INFORMATION AND COMPUTATION 121, 256–285 (1995)

Boosting a Weak Learning Algorithm by Majority

YOAV FREUND

AT & T Bell Laboratories, Murray Hill, New Jersey
E-mail: yoav@research.att.com

We present an algorithm for improving the accuracy of algorithms for learning binary concepts. The improvement is achieved by combining a large number of hypotheses, each of which is generated by training the given learning algorithm on a different set of examples. Our algorithm is based on ideas presented by Schapire and represents an improvement over his results. The analysis of our algorithm provides general upper bounds on the resources required for learning in Valiant's polynomial PAC learning framework, which are the best general upper bounds known today. We show that the number of hypotheses that are combined by our algorithm is the smallest number possible. Other outcomes of our analysis are results regarding the representational power of threshold circuits, the relation between learnability and compression, and a method for parallelizing PAC learning algorithms. We provide extensions of our algorithms to cases in which the concepts are not binary and to the case where the accuracy of the learning algorithm depends on the distribution of the instances.

1. INTRODUCTION

The field of computational learning is concerned with mathematical analysis of algorithms that learn from their experience. One of the main problems studied in computational learning theory is that of *concept learning*. Informally, a concept is a rule that divides the world into positive and negative examples. For instance, the concept of "being blue" divides all objects into those that are blue and those that are not blue. The learning algorithm is presented with examples of blue and non-blue objects and is required to deduce the general rule. More formally, we define the set of all possible objects as the *instance space* and define concepts as functions from the instance space to the labels "$-$" and "$+$". An instance, together with its label, is called an example. The goal of concept learning is to generate a (description of) another function, called the *hypothesis*, which is close to the concept, using a set of examples. In general, we require that the learning algorithm observe just a small fraction of the instance space and that the learner can *generalize* the information provided by these examples to instances that have not been previously observed. It is clear that in order to do that the learner must have some prior knowledge about the set of possible (or likely) concepts. This knowledge is defined in terms of the *concept class*, which is the set of all a priori possible concepts.

In this paper we study concept learning in a probabilistic setting. Here the examples that are given to the learning algorithm are generated by choosing the instances at random from a distribution over the instance space. This distribution is arbitrary and unknown to the learner. The central measure of the quality of a learning algorithm in the probabilistic setting is the accuracy of the hypotheses that it generates. The accuracy of a hypothesis is the probability that it classifies a random instance correctly. The accuracy of the hypotheses that are generated by a learning algorithm is expected to improve as the resources available to the algorithm are increased. The main resources we consider are the number of examples used for learning and the time and space available to the learning algorithm. One of the main results of this paper is an upper bound on the resources required for learning in the distribution-free model of learnability introduced by Valiant [Val84].

In Valiant's model, commonly referred to as the PAC (probably approximately correct) learning model or the *distribution-free* learning model, the quality of a learning algorithm is defined as follows. A learner is said to have *accuracy* $1-\varepsilon$ with *reliability* $1-\delta$ if it generates a hypothesis whose accuracy is at least $1-\varepsilon$ with probability at least $1-\delta$. The probability that the algorithm fails is measured with respect to the random choice of the examples given to the learning algorithm and possible internal randomization of the algorithm.[1]

As was recognized by Haussler *et al.* [HKLW91], increasing the reliability of any learning algorithm is easy. This can be done by testing the hypothesis generated by the algorithm on an independent set of examples to validate its accuracy. If the accuracy is not sufficient, the algorithm is run again on a new set of random examples. It is easy to show that increasing the reliability from $1-\delta_1$ to $1-\delta_2$ can be achieved by running the algorithm $O(\log(1/\delta_2)/(1-\delta_1))$ times.[2]

Improving the accuracy of a learning algorithm is much harder. Two different variants of the PAC model were introduced by Kearns and Valiant [KV94] to address this

[1] The exact definition of the PAC learning model is given in Section 3.1.
[2] A full analysis of this algorithm is given in Appendix B.

0890-5401/95 $12.00

issue. In *strong* PAC learning, which is the more common model, the learner is given the required accuracy, ε, as input and is required to generate a hypothesis whose error is smaller than ε. The resources used by the algorithm can grow at most polynomially in $1/\varepsilon$. On the other hand, in *weak* PAC learning the accuracy of the hypothesis is required to be just slightly better than 1/2, which is the accuracy of a completely random guess. When learning with respect to a given distribution over the instances, weak and strong learning are not equivalent. Kearns and Valiant [KV94] proved that monotone boolean functions can be learned weakly, but not strongly, with respect to the uniform distribution.

This seemed to indicate that weak and strong distribution-free learning should also be separated. However, Schapire [Sch90] proved that weak and strong PAC learning are equivalent in the distribution-free case. Schapire presented an algorithm that, given access to a weak learning algorithm, can generate hypotheses of arbitrary accuracy using time and space resources that are polynomial in $1/\varepsilon$. This algorithm is called a "boosting" algorithm. The main idea is to run the weak learning algorithm several times, each time on a different distribution of instances, to generate several different hypotheses. We refer to these hypotheses as the "weak" hypotheses. These weak hypotheses are combined by the boosting algorithm into a single more complex and more accurate hypothesis. The different distributions are generated using an ingenious "filtering" process by which part of the random examples that are presented to the boosting algorithm are discarded and only a subset of the examples are passed on to the weak learning algorithm. It turns out that corollaries of this important result give good upper bounds on the time and space complexity of distribution-free learning. Schapire's result also has many important implications related to group learning, data compression, and approximation of hard functions.

In this article we present a simpler and more efficient boosting algorithm. Schapire's boosting algorithm is defined recursively. Each level of the recursion is a learning algorithm whose performance is better than the performance of the recursion level below it. The final hypothesis it generates can be represented as a circuit consisting of many three-input majority gates. The inputs to the circuit are the labels produced by the weak hypotheses, and the output is the final label (see Fig. 1). The depth of the circuit is a function of the problem parameters (accuracy and reliability), and its structure can vary between runs. The definition of our boosting algorithm, on the other hand, is not recursive and the final hypothesis can be represented as a single majority gate. This majority gate combines the outputs of all of the weak hypotheses.

In this paper we present two variants of our boosting algorithm. The first is *boosting by finding a consistent hypothesis*. This variant of the algorithm finds a hypothesis which is consistent with a large set of training examples. The analysis of this variant is quite straightforward, and its performance is close to the best performance we achieve. It also seems to be the variant whose application to practical learning problems is more efficient [Dru93]. The major drawback of this method is that it requires storage of the whole training set, which yields a space complexity dependence on ε of $O((\log 1/\varepsilon)^2/\varepsilon)$ (assuming that the concept class is fixed and that its VC dimension is finite). While this space requirement is often taken for granted, Schapire's algorithm demonstrates that boosting can be achieved using only poly-log$(1/\varepsilon)$ space.

We thus present a second variant of our algorithm, which we call *boosting by filtering*. This algorithm selects a small subset of the training examples as they are generated, and rejects all other examples. The sample complexity (number of training examples) of this version of the algorithm with respect to ε is $O((1/\varepsilon)(\log 1/\varepsilon)^{3/2}(\log\log 1/\varepsilon))$, its time complexity is $O((1/\varepsilon)(\log 1/\varepsilon)^{5/2}(\log\log 1/\varepsilon))$, its space complexity is $(\log 1/\varepsilon)(\log\log 1/\varepsilon))$ and the number of weak hypotheses it combines is $O(\log 1/\varepsilon)$.

The boosting by filtering algorithm is a completely general method for improving the accuracy of PAC learning algorithms. Its performance thus gives general upper

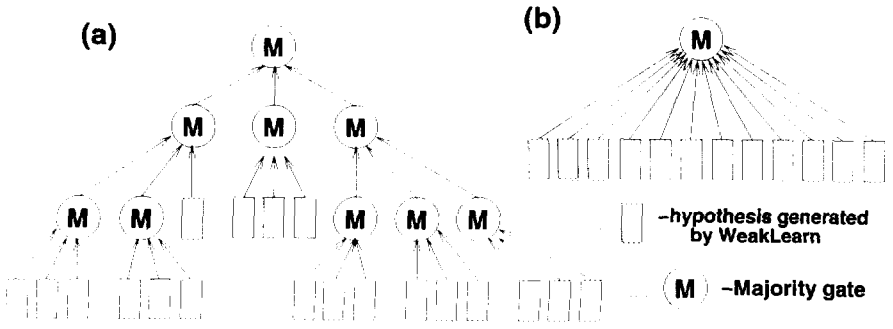

FIG. 1. Final concepts structure: (a) Schapire, (b) a one-layer majority circuit.

bounds on the dependence of the time and space complexity of efficient PAC learning on the desired accuracy. These bounds are, to the best of our knowledge, the best general upper bounds of this kind that are known today. We present some lower bounds that show that the possibilities for additional improvement are very limited. In particular, we show that there cannot be a general boosting algorithm that combines a smaller number of weak hypotheses to achieve the same final accuracy.

We also present generalizations of the algorithm to learning concepts whose output is not binary. One generalization is for concepts with k-valued outputs and is quite straightforward. Another generalization is to real-valued concepts. We show how boosting can be used in this case to transform a learning algorithm that generates functions whose expected error over the domain is bounded by c into a learning algorithm that generates functions whose error is bounded by $2c$ over most of the domain. Intuitively, this gives a method for "spreading" the error of algorithms for learning real valued functions evenly over the domain.

We also extend our result to distribution-specific learning. We show that our algorithm can be used for boosting the performance of learning algorithms whose accuracy depends on the distribution of the instances. Suppose we have a learning algorithm A, which achieves accuracy $1-\varepsilon$ with respect to the distribution $\mathscr{D}$. If A achieves accuracy $1-\varepsilon$ for *every* distribution of the examples then boosting can be used to achieve arbitrary accuracy with respect to $\mathscr{D}$. However, in real world problems, A's accuracy usually depends on the distribution of the examples. We show that if A's accuracy degradation is not too abrupt, then boosting can still be used to generate a hypothesis whose accuracy is better than $1-\varepsilon$.

Schapire [Sch92] noted that the results presented in this paper can be used to show an interesting relationship between representation and approximation using majority gates. These results were independently discovered by Goldmann *et al.* [GHR92]. However, while their proof technique is very elegant, our proof is more constructive (for details see Section 2.2).

It is surprising to note that the boosting algorithm uses only a small fraction of the examples in the training set. While it needs $\Omega(1/\varepsilon)$ examples to generate a hypothesis that has accuracy ε, only $O(\log 1/\varepsilon)$ of them are passed to the weak learners. Two interesting implications arise from this fact. The first implication was pointed out to us by Schapire [Sch92]. It can be shown that if a concept class is learnable then the following type of compression can be achieved: Given a sample of size m, labeled according to some concept in the class, the boosting algorithm can be used to find a subsample of size $O(\log m)$ such that the labeling of all of the instances in the sample can be reconstructed from the labels of the subsample. This strengthens the relationship between compression and learning which has been studied by Floyd and Warmuth [FW93].

The second implication was found together with Shamir [Sha92]. We observed that if training examples can be accumulated in parallel by several parallel processors, then our methods can translate any PAC learning algorithm to a version that runs in time $O(\log 1/\varepsilon)$ on a parallel computer with $\Theta(1/\varepsilon)$ processors. This is because most of the examples that are given to the boosting algorithm are simply discarded and the search for a "good" example can be done by many processors in parallel.

The paper is organized as follows. The main theorem on which our boosting algorithms are based is given in Section 2 using a simple game-theoretic setting that avoids some of the complications of the learning problem while addressing the main underlying problem. In Section 3 we relate the theorem back to the learning problem, in Section 4 we present some extensions, and in Section 5 we summarize and present some open problems.

In Section 2 we present a game, called the "majority-vote" game, between two players, a "weightor" and a "chooser." The game consists of k iterations. For simplicity we now assume that the game is played on the set $\{1, ..., N\}$. In each iteration the weightor assigns to the N points non-negative weights that sum to 1. The chooser has to then "mark" a subset of the points whose weights sum to at least $1/2+\gamma$, where $0<\gamma\leqslant 1/2$ is a fixed parameter of the game. The goal of the weightor is to force the chooser to mark each point in the space in a majority of the iterations, i.e., each point has to receive more than $k/2$ marks. We show that there exists a strategy that lets the weightor achieve that goal in $\lceil \frac{1}{2}\gamma^{-2}\ln N\rceil$ iterations. A similar game can be played on a general probability space, in which case the goal of the weightor is to force the chooser to mark all but an ε fraction of the space in the majority of the iterations. We show that $k=\lceil \frac{1}{2}\gamma^{-2}\ln 1/\varepsilon\rceil$ iterations suffice in this case.

The weightor in this game represents the centerpiece of the boosting algorithm, which is the choice of the distributions that are presented to the weak learning algorithm. The points that the chooser decides to mark correspond to the instances on which the weak learner makes the correct prediction. This represents the freedom of the weak learner to distribute the error of the hypothesis in any way it chooses as long as the probability that a random instance is labeled correctly is at least $1/2+\gamma$. This abstraction bypasses some of the complexities of the PAC learning problem, and can be read independently of the rest of the paper. In Subsection 2.1 we show that in the case of continuous probability spaces, there is a strategy for the chooser such that for any strategy of the weightor, if the game is stopped in less than $k=\lceil \frac{1}{2}\gamma^{-2}\ln 1/\varepsilon\rceil$ iterations, more than ε of the space is marked less than $k/2$ times, i.e. the weightor fails to achieve its goal. Thus our weighting strategy is optimal for the case of continuous probability

spaces. In Subsection 2.2 we present the implication of our analysis of the majority-vote game on the representational power of threshold circuits.

In Section 3 we relate the majority-vote game to the problem of boosting a weak learner and present the two variants of the boosting algorithm and their performance bounds. In order to simplify our analysis we restrict our analysis in Subsections 3.2 and 3.3 to the case in which the weak learning algorithms are deterministic algorithms that generate deterministic hypotheses. In Subsection 3.4 we show that this analysis needs to be changed only slightly to accommodate randomized learning algorithms that generate randomized hypotheses. In order to present the complete dependence of our bounds on the parameters of the problem, we do not use the notational conventions of polynomial PAC learning in our main presentation, but rather give explicit bounds including constants. Later, in Subsection 3.5, we derive upper bounds on the resources required for polynomial PAC learning that are the best general upper bounds of this type that exist to date. In Subsection 3.6 we compare our upper bounds to known lower bounds and discuss which aspects of our bounds are optimal and which might be further improved.

In Section 4 we give several extensions and implications of our main results. In Subsection 4.1 we show that our algorithm for boosting by filtering can work even in situations where the error of the hypotheses generated by the weak learning algorithm is not uniformly bounded for all distributions. In Subsection 4.2 we present a version of the boosting algorithm that works for concepts whose range is a finite set, and in Subsection 4.3 we present a version that works for concepts whose range is real valued. In Subsection 4.4 we show how boosting can be used to parallelize learning algorithms. We conclude the paper with a summary and a list of open problems in Section 5. In the appendixes to the paper we give a summary of our notation and proofs of three lemmas. The meaning of some of the notation used in this paper has slightly different interpretations in different parts of the paper. The table in Appendix A.3 summarizes the different interpretations and might be a useful reference when reading the paper.

