the number of queries required by the player is

$$\frac{\frac{d}{2}\log\frac{N}{2ed}}{\log(d+1)} = \Omega\left(\frac{d}{\log d}\log N\right)$$

for $N = \Omega(d^{1+\alpha})$. □

[3] We use the standard combinatorial notation $\left(\!\binom{n}{r}\!\right) = \sum_{i=0}^{r}\binom{n}{i}$.

Now consider a learning problem defined as follows. Our instance space X is the set of natural numbers $\mathcal{N}$, and our function class is the set of all indicator functions corresponding to subsets of $\mathcal{N}$ of size d. This function class is easily learnable in the SQ model. In what follows, we show that any deterministic SQ algorithm for this class requires $\Omega(\frac{d}{\log d} \log \frac{1}{\epsilon})$ queries with additive error $O(\epsilon)$.

Theorem 5 *There exists a parameterized family of function classes which requires $\Omega(\frac{d}{\log d} \log \frac{1}{\epsilon})$ queries with additive error $O(\epsilon)$ to learn in the SQ model.*

Proof: Consider the two-player game as defined above. For an instance of the game specified by t, d and N (where $d \geq 4$, $t = d/4$ and $N = \Omega(d^{1+\alpha})$), we create an instance of the learning problem as follows. We define our distribution D over $\mathcal{N}$ to place weight $4/Nd$ on each point $1, \ldots, N$ and to place weight $1 - 4/d$ on the point $N + 1$. All other points have zero weight. We set $\epsilon = 1/N$ and call the deterministic SQ learning algorithm. Note that the target subset has weight $4/N$, so if the SQ algorithm submits a query with additive error greater than $4\epsilon = 4/N$ we may answer the query ourselves (as if the target subset were "empty"). For any query χ submitted with tolerance less than 4ϵ, we determine the exact answer as follows. Begin with an answer of 0. If $\chi(N + 1, 0) = 1$, then add $1 - 4/d$ to the answer. Determine the following three subsets of $[N]$: X_1^0, X_1^1 and X_2 where $x \in X_1^0$ if $\chi(x, 0) = 1$ and $\chi(x, 1) = 0$, $x \in X_1^1$ if $\chi(x, 0) = 0$ and $\chi(x, 1) = 1$, and $x \in X_2$ if $\chi(x, 0) = 1$ and $\chi(x, 1) = 1$. Add $|X_2| \cdot 4/Nd$ to the answer. Submit the query X_1^0 to the adversary, and for a response r add $(|X_1^0| - r) \cdot 4/Nd$ to the answer. Submit the query X_1^1 to the adversary, and for a response r add $r \cdot 4/Nd$ to the answer. Return the final value of the answer to the SQ algorithm.

Note that we are able to answer each SQ algorithm query by submitting only two queries to the adversary, and we need not submit any queries to the adversary if the requested additive error is greater than 4ϵ. Since $\Omega(\frac{d}{\log d} \log N)$ queries of the adversary are required, the SQ algorithm must ask $\Omega(\frac{d}{\log d} \log N) = \Omega(\frac{d}{\log d} \log \frac{1}{\epsilon})$ queries with additive error $O(\epsilon)$. $\square$

Using techniques similar to those found in Kearns' lower bound proof [19], the above proof can easily be modified to show that even if the adversary chooses his subset *randomly and uniformly* before the game starts, then there exists some constant probability with which *any* SQ algorithm (deterministic *or* probabilistic) will fail if it asks $o(\frac{d \log(1/\epsilon)}{\log(d \log(1/\epsilon))})$ queries with additive error $O(\epsilon)$. This result is given in the appendix.

3.3 Simulating SQ Algorithms in the Classification Noise Model

In this section, we describe an improved method for efficiently simulating a statistical query algorithm in the classification noise model. The advantages of this new method are twofold. First, our simulation employs a new technique which significantly reduces the running time of simulating SQ algorithms. Second, our formulation for estimating individual queries is simpler and more easily generalized.

Kearns' procedure for simulating SQ algorithms works in the following way. Kearns shows that given a query χ, P_χ can be written as an expression involving the unknown noise rate η and other probabilities which can be estimated from the noisy example oracle $EX^{\eta}_{\mathrm{CN}}(f, D)$. We note that the derivation of this expression relies on χ being $\{0, 1\}$-valued. The actual expression obtained is given below.

$$P_\chi = \frac{1}{1-2\eta} P_\chi^\eta + \left(1 - \frac{1}{1-2\eta}\right) p_2 P_\chi^2 - \frac{\eta}{1-2\eta} p_1 \tag{3.7}$$

In order to estimate P_χ with additive error τ, a sensitivity analysis is employed to determine how accurately each of the components on the right-hand side of Equation 3.7 must be known. Kearns shows that for some constants c_1 and c_2, if η is estimated with additive error $c_1\tau(1-2\eta)^2$ and each of the probabilities is estimated with additive error $c_2\tau(1-2\eta_b)$, then the estimate obtained for P_χ from Equation 3.7 will be sufficiently accurate. Since the value of η is not known, the procedure for simulating SQ algorithms essentially guesses a set of values for η, $\{\eta_0, \eta_1, \ldots, \eta_i\}$, such that at least one η_j satisfies $|\eta_j - \eta| \le c_1\tau_*(1-2\eta)^2$ where τ_* is a lower bound on the tolerance of the SQ algorithm. Since $c_1\tau_*(1-2\eta_b)^2 \le c_1\tau_*(1-2\eta)^2$, the simulation uniformly guesses $\Theta(\frac{1}{\tau_*(1-2\eta_b)^2})$ values of η between 0 and η_b. For each guess of η, the simulation runs a separate copy of the SQ algorithm and estimates the various queries using the formula given above. Since some guess at η was good, at least one of the runs will have produced a good hypothesis with high probability. The various hypotheses are then tested to find a good hypothesis, of which at least one exists. Note that the η-guessing has a significant impact on the running time of the simulation.

In what follows, we show a new derivation of P_χ which is simpler and more easily generalizable than Kearns' original version. We also show that to estimate an individual P_χ, it is only necessary to have an estimate of η within additive error $c\tau(1-2\eta)$ for some constant c. We further show that the number of η-guesses need only be $O(\frac{1}{\tau_*}\log\frac{1}{1-2\eta_b})$, thus significantly reducing the time complexity of the SQ simulation.

3.3.1 A New Derivation for P_χ

In this section, we present a simpler derivation of an expression for P_χ. In previous sections, it was convenient to view a $\{0,1\}$-valued χ as a predicate so that $P_\chi = \Pr_D[\chi(x, f(x))]$. In this section, it will be more convenient to view χ as a function so that $P_\chi = \mathbf{E}_D[\chi(x, f(x))]$. Further, by making no assumptions on the range of χ, the results obtained herein can easily be generalized; these generalizations are discussed in Chapter 5.

Let X be the instance space, and let $Y = X \times \{0,1\}$ be the labelled example space. We consider a number of different examples oracles and the distributions these example oracles impose on the space of labelled examples. For a given target function f and distribution D over X, let $EX(f,D)$ be the standard, noise-free example oracle. In addition, we define the following example oracles: Let $EX(\bar{f},D)$ be the *anti-example oracle*, $EX^\eta_{\mathrm{CN}}(f,D)$ be the *noisy example oracle* and $EX^\eta_{\mathrm{CN}}(\bar{f},D)$ be the *noisy anti-example oracle*. Note that we have access to $EX^\eta_{\mathrm{CN}}(f,D)$ and we can easily construct $EX^\eta_{\mathrm{CN}}(\bar{f},D)$ by simply flipping the label of each example drawn from $EX^\eta_{\mathrm{CN}}(f,D)$.

Each of these oracles imposes a distribution over labelled examples. Let D_f, $D_{\bar{f}}$, D^η_f and $D^\eta_{\bar{f}}$ be these distributions, respectively. Note that $P_\chi = \mathbf{E}_D[\chi(x, f(x))] = \mathbf{E}_{D_f}[\chi]$.

Finally, for a labelled example $y = \langle x, l\rangle$, let $\overline{y} = \langle x, \bar{l}\rangle$. We define $\overline{\chi}(y) = \chi(\overline{y})$. Note that $\overline{\chi}$ is a new function which, on input $\langle x, l\rangle$, simply outputs $\chi(x, \bar{l})$. The function $\overline{\chi}$ is easily constructed from χ.

Theorem 6

$$P_\chi = \mathbf{E}_{D_f}[\chi] = \frac{(1-\eta)\mathbf{E}_{D^\eta_f}[\chi] - \eta\mathbf{E}_{D^\eta_{\bar{f}}}[\overline{\chi}]}{1-2\eta} \tag{3.8}$$

Proof: We begin by relating the various example oracles defined above. Recall that the noisy example oracle $EX^\eta_{\mathrm{CN}}(f,D)$ is defined as follows: Draw an instance $x \in X$ according to D

and output $\langle x, f(x)\rangle$ with probability $1-\eta$ or $\langle x, \neg f(x)\rangle$ with probability η. The draw of x is performed randomly and independently for each call to $EX_{\mathrm{CN}}^{\eta}(f, D)$, and the correct or incorrect labelling of x is performed randomly and independently for each call to $EX_{\mathrm{CN}}^{\eta}(f, D)$. In particular, the correct or incorrect labelling of x is not dependent on the instance x itself.

Given the independence described above, we may equivalently define $EX_{\mathrm{CN}}^{\eta}(f, D)$ (and $EX_{\mathrm{CN}}^{\eta}(\bar{f}, D)$) as follows:

$$EX_{\mathrm{CN}}^{\eta}(f, D) = \begin{cases} EX(f, D) & \textit{with probability } 1-\eta \\ EX(\bar{f}, D) & \textit{with probability } \eta \end{cases}$$

$$EX_{\mathrm{CN}}^{\eta}(\bar{f}, D) = \begin{cases} EX(\bar{f}, D) & \textit{with probability } 1-\eta \\ EX(f, D) & \textit{with probability } \eta \end{cases}$$

We may use these equivalent definitions to deduce the following:

$$\mathbf{E}_{D_f^\eta}[\chi] = (1-\eta)\mathbf{E}_{D_f}[\chi] + \eta\mathbf{E}_{D_{\bar{f}}}[\chi] \tag{3.9}$$

$$\mathbf{E}_{D_{\bar{f}}^\eta}[\chi] = (1-\eta)\mathbf{E}_{D_{\bar{f}}}[\chi] + \eta\mathbf{E}_{D_f}[\chi] \tag{3.10}$$

Multiplying Equation 3.9 by $(1-\eta)$ and Equation 3.10 by η, we obtain:

$$(1-\eta)\mathbf{E}_{D_f^\eta}[\chi] = (1-\eta)^2\mathbf{E}_{D_f}[\chi] + \eta(1-\eta)\mathbf{E}_{D_{\bar{f}}}[\chi] \tag{3.11}$$

$$\eta\mathbf{E}_{D_{\bar{f}}^\eta}[\chi] = \eta(1-\eta)\mathbf{E}_{D_{\bar{f}}}[\chi] + \eta^2\mathbf{E}_{D_f}[\chi] \tag{3.12}$$

Subtracting Equation 3.12 from Equation 3.11 and solving for $\mathbf{E}_{D_f}[\chi]$, we finally obtain:

$$\mathbf{E}_{D_f}[\chi] = \frac{(1-\eta)\mathbf{E}_{D_f^\eta}[\chi] - \eta\mathbf{E}_{D_{\bar{f}}^\eta}[\chi]}{1-2\eta}.$$

To obtain Equation 3.8, we simply note that $\mathbf{E}_{D_{\bar{f}}^\eta}[\chi] = \mathbf{E}_{D_f^\eta}[\overline{\chi}]$. □

Note that in the derivation given above, we have not assumed that χ is $\{0,1\}$-valued. This derivation is quite general and can be applied to estimating the expectations of real-valued queries. This result is given in Chapter 5.

Finally, note that if we define

$$\chi^\eta(y) = \frac{(1-\eta)\chi(y) - \eta\overline{\chi}(y)}{1-2\eta},$$

then $P_\chi = \mathbf{E}_{D_f}[\chi] = \mathbf{E}_{D_f^\eta}[\chi^\eta]$. Thus, given a χ whose expectation we require with respect to the noise-free oracle, we can construct a new χ whose expectation with respect to the noisy oracle is identical to the answer we require. This formulation may even be more convenient if one has the capability of estimating the expectation of real-valued functions; we discuss this generalization in Chapter 5.

3.3.2 Sensitivity Analysis

In this section, we provide a sensitivity analysis of Equation 3.8 in order to determine the accuracy with which various quantities must be estimated. We make use of the following claim.

Claim 3 *If* $0 \le a, b, c, \tau \le 1$ *and* $\{a = b/c,\ a = b \cdot c,\ a = b - c\}$, *then to obtain an estimate of* a *within additive error* τ, *it is sufficient to obtain estimates of* b *and* c *within additive error* $\{c\tau/3,\ \tau(\sqrt{2}-1),\ \tau/2\}$, *respectively.*

Proof: The $a = b/c$ case is proven in Claim 1. The $a = b \cdot c$ case is proven as follows.

$$\begin{aligned}
(b+\tau(\sqrt{2}-1))\cdot(c+\tau(\sqrt{2}-1)) &= b\cdot c + b\tau(\sqrt{2}-1) + c\tau(\sqrt{2}-1) + \tau^2(\sqrt{2}-1)^2 \\
&= a + b\tau(\sqrt{2}-1) + c\tau(\sqrt{2}-1) + \tau^2(3-2\sqrt{2}) \\
&\le a + \tau(\sqrt{2}-1) + \tau(\sqrt{2}-1) + \tau(3-2\sqrt{2}) \\
&= a + \tau \\
(b-\tau(\sqrt{2}-1))\cdot(c-\tau(\sqrt{2}-1)) &= b\cdot c - b\tau(\sqrt{2}-1) - c\tau(\sqrt{2}-1) + \tau^2(\sqrt{2}-1)^2 \\
&= a - b\tau(\sqrt{2}-1) - c\tau(\sqrt{2}-1) + \tau^2(3-2\sqrt{2}) \\
&\ge a - \tau(\sqrt{2}-1) - \tau(\sqrt{2}-1) \\
&\ge a - \tau
\end{aligned}$$

The $a = b - c$ case is trivial. □

Lemma 2 *Let $\hat{\eta}$, $\hat{\mathbf{E}}_{D_f^\eta}[\chi]$ and $\hat{\mathbf{E}}_{D_f^\eta}[\overline{\chi}]$ be estimates of η, $\mathbf{E}_{D_f^\eta}[\chi]$ and $\mathbf{E}_{D_f^\eta}[\chi]$ each within additive error $\tau(1-2\eta)(\sqrt{2}-1)/6$. Then the quantity*

$$\frac{(1-\hat{\eta})\hat{\mathbf{E}}_{D_f^\eta}[\chi] - \hat{\eta}\hat{\mathbf{E}}_{D_f^\eta}[\overline{\chi}]}{1-2\hat{\eta}}$$

is within additive error τ of $P_\chi = \mathbf{E}_{D_f}[\chi]$.

Proof: To obtain an estimate of the right-hand side of Equation 3.8 within additive error τ, it is sufficient to obtain estimates of the numerator and denominator within additive error $(1-2\eta)\tau/3$. This condition holds for the denominator if η is estimated with additive error $(1-2\eta)\tau/6$.

To obtain an estimate of the numerator within additive error $(1-2\eta)\tau/3$, it is sufficient to estimate the summands of the numerator with additive error $(1-2\eta)\tau/6$. Similarly, to obtain accurate estimates of these summands, it is sufficient to estimate η, $\mathbf{E}_{D_f^\eta}[\chi]$ and $\mathbf{E}_{D_f^\eta}[\overline{\chi}]$ each with additive error $(1-2\eta)\tau(\sqrt{2}-1)/6$. $\square$

Estimates for $\mathbf{E}_{D_f^\eta}[\chi]$ and $\mathbf{E}_{D_f^\eta}[\overline{\chi}]$ are obtained by sampling, and an "estimate" for η is obtained by guessing. We address these issues in the following sections.

3.3.3 Estimating $\mathbf{E}_{D_f^\eta}[\chi]$ and $\mathbf{E}_{D_f^\eta}[\overline{\chi}]$

One can estimate the expected values of all queries submitted by drawing separate samples for each of the corresponding χ and $\overline{\chi}$'s and applying Lemma 2. However, better results are obtained by appealing to uniform convergence.

Let $\mathcal{Q}$ be the query space of the SQ algorithm and let $\overline{\mathcal{Q}} = \{\overline{\chi} : \chi \in \mathcal{Q}\}$. The query space of our simulation is $\mathcal{Q}' = \mathcal{Q} \cup \overline{\mathcal{Q}}$. Note that for finite $\mathcal{Q}$, $|\mathcal{Q}'| \le 2|\mathcal{Q}|$. One can further show that for all $\mathcal{Q}$, $VC(\mathcal{Q}') \le c \cdot VC(\mathcal{Q})$ for a constant $c \approx 4.66$. This result is given in the appendix.

If τ_* is a lower bound on the minimum additive error requested by the SQ algorithm and η_b is an upper bound on the noise rate, then by Lemma 2, $(1-2\eta_b)\tau_*(\sqrt{2}-1)/6$ is a sufficient additive error with which to estimate all expectations. Standard uniform convergence results can be applied to show that all expectations can be estimated within the given additive error using a single noisy sample of size

$$m_1 = O\Big(\frac{1}{\tau_*^2(1-2\eta_b)^2}\log\frac{|\mathcal{Q}|}{\delta}\Big)$$

in the case of a finite query space, or a single noisy sample of size

$$m_1 = O\Big(\frac{VC(\mathcal{Q})}{\tau_*^2(1-2\eta_b)^2}\log\frac{1}{\tau_*(1-2\eta_b)} + \frac{1}{\tau_*^2(1-2\eta_b)^2}\log\frac{1}{\delta}\Big)$$

in the case of an infinite query space of finite VC-dimension.

3.3.4 Guessing the Noise Rate η

By Lemma 2, to obtain an estimate for P_χ, it is sufficient to have an estimate of the noise rate η within additive error $(1-2\eta)\tau_*(\sqrt{2}-1)/6$. Since the noise rate is unknown, the simulation guesses various values of the noise rate and runs the SQ algorithm for each guess. If one of the noise rate guesses is sufficiently accurate, then the corresponding run of the SQ algorithm will produce the desired accurate hypothesis.

To guarantee that an accurate η-guess is used, one could simply guess $\Theta(\frac{1}{\tau_*(1-2\eta_b)})$ values of η spaced uniformly between 0 and η_b. This is essentially the approach adopted by Kearns. Note that this would cause the simulation to run the SQ algorithm $\Theta(\frac{1}{\tau_*(1-2\eta_b)})$ times.

We now show that this "branching factor" can be reduced to $O(\frac{1}{\tau_*}\log\frac{1}{1-2\eta_b})$ by constructing our η-guesses in a much better way. The result follows immediately from the following lemma when $\gamma = \tau_*(\sqrt{2}-1)/6$

Lemma 3 *For all $\gamma, \eta_b < 1/2$, there exists a sequence of η-guesses $\{\eta_0, \eta_1, \ldots, \eta_i\}$ where $i = O(\frac{1}{\gamma}\log\frac{1}{1-2\eta_b})$ such that for all $\eta \in [0, \eta_b]$, there exists an η_j which satisfies $|\eta - \eta_j| \le \gamma(1-2\eta)$.*

Proof: The sequence is constructed as follows. Let $\eta_0 = 0$ and consider how to determine η_j from η_{j-1}. The value η_{j-1} is a valid estimate for all $\eta \ge \eta_{j-1}$ which satisfy $\eta - \gamma(1-2\eta) \le \eta_{j-1}$. Solving for η, we find that η_{j-1} is a valid estimate for all $\eta \in [\eta_{j-1}, \frac{\eta_{j-1}+\gamma}{1+2\gamma}]$. Consider an $\eta_j > \frac{\eta_{j-1}+\gamma}{1+2\gamma}$. The value η_j is a valid estimate for all $\eta \le \eta_j$ which satisfy $\eta + \gamma(1-2\eta) \ge \eta_j$. Solving for η, we find that η_j is a valid estimate for all $\eta \in [\frac{\eta_j - \gamma}{1-2\gamma}, \eta_j]$. To ensure that either

η_{j-1} or η_j is a valid estimate for any $\eta \in [\eta_{j-1}, \eta_j]$, we set

$$\frac{\eta_{j-1} + \gamma}{1 + 2\gamma} = \frac{\eta_j - \gamma}{1 - 2\gamma}.$$

Solving for η_j in terms of η_{j-1}, we obtain

$$\eta_j = \frac{1 - 2\gamma}{1 + 2\gamma} \cdot \eta_{j-1} + \frac{2\gamma}{1 + 2\gamma}.$$

Substituting $\gamma' = 2\gamma/(1 + 2\gamma)$, we obtain the following recurrence:

$$\eta_j = (1 - 2\gamma') \cdot \eta_{j-1} + \gamma'$$

Note that if $\gamma < 1/2$, then $\gamma' < 1/2$ as well.

