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

Javed A. Aslam*
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Scott E. Decatur†
Aiken Computation Laboratory
Harvard University
Cambridge, MA 02138

Abstract

We derive general bounds on the complexity of learning in the Statistical Query 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 Statistical Query model. This new model was introduced by Kearns [12] 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. Specifically, we derive simultaneous upper bounds with respect to ϵ on the number of queries, $O(\log^2 \frac{1}{\epsilon})$, the Vapnik-Chervonenkis dimension of the query space, $O(\log \frac{1}{\epsilon} \log\log \frac{1}{\epsilon})$, and the inverse of the minimum tolerance, $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$. In addition, we show that these general upper bounds are nearly optimal by describing a class of learning problems for which we simultaneously lower bound the number of queries by $\Omega(\log \frac{1}{\epsilon})$ and the inverse of the minimum tolerance by $\Omega(\frac{1}{\epsilon})$.

We further apply our boosting results in the SQ model to learning in the PAC model with classification noise. Since nearly all PAC learning algorithms can be cast in the SQ model, we can apply our boosting techniques to convert these PAC algorithms into highly efficient SQ algorithms. By simulating these efficient SQ algorithms in the PAC model with classification noise, we show that nearly all PAC algorithms can be converted into highly efficient PAC algorithms which tolerate classification noise. We give an upper bound on the sample complexity of these noise-tolerant PAC algorithms which is nearly optimal with respect to the noise rate. We also give upper bounds on space complexity and hypothesis size and show that these two measures are in fact *independent* of the noise rate. We note that the running times of these noise-tolerant PAC algorithms are efficient.

This sequence of simulations also demonstrates 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 [15] and the first theoretical evidence for an empirical result of Drucker, Schapire and Simard [4].

1 Introduction

We derive general bounds on the complexity of learning in the Statistical Query model and in the PAC model with classification noise. We do so by considering the problem of improving the accuracy of learning algorithms, and in particular we study the problem of "boosting" the accuracy of "weak" learning algorithms which fall within the recently introduced Statistical Query model. We show that it is possible to improve the accuracy of weak learning algorithms in the Statistical Query model to any arbitrary accuracy, and we derive a number of interesting consequences from this result.

Since Valiant's introduction of the Probably Approximately Correct model of learning [18], 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 example according to a distribution D and returns the example along with its label according to f. The error rate of an hypothesis

*Author was supported by DARPA Contract N00014-87-K-825 and by NSF Grant CCR-89-14428. Author's net address: jaa@theory.lcs.mit.edu

†Author was supported by an NDSEG Fellowship and by NSF Grant CCR-92-00884. Author's net address: sed@das.harvard.edu

output by the learner is the probability that an example 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 then the accuracy parameter. Two standard complexity measures studied in the PAC model are *sample complexity* and *time complexity*. Efficient PAC learning algorithms have been developed for many function classes [1], and PAC learning continues to be a popular model of machine learning.

The model of learning stated above is often referred to as *strong learning* since the learner could be required to output an arbitrarily accurate hypothesis depending on the accuracy parameter. A variant of strong learning called *weak learning* is identical except that there is no accuracy parameter, and the output hypothesis need only have error rate slightly less than 1/2 (*i.e.* slightly better than random guessing). A fundamental and surprising result first shown by Schapire [15, 16] and later improved upon by Freund [6, 7] states that any algorithm which efficiently weakly learns can be transformed into an algorithm which efficiently strongly learns. These results have important consequences for PAC learning, including providing upper bounds on the time and sample complexities of strong learning.

One criticism of the PAC model is that the data presented to the learner is assumed to be *noise-free*. In fact, most of the standard PAC learning algorithms would fail if even a small number of the labelled examples given to the learning algorithm were "noisy". A popular noise model for both theoretical and experimental research is the *classification noise* model introduced by Angluin and Laird [2, 13]. In this model, each example received by the learner is mislabelled randomly and independently with some fixed probability. While a limited number of efficient PAC algorithms have been developed which can tolerate classification noise [2, 9, 14], no general framework for efficient learning[1] in the presence of classification noise was known until Kearns introduced the Statistical Query model [12].

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

Kearns [12] has demonstrated two important properties of the SQ model which make it worthy of study. First, he has shown that nearly every PAC learning algorithm can be cast within the SQ model, thus demonstrating that the SQ model is quite general and imposes a rather weak restriction on learning algorithms. Second, he has shown that calls to the statistics oracle can be efficiently simulated (with high probability) by a sufficiently large sample drawn from a *classification noise oracle*. An immediate consequence of these two properties is that nearly every efficient PAC learning algorithm can be transformed into one which tolerates arbitrary amounts of classification noise.