2. THE MAJORITY-VOTE GAME

In this section we define a two-player, complete information, zero-sum game. The players are the "weightor," D, and the "chooser," C. The game is played over a probability space $\langle X, \Sigma, V \rangle$, where X is the sample space, Σ is a σ-algebra over X, and V is a probability measure. We shall refer to the probability of a set $A \in \Sigma$ as the *value* of the set and denote it by $V(A)$. A real-valued parameter $0 < \gamma \leq 1/2$ is fixed before the game starts.

The game proceeds in iterations; in each iteration we have the following steps:

1. The weightor picks a *weight* measure on X. The weight measure is a probability measure on $\langle X, \Sigma \rangle$. We denote the weight of a set A by $W(A)$.

2. The chooser selects a set $U \in \Sigma$ such that $W(U) \geq \frac{1}{2} + \gamma$, and *marks* the points of this set.

These two-step iterations are repeated until the weightor decides to stop. It then receives, as its payoff, the subset of X that includes those points of X that have been marked in more than half of the iterations played (if the number of iterations is even this set *does not* include points that have been marked exactly half the time). We shall refer to this set as the *reward* set and to its value as the *reward*. The complement of the reward set is the *loss* set. The goal of the weightor is to maximize the reward, and the goal of the chooser is to minimize it.

The question about this game in which we are interested is whether there exists a general strategy independent of the specific probability space that guarantees the weightor a large reward. An affirmative answer to this question is given in this section. We describe a general strategy for the weightor such that for any probability space $\langle X, \Sigma, V \rangle$ and any $\varepsilon, \gamma > 0$, the weightor can guarantee that the reward is larger than $1 - \varepsilon$ after at most $\frac{1}{2}(1/\gamma)^2 \ln(1/\varepsilon)$ iterations.

We shall present the weighting strategy in the following way. We start by giving some insight and show what weighting strategies are reasonable. We then present the weighting strategy and prove a bound on the reward that it guarantees. Finally, we show that for non-singular sample spaces (such as a density distribution on R^n) there is a matching strategy for the adversary, implying that our strategy is the optimal minimax strategy when the sample space is non-singular.

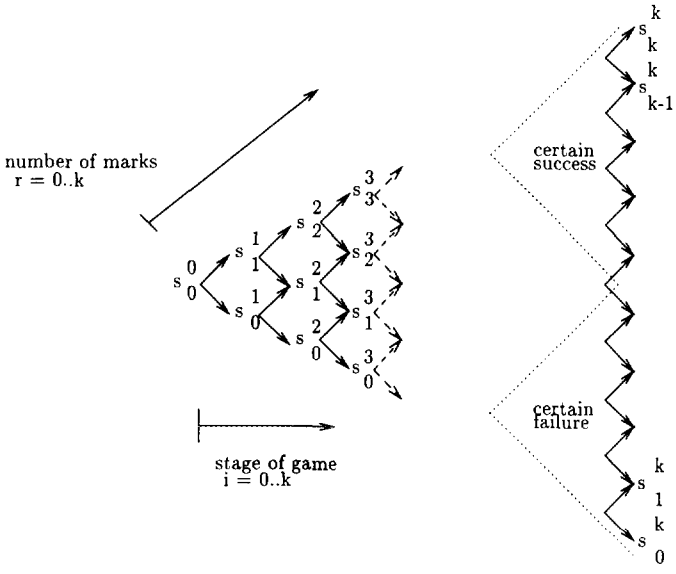

FIG. 2. Transitions between consecutive partitions.

In the following discussion we are fixing a particular instance of the game, i.e. we consider a particular sequence of moves taken by the two players. Let k be the number of iterations in the game. For $0 \leqslant i \leqslant k$ define $\{X_0^i, X_1^i, ..., X_i^i\}$ to be a partition of X into $i+1$ sets where X_r^i consists of those points in X that have been marked r times after i turns of the game have been played.

As graphically presented in Fig. 2, in iteration i the chooser decides for each point in X_r^i whether to mark it or not, thus placing it in X_{r+1}^{i+1} or in X_r^{i+1}. The goal of the chooser is to minimize the value of the reward set: $\bigcup_{r=\lfloor k/2 \rfloor+1}^{k} X_r^k$. The goal of the weightor is to maximize this value. By giving some points more weight than others, the weightor forces the chooser to mark more of those points. In the extreme, by placing all the weight on a single point it guarantees that this point will be marked while at the same time allowing the chooser not to mark any other point, moving them closer to the loss set.

Let us define some notation (a complete notation table appears in Appendix A.3):

k	the total number of iterations the game is played
X_r^i	the set of points that have been marked r times in the first i iterations
$M_r^i = X_r^i \cap X_{r+1}^{i+1}$	the subset of X_r^i that is marked in iteration i
$q_r^i = V(X_r^i)$	the value of X_r^i
$x_r^i = V(M_r^i)/V(X_r^i)$	the fraction of X_r^i that is marked in iteration i
L	the loss set, i.e., those points that are in the end marked less than or equal to half the time.

Note that $X_0^0 = X$ and thus $q_0^0 = 1$.

Observe that if $r > k/2$, then points in X_r^i are guaranteed to be in the reward set. Likewise, if $i - r \geqslant k/2$ then points in X_r^i are guaranteed to be in the loss set. Thus it is intuitively clear that any reasonable weighting scheme will give zero weight to these points and place all the weight on those points for which both failure and success are still possible. In particular, the only points that should be assigned a non-zero weight in the final iteration are points in $X_{\lfloor k/2 \rfloor}^{k-1}$. We now present a weighting strategy that agrees with this intuition and prove that this strategy guarantees the claimed performance.

The weighting strategy assigns a *weighting factor* α_r^i to each set X_r^i where $0 \leqslant r \leqslant i \leqslant k-1$. If the space is discrete then the weight assigned to the point $x \in X_r^i$ on round i is the value of the point times α_r^i times a constant normalization factor that makes the total weight be one (the definition of the weighting for non-discrete spaces is given in the statement of Theorem 2.1).

The weighting factor is defined inductively as

$$\alpha_r^{k-1} = \begin{cases} 1 & \text{if } r = \left\lfloor \frac{k}{2} \right\rfloor \\ 0 & \text{otherwise.} \end{cases}$$

and for $0 \leqslant i \leqslant k-2$,

$$\alpha_r^i = (\tfrac{1}{2} - \gamma)\,\alpha_r^{i+1} + (\tfrac{1}{2} + \gamma)\,\alpha_{r+1}^{i+1}.$$

Recall that γ is a parameter of the majority-vote game that is fixed before the game starts. Clearly there is only one function of i, k, r, and γ that satisfies this inductive definition. It can be verified that this function is the following binomial distribution.

$$\alpha_r^i = \binom{k-i-1}{\lfloor k/2 \rfloor - r} \left(\frac{1}{2}+\gamma\right)^{\lfloor k/2 \rfloor - r} \left(\frac{1}{2}-\gamma\right)^{\lceil k/2 \rceil - i - 1 + r}. \tag{1}$$

Here, and throughout the paper, we define $\binom{n}{m} = 0$ if $m < 0$ or $m > n$ and $\binom{0}{0} = 1$. The performance of our weighting strategy is given in the following theorem:

THEOREM 2.1. *For any probability space $\langle X, \Sigma, V \rangle$ and any $\varepsilon, \gamma > 0$, if the weightor plays the majority-vote game for k iterations, where k satisfies*

$$\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{k}{j} \left(\frac{1}{2}+\gamma\right)^{j} \left(\frac{1}{2}-\gamma\right)^{k-j} \leqslant \varepsilon, \tag{2}$$

and uses the following weighting[3] in iteration i:

For any set A in the σ-algebra Σ,

$$W(A) = \sum_{r=0}^{i} V(A \cap X_r^i)\,\alpha_r^i / Z_i, \tag{3}$$

where

$$Z_i = \sum_{r=0}^{i} V(X_r^i)\,\alpha_r^i,$$

then the reward at the end of the game is at least $1-\varepsilon$, independent of the strategy used by the chooser.

Before proving the theorem, we define the function β_r^i over $0 \leqslant r \leqslant i \leqslant k$ which we call the "potential" of the set X_r^i. As we shall see the potential of X_r^i predicts, in some sense, the fraction of points in X_r^i that will end up in the loss set. As at the end of the game we know which points are in the

[3] In the special case where X is discrete, it is sufficient to define the weight of each point. In this case we set $W(x) = \alpha_r^i V(x)$ for all $x \in X_r^i$.

loss set and which are in the reward set; it is reasonable to define the potential for $i=k$ as

$$\beta_r^k = \begin{cases} 0 & \text{if } r > \frac{k}{2} \\ 1 & \text{if } r \leq \frac{k}{2}. \end{cases} \tag{4}$$

For $i<k$ we define the potential recursively:

$$\beta_r^i = (\tfrac{1}{2}-\gamma)\,\beta_r^{i+1} + (\tfrac{1}{2}+\gamma)\,\beta_{r+1}^{i+1}. \tag{5}$$

It can be easily verified that a closed form formula for β_r^i is given by the tail of the binomial distribution:

$$\beta_r^i = \sum_{j=0}^{\lfloor k/2 \rfloor - r} \binom{k-i}{j} \left(\frac{1}{2}+\gamma\right)^j \left(\frac{1}{2}-\gamma\right)^{k-i-j}. \tag{6}$$

The weight factor function, α_r^i, is in some sense a discrete derivative of the potential function along the r axis:

$$\alpha_r^i = \beta_r^{i+1} - \beta_{r+1}^{i+1}. \tag{7}$$

The main property of the weighting scheme is that it guarantees that the average potential does not increase at any step. This property is proved in the following lemma.

LEMMA 2.2. *If the weighting scheme described in Eq.* (3) *is used by the weightor, then*

$$\beta_0^0 \geq \sum_{r=0}^{1} q_r^1 \beta_r^1 \geq \sum_{r=0}^{2} q_r^2 \beta_r^2 \geq \cdots \geq \sum_{r=0}^{k} q_r^k \beta_r^k,$$

for any strategy of the chooser.

Proof of Lemma 2.2. Recall that $q_r^i = V(X_r^i)$ and $x_r^i = V(M_r^i)/V(X_r^i)$. At each iteration i the adversary chooses the variables $0 \leq x_r^i \leq 1$, and we get the following formula for the transition to the next iteration:

$$\begin{aligned} q_r^{i+1} &= q_{r-1}^i x_{r-1}^i + q_r^i(1-x_r^i) && \text{for } 1 \leq r \leq i, \\ q_0^{i+1} &= q_0^i(1-x_0^i) && \text{for } r=0, \\ q_{i+1}^{i+1} &= q_i^i x_i^i && \text{for } r=i+1. \end{aligned} \tag{8}$$

Using this we can get a formula that relates the sum $\sum_{r=0}^{i} q_r^i \beta_r^i$ for consecutive iterations,

$$\begin{aligned} \sum_{r=0}^{i+1} q_r^{i+1}\beta_r^{i+1} &= q_0^i(1-x_0^i)\,\beta_0^{i+1} \\ &\quad + \sum_{r=1}^{i} [q_{r-1}^i x_{r-1}^i + q_r^i(1-x_r^i)]\,\beta_r^{i+1} \\ &\quad + q_i^i x_i^i \beta_{i+1}^{i+1}. \end{aligned}$$

Rearranging the sum gives us that

$$\begin{aligned} \sum_{r=0}^{i+1} q_r^{i+1}\beta_r^{i+1} &= \sum_{r=0}^{i} q_r^i[(1-x_r^i)\,\beta_r^{i+1} + x_r^i \beta_{r+1}^{i+1}] \\ &= \sum_{r=0}^{i} q_r^i \beta_r^{i+1} + \sum_{r=0}^{i} q_r^i x_r^i(\beta_{r+1}^{i+1} - \beta_r^{i+1}). \end{aligned} \tag{9}$$

On the other hand, from the weight restriction we obtain

$$\sum_{r=0}^{i} W(M_r^i) \geq \tfrac{1}{2}+\gamma, \tag{10}$$

and as $M_r^i \subseteq X_r^i$ the definition of the weight function gives

$$\frac{1}{Z_i}\sum_{r=0}^{i} V(M_r^i)\,\alpha_r^i \geq \frac{1}{2}+\gamma. \tag{11}$$

Using the definitions of q_r^i, x_r^i, and Z_i this can be written as

$$\frac{\sum_{r=0}^{i} q_r^i x_r^i \alpha_r^i}{\sum_{r=0}^{i} q_r^i \alpha_r^i} \geq \frac{1}{2}+\gamma. \tag{12}$$

Using Eq. (7) and the fact that $\beta_r^{i+1} > \beta_{r+1}^{i+1}$ we obtain that

$$\sum_{r=0}^{i} q_r^i x_r^i(\beta_{r+1}^{i+1} - \beta_r^{i+1}) \leq (\tfrac{1}{2}+\gamma) \sum_{r=0}^{i} q_r^i(\beta_{r+1}^{i+1} - \beta_r^{i+1}). \tag{13}$$

Substituting (13) into the right-hand side of (9) we finally obtain that

$$\begin{aligned} \sum_{r=0}^{i+1} q_r^{i+1}\beta_r^{i+1} &\leq \sum_{r=0}^{i} q_r^i \beta_r^{i+1} + (\tfrac{1}{2}+\gamma) \sum_{r=0}^{i} q_r^i(\beta_{r+1}^{i+1} - \beta_r^{i+1}) \\ &= \sum_{r=0}^{i} q_r^i((\tfrac{1}{2}+\gamma)\,\beta_{r+1}^{i+1} + (\tfrac{1}{2}-\gamma)\,\beta_r^{i+1}) \\ &= \sum_{r=0}^{i} q_r^i \beta_r^i. \end{aligned}$$

The last equality is based on Eq. (5). ∎

Proof of Theorem 2.1. From Eq. (6) it is immediate that the left hand side of Eq. (2) is equal to β_0^0, thus, by choice of k, $\beta_0^0 \leq \varepsilon$, which means that the initial expected potential is small. Combining this with the inequality from Lemma 2.2 that implies that the potential never increases, we get that

$$\varepsilon \geq \beta_0^0 \geq \sum_{r=0}^{k} \beta_r^k q_r^k.$$

On the other hand, from Eq. (4) we have

$$\sum_{r=0}^{k} \beta_r^k q_r^k = \sum_{r=0}^{\lfloor k/2 \rfloor} q_r^k = V(L).$$

Thus the value of the loss set L is at most ε. ∎

In order to see that the result given in Theorem 2.1 is meaningful, we give an explicit choice for k that is close to the optimal choice for small ε and γ.

