General Bounds on Statistical Query Learning and PAC Learning with Noise via Hypothesis Boosting

Javed A. Aslam*
Department of Computer Science
Dartmouth College
Hanover, NH 03755

Scott E. Decatur†
DIMACS Center
Rutgers University
Piscataway, NJ 08855

*This work was performed while the author was at Harvard University supported by Air Force Contract F49620-92-J-0466 and at M.I.T. supported by DARPA Contract N00014-87-K-825 and by NSF Grant CCR-89-14428. Email: `jaa@cs.dartmouth.edu`

†This work was performed while the author was at Harvard University supported by an NDSEG Fellowship and by NSF Grant CCR-92-00884. Email: `sed@dimacs.rutgers.edu`

Running Title: General Bounds on SQ Learning

Contact address:

Scott E. Decatur
DIMACS Center
Busch Campus
Rutgers University
Piscataway, NJ 08855

Abstract

We derive general bounds on the complexity of learning in the Statistical Query (SQ) model and in the PAC model with classification noise. We do so by considering the problem of boosting the accuracy of weak learning algorithms which fall within the SQ model. This new model was introduced by Kearns to provide a general framework for efficient PAC learning in the presence of classification noise.

We first show a general scheme for boosting the accuracy of weak SQ learning algorithms, proving that weak SQ learning is equivalent to strong SQ learning. The boosting is efficient and is used to show our main result of the first general upper bounds on the complexity of strong SQ learning. Since all SQ algorithms can be simulated in the PAC model with classification noise, we also obtain general upper bounds on learning in the presence of classification noise for classes which can be learned in the SQ model.

List of Symbols

Symbol	Identity
abcdefghijklm nopqrstuvwxyz	Lowercase letters
ABCDEFGHIJKLM NOPQRSTUVWXYZ	Uppercase letters
abcdefghijklm *nopqrstuvwxyz*	Math/Italic lowercase letters
ABCDEFGHIJKLM *NOPQRSTUVWXYZ*	Math/Italic uppercase letters
HEAD	Small capital letters
ℓ	Lowercase cursive letter "ell"
0123456789	Numerals
$\alpha\beta\gamma\delta\varepsilon\eta\lambda\tau\chi$	Greek lowercase letters "alpha, beta, gamma, delta, epsilon, eta, lambda, tau, chi"
$\Gamma\Pi\Omega$	Greek uppercase letters "gamma, pi, omega"
$\mathcal{ABFGHQS}$	Script uppercase letters "ABFGHQS"
$\mathbb{N}$ $\mathbb{R}$	Denotes the set of natural numbers Denotes the set of real numbers
$\tilde{O}, \tilde{\Omega}, \tilde{\Theta}$ $\hat{h}, \hat{\mathcal{H}}, \hat{P}, \hat{t}$	Various symbols with "tilde" above Various symbols with "hat" above
∞	Infinity
@	"At" symbol
$\square$	Open box (end of proof symbol)
$=, \neq$	Equality and inequality
$<, \leq, >, \geq$	Less than, less than or equal, greater than, greater than or equal
$\approx$	Approximately equal
$\sum$	Summation
$\neg, \wedge, \equiv$	Boolean negation, AND and equivalence operator
$\times$	cross-multiplication
$*$	Asterisk
$\rightarrow$	Right arrow (maps to)
$\Leftarrow$	Double left arrow (reverse implication)
$\Leftrightarrow$	Double left-right arrow (double implication)
$\in$	Set membership
$\subseteq, \subset$	Subset and proper subset
$\triangle$	Symmetric difference
$\cap, \cup$	Set intersection and set union
$\vert$ and $\vert$	Set cardinality or absolute value
(and)	Left and right parentheses
[and]	Left and right square braces
{ and }	Left and right curly braces
$\langle$ and $\rangle$	Left and right angle braces
$\lceil$ and $\rceil$	Left and right ceiling braces
$\lfloor$ and $\rfloor$	Left and right floor braces
((and))	Left and right double parentheses

1 Introduction

Since Valiant's introduction of the Probably Approximately Correct model of learning (Valiant, 1984), 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 important complexity measures studied in the PAC model are *sample complexity* and *time complexity*. Angluin (1992) gives a survey of research in this model.

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. A 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 (1990; 1992) and later improved upon by Freund (1990; 1992) states that any algorithm which weakly learns can be transformed into an algorithm which 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." A popular noise model for both theoretical and experimental research is the *classification noise* model introduced by Angluin and Laird (1988). In the classification noise model, each example received by the learner is mislabelled randomly and independently with some fixed probability. While a limited number of efficient PAC algorithms had been developed which tolerate classification noise (Angluin and Laird, 1988; Goldman, Kearns, and Schapire, 1990; Sakakibara, 1991), no general framework for efficient learning[1] in the presence of classification noise was known until Kearns introduced the Statistical Query (SQ) model (Kearns, 1993).

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 additive error specified by the algorithm. 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 may view this new oracle as an intermediary which effectively limits the way in which a learning algorithm may 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, one would like to characterize these measures as well as possible.

Kearns (1993) demonstrates two interesting properties of the SQ model. First, he shows 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 shows 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 properties is that nearly every PAC learning algorithm can be transformed into one which tolerates classification noise.

While greatly expanding the function classes known to be learnable in the presence of noise, Kearns' technique does not make use of 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 (Helmbold, Sloan, and Warmuth, 1992), Kearns shows that this class is provably not learnable in the SQ model. Kearns' method for converting PAC algorithms to SQ algorithms consists of a few general strategies, 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 *ad hoc* conversion of a PAC algorithm to an SQ algorithm. A consequence of this fact is that general upper bounds on the complexity of PAC learning in the presence of noise are not directly obtainable either.

We instead 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" (Freund, 1990; Freund, 1992) which is nearly optimal in terms of its dependence on the accuracy parameter ε.

Schapire (1990) used his boosting result in the PAC model to upper bound various complexity measures for PAC learning. Similarly, we use our SQ boosting result to derive general upper bounds on various complexity measures for SQ learning. Specifically, we derive simultaneous upper bounds with respect to ε on the number of queries, $O(\log^2 \frac{1}{\varepsilon})$, the Vapnik-Chervonenkis dimension of the query space, $O(\log \frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon})$, and the inverse of the minimum tolerance, $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$. 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 $\Omega(\frac{d}{\log d} \log \frac{1}{\varepsilon})$ and the inverse of the minimum tolerance by $\Omega(\frac{1}{\varepsilon})$. Here d is the Vapnik-Chervonenkis dimension of the function class to be learned.

We further apply our boosting result and bounds in the SQ model to derive bounds in the PAC model with classification noise. Since nearly all PAC algorithms can be cast in the SQ model and since all SQ algorithms can be simulated in the PAC model with classification noise, we effectively demonstrate that it is possible to boost the accuracy of nearly all PAC algorithms even in the presence of noise. This provides a partial answer to an open problem of Schapire (1990) on whether boosting techniques can be used in the presence of noise. It also provides the first theoretical evidence for an empirical result obtained by Drucker, Schapire and Simard (1992) on improving the performance of a neural network in the presence of noise.