While greatly expanding the class of functions known to be learnable in the presence of classification noise, Kearns' technique does not constitute a formal reduction from PAC learning to SQ learning. In fact such a reduction cannot exist since, while the class of parity functions is known to be PAC learnable [11], Kearns has shown that this class is provably unlearnable in the SQ model. Kearns' technique for converting PAC algorithms to SQ algorithms consists of a few general rules, but each PAC algorithm must be examined in turn and converted to an SQ algorithm individually. Thus, one cannot derive general upper bounds on the complexity of SQ learning from upper bounds on the complexity of PAC learning, due to the dependence on the specific conversion of a PAC algorithm to an SQ algorithm. A consequence of this fact is that general upper bounds on the time and sample complexities of PAC learning in the presence of classification noise are not directly obtainable either.

We obtain bounds for SQ learning and PAC learning with classification 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" [7] which is nearly optimal in terms of its dependence on hypothesis accuracy.

In the SQ model, as in the PAC model, the boosting result allows us to derive upper bounds

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

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

We further apply our boosting result and bounds in the SQ model to derive a boosting result and 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 [15] 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 [4] 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 and η_b) of $\tilde{O}(\frac{1}{\epsilon^2(1-2\eta_b)^2})$ on the sample complexity of PAC learning in the presence of noise which is nearly optimal with respect to the noise rate.[2] We also give bounds on the space complexity and hypothesis size. Our bound on the hypothesis size is *independent* of the noise rate, and by using a modest increase in sample size we achieve a space complexity that is also independent of the noise rate.

The remainder of this paper is organized as follows. In Section 2, we formally define the learning models of interest. 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. Sections 5 and 6 describe the consequences of these results for SQ learning and PAC learning in the presence of classification noise. We conclude the paper with some extensions and open questions in Section 7.

[2]When $b > 1$, we define $\tilde{O}(b)$ to mean $O(b \log^c b)$ for some $c > 1$. We define $\tilde{\Omega}$ and $\tilde{\Theta}$ similarly.

2 Model Definitions

In this section, we formally define the relevant models of machine learning necessary for the exposition that follows. 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 example space X. The example space X is typically either the Boolean hypercube $\{0,1\}^n$ or n-dimensional Euclidean space $\Re^n$. We use the parameter n to denote the common length of each example $x \in X$.

We assume that the examples are distributed according to some unknown probability distribution D on X. The learner is given access to a 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 example $x \in X$ is drawn randomly and independently according to the unknown distribution D, and the label $l = f(x)$. We often refer to a sequence of labelled examples drawn from an example oracle as a sample.

The learning algorithm will draw a sample from $EX(f, D)$ and eventually output 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 distribution weight of those examples in X where h and f differ. By using the notation $\Pr_D[P(x)]$ to denote the distribution weight of examples in X which satisfy the predicate P, we may define $error(h) = \Pr_D[h(x) \neq f(x)]$. We often think of $\mathcal{H}$ as a class of representations of functions in $\mathcal{F}$, and as such we define $size(f)$ 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 *error parameter* ϵ and *confidence parameter* δ. A learning algorithm is said to be efficient if its running time is polynomial in $1/\epsilon$, $1/\delta$, n and $size(f)$. We formally define PAC learning as follows (adapted from Kearns [12]):

Definition 1 *(Strong PAC learning)*
Let $\mathcal{F}$ and $\mathcal{H}$ be function classes defined over X. The class $\mathcal{F}$ is said to be efficiently learnable by $\mathcal{H}$ if there exists a learning algorithm A and a polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that for any $f \in \mathcal{F}$, for any distribution D on X, for any error parameter ϵ, $0 < \epsilon \leq 1$,

and for any confidence parameter δ, $0 < \delta \leq 1$, the following holds: if A is given inputs ϵ and δ, and access to an example oracle $EX(f, D)$, then A halts in time bounded by $p(1/\epsilon, 1/\delta, n, size(f))$ and outputs an hypothesis $h \in \mathcal{H}$ that with probability at least $1 - \delta$ satisfies $error(h) \leq \epsilon$.

As stated, this is often referred to as *strong learning* since the learner could 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 error 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

One criticism of the PAC model is that the data presented to the learner is required to be *noise-free.* A popular model of noise for both experimental and theoretical purposes was introduced by Angluin and Laird [2, 13]. In the *classification noise model,* the labelled example oracle $EX(f, D)$ is replaced by a noisy example oracle $EX^{\eta}(f, D)$. Each time the noisy example oracle is called, an example $x \in X$ is drawn according to D. The oracle then outputs $\langle x, f(x)\rangle$ with probability $1 - \eta$ and $\langle x, \neg f(x)\rangle$ with probability η, randomly and independently for each example drawn. Classification noise is intended to model the simplest type of "white noise" which can affect the labels. 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 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