COROLLARY 2.3. *Theorem* 2.1 *holds if the number of iterations satisfies*

$$k \geqslant \frac{1}{2\gamma^2} \ln \frac{1}{2\varepsilon}.$$

Proof. Suppose that a biased coin, whose probability for heads is $1/2+\gamma$, is tossed k times. The left-hand side of Inequality (2) describes the probability that at most half of the coin tosses come up heads. We can upper bound the probability of this event as follows. The number of sequences of k coin tosses that have at least as many heads as tails is at most 2^{k-1}. The probability of any single sequence is at most $[(1/2-\gamma)(1/2+\gamma)]^{k/2}$. Thus we get that a sufficient condition on k for Inequality (2) to hold is

$$2^{k-1}[(1/2-\gamma)(1/2+\gamma)]^{k/2} \leqslant \varepsilon.$$

Reordering this inequality we get that it is equivalent to

$$k/2 \geqslant \frac{\ln(1/2\varepsilon)}{-\ln(1-4\gamma^2)}.$$

Using the fact that $\ln(1-x) \leqslant -x$ for $0 \leqslant x < 1$ we find that the condition on k in the statement of the corollary is a sufficient condition condition for Inequality (2). ∎

2.1. *Optimality of the Weighting Scheme*

We shall now show that, in some natural cases, the weighting strategy devised in this section is optimal in that it guarantees the weightor the minimal possible loss which is achievable in k iterations. We show this by giving a strategy for the chooser which guarantees a loss which is at least as large as the loss which our weighting strategy suffers. The idea of the choosers strategy is to mark points so that the values of the marked sets are equal to the expected value if the points where marked independently at random with probability $1/2+\gamma$.

We now define a property of the probability space $\langle X, \Sigma, V \rangle$ which is sufficient for showing optimality of our weighting scheme. We say that $\langle X, \Sigma, V \rangle$ is *divisible* if for any measurable set $A \in \Sigma$ there exists another set $A' \in \Sigma$ such that $V(A') = \frac{1}{2} V(A)$. One natural example of a divisible space is the Euclidean space $X = R^n$, where Σ is the Borel algebra over R^n and the measure V is a density measure that assigns all single points a value of zero.

The weighting strategy defined in this section is optimal for divisible probability spaces, as is summarized in the following theorem.

THEOREM 2.4. *If the probability space $\langle X, \Sigma, V \rangle$ is divisible, then there exists a strategy for the chooser such that for any k and γ, the loss is at least*

$$\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{k}{j} \left(\frac{1}{2}+\gamma\right)^j \left(\frac{1}{2}-\gamma\right)^{k-j}. \tag{14}$$

This means that the choice of k in Eq. (2) of Theorem 2.1 is the smallest possible.

In order to prove the theorem, we need the following technical lemma, whose proof is given in the appendix.

LEMMA 2.5. *Suppose that the probability space $\langle X, \Sigma, V \rangle$ is divisible, and that W is another probability measure defined on $\langle X, \Sigma \rangle$.*

Then for any set $A \in \Sigma$ there exists a set $A' \subset A$, $A' \in \Sigma$ such that $V(A') = (1/2+\gamma)\, V(A)$ and $W(A') \geqslant (1/2+\gamma)\, W(A)$.

Proof of the Theorem. Let us denote the set of points in X that have been marked in a particular way by $X_{\mathbf{s}}$, where $s = \langle s_1, s_2, \ldots, s_i \rangle$ is a binary vector of length $i \leqslant k$ such that s_j is 1 if x has been marked on iteration j and 0 otherwise. The goal of the chooser is to mark points so that for each sequence of marks $\mathbf{s}$, the probability of $X_{\mathbf{s}}$ would be equal to the probability that s is generated by i random draws of a biased coin whose probability of heads is $1/2+\gamma$.

The strategy is defined as follows. In the ith iteration, the space X is divided into the 2^{i-1} sets $X_{\mathbf{s}}$ corresponding to the different binary sequences of length $i-1$. From Lemma 2.5 we get that for each $\mathbf{s}$ there exists a set $A_{\mathbf{s}} \subset X_{\mathbf{s}}$ in Σ such that $V(A_{\mathbf{s}}) = (1/2+\gamma)\, V(X_{\mathbf{s}})$ and $W(A_{\mathbf{s}}) \geqslant (1/2+\gamma)\, W(X_{\mathbf{s}})$. The chooser marks the points in the set $\bigcup_{\mathbf{s}} A_{\mathbf{s}}$ which makes $X_{\mathbf{s}1}$ be $A_{\mathbf{s}}$ and $X_{\mathbf{s}0}$ be $X_{\mathbf{s}} \cap \overline{A_{\mathbf{s}}}$. This is a legal marking because of the condition on $W(A_{\mathbf{s}})$.

The loss set is

$$\bigcup \left\{ X_{\mathbf{s}} \;\middle|\; \mathbf{s} = \langle s_1, \ldots, s_k \rangle, \sum_{i=1}^{k} s_i \leqslant \frac{k}{2} \right\}.$$

It is easy to calculate the value of the loss set and show that it is equal to the loss defined in (14), which completes the proof. ∎

It is easy to construct cases of non-divisible spaces. In particular, any finite space is not divisible. Thus, it is very likely that some improvement to the results given in this paper are possible for finite sample spaces. However, it

seems that the possibility for improvement decreases rapidly as the size of the sample space increases.

2.2. *The Representational Power of Majority Gates*

Our analysis of the majority-vote game can be used to prove an interesting result regarding the representation of Boolean functions as a majority over other Boolean functions. This application of boosting has been discovered by Schapire [Sch92]. A slightly weaker version of this result was independently proven by Goldmann *et al.* [GHR92] using a completely different proof technique. In the following presentation we follow their notation.

Let f denote a Boolean function whose domain is $\{-1, 1\}^n$ and range is $\{-1, 1\}$. Let H be a set of Boolean functions defined over the same domain and range. Intuitively, the result is that if, for any distribution over the domain $\{-1, 1\}^n$, there is some function $h \in H$ such that f and h are correlated, then f can be represented as a majority over a small number of functions in H.

In order to give the formal statement we define the following notation. We use $\mathscr{D}$ to denote a distribution over the domain $\{-1, 1\}^n$ and define the correlation between f and H with respect to $\mathscr{D}$ as

$$D_H^{\mathscr{D}}(f) \doteq \max_{h \in H} E_{\mathscr{D}}[f(x)\, h(x)].$$

The distribution-free correlation between f and H is defined as

$$D_H(f) \doteq \min_{\mathscr{D}} D_H^{\mathscr{D}}(f).$$

The majority function is defined as

$$\mathrm{MAJ}(x_1, \ldots, x_k) = \mathrm{sign}\left(\sum_{i=1}^{k} x_i\right),$$

where

$$\mathrm{sign}(x) = \begin{cases} 1, & \text{if } x \geqslant 0 \\ -1, & \text{otherwise.} \end{cases}$$

Using our boosting algorithm we prove the following result:

THEOREM 2.6. *Let f be a Boolean function over $\{-1, 1\}^n$ and H be a set of functions over the same domain. Then if $k > 2\ln(2)\, n D_H^{-2}(f)$, then f can be represented as*

$$f(x) = \mathrm{MAJ}(h_1(x), \ldots, h_k(x)),$$

for some $h_i \in H$.

Proof. Assume the majority-vote game is played over the domain $\{-1, 1\}^n$ and that the value of a set is the number of points in it divided by 2^n. Assume the chooser in the majority-vote game chooses which points to mark by selecting a function $h \in H$ such that $\Pr_{\mathscr{D}}(h(x) \neq f(x)) \leqslant 1/2 - \gamma$ and marking all x such that $h(x) = f(x)$. By definition of $D_H(f)$, such a function exists for every distribution $\mathscr{D}$ if $\gamma = D_H(f)/2$. Theorem 2.1 provides us with a method for selecting the distributions $\mathscr{D}_i$, which correspond to the the weightings W_i. This selection guarantees that the majority over the corresponding hypotheses will be very close to f. More specifically, it guarantees that if $k = 1/2\gamma^{-2} \ln 1/\varepsilon$, the number of points $x \in \{-1, 1\}^n$ such that $\mathrm{MAJ}(h_1(x), \ldots, h_k(x)) \neq f(x)$ is smaller than $\varepsilon 2^n$; by setting $\varepsilon < 2^{-n}$ we guarantee that $\mathrm{MAJ}(h_1(x), \ldots, h_k(x)) = f(x)$ for all $x \in \{-1, 1\}^n$. Plugging our selection for γ and ε into $k = 1/2\gamma^{-2} \ln \varepsilon$ we finish the proof. ∎

Goldmann *et al.* [GHR92], prove Theorem 2.6 using an elegant application of von Neumann's Min-Max Theorem. However, their proof does not show how one can find the functions $h_i \in H$. On the other hand, our proof is constructive in that it shows how to generate the distributions that correspond to the desired functions.

For completeness we give a simple lemma (Lemma 4 in [GHR92]) that gives an approximate converse to Theorem 2.6.

LEMMA 2.7. *Let f and H be as in Theorem 2.6. Then if f can be represented as*

$$f(x) = \mathrm{MAJ}(h_1(x), \ldots, h_k(x)),$$

where $h_i \in H$ and k is odd, then $D_H(f) \geqslant 1/k$.

Proof. From the definition of the majority function and the fact that k is odd, we get that for every $x \in \{-1, 1\}^n$ there are at least $(k+1)/2$ indices i such that $h_i(x) = f(x)$. Fixing any distribution $\mathscr{D}$ over $\{-1, 1\}^n$, we get that

$$\sum_{i=1}^{k} \Pr_{\mathscr{D}}(h_i(x) = f(x))$$

$$= \sum_{x \in \{-1, 1\}^n} \Pr_{\mathscr{D}}(x: |\{1 \leqslant i \leqslant k : h_i(x) = f(x)\}| \geqslant (k+1)/2).$$

The pigeon-hole principle guarantees that there exists at least one index $1 \leqslant i \leqslant k$ such that $\Pr_{\mathscr{D}}(h_i(x) = f(x)) \geqslant (k+1)/2$. This implies that $D_H^{\mathscr{D}}(f) \geqslant 1/k$. As this holds for all $\mathscr{D}$, we get the statement of the lemma. ∎

3. BOOSTING A WEAK LEARNER USING A MAJORITY VOTE

In this section we shall describe the connection between the majority-vote game and the problem of boosting a weak learning algorithm.

We start by presenting a minimal formal framework for analyzing our boosting algorithms. We then present our algorithms and their analysis. Later, in Section 3.5, we give a more complete notational framework and use this framework to relate our results to other results in PAC learning theory.

3.1. *Preliminaries*

We start by giving the definitions of a minimal framework of distribution-free concept learning that is needed for presenting our main results. A *concept* is a binary-valued mapping over some domain X. We use the letter c to denote a concept and $c(x)$ to denote the label of the instance x according to the concept c. A *concept class* $\mathbf{C}$ is a collection of concepts.[4]

The learner's task is to learn an approximation to a concept c. The learner knows *a-priori* that the concept is in some known class $\mathbf{C}$, but has no prior knowledge of the specific choice of $c \in \mathbf{C}$. The learner is assumed to have access to a source **EX** of examples. Each time **EX** is called, one instance is randomly and independently chosen from X according to some fixed but unknown and arbitrary distribution $\mathscr{D}$.[5] The oracle returns the chosen instance $x \in X$, along with its label according to the concept c, which is denoted $c(x)$. Such a labeled instance is called an *example*. We assume **EX** runs in unit time.

Given access to **EX** the learning algorithm runs for some time and finally outputs a *hypothesis* h. The hypothesis is a description of an algorithm (possibly probabilistic) that receives as input an instance $x \in X$ and generates a binary output. This output is called the "prediction" of the hypothesis for the label $c(x)$. We write $P(h(x) = c(x))$ to indicate the probability, over the distribution $\mathscr{D}$ on X and random coin flips of the hypothesis, that the hypothesis correctly predicts the labels of the concept c. This probability is called the *accuracy* of the hypothesis h. The probability $P(h(x) \neq c(x))$ is called the *error* of h with respect to c under $\mathscr{D}$; if the error is no more than ε, then we say h is ε-good with respect to the target concept c and the distribution $\mathscr{D}$.

[4] In order to define *polynomial* PAC learnability, the complexity of the sample space and that of the concept class need to be parameterized. In our *initial basic setting we suppress this parameterization and the issue of polynomial* versus nonpolynomial learning; we return to fully discuss this issue in Section 3.5.

[5] More formally, we assume that $\langle X, \Sigma, \mathscr{D} \rangle$ is a probability space and that $\mathbf{C}$ is a set of functions that are measurable with respect to Σ. Moreover, we assume that all subsets of X that are considered in this paper are measurable with respect to Σ.

We say that a learning algorithm A has a *uniform* sample complexity $m(\varepsilon, \delta)$ if it achieves the following performance. For all $0 < \varepsilon, \delta < 1$, all $\mathscr{D}$, and all $c \in \mathbf{C}$, after receiving the parameters ε and δ as inputs, algorithm A makes at most $m(\varepsilon, \delta)$ calls to **EX** and outputs a hypothesis h that with probability at least $1 - \delta$ is an ε-good approximation of c under $\mathscr{D}$. Similarly we define the time and space complexity of A to be functions that bound the time and space required by A and denote them by $t(\varepsilon, \delta)$ and $s(\varepsilon, \delta)$ respectively. If a learning algorithm cannot achieve some values of ε and δ, or if the resources required for achieving these values are not uniformly bounded for all distributions and concepts, we define $m(\varepsilon, \delta)$, $t(\varepsilon, \delta)$, and $s(\varepsilon, \delta)$ to be infinite for these values.

The concept of a *boosting* algorithm was first presented by Schapire in [Sch90]. A boosting algorithm is a learning algorithm that uses as a subroutine a different learning algorithm. The goal of the boosting algorithm is to efficiently generate high-accuracy hypotheses using a learning algorithm that can efficiently generate only low-accuracy hypotheses. The boosting algorithm invented by Schapire [Sch90] was a breakthrough in that it showed that any polynomial time learning algorithm that generates hypotheses whose error is just slightly smaller than $1/2$ can be transformed into a polynomial time learning algorithm that generates hypotheses whose error is arbitrarily small. The boosting algorithms presented in this paper achieve better performance than those presented by Schapire and the resulting hypotheses are simpler. A comparison of the performance of the algorithms is given in Section 3.5.

We use the generic name **WeakLearn** to refer to the learning algorithm whose performance we wish to boost, and we refer to those hypotheses generated by **WeakLearn** that have the guaranteed accuracy as *weak* hypotheses. We assume that there exist some real values $0 \leqslant \varepsilon_0 < 1/2$ and $0 \leqslant \delta_0 < 1$ such that **WeakLearn**, given m_0 examples labeled according to some concept $c \in \mathbf{C}$, generates a hypothesis whose error is at most ε_0 (i.e., a weak hypothesis) with probability at least $1 - \delta_0$ over the distribution of the training examples. We denote by m_0, t_0, and s_0 uniform upper bounds on the sample size, time, and space required by **WeakLearn** to achieve this accuracy. The boosting algorithms that we shall describe are able to generate hypotheses of arbitrary accuracy ε with arbitrarily high reliability $1 - \delta$.