By creating efficient SQ algorithms and simulating them in the PAC model with classification noise, we effectively show that nearly every PAC algorithm can be converted into a highly efficient PAC algorithm which tolerates classification noise. We show an upper bound (with respect to ε) of $O(\frac{1}{\varepsilon^2} \log^c \frac{1}{\varepsilon})$ on the sample complexity of PAC learning in the presence of noise. Bounds for other complexity measures may also be derived by combining our bounds on SQ learning with an analysis of the complexity of the various PAC simulations of SQ algorithms in the presence of classification noise.

The remainder of the paper is organized as follows. In Section 2, we give formal definitions of the relevant learning models. Section 3 presents an overview of the PAC model boosting result of Freund on which our boosting technique is modelled. In Section 4, we describe our boosting results in the SQ model. Section 5 describes both upper bounds for SQ learning based on our boosting results and a lower bound for SQ learning. Section 6 describes the consequences of the SQ boosting results for PAC learning in the presence of classification noise. We conclude the paper in Section 7 and derive a number of technical results in the appendices that follow.

2 Learning Models

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

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 $\mathbb{R}^n$. We use the parameter n to denote the common length of each instance $x \in X$, and we denote the i^{th} component of an instance x by x_i.

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. Using the notation $\Pr_D[P(x)]$ to denote the probability that predicate P is satisfied by an instance drawn randomly according to D, we 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 $1/\varepsilon$, $1/\delta$, n and $size(f)$. We formally define PAC learning as follows (adapted from Kearns (1993)):

Definition 1 *(Strong PAC Learning)*
Let $\mathcal{F}$ and $\mathcal{H}$ be $\{0,1\}$-valued function classes defined over X. The class $\mathcal{F}$ is said to be learnable by $\mathcal{H}$ if there exists a learning algorithm $\mathcal{A}$ and a polynomial $p_1(\cdot,\cdot,\cdot,\cdot)$ such that for any $f \in \mathcal{F}$, for any distribution D on X, for any accuracy parameter ε, $0 < \varepsilon \leq 1$, 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 with probability at least $1-\delta$ $\mathcal{A}$ halts in time bounded by $p_1(1/\varepsilon, 1/\delta, n, size(f))$ and outputs an hypothesis $h \in \mathcal{H}$ that satisfies $error(h) \leq \varepsilon$.

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.* $error(h) \leq \frac{1}{2} - \gamma = \frac{1}{2} - \frac{1}{p(n, size(f))}$ 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 The Classification Noise Model

In the *classification noise model*, the example oracle $EX(f, D)$ is replaced by a noisy example oracle $EX^{\eta}(f, D)$. 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 $error(h) = \Pr_D[h(x) \neq f(x)]$ 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 $\frac{1}{1-2\eta_b}$.

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 $x_4 = 0$ 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.* $\chi : X \times \{0,1\} \rightarrow \{0,1\}$) 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 $P_\chi \stackrel{\triangle}{=} \Pr_D[\chi(x, f(x))]$ which satisfies $|\hat{P}_\chi - P_\chi| \leq \tau$.

A call to $STAT(f, D)$ could 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 $1/\varepsilon$, n and $size(f)$. We formally define polynomially efficient learning in the statistical query model as follows (adapted from Kearns (1993)):

Definition 2 *(Strong SQ Learning)*
Let $\mathcal{F}$ and $\mathcal{H}$ be $\{0,1\}$-valued function classes defined over X. The class $\mathcal{F}$ is said to be 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, $0 < \varepsilon \leq 1$, 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 $p_1(1/\varepsilon, n, size(f))$ and $1/\tau$ is bounded by $p_2(1/\varepsilon, n, size(f))$, and (2) $\mathcal{A}$ halts in time bounded by $p_3(1/\varepsilon, n, size(f))$ and outputs an hypothesis $h \in \mathcal{H}$ that satisfies $error(h) \leq \varepsilon$.

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.* $error(h) \leq \frac{1}{2} - \gamma = \frac{1}{2} - \frac{1}{p(n, size(f))}$ for some polynomial p.

We define the *query complexity* of SQ algorithm $\mathcal{A}$ to be the maximum number of queries that $\mathcal{A}$ makes in any run, and let $N_* = N_*(\varepsilon, n, size(f))$ be an upper bound on the query complexity of $\mathcal{A}$. We define the *tolerance* of SQ algorithm $\mathcal{A}$ to be the smallest additive error of the queries made by $\mathcal{A}$ in any run, and let $\tau_* = \tau_*(\varepsilon, n, size(f))$ be a lower bound on the tolerance of $\mathcal{A}$. Note that $N_* \leq p_3(1/\varepsilon, n, size(f))$ and $\tau_* \geq 1/p_2(1/\varepsilon, n, size(f))$.

Although a reasonably efficient simulation of an SQ algorithm can be obtained by drawing a separate sample for each call to the statistics oracle, a better sample complexity can be 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 a single sample sufficient is independent of the query complexity of $\mathcal{A}$ but depends on either the size of $\mathcal{Q}$, if $\mathcal{Q}$ is finite, or the Vapnik-Chervonenkis dimension[2] of $\mathcal{Q}$ (Kearns, 1993). $\mathcal{Q}$ is referred to as the *query space* of the SQ algorithm $\mathcal{A}$.

Recall that in addition to a noise-free simulation, an SQ algorithm can also be simulated, with high probability, by a procedure which draws a sample from a *classification noise* oracle (Kearns, 1993). We give the results of a more efficient simulation due to Aslam and Decatur (1994) below:

Theorem 1 *Let $\mathcal{F}$ and $\mathcal{H}$ be $\{0,1\}$-valued function classes defined over X. Suppose that $\mathcal{F}$ is polynomially learnable via statistical queries by $\mathcal{H}$ using learning algorithm $\mathcal{A}$. Then $\mathcal{F}$ is polynomially learnable by $\mathcal{H}$ in the classification noise model. If $\mathcal{A}$ uses query space $\mathcal{Q}$ and τ_* is a lower bound on the tolerance of $\mathcal{A}$, then the number of calls to $EX^{\eta}(f,D)$ required to simulate $\mathcal{A}$ in the classification noise model is*

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

if $\mathcal{Q}$ is finite, or

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

if $\mathcal{Q}$ has a finite VC-dimension.[3]

Given that nearly every PAC learning algorithm can be converted to an SQ algorithm, an immediate consequence of this result is that nearly every PAC algorithm can be transformed into one which tolerates noise. The complexities of these noise-tolerant PAC algorithms 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. By showing that weak SQ learning algorithms can be "boosted" to strong SQ learning algorithms, we derive general upper bounds on the inverse of 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 of noise-tolerant PAC learning via the statistical query model.

3 Boosting in the PAC Model

In this section, we describe Freund's PAC model boosting techniques on which our SQ model boosting results are based.