While a limited number of PAC algorithms have been developed which can tolerate arbitrary amounts of classification noise (up to the information-theoretic limit of 1/2) [2, 9, 14], no general framework for efficient learning in the presence of classification noise was known until Kearns introduced the Statistical Query model [12]. In the SQ model, the example oracle $EX(f, D)$ from the standard PAC model is replaced by a statistics oracle $STAT(f, D)$. An SQ algorithm queries the $STAT$ oracle for the values of various statistics on the distribution of labelled examples (*e.g.* "What is the probability that a randomly chosen labelled example $\langle x, l\rangle$ has variable $x_i = 0$ and $l = 1$?"), and the $STAT$ oracle returns the requested statistics to within some specified tolerance. 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 a tolerance parameter. A call to $STAT(f, D)[\chi, \tau]$ returns an *estimate* $\hat{P}_\chi$ of $P_\chi = \Pr_D[\chi(x, f(x))]$ which satisfies $|\hat{P}_\chi - P_\chi| \leq \tau$.

We note that a call to $STAT(f, D)$ can easily be simulated (with probability at least $1 - \delta$) by drawing a sufficiently large sample from $EX(f, D)$ and outputting the fraction of labelled examples which satisfy $\chi(x, f(x))$ as our estimate $\hat{P}_\chi$. The size of the sample required will be polynomial in $1/\tau$ and $\log 1/\delta$, and the simulation time will also be dependent on the time required to evaluate χ. We formally define efficient learning in the Statistical Query model as follows (adapted from Kearns [12]):

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

Since calls to the statistics oracle $STAT(f, D)$ can be simulated (with high probability) by a sample drawn from the example oracle $EX(f, D)$, we can view the Statistical Query model as simply restricting the way in which learning algorithms can use labelled examples. Kearns has shown that this restriction is rather weak in that nearly every PAC algorithm can be cast in the SQ model. An important property of this model is that calls to the statistics oracle $STAT(f, D)$ can also be efficiently simulated (with high probability) by a sample drawn from a *noisy* example oracle $EX^{\eta}(f, D)$.[3] The size of the sample

[3]The statistics oracle can actually be simulated by a *variety* of "faulty" example oracles [3].

required is polynomial in $1/\tau$, $1/(1-2\eta_b)$ and $\log 1/\delta$. While an efficient simulation of an SQ algorithm can be obtained by drawing a separate sample for each call to the statistics oracle, better bounds on the sample complexity of the simulation are obtained by drawing one large sample and estimating each statistical query using that single sample. If we let $\mathcal{Q}$ be the function space from which the learning algorithm selects its queries (the ***query space***), then the size of the single sample required in this case is independent of the query complexity, but depends on the VC-dimension of $\mathcal{Q}$ (a standard complexity measure for a space of functions). Kearns has shown the following theorem on the sample size required for this simulation:

Theorem 1 (Kearns) *Let $\mathcal{F}$ and $\mathcal{H}$ be function classes defined over X. Suppose that $\mathcal{F}$ is efficiently learnable via statistical queries by $\mathcal{H}$ using a learning algorithm A. Then $\mathcal{F}$ is efficiently learnable in the classification noise model by $\mathcal{H}$. If A uses query space $\mathcal{Q}$ of VC dimension q and τ_0 is a lower bound on the tolerance of every query made by A, then the number of calls to $EX^{\eta}(f, D)$ required to learn with noise is*

$$O\left(\frac{q\log(1/\tau_0(1-2\eta_b))}{{\tau_0}^2(1-2\eta_b)^2} + \frac{\log(1/\delta)}{{\tau_0}^2(1-2\eta_b)^2} + \frac{1}{\epsilon^2}\log\frac{1}{\delta\tau_0(1-2\eta_b)}\right).$$

Given that nearly every PAC algorithm can be converted to an SQ algorithm, an immediate consequence of this theorem is that nearly every PAC algorithm can be transformed into one which tolerates classification noise. The sample and time complexities of the noise-tolerant versions depend on τ_0 and $\mathcal{Q}$, which themselves are a function of the ***ad hoc*** conversion of the PAC algorithms to SQ algorithms. Thus, one cannot show general upper bounds on the sample and time complexities of these noise-tolerant versions of converted PAC algorithms.

We define weak SQ learning identically to strong SQ learning except that there is no error parameter ϵ. In this case the output hypothesis need only have error rate slightly less than 1/2, *i.e.* $error(h) \le \frac{1}{2} - \gamma = \frac{1}{2} - \frac{1}{p(n, size(f))}$ for some polynomial p. By showing that weak SQ learning algorithms can be "boosted" to strong SQ learning algorithms, we are able to bound the tolerance τ_0 of strong SQ algorithms and limit the query space $\mathcal{Q}$ as well. We are then able to show an upper bound on the sample and time complexities of the noise-tolerant versions of converted PAC algorithms. These results are given in Sections 4, 5 and 6.