The parameters ε_0 and δ_0 measure the discrepancy between the performance of **WeakLearn** and the performance of an "ideal" learning algorithm that always generates a hypothesis that has no error with respect to the target concept. The performance of the weak learning algorithms that we discuss is extremely poor. They are almost completely unreliable, and even when they succeed they output a hypothesis whose error is close to that of a random guess. We thus find it useful to define two new quantities $\gamma = 1/2 - \varepsilon_0$ and $\lambda = 1 - \delta_0$. These parameters measure how

Algorithm $\mathbf{B}_{\text{Samp}}$
Input: **EX**, **WeakLearn**, γ, m
Output: A hypothesis that is consistent on a random sample of size m.

1. Call **EX** m times to generate a sample $S = \{(x_1, l_1), \ldots, (x_m, l_m)\}$.
   To each example (x_j, l_j) in S corresponds a weight w_j and a count r_j.
   Initially, all weights are $1/m$ and all counts are zero.
2. Find a (small) k that satisfies

$$\sum_{i=\lceil k/2 \rceil}^{k} \binom{k}{i} (1/2 - \gamma)^i (1/2 + \gamma)^{k-i} < \frac{1}{m}$$

   (For example, any $k > 1/(2\gamma^2) \ln(m/2)$ is sufficient.)
3. Repeat the following steps for $i = 1 \ldots k$.
   (a) repeat the following steps for $l = 1 \ldots (1/\lambda) \ln(2k/\delta)$
   or until a weak hypothesis is found.
      i. Call **WeakLearn**, referring it to **FiltEX** as its source of examples,
      and save the returned hypothesis as h_i.
      ii. Sum the weights of the examples on which $h_i(x_j) \neq l_j$.
      If the sum is smaller than $1/2 - \gamma$
      then declare h_i a weak hypothesis and exit the loop.
   (b) Increment r_j by one for each example on which $h_i(x_j) = l_j$.
   (c) Update the weights of the examples according to $w_j = \alpha_{r_j}^1$,
   α_r^1 is defined in Equation (1).
   (d) Normalize the weights by dividing each weight by $\sum_{j=1}^{m} w_j$.
4. Return as the final hypothesis, h_M, the majority vote over $h_1, \ldots, h_k$.

Subroutine FiltEX

1. choose a real number x uniformly at random in the range $0 \leq x < 1$.
2. Perform a binary search for the index j for which

$$\sum_{i=1}^{j-1} w_i \leq x < \sum_{i=1}^{j} w_i$$

   ($\sum_{i=1}^{0} w_i$ is defined to be zero.)
3. Return the example (x_j, l_j)

FIG. 3. A description of the algorithm for boosting by sub-sampling.

far the learning algorithm is from a completely useless algorithm and arise naturally in the design and analysis of our boosting algorithms. We shall show that the resources required by our algorithms are uniformly bounded by functions whose dependence on $1/\gamma$, $1/\lambda$, $1/\varepsilon$, and $1/\delta$ is either logarithmic or low-order polynomial.

For the main part of our analysis, in Sections 3.2 and 3.3, we restrict ourselves to boosting deterministic learning algorithms that generate deterministic hypotheses. Later, in Section 3.4, we show that all of our algorithms and their analysis hold, with very little change, for the case that the learning algorithm and the resulting hypotheses are randomized.

3.2. *Boosting Using Sub-sampling*

One simple way of applying the results of the majority-vote game to boost the performance of **WeakLearn** is by using it to find a small hypothesis that is consistent with a large set of training examples. The algorithm $\mathbf{B}_{\text{Samp}}$, which is summarized in Fig. 3, is based on this principle.

The first step of $\mathbf{B}_{\text{Samp}}$ is to collect a training set. Formally, this means making m calls to **EX**, generating the set $S = \{(x_1, l_1), \ldots, (x_m, l_m)\}$.[6] The goal of boosting is to generate a hypothesis that is correct on all examples in S.

As the sample is a finite set of size m, the requirement that a hypothesis is correct on all points in the sample is equivalent to the requirement that the hypothesis has error smaller than $1/m$ with respect to the uniform distribution on the sample. In order to do that, $\mathbf{B}_{\text{Samp}}$ generates different distributions on the training sample, and each time calls **WeakLearn** to generate a weak hypothesis, that is, a hypothesis that has error smaller than $1/2 - \gamma$ with respect to the given distribution. Each different distribution forces **WeakLearn** to generate a weak hypothesis whose errors are on different sample points.[7] The goal of the boosting

[6] In many actual machine learning scenarios, the training set S is the basic input to the learning algorithm, and thus this step is only formal.

[7] Ignoring, for a moment, the fact that **WeakLearn** has probability δ_0 of failing to generate a weak hypothesis.

algorithm is to control the location of these errors in such a way that after a small number of weak hypotheses have been generated, the majority vote over all weak hypotheses will give the correct label on each point. In other words, for each point in S, the fraction of the weak hypotheses that assign the point with the correct label is larger than half.[8]

The problem of generating these distributions is equivalent to the problem of the booster in the majority-vote game described in the previous section, under the following correspondence of terms. The *value* of a point corresponds to the probability assigned to the point by the target distribution (the uniform distribution in our case). The *weight* of a point corresponds to the probability assigned to it by the boosting algorithm. The decision of the adversary to *mark* a point corresponds to the decision by **WeakLearn** to generate a weak hypothesis that is correct on the point. The *reward set* corresponds to the set on which the majority vote over the weak hypotheses is correct and the *loss* is the probability that the majority makes a mistake, measured with respect to the target distribution. This correspondence lies at the center of the analysis of algorithm $\mathbf{B}_{\text{Samp}}$.

Before we give the first theorem regarding the performance of $\mathbf{B}_{\text{Samp}}$ we must address the fact that **WeakLearn** is not guaranteed to always generate a weak hypothesis. This event is only guaranteed to happen with probability λ. However, it is easy to check the hypothesis returned by **WeakLearn** and calculate its error on the sample. If this error is larger than $\varepsilon_0 = 1/2 - \gamma$, **WeakLearn** is called again, using a different subset of the examples in S.[9] This is the role of statement 3.a.ii of $\mathbf{B}_{\text{Samp}}$. However, this test has non-zero probability of failing any arbitrary number of times. In order to guarantee that the boosting algorithm has uniform finite running time, $\mathbf{B}_{\text{Samp}}$ tests only a pre-specified number of hypotheses. As we shall show in the second part of the Proof of Theorem 3.2, the probability that all these hypotheses will have error larger than ε_0 is smaller than $\delta/2$. The following theorem shows that if all k iterations manage to find a weak hypothesis, then the final hypothesis generated by $\mathbf{B}_{\text{Samp}}$ is consistent with all the labels in the sample.

THEOREM 3.1. *If all the hypotheses that are used by algorithm* $\mathbf{B}_{\text{Samp}}$ *are* ε_0*-good, then the hypothesis* h_M*, output by* $\mathbf{B}_{\text{Samp}}$*, is consistent on the sample* S.

Proof. From the correspondence with the majority-vote game defined above, and from Theorem 2.1, we get that the error of the hypothesis output by $\mathbf{B}_{\text{Samp}}$ is smaller than $1/m$. As the target distribution is uniform it assigns each point in S with probability $1/m$. Thus the output hypothesis must be correct on all points in S. ∎

Two issues remain in order to show that $\mathbf{B}_{\text{Samp}}$ is an effective learning algorithm. First, we need to show that there is a way for selecting m, the size of the sample S, so that the hypothesis generated by $\mathbf{B}_{\text{Samp}}$, which is guaranteed to be consistent on S, will also have a small probability of error on a random example outside of S. Second, we need to show that the algorithm uses uniformly bounded resources.

The fact that using a large enough sample guarantees that a consistent hypothesis will have small error on the whole domain stems from the fact that k, the number of hypotheses that are combined by the majority rule, increases like $O(\log |S|)$, as was proved in Corollary 2.3. Before getting into a detailed proof, let us give a rough sketch of a proof for a simple special case. Assume that the hypotheses generated by **WeakLearn** are chosen from a *finite* set of hypotheses H. Denote the set of hypotheses generated by $\mathbf{B}_{\text{Samp}}$ by H_M. The size of H_M is $|H|^{c \log m}$, where $c = 1/(2\gamma^2)$. Following the well-known analysis of Occam's razor principle [BEHW87] we get that the probability that the final hypothesis is consistent with a random sample of size m but has error larger than ε is smaller than $|H_M|(1-\varepsilon)^m = |H|^{c \log m}(1-\varepsilon)^m$. This quantity decreases rapidly with m. In particular, selecting m large enough that $m \geq (1/\varepsilon)(\log(1/\delta) + (1/2\gamma^2) \log m \log |H|)$ guarantees that the hypothesis will have error smaller than ε with probability larger than $1-\delta$.

Although this simple analysis gives the correct orders of magnitude, it is incomplete in that it depends on the size of H. In many cases this size is very large; moreover, often H is infinite or even uncountable. These cases can be analyzed using the notion of VC-dimension. However, Schapire [Sch90] suggested the following elegant proof that is based only on the assumption that the size of the sample used by **WeakLearn** is uniformly bounded. Although the final hypothesis is guaranteed to be consistent with the whole sample, which is of size m, the number of examples from the sample that are ever used by **WeakLearn** is $O(\log m)$. In other words, for large m only a small fraction of the training examples are ever used by **WeakLearn**!

This small subset of S ordered in the way which they were generated by **FiltEX** can be seen as a *representation* of the final hypothesis, h_M. For the sake of analysis we can imagine replacing $\mathbf{B}_{\text{Samp}}$ by the following version, which has the same external functionality. Instead of saving the hypotheses generated by **WeakLearn**, the boosting algorithm saves the sets of examples that were returned by **FiltEX** when it was called by **WeakLearn**. Later, when the value of $h_M(x)$ has to be calculated on some new example x, **WeakLearn** is rerun. The saved sequences of examples are

[8] Here, and in the rest of this section, we make no assumption about the output of the majority vote when the number of votes is split evenly. When we calculate upper bounds on the probability of mistake we use the pessimistic assumption that all these cases are decided incorrectly. We use the opposite assumption for the lower bounds.

[9] Note that as **WeakLearn** is guaranteed to succeed with probability at least λ on *any* distribution over the sample space, it is guaranteed to succeed on the uniform distribution over S.

used by **WeakLearn** to regenerate the weak hypotheses;[10] then using these weak hypotheses, $h_M(x)$ is reconstructed. Representing hypotheses by means of a subset of the training examples has been further studied by Littlestone *et al.* [LW86, FW93].

We now use prove a bound on the size of the sample that $\mathbf{B}_{\text{Samp}}$ has to use in order to guarantee that the final hypothesis has error smaller than ε. In the proof of this theorem we use a technique invented by Littlestone and Warmuth [LW86] in the above-mentioned work which appears as Appendix A in [FW93].

THEOREM 3.2. *Let* **WeakLearn** *be a deterministic learning algorithm that generates, with probability $\lambda > 0$ over the random training examples with which it is trained, a deterministic hypothesis whose error is smaller than $1/2 - \gamma$, for some $\gamma > 0$. Assume the number of training examples required to achieve this is uniformly bounded by m_0. Then the hypothesis h_M generated by $\mathbf{B}_{\text{Samp}}$ has the following property.*

For any $\varepsilon, \delta > 0$, if $\mathbf{B}_{\text{Samp}}$ uses a sample of size at least m, where

$$m \geq \frac{1}{\varepsilon}\left(\ln\frac{2}{\delta} + \frac{m_0}{2}\left(\frac{\ln m + 1}{\gamma}\right)^2\right),$$

then the probability that h_M has error larger than ε is smaller than δ. Here the probability is defined over the random choice of the sample S and over the internal random coin flips in $\mathbf{B}_{\text{Samp}}$.

Proof. We are interested in bounding the probability of the set of samples and internal coin flips of $\mathbf{B}_{\text{Samp}}$ that generate a hypothesis that has error larger than ε. We do that by covering this set by two disjoint sets. The first set is the set of samples and coin flips that cause $\mathbf{B}_{\text{Samp}}$ to generate a hypothesis that is consistent with the sample and yet has error larger than ε. The second is the set of samples and coin flips that cause $\mathbf{B}_{\text{Samp}}$ to generate a hypothesis that is inconsistent with the sample. The first and second parts of the proof bound the probabilities of these two sets respectively.

Part 1. We want to show that there is only a small probability that a random sequence of training examples $S = \langle (x_1, l_1), \ldots, (x_m, l_m) \rangle$, labeled according to $c \in \mathbf{C}$, can cause $\mathbf{B}_{\text{Samp}}$ to generate a hypothesis that is consistent with S but has error larger than ε.

We first sketch the argument. We consider the following mapping of arbitrary sequences of km_0 labeled examples into hypotheses. The sequence is partitioned into k blocks of length m_0, and each block is fed into **WeakLearn**. Using this block **WeakLearn** generates a hypothesis.[11] Finally, these k hypotheses are combined by a majority vote to generate a single hypothesis. We define two properties on sequences chosen out of S that are based on the hypothesis to which these sequences are mapped. The first property is that the hypothesis is consistent with all the examples in S; the second property is that the hypothesis has error larger than ε with respect to the distribution $\mathscr{D}$ and the underlying concept. We call sequences that have both properties "bad" sequences. We show that the probability of a sample S from which a bad sequence can be chosen is very small. However, if by using some sequence of coin flips $\mathbf{B}_{\text{Samp}}$ can generate a consistent hypothesis that has a large error, then there *exists* a way of choosing a bad sequence out of S, which means that the probability of $\mathbf{B}_{\text{Samp}}$ generating such a hypothesis is small.

To bound the probability of samples S from which a bad sequence can be chosen, one can view the elements of S that are not in the sequence as random test points on which the hypothesis is tested. As most of the points in S are not in the sequence, it is very unlikely that the hypothesis is consistent with all these examples and yet has a large probability of making an error. This observation, together with the fact that the total number of sequences of km_0 elements from S is not too large, gives us the proof of this part of the theorem.

We now give the formal proof, which is an adaptation of a technique used by Warmuth and Littlestone in [LW86]. Fix any concept $c \in \mathbf{C}$. Let $S = \langle (x_1, l_1), \ldots, (x_m, l_m) \rangle$ be the sequence of randomly drawn training examples returned by **EX** in step 1 of a specific run of $\mathbf{B}_{\text{Samp}}$ such that for all i, $l_i = c(x_i)$. Let $S' = \langle (x_{t_1}, l_{t_1}), \ldots, (x_{t_d}, l_{t_d}) \rangle$ denote a sequence of examples chosen out of S.

Let $\mathscr{T}$ be the collection of all m^d sequences of length $d = km_0$ of integers in $\{1, \ldots, m\}$. For any sequence of examples $S = \langle (x_1, l_1), \ldots, (x_m, l_m) \rangle$ and for any $T \in \mathscr{T}$ we denote $\langle (x_{t_1}, l_{t_1}), \ldots, (x_{t_d}, l_{t_d}) \rangle$ by S'_T. We denote the hypothesis to which this sequence is mapped by the mapping defined above by $h_M(S'_T)$.