Schapire (1990; 1992) and Freund (1990; 1992) use similar strategies for converting weak PAC learning algorithms into strong PAC learning algorithms. Both methods create a strong hypothesis by combining hypotheses obtained from multiple runs of a weak learning algorithm. The boosting schemes derive their power by forcing successive runs of the weak learning algorithm to approximate the target function f with respect to *new distributions* each of which adjusts the weights of examples based on their classification with respect to previously generated weak hypotheses. 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 developed two similar methods (which we call Scheme 1 and Scheme 2) for converting weak learning algorithms into strong learning algorithms. One is more efficient with respect to ε while the other is more efficient with respect to γ. Freund also developed 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 methods separately and then show how they are combined.

3.1 Boosting via Scheme 1 in the PAC Model

Scheme 1 uses a weak learning algorithm to create a set of $k_1 \stackrel{\triangle}{=} \frac{1}{2\gamma^2} \ln \frac{1}{\varepsilon}$ 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 the i weak 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 a total of k weak hypotheses are to be generated, the set of probabilities $\{\alpha_r^i\}$ are fixed according to the following binomial distribution:

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

Freund shows that 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 $t_i < c\varepsilon^2$ for any constant $c \le 1/13$, then the majority vote of $h_1, \ldots, h_i$ is already a strong hypothesis. The boosting algorithm estimates t_i, and if t_i is below the cutoff, the algorithm halts and outputs 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 $\tilde{O}(1/\varepsilon^2)$.[4]

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 $k_2 \stackrel{\triangle}{=} 2k_1 = \frac{1}{\gamma^2}\ln\frac{1}{\varepsilon}$ weak hypotheses, while the filtered example oracle remains the same. The main difference is the observation that if $t_i < \frac{\varepsilon(1-\varepsilon)\gamma}{\ln(1/\varepsilon)}$, then we may simply use a "fair coin" in place of h_{i+1} and still be guaranteed that the final majority of k_2 hypotheses has error rate no more than ε.[5] 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 $\tilde{\Theta}(1/\varepsilon)$.

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 $1/\varepsilon$. One therefore uses the first method to boost from $\frac{1}{2}-\gamma$ to a constant and uses 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 $\mathcal{A}_\varepsilon$ 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 method's time and sample complexity dependence on γ is $\tilde{\Theta}(1/\gamma^2)$, while its dependence on ε is $\tilde{\Theta}(1/\varepsilon)$.

4 Boosting in the Statistical Query Model

Hypothesis boosting is accomplished essentially 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 section are based on the ability to construct weak hypotheses with respect to a sequence of distributions $\{D_j\}$. In the PAC model, a learner interacts with a 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 show how to boost a weak SQ algorithm using Scheme 1, Scheme 2 and the hybrid method. Although it is possible to boost in the SQ model using Schapire's method, we do not describe these results here since they are somewhat weaker and more complex to present than the results we achieve using Freund's methods.

4.1 Boosting via Scheme 1 in the Statistical Query Model

We use Scheme 1 to boost weak SQ learning algorithms by answering statistical queries made with respect to modified distributions. Therefore, we must 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 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 probabilities 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 Eq. (2) from Eq. (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 $\Pr_{D_{i+1}}[\chi(x, f(x))]$ as follows:

$$\begin{aligned}
\Pr_{D_{i+1}}[\chi(x, f(x))] &= \sum_{r=0}^{i} \Pr_{D_{i+1}}[\chi(x, f(x)) | (x \in X_r^i)] \cdot \Pr_{D_{i+1}}[x \in X_r^i] && (1)\\
&= \sum_{r=0}^{i} \Pr_{D}[\chi(x, f(x)) | (x \in X_r^i)] \cdot \Pr_{D_{i+1}}[x \in X_r^i] && (2)\\
&= \sum_{r=0}^{i} \frac{\Pr_D[\chi(x, f(x)) \wedge (x \in X_r^i)]}{\Pr_D[x \in X_r^i]} \cdot \frac{\alpha_r^i \cdot \Pr_D[x \in X_r^i]}{\sum_{j=0}^{i} \alpha_j^i \cdot \Pr_D[x \in X_j^i]} \\
&= \frac{\sum_{r=0}^{i} \alpha_r^i \cdot \Pr_D[\chi(x, f(x)) \wedge (x \in X_r^i)]}{\sum_{j=0}^{i} \alpha_j^i \cdot \Pr_D[x \in X_j^i]}. && (3)
\end{aligned}$$

Note that the denominator of Eq. (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 Eq. (3) can be stated as queries to $STAT(f, D)$ as follows

$$STAT(f, D_{i+1})[\chi] = \frac{\sum_{j=0}^{i} \alpha_j^i \cdot STAT(f, D)[\chi \wedge \chi_j^i]}{\sum_{j=0}^{i} \alpha_j^i \cdot STAT(f, D)[\chi_j^i]} \tag{4}$$

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 one must ask these queries so that the final result is within the desired additive error τ. We make use of the following two simple lemmas:

Lemma 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 $(b - c\tau/3)/(c + c\tau/3) \geq a - \tau$. These two inequalities are shown as follows:

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

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

□

Lemma 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 exactly.

Proof: The lemma 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 Lemmas 1 and 2 to Eq. (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. However, there are two complications with this strategy. 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 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 $t_i < c\varepsilon^2$, then the majority vote of the hypotheses generated thus far is a strong hypothesis. We make each $STAT$ call on the right-hand side of Eq. (4) with additive error $\frac{c\varepsilon^2\tau}{3+2\tau}$. Let $\hat{t}_i$ be the estimate for t_i obtained, and note that by Lemma 2, $\hat{t}_i$ is within additive error $\frac{c\varepsilon^2\tau}{3+2\tau}$ of t_i. If $\hat{t}_i < c\varepsilon^2(1 - \frac{\tau}{3+2\tau})$, then $t_i < c\varepsilon^2$, and in this case, we halt and output the majority vote of the hypotheses created thus far. If $\hat{t}_i \geq c\varepsilon^2(1 - \frac{\tau}{3+2\tau})$, then $t_i \geq c\varepsilon^2(1 - \frac{2\tau}{3+2\tau}) = c\varepsilon^2(\frac{3}{3+2\tau})$. In this case, our estimate $\hat{t}_i$ is sufficiently accurate since the additive error required by Lemma 1 is $t_i \cdot \tau/3$, and $t_i \cdot \tau/3 \geq c\varepsilon^2(\frac{3}{3+2\tau}) \cdot \tau/3 = \frac{c\varepsilon^2\tau}{3+2\tau}$ 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 Lemma 1.

Thus, we bound the tolerance of strong SQ learning algorithms obtained by Scheme 1 boosting as follows. If $\tau_0 = \tau_0(n, size(f))$ is a lower bound on the tolerance of a weak SQ learning algorithm, then $\Omega(\tau_0\varepsilon^2)$ 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 SQ learning algorithm. In Eq. (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 $k_1 = \frac{1}{2\gamma^2} \ln \frac{1}{\varepsilon}$ is an upper bound on the number of weak learners run in the boosting scheme, $O(N_0 {k_1}^2) = O(N_0 \frac{1}{\gamma^4} \log^2 \frac{1}{\varepsilon})$ is an upper bound on the query complexity of the strong SQ learning algorithm obtained by Scheme 1 boosting.