By constructing η-guesses using this recurrence, we ensure that for all $\eta \in [0, \eta_i]$, at least one of $\{\eta_0, \ldots, \eta_i\}$ is a valid estimate. Solving this recurrence, we find that

$$\eta_i = \gamma' \sum_{j=0}^{i-1} (1 - 2\gamma')^j + \eta_0 (1 - 2\gamma')^i.$$

Since $\eta_0 = 0$ and we are only concerned with $\eta \leq \eta_b$, we may bound the number of guesses required by finding the smallest i which satisfies

$$\gamma' \sum_{j=0}^{i-1} (1 - 2\gamma')^j \geq \eta_b.$$

Given that

$$\gamma' \sum_{j=0}^{i-1} (1 - 2\gamma')^j = \gamma' \cdot \frac{1 - (1 - 2\gamma')^i}{1 - (1 - 2\gamma')} = \frac{1 - (1 - 2\gamma')^i}{2}$$

we need $(1 - 2\gamma')^i \leq 1 - 2\eta_b$. Solving for i, we find that any $i \geq \ln \frac{1}{1-2\eta_b} / \ln \frac{1}{1-2\gamma'}$ is sufficient. Using the fact that $1/x > 1/\ln \frac{1}{1-x}$ for all $x \in (0, 1)$, we find that

$$i = \frac{1}{2\gamma'} \cdot \ln \frac{1}{1 - 2\eta_b} = \frac{1 + 2\gamma}{4\gamma} \cdot \ln \frac{1}{1 - 2\eta_b}$$

is an upper bound on the number of guesses required. □

3.3.5 The Overall Simulation

We now combine the results of the previous sections to obtain an overall simulation as follows:

1. Draw m_1 labelled examples from $EX_{\mathrm{CN}}^{\eta}(f, D)$ in order to estimate the expectations in Step 2.

2. Run the SQ algorithm once for each of the $O(\frac{1}{\tau_*} \log \frac{1}{1-2\eta_b})$ η-guesses, estimating the various queries by applying Lemma 2 and using the sample drawn.

3. Draw m_2 samples and test the $O(\frac{1}{\tau_*} \log \frac{1}{1-2\eta_b})$ hypotheses obtained in Step 2. Output one of these hypotheses whose error rate is at most ϵ.

Step 3 can be accomplished by a generalization of a technique due to Laird [21]. The sample size required is

$$m_2 = O\left(\frac{1}{\epsilon(1-2\eta_b)^2} \log(\frac{1}{\delta\tau_*} \log \frac{1}{1-2\eta_b}) \right).$$

Since $1/\tau_* = \Omega(1/\epsilon)$ for all SQ algorithms [19], we obtain the following theorem on the total sample complexity of this simulation.

Theorem 7 *If $\mathcal{F}$ is learnable by a statistical query algorithm which makes queries from query space $\mathcal{Q}$ with worst case additive error τ_*, then $\mathcal{F}$ is PAC learnable in the presence of classification noise. If $\eta_b < 1/2$ is an upper bound on the noise rate, then the sample complexity required is*

$$O\Big(\tfrac{1}{\tau_*^2(1-2\eta_b)^2} \log \tfrac{|\mathcal{Q}|}{\delta} + \tfrac{1}{\epsilon(1-2\eta_b)^2} \log \log \tfrac{1}{1-2\eta_b} \Big)$$

when $\mathcal{Q}$ is finite or

$$O\Big(\tfrac{VC(\mathcal{Q})}{\tau_*^2(1-2\eta_b)^2} \log \tfrac{1}{\tau_*(1-2\eta_b)} + \tfrac{1}{\tau_*^2(1-2\eta_b)^2} \log \tfrac{1}{\delta} \Big)$$

when $\mathcal{Q}$ has finite VC-dimension.

By combining our results on general bounds for SQ learning and classification noise simulation, we immediately obtain the following corollary.

Corollary 1 *If $\mathcal{F}$ is SQ learnable, then $\mathcal{F}$ is PAC learnable in the presence of classification noise. The dependence on ϵ and η_b of the required sample complexity is $\tilde{O}(\frac{1}{\epsilon^2(1-2\eta_b)^2})$.*

To determine the running time of our simulation, one must distinguish between two different types of SQ algorithms. Some SQ algorithms submit a fixed set of queries independent of the estimates they receive for previous queries. We refer to these algorithms as "batch" SQ algorithms. Other SQ algorithms submit various queries based upon the estimates they receive for previous queries. We refer to these algorithms as "dynamic" SQ algorithms.[4] Note that multiple runs of a dynamic SQ algorithm may produce many more queries which need to be estimated. Since the vast majority of the time required to simulate most SQ algorithms is spent estimating queries using a large sample, the time complexity of simulating dynamic SQ algorithms is greatly affected by the "branching factor" of the simulation. By reducing the "branching factor" of the simulation from $\Theta(\frac{1}{\tau_*(1-2\eta_b)^2})$ to $\Theta(\frac{1}{\tau_*}\log\frac{1}{1-2\eta_b})$, the asymptotic running time of our simulation is greatly improved.

With respect to η, the running time of our simulation is $\tilde{O}(\frac{1}{(1-2\eta_b)^2})$. Simon [30] has shown a sample and time complexity lower bound of $\Omega(\frac{1}{(1-2\eta)^2})$ for PAC learning in the presence of classification noise. We therefore note that the running time of our simulation is optimal with respect to the noise rate (modulo lower order logarithmic factors). For dynamic algorithms, the time complexity of our new simulation is in fact a $\tilde{\Theta}(\frac{1}{(1-2\eta_b)^2})$ factor better than the current simulation.

[4] Note that we consider any SQ algorithm which uses a polynomially sized query space to be a "batch" algorithm since all queries may be processed in advance.

CHAPTER 4

Learning Results in the Relative Error SQ Model

In this chapter, we propose a new model of statistical query learning based on *relative error*. We show that learnability in this new model is polynomially equivalent to learnability in the standard, additive error model; however, this new model is advantageous in that SQ algorithms specified in this model can be simulated more efficiently in some important cases.

4.1 Introduction

In the standard model of statistical query learning, a learning algorithm asks for an estimate of the probability that a predicate χ is true. The required accuracy of this estimate is specified by the learner in the form of an additive error parameter. The limitation of this model is clearly evident in even the standard, *noise-free* statistical query simulation [19]. This simulation uses $\Omega(1/\tau_*^2)$ examples. Since $1/\tau_* = \Omega(1/\epsilon)$ for all SQ algorithms [19], this simulation effectively uses $\Omega(1/\epsilon^2)$ examples. However, the ϵ-dependence of the general bound on the sample complexity of PAC learning is $\tilde{\Theta}(1/\epsilon)$ [7, 11].

This $\Omega(1/\tau_*^2) = \Omega(1/\epsilon^2)$ sample complexity results from the worst case assumption that large probabilities may need to be estimated with small additive error in the SQ model. Either the nature of statistical query learning is such that learning sometimes requires the estimation of large probabilities with small additive error, or it is always sufficient to estimate each probability with an additive error comparable to the probability. If the former were the case, then the present model and simulations would be the best that one could hope for. We show that the latter is true, and that a model in which queries are specified with *relative error* is a more natural and strictly more powerful tool.

We define such a model of relative error statistical query learning and we show how this new model relates to the standard additive error model. We also show general upper bounds on learning in this new model which demonstrate that for *all* classes learnable by statistical queries, it is sufficient to make estimates with relative error independent of ϵ. We then give roughly optimal PAC simulations for relative error SQ algorithms. Finally, we demonstrate natural problems which only require estimates with *constant* relative error.

4.2 The Relative Error Statistical Query Model

Given the motivation above, we modify the standard model of statistical query learning to allow for estimates being requested with relative error. We replace the additive error $STAT(f, D)$ oracle with a relative error *Rel-*$STAT(f, D)$ oracle which accepts a query χ, a relative error parameter μ, and a threshold parameter θ. The value $P_\chi = \Pr_D[\chi(x, f(x))]$ is defined as before. If P_χ is less than the threshold θ, then the oracle may return the symbol $\perp$. If the oracle does not return $\perp$, then it must return an estimate $\hat{P}_\chi$ such that

$$P_\chi(1 - \mu) \leq \hat{P}_\chi \leq P_\chi(1 + \mu)$$

Note that the oracle may chose to return an accurate estimate even if $P_\chi < \theta$. A class is said to be learnable by relative error statistical queries if it satisfies the same conditions of additive error statistical query learning except we instead require that $1/\mu$ and $1/\theta$ are polynomially bounded. Let μ_* and θ_* be the lower bounds on the relative error and threshold of every query made by an SQ algorithm. Given this definition of relative error statistical query learning, we show the following desirable equivalence.

Theorem 8 *$\mathcal{F}$ is learnable by additive error statistical queries if and only if $\mathcal{F}$ is learnable by relative error statistical queries.*

Proof: One can take any query χ to the additive error oracle which requires additive error τ and simulate it by calling the relative error oracle with relative error τ and threshold τ. If $\hat{P}_\chi = \perp$, then return 0; else, return $\hat{P}_\chi$.

Similarly, one can take any query to the relative error oracle which requires relative error μ and threshold θ and simulate it by calling the additive error oracle with additive error $\mu\theta/3$. If $\hat{P}_\chi < \theta(1 - \mu/3)$, then return $\perp$; else, return $\hat{P}_\chi$.

In each direction, the simulation uses polynomially bounded parameters if and only if the original algorithm uses polynomially bounded parameters. □

Kearns [19] shows that almost all classes known to be PAC learnable are learnable with additive error statistical queries. By the above theorem, these classes are also learnable with relative error statistical queries. In addition, the hardness results of Kearns [19] for learning parity functions and the general hardness results of Blum *et al.* [6] based on Fourier analysis also hold for relative error statistical query learning.

4.3 A Natural Example of Relative Error SQ Learning

In this section we examine a learning problem which has both a simple additive error SQ algorithm and a simple relative error SQ algorithm. We consider the problem of learning a monotone conjunction of Boolean variables in which the learning algorithm must determine which subset of the variables $\{x_1, \ldots, x_n\}$ are contained in the unknown target conjunction f.

We construct an hypothesis h which contains all the variables in the target function f, and thus h will not misclassify any negative examples. We further guarantee that for each variable x_i in h, the distribution weight of examples which satisfy "$x_i = 0$ and $f(x) = 1$" is at most ϵ/n. Therefore, the distribution weight of positive examples which h will misclassify is at most ϵ. Such an hypothesis has error rate at most ϵ.

Consider the following query: $\chi_i(x, l) = [(x_i = 0) \wedge (l = 1)]$. P_{χ_i} is simply the probability that x_i is false and $f(x)$ is true. If variable x_i is in f, then $P_{\chi_i} = 0$. If we mistakenly include a variable x_i in our hypothesis which is not in f, then the error due to this inclusion is at most P_{χ_i}. We simply construct our hypothesis by including all target variables, but no variables x_i for which $P_{\chi_i} > \epsilon/n$.

An additive error SQ algorithm queries each χ_i with additive error $\epsilon/2n$ and includes all variables for which the estimate $\hat{P}_{\chi_i} \leq \epsilon/2n$. Even if $P_{\chi_i} = 1/2$, the oracle is constrained to return an estimate with additive error less than $\epsilon/2n$. A relative error SQ algorithm queries each χ_i with relative error $1/2$ and threshold ϵ/n and includes all variables for which the estimate $\hat{P}_{\chi_i}$ is 0 or $\perp$.

The sample complexity of the standard, noise-free PAC simulation of additive error SQ algorithms depends linearly on $1/\tau_*^2$ [19], while in Section 4.5, we show that the sample complexity of a noise-free PAC simulation of relative error SQ algorithms depends linearly on $1/\mu_*^2\theta_*$. Note that in the above algorithms for learning conjunctions, $1/\tau_*^2 = \Theta(n^2/\epsilon^2)$ while $1/\mu_*^2\theta_* = \Theta(n/\epsilon)$. We further note that μ_* is *constant* for learning conjunctions. We show in Section 4.4 that *no* learning problem requires μ_* to depend on ϵ and in Section 4.6 that μ_* is actually a constant in many algorithms.

4.4 General Bounds on Learning in the Relative Error SQ Model

In this section, we prove general upper bounds on the complexity of relative error statistical query learning. We do so by applying boosting techniques [14, 15, 28] and specifically, these techniques as applied in the statistical query model. We first prove some useful lemmas which allow us to decompose relative error estimates of ratios and sums.

Lemma 4 *If* $0 \leq a, b, c, \mu, \theta, \Phi \leq 1$ *and* $a = b/c$*, then to estimate* a *with* (μ, θ) *error provided that* $c \geq \Phi$*, it is sufficient to estimate* c *with* $(\mu/3, \Phi)$ *error and* b *with* $(\mu/3, \theta\Phi/2)$ *error.*

Proof: If the estimate $\hat{c}$ is $\perp$ or less than $\Phi(1-\mu/3)$, then $c < \Phi$. Therefore an estimate for a is not required, and we may halt. Otherwise $\hat{c} \geq \Phi(1-\mu/3)$, and therefore $c \geq \Phi\frac{1-\mu/3}{1+\mu/3} \geq \Phi/2$. If the estimate $\hat{b}$ is $\perp$, then $b < \theta\Phi/2$. Therefore $a = b/c < \theta$, so we may answer $\hat{a} = \perp$. Otherwise, $\hat{b}$ and $\hat{c}$ are estimates of b and c, each within a $1 \pm \mu/3$ factor. The theorem follows by noting the following facts.

$$\begin{aligned}
\frac{b\cdot(1+\mu/3)}{c\cdot(1-\mu/3)} &= a\cdot\frac{1+\mu/3}{1-\mu/3} \\
&= a\cdot(1+\mu/3)\left(1+\frac{\mu/3}{1-\mu/3}\right) \\
&\leq a\cdot(1+\mu/3)(1+\mu/2) \\
&= a\cdot(1+\mu/3+\mu/2+\mu^2/6) \\
&\leq a\cdot(1+\mu)
\end{aligned}$$

$$\begin{aligned}
\frac{b\cdot(1-\mu/3)}{c\cdot(1+\mu/3)} &= a\cdot\frac{1-\mu/3}{1+\mu/3} \\
&= a\cdot(1-\mu/3)\left(1-\frac{\mu/3}{1+\mu/3}\right) \\
&\geq a\cdot(1-\mu/3)(1-\mu/3) \\
&= a\cdot(1-2\mu/3+\mu^2/9) \\
&> a\cdot(1-\mu)
\end{aligned}$$

□

Lemma 5 *If $0 \leq s, p_i, z_i, \mu \leq 1$, $\sum_i p_i \leq 1$ and $s = \sum_i p_i z_i$, then to estimate s with (μ, θ) error, it is sufficient to estimate each z_i with $(\mu/3, \mu\theta/3)$ error provided that the p_i coefficients are known.*

Proof: Let $B = \{i : \text{estimate of } z_i \text{ is } \perp\}$, $E = \{i : \text{estimate of } z_i \text{ is } \hat{z}_i\}$, $s_B = \sum_B p_i z_i$ and $s_E = \sum_E p_i z_i$. Note that $s_B < \theta\mu/3$. Let $\hat{s}_E = \sum_E p_i \hat{z}_i$. If $\hat{s}_E < \theta(1-\mu/3)^2$ then we return $\perp$, otherwise we return $\hat{s}_E$.

If $\hat{s}_E < \theta(1-\mu/3)^2$, then $s_E < \theta(1-\mu/3)$. But in this case $s = s_E + s_B < \theta(1-\mu/3) + \theta\mu/3 = \theta$, so we are correct in returning $\perp$.

Otherwise we return $\hat{s}_E$ which is at least $\theta(1-\mu/3)^2$. If $B = \emptyset$, then it is easy to see that $\hat{s}_E$ is within a $1 \pm \mu/3$ (and therefore $1 \pm \mu$) factor of s. Otherwise, we are implicitly setting $\hat{z}_i = 0$ for each $i \in B$, and therefore it is enough to show that $\hat{s}_E \geq s(1-\mu)$.

Since $\hat{s}_E \geq \theta(1-\mu/3)^2$, we have $s_E \geq \theta(1-\mu/3)^2/(1+\mu/3)$. Using the fact that for all $\mu \leq 1$, $(1-\mu/3)/(1+\mu/3) \geq 1/2$, we have $s_E \geq \theta(1-\mu/3)/2$. If $\hat{s}_E \geq (\theta\mu/3 + s_E)(1-\mu)$, then $\hat{s}_E \geq s(1-\mu)$ since $s_B < \theta\mu/3$ and $s = s_B + s_E$. But since $\hat{s}_E \geq s_E(1-\mu/3)$, this condition holds when $s_E(1-\mu/3) \geq (\theta\mu/3 + s_E)(1-\mu)$. Solving for s_E, this final condition holds when $s_E \geq \theta(1-\mu)/2$ which we have shown to be true whenever an estimate is returned. □

Theorem 9 *If the concept class $\mathcal{F}$ is strongly SQ learnable, then $\mathcal{F}$ is strongly SQ learnable by an algorithm whose query complexity is $O(N_0 \log^2 \frac{1}{\epsilon})$, whose minimum requested relative error is $\Omega(\mu_0)$ and whose minimum requested threshold is $\Omega(\mu_0\theta_0\epsilon/\log(1/\epsilon))$ where $N_0 = p_1(n, size(f))$, $\mu_0 = 1/p_2(n, size(f))$ and $\theta_0 = 1/p_3(n, size(f))$ for some polynomials p_1, p_2 and p_3.*

Proof: If $\mathcal{F}$ is strongly SQ learnable, then there exists a relative error statistical query algorithm $\mathcal{A}$ for learning $\mathcal{F}$. Hardwire the accuracy parameter of $\mathcal{A}$ to $1/4$ and apply Scheme 2 boosting. The boosting scheme will run $16\ln(1/\epsilon)$ copies of $\mathcal{A}$ with respect to $16\ln(1/\epsilon)$ different distributions over the instance space. Each run makes at most $N_0 = N_*(1/4, n, size(f))$ queries, each with relative error no smaller than $\mu_0 = \mu_*(1/4, n, size(f))$ and threshold no smaller than $\theta_0 = \theta_*(1/4, n, size(f))$. In run $i+1$, the algorithm makes queries to $STAT(f, D_{i+1})$ where D_{i+1} is a distribution based on D. Since we only have access to a statistics oracle for D, queries to $STAT(f, D_{i+1})$ are simulated by a sequence of new queries to $STAT(f, D)$ as follows:

$$STAT(f, D_{i+1})[\chi(x, f(x))] = \frac{\sum_{j=0}^{w} \lambda_j^w \cdot STAT(f, D)[\chi \wedge \chi_j^w]}{\sum_{j=0}^{w} \lambda_j^w \cdot STAT(f, D)[\chi_j^w]} \tag{4.1}$$

In the above equation $w \leq i$, the values $\lambda_j^w \in [0, 1]$ are known, and $\sum_j \lambda_j^w \leq 1$. Also note that if the denominator of Equation 4.1 is less than $\Phi = \frac{\epsilon(1-\epsilon)}{4\ln(1/\epsilon)}$, then the query need not be estimated (this is the "abort" condition of Scheme 2). Applying Lemmas 4 and 5, we find that the queries in the denominator can be estimated with $(\mu_0/9, \mu_0\Phi/9)$ error, and the queries in the numerator can be estimated with $(\mu_0/9, \mu_0\theta_0\Phi/18)$ error. Since a query to $STAT(f, D_{i+1})$ requires $O(i)$ queries to $STAT(f, D)$, the total number of queries made is $O(N_0 \log^2(1/\epsilon))$. □

We finally note that the query space complexity obtained here is identical in form to the query space complexity obtained in Section 3.1.2.

4.5 Simulating Relative Error SQ Algorithms in the PAC Model

In this section, we derive the complexity of simulating relative error SQ algorithms in the PAC model, both in the absence and presence of noise. We also give general upper bounds on the complexity of PAC algorithms derived from SQ algorithms based on the simulations and the general bounds of the previous section. Note that there do not exist two-sided bounds for uniform convergence based on VC-dimension, so some of our results are based on drawing a separate sample for each query.

4.5.1 PAC Model Simulation

The simulation of relative error SQ algorithms in the noise-free PAC model is based on a Chernoff bound analysis. Let $GE(p, m, n)$ be the probability of *at least* n successes in m Bernoulli trials, where each trial has probability of success p. Similarly, let $LE(p, m, n)$ be the probability of *at most* n successes in m Bernoulli trials, where each trial has probability of success p. Chernoff's bounds may then be stated as follows [3]:

$$\begin{aligned} GE(p, m, mp(1+\alpha)) &\leq e^{mp\alpha^2/3} \\ LE(p, m, mp(1-\alpha)) &\leq e^{mp\alpha^2/2} \end{aligned}$$

Furthermore, we often make use of the following properties of GE and LE:

$$p \geq p' \implies LE(p, m, n) \leq LE(p', m, n) \tag{4.2}$$

$$p \leq p' \implies GE(p, m, n) \leq GE(p', m, n) \tag{4.3}$$

We may now prove following theorem.

Theorem 10 *If $\mathcal{F}$ is learnable by a statistical query algorithm which makes at most N_* queries from query space $\mathcal{Q}$ with worst case relative error μ_* and worst case threshold θ_*, then $\mathcal{F}$ is PAC learnable with sample complexity $O(\frac{1}{\mu_*^2\theta_*}\log\frac{|\mathcal{Q}|}{\delta})$ when $\mathcal{Q}$ is finite or $O(\frac{N_*}{\mu_*^2\theta_*}\log\frac{N_*}{\delta})$ when drawing a separate sample for each query.*

Proof: We first demonstrate how to estimate the value of a single query, and we then extend this technique to yield the desired result. Let $[\chi, \mu, \theta]$ be a query to be estimated, and let $p = P_\chi$. For a given sample of size m, let $\hat{p}$ be the fraction of examples which satisfy χ. In order to properly estimate the value of this query, we choose m large enough to ensure that each of the following hold with high probabilty:

1. If $\hat{p} < \theta/2$, then $p < \theta$.
2. If $\hat{p} \geq \theta/2$, then $p \geq \theta/4$.
3. If $p \geq \theta/4$, then $\hat{p} \geq (1-\mu)p$.
4. If $p \geq \theta/4$, then $\hat{p} \leq (1+\mu)p$.