Fixing T, let U_T be the set of all sequences of examples S such that the hypothesis $h_M(S'_T)$ has error larger than ε. Recall that the error of h_M is the probability, with respect to the distribution $\mathscr{D}$, of the symmetric difference between h_M and c. Let C_T be the set of all sequences S such that $h_M(S'_T)$ is consistent with all the examples in S. Observe that each run of $\mathbf{B}_{\text{Samp}}$ in which it generates a consistent hypothesis corresponds to a sequence of indices T such that C_T contains the training set S that was used by the algorithm. If $\mathbf{B}_{\text{Samp}}$ has non-zero probability of generating a consistent hypothesis that has a large error when using the sample S, then there must exist some $T \in \mathscr{T}$ such that $S \in C_T \cap U_T$.

[10] Note that this analysis is valid only when **WeakLearn** is deterministic. In Section 3.4 we show how to analyze the non-deterministic case.

[11] We assume that **WeakLearn** is deterministic and returns a hypothesis for any sequence of m_0 examples.

We can thus upper bound the probability of failure over the random choice of S by requiring that

$$\sum_{T \in \mathcal{T}} P^m(C_T \cap U_T) \leq \delta/2.$$

The choice of the hypothesis $h_M(\langle (x_{t_1}, l_{t_1}), \ldots, (x_{t_d}, l_{t_d}) \rangle)$ is only a function of the d elements of S'_T. If $S \in U_T$, the hypothesis has probability at least $1-\varepsilon$ of making a mistake on any of the remaining $m-d$ elements of S which are chosen at random, independent of the elements in S'_T and of the fact that $S \in U_T$. Thus the probability that S is in C_T, given that it is in U_T, is at most $(1-\varepsilon)^{m-d}$. Multiplying this probability by the size of $\mathcal{T}$ we get

$$m^d(1-\varepsilon)^{m-d} \leq \delta/2. \tag{15}$$

By substituting $d = km_0$ we find that it is sufficient to require that

$$m^{km_0}(1-\varepsilon)^{m-km_0} \leq \frac{\delta}{2},$$

which can be translated to the following stronger requirement on m:

$$m \geq \frac{1}{\varepsilon}\left(\ln\frac{2}{\delta} + km_0(\ln m + \varepsilon)\right).$$

We now use $1/(2\gamma^2)\ln(m/2)$ as a choice for k, the number of weak hypotheses that are combined by **WeakLearn**. Corollary 2.3 shows that this choice obeys the inequality of line 2 in $\mathbf{B}_{\text{Samp}}$. We thus find that it is sufficient to require that

$$m \geq \frac{1}{\varepsilon}\left(\ln(2/\delta) + m_0\frac{\ln(m/2)}{2\gamma_2}(\ln m + \varepsilon)\right).$$

As the statement of the theorem places a slightly stronger requirement on the minimal value of m, we get that if $\mathbf{B}_{\text{Samp}}$ generates a consistent hypothesis than this hypothesis has error smaller than ε with probability at least $1-\delta/2$.

Part 2. We now bound the probability that $\mathbf{B}_{\text{Samp}}$ generates a hypothesis that is not consistent with the sample. From Theorem 3.1 we know that if all of the k hypotheses generated by **WeakLearn** have error smaller than ε_0 with respect to the corresponding weightings of the sample, then the final hypothesis is consistent with the whole sample. It thus remains to be shown that for any sample S the probability, over the random choice made in $\mathbf{B}_{\text{Samp}}$ that any of the k hypotheses generated by **WeakLearn** has error larger than ε_0, is smaller than $\delta/2k$.

Note that each time a hypothesis is returned from **WeakLearn** its error on the weighted sample is checked, and it is rejected if the error is too large. Thus the only case in which a hypothesis used by $\mathbf{B}_{\text{Samp}}$ has an error larger than ε_0 is when all of the iterations of statement 3.a fail to generate a hypothesis with small error. As the probability that any single call to **WeakLearn** generates a good hypothesis is at least λ, the probability that all of the $(1/\lambda)\ln(2k/\delta)$ runs of **WeakLearn** performed in statement 3.a fail to generate a good hypothesis is at most

$$(1-\lambda)^{(1/\lambda)\ln(2k/\delta)} \leq \frac{\delta}{2k}.$$

Thus the probability that any of the k hypotheses used is not good is at most $\delta/2$. ∎

Theorem 3.2 gives a uniform upper bound on the sample complexity of $\mathbf{B}_{\text{Samp}}$. The bound is given in terms of an implicit inequality on m, which cannot be written as an exact explicit bound. The following corollary gives an explicit upper bound on the sample complexity needed for boosting using $\mathbf{B}_{\text{Samp}}$.

COROLLARY 3.3. *Let* **WeakLearn** *be a deterministic learning algorithm that generates, with probability $\lambda > 0$ over the random training examples with which it is trained, a deterministic hypothesis whose error is smaller than $1/2-\gamma$, for some $\gamma > 0$. Assume the number of training examples required to achieve this is uniformly bounded by m_0. Then, given any $\varepsilon, \delta > 0$, if $\mathbf{B}_{\text{Samp}}$ is required to generate a hypothesis that is consistent with a sample of size*

$$m \geq \max\left\{208, \frac{2}{\varepsilon}\ln\frac{2}{\delta}, 16\frac{m_0}{\varepsilon\gamma^2}\left(\ln\frac{m_0}{\varepsilon\gamma^2}\right)^2\right\},$$

then with probability larger than $1-\delta$ the hypothesis output by $\mathbf{B}_{\text{Samp}}$ has error smaller than ε.

Proof. We want to find m that will satisfy

$$m \geq \frac{1}{\varepsilon}\left(\ln\frac{2}{\delta} + \frac{m_0}{2}\left(\frac{\ln m + 1}{\gamma}\right)^2\right).$$

It suffices if m is larger than the maximum of twice each of the two terms in the right-hand side. From the first term we get $m > (2/\varepsilon)\ln(2/\delta)$. To bound m with respect to the second term, we observe that, in general, in order to satisfy $m > a(\ln m + 1)^2$ it suffices to choose $m = 16a(\ln a)^2$, if $a \geq 5$. It thus suffices if $m > 16a(\ln a)^2 = 16 * 5 * (\ln 5)^2$, or if $m > 208$. ∎

We now discuss the time and space complexity of $\mathbf{B}_{\text{Samp}}$. One easily observes that the total number of times that **WeakLearn** is called is

$$O\left(\frac{k}{\lambda}\ln\frac{2k}{\delta}\right) = O\left(\frac{\ln m}{\gamma^2\lambda}\left(\ln\frac{1}{\gamma^2\delta} + \ln\ln m\right)\right).$$

On the other hand, statements 3.a.ii, 3.b, 3.c, and 3.d in Fig. 3, which test and update the weights associated with the sample, each take $O(m)$ time to execute. It is thus clear that for large values of m, the time complexity of $\mathbf{B}_{\text{Samp}}$ is dominated by the time for manipulating the sample and not by the time taken by **WeakLearn**. This gives the $\mathbf{B}_{\text{Samp}}$ a time complexity of $O((km/\lambda)\ln(k/\delta))$.

The space complexity of $\mathbf{B}_{\text{Samp}}$ is dominated by the storage of the sample. The sample size is, ignoring log factors, $\tilde{O}(1/\varepsilon)$ (Corollary 3.3), while the storage of the hypotheses generated by **WeakLearn** is $O(k) = O(1/\gamma^2 \log 1/\varepsilon)$. In the next section we present a different boosting algorithm whose space complexity is $O(\log 1/\varepsilon)$ rather than $\tilde{O}(1/\varepsilon)$.

3.3. *Boosting Using Filtering*

In the previous section we have developed one way of applying the optimal weightor strategy for the majority-vote game to the problem of boosting a weak learner. While the complexity bounds for this method are reasonably good, considerable improvement is possible in the space complexity. The space complexity of $\mathbf{B}_{\text{Samp}}$ is dominated by the storage of the training examples. In some applications the training set is in the memory anyway and this cost is taken for granted. However, in other cases (such as on-line learning), storing all the training examples in memory might be very expensive. Recall that in order to find a hypothesis with error smaller than ε, only $O(\log(1/\varepsilon))$ out of the $O(1/\varepsilon(\log(1/\varepsilon))^2)$ training examples in the sample are ever used by the weak learning algorithm. In this section we present algorithms that select the examples used by **WeakLearn** in an on-line fashion from the sequence of examples supplied by **EX**. This avoids storing many examples in memory and decreases the space complexity to $O(\log(1/\varepsilon))$. Selecting examples directly out of the input stream is the basis of Schapire's boosting algorithm [Sch90]. Schapire used the term "filtering" to describe this

Algorithm $\mathbf{B}_{\text{Filt}}$
Input: **EX**, **WeakLearn**, $\gamma, \lambda, \epsilon, \delta$
Output: A hypothesis h_M, that has error smaller than ϵ with probability at least $1 - \delta$.

1. Find a (small) k that satisfies

$$\sum_{i=\lceil k/2 \rceil}^{k} \binom{k}{i} (1/2 - \gamma/2)^i (1/2 + \gamma/2)^{k-i} < \epsilon^2$$

   (For example, $k \geq 4/\gamma^2 \ln(1/\epsilon)$ suffices)

2. Repeat the following steps for $i = 0 \ldots k-1$,
   setting #accept and #reject to zero before each iteration.
   (a) Call $\mathbf{B}_{\text{Rel}}$, referring it to **FiltEX** and **WeakLearn**.
   If $\mathbf{B}_{\text{Rel}}$ does not abort it generates a hypothesis: h_{i+1},
   whose error is smaller than $1/2 - \gamma/2$ with probability at least $1 - \delta/2k$.
   (b) If $\mathbf{B}_{\text{Rel}}$ aborts, then define h_{i+1} to be a hypothesis that always makes a random prediction using a fair coin.
3. Return as the final hypothesis, h_M, the majority vote over $h_1, \ldots, h_k$.

Subroutine FiltEX
Repeat the following command until an example is *accepted* or until the abort condition is satisfied.

1. Call **EX**, and receive a labeled example (x, l).
2. If $i = 0$ then accept the example and return, else continue to 3.
3. Set r to be the number of indices $1 \leq j \leq i$ such that $h_j(x) = l$, and calculate

$$\alpha_r^i = \binom{k-i-1}{\lfloor \frac{k}{2} \rfloor - r} (1/2 + \gamma/2)^{\lfloor \frac{k}{2} \rfloor - r} (1/2 - \gamma/2)^{\lceil \frac{k}{2} \rceil - i - 1 + r} \quad ; \quad \alpha_{\max}^i = \max_{0 \leq r \leq i} \alpha_r^i$$

4. choose a real number x uniformly at random from the range $0 \leq x \leq 1$.
5. If $x < \alpha_i^i / \alpha_{\max}^i$ then accept the example, and return it as the result,
   else reject it and jump to 1.
   In each case, update #accept and #reject accordingly.

The abort condition:

$$\#\text{accept} + \#\text{reject} > \frac{k\gamma\alpha_{\max}^i}{\epsilon(1-\epsilon)} \max\left(\#\text{accept}, 4\ln\frac{8k^2\gamma\alpha_{\max}^i}{\delta\epsilon(1-\epsilon)}\right)$$

FIG. 4. A description of the algorithm for boosting by filtering.

process. The selection is viewed as a "filter" that lies between the source of examples, **EX**, and the weak learning algorithm. This filter observes each example generated by **EX** and either *rejects* it and throws it away or *accepts* it and passes it on to **WeakLearn**.

The description of the algorithm is given in Fig. 4. The overall structure of the algorithm is very similar to that of $\mathbf{B}_{\text{Samp}}$. The boosting algorithm generates k weak hypotheses by calling **WeakLearn** k times, each time presenting it with a different distribution over the training examples. However, while in $\mathbf{B}_{\text{Samp}}$ the examples are drawn from a set of examples that is fixed, once and for all, at the beginning of the process; in $\mathbf{B}_{\text{Filt}}$ new examples are continually drawn from the sample space by calling **EX**. Each time a new example is drawn, its weight is calculated, and a stochastic decision is made whether to accept or reject the example, such that the probability of acceptance is proportional to the weight. The proportionality constant, $1/\alpha^i_{\max}$, is chosen to maximize the probability of accepting a random example without violating the condition that the probability of acceptance should be in the range $[0, 1]$.

The analysis of $\mathbf{B}_{\text{Filt}}$ corresponds to playing the majority-vote game directly on the sample space, X, and the input distribution, $\mathscr{D}$, and not on the uniform distribution over a sample, as is the case with $\mathbf{B}_{\text{Samp}}$. This simplifies the analysis with respect to the analysis of $\mathbf{B}_{\text{Samp}}$ in that there is no gap between the expected error on the training set and the expected error on a random example. On the other hand, the analysis becomes more involved as a result of the following potential problem. It might happen that during some iterations of statement (2) a large fraction of the examples generated by **EX** are rejected. As a result, the number of examples that have to be filtered in order to generate the training examples required by **WeakLearn** becomes prohibitively large. Luckily, as we shall show, the accuracy of the hypotheses that are generated by **WeakLearn** in such iterations has very little influence on the accuracy of the final hypothesis, h_M, that is the final result of $\mathbf{B}_{\text{Filt}}$.

We use this property by defining an "abort" condition. This condition, defined at the bottom of Fig. 4, detects iterations in which the fraction of accepted examples is small. We refer to such an event as *triggering* the abort condition. When the abort is triggered, it stops the execution of procedure **FiltEX** *and* the run of procedure **WeakLearn** that called it, and returns control to statement (2.b). A random hypothesis is then put in place of the hypothesis that was supposed to be generated by **WeakLearn**. The random hypothesis is simply an algorithm that for any $x \in X$ generates a label in $\{0, 1\}$ by flipping a fair coin. The abort condition is defined as a function of two counters, `#accept` and `#reject`, that are incremented each time an example, generated by **EX**, is accepted or rejected respectively. Both counters are reset to zero each time the index i in statement (2) is incremented.

In order to analyze Algorithm $\mathbf{B}_{\text{Filt}}$ we need to go back to the analysis of the underlying majority-vote game. In order to do that we introduce again some of the notation used in Section 2 and define it in the context of our new problem.

Let X be the sample space over which a probability distribution $\mathscr{D}$ is defined. Define $\{X^i_0, X^i_1, \ldots, X^i_i\}$ to be a partition of X into $i+1$ sets where X^i_r consists of that set of the sample space that is labeled correctly by r out of the first i hypotheses. Define the following quantities related to this partition (a complete notation table appears in Appendix A.3):

$M^i_r = X^i_r \cap X^{i+1}_{r+1}$	the subset of X^i_r that is correctly labeled by the $i+1$st hypothesis
$q^i_r = \Pr(X^i_r)$	
$\alpha^i_r = \Pr(M^i_r)/\Pr(X^i_r)$	the probability of a random example to be correctly labeled by h_M given that it is in X^i_r
$t_i = \sum_{r=0}^{i} q^i_r \alpha^i_r$	the probability of accepting a random example during the construction of h_{i+1} is $t_i/\alpha^i_{\max}$.

We start our analysis by quantifying the reliability of the abort condition. We say that the triggering of the abort condition is *justified* if $t_i < 2\varepsilon(1-\varepsilon)/(k\gamma)$. The following lemma shows that most trigering events are justified.