We finally examine the complexity of $\mathcal{Q}_B$, the query space 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 bound the *size* of the query space used by the strong SQ learning algorithm obtained by boosting. If the instance space is *continuous*, *e.g.* n-dimensional Euclidean space $\mathbb{R}^n$, then the query space and hypothesis class used by an SQ algorithm are generally infinite. In this case, we bound the *VC-dimension* of the query space used by the strong SQ learning algorithm obtained by boosting.

In Section A of the appendix, we show that the size of the query space is bounded as follows, when $\mathcal{Q}_0$ and $\mathcal{H}_0$, the query space and hypothesis space of the weak learning algorithm, are finite:

$$\log |\mathcal{Q}_B| = O(\log |\mathcal{Q}_0| + k_1 \log |\mathcal{H}_0|).$$

Furthermore, we show 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 2 *Given a weak SQ learning algorithm for $\mathcal{F}$ 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 for $\mathcal{F}$ can be constructed whose query complexity is $O(N_0 \frac{1}{\gamma^4} \log^2 \frac{1}{\varepsilon})$ and whose tolerance is $\Omega(\tau_0 \varepsilon^2)$. The query space complexity is given by*

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

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

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

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

4.2 Boosting via Scheme 2 in the Statistical Query Model

We use Scheme 2 to boost weak SQ learning algorithms in a manner 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 $\lambda_r^{i,w}$ 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, this probability is α_r^i. If j of the "fair coin" hypotheses agree with f, this probability is α_{r+j}^i. Thus, $\lambda_r^{i,w} = \sum_{j=0}^{i-w} \alpha_{r+j}^i \beta_j^{i-w}$ 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 the w weak hypotheses which agree with f on x.
4. Flip a biased coin with $\Pr[\textsc{head}] = \lambda_r^{i,w}$.
5. If HEAD, then output example $\langle x, f(x) \rangle$, otherwise go to Step 1.

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

$$\begin{aligned}
\Pr_{D_{i+1}}[\chi(x, f(x))] &= \sum_{r=0}^{w} \Pr_{D_{i+1}}[\chi(x, f(x)) | (x \in X_r^w)] \cdot \Pr_{D_{i+1}}[x \in X_r^w] \\
&= \sum_{r=0}^{w} \Pr_D[\chi(x, f(x)) | (x \in X_r^w)] \cdot \Pr_{D_{i+1}}[x \in X_r^w] \\
&= \sum_{r=0}^{w} \frac{\Pr_D[\chi(x, f(x)) \wedge (x \in X_r^w)]}{\Pr_D[x \in X_r^w]} \cdot \frac{\lambda_r^{i,w} \cdot \Pr_D[x \in X_r^w]}{\sum_{j=0}^{w} \lambda_j^{i,w} \cdot \Pr_D[x \in X_j^w]} \\
&= \frac{\sum_{r=0}^{w} \lambda_r^{i,w} \cdot \Pr_D[\chi(x, f(x)) \wedge (x \in X_r^w)]}{\sum_{j=0}^{w} \lambda_j^{i,w} \cdot \Pr_D[x \in X_j^w]}. \qquad (5)
\end{aligned}$$

Note that the denominator of Eq. (5) again corresponds to t_i, the probability that a labelled example is accepted by the probabilistic filter. Also note that $\sum_{r=0}^{w} \lambda_r^{i,w} = \sum_{r=0}^{w} \sum_{j=0}^{i-w} \alpha_{r+j}^i \beta_j^{i-w} \leq 1$ since the terms in the double summation are distinct and contained in the product $(\sum_{r=0}^{i} \alpha_r^i)(\sum_{j=0}^{i-w} \beta_j^{i-w}) = 1$.

Ignoring the additive error parameter for the moment, the probabilities in Eq. (5) can be stated as queries to $STAT(f, D)$ as follows:

$$STAT(f, D_{i+1})[\chi] = \frac{\sum_{j=0}^{w} \lambda_j^{i,w} \cdot STAT(f,D)[\chi \wedge \chi_j^w]}{\sum_{j=0}^{w} \lambda_j^{i,w} \cdot STAT(f,D)[\chi_j^w]}. \tag{6}$$

Applying Lemmas 1 and 2 to Eq. (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, two complications arise with this strategy. 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 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 $t_i < \varepsilon(1-\varepsilon)\gamma/\ln(1/\varepsilon)$, then a "fair coin" can be used in place of h_{i+1}. We make each $STAT$ call on the right-hand side of Eq. (6) with additive error $\frac{\tau\varepsilon(1-\varepsilon)\gamma/\ln(1/\varepsilon)}{3+2\tau}$. Let $\hat{t}_i$ be the estimate obtained for t_i, and note that by Lemma 2, $\hat{t}_i$ is within additive error $\frac{\tau\varepsilon(1-\varepsilon)\gamma/\ln(1/\varepsilon)}{3+2\tau}$ of t_i. If $\hat{t}_i < \frac{\varepsilon(1-\varepsilon)\gamma}{\ln(1/\varepsilon)}(1 - \frac{\tau}{3+2\tau})$, then $t_i < \varepsilon(1-\varepsilon)\gamma/\ln(1/\varepsilon)$. In this case, we use a "fair coin" in place h_{i+1} and proceed to the next distribution. If $\hat{t}_i \geq \frac{\varepsilon(1-\varepsilon)\gamma}{\ln(1/\varepsilon)}(1 - \frac{\tau}{3+2\tau})$, then $t_i \geq \frac{\varepsilon(1-\varepsilon)\gamma}{\ln(1/\varepsilon)}(1 - \frac{2\tau}{3+2\tau}) = \frac{\varepsilon(1-\varepsilon)\gamma}{\ln(1/\varepsilon)}(\frac{3}{3+2\tau})$. In this case, the estimate $\hat{t}_i$ is sufficiently accurate since the additive error required by Lemma 1 is $t_i \cdot \tau/3$, and $t_i \cdot \tau/3 \geq \frac{\varepsilon(1-\varepsilon)\gamma}{\ln(1/\varepsilon)}(\frac{3}{3+2\tau}) \cdot \tau/3 = \frac{\tau\varepsilon(1-\varepsilon)\gamma/\ln(1/\varepsilon)}{3+2\tau}$ 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 Lemma 1.

Thus, we bound the tolerance of strong SQ learning algorithms obtained by Scheme 2 boosting as follows. If $\tau_0 = \tau_0(n, size(f))$ is a lower bound on the tolerance of a weak SQ learning algorithm, then $\Omega(\tau_0\varepsilon\gamma/\log(1/\varepsilon))$ 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 $N_0 = N_0(n, size(f))$ be an upper bound on the query complexity of a weak SQ learning algorithm. In Eq. (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 $k_2 = \frac{1}{\gamma^2}\ln\frac{1}{\varepsilon}$ is an upper bound on the number of weak learners run in the boosting scheme, $O(N_0 {k_2}^2) = O(N_0 \frac{1}{\gamma^4}\log^2\frac{1}{\varepsilon})$ is an upper bound on the query complexity of the strong SQ learning algorithm obtained by Scheme 2 boosting.