3 Boosting in the PAC model

Schapire and Freund use similar strategies for boosting weak learning algorithms into strong learning algorithms. They both create a strong hypothesis by combining many hypotheses obtained from the weak learning algorithm. The boosting mechanisms derive their power by presenting the different calls to the weak learning algorithm with modified versions of the original distribution over labelled examples. Freund [6, 7] has developed two similar schemes (which we call Scheme 1 and Scheme 2) for boosting a weak learning algorithm into a strong learning algorithm. One is more efficient with respect to ϵ while the other is more efficient with respect to γ. Freund constructs a hybrid scheme more efficient than either Scheme 1 or Scheme 2 by combining these two methods in order to capitalize on the advantages of each. We first describe the two schemes separately and then show how to combine them.

Scheme 1 uses the weak learning algorithm to create a set of $k_1 = \frac{1}{2\gamma^2}\ln\frac{1}{\epsilon}$ weak hypotheses and outputs the majority vote of these hypotheses as the strong hypothesis. The weak hypotheses are created by asking the weak learner to learn on various modified distributions. The distribution used to generate a given weak hypothesis is based on the performance of the previously generated weak hypotheses. Specifically, to create the $(i+1)^{st}$ hypothesis, instead of the given example oracle $EX(f, D)$, the weak learner is provided a ***filtered*** example oracle $EX(f, D_{i+1})$ defined as follows:

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

Figure 1: Filtered Example Oracle $EX(f, D_{i+1})$

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

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

Freund shows that the majority vote of $h_1, \ldots, h_{k_1}$ will have error rate no more than ϵ on D if each h_i has error rate no more than $\frac{1}{2} - \gamma$ on D_i.

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 $t_i < c_1\epsilon^2$, then the majority vote of $h_1, \ldots, h_i$ is already a strong hypothesis. The boosting algorithm can estimate t_i, and if it is below the cutoff, can output the majority vote of the hypotheses created thus far. The boosting algorithm's time and sample complexity dependence on γ is $\tilde{\Theta}(1/\gamma^2)$ while its dependence on ϵ is $\tilde{O}(1/\epsilon^2)$.

Scheme 2 is very similar to Scheme 1. The weak learner is again called many times to provide weak hypotheses on filtered distributions. This method uses $k_2 = 2k_1 = \frac{1}{\gamma^2}\ln\frac{1}{\epsilon}$ weak hypotheses, while the filtered oracle remains the same. The main difference is the observation that if $t_i < \frac{\epsilon(1-\epsilon)\gamma}{\ln(1/\epsilon)}$, then we may simply use a "fair coin" in place of h_{i+1} and still be guaranteed that the final majority of k_2 hypotheses has error rate less than ϵ.[4] The boosting algorithm tests to see if t_i is below this new threshold. If so, a "fair coin" is used as the $(i+1)^{st}$ weak hypothesis, and the algorithm proceeds to find a weak hypothesis on the next distribution. The boosting algorithm's time and sample complexity dependence on ϵ is $\tilde{\Theta}(1/\epsilon)$ while its dependence on γ is $\tilde{O}(1/\gamma^3)$.

The improvement on the 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 is more efficient in $1/\epsilon$. We therefore use the first to boost from $\frac{1}{2} - \gamma$ to a constant and use the second to boost from that constant to ϵ. We define $A_{\frac{1}{4}}$ to be a learning algorithm which uses the first boosting scheme and makes calls to the weak learning algorithm $A_{\frac{1}{2}-\gamma}$. The strong learning algorithm A_ϵ uses the second boosting scheme and makes calls to $A_{\frac{1}{4}}$ 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 for various distributions. The hybrid's time and sample complexity dependence on ϵ is $\tilde{\Theta}(1/\epsilon)$ while its dependence on γ is $\tilde{\Theta}(1/\gamma^2)$.

4 Boosting in the SQ model

Boosting requires the learner to interact with modified distributions over labelled examples. Specifically, the boosting methods of the previous section are based on the observation that the majority vote of $h_1, \ldots, h_k$ has error rate less than ϵ on D if each constituent h_j has error rate less than $\frac{1}{2} - \gamma$ on D_j. In the PAC model, the learner interacts with the distribution over labelled examples 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, the learner interacts with the distribution over labelled examples through calls to a statistics oracle. Therefore, boosting in the SQ model is accomplished by constructing $STAT(f, D_j)$ from the original statistics oracle $STAT(f, D)$.