LEMMA 3.4. *For all $0 \leqslant i \leqslant k-1$, the probability, over the distribution of the examples, that an abort is triggered during the generation of h_{i+1}, given that $t_i \geqslant 2\varepsilon(1-\varepsilon)/(k\gamma)$, is smaller than $\delta/2k$.*

Proof. We start by recasting the abort condition in a notation that is more convenient for the analysis. Let $n = \#\texttt{accept} + \#\texttt{reject}$ and $m = \#\texttt{accept}$. We define the constants $c = 2\varepsilon(1-\varepsilon)/k\gamma\alpha^i_{\max}$ and $n_0 = (8/c)\ln(16k/c\delta)$. Using this notation we see that an abort occurs after testing the nth example if and only if $n > n_0$ and $m < cn/2$. We use $q = t_i/\alpha^i_{\max}$ to denote the probability that **FiltEX** accepts a random example generated by **EX**. Thus the claim that we want to prove is that if $q \geqslant c$ then the probability of an abort (during any one of the k iterations) is smaller than $\delta/2k$. This probability can be written as a sum of the probabilities of aborting after each example after example number n_0. We can bound the probability of aborting after the nth example using Chernoff bounds as follows:

$$\Pr(m < cn/2) \leqslant e^{-cn/8}.$$

Summing this probability over all possible values of n we get that

$$\Pr(\text{abort occurs after } n > n_0 \text{ examples}) < \sum_{n=n_0}^{\infty} e^{-cn/8} = \frac{e^{-cn_0/8}}{1-e^{-c/8}} < \frac{8}{c} e^{-cn_0/8} < \frac{\delta}{2k},$$

which proves the claim. ∎

In order for the algorithm $\mathbf{B}_{\text{Filt}}$ to work successfully, we need the reliability of **WeakLearn** to be high. However, as noted by Haussler *et al.* [HKLW91], it is easy to boost the reliability of a learning algorithm. We give the performance of one possible reliability-boosting algorithm, $\mathbf{B}_{\text{Rel}}$, in the following lemma. The proof of the lemma and the description of the algorithm are given in Appendix B.

LEMMA 3.5. *Assume* **WeakLearn** *is a learning algorithm that generates hypotheses whose error is smaller than* $1/2-\gamma$ *with probability at least* $\lambda > 0$, *using* m_0 *examples. Then, for any* $\delta > 0$, *Algorithm* $\mathbf{B}_{\text{Rel}}$ *will generate hypotheses whose error is smaller than* $1/2-\gamma/2$ *with probability* $1-\delta$. *Furthermore, the number of examples required by algorithm* $\mathbf{B}_{\text{Rel}}$ *is at most*

$$\frac{8}{\gamma^2}\left(\ln\ln\frac{2}{\delta}+\ln\frac{1}{\delta\lambda}\right)+\frac{m_0}{\lambda}\ln\frac{2}{\delta}.$$

We now give the two main theorems regarding $\mathbf{B}_{\text{Filt}}$. The first theorem proves the correctness of the algorithm and the second proves a bound on the number of training examples required by the algorithm.

THEOREM 3.6. *Let* **WeakLearn** *be a learning algorithm that, for any distribution over the sample space* X *and any* $c \in \mathbf{C}$, *generates a hypothesis whose error is smaller than* $1/2-\gamma$ *with probability* λ *for some* $\gamma, \lambda > 0$. *Then, for any* $\delta, \varepsilon > 0$, *the algorithm* $\mathbf{B}_{\text{Filt}}$, *given* λ, γ, ε, *and* δ, *generates a hypothesis whose error is smaller than* ε *with probability at least* $1-\delta$.

The proof of this theorem is based on the potential function, β_r^i, defined in Section 2. In order to analyze the behavior of the average potential on aborted iterations we use the following refinement of Lemma 2.2. Recall that in the majority vote game the chooser is required to choose sets whose weight is larger than some constant larger than $1/2$; here we denote this constant by $1/2+\gamma'$. The weightor assigns weights to sets according to γ'. Suppose that the *actual* weight of the set that the chooser chooses on iteration i is $\hat{\gamma}_i$. Lemma 2.2 guarantees that if the chooser makes a legal choice, i.e., if $\hat{\gamma}_i \geq \gamma'$, then the average potential does not increase. The following lemma gives a more refined statement, which expresses the change in the average potential as a function of the difference $\gamma' - \hat{\gamma}_i$.

LEMMA 3.7. *Suppose that the weightor in the majority vote game assigns weights according to* γ', *and that* $\hat{\gamma} = \sum_{r=0}^{i} W(M_r^i)$. *Then the increase in the average potential, where* β_r^i *is defined according to* γ', *is*

$$\sum_{r=0}^{i+1} q_r^{i+1}\beta_r^{i+1} = \sum_{r=0}^{i} q_r^i\beta_r^i + (\gamma'-\hat{\gamma}_i)\sum_{r=0}^{i} q_r^i\alpha_r^i.$$

Proof. Recall Eq. (9) from the proof of Lemma 2.2:

$$\begin{aligned}\sum_{r=0}^{i+1} q_r^{i+1}\beta_r^{i+1} &= \sum_{r=0}^{i} q_r^i[(1-x_r^i)\beta_r^{i+1} + x_r^i\beta_{r+1}^{i+1}] \\ &= \sum_{r=0}^{i} q_r^i\beta_r^{i+1} + \sum_{r=0}^{i} q_r^i x_r^i(\beta_{r+1}^{i+1}-\beta_r^{i+1}).\end{aligned}$$

From the definition of $\hat{\gamma}_i$, we obtain, following the same line of argument as that in Eqs. (10) to (13), that

$$\sum_{r=0}^{i} q_r^i x_r^i(\beta_{r+1}^{i+1}-\beta_r^{i+1}) = (1/2+\hat{\gamma}_i)\sum_{r=0}^{i} q_r^i(\beta_{r+1}^{i+1}-\beta_r^{i+1}). \tag{16}$$

Combining Eqs. (9) and (16), we obtain

$$\begin{aligned}\sum_{r=0}^{i+1} q_r^{i+1}\beta_r^{i+1} &= \sum_{r=0}^{i} q_r^i\beta_r^{i+1} + (1/2+\hat{\gamma}_i)\sum_{r=0}^{i} q_r^i(\beta_{r+1}^{i+1}-\beta_r^{i+1}) \\ &= \sum_{r=0}^{i} q_r^i\beta_r^{i+1} + (1/2+\gamma)\sum_{r=0}^{i} q_r^i(\beta_{r+1}^{i+1}-\beta_r^{i+1}) \\ &\quad + (\gamma-\hat{\gamma}_i)\sum_{r=0}^{i} q_r^i(\beta_r^{i+1}-\beta_{r+1}^{i+1}) \\ &= \sum_{r=0}^{i} q_r^i[(1/2+\gamma)\beta_{r+1}^{i+1} + (1/2-\gamma)\beta_r^{i+1}] \\ &\quad + (\gamma-\hat{\gamma}_i)\sum_{r=0}^{i} q_r^i(\beta_r^{i+1}-\beta_{r+1}^{i+1}).\end{aligned}$$

Using Eq. (5) for the first term and Eq. (7) for the second term we find that

$$\sum_{r=0}^{i+1} q_r^{i+1}\beta_r^{i+1} = \sum_{r=0}^{i} q_r^i\beta_r^i + (\gamma-\hat{\gamma}_i)\sum_{r=0}^{i} q_r^i\alpha_r^i,$$

which is the statement of the lemma. ∎

Proof of Theorem 3.6. From Lemma 3.4 we know that the probability that any of the times the abort condition has been triggered is unjustified is smaller than $\delta/2$. On the other hand, the properties of Algorithm $\mathbf{B}_{\text{Rel}}$, given in Lemma 3.5, guarantee that for each iteration, $0 \leq i \leq k-1$, the probability that the error of h_i is larger than $1/2-\gamma/2$ is smaller than $\delta/2k$. Combining these claims we get that with probability at least $1-\delta$ all the hypotheses have error smaller than $1/2-\gamma/2$ and all the times the abort condition is triggered are justified. We shall now show that in this case the error of h_M is smaller than ε.

In applying Lemma 3.7 to our analysis, we choose γ' to be $\gamma/2$. It is easy to check that the probability of any set of examples A according to the filtered distribution generated by **FiltEX** is equal to the weight assigned to A by the

weighting scheme defined in Eq. (3) where γ' replaces γ. We equate the set chosen by the chooser with the set of instances on which $h_i(x) = c(x)$. Under this correspondence, the probability that $h_i(x)$ is correct, when measured with respect to the filtered distribution, corresponds to the weight of the set chosen by the chooser.

Consider first the iterations $1 \leqslant i \leqslant k$ in which the abort condition is not triggered, i.e., the hypothesis h_i is successfully generated and has error smaller than $1/2 - \gamma'$. In this case we have that $\hat{\gamma}_i > \gamma'$ and thus Lemma 3.7 implies that the average potential can only decrease.

Next consider the aborted iterations. The error of a random coin flip with respect to *any* distribution over the examples is, by definition, one half. Thus $\hat{\gamma}_i = 1/2$ and we get from Lemma 3.7 that in the aborted iterations the average potential increases by at most $\gamma' \sum_{r=0}^{i} q_r^i \alpha_r^i = \gamma t_i/2$. As we assume all the aborts are justified, we know that $t_i < 2\varepsilon(1-\varepsilon)/k\gamma$.

Combining the aborted and non-aborted iterations, we find that the total potential increase in all k iterations is at most $\varepsilon(1-\varepsilon)$. We now follow the same argument as in the Proof of Theorem 2.1. As the number of iterations, k, is chosen so that $\beta_0^0 \leqslant \varepsilon^2$, we get that

$$\Pr(h_M(x) \neq c(x)) = \sum_{r=0}^{k} q_r^k \beta_r^k \leqslant \beta_0^0 + \varepsilon(1-\varepsilon) \leqslant \varepsilon^2 + \varepsilon(1-\varepsilon) = \varepsilon,$$

where the probability is taken with respect to both the random choice of x according to $\mathcal{D}$ and the random coin flips of the dummy weak hypotheses. ∎

THEOREM 3.8. *The number of training examples required by* $\mathbf{B}_{\text{Filt}}$ *is smaller than*

$$m = \frac{4\sqrt{2}\, e^{1/12}}{\sqrt{3\pi}} \frac{k^{3/2}\gamma}{\varepsilon(1-\varepsilon)} \max\left(m_R,\ 4 \ln \frac{16k^2\gamma}{\delta\varepsilon}\right) < \frac{33}{\varepsilon\gamma^2}\left(\ln \frac{1}{\varepsilon}\right)^{3/2} \max\left(m_R,\ 12 \ln \frac{8 \ln 1/\varepsilon}{\gamma\delta\varepsilon}\right), \quad (17)$$

where k is the number of iterations as chosen in line 1 of $\mathbf{B}_{\text{Filt}}$ *and the inequality is obtained by using the suggested choice of k. The variable m_R denotes the number of examples for generating a weak hypothesis with reliability $1 - \delta/2k$ by* $\mathbf{B}_{\text{Rel}}$ *and is equal to*

$$m_R = \frac{m_0}{\lambda} \ln \frac{4k}{\delta} + \frac{8}{\gamma^2}\left(\ln \ln \frac{4k}{\delta} + \ln \frac{2k}{\delta\lambda}\right).$$

As discussed above, the factor $\alpha_{\max}^i$ is chosen so that the probability of accepting a random example is maximized without distorting the simulated distribution. As the value of $\alpha_{\max}^i$ plays a critical role in the proof of the Theorem 3.8, we start by presenting a tight upper bound on this value.

LEMMA 3.9. *For all iterations $0 \leqslant i \leqslant k-2$ of* $\mathbf{B}_{\text{Filt}}$,

$$\alpha_{\max}^i \leqslant \sqrt{8/3\pi(k-i-1)}\, e^{1/12}$$

The proof is given in Appendix D.

Proof of Theorem 3.8. The number of examples that are required by $\mathbf{B}_{\text{Rel}}$ to generate a hypothesis that has error smaller than $1/2 - \gamma/2$ with probability larger than $1 - \delta/2k$, denoted m_R, is easily bounded using Lemma 3.5. The abort condition guarantees that the number of examples that are tested by **FiltEX** during iteration i is at most

$$\frac{k\gamma\alpha_{\max}^i}{\varepsilon(1-\varepsilon)} \max\left(m_R,\ 4 \ln \frac{16k^2\gamma}{\delta\varepsilon}\right).$$

Thus the total number of examples is bounded by

$$\max\left(m_R,\ 4 \ln \frac{16k^2\gamma}{\delta\varepsilon}\right) \frac{k\gamma}{\varepsilon(1-\varepsilon)} \sum_{i=0}^{k-1} \alpha_{\max}^i. \quad (18)$$

Using Lemma 3.9 for $0 \leqslant i \leqslant k-2$ and observing that $\alpha_{\max}^{k-1} = 1$, we can bound the sum by

$$\sum_{i=0}^{k-1} \alpha_{\max}^i \leqslant \sqrt{\frac{8e^{1/6}}{3\pi}}\left(\sum_{j=1}^{k-1} \frac{1}{\sqrt{j}} + 1\right) < \sqrt{\frac{8e^{1/6}}{3\pi}}\, 2\sqrt{k}, \quad (19)$$

where the last inequality is true because

$$\sum_{j=1}^{k-1} \frac{1}{\sqrt{j}} < 1 + \int_1^{k-1} \frac{1}{\sqrt{x}} = 2\sqrt{k-1} - 1.$$

Combining (18) and (19) we get the equality in (17), and plugging in the choice $k = (4/\gamma^2)\ln(1/\varepsilon)$ we get the inequality. ∎

We conclude this section by briefly discussing the time and space complexity of $\mathbf{B}_{\text{Filt}}$. Assuming a uniform bound on the running time of **WeakLearn**, it is clear that the time complexity of $\mathbf{B}_{\text{Filt}}$ is dominated by the time spent in line 3 of **FiltEX** to calculate the labels assigned to the prospective example by the currently available weak hypotheses. As this time is proportional to the number of weak hypotheses available, we get that the time complexity of $\mathbf{B}_{\text{Filt}}$ is at most k times the sample complexity of $\mathbf{B}_{\text{Filt}}$. Similarly, assuming a uniform space complexity on **WeakLearn** and on the size of the hypotheses that it generates, it is clear that the space complexity of $\mathbf{B}_{\text{Filt}}$ is proportional to k, the number of hypotheses that need to be stored in memory.

3.4. *Randomized Learning Algorithms and Randomized Hypotheses*

In our discussion so far, we have concentrated on boosting deterministic weak learning algorithms that generate deterministic hypotheses. In this section we show that our results transfer, with little or no change, to the more general case in which both the weak learning algorithm and its hypotheses are allowed to be randomized, i.e., make use of flipping random coins.

Note that the data sent to the learning algorithm and the hypothesis already have a large degree of randomness, as the data consist of examples that are chosen at random. We now show a simple transformation that translates randomized learning algorithms into deterministic learning algorithms on a different sample space.