Finally, we note that the query space complexity results for Scheme 2 boosting are identical to those for Scheme 1 boosting where k_1 is replace by k_2.

Theorem 3 *Given a weak SQ learning algorithm for $\mathcal{F}$ 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 for $\mathcal{F}$ can be constructed whose query complexity is $O(N_0 \frac{1}{\gamma^4} \log^2 \frac{1}{\varepsilon})$ and whose tolerance is $\Omega(\tau_0 \varepsilon \gamma / \log(1/\varepsilon))$. The query space complexity is given by*

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

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

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

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

4.3 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 2 and Theorem 3, 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 proven in Sections A and B of the appendix. We thus obtain the following boosting result.

Theorem 4 *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 $O(N_0 \frac{1}{\gamma^4} \log^2 \frac{1}{\varepsilon})$ and whose tolerance is $\Omega(\tau_0 \varepsilon / \log(1/\varepsilon))$. The query space complexity is given by*

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

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

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

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.

5 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.

5.1 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 (Schapire, 1990), 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 may convert any strong SQ learning algorithm into a weak SQ learning algorithm by "hard-wiring" the accuracy parameter ε to a constant. We 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 5 *Suppose a class $\mathcal{F}$ is SQ learnable by algorithm $\mathcal{A}$ whose query complexity is upper bounded by $N_*(n, size(f), \varepsilon)$, whose tolerance is lower bounded by $\tau_*(n, size(f), \varepsilon)$, whose query space and hypothesis class are $\mathcal{Q}(n, size(f), \varepsilon)$ and $\mathcal{H}(n, size(f), \varepsilon)$, respectively, and let N_0, τ_0, $\mathcal{Q}_0$ and $\mathcal{H}_0$ be these complexity measures when $\mathcal{A}$ is run with a constant accuracy parameter. Then $\mathcal{F}$ is SQ learnable by an algorithm whose query complexity is $O(N_0 \log^2 \frac{1}{\varepsilon})$, whose tolerance is $\Omega(\tau_0 \varepsilon / \log(1/\varepsilon))$ and whose query space complexity is given by*

$$\log |\mathcal{Q}_{HB}| = O(\log |\mathcal{Q}_0| + \log \tfrac{1}{\varepsilon} \log |\mathcal{H}_0|)$$

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

$$VC(\mathcal{Q}_{HB}) = O(VC(\mathcal{Q}_0) + VC(\mathcal{H}_0) \cdot \log \tfrac{1}{\varepsilon} \log\log \tfrac{1}{\varepsilon})$$

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

Corollary 1 *If a class $\mathcal{F}$ is SQ learnable, then $\mathcal{F}$ is SQ learnable by an algorithm whose complexity depends on ε as follows: The query complexity is $O(\log^2 \frac{1}{\varepsilon})$, the tolerance is $\Omega(\varepsilon / \log(1/\varepsilon))$ and the query space is of size $O(\log \frac{1}{\varepsilon})$ when the query space is finite or has VC-dimension $O(\log \frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon})$ when the query space has finite VC-dimension.*

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 $O(\log^2 \frac{1}{\varepsilon})$, $O(\log \frac{1}{\varepsilon})$ and $O(\log \frac{1}{\varepsilon})$, respectively.[6]

Kearns (1993) shows that for any function class of VC-dimension d, learning in the SQ model requires $\Omega(d/\log d)$ queries each with additive error $O(\varepsilon)$. 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 Corollary 1 is optimal to within a logarithmic factor.

5.2 A Specific Lower Bound for Learning in the SQ Model

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. In this section we show that this is not the case by demonstrating a specific learning problem which requires $\Omega(\frac{d}{\log d} \log \frac{1}{\varepsilon})$ queries each with additive error $O(\varepsilon)$ in the SQ model. Thus, with respect to ε, our general upper bound on query complexity is within a $\log \frac{1}{\varepsilon}$ factor of the best possible general upper bound. 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 $S \subseteq [N]$ of size d, and the goal of the *player* is to output a set $T \subseteq [N]$ such that $|S \triangle T| \leq t$.[7] The player is allowed to ask *queries* of the form $Q \subseteq [N]$ to which the adversary returns $|Q \cap S|$.

Lemma 3 *For any $d \geq 4$, $t \leq d/4$ and $N = \Omega(d^{1+\alpha})$ for any constant $\alpha > 0$, the player requires $\Omega(\frac{d}{\log d} \log N)$ queries, 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 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 $S \in \mathcal{S}_0^i$ satisfies $|S \cap Q_1| = i$. 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 $\mathcal{S} \subseteq 2^{[N]}$, define $width(\mathcal{S}) = \max_{S_i, S_j \in \mathcal{S}} \{|S_i \triangle S_j|\}$. Note that if $width(\mathcal{S}_k) > 2t$, 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). If the player were to stop and output any set T at this point, then the malicious adversary could always force the player to lose. We now bound $width(\mathcal{S}_k)$ 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 (Graham, Knuth, and Patashnik, 1994):

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

where we use the standard combinatorial notation $\left(\!\binom{n}{r}\!\right) = \sum_{i=0}^{r} \binom{n}{i}$.

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 $width(\mathcal{S}_k) > 2t$. We now note that any k satisfying the following inequalities will guarantee that $width(\mathcal{S}_k) > 2t$:

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

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 $N = \Omega(d^{1+\alpha})$. □

Now consider a learning problem defined as follows. Our instance space X is the set of natural numbers $\mathbb{N}$, and our function class is the set of all indicator functions corresponding to subsets of $\mathbb{N}$ of size d. This function class is learnable in the SQ model.[8] In what follows, we show that any deterministic SQ algorithm for this class requires $\Omega(\frac{d}{\log d}\log\frac{1}{\varepsilon})$ queries, each with additive error $O(\varepsilon)$.

Theorem 6 *There exists a parameterized family of function classes which is learnable in the SQ model and requires* $\Omega(\frac{d}{\log d}\log\frac{1}{\varepsilon})$ *queries, each with additive error* $O(\varepsilon)$.

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 distribution D over $\mathbb{N}$ to have weight $4/Nd$ on each point in the set $\{1, \ldots, N\}$ and to have weight $1 - 4/d$ on the point $N + 1$. All other points have zero weight. We set $\varepsilon = 1/N$ and invoke the deterministic SQ learning algorithm. Note that if the SQ algorithm returns an hypothesis whose error rate is at most $\varepsilon = 1/N$, then this hypothesis corresponds to a set whose symmetric difference with the target set is at most $t = d/4$.