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

4.1 Scheme 1 via Statistical Queries

We can use Scheme 1 to create a strong SQ learner by simply answering statistical queries made with respect to modified distributions. Therefore, we must be able to simulate queries to $STAT(f, D_j)$ by making queries to $STAT(f, D)$. We first show how to specify the exact value of a query with respect to D_j in terms of queries with respect to D and then determine the accuracy with which we need to make these queries with respect to D in order to obtain the correct 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 Figure 1. Note that steps 2 and 3 partition the instance space into $i + 1$ regions corresponding to those examples which are correctly classified by the same number of hypotheses.[5] Let $X_r^i \subseteq X$ be the set of examples which are correctly classified by exactly r of the i hypotheses. We define an induced distribution D_Z on a set Z with respect to distribution D as follows: For any $Y \subseteq Z$, $D_Z[Y] = D[Y]/D[Z]$. By construction, for any given X_r^i region, the filtered example oracle uniformly scales the probability with which examples from that region are drawn. Therefore, the induced distribution on X_r^i with respect to D_{i+1} is the same as the induced distribution on X_r^i with respect to D. (This fact will be used to obtain Equation 2 from Equation 1 below.)

A query by the weak learner to $STAT(f, D_{i+1})$ is a call for an estimate of $\Pr_{D_{i+1}}[\chi(x, f(x))]$ as given in Equation 1. Note that the denominator of Equation 3 is the probability that a random example drawn from $EX(f, D)$ passes through the filter which defines $EX(f, D_{i+1})$. Recall that Freund calls this probability t_i.

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

[5] For ease of presentation, throughout this paper we assume that the hypothesis class $\mathcal{H}$ is composed of deterministic hypotheses. In Section 7 we describe the generalizations used to accommodate probabilistic hypotheses.

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

Ignoring tolerances for the moment, the probabilities in Equation 3 may be stated as queries to $STAT(f,D)$ as follows:

$$
\begin{aligned}
&STAT(f,D_{i+1})[\chi(x,f(x))]\\
&\quad = \frac{\sum_{r=0}^{i} \alpha_r^i \cdot STAT(f,D)[\chi(x,f(x)) \wedge (x \in X_r^i)]}{\sum_{j=0}^{i} \alpha_j^i \cdot STAT(f,D)[x \in X_j^i]} \qquad (4)
\end{aligned}
$$

Note that the condition $x \in X_j^i$, while possibly not a small Boolean formula, is polynomially evaluatable given $h_1,\ldots,h_i$, thus satisfying the efficiency condition given in the definition of SQ learning. We next determine the accuracy with which we must ask these queries to $STAT(f,D)$ so that the final result is within the desired tolerance τ.

Since Equation 4 is the ratio of two quantities, we first show a sufficient tolerance with which one can estimate two quantities in order to determine an accurate estimate of their ratio.

Claim 1 *Let $0 \le a,b,c,\tau \le 1$ where $a = b/c$. If $\hat{b}$ and $\hat{c}$ are estimates of b and c, respectively, each with additive error no more than $\tau c/3$, then $|a - \hat{b}/\hat{c}| \le \tau$.*

The claim follows immediately from the simple verification of the following two inequalities:

$$\frac{b+\tau c/3}{c-\tau c/3} \le a+\tau \quad\text{and}\quad \frac{b-\tau c/3}{c+\tau c/3} \ge a-\tau$$

We now have that it is sufficient to estimate both the numerator and denominator of Equation 4 each with tolerance $\tau t_i/3$ in order to simulate a call to $STAT(f,D_{i+1})$ with tolerance τ. Both the numerator and denominator are of the form: $\sum_i p_i z_i$. When $\sum_i |p_i| \le 1$, one can easily show that to estimate this sum to within some additive error, it is sufficient to estimate each z_i to within this same additive error. Since the known α_r^i probabilities are binomially distributed, their sum is 1. Therefore, to estimate the numerator and denominator of Equation 4 each to within $\tau t_i/3$, it is sufficient to estimate each of the individual queries to within this same tolerance. It is therefore sufficient to estimate each call to $STAT(f,D)$ with tolerance $\tau t_i/3$ in order to simulate a call to $STAT(f,D_{i+1})$ with tolerance τ.

If t_i is very small, then we must make estimates with very small tolerances. We would therefore like to have some good lower bound on t_i when we are simulating calls to $STAT(f,D_{i+1})$. We obtain such a lower bound by employing the bailout condition of Freund which either lower bounds t_i or aborts the search for h_{i+1}.

If $t_i < c_1\epsilon^2$, then the majority vote of the hypotheses generated thus far is a strong hypothesis. We therefore estimate t_i, and if it is less than the cutoff, we return the majority vote of the hypotheses found so far. Otherwise t_i is lower bounded by $\Omega(\epsilon^2)$, and the required tolerances for estimating queries to $STAT(f,D)$ are lower bounded by $\Omega(\tau\epsilon^2)$, where τ is the tolerance requested by the call to $STAT(f,D_{i+1})$. If we let $\tau_0 = \tau_0(n, size(f))$ be the minimum query tolerance requested by a weak learner, then the minimum query tolerance required by our simulation is bounded by $\Omega(\tau_0\epsilon^2)$.