Thus, if $\hat{p} < \theta/2$, we may output $\perp$, and if $\hat{p} \geq \theta/2$, we may output $\hat{p}$. To ensure a failure probability of at most δ, we choose m large enough to guarantee that each of the properties fails to hold with probability at most $\delta/4$. Let $m = \frac{12}{\mu^2\theta} \ln \frac{4}{\delta}$.

Suppose that $p \geq \theta$. Then the probability that $\hat{p} < \theta/2$ is bounded by:

$$\begin{aligned} LE(p, m, \theta/2) &\leq LE(\theta, m, \theta/2) \\ &\leq e^{-m\theta/8} \end{aligned}$$

Since $m > \frac{8}{\theta} \ln \frac{4}{\delta}$, this probability is less than $\delta/4$. Therefore, the probability that $\hat{p} \geq \theta/2$ is at least $1 - \delta/4$. Thus, we have shown the following: With probabilty at least $1 - \delta/4$,

$$p \geq \theta \Longrightarrow \hat{p} \geq \theta/2.$$

Since Property 1 is the contrapositive of the above statement, we have shown that it will fail to hold with probability at most $\delta/4$.

Property 2 is shown to hold in a similar manner, and Properties 3 and 4 are direct consequences of Chernoff bounds.

Now, by choosing $m = \frac{12}{\mu_*^2\theta_*} \ln \frac{4|\mathcal{Q}|}{\delta}$, we can ensure that all four properties will hold for all $\chi \in \mathcal{Q}$, with probability at least $1 - \delta$. If, on the other hand, we draw N_* separate samples each of size $m = \frac{12}{\mu_*^2 \theta_*} \ln \frac{4N_*}{\delta}$, we guarantee that all four properties will hold for each of the N_* queries estimated, with probability at least $1 - \delta$. $\square$

Corollary 2 *If $\mathcal{F}$ is SQ learnable, then $\mathcal{F}$ is PAC learnable with a sample complexity whose dependence on ϵ is $\tilde{\Theta}(1/\epsilon)$.*

Although one could use boosting techniques in the PAC model to achieve this nearly optimal sample complexity, these boosting techniques would result in a more complicated algorithm and output hypothesis (a circuit whose inputs were hypotheses from the original hypothesis class). If instead we have a relative error SQ algorithm meeting the bounds of Theorem 9, then we achieve this PAC sample complexity directly.

4.5.2 Classification Noise Model Simulation

For SQ simulations in the classification noise model, we achieve the sample complexity given in Theorem 11 below. This sample complexity is obtained by simulating an additive error SQ algorithm with $\tau = \mu\theta/3$ as in Theorem 8. Although this result does not improve the sample complexity of SQ simulations in the presence of classification noise, we believe that to improve upon this bound requires the use of relative error statistical queries for the reasons discussed in Section 4.1.

Theorem 11 *If $\mathcal{F}$ is learnable by a statistical query algorithm which makes queries from query space $\mathcal{Q}$ with worst case relative error μ_* and worst case threshold θ_*, then $\mathcal{F}$ is PAC learnable in the presence of classification noise. If $\eta_b < 1/2$ is an upper bound on the noise rate, then the sample complexity required is*

$$O\Big(\frac{1}{\mu_*^2 \theta_*^2 (1-2\eta_b)^2} \log \frac{|\mathcal{Q}|}{\delta} + \frac{1}{\epsilon(1-2\eta_b)^2} \log \log \frac{1}{1-2\eta_b}\Big)$$

when $\mathcal{Q}$ is finite or

$$O\Big(\frac{VC(\mathcal{Q})}{\mu_*^2 \theta_*^2 (1-2\eta_b)^2} \log \frac{1}{\mu_* \theta_* (1-2\eta_b)} + \frac{1}{\mu_*^2 \theta_*^2 (1-2\eta_b)^2} \log \frac{1}{\delta}\Big)$$

when $\mathcal{Q}$ has finite VC-dimension.

Corollary 3 *If $\mathcal{F}$ is SQ learnable, then $\mathcal{F}$ is PAC learnable in the presence of classification noise. The dependence on ϵ and η_b of the required sample complexity is $\tilde{O}(\frac{1}{\epsilon^2(1-2\eta_b)^2})$.*

4.5.3 Malicious Error Model Simulation

We next consider the simulation of relative error SQ algorithms in the presence of malicious errors. Decatur [9] has shown that an SQ algorithm can be simulated in the presence of malicious errors with a maximum allowable error rate which depends on τ_*, the smallest additive error required by the SQ algorithm. In Theorem 12, we show that an SQ algorithm can be simulated in the presence of malicious errors with a maximum allowable error rate and sample complexity which depend on μ_* and θ_*, the minimum *relative error* and *threshold* required by the SQ algorithm.

The key idea in this simulation is to draw a large enough sample such that for each query, the combined error in an estimate due to both the adversary and the statistical fluctuation on error-free examples is less than the accuracy required. We formally state this idea in the claim given below.

Claim 4 *Let P^*_χ be the fraction of examples satisfying χ in a noise-free sample of size m, and let $\hat{P}_\chi$ be the fraction of examples satisfying χ in a sample of size m drawn from $EX^{\beta}_{\mathrm{MAL}}(f, D)$. Then to ensure $|P_\chi - \hat{P}_\chi| \leq \tau_1 + \tau_2$, it is sufficient to draw a sample of size m which simultaneously ensures that:*

(1) The adversary corrupts at most a τ_1 fraction of the examples drawn from $EX^{\beta}_{\mathrm{MAL}}(f, D)$.

(2) $|P_\chi - P^*_\chi| \leq \tau_2$.

Theorem 12 *If $\mathcal{F}$ is learnable by a statistical query algorithm which makes at most N_* queries from query space $\mathcal{Q}$ with worst case relative error μ_* and worst case threshold θ_*, then $\mathcal{F}$ is PAC learnable in the presence of malicious errors. The maximum allowable error rate is $\beta_* = \Omega(\mu_*\theta_*)$, and the sample complexity required is $O(\frac{1}{\mu_*^2\theta_*}\log\frac{|\mathcal{Q}|}{\delta})$ when $\mathcal{Q}$ is finite or $O(\frac{N_*}{\mu_*^2\theta_*}\log\frac{N_*}{\delta})$ when drawing a separate sample for each query.*

Proof: We first analyze the tolerable error and sample complexity for simulating a single query and then determine these values for simulating the entire algorithm.

For a given query $[\chi, \mu, \theta]$, P_χ is the probability with respect to the noise-free example oracle which needs to be estimated with (μ, θ) error. Assume that $\beta \leq \mu\theta/16$ and let $\hat{\beta}$ be the actual fraction of the sample corrupted by the malicious adversary. We choose m large enough to ensure that the following hold with high probability:

1. If $\beta \le \mu\theta/16$, then $\hat{\beta} \le \mu\theta/8$.

2. If $P^*_\chi < 5\theta/8$, then $P_\chi < \theta$.

3. If $P^*_\chi \ge 3\theta/8$, then $P_\chi \ge \theta/4$.

4. If $P_\chi \ge \theta/4$, then $P^*_\chi \ge (1-\mu/2)P_\chi$.

5. If $P_\chi \ge \theta/4$, then $P^*_\chi \le (1+\mu/2)P_\chi$.

Suppose that Properties 1 through 5 all hold. If $\hat{P}_\chi < \theta/2$, then by Property 1, $P^*_\chi < 5\theta/8$, and by Property 2, $P_\chi < \theta$. Thus, we may return $\perp$.

If, on the other hand, $\hat{P}_\chi \ge \theta/2$, then by Property 1, $P^*_\chi \ge 3\theta/8$, and by Property 3, $P_\chi \ge \theta/4$. Property 4 then implies that $P^*_\chi \ge (1-\mu/2)P_\chi$, and by Property 1, we have the following:

$$\hat{P}_\chi \ge (1-\mu/2)P_\chi - \mu\theta/8 \ge (1-\mu/2)P_\chi - \mu P_\chi/2 = (1-\mu)P_\chi$$

By applying Property 5, we may similarly show the following:

$$\hat{P}_\chi \le (1+\mu/2)P_\chi + \mu\theta/8 \le (1+\mu/2)P_\chi + \mu P_\chi/2 = (1+\mu)P_\chi$$

Thus, we may return $\hat{P}_\chi$.

We can ensure that Properties 1 through 5 collectively hold with probability at least $1-\delta$ by letting $m = \frac{48}{\mu^2\theta}\ln\frac{5}{\delta}$. The proofs that each of these properties hold with high probability given this sample size are analogous to the proofs for the similar properties used in Theorem 10.

Now, by choosing $m = \frac{48}{\mu_*^2\theta_*}\ln\frac{5|\mathcal{Q}|}{\delta}$, we can ensure that all five properties will hold for all $\chi \in \mathcal{Q}$, with probability at least $1-\delta$. If, on the other hand, we draw N_* separate samples each of size $m = \frac{48}{\mu_*^2\theta_*}\ln\frac{5N_*}{\delta}$, we guarantee that all five properties will hold for each of the N_* queries estimated, with probability at least $1-\delta$. $\square$

Corollary 4 *If $\mathcal{F}$ is SQ learnable, then $\mathcal{F}$ is PAC learnable in the presence of malicious errors. The dependence on ϵ of the maximum allowable error rate is $\tilde{\Omega}(\epsilon)$, while the dependence on ϵ of the required sample complexity is $\tilde{\Theta}(1/\epsilon)$.*

Note that we are within logarithmic factors of both the $O(\epsilon)$ maximum allowable malicious error rate [20] and the $\Omega(1/\epsilon)$ lower bound on the sample complexity of *noise-free* PAC learning [11]. In this malicious error tolerant PAC simulation, the sample, time, space and hypothesis size complexities are asymptotically identical to the corresponding complexities in our noise-free PAC simulation.

4.6 Very Efficient Learning in the Presence of Malicious Errors

In previous sections, we have shown general upper bounds on the required complexity of relative error SQ algorithms and the efficiency of PAC algorithms derived from them. In this section, we describe relative error SQ algorithms which actually achieve these bounds and therefore have very efficient, malicious error tolerant PAC simulations. We first present a very efficient algorithm for learning conjunctions[1] in the presence of malicious errors when there are many irrelevant attributes. We then highlight a property of this SQ algorithm which allows for its efficiency, and we further show that many other SQ algorithms naturally exhibit this property as well. We can simulate these SQ algorithms in the malicious error model with roughly optimal malicious error tolerance *and* sample complexity.

Decatur [9] gives an algorithm for learning conjunctions which tolerates a malicious error rate *independent* of the number of irrelevant attributes, thus depending only on the number of *relevant* attributes and the desired accuracy. This algorithm, while reasonably efficient, is based on an additive error SQ algorithm of Kearns [19] and therefore does not have an optimal sample complexity.

We present an algorithm based on relative error statistical queries which tolerates the *same* malicious error rate and has a sample complexity whose dependence on ϵ roughly matches the general lower bound for *noise-free* PAC learning.

Theorem 13 *The class of conjunctions of size k over n variables is PAC learnable with malicious errors. The maximum allowable malicious error rate is $\Omega(\frac{\epsilon}{k\log\frac{1}{\epsilon}})$, and the sample complexity required is $O\left(\frac{k^2}{\epsilon}\log^2\frac{1}{\epsilon}\log n + \frac{k}{\epsilon}\log\frac{1}{\epsilon}\log\frac{1}{\delta}\right)$.*

Proof: We present a proof for learning *monotone* conjunctions of size k, and we note that this proof can easily be extended for learning non-monotone conjunctions of size k.

[1] By duality, identical results also hold for learning *disjunctions*.

The target function f is a conjunction of k variables. We construct an hypothesis h which is a conjunction of $r = O(k \log \frac{1}{\epsilon})$ variables such that the distribution weight of misclassified positive examples is at most $\epsilon/2$ and the distribution weight of misclassified negative examples is also at most $\epsilon/2$.

First, all variables which could contribute more than $\epsilon/2r$ error on the positive examples are eliminated from consideration. This is accomplished by using the same queries that the monotone conjunction SQ algorithm of Section 4.3 uses. The queries are asked with relative error $1/2$ and threshold $\epsilon/2r$.

Next, the negative examples are greedily "covered" so that the distribution weight of misclassified negative examples is no more than $\epsilon/2$. We say that a variable covers all negative examples for which this variable is false. We know that the set of variables in f is a cover of size k for the entire space of negative examples. We iteratively construct h by conjoining new variables such that the distribution weight of negative examples covered by each new variable is at least a $\frac{1}{2k}$ fraction of the distribution weight of negative examples remaining to be covered.

Given a partially constructed hypothesis $h_j = x_{i_1} \wedge x_{i_2} \wedge \cdots \wedge x_{i_j}$, let X_j^- be the set of negative examples not covered by h_j, *i.e.* $X_j^- = \{x : (f(x) = 0) \wedge (h_j(x) = 1)\}$. Let D_j^- be the *conditional distribution* on X_j^- induced by D, *i.e.* for any $x \in X_j^-$, $D_j^-(x) = D(x)/D(X_j^-)$. By definition, X_0^- is the space of negative examples and D_0^- is the conditional distribution on X_0^-. We know that the target variables not yet in h_j cover the remaining examples in X_j^-; hence, there exists a cover of X_j^- of size at most k. Thus there exists at least one variable which covers a set of negative examples in X_j^- whose distribution weight with respect to D_j^- is at least $1/k$.

Given h_j, for each x_i, let $\chi_{j,i}(x,l) = [A|B] = [x_i = 0 | (l = 0) \wedge (h_j(x) = 1)]$. Note that $P_{\chi_{j,i}}$ is the distribution weight, with respect to D_j^-, of negative examples in X_j^- covered by x_i. Thus there exists a variable x_i such that $P_{\chi_{j,i}}$ is at least $1/k$. To find such a variable, we ask queries of the above form with relative error $1/3$ and threshold $2/3k$. [Note that this is a query for a *conditional* probability, which must be determined by the ratio of two *unconditional* probabilities. We show how to do this below.] Since there exists a variable x_i such that $P_{\chi_{j,i}} \geq 1/k$, we are guaranteed to find some variable $x_{i'}$ such that the estimate $\hat{P}_{\chi_{j,i'}}$ is at least $\frac{1}{k}(1-\frac{1}{3}) = \frac{2}{3k}$. Note that if $\hat{P}_{\chi_{j,i'}} \geq \frac{2}{3k}$, then $P_{\chi_{j,i'}} \geq \frac{2}{3k}/(1+\frac{1}{3}) = \frac{1}{2k}$. Thus, by conjoining $x_{i'}$ to h_j, we are guaranteed to cover a set of negative examples in X_j^- whose distribution weight with respect to D_j^- is at least $1/2k$. Since the distribution weight, with respect to D_0^-, of uncovered negative examples is reduced by at least a $(1-\frac{1}{2k})$ factor in each iteration, it is easy to show that this method requires no more than $r = O(k \log\frac{1}{\epsilon})$ iterations to cover all but a set of negative examples whose distribution weight, with respect to D_0^- (and therefore with respect to D) is at most $\epsilon/2$.

We now show how to estimate the conditional probability query $[A|B]$ with relative error $\mu = 1/3$ and threshold $\theta = 2/3k$. We estimate both queries which constitute the standard expansion of the conditional probability. Appealing to Lemma 4, we first estimate $[B]$, the probability that a negative example is not covered by h, using relative error $\mu/3 = 1/9$ and threshold $\epsilon/2$. If this estimate is $\perp$ or less than $\frac{\epsilon}{2}(1-\frac{1}{9}) = \frac{4\epsilon}{9}$, then the weight of negative examples misclassified by h is at most $\epsilon/2$, so we may halt and output h. Otherwise, we estimate $[A \wedge B]$ with relative error $\mu/3 = 1/9$ and threshold $\theta(\epsilon/2)/2 = \frac{\epsilon}{6k}$. If this estimate is $\perp$, then we may return $\perp$, and if a value is returned, then we can return the ratio of our estimates for $[A \wedge B]$ and $[B]$ as an estimate for $[A|B]$.

For this algorithm, the worst case relative error is $\Omega(1)$, the worst case threshold is $\Omega(\frac{\epsilon}{k \log\frac{1}{\epsilon}})$, and $\log|\mathcal{Q}| = O(k \log\frac{1}{\epsilon} \log n)$. Therefore, the theorem follows from Theorem 12. □

An important property of this statistical query algorithm is that for every query, we need only to determine whether P_χ falls below some threshold or above some constant fraction of this threshold. This allows the relative error parameter μ to be a constant. The learning algorithm described in Section 4.3 for monotone conjunctions has this property, and we note that many other learning algorithms which involve "covering" also have this property (*e.g.* the standard SQ algorithms for learning decision lists and axis parallel rectangles). In all these cases we obtain very efficient, malicious error tolerant algorithms.

CHAPTER 5

Extensions

Throughout this thesis, we have assumed that queries submitted to the statistical query oracle were restricted to being $\{0,1\}$-valued functions of labelled examples. In this case, the oracle returned an estimate of the probability that $\chi(x, f(x)) = 1$ on an example x chosen randomly according to D.

We now generalize the SQ model to allow algorithms to submit queries which are *real-valued*. Formally, we define a real-valued query to be a mapping from labelled examples to the real interval $[0, M]$, $\chi : X \times \{0,1\} \to [0, M]$.[1] We define P_χ to be the expected value of χ, $P_\chi = \mathbf{E}_D[\chi(x, f(x))] = \mathbf{E}_{D_f}[\chi]$.

This generalization can be quite useful. If the learning algorithm requires the expected value of some function of labelled examples, it may simply specify this using a real-valued query. By suitably constructing new queries, the learning algorithm may calculate variance and other moments as well. This generalization gives the algorithm designer more freedom and power. Furthermore, the ability to efficiently simulate these algorithms in the PAC model, in both the absence and presence of noise, is retained as shown below.

The results given below are proven almost identically to their counterparts by simply applying Hoeffding and Chernoff style bounds for bounded real random variables. The following is a simple extension of results contained in McDiarmid [23]:

[1]The range $[0, M]$ is used so that we can derive efficient simulations of relative error SQ algorithms. For additive error SQ algorithms, one may consider any interval $[a, b]$ where $M = b - a$.

Theorem 14 *Let $X_1, X_2, \ldots, X_m$ be independent and identically distributed random variables where $0 \le X_i \le M$ and $p = \mathbf{E}[X_i]$, and let $\hat{p} = \frac{1}{m}\sum_{i=1}^{m} X_i$. For any $\alpha > 0$,*

$$\begin{aligned} \Pr[\hat{p} \ge p + \alpha] &\le e^{-2m\alpha^2/M^2} \\ \Pr[\hat{p} \le p - \alpha] &\le e^{-2m\alpha^2/M^2}. \end{aligned}$$

For any γ, $0 < \gamma < 1$,

$$\begin{aligned} \Pr[\hat{p} \ge p(1+\gamma)] &\le e^{-mp\gamma^2/3M} \\ \Pr[\hat{p} \le p(1-\gamma)] &\le e^{-mp\gamma^2/2M}. \end{aligned}$$

Note that when $M = 1$, the following sample complexities and noise tolerances are essentially identical to those for $\{0,1\}$-valued queries.

Theorem 15 *If $\mathcal{F}$ is learnable by a statistical query algorithm which makes at most N_* $[0, M]$-valued queries from query space $\mathcal{Q}$ with worst case additive error τ_*, then $\mathcal{F}$ is PAC learnable with sample complexity $O(\frac{M^2}{\tau_*^2}\log\frac{|\mathcal{Q}|}{\delta})$ when $\mathcal{Q}$ is finite or $O(\frac{N_* M^2}{\tau_*^2}\log\frac{N_*}{\delta})$ when drawing a separate sample for each query.*

Theorem 16 *If $\mathcal{F}$ is learnable by a statistical query algorithm which makes at most N_* $[0, M]$-valued queries from query space $\mathcal{Q}$ with worst case relative error μ_* and worst case threshold θ_*, then $\mathcal{F}$ is PAC learnable with sample complexity $O(\frac{M}{\mu_*^2\theta_*}\log\frac{|\mathcal{Q}|}{\delta})$ when $\mathcal{Q}$ is finite or $O(\frac{N_* M}{\mu_*^2\theta_*}\log\frac{N_*}{\delta})$ when drawing a separate sample for each query.*

Theorem 17 *If $\mathcal{F}$ is learnable by a statistical query algorithm which makes at most N_* $[0, M]$-valued queries from query space $\mathcal{Q}$ with worst case additive error τ_*, then $\mathcal{F}$ is PAC learnable in the presence of classification noise. If $\eta_b < 1/2$ is an upper bound on the noise rate, then the sample complexity required is*

$$O\left(\tfrac{M^2}{\tau_*^2(1-2\eta_b)^2}\log\tfrac{|\mathcal{Q}|}{\delta} + \tfrac{1}{\epsilon(1-2\eta_b)^2}\log\log\tfrac{1}{1-2\eta_b}\right)$$

when $\mathcal{Q}$ is finite or

$$O\left(\tfrac{N_* M^2}{\tau_*^2(1-2\eta_b)^2}\log\tfrac{N_*}{\delta} + \tfrac{1}{\epsilon(1-2\eta_b)^2}\log\log\tfrac{1}{1-2\eta_b}\right)$$

when drawing a separate sample for each query.