Since the target subset has weight $4/N$, if the SQ algorithm submits a query with additive error greater than or equal to $4\varepsilon = 4/N$, then we may answer the query without consulting the adversary by assuming that the target subset is "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 or equal to 4ε. Since $\Omega(\frac{d}{\log d}\log N)$ queries of the adversary are required, the SQ algorithm must ask $\Omega(\frac{d}{\log d}\log N) = \Omega(\frac{d}{\log d}\log\frac{1}{\varepsilon})$ queries, each with additive error $O(\varepsilon)$. $\Box$

Using techniques similar to those found in Kearns' lower bound proof (Kearns, 1993), the above proof can 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 unless it asks $\Omega(\frac{d\log(1/\varepsilon)}{\log(d\log(1/\varepsilon))})$ queries, each with additive error $O(\varepsilon)$. This result is given in Section C of the appendix.

6 General Bounds on PAC Learning with Classification Noise

Given the bounds of the previous section and Theorem 1, we obtain upper bounds on the sample complexity of PAC learning in the presence of classification noise for classes whose PAC algorithms can be stated in the SQ model.

Theorem 7 *Suppose a class $\mathcal{F}$ is SQ learnable by an algorithm $\mathcal{A}$ whose query complexity is upper bounded by $N_*(n, size(f), \varepsilon)$, whose tolerance is lower bounded by $\tau_*(n, size(f), \varepsilon)$, whose query space and hypothesis class are $\mathcal{Q}(n, size(f), \varepsilon)$ and $\mathcal{H}(n, size(f), \varepsilon)$, respectively, and let N_0, τ_0, $\mathcal{Q}_0$ and $\mathcal{H}_0$ be these complexity measures when $\mathcal{A}$ is run with a constant accuracy parameter. Then $\mathcal{F}$ is PAC learnable in the presence of classification noise by an algorithm whose sample complexity is*

$$O\left(\frac{1}{{\tau_0}^2\varepsilon^2(1-2\eta_b)^2}\log^2\frac{1}{\varepsilon}\cdot\left(\log|\mathcal{Q}_0| + \log\frac{1}{\varepsilon}\log|\mathcal{H}_0| + \log\frac{1}{\delta}\right) + \frac{1}{\varepsilon(1-2\eta_b)^2}\log\log\frac{1}{1-2\eta_b}\right)$$

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

$$O\left(\frac{1}{{\tau_0}^2\varepsilon^2(1-2\eta_b)^2}\log^2\frac{1}{\varepsilon}\cdot\left[\left(VC(\mathcal{Q}_0) + VC(\mathcal{H}_0)\log\frac{1}{\varepsilon}\log\log\frac{1}{\varepsilon}\right)\cdot\log\left(\frac{1}{\tau_0\varepsilon(1-2\eta_b)}\log\frac{1}{\varepsilon}\right) + \log\frac{1}{\delta}\right]\right)$$

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

Corollary 2 *If a class $\mathcal{F}$ is SQ learnable, then $\mathcal{F}$ is PAC learnable in the presence of classification noise by an algorithm whose sample complexity depends on ε as follows: $O(\frac{1}{\varepsilon^2}\log^3\frac{1}{\varepsilon})$ when the query space and hypothesis class are finite or $O(\frac{1}{\varepsilon^2}\log^4\frac{1}{\varepsilon}\log\log\frac{1}{\varepsilon})$ when the query space and hypothesis class have finite VC-dimension.*

Note that the hypothesis output by the simulation of an SQ algorithm in the presence of classification noise is simply an hypothesis generated by the SQ algorithm itself. Therefore, the hypothesis size required for PAC learning in the presence of classification noise can be bounded, with respect to ε, by $O(\log\frac{1}{\varepsilon})$.

Bounds on additional complexity measures, such as space and time, may also be obtained by combining our general upper bounds for SQ learning with an analysis of the complexity of a simulation of SQ algorithms in the PAC model with classification noise. As the various simulations exhibit tradeoffs among these complexity measures, we refer the reader to the details of these simulations given by Kearns (1993) and Aslam and Decatur (1994; 1995).

7 Discussion

Blumer *et al.* (1989) show that the sample complexity of PAC learning depends at least linearly on $1/\varepsilon$, and clearly this bound holds for learning in the presence of classification noise as well. Laird (1988) has developed a general technique for learning in the presence of classification noise whose sample complexity depends only linearly on $1/\varepsilon$; however, this technique does not yield computationally efficient algorithms. The upper bound given in Theorem 7 has a roughly quadratic dependence on $1/\varepsilon$. Since the tolerance of our boosting scheme has a roughly optimal dependence on ε, one cannot significantly improve the sample complexity bound given in Theorem 7 by improving the boosting scheme. An interesting open question is whether there exists a time-efficient, noise-tolerant PAC simulation of $STAT(f, D)$ whose sample complexity dependence on ε is $o(1/\varepsilon^2)$. Such a simulation would immediately yield improved sample complexity bounds for learning in the presence of classification noise. Conversely, if one could show that such a simulation does not exist, then our classification noise bounds are roughly the strongest obtainable through the use of the Statistical Query model.

Aslam and Decatur (1995) provide a partial answer to this question by introducing a new simulation of SQ algorithms and demonstrating a function class and corresponding SQ learning algorithm which takes advantage of this simulation in order to achieve a roughly linear sample complexity dependence on $1/\varepsilon$. Note that this linear dependence is only achieved for the simulation of some SQ algorithms and therefore does not provide a general technique for achieving a $o(1/\varepsilon^2)$ sample complexity.

Acknowledgements

We would like to thank Mike Kearns, Yoav Freund, Ron Rivest and Les Valiant for their questions, comments and suggestions.

Appendices

A The Finite Query Space Complexity of Boosting

In this section of the appendix, we calculate the size of the query space of boosting by Scheme 1, Scheme 2 and hybrid boosting. These results apply when the query space and hypothesis class of the weak SQ algorithm are finite.

A.1 The Size of the Query Space of Scheme 1 and Scheme 2 Boosting

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 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 $\mathcal{H}_0$. 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 equal 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. Thus, the number of unique χ_j^i queries is $\binom{|\mathcal{H}_0|+i-1}{i}$ (Feller, 1968). 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 $k = k_1$ or k_2, the total number of χ_j^i queries is given by $\sum_{i=1}^{k}(i+1) \cdot \binom{|\mathcal{H}_0|+i-1}{i}$. 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 or Scheme 2 boosting as follows

$$|\mathcal{Q}_B| = |\mathcal{Q}_0| + \sum_{i=1}^{k}(i+1) \cdot \binom{|\mathcal{H}_0|+i-1}{i} + |\mathcal{Q}_0| \sum_{i=1}^{k}(i+1) \cdot \binom{|\mathcal{H}_0|+i-1}{i} \tag{7}$$

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.