We next examine the *number* of queries required for boosting as a function of the number of queries required by an individual weak learner. Let $N = N(n, size(f))$ be the maximum number of queries made by a weak learner. By Equation 4, we note that the number of queries to $STAT(f,D)$ required to simulate a query to $STAT(f,D_{i+1})$ is proportional to i. We therefore need $O(Nk_1{}^2) = O(N\frac{1}{\gamma^4}\log^2\frac{1}{\epsilon})$ queries to simulate all of the queries requested by all k_1 of the weak learners.

4.2 Scheme 2 via Statistical Queries

The SQ simulation of Scheme 2 is very similar to the simulation of Scheme 1. Since the bailout condition of Scheme 2 introduces "coin flip" hypotheses in addition to the usual weak hypotheses, we first rederive the probability of $\chi(x,f(x))$ being true with respect to

$$\Pr_{D_{i+1}}[\chi(x,f(x))] = \frac{\sum_{r=0}^{i} \alpha_r^i \cdot \Pr_D[\chi(x,f(x)) \wedge (x \in T_r^i)]}{\sum_{j=0}^{i} \alpha_j^i \cdot \Pr_D[x \in T_j^i]} \tag{5}$$

$$= \frac{\sum_{r=0}^{i} \alpha_r^i \cdot (\sum_{l=0}^{v} \beta_l^v \cdot \Pr_D[\chi(x,f(x)) \wedge (x \in X_{r-l}^w)])}{\sum_{j=0}^{i} \alpha_j^i \cdot \left(\sum_{l=0}^{v} \beta_l^v \cdot \Pr_D[x \in X_{j-l}^w]\right)} \tag{6}$$

$$STAT(f, D_{i+1})[\chi(x,f(x))] = \frac{\sum_{r=0}^{i} \alpha_r^i \cdot (\sum_{l=0}^{v} \beta_l^v \cdot STAT(f,D)[\chi(x,f(x)) \wedge (x \in X_{r-l}^w)])}{\sum_{j=0}^{i} \alpha_j^i \cdot \left(\sum_{l=0}^{v} \beta_l^v \cdot STAT(f,D)[x \in X_{j-l}^w]\right)}$$

D_{i+1} in terms of probabilities with respect to D.

After i hypotheses have been generated we let w be the number of these which are actual weak hypotheses and $v = i - w$ be the number of these which are "coin flip" hypotheses. If we evaluate all i hypotheses on an example x, we say that $x \in T_r^i$ if exactly r of the i hypotheses agree with f. We say that $x \in X_r^w$ if exactly r of the w *weak* hypotheses agree with f. Note that $\Pr_D[x \in T_r^i] = \sum_{l=0}^{v} \beta_l^v \cdot \Pr_D[x \in X_{r-l}^w]$ where we define β_l^v to be the probability that exactly l of v fair coin flips are HEADS and $\Pr_D[x \in X_{r-l}^w] = 0$ if $r - l < 0$. In Equations 5 and 6, we formulate the probability of $\chi(x, f(x))$ being satisfied with respect to D_{i+1} in terms of probabilities with respect to D. Note that the denominators of these equations again correspond to the probability t_i.

The bailout condition in this scheme implies that if we must produce a weak hypothesis, then we have an $\Omega(\epsilon\gamma/\log\frac{1}{\epsilon})$ bound on the denominator t_i. Since the β_l^v (and α_r^i) coefficients are binomially distributed and therefore sum to 1, we may make queries to $STAT(f, D)$ with additive error $\Omega(\tau\epsilon\gamma/\log\frac{1}{\epsilon})$ to achieve additive error τ on the simulation of queries to $STAT(f, D_{i+1})$. If we let $\tau_0 = \tau_0(n, size(f))$ be the minimum query tolerance requested by a weak learner, then the minimum query tolerance required by our simulation is bounded by $\Omega(\tau_0\epsilon\gamma/\log\frac{1}{\epsilon})$. The number of queries required in this boosting scheme is $O(Nk_2{}^2) = O(N\frac{1}{\gamma^4}\log^2\frac{1}{\epsilon})$ where N is the maximum number of queries made by a weak learner.

4.3 Hybrid SQ boosting

We obtain a more efficient boosting scheme in the SQ model by combining the two previously described methods. As was done 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. We can therefore boost from weak to strong learning in the SQ model as stated in the following theorem.

Theorem 2 *Given a weak SQ learning algorithm which makes at most $N = N(n, size(f))$ queries each of tolerance at least $\tau_0 = \tau_0(n, size(f))$ and outputs an hypothesis with error no more than $\frac{1}{2} - \gamma$, we can construct a strong SQ learning algorithm which makes $O(N\frac{1}{\gamma^4}\log^2\frac{1}{\epsilon})$ queries each of tolerance $\Omega(\tau_0\epsilon/\log\frac{1}{\epsilon})$.*

Note that by using this hybrid boosting scheme, the minimum query tolerance of the constructed strong SQ learning algorithm has no dependence on γ.