Theorem 18 *If $\mathcal{F}$ is learnable by a statistical query algorithm which makes at most N_* $[0, M]$-valued queries from query space $\mathcal{Q}$ with worst case relative error μ_* and worst case threshold θ_*, then $\mathcal{F}$ is PAC learnable in the presence of classification noise. If $\eta_b < 1/2$ is an upper bound on the noise rate, then the sample complexity required is*

$$O\left(\frac{M^2}{\mu_*^2\theta_*^2(1-2\eta_b)^2}\log\frac{|\mathcal{Q}|}{\delta} + \frac{1}{\epsilon(1-2\eta_b)^2}\log\log\frac{1}{1-2\eta_b}\right)$$

when $\mathcal{Q}$ is finite or

$$O\left(\frac{N_*M^2}{\mu_*^2\theta_*^2(1-2\eta_b)^2}\log\frac{N_*}{\delta} + \frac{1}{\epsilon(1-2\eta_b)^2}\log\log\frac{1}{1-2\eta_b}\right)$$

when drawing a separate sample for each query.

Theorem 19 *If $\mathcal{F}$ is learnable by a statistical queries algorithm which makes at most N_* $[0, M]$-valued queries from query space $\mathcal{Q}$ with worst case additive error τ_*, then $\mathcal{F}$ is PAC learnable in the presence of malicious errors. The maximum allowable error rate is $\Omega(\tau_*/M)$ and the sample complexity required is $O(\frac{M^2}{\tau_*^2}\log\frac{|\mathcal{Q}|}{\delta})$ when $\mathcal{Q}$ is finite or $O(\frac{N_*M^2}{\tau_*^2}\log\frac{N_*}{\delta})$ when drawing a separate sample for each query.*

Theorem 20 *If $\mathcal{F}$ is learnable by a statistical queries algorithm which makes at most N_* $[0, M]$-valued queries from query space $\mathcal{Q}$ with worst case relative error μ_* and worst case threshold θ_*, then $\mathcal{F}$ is PAC learnable in the presence of malicious errors. The maximum allowable error rate is $\Omega(\mu_*\theta_*/M)$ and the sample complexity required is $O(\frac{M}{\mu_*^2\theta_*}\log\frac{|\mathcal{Q}|}{\delta})$ when $\mathcal{Q}$ is finite or $O(\frac{N_*M}{\mu_*^2\theta_*}\log\frac{N_*}{\delta})$ when drawing a separate sample for each query.*

Chapter 6

Conclusions and Open Questions

We have examined the statistical query model of learning and derived the first general bounds on the complexity of learning in this model. We have further shown that our general bounds are nearly optimal in many respects by demonstrating a specific class of functions whose minimum learning complexity nearly matches our general bounds. We have also improved the current strategy for simulating SQ algorithms in the classification noise model by demonstrating a new simulation which is both more efficient and more easily generalized.

The standard statistical query model of learning has a number of demonstrable deficiencies, and we have proposed a variant of the statistical query model based on relative error in order to combat these deficiencies. We have demonstrated the equivalence of additive error and relative error SQ learnability, and we have derived general bounds on the complexity of learning in this new relative error SQ model. We have demonstrated strategies for simulating relative error SQ algorithms in the PAC model, both in the absence and presence of noise. Our simulations in the absence of noise and in the presence of malicious errors yield nearly optimal noise-tolerant PAC learning algorithms.

Finally, we have shown that our results in both the additive and relative error SQ models can be extended to allow for real-valued queries.

The question of what sample complexity is required to simulate statistical query algorithms in the presence of classification noise remains open. The current simulations of both additive and relative error SQ algorithms yield PAC algorithms whose sample complexities depend quadraticly on $1/\epsilon$. However, in the absence of computational restrictions, all finite concept classes can be learned in the presence of classification noise using a sample complexity which depends linearly on $1/\epsilon$ [21]. It seems highly unlikely that a $\tilde{O}(1/\epsilon)$ strategy for simulating additive error SQ algorithms exists; however, such a strategy for simulating relative error SQ algorithms seems plausible. This line of research is currently being pursued.

As discussed in Section 4.6, many classes which are SQ learnable have algorithms with a constant worst case relative error μ_*. Can one show that *all* classes which are SQ learnable have algorithms with this property, or instead characterize exactly which classes do?

Chapter A

Appendix

In this chapter, we prove a number of technical results used in the previous chapters.

A.1 The Finite Query Space Complexity of Boosting

In this section of the appendix, we show how to simplify the expression for the size of the query space of boosting and how to derive an expression for the size of the query space of hybrid boosting. These results apply when the query space and hypothesis class are finite.

A.1.1 The Size of the Query Space of Boosting

In Section 3.1, the following expression was obtained for the size of the query space of boosting:

$$|\mathcal{Q}_B| = |\mathcal{Q}_0| + \sum_{i=1}^{k}(i+1)\cdot\binom{|\mathcal{H}_0|+i-1}{i} + |\mathcal{Q}_0|\sum_{i=1}^{k}(i+1)\cdot\binom{|\mathcal{H}_0|+i-1}{i} \tag{A.1}$$

where k is k_1 or k_2 depending on the type of boosting used.

We begin by simplifying the expression $\sum_{i=1}^{k}(i+1)\cdot\binom{N+i-1}{i}$. In order to obtain a closed-form expression for this sum, we first eliminate the $(i+1)$ factor.

$$(i+1)\cdot\binom{N+i-1}{i} \quad = \quad i\cdot\frac{(N+i-1)!}{(N-1)!\,i!}+\binom{N+i-1}{i}$$

$$\begin{aligned}
&= N \cdot \frac{(N+i-1)!}{N!\,(i-1)!} + \binom{N+i-1}{i} \\
&= N \cdot \binom{N+i-1}{i-1} + \binom{N-1+i}{i}
\end{aligned}$$

Using the fact that $\sum_{i=0}^{m} \binom{n+i}{i} = \binom{n+m+1}{m}$, we now have the following:

$$\begin{aligned}
\sum_{i=1}^{k} (i+1) \cdot \binom{N+i-1}{i} &= N \sum_{i=1}^{k} \binom{N+i-1}{i-1} + \sum_{i=1}^{k} \binom{N-1+i}{i} \\
&= N \sum_{j=0}^{k-1} \binom{N+j}{j} + \sum_{i=0}^{k} \binom{N-1+i}{i} - 1 \\
&= N \cdot \binom{N+k}{k-1} + \binom{N+k}{k} - 1
\end{aligned}$$

Applying this fact to Equation A.1 above, we obtain the following closed-form expression:

$$|\mathcal{Q}_B| = (|\mathcal{Q}_0|+1) \binom{|\mathcal{H}_0|+k}{k} + |\mathcal{H}_0|(|\mathcal{Q}_0|+1) \binom{|\mathcal{H}_0|+k}{k-1} - 1 \tag{A.2}$$

In order to bound the above expression, we make use of the following inequality:

$$\begin{aligned}
\binom{n+m}{m} &= \frac{(n+m)(n+m-1)\cdots(n+1)}{m(m-1)\cdots 1} \\
&= \left(1+\frac{n}{m}\right)\left(1+\frac{n}{m-1}\right)\cdots\left(1+\frac{n}{1}\right) \\
&\leq (1+n)^m
\end{aligned}$$

Applying this inequality, we now have:

$$\begin{aligned}
|\mathcal{Q}_B| &\leq (|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+1)^k + |\mathcal{H}_0|(|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^{k-1} - 1 \\
&< (|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^k + (|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^k \\
&= 2(|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^k
\end{aligned} \tag{A.3}$$

The complexity of simulating an SQ algorithm depends on $\log|\mathcal{Q}_B|$. We have effectively shown the following:

$$\log|\mathcal{Q}_B| = O(\log|\mathcal{Q}_0| + k \log|\mathcal{H}_0|)$$

A.1.2 The Size of the Query Space of Hybrid Boosting

In the hybrid boosting scheme, the Scheme 1 and Scheme 2 boosting schemes are combined to obtain improved overall complexities. The Scheme 2 booster uses the Scheme 1 booster, run with $\epsilon = 1/4$, as its "weak learner," while the Scheme 1 booster uses the actual weak learner. Thus, $k_1 = k_1(\gamma, 1/4)$ and $k_2 = k_2(1/4, \epsilon)$. Let $\mathcal{Q}_{HB}$ be the query space of the hybrid booster, and let $\mathcal{Q}_{1/4}$ and $\mathcal{H}_{1/4}$ be the query space and hypothesis class of the Scheme 1 booster. By the results of the previous section, we have the following:

$$\begin{aligned} |\mathcal{Q}_{HB}| &< 2(|\mathcal{Q}_{1/4}|+1)(|\mathcal{H}_{1/4}|+2)^{k_2} \\ |\mathcal{Q}_{1/4}| &< 2(|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^{k_1} \end{aligned}$$

The hypotheses in $\mathcal{H}_{1/4}$ are majority functions of up to k_1 hypotheses from $\mathcal{H}_0$. The number of unique majority functions of i hypotheses from $\mathcal{H}_0$ is given by $\binom{|\mathcal{H}_0|+i-1}{i}$, and therefore the number of unique majority functions of up to k_1 hypotheses from $\mathcal{H}_0$ is given by $\sum_{i=1}^{k_1}\binom{|\mathcal{H}_0|+i-1}{i}$. Using the techniques of the previous section, we can simplify this expression as follows:

$$\begin{aligned} |\mathcal{H}_{1/4}| &= \sum_{i=1}^{k_1}\binom{|\mathcal{H}_0|+i-1}{i} \\ &= \sum_{i=0}^{k_1}\binom{|\mathcal{H}_0|-1+i}{i} - 1 \\ &= \binom{|\mathcal{H}_0|+k_1}{k_1} - 1 \\ &\leq (|\mathcal{H}_0|+1)^{k_1} - 1 \end{aligned}$$

Combining these results, we obtain the following:

$$|\mathcal{Q}_{HB}| < 2\left(2(|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^{k_1}+1\right)\left((|\mathcal{H}_0|+1)^{k_1}+1\right)^{k_2}$$

The complexity of simulating an SQ algorithm depends on $\log|\mathcal{Q}_{HB}|$. We have effectively shown the following:

$$\log|\mathcal{Q}_{HB}| = O(\log|\mathcal{Q}_0| + k_1 k_2 \log|\mathcal{H}_0|)$$

A.2 Proofs Involving VC-Dimension

In this section of the appendix, we prove a number of technical lemmas which involve the concept of Vapnik-Chervonenkis dimension [36]. We begin by defining VC-dimension and introducing a number of preliminary results.

A.2.1 Preliminaries

Let $\mathcal{G}$ be a set of $\{0,1\}$-valued functions defined over a domain X. For any countable set $S = \{x_1, \ldots, x_m\} \subseteq X$ and function $g \in \mathcal{G}$, g defines a *labelling* of S as follows: $\langle g(x_1), \ldots, g(x_m)\rangle$. S is said to be *shattered* by $\mathcal{G}$ if S can be labelled in all possible 2^m ways by functions in $\mathcal{G}$. The VC-dimension of $\mathcal{G}$, $VC(\mathcal{G})$, is defined to be the cardinality of the largest shattered set.

VC-dimension is often defined in terms of set-theoretic notation. One can view a function $g \in \mathcal{G}$ as an *indicator function* for a set $X_g \subseteq X$ where $X_g = \{x \in X : g(x) = 1\}$. For any set $S \subseteq X$, let $\Pi_{\mathcal{G}}(S) = \{S \cap X_g : g \in \mathcal{G}\}$. One can view $\Pi_{\mathcal{G}}(S)$ as the set of subsets of S "picked out" by functions in $\mathcal{G}$. Note that if $\Pi_{\mathcal{G}}(S) = 2^S$, then S is shattered by $\mathcal{G}$. For any integer $m \geq 1$, let $\Pi_{\mathcal{G}}(m) = \max\{|\Pi_{\mathcal{G}}(S)| : S \subseteq X, |S| = m\}$. One can view $\Pi_{\mathcal{G}}(m)$ as the maximum number of subsets of any set of size m "picked out" by functions in $\mathcal{G}$. Note that if $\Pi_{\mathcal{G}}(m) = 2^m$, then there exists a set of size m shattered by $\mathcal{G}$. One may define VC-dimension in terms of $\Pi_{\mathcal{G}}(m)$ as follows: $VC(\mathcal{G}) = \max\{m : \Pi_{\mathcal{G}}(m) = 2^m\}$.

We next prove a lemma concerning $\Pi_{\mathcal{G}}(m)$ which is used extensively in the sections that follow.

Lemma 6 *If $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$, then $\Pi_{\mathcal{G}}(m) \leq \Pi_{\mathcal{G}_1}(m) + \Pi_{\mathcal{G}_2}(m)$.*

Proof: For any m, let S_m be a set of size m such that $|\Pi_{\mathcal{G}}(S_m)| = \Pi_{\mathcal{G}}(m)$. Note that such a set is guaranteed to exist by the definition of $\Pi_{\mathcal{G}}(m)$. We next note that $\Pi_{\mathcal{G}}(S_m) = \Pi_{\mathcal{G}_1}(S_m) \cup \Pi_{\mathcal{G}_2}(S_m)$, and therefore $|\Pi_{\mathcal{G}}(S_m)| \leq |\Pi_{\mathcal{G}_1}(S_m)| + |\Pi_{\mathcal{G}_2}(S_m)|$. The proof is completed by noting that $|\Pi_{\mathcal{G}_1}(S_m)| \leq \Pi_{\mathcal{G}_1}(m)$ and $|\Pi_{\mathcal{G}_2}(S_m)| \leq \Pi_{\mathcal{G}_2}(m)$. □

The growth of the $\Pi_{\mathcal{G}}(m)$ function plays an important role in proving a number of results in PAC learning. Note that for any $m \leq VC(\mathcal{G})$, $\Pi_{\mathcal{G}}(m) = 2^m$. The following result due to Sauer [27] upper bounds the growth of $\Pi_{\mathcal{G}}(m)$ for all $m \geq VC(\mathcal{G})$.

Lemma 7 (Sauer's Lemma) *Let $\mathcal{G}$ be a set of $\{0,1\}$-valued functions, and let $d = VC(\mathcal{G})$. For all integers $m \geq d$, $\Pi_{\mathcal{G}}(m) \leq \sum_{i=0}^{d} \binom{m}{i}$.*

Blumer *et al.* [7] have shown that for all integers $m \geq d \geq 1$, $\sum_{i=0}^{d} \binom{m}{i} < (em/d)^d$ where e is the base of the natural logarithm. We present a new and simpler proof of this result below.

Lemma 8 *For all integers $m \geq d \geq 1$, $\sum_{i=0}^{d} \binom{m}{i} < (em/d)^d$*

Proof: Since $0 < d/m \leq 1$, we have:

$$\begin{aligned}
\left(\frac{d}{m}\right)^d \sum_{i=0}^{d} \binom{m}{i} &\leq \sum_{i=0}^{d} \left(\frac{d}{m}\right)^i \binom{m}{i} \\
&\leq \sum_{i=0}^{d} \left(\frac{d}{m}\right)^i \frac{m^i}{i!} \\
&= \sum_{i=0}^{d} \frac{d^i}{i!} \\
&< \sum_{i=0}^{\infty} \frac{d^i}{i!} \\
&= e^d
\end{aligned}$$

Dividing both sides of this inequality by $(d/m)^d$ yields the desired result. □

We may now characterize the growth of the $\Pi_{\mathcal{G}}(m)$ function as follows: $\Pi_{\mathcal{G}}(m)$ grows exponentially up to $m = VC(\mathcal{G})$, and $\Pi_{\mathcal{G}}(m)$ grows at most polynomially after that. We may use this fact to obtain an upper bound on the VC-dimension of $\mathcal{G}$ in the following way. Suppose that for some m, we could show that $\Pi_{\mathcal{G}}(m) < 2^m$. Then m must be larger than the VC-dimension of $\mathcal{G}$.

A.2.2 VC-Dimension of $\mathcal{Q}' = \mathcal{Q} \cup \overline{\mathcal{Q}}$

We now prove a result used in Section 3.3 concerning the VC-dimension of the query space used by our simulation of an SQ algorithm in the classification noise model. Recall that X is our instance space, and $Y = X \times \{0,1\}$ is our labelled example space. For any labelled example $y = \langle x, l \rangle$, we define $\overline{y} = \langle x, \overline{l} \rangle$, and for any query χ, we define $\overline{\chi}(y) = \chi(\overline{y})$. Finally, for any set of queries $\mathcal{Q}$, we define $\overline{\mathcal{Q}} = \{\overline{\chi} : \chi \in \mathcal{Q}\}$.

If $\mathcal{Q}$ is the query space of an SQ algorithm, then $\mathcal{Q}' = \mathcal{Q} \cup \overline{\mathcal{Q}}$ is the query space of our simulation of this SQ algorithm. We may bound the VC-dimension of $\mathcal{Q}'$ as follows.

Lemma 9 *If $\mathcal{Q}' = \mathcal{Q} \cup \overline{\mathcal{Q}}$, then $VC(\mathcal{Q}') \leq c \cdot VC(\mathcal{Q})$ for a constant $c \approx 4.66438$.*

Proof: We first claim that $VC(\overline{\mathcal{Q}}) = VC(\mathcal{Q})$. This fact can be shown as follows. For any $\chi \in \mathcal{Q}$, we have that $\chi(y) = \overline{\chi}(\overline{y})$. Note that if $\chi \in \mathcal{Q}$, then $\overline{\chi} \in \overline{\mathcal{Q}}$. For any countable set $T = \{y_1, \ldots, y_m\}$, the labelling of T induced by χ is identical to the labelling of $\overline{T}$ induced by $\overline{\chi}$ where $\overline{T} = \{\overline{y}_1, \ldots, \overline{y}_m\}$. Therefore, if there exits a set of size m shattered by $\mathcal{Q}$, then there exists a set of size m shattered by $\overline{\mathcal{Q}}$. This implies that $VC(\overline{\mathcal{Q}}) \geq VC(\mathcal{Q})$. The fact that $VC(\mathcal{Q}) \geq VC(\overline{\mathcal{Q}})$ is shown similarly, and thus $VC(\overline{\mathcal{Q}}) = VC(\mathcal{Q})$.

Let $d = VC(\mathcal{Q}) = VC(\overline{\mathcal{Q}})$. For any $m \geq d$, we have both $\Pi_{\mathcal{Q}}(m) < (em/d)^d$ and $\Pi_{\overline{\mathcal{Q}}}(m) < (em/d)^d$. Thus, for any $m \geq d$, we have $\Pi_{\mathcal{Q}'}(m) \leq \Pi_{\mathcal{Q}}(m) + \Pi_{\overline{\mathcal{Q}}}(m) < 2\,(em/d)^d$.

If $\Pi_{\mathcal{Q}'}(m) < 2^m$ for some m, then $m > VC(\mathcal{Q}')$. Thus, any $m \geq d$ which satisfies

$$2\,(em/d)^d \leq 2^m$$

is an upper bound on $VC(\mathcal{Q}')$. Setting $m = c \cdot d$ and solving for c, we obtain:

$$\begin{array}{rrcl} & 2\,(ecd/d)^d & \leq & 2^{cd} \\ \Leftrightarrow & 2\,(ec)^d & \leq & (2^c)^d \\ \Leftarrow & (2ec)^d & \leq & (2^c)^d \\ \Leftrightarrow & 2ec & \leq & 2^c \\ \Leftarrow & c & \geq & 4.66438 \end{array}$$

Thus, $VC(\mathcal{Q}') \leq c \cdot VC(\mathcal{Q})$ for a constant $c \approx 4.66438$ □

A.2.3 The VC-Dimension of the Query Space of Boosting

We now prove a result used in Section 3.1 concerning the VC-dimension of the query space of boosting. Let $\mathcal{Q}_0$ and $\mathcal{H}_0$ be the query space and hypothesis class used by a weak SQ learning algorithm. The queries used by the strong SQ learning algorithm obtained by either Scheme 1 or Scheme 2 boosting are of the form χ, χ_j^i and $\chi \wedge \chi_j^i$ where $\chi \in \mathcal{Q}_0$ and χ_j^i is constructed from hypotheses in $\mathcal{H}_0$.

A particular query χ_j^i is defined by i hypotheses and an integer j, $0 \leq j \leq i$. $\chi_j^i(x,l)$ is 1 if exactly j of the i hypotheses map x to l, and $\chi_j^i(x,l)$ is 0 otherwise. Note that i is bounded by $k_1 = \frac{1}{2\gamma^2} \ln \frac{1}{\epsilon}$ in Scheme 1 boosting, and i is bounded by $k_2 = \frac{1}{\gamma^2} \ln \frac{1}{\epsilon}$ in Scheme 2 boosting. Also note that the hypotheses used to construct a particular χ_j^i need not be distinct.

For fixed i and j, let Γ_j^i be the set of all χ_j^i queries. In addition, we make the following two definitions:

$$\begin{aligned} \Gamma^i &= \bigcup_{j=0}^{i} \Gamma_j^i \\ \Gamma^{[k]} &= \bigcup_{i=1}^{k} \Gamma^i \end{aligned}$$

For any two sets of $\{0,1\}$-valued functions $\mathcal{A}$ and $\mathcal{B}$, we define

$$\mathcal{A} \wedge \mathcal{B} = \{f_a \wedge f_b : f_a \in \mathcal{A}, f_b \in \mathcal{B}\}.$$

The query space of boosting, $\mathcal{Q}_B$, may then be given as follows:

$$\mathcal{Q}_B = \mathcal{Q}_0 \cup \Gamma^{[k]} \cup \mathcal{Q}_0 \wedge \Gamma^{[k]}$$

Note that $k = k_1$ in the case of Scheme 1 boosting, and $k = k_2$ in the case of Scheme 2 boosting.