For our analysis we use the convention that the random bits that are used by a randomized algorithm are given to the algorithm as input when it is called. More specifically, we assume the algorithm is given a real-valued random number, r, chosen uniformly at random from $[0, 1]$ whose binary expansion is used as an infinite source of random bits.[12] We shall take special care that each bit in the binary expansion is used at most once during the run of the algorithm. Thus any random bit used at any point in the algorithm is independent of any other bit. For that reason the distribution of the outcome of the algorithm is equivalent to the distribution generated if each random bit is chosen by an independent coin flip. The transformation we present is only an analytical tool. As we shall see, the result of the analysis is that $\mathbf{B}_{\text{Filt}}$ can be applied to the randomized case without any change. The only change that is required for using $\mathbf{B}_{\text{Samp}}$ in the randomized case is that the sample size has to be slightly increased.

Assume A is a randomized weak learning algorithm that generates randomized hypotheses. Assume A can learn the concept class $\mathbf{C}$ for any distribution $\mathscr{D}$ on the sample space X. We now define a mapping μ that maps X, $\mathbf{C}$, A, and $\mathscr{D}$ to X', $\mathbf{C}'$, A', and $\mathscr{D}'$, where A' is a deterministic learning algorithm that generates deterministic hypotheses. The sample space X' consists of pairs of the form $\langle x, r\rangle$, where $x \in X$ and $r \in [0, 1)$. The probability measure $\mathscr{D}'$ is the measure generated by the crossproduct between the distribution $\mathscr{D}$ and the uniform distribution on $[0, 1)$. Each concept $c \in \mathbf{C}$ is mapped to a concept $c' \in \mathbf{C}'$ such that, for all $\langle x, r\rangle$, $c'(\langle x, r\rangle) = c(x)$. Finally, the algorithm A', receiving the training examples $\{(\langle x_1, r_1\rangle, l_1), \ldots, (\langle x_m, r_m\rangle, l_m)\}$, runs the algorithm A on the sample $\{(x_1, l_1), \ldots, (x_m, l_m)\}$, together with the number r_1 that is used by A as its source of random bits. The hypothesis h, generated by A, is transformed in a similar way: h', upon receiving an instance $\langle x, r\rangle$ as input, calls h to label x, giving it r as its source of random bits.

Note that an infinite sequence of bits can be partitioned into an infinite number of infinite subsequences. For concreteness, we define the nth subsequence of r to consist of the bits whose indices can be written as $(2i-1)2^{n-1}$ for some positive integer i. We denote this subsequence by r_n. Note that if r is chosen uniformly at random then all of its subsequences are also uniformly distributed.

Using these definitions we can now show how boosting the randomized learning algorithm A can be viewed as boosting the deterministic algorithm A' over the larger sample space. Transforming the algorithm for boosting by filtering, $\mathbf{B}_{\text{Filt}}$, is simpler. The change takes place in the procedure **FiltEX**. In each iteration the procedure receives an example $\langle x, r\rangle \in X'$ chosen at random according to $\mathscr{D}'$. It then separates x and r, and maps r into $r_1, \ldots, r_{i+1}$, which are independent random bit sequences. Sequences 1 to i are used for calculating $h_1(x, r_1), \ldots, h_i(x, r_i)$. Sequence number $i+1$ is returned to **WeakLearn**, in this case the algorithm A, for use as its source of random bits. By using this transformation the Proofs of Theorems 3.6 and 3.8 can be used without change, and thus $\mathbf{B}_{\text{Filt}}$ works equally well for randomized and deterministic learning algorithms.

The analysis of the algorithm for boosting by sampling, $\mathbf{B}_{\text{Samp}}$, is somewhat more complicated. That is because the same examples are repeatedly fed into A. Since the examples include the source of random bits, this might induce undesired dependencies between random bits used in different runs of A. To avoid this problem, we assume that an additional integer parameter, which we denote q, is supplied to A. This parameter directs algorithm A to use, as it source of random bits, the qth *subsequence* of the random sequence with which it is supplied. The parameter q is different each time A is called, and thus the random bits used by A are guaranteed to be independent. However, this addition changes somewhat the proof of Theorem 3.2, forcing us to increase the size of the sample that is used by $\mathbf{B}_{\text{Samp}}$, as is summarized in the following theorem.

THEOREM 3.10. *Let* **WeakLearn** *be a randomized learning algorithm that generates, with probability $\lambda > 0$ over its internal randomization and the random choice of the training examples, a randomized hypothesis whose error is smaller than $1/2 - \gamma$, for some $\gamma > 0$. Assume the number of training examples required to achieve this is uniformly bounded by m_0. Suppose that m, the size of the sample used by $\mathbf{B}_{\text{Samp}}$, obeys the following inequality:*

$$m \geq \max\left\{208, \frac{2}{\varepsilon}\ln\frac{2}{\delta}, 16a(\ln a)^2\right\},$$

$$\text{where} \quad a = \left(m_0 + \ln\frac{1}{\lambda} + \ln\ln\frac{1}{\gamma^2\delta}\right)\Big/\gamma^2. \qquad (20)$$

[12] We assume some convention is used for selecting one of the binary expansions when the expansion is not unique.

Then with probability at least $1-\delta$, *the hypothesis* h_M *generated by* $\mathbf{B}_{\text{Samp}}$ *has error smaller than* ε.

Note that this bound is similar to the one given in Corollary 3.3, but there is a dependence of the sample size on λ which does not exist when the algorithm is deterministic.

Proof. The essential difference from the Proof of Theorem 3.2 is that the number of possible hypotheses that can be generated from the sample is larger. In Theorem 3.2 this number is equal to the number of subsequences of size d that can be chosen from a sequence of size m, i.e., m^d. In our case it is the number of subsequences times the number of combinations of values of the parameter q that could have been used in the generation of the k good hypotheses. Assume that $q = ir + l$ where $i = 0, \ldots, k-1$ is the number of hypotheses that have been generated so far, $l = 1, \ldots, r$ is the counter of the attempts to generate a good ith hypothesis, and $r = (1/\lambda)\ln(2k/\delta)$. (These indices are used in statement 3 and 3.a in Fig. 3.) Using this convention it is clear that each one of the hypotheses can be chosen using one of r values, and the total number of combinations of values of q is r^k. Thus the basic inequality that replaces Inequality (15) is

$$r^k m^d (1-\varepsilon)^{m-d} < \delta/2. \tag{21}$$

And solving for the m that satisfies the inequality we get the following inequality:

$$m \geqslant \frac{1}{\varepsilon}\left(\ln\frac{2}{\delta} + \frac{m_0}{2}\left(\frac{\ln m + 1}{\gamma}\right)^2 + \frac{\ln m}{2\gamma^2}\left(\ln\frac{1}{\lambda}\ln\ln\frac{1}{\gamma^2\delta} + \ln\ln\ln m\right)\right).$$

Using the same argument that was used to prove Corollary 3.3 we get the statement of the theorem. ∎

3.5. *The Resources Needed for Polynomial PAC Learning*

So far in this paper we have considered learning algorithms that are designed to work for a single fixed concept class defined over a single fixed sample space. However, most learning algorithms can be used for a *family* of concept classes, and one is then interested in the way the performance of the learning algorithm depends on the complexity of the concept class. Valiant [Val84] presented a framework, called the PAC[13] learning framework, in which such quantification can be done. This framework is one of the most studied frameworks in computational learning theory. In this section we show the implications of our work in this framework.

[13] PAC learning stands for probably approximately correct learning.

We start by presenting some notation following Haussler *et al.* [HKLW91]. Assume that the sample space is a union of sample spaces of increasing complexity: $X = \bigcup_{n=1}^{\infty} X_n$. Similarly assume that the concept class that maps points in X_n to $\{0, 1\}$ is defined as a union of concept classes of increasing complexity: $\mathbf{C}_n = \bigcup_{s=1}^{\infty} \mathbf{C}_{n,s}$. The indices n and s usually denote the length of the description of an instance and a concept in some encoding scheme for X and for $\mathbf{C}$ respectively.

We say that a concept class $\mathbf{C}$ is *learnable*, or *strongly learnable*, if there exists a learning algorithm A and polynomials $p_1(\cdot,\cdot,\cdot,\cdot)$, $p_2(\cdot,\cdot,\cdot,\cdot)$ such that:

- For any n, s and any $\varepsilon, \delta > 0$, the algorithm A, given n, s, ε, δ and access to an example oracle **EX**, can learn any concept $c \in \mathbf{C}_{n,s}$ with respect to any distribution $\mathscr{D}$ on X_n and generate a hypothesis that has error smaller than ε with probability larger than $1-\delta$.
- The sample complexity of A, i.e., the number of calls that A makes to **EX**, is smaller than $p_1(n, s, 1/\varepsilon, 1/\delta)$.
- The running time of A is polynomial in $p_2(n, s, 1/\varepsilon, 1/\delta)$.

Kearns and Valiant [KV88, KV94] introduced a weaker form of learnability in which the error cannot necessarily be made arbitrarily small. A concept class $\mathbf{C}$ is *weakly learnable* if there exists a learning algorithm A and polynomials $p_1(\cdot,\cdot,\cdot)$, $p_2(\cdot,\cdot,\cdot)$, and $p_3(\cdot,\cdot)$ such that:

- For any n, s and any $\delta > 0$, the algorithm A, given n, s, δ and access to an example oracle **EX**, can learn any concept $c \in \mathbf{C}_{n,s}$ with respect to any distribution $\mathscr{D}$ on X_n and generate a hypothesis that has error smaller than $1/2 - 1/p_3(n, s)$ with probability larger than $1-\delta$.
- The sample complexity of A, i.e., the number of calls that A makes to **EX**, is smaller than $p_1(n, s, 1/\delta)$.
- The running time of A is polynomial in $p_2(n, s, 1/\delta)$.

In other words, a weak learning algorithm produces a prediction rule that performs just slightly better than random guessing.

Schapire [Sch90] has shown that the notions of weak and strong PAC learning are equivalent. Moreover, the boosting algorithm he invented provides an effective way for translating any weak learning algorithm into a strong learning algorithm. The boosting algorithm $\mathbf{B}_{\text{Filt}}$ presented in this paper provides a more efficient translation of weak learning algorithms to strong learning algorithms. A simple application of Theorem 3.8 gives the following upper bound on the resources required for PAC learning.

THEOREM 3.11. *If* $\mathbf{C}$ *is a weakly PAC learnable concept class, parameterized by* n *and* s *in the standard way* [HKLW91], *then there exists a PAC learning algorithm for* $\mathbf{C}$ *that learns with accuracy* ε *and reliability* δ *and*

- *requires a sample of size* $(1/\varepsilon)(\log 1/\varepsilon)^{3/2}(\log\log 1/\varepsilon + \log 1/\delta)\, p_1(n, s)$,
- *halts in time* $(1/\varepsilon)(\log 1/\varepsilon)^{5/2}(\log\log 1/\varepsilon + \log 1/\delta)\, p_2(n, s)$,
- *uses space* $(\log 1/\varepsilon)(\log\log 1/\varepsilon + \log 1/\delta)\, p_3(n, s)$, *and*
- *outputs hypotheses of size* $(\log 1/\varepsilon)\, p_4(n, s)$ *evaluatable in time* $(\log 1/\varepsilon)\, p_5(n, s)$ *for some polynomials* p_1, p_2, p_3, p_4, *and* p_5.

Proof. The PAC learning algorithm that we refer to is algorithm $\mathbf{B}_{\text{Filt}}$ applied to the given weak PAC learning algorithm. As n and s are passed without change to the weak learning algorithm, the dependence on n and s remains polynomial. Fixing λ and γ, Theorem 3.8 gives the dependence of the sample complexity on ε and δ. The time and space complexity of the boosted algorithm, as well as the size of the final hypothesis, follow from the discussion of the resources required by $\mathbf{B}_{\text{Filt}}$ which follows Theorem 3.8. ∎

We now compare this theorem to Theorem 4 in [Sch90]. The statement there is that the dependence of the sample and time complexity on ε is $O(1/\varepsilon\ \text{poly}(1/\varepsilon))$, and that the other dependencies on $1/\varepsilon$ are poly-logarithmic. Our theorem tightens these bounds by giving the explicit powers in the polynomials over $\log(1/\varepsilon)$ and $\log(1/\delta)$. Moreover, our more detailed bound, given in Theorem 3.8, shows explicitly the dependence on the parameters γ and m_0, which are hidden in Schapire's analysis. In the next section we show that some of these upper bounds are optimal.

3.6. *Relations to Other Bounds*

The bounds given in Theorems 3.8 and 3.11 are currently the best known bounds on the resources required for polynomial PAC learning of an arbitrary PAC learnable class. In this section we relate our results to known lower bounds and indicate where further improvement might be possible.

Theorem 3.11 shows that for any learnable concept class there exists a learning algorithm in *RP* for which the dependence of the sample size on the required accuracy, when all other parameters are fixed, is $O(1/\varepsilon(\log 1/\varepsilon)^{3/2})$. A general lower bound of $\Omega(1/\varepsilon)$ is given in [BEHW89] for learning any "non-trivial" concept class. This lower bound holds without regard to computational constraints on the learning algorithm. There exists a matching upper bound, given in [HLW88, Theorem 5.1], which says that, ignoring computational complexity, any concept class that can be learned using a sample of size polynomial in $1/\varepsilon$ can also be learned using a sample of size $O(1/\varepsilon)$ (ignoring the dependence on other problem parameters). The truth might be either that our upper bound can be reduced to match the lower bound or that there exists a better lower bound on the sample complexity of learning algorithms that are in *RP*. However, an improved lower bound would have to either assume or imply that $RP \neq NP$.

The number of weak hypotheses that are combined by our boosting algorithms is $O(1/\gamma^2 \ln(1/\varepsilon))$. We now show that this dependence of the number of required weak hypotheses on ε and γ is the best possible for any general boosting algorithm. A *general* boosting algorithm is a learning algorithm that can improve the accuracy of *any* PAC learning algorithm. As such, it cannot depend on any properties of the concept class. Knowledge about the concept class may only be used by the weak learning algorithm.

A *general boosting algorithm* receives as input four positive real valued parameters: ε, γ, δ, and δ_0. Its goal is to generate a hypothesis $h: X \to \{0, 1\}$ which is a close approximation of a target concept $c: X \to \{0, 1\}$ from an *unknown* concept class $\mathbf{C}$ with respect to the target distribution $\mathscr{D}$. The boosting algorithm operates in k iterations as follows. In iteration i the boosting algorithm defines a distribution $\mathscr{D}_i$ over the instance space X. The examples oracle **FiltEX** selects random instances from X according to the distribution $\mathscr{D}_i$ and labels them according to c. The weak learning algorithm, **WeakLearn**, which knows the concept class $\mathbf{C}$, is given access to **FiltEX** and generates a hypothesis $h_i: X \to \{0, 1\}$ such that $\Pr_{\mathscr{D}_i}(h_i(x) \neq c(x)) < 1/2 - \gamma$ with probability at least $1 - \delta_0$. The boosting algorithm receives these k weak hypotheses. In addition, it can receive m examples drawn according to the target distribution $\mathscr{D}$. Using this information, the boosting algorithm is required to generate a hypothesis h such that $\Pr_{\mathscr{D}_i}(h(x) \neq c(x)) < \varepsilon$ with probability at least $1 - \delta$.