5 General Bounds on Learning in the Statistical Query Model

In the same way that the sample complexity of boosting in the PAC model yields general upper bounds on the sample complexity of strong PAC learning algorithms, the query and tolerance complexities of boosting in the SQ model yield general bounds on the query and tolerance complexities of strong SQ learning algorithms.

We can convert any strong SQ learning algorithm into a weak SQ learning algorithm by "hardwiring" the accuracy parameter ϵ to a constant. We may then use the boosting technique to arrive at a strong SQ learning algorithm with nearly optimal dependence on ϵ.

Theorem 3 *If concept class $\mathcal{F}$ is strongly SQ learnable then $\mathcal{F}$ is strongly SQ learnable with $O(N\log^2\frac{1}{\epsilon})$ queries each of tolerance $\Omega(\tau_0\epsilon/\log\frac{1}{\epsilon})$. Here N and τ_0 are the number of queries and minimum tolerance, respectively, of the original strong SQ algorithm run with a constant accuracy parameter. Both N and τ_0 are independent of ϵ.*

Further analysis of the boosting scheme shows that the dependence on ϵ of the time complexity, space complexity and hypothesis size of strong SQ learning are $O(\log^2\frac{1}{\epsilon})$, $O(\log\frac{1}{\epsilon})$ and $O(\log\frac{1}{\epsilon})$, respectively.

Kearns [12] has shown a general lower bound of $\Omega(d/\log d)$ queries each of tolerance $O(\epsilon)$ for learning any concept class of VC-dimension d in the SQ model. Whereas Kearns simultaneously lower bounds the number of queries and upper bounds the minimum tolerance, here we have simultaneously upper bounded the number of queries and lower bounded the minimum tolerance in terms of their dependence on ϵ. Note that the tolerance we give in Theorem 3 is optimal to within a logarithmic factor. While Kearns' general lower bound leaves open the possibility that there may exist a general upper bound on the query complexity which is independent of ϵ, we show that this is not the case by demonstrating a specific learning problem which requires $\Omega(\frac{d}{\log d}\log\frac{1}{\epsilon})$ queries of tolerance $O(\epsilon)$ in the SQ model. Thus, with respect to ϵ, our general upper bound on the query complexity is within a $\log\frac{1}{\epsilon}$ factor of the best possible general upper bound.

Theorem 4 *There exists a parameterized family of function classes which require $\Omega(\frac{d}{\log d}\log\frac{1}{\epsilon})$ queries of tolerance $O(\epsilon)$ to learn in the SQ model.*

Proof (Sketch): Consider the following two-player game parameterized by t, d and N where $t \leq d \leq N$. The *adversary* chooses a set[6] $S \subseteq [N]$ of size d, and the goal of the *player* is to output a set $T \subseteq [N]$ such that $|S \Delta T| \leq t$. The player is allowed to ask *queries* of the form $Q \subseteq [N]$ to which the adversary returns $|Q \cap S|$. For any $d \geq 4$, $t \leq d/4$ and $N = \Omega(d^{1+\alpha})$ for some $\alpha > 0$, we show that the player requires $\Omega(\frac{d}{\log d}\log N)$ queries to the adversary, in the worst case.

We reduce this two-player game to the following learning problem. The example space X is the set of natural numbers $\mathcal{N}$, and the function class $\mathcal{F}$ is the set of all indicator functions corresponding to subsets of $\mathcal{N}$ of size d. By appropriately creating a distribution over $\mathcal{N}$ and setting $\epsilon = 1/N$, the reduction allows us 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 tolerance is greater than 4ϵ. Since $\Omega(\frac{d}{\log d}\log N)$ queries of the adversary are required, the SQ algorithm must ask $\Omega(\frac{d}{\log d}\log N) = \Omega(\frac{d}{\log d}\log\frac{1}{\epsilon})$ queries of tolerance $O(\epsilon)$. □

6 General Bounds on PAC Learning with Classification Noise

Given the bounds of the previous section, an analysis of the VC-dimension of the query space $\mathcal{Q}$ used in the boosting scheme, and Theorem 1 of Kearns, 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).