We may bound the VC-dimension of $\mathcal{Q}_B$ in terms of the VC-dimensions of $\mathcal{Q}_0$ and $\mathcal{H}_0$ in a manner similar to that used in the previous section. In particular, we bound $\Pi_{\mathcal{Q}_0}(m)$, $\Pi_{\Gamma^{[k]}}(m)$ and $\Pi_{\mathcal{Q}_0 \wedge \Gamma^{[k]}}(m)$. By applying Lemma 6, we obtain a bound on $\Pi_{\mathcal{Q}_B}(m)$. From this bound, we obtain a bound on the VC-dimension of $\mathcal{Q}_B$. We begin by examining Γ_j^i.

For any hypothesis $h : X \rightarrow \{0,1\}$, we define $\hat{h} : X \times \{0,1\} \rightarrow \{0,1\}$ as follows

$$\hat{h}(x,l) = (h(x) \equiv l)$$

where $\equiv$ is the binary equivalence operator. Thus, $\hat{h}(x,l)$ is true if and only if the hypothesis h maps x to l. Let $\widehat{\mathcal{H}}_0 = \{\hat{h} : h \in \mathcal{H}_0\}$. We may now define a query $\chi_j^i \in \Gamma_j^i$ as follows. Let

$h_1, \ldots, h_i$ be the i hypotheses used to construct χ_j^i.

$$\chi_j^i(x,l) = \begin{cases} 1 & \text{if exactly } j \text{ of } \hat{h}_1(x,l), \ldots, \hat{h}_i(x,l) \text{ are } 1 \\ 0 & \text{otherwise} \end{cases}$$

From a set-theoretic perspective, we can view χ_j^i and $\hat{h}$ as indicator functions for subsets of $Y = X \times \{0,1\}$. We then have the following:

$$Y_{\chi_j^i} = \{y \in Y : y \text{ is an element of exactly } j \text{ of the sets } Y_{\hat{h}_1}, \ldots, Y_{\hat{h}_i}\}$$

We next relate $\Pi_{\Gamma_j^i}(m)$ with $\Pi_{\widehat{\mathcal{H}}_0}(m)$ as follows.

Claim 5 $\Pi_{\Gamma_j^i}(m) \leq \binom{\Pi_{\widehat{\mathcal{H}}_0}(m) + i - 1}{i}$

Proof: Consider a particular χ_j^i. We can view χ_j^i as either a mapping from Y to $\{0,1\}$ or as an indicator function for a set $Y_{\chi_j^i} \subseteq Y$. In the discussion that follows, it will be more convenient to view χ_j^i as an indicator function.

Let T be any subset of Y of size m. $\Pi_{\widehat{\mathcal{H}}_0}(T)$ is the set of subsets of T picked out by functions $\hat{h} \in \widehat{\mathcal{H}}_0$, and $\Pi_{\Gamma_j^i}(T)$ is the set of subsets of T picked out by functions χ_j^i in Γ_j^i. By the definition of χ_j^i, note that each *unique* set in $\Pi_{\Gamma_j^i}(T)$ must correspond to a *unique collection* of i sets in $\Pi_{\widehat{\mathcal{H}}_0}(T)$. However, the i sets in each unique collection need not be distinct. Therefore, the number of unique collections is given by the number of arrangements of i indistinguishable balls in $|\Pi_{\widehat{\mathcal{H}}_0}(T)|$ bins. We thus have

$$|\Pi_{\Gamma_j^i}(T)| \leq \binom{|\Pi_{\widehat{\mathcal{H}}_0}(T)| + i - 1}{i}$$

which implies the desired result. $\square$

By Lemma 6 and the definition of Γ^i, we now have

$$\Pi_{\Gamma^i}(m) \leq (i+1) \cdot \binom{\Pi_{\widehat{\mathcal{H}}_0}(m) + i - 1}{i}.$$

Furthermore, by Lemma 6 and the definition of $\Gamma^{[k]}$, we have

$$\Pi_{\Gamma^{[k]}}(m) \leq \sum_{i=1}^{k}(i+1)\cdot\binom{\Pi_{\widehat{\mathcal{H}}_0}(m)+i-1}{i}.$$

By applying the well known fact that $\Pi_{\mathcal{A}\wedge\mathcal{B}}(m) \leq \Pi_{\mathcal{A}}(m)\cdot\Pi_{\mathcal{B}}(m)$ [4, pg. 104], we now have

$$\Pi_{\mathcal{Q}_0\wedge\Gamma^{[k]}}(m) \leq \Pi_{\mathcal{Q}_0}(m)\sum_{i=1}^{k}(i+1)\cdot\binom{\Pi_{\widehat{\mathcal{H}}_0}(m)+i-1}{i}.$$

Finally, by Lemma 6 and the definition of $\mathcal{Q}_B$, we have

$$\Pi_{\mathcal{Q}_B}(m) \leq$$
$$\Pi_{\mathcal{Q}_0}(m)+\sum_{i=1}^{k}(i+1)\cdot\binom{\Pi_{\widehat{\mathcal{H}}_0}(m)+i-1}{i}+\Pi_{\mathcal{Q}_0}(m)\sum_{i=1}^{k}(i+1)\cdot\binom{\Pi_{\widehat{\mathcal{H}}_0}(m)+i-1}{i}. \quad \text{(A.4)}$$

Note that Equation A.4 is of the same form as Equation A.1. We can therefore simplify Equation A.4 in a similar manner to obtain:

$$\Pi_{\mathcal{Q}_B}(m) < 2(\Pi_{\mathcal{Q}_0}(m)+1)(\Pi_{\widehat{\mathcal{H}}_0}(m)+2)^k \quad \text{(A.5)}$$

In order to bound the VC-dimension of $\mathcal{Q}_B$, we must relate the VC-dimension of $\widehat{\mathcal{H}}_0$ with the VC-dimension of $\mathcal{H}_0$.

Claim 6 $VC(\widehat{\mathcal{H}}_0) = VC(\mathcal{H}_0)$

Proof: We begin by noting that for any instance $x \in X$, $h(x) = 1$ if and only if $\hat{h}(x,1) = 1$. For any countable set $S = \{x_1,\ldots,x_m\}$ and hypothesis $h \in \mathcal{H}_0$, the labelling of S induced by h is identical to the labelling of T induced by $\hat{h}$ where $T = \{\langle x_1,1\rangle,\ldots,\langle x_m,1\rangle\}$. Thus, if there exists a set of size m shattered by $\mathcal{H}_0$, then there exists a set of size m shattered by $\widehat{\mathcal{H}}_0$. This implies that $VC(\widehat{\mathcal{H}}_0) \geq VC(\mathcal{H}_0)$.

We next note that for all functions $\hat{h} \in \widehat{\mathcal{H}}_0$, $\hat{h}(x,l) = \neg\hat{h}(x,\bar{l})$. Now let

$$T = \{\langle x_1,l_1\rangle,\ldots,\langle x_m,l_m\rangle\}$$

be any countable set shattered by $\widehat{\mathcal{H}}_0$. If $\langle x,l\rangle \in T$, then $\langle x,\bar{l}\rangle \notin T$ since $\langle x,l\rangle$ and $\langle x,\bar{l}\rangle$ cannot be labelled identically, which is required for shattering. Thus, $S = \{x_1, \ldots, x_m\}$ is of size m.

Now note that $h(x) = b$ if and only if $\hat{h}(x, l) = (b \equiv l)$. Consider any labelling $\langle b_1, \ldots, b_m \rangle$ of S. This labelling of S would be induced by the hypothesis $h \in \mathcal{H}_0$ corresponding to the function $\hat{h} \in \widehat{\mathcal{H}}_0$ which labels T as follows: $\langle (b_1 \equiv l_1), \ldots, (b_m \equiv l_m) \rangle$. Since T is shattered by $\widehat{\mathcal{H}}_0$, such a function and corresponding hypothesis must exist. Thus, if there exists a set of size m shattered by $\widehat{\mathcal{H}}_0$, then there exists a set of size m shattered by $\mathcal{H}_0$. This implies that $VC(\mathcal{H}_0) \geq VC(\widehat{\mathcal{H}}_0)$. □

We are now in a position to prove the main result of this section.

Lemma 10 $VC(\mathcal{Q}_B) = O(VC(\mathcal{Q}_0) + VC(\mathcal{H}_0) \cdot k \log k)$

Proof: In order to bound the VC-dimension of $\mathcal{Q}_B$, we need only find an m which satisfies $\Pi_{\mathcal{Q}_B}(m) < 2^m$. We begin by further simplifying the expression for $\Pi_{\mathcal{Q}_B}(m)$.

Assume that $\Pi_{\mathcal{Q}_0}(m) \geq 1$ and $\Pi_{\widehat{\mathcal{H}}_0}(m) \geq 2$. Each of these assumptions is assured when $m \geq 1$ and the VC-dimensions of $\mathcal{Q}_0$ and $\widehat{\mathcal{H}}_0$ are at least 1. We then have the following:

$$
\begin{aligned}
\Pi_{\mathcal{Q}_B}(m) &< 2(\Pi_{\mathcal{Q}_0}(m) + 1)(\Pi_{\widehat{\mathcal{H}}_0}(m) + 2)^k \\
&\leq 2(2\Pi_{\mathcal{Q}_0}(m))(2\Pi_{\widehat{\mathcal{H}}_0}(m))^k \\
&= 2^{k+2}\, \Pi_{\mathcal{Q}_0}(m)\, (\Pi_{\widehat{\mathcal{H}}_0}(m))^k
\end{aligned}
$$

Let $q_0 = VC(\mathcal{Q}_0)$ and let $d_0 = VC(\widehat{\mathcal{H}}_0) = VC(\mathcal{H}_0)$. For any $m \geq \max\{q_0, d_0\}$, we have both $\Pi_{\mathcal{Q}_0}(m) < (em/q_0)^{q_0}$ and $\Pi_{\widehat{\mathcal{H}}_0}(m) < (em/d_0)^{d_0}$. We now have:

$$
\Pi_{\mathcal{Q}_B}(m) < 2^{k+2}(em/q_0)^{q_0}(em/d_0)^{d_0 k}
$$

To bound the VC-dimension of $\mathcal{Q}_B$, we need only find an m which guarantees that the right-hand side of the above inequality is at most 2^m.

$$
\begin{aligned}
& 2^{k+2}(em/q_0)^{q_0}(em/d_0)^{d_0 k} &\leq&\ 2^m \\
\Leftarrow\quad & (k+2) + q_0 \lg(em/q_0) + d_0 k \lg(em/d_0) &\leq&\ m \qquad \text{(A.6)}
\end{aligned}
$$

For fixed d_0, q_0 and k, the above inequality has the form $m \geq g_1(m) + g_2(m) + g_3(m)$ where each function $g_i(m)$ "grows" more slowly than m. In particular, each function g_i satisfies the following property (recall that we are restricted to values $m \geq \max\{q_0, d_0\}$): If $m_i \geq g_i(3m_i)$, then $m \geq g_i(3m)$ for all $m \geq m_i$. Our strategy is as follows. Find appropriate values of m_i which satisfy $m_i \geq g_i(3m_i)$, and let $m = 3\max\{m_1, m_2, m_3\}$. Then m must satisfy $m \geq g_1(m) + g_2(m) + g_3(m)$. The reasoning is as follows. Suppose, without loss of generality, that $m_1 = \max\{m_1, m_2, m_3\}$. We then have $m = 3m_1$. Furthermore, $m_1 \geq g_1(3m_1)$, and since $m_1 \geq m_2$ and $m_1 \geq m_3$, we also have $m_1 \geq g_2(3m_1)$ and $m_1 \geq g_3(3m_1)$. Combining these inequalities, we have $3m_1 \geq g_1(3m_1) + g_2(3m_1) + g_3(3m_1)$ which implies the desired result.

For $g_1(m) = k + 2$, we may simply choose $m_1 = k + 2$. For $g_2(m) = q_0 \lg(em/q_0)$, we chose $m_2 = 6q_0$, which is verified as follows:

$$\begin{array}{rrcl} & 6q_0 & \geq & q_0 \lg(e(3 \cdot 6q_0)/q_0) \\ \Leftrightarrow & 6 & \geq & \lg(18e) \approx 5.613 \end{array}$$

For $g_3(m) = d_0 k \lg(em/d_0)$, we choose $m_3 = 9d_0 k \lg k$, which is verified as follows:

$$\begin{array}{rrcl} & 9d_0 k \lg k & \geq & d_0 k \lg(e(3 \cdot 9d_0 k \lg k)/d_0) \\ \Leftrightarrow & 9 \lg k & \geq & \lg(27ek \lg k) \\ \Leftrightarrow & k^9 & \geq & 27ek \lg k \\ \Leftarrow & k^7 & \geq & 27e \end{array}$$

This final inequality is true for any $k \geq 2$. We have effectively shown the following, which completes the proof: $m = 3\max\{k + 2, 6q_0, 9d_0 k \lg k\} = O(q_0 + d_0 k \log k)$. □

A.2.4 The VC-Dimension of the Query Space of Hybrid Boosting

We now prove a result used in Section 3.1 concerning the VC-dimension of the query space of hybrid boosting. As in Section A.1.2, let $\mathcal{Q}_{HB}$ be the query space of the hybrid booster, and let $\mathcal{Q}_{1/4}$ and $\mathcal{H}_{1/4}$ be the query space and hypothesis class of the Scheme 1 booster. Furthermore, let $k_1 = k_1(\gamma, 1/4)$ and $k_2 = k_2(1/4, \epsilon)$. We then have the following analogs of Equation A.5:

$$\begin{aligned}
\Pi_{\mathcal{Q}_{HB}}(m) &< 2(\Pi_{\mathcal{Q}_{1/4}}(m)+1)(\Pi_{\widehat{\mathcal{H}}_{1/4}}(m)+2)^{k_2} \\
\Pi_{\mathcal{Q}_{1/4}}(m) &< 2(\Pi_{\mathcal{Q}_0}(m)+1)(\Pi_{\widehat{\mathcal{H}}_0}(m)+2)^{k_1}
\end{aligned}$$

Now, $\mathcal{H}_{1/4}$ is the set of hypotheses which are majority functions of up to k_1 hypotheses from $\mathcal{H}_0$, and $\widehat{\mathcal{H}}_{1/4} = \{\hat{h} : h \in \mathcal{H}_{1/4}\}$. However, one can also define each function $\hat{h} \in \widehat{\mathcal{H}}_{1/4}$ as follows. Given a function $\hat{h} \in \widehat{\mathcal{H}}_{1/4}$, $\hat{h}$ corresponds to some hypothesis $h \in \mathcal{H}_{1/4}$ which in turn corresponds to some set of hypotheses $\{h_1, \dots, h_j\}$ from $\mathcal{H}_0$ where $j \le k_1$. By definition, we have

$$\hat{h}(x,l) = (\mathrm{maj}\,\{h_1(x), \dots, h_j(x)\} \equiv l).$$

However, it is also the case that

$$(\mathrm{maj}\,\{h_1(x), \dots, h_j(x)\} \equiv l) = \mathrm{maj}\,\{\hat{h}_1(x,l), \dots, \hat{h}_j(x,l)\}.$$

Thus, we can think of $\widehat{\mathcal{H}}_{1/4}$ as the set of majority functions of up to k_1 functions from $\widehat{\mathcal{H}}_0$.

Now, let $\widehat{\mathcal{H}}^j_{1/4}$ be the set of majority functions of j functions from $\widehat{\mathcal{H}}_0$, and let T be any subset of Y of size m. $\Pi_{\widehat{\mathcal{H}}_0}(T)$ is the set of subsets of T picked out by functions $\hat{h} \in \widehat{\mathcal{H}}_0$, and $\Pi_{\widehat{\mathcal{H}}^j_{1/4}}(T)$ is the set of subsets of T picked out by functions $\hat{h} \in \widehat{\mathcal{H}}^j_{1/4}$. Note that each *unique* set in $\Pi_{\widehat{\mathcal{H}}^j_{1/4}}(T)$ must correspond to a *unique collection* of j sets in $\Pi_{\widehat{\mathcal{H}}_0}(T)$. Since the j sets in each unique collection need not be distinct, the number of such unique collections is given by the number of arrangements of j indistinguishable balls in $|\Pi_{\widehat{\mathcal{H}}_0}(T)|$ bins. Thus, we have $|\Pi_{\widehat{\mathcal{H}}^j_{1/4}}(T)| \le \binom{|\Pi_{\widehat{\mathcal{H}}_0}(T)|+j-1}{j}$ which implies that

$$\Pi_{\widehat{\mathcal{H}}^j_{1/4}}(m) \le \binom{\Pi_{\widehat{\mathcal{H}}_0}(m)+j-1}{j}.$$

Since $\widehat{\mathcal{H}}_{1/4} = \bigcup_{j=1}^{k_1} \widehat{\mathcal{H}}^j_{1/4}$, by Lemma 6, we have:

$$\Pi_{\widehat{\mathcal{H}}_{1/4}}(m) \le \sum_{j=1}^{k_1} \binom{\Pi_{\widehat{\mathcal{H}}_0}(m)+j-1}{j}$$

$$
\begin{aligned}
&= \sum_{j=0}^{k_1} \binom{\Pi_{\widehat{\mathcal{H}}_0}(m)+j-1}{j} - 1 \\
&= \binom{\Pi_{\widehat{\mathcal{H}}_0}(m)+k_1}{k_1} - 1 \\
&\leq (\Pi_{\widehat{\mathcal{H}}_0}(m)+1)^{k_1} - 1
\end{aligned}
$$

Combining the above results, we have the following:

$$\Pi_{\mathcal{Q}_{HB}}(m) < 2\left(2(\Pi_{\mathcal{Q}_0}(m)+1)(\Pi_{\widehat{\mathcal{H}}_0}(m)+2)^{k_1}+1\right)\left((\Pi_{\widehat{\mathcal{H}}_0}(m)+1)^{k_1}+1\right)^{k_2}$$

Now, assume that $\Pi_{\mathcal{Q}_0}(m) \geq 1$ and $\Pi_{\widehat{\mathcal{H}}_0}(m) \geq 2$. Each of these assumptions is assured when $m \geq 1$ and the VC-dimensions of $\mathcal{Q}_0$ and $\widehat{\mathcal{H}}_0$ are at least 1. We then have the following:

$$
\begin{aligned}
\Pi_{\mathcal{Q}_{HB}}(m) &< 2\left(2(\Pi_{\mathcal{Q}_0}(m)+1)(\Pi_{\widehat{\mathcal{H}}_0}(m)+2)^{k_1}+1\right)\left((\Pi_{\widehat{\mathcal{H}}_0}(m)+1)^{k_1}+1\right)^{k_2} \\
&< 2\left(2(2\Pi_{\mathcal{Q}_0}(m))(2\Pi_{\widehat{\mathcal{H}}_0}(m))^{k_1}+1\right)\left((2\Pi_{\widehat{\mathcal{H}}_0}(m))^{k_1}+1\right)^{k_2} \\
&< 2\left(2^{k_1+3}\Pi_{\mathcal{Q}_0}(m)(\Pi_{\widehat{\mathcal{H}}_0}(m))^{k_1}\right)\left(2^{k_1+1}(\Pi_{\widehat{\mathcal{H}}_0}(m))^{k_1}\right)^{k_2} \\
&= 2^{(k_1+1)(k_2+1)+3}\,\Pi_{\mathcal{Q}_0}(m)\,(\Pi_{\widehat{\mathcal{H}}_0}(m))^{k_1(k_2+1)}
\end{aligned}
$$

Let $q_0 = VC(\mathcal{Q}_0)$ and let $d_0 = VC(\widehat{\mathcal{H}}_0) = VC(\mathcal{H}_0)$. For any $m \geq \max\{q_0, d_0\}$, we have both $\Pi_{\mathcal{Q}_0}(m) < (em/q_0)^{q_0}$ and $\Pi_{\widehat{\mathcal{H}}_0}(m) < (em/d_0)^{d_0}$. We now have:

$$\Pi_{\mathcal{Q}_{HB}}(m) < 2^{(k_1+1)(k_2+1)+3}(em/q_0)^{q_0}(em/d_0)^{d_0k_1(k_2+1)}$$

To bound the VC-dimension of $\mathcal{Q}_{HB}$, we need only find an m which guarantees that the right-hand side of the above inequality is at most 2^m.

$$
\begin{aligned}
2^{(k_1+1)(k_2+1)+3}(em/q_0)^{q_0}(em/d_0)^{d_0k_1(k_2+1)} &\leq 2^m \\
\Leftarrow \quad ((k_1+1)(k_2+1)+3) + q_0\lg(em/q_0) + d_0k_1(k_2+1)\lg(em/d_0) &\leq m \qquad \text{(A.7)}
\end{aligned}
$$

Inequality A.7 has the same form as Inequality A.6. By appropriate substitution, we find that $m = 3\max\{(k_1+1)(k_2+1)+3, 6q_0, 9d_0k_1(k_2+1)\lg(k_1(k_2+1))\} = O((q_0 + d_0k_1k_2\log(k_1k_2))$ is sufficient to satisfy the above inequality. We have effectively proven the following:

Lemma 11 $VC(\mathcal{Q}_{HB}) = O(VC(\mathcal{Q}_0) + VC(\mathcal{H}_0) \cdot k_1 k_2 \log(k_1 k_2))$

A.3 A Lower Bound for Probabilistic SQ Algorithms

Throughout this thesis, we have assumed that SQ algorithms are *deterministic* and output an accurate hypothesis *with certainty.* In this section of the appendix, we relax this condition by allowing *probabilistic* SQ algorithms which output accurate hypotheses *with high probability.* In particular, we show a lower bound on the query and tolerance complexities of such SQ algorithms which is analogous to the result obtained in Section 3.2.2.