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

Using the fact that $\sum_{i=0}^{m} \binom{n+i}{i} = \binom{n+m+1}{m}$ (Graham, Knuth, and Patashnik, 1994), we now have the following:

$$\sum_{i=1}^{k}(i+1) \cdot \binom{N+i-1}{i} = N \sum_{i=1}^{k} \binom{N+i-1}{i-1} + \sum_{i=1}^{k} \binom{N-1+i}{i}$$

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

Applying this fact to Eq. (7) above, we obtain the following closed-form expression:

$$
|\mathcal{Q}_B| = (|\mathcal{Q}_0|+1)\binom{|\mathcal{H}_0|+k}{k}+|\mathcal{H}_0|(|\mathcal{Q}_0|+1)\binom{|\mathcal{H}_0|+k}{k-1}-1. \tag{8}
$$

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:

$$
\begin{aligned}
|\mathcal{Q}_B| &\leq (|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+1)^k+|\mathcal{H}_0|(|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^{k-1}-1 \\
&< (|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^k+(|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^k \\
&= 2(|\mathcal{Q}_0|+1)(|\mathcal{H}_0|+2)^k. \qquad (9)
\end{aligned}
$$

The PAC sample complexity of simulating an SQ algorithm depends on $\log|\mathcal{Q}_B|$, and we have shown that:

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

A.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 $\varepsilon = 1/4$, as its "weak learner," while the Scheme 1 booster uses the actual weak learner. Thus, $k_1 = k_1(\gamma, 1/4)$ and $k_2 = k_2(1/4, \varepsilon)$. 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

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

which yields

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

B The General Query Space Complexity of Boosting

In this section of the appendix, we prove bounds on the Vapnik-Chervonenkis dimension (Vapnik and Chervonenkis, 1971) of the query space of boosting. We begin by defining VC-dimension and introducing a number of preliminary results.

B.1 Preliminaries

Let $\mathcal{G}$ be a set of $\{0,1\}$-valued functions defined over a domain X. For any 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 $m \geq 1$, define $\Pi_{\mathcal{G}}(m) = \max\{|\Pi_{\mathcal{G}}(S)| : S \subseteq X, |S| = m\}$. 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 throughout the sections that follow.

Lemma 4 *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 function $\Pi_{\mathcal{G}}(m)$ 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 (1972) upper bounds the growth of $\Pi_{\mathcal{G}}(m)$ for all $m \geq VC(\mathcal{G})$.

Lemma 5 (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.* (1989) show 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 6 *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:

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

$$
\begin{aligned}
&= \sum_{i=0}^{d} \frac{d^i}{i!} \\
&< \sum_{i=0}^{\infty} \frac{d^i}{i!} \\
&= e^d.
\end{aligned}
$$

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

Thus, the growth of the function $\Pi_{\mathcal{G}}(m)$ can be characterized 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.

B.2 The VC-Dimension of the Query Space of Scheme 1 and Scheme 2 Boosting

We now prove the result used in Section 4 bounding the VC-dimension of the query space of Scheme 1 and Scheme 2 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 $k_1 = \frac{1}{2\gamma^2} \ln \frac{1}{\varepsilon}$ in Scheme 1 boosting, and i is bounded by $k_2 = \frac{1}{\gamma^2} \ln \frac{1}{\varepsilon}$ 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 bound the VC-dimension of $\mathcal{Q}_B$ in terms of the VC-dimensions of $\mathcal{Q}_0$ and $\mathcal{H}_0$ as follows. We first bound $\Pi_{\mathcal{Q}_0}(m)$, $\Pi_{\Gamma^{[k]}}(m)$ and $\Pi_{\mathcal{Q}_0 \wedge \Gamma^{[k]}}(m)$ and then apply Lemma 4 to obtain a bound on $\Pi_{\mathcal{Q}_B}(m)$. From this bound, we obtain a bound on the VC-dimension of $\mathcal{Q}_B$ by finding an m such that $\Pi_{\mathcal{Q}_B}(m) < 2^m$. We begin by examining Γ_j^i.

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

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

where $\equiv$ is the Boolean equivalence operator. Thus, $\hat{h}(x,l)$ is true if and only if the hypothesis h maps x to l. Let $\widehat{\mathcal{H}}_0 = \{\hat{h} : h \in \mathcal{H}_0\}$. A query χ_j^i based on hypotheses $h_1, \ldots, h_i$ can be specified as follows:

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

From a set-theoretic perspective, we may 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 1 $\Pi_{\Gamma_j^i}(m) \le \binom{\Pi_{\widehat{\mathcal{H}}_0}(m) + i - 1}{i}$

Proof: We will view each $\chi_j^i \in \Gamma_j^i$ and $\hat{h} \in \widehat{\mathcal{H}}_0$ as indicator functions for sets $Y_{\chi_j^i} \subseteq Y$ and $Y_{\hat{h}} \subseteq Y$, respectively. Let T be any subset of Y of size m. Recall that $\Pi_{\widehat{\mathcal{H}}_0}(T)$ 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 $\chi_j^i \in \Gamma_j^i$. By the definition of χ_j^i, note that each set in $\Pi_{\Gamma_j^i}(T)$ must correspond to some collection of i sets in $\Pi_{\widehat{\mathcal{H}}_0}(T)$. However, the i sets in any collection need not be distinct. Since 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 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 4 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 4 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 simple fact that $\Pi_{\mathcal{A} \wedge \mathcal{B}}(m) \leq \Pi_{\mathcal{A}}(m) \cdot \Pi_{\mathcal{B}}(m)$ (Anthony and Biggs, 1992, pg. 104), we 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 4 and the definition of $\mathcal{Q}_B$, we have

$$\Pi_{\mathcal{Q}_B}(m) \leq$$
$$\Pi_{\mathcal{Q}_0}(m) + \sum_{i=1}^{k} (i+1) \cdot \binom{\Pi_{\widehat{\mathcal{H}}_0}(m) + i - 1}{i} + \Pi_{\mathcal{Q}_0}(m) \sum_{i=1}^{k} (i+1) \cdot \binom{\Pi_{\widehat{\mathcal{H}}_0}(m) + i - 1}{i}. \quad (10)$$

Note that Eq. (10) is of the same form as Eq. (7), and we may therefore simplify Eq. (10) in a similar manner to obtain:

$$\Pi_{\mathcal{Q}_B}(m) < 2(\Pi_{\mathcal{Q}_0}(m) + 1)(\Pi_{\widehat{\mathcal{H}}_0}(m) + 2)^k. \quad (11)$$

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

Claim 2 $VC(\widehat{\mathcal{H}}_0) = VC(\mathcal{H}_0)$

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 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 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, $x_i \neq x_j$ if $i \neq j$, and therefore $S = \bigcup_{i=1}^{m}\{x_i\}$ 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 a hypothesis $h \in \mathcal{H}_0$ corresponding to a 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 now prove the main result of this section.

Lemma 7 $VC(\mathcal{Q}_B) = O(VC(\mathcal{Q}_0) + VC(\mathcal{H}_0) \cdot k \log k)$

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 bound for $\Pi_{\mathcal{Q}_B}(m)$ given in Eq. (11).