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

Lemma 1 *Let $\mathcal{Q}_0$ and $\mathcal{H}_0$ be the query space and hypothesis class, respectively, of the weak learning algorithm. Let $\mathcal{Q}$ be the query space of the two stage boosting scheme. Then $VCD(\mathcal{Q}) = \tilde{O}(kd_0 + q_0)$ where $k = O(\frac{1}{\gamma^2}\log\frac{1}{\epsilon})$ is the number of calls to the weak learning algorithm in the two stage boosting scheme, $d_0 = VCD(\mathcal{H}_0)$ and $q_0 = VCD(\mathcal{Q}_0)$.*

Theorem 5 *Let $\mathcal{F}$ and $\mathcal{H}_0$ be function classes defined over X. Suppose that $\mathcal{F}$ is efficiently learnable via statistical queries by $\mathcal{H}_0$ using learning algorithm A_0. If A_0, run with a fixed constant accuracy, uses query space $\mathcal{Q}_0$ of VC-dimension q_0 and $\tau_0 = \tau_0(n, size(f))$ is a lower bound on the tolerance of every query made by A_0, then $\mathcal{F}$ is efficiently learnable in the classification noise model, and ignoring lower order factors, the number of calls to $EX^\eta(f,D)$ required is*

$$O\left(\frac{q_0 + d_0}{{\tau_0}^2\epsilon^2(1-2\eta_b)^2} + \frac{1}{{\tau_0}^2\epsilon^2(1-2\eta_b)^2}\log\frac{1}{\delta}\right).$$

One can also show that the dependence on ϵ and η_b of the space complexity of learning $\mathcal{F}$ is $\tilde{O}(\frac{1}{\epsilon^2(1-2\eta_b)^2})$, while the dependence on ϵ and η_b of the hypothesis size is $O(\log\frac{1}{\epsilon})$. Simon [17] has shown that the sample complexity of PAC learning with classification noise is $\Omega(\frac{1}{\epsilon(1-2\eta)^2})$. Therefore our sample complexity bound is roughly optimal with respect to η_b. Note that the hypothesis size is independent of the noise rate. Furthermore, with a polynomial increase in sample size, one can simulate each query with a separate sample and achieve a reduced space complexity of $O(\log\frac{1}{\epsilon})$, which is also independent of the noise rate. We finally note that the time complexity of learning $\mathcal{F}$ is polynomially bounded in all relevant learning parameters.[7]

7 Discussion

Throughout this paper, we have assumed that the hypothesis classes used by all weak learning algorithms are composed solely of deterministic hypotheses. However, Goldman, Kearns and Schapire [10] have shown that in many cases, algorithms which are allowed to output probabilistic hypotheses are more efficient than algorithms which are required to output deterministic hypotheses. By allowing weak learning algorithms to output probabilistic hypotheses, our boosting algorithm may construct probabilistic χ's. We note that the techniques given in this paper can easily be modified to allow for probabilistic hypothesis classes and probabilistic χ's — in addition to taking all probabilities with respect to the random choice of examples, we also take these probabilities with respect to

[7]The time complexity of learning $\mathcal{F}$ is dependent on the exact simulation of $STAT(f,D)$ by $EX^\eta(f,D)$ in Theorem 1. We give a complete discussion of time complexity in the full paper.

the "randomness" of the χ's. In fact, the techniques given in this paper can also be modified to allow for real-valued χ's. In this case, a query submitted to the statistics oracle requests the *expected value* of χ as opposed to the probability that $\chi = 1$. Estimating the expectation of real-valued χ's in the presence of noise requires new techniques which we describe in the full paper. One can show that by using real-valued χ's, the query complexity of boosting in the SQ model can be somewhat reduced [8]; however, the complexity of the PAC simulation is not significantly improved.

Ehrenfeucht, *et al.* [5] have shown that the sample complexity of PAC learning depends at least linearly on $1/\epsilon$, and clearly this bound holds for learning in the presence of classification noise as well. Laird [13] has developed a general technique for learning in the presence of classification noise whose sample complexity depends only linearly on $1/\epsilon$; however, this technique does not yield computationally efficient algorithms. The upper bound given in Theorem 5 has a roughly quadratic dependence on $1/\epsilon$. 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 5 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 $o(1/\epsilon^2)$. Such a simulation would immediately yield improved sample complexity bounds for learning in the presence of classification noise. Conversely, if one can 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.

Acknowledgements

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

References

[1] Dana Angluin. Computational learning theory: Survey and selected bibliography. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, pages 351–369, May 1992.

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

[3] Scott E. Decatur. Statistical queries and faulty PAC oracles. In *Proceedings of the Sixth Annual ACM Workshop on Computational Learning Theory*. ACM Press, 1993.

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

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

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

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

[8] Yoav Freund. Personal communication, 1993.

[9] Sally A. Goldman, Michael J. Kearns, and Robert E. Schapire. Exact identification of circuits using fixed points of amplification functions. In *Proceedings of the 31st Symposium on Foundations of Computer Science*, pages 193–202. IEEE, October 1990.

[10] Sally A. Goldman, Michael J. Kearns, and Robert E. Schapire. On the sample complexity of weak learning. In *Proceedings of COLT '90*, pages 217–231. Morgan Kaufmann, 1990.

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

[12] Michael Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, 1993.

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

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

[15] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

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

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

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