Consider the two player game described in Section 3.2.2. We modify this game by allowing the player to be probabilistic, and we only require that the player output an acceptable set with probability at least $1 - \delta$, for some some $\delta > 0$. We may now show the following.

Lemma 12 *For any $d \geq 4$, $t \leq d/4$, $\delta < 1/8$ and $N = \Omega(d^{1+\alpha})$ for some $\alpha > 0$, the probabilistic player requires $\Omega(\frac{d \log N}{\log(d \log N)})$ queries of the oracle.*

Proof: By incorporating techniques from Kearns' lower bound proof [19], we can modify the original proof of Lemma 1 as follows. The adversary chooses the target set S randomly and uniformly from the set $\mathcal{S}_0$ of all $\binom{N}{d}$ subsets of $[N]$ of size d. Consider the first query, Q_1, submitted by the player. Q_1 partitions $\mathcal{S}_0$ into $d+1$ sets $\mathcal{S}_0^0, \mathcal{S}_0^1, \ldots, \mathcal{S}_0^d$ where each subset $S \in S_0^i$ has $|S \cap Q_1| = i$. Since the choice of the target set was random, uniform and independent of Q_1, the probability that the target set is an element of $\mathcal{S}_0^i$ is proportional to $|\mathcal{S}_0^i|$. Note that $\mathcal{S}_1$, by definition, is the set $\mathcal{S}_0^i$ of which the target is a member.

For any $k \geq 2$, consider all $\mathcal{S}_0^i$ for which $|\mathcal{S}_0^i| < |\mathcal{S}_0|/(k \cdot (d+1))$. Since there are only $d+1$ sets, the total cardinality of such "small" sets is less than $|\mathcal{S}_0|/k$. Thus, we have the following:

$$\Pr\left[|\mathcal{S}_1| \geq \frac{|\mathcal{S}_0|}{k \cdot (d+1)}\right] > \frac{|\mathcal{S}_0| - |\mathcal{S}_0|/k}{|\mathcal{S}_0|} = 1 - 1/k$$

By successively applying this result through k queries, we obtain the following:

$$\Pr\left[|\mathcal{S}_k| \geq \frac{|\mathcal{S}_0|}{(k \cdot (d+1))^k}\right] > (1 - 1/k)^k$$

Note that for any $k \geq 2$, $(1-1/k)^k \in [1/4, 1/e)$. Thus, with probability at least 1/4, we have a lower bound on the size of $|\mathcal{S}_k|$. We next show that if $|\mathcal{S}_k|$ is sufficiently large, then there is a significant probability that the player will fail if it halts and outputs a set at this point.

Let T be any set output by the player at the end of the game. For any i, $0 \leq i \leq N$, note that there are exactly $\binom{N}{i}$ sets $S \in 2^{[N]}$ such that $|S \triangle T| = i$. Thus, there are exactly $\left(\!\binom{N}{t}\!\right)$ sets $S \in 2^{[N]}$ such that $|S \triangle T| \leq t$. Now suppose that $|\mathcal{S}_k| \geq 2\left(\!\binom{N}{t}\!\right)$. Since the target set is equally likely to be any element of $\mathcal{S}_k$, the probability that T is an acceptable set is at most 1/2. Thus, the player will fail with probability at least 1/8 if it halts after k questions for any k which satisfies the following inequality:

$$2\left(\!\binom{N}{t}\!\right) \leq 2\left(\!\binom{N}{d/4}\!\right) \leq 2\left(\frac{eN}{d/4}\right)^{d/4} \leq \frac{\left(\frac{N}{d}\right)^d}{(k\cdot(d+1))^k} \leq \frac{\binom{N}{d}}{(k\cdot(d+1))^k} \leq |\mathcal{S}_k|$$

Solving the third inequality for $(k\cdot(d+1))^k$ and noting that $d \geq 4$, we have the following:

$$\begin{array}{rrcl} & (k\cdot(d+1))^k & \leq & \frac{1}{2}\left(\frac{N}{d}\right)^d\left(\frac{d/4}{eN}\right)^{d/4} \\ \Leftarrow & (k\cdot(d+1))^k & \leq & \left(\frac{N}{4ed}\right)^{3d/4} \\ \Leftrightarrow & k\log k + k\log(d+1) & \leq & \frac{3d}{4}\log\left(\frac{N}{4ed}\right) \end{array}$$

The latter inequality is implied by the following two inequalities:

$$\begin{array}{rrcl} & k\log k & \leq & \frac{3d}{8}\log\left(\frac{N}{4ed}\right) \\ \Leftarrow & k & \leq & \frac{\frac{3d}{8}\log\frac{N}{4ed}}{\log(\frac{3d}{8}\log\frac{N}{4ed})} \\ & & & \\ & k\log(d+1) & \leq & \frac{3d}{8}\log\left(\frac{N}{4ed}\right) \\ \Leftrightarrow & k & \leq & \frac{\frac{3d}{8}\log\frac{N}{4ed}}{\log(d+1)} \end{array}$$

Each of these inequalities is implied by

$$k = \frac{\frac{3d}{8}\log\frac{N}{4ed}}{\log((d+1)\log\frac{N}{4ed})} = \Omega\left(\frac{d\log N}{\log(d\log N)}\right)$$

for $N = \Omega(d^{1+\alpha})$. $\square$

Combining this result with the proof of Theorem 5, we immediately obtain the following:

Theorem 21 *There exists a parameterized family of function classes which require $\Omega(\frac{d\log(1/\epsilon)}{\log(d\log(1/\epsilon))})$ queries with additive error $O(\epsilon)$ to learn in the SQ model by a probabilistic algorithm.*

Part II

Searching in the Presence of Errors

CHAPTER 7

Introduction

Coping with errors during computation has been a subject of long-standing interest. It has motivated research in such areas as error-correcting codes, fault-tolerant networks, boolean circuit evaluation with faulty gates, and learning in the presence of errors. In the following chapters, we focus on the problem of searching in the presence of errors.

7.1 Introduction

Our goal is to find an unknown quantity x in a previously specified, discrete, but not necessarily finite, domain by asking "yes-no" questions, when some questions are answered incorrectly. We show that it is possible to cope with errors whose number may grow linearly with the number of questions asked, and, depending on the class of questions allowed, to do so with an asymptotically optimal number of questions. Examining both adversarial and random errors, we find that even in a fairly restricted adversarial error model, searching is at least as difficult as in the random error model.

The problem can be further qualified by:

- Kinds of questions that may be asked.
  - **Comparison questions**: "Is x less than y?"

  - **Membership questions**: "Is x in the set S?", where S is some subset of the domain.

- Kinds of errors possible.
  - **Constant number**: It is known *a priori* that there will be at most k errors, where k is some fixed constant.
  - **Probabilistic**: The answer to each question is erroneous independently with some probability p, $0 < p < \frac{1}{2}$.
  - **Linearly Bounded**: For some constant r, $0 < r < \frac{1}{2}$, any initial sequence of i answers has at most ri errors. This model allows the answers to be erroneous in a malicious way. Unlikely scenarios in the probabilistic model (such as a long sequence of correct answers followed by a short sequence of false ones) must be dealt with here.
- Domain of the quantity being sought.
  - **Bounded**: $x \in \{1, \ldots, n\}$, for some known n.
  - **Unbounded**: x may be any positive integer.

Much research has been devoted to the subject of searching in the presence of errors. Rivest *et al.* [25] have shown that in the bounded domain with at most k errors, x can be determined exactly with $\lg n + k \lg\lg n + O(k \lg k)$ comparison questions.[1] Here k can be a function of n, but not of the number of questions asked. When k is a constant, this is an asymptotically optimal bound since $\lceil \lg n \rceil$ is a lower bound on the number of questions needed to search even without errors. Naturally, this bound also applies to searching with membership questions, since comparison questions are a restricted version of membership questions.

In the probabilistic error model, where errors occur randomly and independently with probability p, we must find the correct x with probability of failure at most δ. Since δ is previously known and fixed, we consider δ a constant for the purpose of measuring the complexity of the searching algorithm.[2] Pelc [24] showed that in the probabilistic error model, with error probability $p < 1/2$, $O(\lg^2 n)$ questions are sufficient to search in the bounded domain. Frazier [13]

[1] The term $\lg n$ denotes $\log_2 n$ throughout this thesis.
[2] Typically, the complexity of such algorithms depends on $\log(1/\delta)$, as does the complexity of our algorithm.

	Membership Questions		Comparison Questions	
	$0 < r < \frac{1}{3}$	$\frac{1}{3} \leq r < \frac{1}{2}$	$0 < r < \frac{1}{3}$	$\frac{1}{3} \leq r < \frac{1}{2}$
Pelc	$O(\lg n)$	$O(n^{\lg \frac{1}{1-2r}})$	$O(\lg n)$	$O(n^{\lg \frac{1}{1-2r}})$
Thesis	$O(\lg n)$		$O(n^{\lg \frac{1}{1-r}})$	

Figure 7.1: Bounds for searching in the bounded domain with linearly bounded errors. Here n is a bound on the number being sought.

improved the bound to $O(\lg n \lg \lg n)$ questions using a somewhat complicated analysis. Finally, using standard Chernoff bound techniques, Feige *et al.* [12] showed that $O(\lg n)$ questions are sufficient for any $p < 1/2$. Our contribution here is a formal reduction from the problem of searching in the probabilistic error model to that of searching in the linearly bounded error model. To state this result informally, we show that an algorithm for searching in the presence of linearly bounded errors can be transformed into an algorithm for searching in the presence of random errors. In this sense, searching with linearly bounded errors is at least as difficult as searching with random errors. When we are allowed to ask membership questions, this reduction together with the results from the linearly bounded error model mentioned below matches the Feige *et al.* $O(\lg n)$ bound in the bounded domain. We also generalize this bound to the unbounded domain.[3]

In the linearly bounded error model, Pelc [24] showed that x can be determined exactly in $O(\lg n)$ questions in both the bounded and unbounded domains. However, these bounds only hold for $r < 1/3$. The best known bound using comparison or membership questions in the bounded domain for $1/3 \leq r < 1/2$ was $O(n^{\lg \frac{1}{1-2r}})$. Note that the degree of the polynomial in this bound is unbounded as r approaches $1/2$. This bound comes from an analysis of a "brute-force" binary search, where each question of the search is asked enough times so that the correct answer can be determined by majority. A simple argument [13, 32] shows that the search problem cannot be solved (with either membership or comparison questions) if $r \geq 1/2$.

We show significantly improved bounds in the linearly bounded error model which hold for the entire range $0 < r < 1/2$. With memberships questions, we show that x can be determined exactly in $O(\lg n)$ questions in both the bounded and unbounded domains. These bounds are

[3]In the unbounded domain, n now refers to the unknown number.

	Membership Questions		Comparison Questions	
	$0 < r < \frac{1}{3}$	$\frac{1}{3} \le r < \frac{1}{2}$	$0 < r < \frac{1}{3}$	$\frac{1}{3} \le r < \frac{1}{2}$
Pelc	$O(\lg n)$	$O(n^{\lg \frac{1}{1-2r}})$	$O(\lg n)$	$O(n^{\lg \frac{1}{1-2r}})$
Thesis	$O(\lg n)$		$O([n \lg^2 n]^{\lg \frac{1}{1-r}})$	

Figure 7.2: Bounds for searching in the unbounded domain with linearly bounded errors. Here n is the unknown number.

tight since searching has a trivial $\Omega(\lg n)$ lower bound. With comparison questions, we improve the bounds to $O(n^{\lg \frac{1}{1-r}}) = o(n)$ questions for the bounded domain and $O([n \lg^2 n]^{\lg \frac{1}{1-r}}) = o(n)$ in the unbounded domain. A comparison of this work with the best known previous results can be found in Figures 7.1 and 7.2. Our results are obtained by looking at the search problem in the framework of chip games. These chip games have also proved useful in modeling a hypergraph 2-coloring problem [5]. In general, chip games model computational problems in such a way that winning strategies for the players translate into bounds on the critical resource. This critical resource is represented by some aspect of the chip game, such as number of chips used or number of moves in the game.

Spencer and Winkler [32] have also examined this problem. They have arrived independently at one of the theorems in this paper using different proof techniques. Their paper as well as one by Dhagat, Gács, and Winkler [10] considers another linearly bounded model of errors.

We begin in Section 7.2 by developing the framework of chip games within which we solve the search problem. Chapter 8 begins with a simple strategy for solving our problem in the linearly bounded model in the bounded domain which works with either comparison or membership questions, but whose obvious analysis gives an inefficient bound on the number of questions. We then improve this bound by analyzing this strategy using chip games. Chapter 8 continues by focusing on membership questions and proving an $O(\lg n)$ question bound for this class. The chapter ends with a generalization of the above bounds for the unbounded domain. Chapter 9 contains the aforementioned reduction between the probabilistic and linearly bounded error models in the bounded domain, and the $O(\lg n)$ question bound for the probabilistic error model which follows from it. These results are also generalized to the unbounded domain. Chapter 10 concludes the paper with a summary of the results and mention of some open

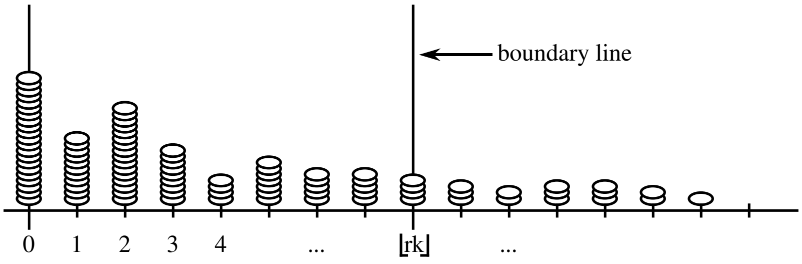

Figure 7.3: Chip Game

problems.

7.2 Searching and Games

Searching for an unknown number x in $\{1, \ldots, n\}$ by asking "yes-no" questions can be restated in terms of the game of "Twenty Questions". In this game between two players, whom we denote Paul and Carole,[4] Carole thinks of a number between 1 and n. Paul must guess this number after asking some number of "yes-no" questions which is previously fixed. Our goal in this game is to determine how many questions Paul must be allowed in order for him to have a winning strategy in the game. Clearly, $\lceil \lg n \rceil$ questions are sufficient if Carole always answers truthfully. The problem of searching with errors thus translates into playing "Twenty Questions" with a liar [33]. Corresponding to the aforementioned error models, we consider both a probabilistic and an adversarial linearly bounded liar.

The game against a linearly bounded liar can now be further reformulated as a Chip Game between two players: the **Pusher** and the **Chooser**. Pusher-Chooser games were first used by Spencer [31] to solve a different problem in his notes on the probabilistic method. The Chip Game starts with a unidimensional board marked in levels from 0 on upwards (see Figure 7.3). We start with n chips on level 0, each chip representing one number in $\{1, \ldots, n\}$. At each step, the Pusher selects some chips from the board. These chips correspond to the subset S of $\{1, \ldots, n\}$ that Paul wants to ask about. In other words, selecting S is tantamount to asking "Is $x \in S$?". The Chooser then either moves the set of chips picked by the Pusher to the next level, indicating a "no" answer from Carole (x is *not* in S), or it moves the set of chips not

[4]An anagram for the word "oracle," as this is her role in the game.

picked by the Pusher, indicating a "yes" answer from Carole. Therefore, a chip representing the number y is moved to the right if and only if Carole says that y is not the answer. The presence of a chip representing the number y at level i says that if y is the unknown number x, then there have been i lies in the game. After some k steps, if a chip is at any level greater than $\lfloor rk \rfloor$, then it may be thrown away since the corresponding number cannot possibly be the answer (too many lies will have occurred). To win, the Pusher must eliminate all but one chip from the board.

To clarify which chips may be thrown away, we maintain a boundary line on the board. After k steps, the boundary line will be at level $\lfloor rk \rfloor$. Thus the Pusher may dispose of the chips at levels to the right of the boundary line at any time. Note that the boundary line moves one level to the right after approximately $1/r$ steps. The number of questions that we need to ask to determine x exactly is the same as the number of steps needed for the Pusher to win the above Chip Game.

C H A P T E R 8

The Linearly Bounded Error Model

In this chapter, we show an $O(n^{\lg \frac{1}{1-r}})$ question bound for searching with comparison questions and an $O(\lg n)$ question bound for searching with membership questions. We first show an $\Omega(n^{\lg \frac{1}{1-2r}})$ lower bound for a "brute-force" strategy. Strategies similar to this "brute-force" method are given by Pelc [24] and Frazier [13], and these were the best known results for $1/3 \leq r < 1/2$ prior to this work.

8.1 A Brute-Force Strategy

To determine an unknown number $x \in \{1, \ldots, n\}$, a "brute-force" strategy simply performs a binary search, repeating each question enough times so that majority gives the correct answer. Let $q(i)$ be the number of times question i is repeated, and let $Q(i)$ be the total number of queries through question i ($Q(i) = \sum_{j=1}^{i} q(j)$). To guarantee that majority gives the correct answer, we insure that the number of lies the malicious oracle can tell is less than half the number of times question k is repeated. We thus obtain the following:

$$\begin{aligned} r(Q(k-1) + q(k)) &< q(k)/2 \\ q(k) &> \frac{2r}{1-2r} Q(k-1) \end{aligned}$$

We now use the fact that $Q(k) = Q(k-1) + q(k)$:

$$\begin{aligned} Q(k) &= Q(k-1) + q(k) \\ &> Q(k-1) + \frac{2r}{1-2r} Q(k-1) \\ &= \frac{1}{1-2r} Q(k-1) \end{aligned}$$

Thus, $Q(k) = \Omega((\frac{1}{1-2r})^k)$. Since the correct answers to $\lceil \lg n \rceil$ binary search questions must be obtained, we obtain the following lower bound for the "brute-force" strategy:

$$\begin{aligned} \Omega((\tfrac{1}{1-2r})^{\lceil \lg n \rceil}) &= \Omega((\tfrac{1}{1-2r})^{\lg n}) \\ &= \Omega(n^{\lg \frac{1}{1-2r}}) \end{aligned}$$

A similar upper bound can also be shown for this strategy. While this strategy is sound, its naive analysis yields an unsatisfactory bound. We can improve on this significantly through the use of chip games.

8.2 Searching with Comparison Questions

We now consider an essentially identical strategy in the chip game. The Pusher plays this game in phases. Each phase corresponds to getting the correct answer to a single question. At the beginning of each phase there is a single stack of chips somewhere on the board. The Pusher picks a subset of these chips, generally some half of them which corresponds to a comparison question whose correct answer he wishes to determine. He continues picking the same half-stack throughout this phase until either it or the other half-stack moves beyond the boundary line. Then he begins the next phase with the remaining half-stack. This continues until there is only one chip remaining on the board to the left of the boundary line. Note that if there are m chips on the board initially, $\lceil \lg m \rceil$ questions need to be answered correctly.

Now consider the board before and after some phase j. At the beginning of phase j, there is a stack of chips at a level some distance l_j away from the boundary line (see Figure 8.1). At the end of phase j, one half-stack has moved some distance i from its original position and the other half-stack has moved one position past the boundary line. The boundary line is now at

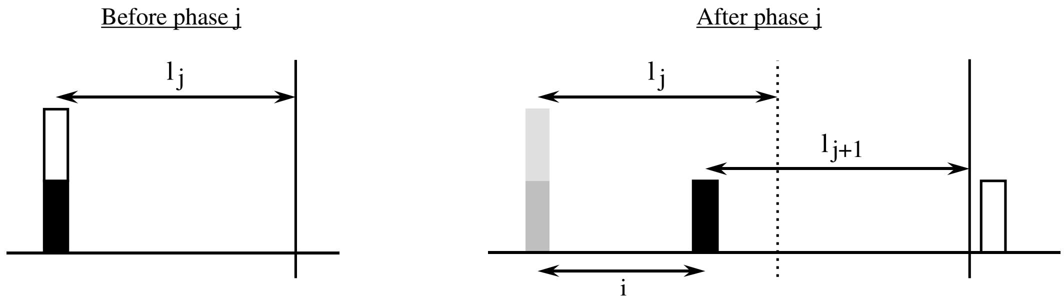

Figure 8.1: Chips before and after phase j.

some distance l_{j+1} from the first half-stack.

Let $T(d, l)$ be the number of steps the Pusher takes to have d questions answered correctly when a single stack of chips on the board is a distance l away from the boundary line. We then have

$$T(d, l_j) = T(d-1, l_{j+1}) + (\text{steps during phase } j).$$

We next bound the number of steps in phase j as a function of r, l_j and l_{j+1}.

Lemma 13 *The total number of steps during phase j is less than $\frac{2l_j - l_{j+1} + 3}{1-2r}$.*

Proof: The total number of steps during phase j is equal to the total number of levels the two half-stacks move. One half-stack moves i levels and the other moves $i + l_{j+1} + 1$ levels, and thus the total number of steps in phase j is $2i + l_{j+1} + 1$.