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

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

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. (We use lg to denote the logarithm base 2.)

$$\begin{aligned} 2^{k+2}(em/q_0)^{q_0}(em/d_0)^{d_0 k} &\leq 2^m \\ \Leftrightarrow \quad (k+2) + q_0 \lg(em/q_0) + d_0 k \lg(em/d_0) &\leq m \end{aligned} \tag{12}$$

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 to 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 $m \geq g_1(m) + g_2(m) + g_3(m)$ by the following argument. 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:

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

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:

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

This final inequality is true for any $k \geq 2$. Thus $m = 3\max\{k+2, 6q_0, 9d_0 k \lg k\} = O(q_0 + d_0 k \log k)$ is an upper bound on $VC(\mathcal{Q}_B)$. $\square$

B.3 The VC-Dimension of the Query Space of Hybrid Boosting

We now prove the result used in Section 4 bounding the VC-dimension of the query space of hybrid boosting. As in Section A.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 $k_2 = k_2(1/4, \varepsilon)$. We then have the following analogs of Eq. (11):

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

Recall that $\mathcal{H}_{1/4}$ 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, we make use of the following equivalent definition of the functions in $\widehat{\mathcal{H}}_{1/4}$. 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 $\{h_1, \ldots, h_j\}$ from $\mathcal{H}_0$ where $j \le k_1$. By definition, we have

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

However, it is also the case that

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

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. Recall that $\Pi_{\widehat{\mathcal{H}}_0}(T)$ 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 set in $\Pi_{\widehat{\mathcal{H}}^j_{1/4}}(T)$ must correspond to some collection of j sets in $\Pi_{\widehat{\mathcal{H}}_0}(T)$. However, the j sets in any collection need not be distinct. Since 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, 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 4, we have:

$$\begin{aligned}
\Pi_{\widehat{\mathcal{H}}_{1/4}}(m) &\leq \sum_{j=1}^{k_1} \binom{\Pi_{\widehat{\mathcal{H}}_0}(m) + j - 1}{j} \\
&= \sum_{j=0}^{k_1} \binom{\Pi_{\widehat{\mathcal{H}}_0}(m) + j - 1}{j} - 1 \\
&= \binom{\Pi_{\widehat{\mathcal{H}}_0}(m) + k_1}{k_1} - 1 \\
&\leq (\Pi_{\widehat{\mathcal{H}}_0}(m) + 1)^{k_1} - 1.
\end{aligned}$$

Combining the above results, we have:

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

Now, assume that $\Pi_{\mathcal{Q}_0}(m) \geq 1$ and $\Pi_{\widehat{\mathcal{H}}_0}(m) \geq 2$. These assumptions are 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:

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

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.

$$\begin{aligned}
2^{(k_1+1)(k_2+1)+3}(em/q_0)^{q_0}(em/d_0)^{d_0 k_1(k_2+1)} &\leq 2^m \\
\Leftrightarrow \quad ((k_1+1)(k_2+1)+3) + q_0 \lg(em/q_0) + d_0 k_1(k_2+1)\lg(em/d_0) &\leq m \qquad (13)
\end{aligned}$$

Inequality (13) has the same form as Inequality (12). 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. Thus, we have proven the following:

Lemma 8 $VC(\mathcal{Q}_{HB}) = O(VC(\mathcal{Q}_0) + VC(\mathcal{H}_0)\cdot k_1k_2\log(k_1k_2))$

C A Lower Bound for Probabilistic SQ Algorithms

Throughout this paper, 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 5.2.

Consider the two player game described in Section 5.2. 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 $\delta > 0$. We show the following:

Lemma 9 *For any $d \geq 4$, $t \leq d/4$, $\delta \leq 1/8$ and $N = \Omega(d^{1+\alpha})$ for any constant $\alpha > 0$, the probabilistic player requires $\Omega(\frac{d\log N}{\log(d\log N)})$ queries to succeed with probability at least $1-\delta$.*

Proof: By incorporating techniques from Kearns' lower bound proof (Kearns, 1993), we can modify the original proof of Lemma 3 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 $S \in \mathcal{S}_0^i$ satisfies $|S \cap Q_1| = i$. 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 aggregate cardinality of such "small" sets is less than $|\mathcal{S}_0|/k$. Thus, we have

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

where the probability is with respect to the random selection of the target set. 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 greater than 1/4, we have a lower bound on the size of $\mathcal{S}_k$. We next show that if the size of $\mathcal{S}_k$ 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 $|S \bigtriangleup T| = i$. Thus, there are exactly $\left(\!\binom{N}{t}\!\right)$ sets $S \in 2^{[N]}$ such that $|S \bigtriangleup T| \leq t$. 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. Furthermore, since $|\mathcal{S}_k| \geq \binom{N}{d}/(k\cdot(d+1))^k$ with probability greater than 1/4, the player will fail with probability greater than 1/8 if it halts after k questions for any k which satisfies the following inequalities:

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

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:

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

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

Each of these inequalities is satisfied 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 6, we immediately obtain the following:

Theorem 8 *There exists a parameterized family of function classes which is learnable in the SQ model and requires* $\Omega(\frac{d\log(1/\varepsilon)}{\log(d\log(1/\varepsilon))})$ *queries, each with additive error* $O(\varepsilon)$*, to learn in the SQ model by a probabilistic algorithm.*

References

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

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

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

Aslam, Javed and Scott Decatur. (1994). Improved noise-tolerant learning and generalized statistical queries. Technical Report TR-17-94, Harvard University, July.

Aslam, Javed and Scott Decatur. (1995). Specification and simulation of statistical query algorithms for efficiency and noise tolerance. In *Proceedings of the Eighth Annual ACM Workshop on Computational Learning Theory.* ACM Press, July. Invited to a special issue of *J Comp Sys Sci.*

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

Drucker, Harris, Robert Schapire, and Patrice Simard. (1992). Improving performance in neural networks using a boosting algorithm. In *Advances in Neural Information Processing Systems.* Morgan Kaufmann.

Feller, William. (1968). *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley & Sons, third edition.

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

Freund, Yoav. (1992). 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.

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

Graham, Ronald, Donald Knuth, and Oren Patashnik. (1994). *Concrete Mathematics: A Foundation for Computer Science.* Addison-Wesley, second edition.

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

Kearns, Michael. (1993). 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.

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

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

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

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

Schapire, Robert E. (1992). *The Design and Analysis of Efficient Learning Algorithms.* MIT Press, Cambridge, MA.

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

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

Notes

1. Angluin and Laird (1988) introduced a general framework for learning in the presence of classification noise; however, their methods do not yield computationally efficient algorithms in most cases.
2. VC-dimension is a standard complexity measure for a space of $\{0, 1\}$-valued functions and is formally defined in Section B of the appendix.
3. Note that since $\tau_* = \Omega(\varepsilon)$ (Kearns, 1993), these bounds depend at least quadraticly on $1/\varepsilon$.
4. 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. Note that it is somewhat different than the standard soft-order notation.
5. A "fair coin" hypothesis ignores its input x and outputs the outcome of a fair coin flip.
6. Note that the time complexity is based on unit cost queries to the statistics oracle.
7. We use the standard notation $[N] = \{1, \ldots, N\}$ and denote set symmetric difference by $\triangle$.
8. For this infinite instance space, we make use of an *unlabelled example oracle* (Kearns, 1993) in order to find all of the "heavily" weighted instances. Standard statistical queries are then used to determine which of these heavily weighted instances are in the target set.