Let s_j be the total number of steps prior to phase j, and let p_j be the position of the boundary line prior to phase j. Note that $p_j = \lfloor rs_j \rfloor$. Using the fact that $x - 1 < \lfloor x \rfloor \leq x$, we have the following:

$$\begin{aligned} p_{j+1} &\leq rs_{j+1} \\ p_j &> rs_j - 1 \end{aligned}$$

Subtracting these two inequalities, we obtain:

$$p_{j+1} - p_j < r(s_{j+1} - s_j) + 1$$

Now, $s_{j+1} - s_j$ is the total number of steps during phase j, and $p_{j+1} - p_j$ is the number of levels the boundary line moves. We therefore have the following inequality

$$i + l_{j+1} - l_j < r(2i + l_{j+1} + 1) + 1 \tag{8.1}$$

which implies that

$$i < \frac{l_j - (1-r)l_{j+1} + 1 + r}{1-2r}.$$

Substituting this bound on i into the expression $2i + l_{j+1} + 1$, we obtain the desired result. □

Now we are ready to show that:

Theorem 22 $T(d, l_0) = \begin{cases} O(l_0(\frac{1}{1-r})^d) & \textit{if } l_0 > 0 \\ O((\frac{1}{1-r})^d) & \textit{if } l_0 = 0 \end{cases}$

Proof: We first show that $\forall j,\ l_{j+1} < \frac{l_j+1+r}{1-r}$. Solving Inequality 8.1 for l_{j+1}, we obtain:

$$\begin{aligned} l_{j+1} &< \frac{l_j + 1 + r + i(2r-1)}{1-r} \\ &\leq \frac{l_j + 1 + r}{1-r} \end{aligned}$$

This last inequality holds due to the fact that $2r - 1 < 0$ and $i \geq 0$. Successively applying this inequality, we obtain the following:

$$\begin{aligned} l_j &< \frac{l_{j-1} + 1 + r}{1-r} \\ &< \frac{l_0}{(1-r)^j} + (1+r)\sum_{i=1}^{j}\left(\frac{1}{1-r}\right)^i \\ &= \frac{l_0}{(1-r)^j} + \frac{1+r}{r}\left[\left(\frac{1}{1-r}\right)^j - 1\right] \\ &= \left(l_0 + \frac{1+r}{r}\right)\left(\frac{1}{1-r}\right)^j - \frac{1+r}{r} \end{aligned}$$

We can now use this fact to obtain a bound on $T(d, l_0)$:

$$\begin{aligned} T(d, l_0) &< T(d-1, l_1) + \frac{2l_0 - l_1 + 3}{1-2r} \\ &= \frac{2l_0 - l_1 + 3}{1-2r} + \frac{2l_1 - l_2 + 3}{1-2r} + \cdots + \frac{2l_{d-1} - l_d + 3}{1-2r} \end{aligned}$$

$$\begin{aligned}
&\le \frac{1}{1-2r}[2l_0 + l_1 + l_2 + \cdots + l_{d-1} + 3d] \\
&< \frac{1}{1-2r}\left[2l_0 + 3d + \sum_{i=1}^{d-1}\left[\left(l_0 + \frac{1+r}{r}\right)\left(\frac{1}{1-r}\right)^i - \frac{1+r}{r}\right]\right] \\
&= \frac{1}{1-2r}\left[2l_0 + 3d + \frac{1}{r}\left(l_0 + \frac{1+r}{r}\right)\left(\left(\frac{1}{1-r}\right)^{d-1} - 1\right) - (d-1)\frac{1+r}{r}\right]
\end{aligned}$$

For any constant r, this expression is $O(l_0(\frac{1}{1-r})^d)$ if $l_0 > 0$ or $O((\frac{1}{1-r})^d)$ if $l_0 = 0$. □

This result will be used throughout this thesis. In particular, consider the problem of searching in the bounded domain with comparison questions. The corresponding chip game begins with n chips and the boundary line at level 0. Using binary search, we require the correct answers to $\lceil \lg n \rceil$ questions. Employing Theorem 22, we immediately obtain:

Theorem 23 *The problem of searching in the linearly bounded error model in the bounded domain $\{1, \ldots, n\}$ with comparison questions and error constant r, $0 < r < 1/2$, can be solved with $O(n^{\lg \frac{1}{1-r}})$ questions.*

8.3 Searching with Membership Questions

We show a winning strategy for the Pusher which requires $O(\lg n)$ steps. The strategy works in three stages. In Stage 1, the Pusher eliminates all but $O(\lg n)$ chips from the board in $O(\lg n)$ steps. In Stage 2, the Pusher eliminates all but $O(1)$ chips from the board in an additional $O(\lg n)$ steps. In Stage 3, the Pusher removes all but one chip from the board in the final $O(\lg n)$ steps.

8.3.1 Stage 1

The strategy employed during Stage 1 is simple. We describe it inductively on the number of steps as follows. Let $h_m(i)$ be the height of the stack of chips at level i after m steps. In the $(m+1)$-st step, the Pusher picks $\left\lfloor \frac{h_m(i)}{2} \right\rfloor$ from each stack of chips at all levels i. He continues this way for $c_1 \lg n$ steps (where c_1 is a constant that will be determined in the analysis).

Before we can analyze this strategy, we will need a few definitions. Define *normalized* *binomial coefficients* $b_m(i)$ as

$$b_m(i) = \frac{n}{2^m}\binom{m}{i}$$

and let

$$\Delta_m(i) = h_m(i) - b_m(i).$$

The normalized binomial coefficient $b_m(i)$ will approximate $h_m(i)$, the height of the stack at level i after m steps, while $\Delta_m(i)$ will account for any discrepancy.

In order to analyze the given strategy, we need to be able to determine the number of chips which are to left of the boundary line after some number of steps in our strategy. After m steps, this is equivalent to $\sum_{i\leq\lfloor rm\rfloor} h_m(i)$ (since r is the rate at which the boundary line moves). This sum is difficult to determine exactly. Instead, we will derive an upper bound for it by using the fact that $\sum_{i\leq\lfloor rm\rfloor} h_m(i) = \sum_{i\leq\lfloor rm\rfloor} b_m(i) + \sum_{i\leq\lfloor rm\rfloor} \Delta_m(i)$. In particular, we will show an upper bound for $\sum_{i\leq\lfloor rm\rfloor} \Delta_m(i)$.

For the strategy given above, we now bound the discrepancy between the actual number of chips in any initial set of j stacks and the number of chips predicted by the normalized binomial coefficients. We will need three lemmas. The first two lemmas handle boundary conditions, while the third is required in the proof of the main theorem.

Lemma 14 $(\forall m \geq 0)$, $\Delta_m(0) \leq 1$.

Proof: The proof is by induction on m.

- **base case:** For $m = 0$, $h_0(0) = n = b_0(0) \Longrightarrow \Delta_0(0) = 0$.

- **inductive step:** Assume $\Delta_{m-1}(0) \leq 1$. We now have the following:

$$\begin{aligned}
h_m(0) &\leq \left\lceil \frac{h_{m-1}(0)}{2} \right\rceil \\
&\leq \frac{h_{m-1}(0)}{2} + \frac{1}{2} \\
&= \frac{b_{m-1}(0)}{2} + \frac{\Delta_{m-1}(0)}{2} + \frac{1}{2} \\
&= b_m(0) + \frac{\Delta_{m-1}(0)}{2} + \frac{1}{2} \\
&\leq b_m(0) + 1
\end{aligned}$$

Thus, $\Delta_m(0) = h_m(0) - b_m(0) \leq 1$. □

Lemma 15 *$(\forall m \geq 0)$, $\sum_{i=0}^{m} \Delta_m(i) = 0$.*

Proof:

$$\sum_{i=0}^{m} h_m(i) = n = \sum_{i=0}^{m} b_m(i) \Longrightarrow \sum_{i=0}^{m} \Delta_m(i) = 0$$

□

Lemma 16 *$\sum_{i=0}^{j-1} b_{m-1}(i) + \frac{b_{m-1}(j)}{2} = \sum_{i=0}^{j} b_m(i)$.*

Proof: We first note the fact that $\binom{a}{b-1} + \binom{a}{b} = \binom{a+1}{b}$. The proof proceeds as follows:

$$\begin{aligned}
\sum_{i=0}^{j-1} b_{m-1}(i) + \frac{b_{m-1}(j)}{2} &= \sum_{i=0}^{j-1} \frac{n}{2^{m-1}} \binom{m-1}{i} + \frac{n}{2^m} \binom{m-1}{j} \\
&= \frac{n}{2^m} \left[\sum_{i=0}^{j-1} 2 \cdot \binom{m-1}{i} + \binom{m-1}{j} \right] \\
&= \frac{n}{2^m} \left[\binom{m-1}{0} + \left[\binom{m-1}{0} + \binom{m-1}{1} \right] + \cdots \right. \\
&\quad \left. + \left[\binom{m-1}{j-1} + \binom{m-1}{j} \right] \right] \\
&= \frac{n}{2^m} \left[\binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{j} \right] \\
&= \frac{n}{2^m} \sum_{i=0}^{j} \binom{m}{i} = \sum_{i=0}^{j} b_m(i)
\end{aligned}$$

□

Theorem 24 *$(\forall m \geq 0)$ $(\forall j \leq m)$, $\sum_{i=0}^{j} \Delta_m(i) \leq j + 1$.*

Proof: The proof of the theorem is by induction on m. The base case of $m = 0$ is trivial. In the inductive step, we show that if the theorem holds for $m - 1$, then the theorem holds for m. The boundary conditions $j = 0$ and $j = m$ are handled by Lemmas 14 and 15. We concentrate on the case $0 < j < m$ below. Consider the following (see Figure 8.2):

$$\sum_{i=0}^{j} h_m(i) \quad \leq \quad \sum_{i=0}^{j-1} h_{m-1}(i) + \left\lceil \frac{h_{m-1}(j)}{2} \right\rceil$$

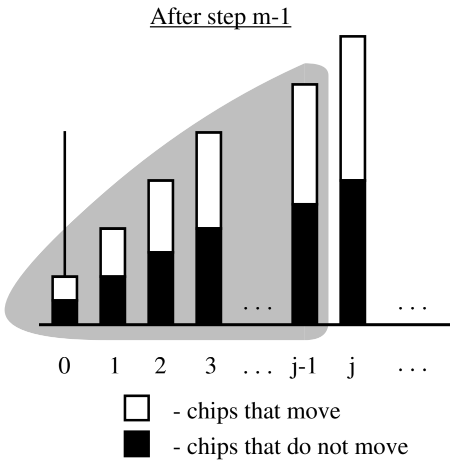

Figure 8.2: Chips before and after step m

$$\begin{aligned}
&\le \sum_{i=0}^{j-1} h_{m-1}(i) + \frac{h_{m-1}(j)}{2} + \frac{1}{2} \\
&= \sum_{i=0}^{j-1} b_{m-1}(i) + \frac{b_{m-1}(j)}{2} + \sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2} \\
&= \sum_{i=0}^{j} b_m(i) + \sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2}
\end{aligned}$$

We now bound the quantity $\sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2}$. There are two cases, depending upon whether $\Delta_{m-1}(j) \le 1$ or $\Delta_{m-1}(j) > 1$. If $\Delta_{m-1}(j) \le 1$, we have the following:

$$\begin{aligned}
\sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2} &\le \sum_{i=0}^{j-1} \Delta_{m-1}(i) + 1 \\
&\le j + 1
\end{aligned}$$

If $\Delta_{m-1}(j) > 1$, then $\frac{\Delta_{m-1}(j)}{2} + \frac{1}{2} < \Delta_{m-1}(j)$. We thus obtain the following:

$$\begin{aligned}
\sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2} &< \sum_{i=0}^{j} \Delta_{m-1}(i) \\
&\le j + 1
\end{aligned}$$

We therefore have

$$\begin{aligned}\sum_{i=0}^{j} h_m(i) &\leq \sum_{i=0}^{j} b_m(i) + \sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2} \\ &\leq \sum_{i=0}^{j} b_m(i) + j + 1\end{aligned}$$

which implies that $\sum_{i=0}^{j} \Delta_m(i) \leq j+1$. □

Now we will bound $\sum_{i \leq \lfloor rm \rfloor} b_m(i)$. We will find a constant c_1 such that for $m = c_1 \lg n$, $\sum_{i=0}^{\lfloor rm \rfloor} b_m(i)$ is a constant. If we can do this, then it follows from the theorem above that $\sum_{i=0}^{\lfloor rm \rfloor} h_m(i)$, the number of chips remaining to the left of the boundary line, is $O(\lg n)$. The reasoning is as follows:

$$\begin{aligned}\sum_{i=0}^{\lfloor rm \rfloor} h_m(i) &= \sum_{i=0}^{\lfloor rm \rfloor} b_m(i) + \sum_{i=0}^{\lfloor rm \rfloor} \Delta_m(i) \\ &\leq \sum_{i=0}^{\lfloor rm \rfloor} b_m(i) + \lfloor rm \rfloor + 1 \\ &= c_2 + \lfloor r \cdot c_1 \lg n \rfloor \\ &\leq c_3 \lg n\end{aligned}$$

for appropriate constants c_2 and c_3.

In order to determine c_1, we make use of the following bound [22]:

$$\sum_{i=0}^{\lfloor \mu k \rfloor} \binom{k}{i} \leq 2^{kH(\mu)}$$

where $0 < \mu < 1/2$ and $H(r)$ is the binary entropy function.[1] We now have:

$$\begin{aligned}\sum_{i=0}^{\lfloor rm \rfloor} b_m(i) &= \frac{n}{2^m} \sum_{i=0}^{\lfloor rm \rfloor} \binom{m}{i} \\ &\leq \frac{n}{2^m} 2^{mH(r)}\end{aligned}$$

[1] $H(r) = -r \lg r - (1-r) \lg(1-r)$

$$= \quad n2^{m(H(r)-1)}$$

This last quantity is $O(1)$ when $m = \frac{\lg n}{1-H(r)}$. Thus if we pick $c_1 = \frac{1}{1-H(r)}$, then after $m = c_1 \lg n$ steps, there will be at most $c_3 \lg n$ chips remaining on the board to the left of the boundary line. The strategy in this stage can also be applied to the game where the boundary line starts out at level $l = O(\lg n)$ instead of at $l = 0$. One can show directly or through the use of the techniques given in Sections 9.1.1 and 9.1.2 that Stage 1 still ends in $O(\lg n)$ steps with at most $O(\lg n)$ chips to the left of the boundary line. This fact will be useful when we examine the unbounded domain.

8.3.2 Stage 2

At the end of Stage 1 we are left with some $c_2 \lg n$ chips on the board with the boundary line at level $c_1 \lg n$ (for appropriate constants c_1 and c_2). After Stage 2, there are $O(1)$ chips on the board to the left of the boundary line after $O(\lg n)$ additional steps.

Before starting Stage 2, we alter the board by moving everything on the board (chips *and* boundary line) to the right by $c_2 \lg n$, so that the boundary line is now at level $(c_1 + c_2) \lg n = c \lg n$. While this new board corresponds to a different game than the one we have played until now (it corresponds to a game in which many more questions and lies have occurred), these two boards are equivalent in the sense that the Pusher can win from the first board within k extra moves if and only if he can win from the second board within k extra moves.

Now move the chips to the left in such a way that there is exactly one chip on each of the first $c_2 \lg n$ levels. Note that the Pusher does not help himself by doing this, since moving chips to the left is in effect ignoring potential lies which he has discovered.

At each step in this stage, the Pusher first orders the chips from left to right, ordering chips on the same level arbitrarily. He then picks every other chip according to this order; that is, he picks the 1st, 3rd, 5th, ... chips. We say that the board is in a **nice** state if no level has more than two chips.

Lemma 17 *Throughout Stage 2, the board is in a* **nice** *state.*

Proof: We show this by induction on the number of steps in Stage 2. Certainly at the beginning of Stage 2, the board is in a **nice** state since each level is occupied by at most one chip. Now suppose that the board is in a **nice** state after i steps. Consider any level j after the $(i+1)$-st step. Since both levels $j-1$ and j had at most two chips before the $(i+1)$-st step, after this step level j retains at most one chip and gains at most one chip, thus ending with at most two chips. □

We now show that after $O(\lg n)$ steps, there are at most $2k$ chips remaining to the left of the boundary line. Here k is a constant (depending only on r) which will be determined later. If there are fewer than $2k$ chips to the left of boundary line, Stage 2 terminates. Let the weight of a chip be the level it is on, and let the weight of the board be the weight of its $2k$ leftmost chips.

Lemma 18 *After each step in Stage 2, the weight of the board increases by at least $k-1$.*

Proof: Of the $2k$ leftmost chips after step i, at least $(2k-1)$ chips remain in the set of leftmost $2k$ chips after step $i+1$. (The $2k$-th chip may be on the same level as the $(2k+1)$-st chip. In this case, if the $2k$-th chip moves in step $i+1$, then the $(2k+1)$-st chip becomes the new $2k$-th chip in the revised ordering.) At least $\left\lfloor \frac{2k-1}{2} \right\rfloor = k-1$ of these chips move to the right one level during step $i+1$, thus increasing the weight of the board by at least $k-1$. □

Let S be the number of steps taken during this stage and let W be the weight of the board at the end of these S steps. Since the weight of the board goes up by at least $k-1$ in each step, and since the initial weight of the board was non-negative, $W \geq (k-1)S$. At the end of the S steps, the boundary line is at $c \lg n + \lfloor rS \rfloor$. Since this stage ends when there are fewer than $2k$ chips to the left of the boundary line, we certainly have $W \leq 2k(c \lg n + rS)$. Combining these two inequalities, we obtain:

$$\begin{aligned} 2k(c \lg n + rS) &\geq S(k-1) \\ S &\leq \frac{2kc}{k-1-2kr} \lg n \end{aligned}$$

If we let $k = \frac{2}{1-2r}$, then $S \leq \frac{4c}{1-2r} \lg n = O(\lg n)$. Thus after $O(\lg n)$ steps, Stage 2 ends leaving at most $2k$ chips to the left of the boundary line.

8.3.3 Stage 3

At the beginning of Stage 3, the Pusher moves all of the remaining chips to level 0. Again this is legal, since he is essentially choosing to ignore some information he has gathered. We now have some $2k$ chips a distance $c \lg n$ away from the boundary line (for appropriate constants c and k). By applying Theorem 22 from Section 8.2, the Pusher can win this game in $O[(c \lg n) \cdot (\frac{1}{1-r})^{\lceil \lg 2k \rceil}] = O(\lg n)$ steps. Since each of the three stages takes $O(\lg n)$ steps, we now have the following:

Theorem 25 *The problem of searching in the linearly bounded error model in the bounded domain $\{1, \ldots, n\}$ with membership questions and error constant r, $0 < r < \frac{1}{2}$, can be solved with $O(\lg n)$ questions.*

8.4 Unbounded Search

Now consider the problem of searching for a positive integer in the presence of errors as before, but where no upper bound on its size is known. Let this unknown integer be n. Using strategies developed in this paper already, we show that n can be found with $O(\lg n)$ membership questions and $O([n \lg^2 n]^{\lg \frac{1}{1-r}}) = o(n)$ comparison questions.

The search occurs in two stages. First, we determine a bound for the unknown number n. Second, given a bound on n, we employ the techniques for bounded searching given above.

8.4.1 Unbounded Search with Membership Questions

Consider the problem of bounding the unknown number n if all of the answers we receive are known to be correct. We could ask questions of the form "Is $x < 2^{2^i}$?". We would begin by asking "Is $x < 2^{2^0}$?". If the answer were "no", we would follow with "Is $x < 2^{2^1}$?", and so on. Since $n \leq 2^{2^{\lceil \lg \lg n \rceil}}$, we will obtain our first "yes" answer (and thus have a bound on n) after at most $\lceil \lg \lg n \rceil$ questions. We further note that our bound is not too large:

$$2^{2^{\lceil \lg \lg n \rceil}} < 2^{2^{\lg \lg n + 1}} = 2^{2 \lg n} = n^2$$

Employing the techniques and results of Section 8.2, we can use the above strategy in the presence of errors. We need the correct answers to $\lceil \lg \lg n \rceil$ questions. By Theorem 22, we can obtain these answers in

$$O((\tfrac{1}{1-r})^{\lceil \lg \lg n \rceil}) = O((\lg n)^{\lg \frac{1}{1-r}}) = o(\lg n)$$

questions.

Having found a bound for n, we have reduced our unbounded search problem to a bounded search problem. We can now apply our bounded search strategy of Section 8.3. It is important to note that since we have already asked $o(\lg n)$ questions, the boundary line will have moved to $o(\lg n)$. But recall that Stage 1 of our bounded search algorithm can tolerate the boundary line starting at $O(\lg n)$. Thus the Pusher can now start with all relevant chips at level 0 and boundary line at level $o(\lg n)$ and apply the bounded search strategy of Section 8.3. Since our bound on the unknown number n is at most n^2, we will finish this stage after $O(\lg(n^2)) = O(\lg n)$ questions. We can now claim the following:

Theorem 26 *The problem of searching in the linearly bounded error model in the unbounded domain with membership questions and error constant r, $0 < r < \frac{1}{2}$, can be solved with $O(\lg n)$ questions, where the number being sought is n.*

8.4.2 Unbounded Searching with Comparison Questions

We can employ techniques similar to those used above to solve the unbounded search problem using comparison questions. We first determine a bound on the unknown integer n using the strategy developed above. We thus bound the unknown number n by at most n^2 using $O((\lg n)^{\lg \frac{1}{1-r}})$ questions. Note that the boundary line will now be at $O((\lg n)^{\lg \frac{1}{1-r}})$.

Having bounded the unknown number n by at most n^2, we could simply use Theorem 22 directly. By performing a simple binary search, we will need correct answers to at most $\lceil \lg(n^2) \rceil$ questions. Using Theorem 22, we obtain an overall question bound of

$$O((\lg n)^{\lg \frac{1}{1-r}} \cdot (\tfrac{1}{1-r})^{\lceil \lg(n^2) \rceil}) = O([n^2 \lg n]^{\lg \frac{1}{1-r}}).$$

This can be improved, however, by adding an extra stage. After bounding the unknown number n by at most n^2, partition this bounded interval into exponentially growing subintervals $I_j = [2^j, 2^{j+1} - 1]\ \forall j \geq 0$. Note that there will be at most $\lceil \lg(n^2) \rceil$ such subintervals. To determine the correct subinterval, we perform a simple binary search on these subintervals requiring correct answers to $\lceil \lg \lceil \lg(n^2) \rceil \rceil$ questions. By Theorem 22, we will need

$$O((\lg n)^{\lg \frac{1}{1-r}} \cdot (\tfrac{1}{1-r})^{\lceil \lg \lceil \lg(n^2) \rceil \rceil}) = O([\lg^2 n]^{\lg \frac{1}{1-r}})$$

additional questions. Since our subintervals grew exponentially, the subinterval containing the unknown number n will be of size at most n. We can thus perform a final binary search on this subinterval and employ Theorem 22 to obtain an overall question bound of

$$O([\lg^2 n]^{\lg \frac{1}{1-r}} \cdot (\tfrac{1}{1-r})^{\lceil \lg n \rceil}) = O([n \lg^2 n]^{\lg \frac{1}{1-r}}) = o(n).$$

Theorem 27 *The problem of searching in the linearly bounded error model in the unbounded domain with comparison questions and error constant r, $0 < r < \frac{1}{2}$, can be solved with $O([n \lg^2 n]^{\lg \frac{1}{1-r}}) = o(n)$ questions, where the number being sought is n.*

CHAPTER 9

The Probabilistic Error Model

Recall that in the probabilistic error model, Carole lies randomly and independently with probability p, and Paul must determine the unknown number x correctly with probability at least $1 - \delta$, for a given $\delta > 0$. In this chapter we give a reduction to show that searching in the probabilistic error model is no more difficult than searching in the linearly bounded model. Formally, we show that if $\mathcal{A}_\ell$ is an algorithm which, given n and r, solves the linearly bounded error problem in $f(n,r)$ questions, then we can construct an algorithm $\mathcal{A}_p$ which solves the probabilistic error problem in $f(cn, \frac{1+2p}{4})$ questions where c is a constant depending on p and δ. An $O(\log n)$ bound for the probabilistic error model with membership questions follows easily from the results of the previous chapter. We also generalize our results to the unbounded domain.

9.1 The Reduction

The terms "algorithm" and "strategy" will be used somewhat interchangeably, since a winning strategy for the Pusher in the Chip Game corresponds to an algorithm to solve the search problem with errors, and vice versa.

We now construct an algorithm which solves the probabilistic error problem from an algorithm which solves the linearly bounded error problem. Let $\mathcal{A}_\ell$ be an algorithm which solves the linearly bounded error problem. $\mathcal{A}_\ell$ requires values for n and r, as well as access to an oracle whose errors are linearly bounded (an oracle which gives at most ri errors to any initial sequence of i questions). Algorithm $\mathcal{A}_\ell$ will ask $f(n, r)$ questions and will return the correct element $x \in \{1, \ldots, n\}$ with certainty. Let $\mathcal{A}_p$ be an algorithm which solves the probabilistic error problem. $\mathcal{A}_p$ requires values for n, p and δ, as well as access to an oracle whose errors are random (an oracle which lies randomly and independently with probability p). Algorithm $\mathcal{A}_p$ will ask $g(n, p, \delta)$ questions and will return the correct element $x \in \{1, \ldots, n\}$ with probability at least $1 - \delta$.

In order to solve a probabilistic error problem with a linearly bounded error algorithm, we must insure that the errors made by the probabilistic oracle fall within those allowed by the linearly bounded error algorithm (with high probability). One method to accomplish this is to set $r > p$. This assures that in the long run, with high probability, the number of lies told by the probabilistic oracle will be fewer than the number of lies the linearly bounded error algorithm can tolerate. The danger here lies at the beginning of the game where it is relatively likely that too many lies will be told, and hence the correct chip will be thrown out by the linearly bounded error algorithm. To overcome this difficulty, we must prevent the linearly bounded error algorithm from throwing out the correct chip in this critical stage.

We proceed in two stages. In the first stage, we play a modified game with excess chips in such a way as to *guarantee* that the correct chip is not thrown out until at least m questions have been asked of the probabilistic oracle. In the second stage, we find the correct chip among those remaining with high probability.

9.1.1 Stage 1

We begin the game by setting r midway between p and $1/2$. Thus, $r = \frac{1/2+p}{2} = \frac{1+2p}{4}$. This insures that the number of errors given by the probabilistic oracle will be fewer than the number of errors which can be tolerated by the linearly bounded error algorithm in the long run with high probability. To insure that the correct chip does not cross the boundary line before the m-th step, we begin the game with $n2^{\frac{1-r}{r}m}$ chips.

In the first critical $\frac{1-r}{r}m$ steps, we intercept algorithm $\mathcal{A}_\ell$'s queries to the oracle and answer them so as to maximize the number of chips which are left at level 0. We first note that after these $\frac{1-r}{r}m$ steps, the boundary line will be at $(1-r)m$. Second, since the number of chips at

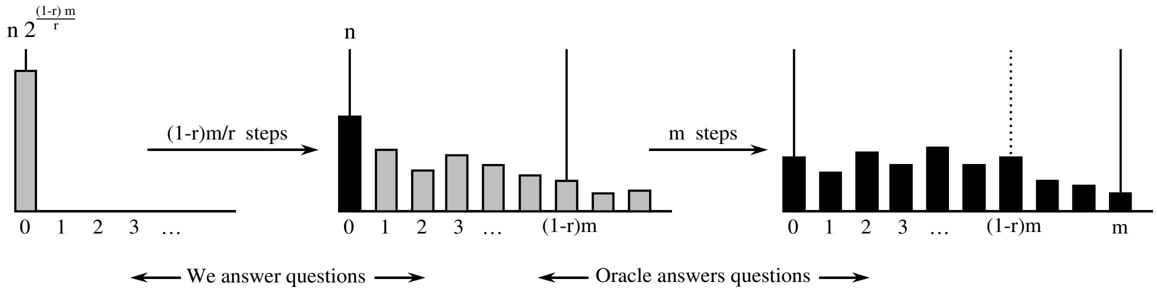

Figure 9.1: Stages of the Reduction

level 0 are reduced by at most half in each step, there will be at least n chips remaining after these $\frac{1-r}{r}m$ steps. See Figure 9.1.

9.1.2 Stage 2

Associated with each chip in the current game is an element of the set $\{1, \ldots, n2^{\frac{1-r}{r}m}\}$. For a chip $\mathtt{u}$, let this be the *OldValue*($\mathtt{u}$). We now establish a new correspondence between n of the remaining chips at level 0 and the set $\{1, \ldots, n\}$. For a chip $\mathtt{u}$, this will be *NewValue*($\mathtt{u}$). This new correspondence is *order-preserving* in the following sense: for chips $\mathtt{u}$ and $\mathtt{v}$, $OldValue(\mathtt{u}) < OldValue(\mathtt{v})$ iff $NewValue(\mathtt{u}) < NewValue(\mathtt{v})$. The necessity for establishing an *order-preserving* correspondence stems from the need to have this reduction apply to the searching problem where only comparison questions are allowed. We now continue running algorithm $\mathcal{A}_\ell$, sending his queries to the probabilistic oracle after translating them thus: Let $C = \{\mathtt{u} : \mathcal{A}_\ell \text{ picks } \mathtt{u} \text{ and } NewValue(\mathtt{u}) \in \{1, \ldots, n\}\}$. That is, C is the set of selected chips which have defined *NewValues*. Let $S_C = \{NewValue(\mathtt{u}) : \mathtt{u} \in C\}$, that is, S_C is the set of associated *NewValues*. If $S_C \neq \emptyset$, then we ask the probabilistic oracle about S_C and return the oracle's answer to $\mathcal{A}_\ell$. If $S_C = \emptyset$, then we could ourselves immediately answer "no". However, it is more convenient in the analysis to have the probabilistic oracle answer all questions in this stage. Thus, when $S_C = \emptyset$, we ask the probabilistic oracle about $\{1, \ldots, n\}$, and return the opposite of its answer to $\mathcal{A}_\ell$. Suppose that $\mathcal{A}_\ell$ finishes and returns chip $\mathtt{u}$. We then return *NewValue*($\mathtt{u}$) or "fail" if *NewValue*($\mathtt{u}$) is not defined.

9.1.3 The Analysis

We now claim that for an appropriate m, the above procedure will terminate with the correct value with probability at least $1 - \delta$. We in fact show that the probability that the "correct chip" *ever* crosses the boundary line is at most δ. If the correct chip *never* crosses the boundary line, then the linearly bounded error algorithm must return the correct chip when it terminates, and hence the correct answer will be obtained.

Our analysis makes use of Hoeffding's Inequality [18] to approximate the tail of a binomial distribution. Let $GE(p, m, n)$ be the probability of *at least* n successes in m Bernoulli trials, where each trial has probability of success p. Hoeffding's Inequality can then be stated as follows:

$$GE(p, m, (p + \alpha)m) \leq e^{-2\alpha^2 m}$$

After the first $\frac{1-r}{r}m$ steps, the correct chip will be at level 0, and the boundary line will be at level $(1 - r)m$ (see Figure 9.1). Since a chip can move at most one level per question and the boundary line moves at a rate r, none of the n remaining chips at level 0 will cross the boundary line until at least m questions have been asked of the probabilistic oracle. For any $j \geq m$, the probability that the correct chip is past the boundary line after j questions have been asked of the oracle is given by $GE(p, j, jr + (1 - r)m)$. The probability that the correct chip is *ever* past the boundary line is therefore at most

$$\sum_{j=m}^{\infty} GE(p, j, jr + (1 - r)m).$$

Given that:

- If $n > n'$ then $GE(p, m, n) < GE(p, m, n')$
- $r = \frac{1+2p}{4} = p + \frac{1-2p}{4}$

we can apply Hoeffding's Inequality:

$$\begin{aligned}
\sum_{j=m}^{\infty} GE(p, j, jr + (1 - r)m) &< \sum_{j=m}^{\infty} GE(p, j, jr) \\
&= \sum_{j=m}^{\infty} GE(p, j, (p + \tfrac{1-2p}{4})j)
\end{aligned}$$

$$\leq \sum_{j=m}^{\infty} e^{-2(\frac{1-2p}{4})^2 j}$$

Since we would like this sum to be at most δ, we can now solve for m. Let $\gamma = e^{-2(\frac{1-2p}{4})^2}$. Note that $\gamma < 1$.

$$\begin{aligned} \sum_{j=m}^{\infty} \gamma^j &\leq \delta \\ \frac{\gamma^m}{1-\gamma} &\leq \delta \\ m &\geq \frac{\ln\frac{1}{\delta} + \ln\frac{1}{1-\gamma}}{\ln\frac{1}{\gamma}} \end{aligned}$$

For $\gamma = e^{-2(\frac{1-2p}{4})^2} = e^{-\frac{(1-2p)^2}{8}}$, we obtain

$$m \geq \frac{8\ln\frac{1}{\delta} + 8\ln[1/(1-e^{-\frac{(1-2p)^2}{8}})]}{(1-2p)^2}.$$

Noting that for $0 < p < \frac{1}{2}$, $0 < \frac{(1-2p)^2}{8} < \frac{1}{8}$, we can use the fact that for $0 < x \leq \frac{1}{8}$, $\frac{16/15}{x} > \frac{1}{1-e^{-x}}$ to pick

$$m = \frac{8\ln\frac{1}{\delta} + 8\ln\frac{128/15}{(1-2p)^2}}{(1-2p)^2}.$$

We can now conclude the following theorem:

Theorem 28 *Let $\mathcal{A}_\ell$ be a linearly bounded error algorithm which requires $f(n,r)$ questions. Then $\mathcal{A}_\ell$ can be used to solve a probabilistic error problem specified by n, p, and δ in $f(cn, \frac{1+2p}{4})$ questions where $c = 2^{\frac{3-2p}{1+2p}m}$ and m is as given above.*

An $O(\log n)$ bound for the probabilistic error model with membership questions now easily follows from the results of the previous section.

Theorem 29 *The problem of searching for an unknown element $x \in \{1,\ldots,n\}$ with confidence probability δ in the presence of random errors (occurring randomly and independently with fixed probability $p < 1/2$) can be solved with $O(\log n)$ membership questions. The dependence of these bounds on p and δ is polynomial in $\frac{1}{1-2p}$ and logarithmic in $1/\delta$.*

9.2 The Unbounded Domain

We now consider the problem of searching for an unknown integer in the presence of random errors where no bound on the unknown number is known. Let this unknown integer be n. Our strategy proceeds in two stages. In the first stage, we obtain a bound for the integer n. In the second stage, we apply our techniques for searching in the bounded domain given above. To insure that our overall procedure fails with probability at most δ, we require that each of these two stages fails with probability at most $\delta' = \delta/2$.

9.2.1 Stage 1

By obtaining the correct answers to $\lceil \lg \lg n \rceil$ questions of the form "Is $x < 2^{2^i}$?" as in section 8.4.1, we can bound the unknown number n by at most n^2.

We might now imagine determining the correct answers to these $\lceil \lg \lg n \rceil$ questions by asking each one sufficiently often so that majority is incorrect with some sufficiently small probability. Unfortunately, to determine how much error is "sufficiently small" requires that we know the value of n. Since n is unknown here, we will require a more subtle querying algorithm.

To insure that our procedure fails with probability at most δ', we require that the correct answer to question i is obtained with error at most $\delta'/2^i$. Consider asking the i-th question $m(i)$ consecutive times and taking the majority vote of the responses to be the "correct" answer. The probability that our posited answer is *incorrect* can be calculated as follows:

$$\begin{aligned}
\Pr[\text{majority vote is wrong}] &= \Pr[\text{at least half errors}] \\
&= GE(p, m(i), m(i)/2) \\
&= GE(p, m(i), (p + [1/2 - p]) \cdot m(i)) \\
&\leq e^{-2(1/2-p)^2 m(i)} \\
&= e^{-\frac{(1-2p)^2 m(i)}{2}}
\end{aligned}$$

Since we require this probability to be at most $\delta'/2^i$, we obtain the following:

$$e^{-\frac{(1-2p)^2 m(i)}{2}} \leq \delta'/2^i$$

$$m(i) \quad \geq \quad \frac{\ln 4}{(1-2p)^2}\left[\lg \frac{1}{\delta'} + i\right]$$

Now, since our procedure will terminate (with probability at least $1-\delta'$) after the correct answers to $\lceil \lg \lg n \rceil$ questions have been obtained, we arrive at an overall question bound of

$$\begin{aligned}
\sum_{i=1}^{\lceil \lg \lg n \rceil} m(i) &= \sum_{i=1}^{\lceil \lg \lg n \rceil} \frac{\ln 4}{(1-2p)^2}\left[\lg \frac{1}{\delta'} + i\right] \\
&= \frac{\ln 4}{(1-2p)^2}\left[\lceil \lg \lg n \rceil \lg \frac{1}{\delta'} + \frac{\lceil \lg \lg n \rceil(\lceil \lg \lg n \rceil + 1)}{2}\right] \\
&= O([\lg \lg n]^2)
\end{aligned}$$

We thus bound the unknown number n by at most n^2 using $O([\lg \lg n]^2)$ (comparison) questions.

9.2.2 Stage 2

We can now simply apply the bounded searching techniques for membership questions described in previous section or the bounds of Feige *et al.* [12] for comparison questions. We can thus obtain the correct answer (with high probability) in an additional $O(\lg n^2) = O(\lg n)$ comparison or membership questions. Thus, we can conclude the following theorem:

Theorem 30 *The problem of searching for an unknown element n in the unbounded domain of all positive integers with confidence probability δ in the presence of random errors (occurring independently with fixed probability $p < 1/2$) can be solved with $O(\lg n)$ comparison or membership questions. The dependence of these bounds on p and δ is polynomial in $\frac{1}{1-2p}$ and logarithmic in $1/\delta$.*

Chapter 10

Conclusions and Open Questions

We have examined the problem of searching in a discrete domain under two different error models: the linearly bounded error model and the probabilistic error model.

In the linearly bounded error model, we have shown that $O(\lg n)$ membership questions are sufficient to search in both the bounded and unbounded domains. With comparison questions, we show bounds of $O(n^{\lg \frac{1}{1-r}})$ and $O([n \lg^2 n]^{\lg \frac{1}{1-r}})$ in the bounded and unbounded domains, respectively.

Our reduction from the probabilistic to the linearly bounded error model shows that the searching problem is at least as difficult to solve in the linearly bounded error model as in the probabilistic error model. This gives evidence that the linearly bounded error model deserves further investigation. A corollary of this reduction gives another proof of the $O(\lg n)$ bound on membership questions required to search with probabilistic errors. Previously known bounds are also extended to the unbounded domain.

Two questions arise directly from this work:

1. In the linearly bounded error model, can we show a logarithmic upper bound on the number of comparison questions required when the error rate is between 1/3 and 1/2? Using techniques similar to ours, Borgstrom and Kosaraju [8] have recently shown that this is the case.

2. Can a strict inequality be shown between the probabilistic and linearly bounded models

with respect to the problem of searching? That is, can it be shown that searching in the presence of linearly bounded errors with some question class requires an asymptotically greater number of questions than searching in the presence of random errors with the same question class? This problem remains open.

Bibliography

[1] Dana Angluin. Computational learning theory: Survey and selected bibliography. In *Proceedings of the* 24^{th} *Annual ACM Symposium on the Theory of Computing*, 1992.

[2] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[3] Dana Angluin and Leslie G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18(2):155–193, April 1979.

[4] Martin Anthony and Norman Biggs. *Computational Learning Theory.* Cambridge Tracts in Theoretical Computer Science (30). Cambridge University Press, 1992.

[5] Javed Aslam and Aditi Dhagat. On-line algorithms for 2-coloring hypergraphs via chip games. *Theoretical Computer Science*, 1993.

[6] Avrim Blum, Merrick Furst, Jeffery Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the* 26^{th} *Annual ACM Symposium on the Theory of Computing*, May 1994.

[7] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–865, 1989.

[8] Ryan S. Borgstrom and S. Rao Kosaraju. Comparison-based search in the presence of errors. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 130–136, 1993.

[9] Scott Decatur. Statistical queries and faulty PAC oracles. In *Proceedings of the Sixth Annual ACM Workshop on Computational Learning Theory*, pages 262–268. ACM Press, July 1993.

[10] Aditi Dhagat, Peter Gács, and Peter Winkler. On playing twenty questions with a liar. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, 1992.

[11] 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.

[12] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Computing with unreliable information. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, pages 128–137, 1990.

[13] Michael Frazier. Searching with a non-constant number of lies. Unpublished manuscript, 1990.

[14] Yoav Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 202–216. Morgan Kaufmann, 1990.

[15] Yoav Freund. An improved boosting algorithm and its implications on learning complexity. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 391–398. ACM Press, 1992.

[16] Sally A. Goldman, Michael J. Kearns, and Robert E. Schapire. On the sample complexity of weak learning. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 217–231. Morgan Kaufmann, 1990.

[17] David Helmbold, Robert Sloan, and Manfred K. Warmuth. Learning integer lattices. *SIAM Journal on Computing*, 21(2):240–266, 1992.

[18] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[19] Michael Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the* 25^{th} *Annual ACM Symposium on the Theory of Computing*, pages 392–401, San Diego, 1993.

[20] Michael Kearns and Ming Li. Learning in the presence of malicious errors. In *Proceedings of the* 20^{th} *Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, May 1988.

[21] Philip D. Laird. *Learning from Good and Bad Data.* Kluwer international series in engineering and computer science. Kluwer Academic Publishers, Boston, 1988.

[22] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*, volume 1, page 310. North Holland Publishing Company, 1977.

[23] Colin McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*, pages 149–188. Cambridge University Press, Cambridge, 1989. London Mathematical Society LNS 141.

[24] Andrzej Pelc. Searching wih known error probability. *Theoretical Computer Science*, 63:185–202, 1989.

[25] R. L. Rivest, A. R. Meyer, D. J. Kleitman, K. Winklmann, and J. Spencer. Coping with errors in binary search procedures. *Journal of Computer and System Sciences*, 20:396–404, 1980.

[26] Yasubumi Sakakibara. *Algorithmic Learning of Formal Languages and Decision Trees.* PhD thesis, Tokyo Institute of Technology, October 1991. (International Institute for Advanced Study of Social Information Science, Fujitsu Laboratories Ltd, Research Report IIAS-RR-91-22E).

[27] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory Series A*, 13:145–147, 1972.

[28] Robert Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–226, 1990.

[29] Robert E. Schapire. *The Design and Analysis of Efficient Learning Algorithms.* MIT Press, Cambridge, MA, 1992.

[30] Hans Ulrich Simon. General bounds on the number of examples needed for learning probabilistic concepts. In *Proceedings of the Sixth Annual ACM Workshop on Computational Learning Theory*, pages 402–411. ACM Press, 1993.

[31] Joel Spencer. *Ten Lectures on the Probabilistic Method*, chapter 4, pages 32–35. SIAM, 1987.

[32] Joel Spencer and Peter Winkler. Three thresholds for a liar. *Combinatorics, Probability and Computing*, 1:81–93, 1992.

[33] S. M. Ulam. *Adventures of a Mathematician.* Charles Scribner's Sons, 1 edition, 1976.

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

[35] Leslie Valiant. Learning disjunctions of conjunctions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985.

[36] V.N. Vapnik and A.Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theor. Probability Appl.*, 16(2):264–280, 1971.