We prove that the dependence of k on γ and ε is $k = \Omega((1/\gamma^2)\log(1/\varepsilon))$. The idea of the proof is simple. Suppose that the weak hypotheses were stochastic rules such that for each $x \in X$, $\Pr(h_i(x) = c(x)) = 1/2 + \gamma$ independently of anything else. In this case it is easy to show that the best way of combining the hypotheses is by a majority vote and that the number of weak hypotheses required is $\Omega((1/\gamma^2)\log(1/\varepsilon))$. The technical caveat in this argument is that one could use the *same* weak hypothesis several times and combine the outcomes to achieve the same small error. We thus need to demonstrate the existence of *deterministic* weak hypotheses whose errors behave in a way that is similar to independent random label noise.

For the sake of simplicity, we assume that the instance space X is the finite set of integers $\{1, \ldots, N\}$. We assume that N is large and define $\mathscr{F}_N$ to be a family of 2^N concept classes as follows. Let $r \in \{0, 1\}^N$ be the index of the concept class and denote by $r(x)$ the value of bit number x in r. We define the concept class $\mathbf{C}_r$ to include the following two concepts: $c_0^r(x) = r(x)$ and $c_1^r(x) = 1 - r(x)$. We further assume that the distribution $\mathscr{D}$ is the uniform distribution over $\{1, \ldots, N\}$.

We show that if k, the number of weak hypotheses, is significantly smaller than the number required by our

boosting algorithm for given values of ε and γ, then there exists some concept class in the family described above and a concept in that class, such that the error of the final hypothesis generated by the boosting algorithm when learning this concept is larger than ε. This is stated more formally in the following theorem.

THEOREM 3.12. *Let ε, γ, and δ be positive real numbers and let k be an integer such that*

$$\sum_{\lfloor k/2 \rfloor + 1}^{k} \binom{k}{i} (1/2-\gamma)^i (1/2+\gamma)^{k-i} > \varepsilon.$$

Let B be a general boosting algorithm. Then there exists a concept class $\mathbf{C}$, a concept $c \in \mathbf{C}$, and a weak learning algorithm **WeakLearn** *for $\mathbf{C}$ such that the following holds with probability at least $1-\delta$ over the random choice of the instances and the internal randomization of* **WeakLearn**.

- **WeakLearn** *returns hypotheses whose errors are at most $1/2-\gamma$ with respect to the corresponding distributions.*
- *If B calls* **WeakLearn** *at most k times then the error of the hypothesis generated by B is at least ε.*

Using standard approximations of the tails of binomial distributions, one can show that the lower bound on k given in the lemma implies that $k = \Omega((1/\gamma^2)\log(1/\varepsilon))$. Moreover, the expression for the number of weak hypotheses required by the boosting algorithms $\mathbf{B}_{\text{Filt}}$ and $\mathbf{B}_{\text{Samp}}$ has a similar binomial form (see statement 2 in Fig. 3 and statement 1 in Fig. 4). This means that the upper and lower bounds on k are closely related even when ε and γ are large.

Proof of the Theorem. We show that for a sufficiently large N, one of the concept classes in $\mathscr{F}_N$ satisfies the statement of the theorem. We prove the existence of this concept class using a probabilistic argument. Suppose that the concept class $\mathbf{C}_r \in \mathscr{F}_N$ is chosen randomly by selecting r uniformly at random from $\{0,1\}^N$. After this selection has been made $c \in \mathbf{C}_r$ is chosen randomly with equal odds to be either c_0^r or c_1^r. We shall show that the *expected* error of any boosting algorithm, with respect to this distribution over the target concepts, is larger than ε.

As each concept class includes only two concepts which disagree on each point of the domain, the learning problem faced by the weak learner is trivial. The weak learner can identify the concept after observing any single example. However, we construct the hypotheses so that the boosting algorithm can gain as little information as possible without violating the requirement on their performance with respect to the corresponding distributions. We assume that the weak learning algorithm is not limited in its computational power. In particular, by making many calls to **FiltEX**, **WeakLearn** can approximate $\Pr_{\mathscr{D}_i}(x)$ to within any desired accuracy for all $x \in X$.

Given the distribution $\mathscr{D}_i$ and the parameters γ and N, the weak learner selects the hypothesis h_i by choosing the set A_i, on which $h_i(x) = c(x)$, in the following way. First, the set A_i' is defined to contain all the elements $x \in X$ for which $\Pr_{\mathscr{D}_i}(x) \geqslant N^{-2/3}$. On the rest of the space, for each element in $X - A_i'$ we flip a random biased coin and add it to A_i'' with probability $1/2 + \gamma + N^{-1/12}$. We use γ' to denote $\gamma + N^{-1/12}$. The set A_i is defined to be $A_i' \cup A_i''$ and the hypothesis h_i is $c(x)$ if $x \in A_i$ and $1 - c(x)$ otherwise.

We need to show that, when N is sufficiently large, this procedure generates a hypothesis whose error, with respect to $\mathscr{D}_i$, is smaller than $1/2 - \gamma$ with high probability. Using the terminology of the majority-vote game, we call the probability with respect to $\mathscr{D}_i$ the *weight* of the set. As $h_i(x) = c(x)$ for all $x \in A_i'$, it suffices to show that the weight of A_i'' is at least $1/2 + \gamma$ of the weight of $X - A_i'$. The expected weight of A_i'' is $(1/2 + \gamma')\Pr_{\mathscr{D}_i}(X - A_i')$. It thus remains to show that the actual weight of A_i'' is, with high probability, close to the expected value. Observe that at most $N^{2/3}$ elements can have sufficient weight to be included in A_i'. Thus there are between $N - N^{2/3}$ and N elements in $X - A_i'$ and the weight of each element is at most $N^{-2/3}$. Thus the variance of the weight of A_i'' is at most $N^{1/2}N^{-2/3} = N^{-1/6} = o(N^{-1/12})$. As $\gamma' - \gamma = N^{-1/12}$, it is easy to show, using standard bounds on the deviation of large sums from their expected value, that the weight of A_i'' is larger than $(1/2 + \gamma)\Pr_{\mathscr{D}_i}(X - A_i')$ with probability approaching 1 as $N \to \infty$. This completes the proof that the weak hypotheses that are generated by the weak learner are legitimate weak hypotheses.

We now lower bound the error of the prediction of $c(x)$ that can be achieved by the boosting algorithm. We show this lower bound by constructing the *Bayes optimal* prediction rule for the specific weak learner described above. The optimal rule is divided into three cases depending on the instance x. In the first case one of the k distributions used during the boosting process assigns x a weight larger than $N^{-2/3}$; in this case we know that the corresponding hypothesis gives the correct label and we are done. As observed above, the total number of elements of this type is $kN^{2/3}$, and thus their total probability (with respect to the uniform distribution which is the target distribution) is at most $kN^{-1/3}$, which approaches zero as $N \to \infty$. In the second case x is an instance whose label has been observed in one of the examples accessed by B. We denote this set of instances by M. In this case the label of x is the one observed in the example. The probability of this event is at most $|M|/N$ which also approaches zero as $N \to \infty$. In the third case, which amounts to most of the probability, each of the k hypotheses has been chosen, independently at random, to be $c(x)$ with probability $1/2 + \gamma'$ and $1 - c(x)$ with probability $1/2 - \gamma'$. We can use Bayes formula to calculate the probability that $c(x) = 1$ given the values of the weak hypotheses. Denoting by $n(x)$ the number of hypotheses

such that $h_i(x)=1$, and setting $Z=X-M-\bigcup_{i=1}^{k} A_i''$, we get

$$\Pr(c(x)=1 \mid x\in Z \text{ and } n(x)=n) = \frac{\Pr(n(x)=n \mid x\in Z \text{ and } c(x)=1)\Pr(c(x)=1)}{\sum_{l\in\{0,1\}}\Pr(n(x)=n \mid x\in Z \text{ and } c(x)=l)\Pr(c(x)=l)}. \quad (22)$$

Here, the probabilities are measured with respect to the random choice of the example x, the concept class $\mathbf{C}_r\in\mathscr{F}_N$, the concept $c\in\mathbf{C}_r$, and the random choices made by **WeakLearn**. The method by which the concept class $\mathbf{C}_r$ and the concept $c\in\mathbf{C}_r$ are chosen implies that $\Pr(c(x)=1)=1/2$. The method by which the elements in A_i'' are chosen imply that

$$\Pr(c(x)=1 \mid x\in Z \text{ and } n(x)=n) = \frac{\frac{1}{2}(\frac{1}{2}+\gamma')^{n(x)}(\frac{1}{2}-\gamma')^{k-n(x)}}{\frac{1}{2}(\frac{1}{2}+\gamma')^{n(x)}(\frac{1}{2}-\gamma')^{k-n(x)}+\frac{1}{2}(\frac{1}{2}-\gamma')^{n(x)}(\frac{1}{2}+\gamma')^{k-n(x)}}. \quad (23)$$

It is easy to see that this equation gives a value larger than $1/2$ if and only if $n(x)>k/2$. Thus the Bayes optimal decision rule when $x\in Z$ is to predict the value of $c(x)$ according to the majority of the weak hypotheses.[14] The probability that this prediction rule is incorrect is lower bounded by

$$\begin{aligned}
&\Pr_{\mathscr{D}}(Z)\Pr(\mathrm{MAJ}(h_1(x),\ldots,h_k(x))\neq c(x))\\
&\quad\geqslant(1-|M|/N-kN^{-1/3})\sum_{i=\lfloor k/2\rfloor+1}^{k}\binom{k}{i}\\
&\qquad\times(1/2-\gamma')^i(1/2+\gamma')^{k-i}\\
&\quad\geqslant(1-|M|/N-kN^{-1/3})\sum_{i=\lfloor k/2\rfloor+1}^{k}\binom{k}{i}\\
&\qquad\times(1/2-\gamma-N^{-1/12})^i(1/2+\gamma+N^{-1/12})^{k-i}\\
&\quad\xrightarrow{N\to\infty}\sum_{i=\lfloor k/2\rfloor+1}^{k}\binom{k}{i}(1/2-\gamma)^i(1/2+\gamma)^{k-i}. \quad (24)
\end{aligned}$$

As this is a lower bound on the probability of mistake of the Bayes rule, it is a lower bound on the (expected) probability of a mistake of the hypothesis generated by *any* boosting algorithm. This, in turn, implies that for any boosting algorithm there exists a choice of concept class $\mathbf{C}_r$ and a concept $c\in\mathbf{C}_r$ such that the probability of mistake of the hypothesis generated by the boosting algorithm for this concept is at least as large as the lower bound given in Eq. (24). This, together with the assumption on k stated in the statement of the theorem, completes the proof. ∎

[14] If $n(x)=k/2$ then the Bayes rule is undefined; for the sake of the lower bound on the error we assume that the prediction made is always correct in this case.

4. EXTENSIONS

4.1. *Using Boosting for Distribution-Specific Learning*

So far, we have followed the distribution-free paradigm in computational learning and assumed that the learning algorithms that we attempt to boost have complexity bounds that hold uniformly for all input distributions. In this section we show that $\mathbf{B}_{\mathrm{Filt}}$, our second boosting algorithm, can boost learning algorithms whose accuracy is not uniformly bounded for all distributions. We will define a measure of discrepancy between distributions and show that the accuracy of **WeakLearn** can be allowed to degrade as the discrepancy increases between the filtered distribution that is fed into **WeakLearn** and the distribution that governs the example oracle **EX**. We shall refer to the distribution governing **EX** as the "target" distribution.

From Lemma 3.7 we know that the increase in the average potential in the ith iteration is equal to

$$\sum_{r=0}^{i+1} q_r^{i+1}\beta_r^{i+1} = \sum_{r=0}^{i} q_r^i\beta_r^i + (\gamma-\hat{\gamma}_i)\sum_{r=0}^{i} q_r^i\alpha_r^i,$$

where $\hat{\gamma}_i$ is the difference between $1/2$ and the error of h_i with respect to the filtered distribution in the ith iteration. We recall the notation defined in Section 3.3, $t_i=\sum_{r=0}^{i} q_r^i\alpha_r^i$, and re-write the previous equation as

$$\sum_{r=0}^{i+1} q_r^{i+1}\beta_r^{i+1} = \sum_{r=0}^{i} q_r^i\beta_r^i + (\gamma-\hat{\gamma}_i)\,t_i.$$

Recall that the probability of accepting a random example that is tested during the ith iteration is $t_i/\alpha_{\max}^i$. Thus, if the probability of accepting a random example during the ith iteration is small, then the sensitivity of the final accuracy to the accuracy of the ith hypothesis is small. We have already used this fact in the Proof of Theorem 3.6. There we used it to show that if the probability of accepting a random example is small enough, then a random coin flip can be used instead of the weak hypothesis. In this section we use the same property to relax the requirements on the accuracy of the hypotheses generated by **WeakLearn** for distributions that are far from the target distribution.

The following lemma shows how the requirements on the accuracy of the hypotheses generated by **WeakLearn** can be relaxed, allowing the generation of hypotheses whose error is larger than $1/2-\gamma$.

LEMMA 4.1. *Let* $0<\gamma,\ \varepsilon\leqslant 1/2$ *be the accuracy parameters supplied to* $\mathbf{B}_{\mathrm{Filt}}$, *and let* k *be the number of*

iterations chosen by $\mathbf{B}_{\text{Filt}}$. *Let* t_i *denote* $\sum_{r=0}^{i} q_r^i \alpha_r^i$ *and let* $1/2 - \hat{\gamma}_i$ *denote the error of* h_i *with respect to the filtered distribution in the* i*th iteration.*

If for each iteration $0 \leq i \leq k-1$ *we have*

$$\hat{\gamma}_i \geq \gamma\left(1 - \frac{\varepsilon(1-\varepsilon)}{t_i \gamma k}\right), \tag{25}$$

then the error of h_M, *the hypothesis output by* $\mathbf{B}_{\text{Filt}}$, *with respect to the target distribution is smaller than* ε.

Proof. From Lemma 3.7 we immediately get that the increase of the average potential in each iteration is at most $\varepsilon(1-\varepsilon)/k$. Thus the total increase in the average potential in all k iterations is $\varepsilon(1-\varepsilon)$. The rest of the proof follows the same line of argument as the one used for the aborted iterations in the proof of Theorem 3.6. ∎

To illustrate the significance of this result, assume that **WeakLearn** generates a hypothesis whose error is $1/2 - \gamma$ when given examples from the target distribution. Our goal is to achieve a higher degree of accuracy on the target distribution by making use of the performance of **WeakLearn** on other distributions. As we know from the main results of this paper, if **WeakLearn** is capable of generating a hypothesis with error smaller than $1/2 - \gamma$ for *any* distribution then boosting can achieve any desired accuracy on the target distribution. However, by using Lemma 4.1 boosting can be used even in cases where the accuracy of the hypotheses generated by **WeakLearn** decreases as the distributions supplied to it become more and more different from the target distribution. The slower the decrease in accuracy, the higher the quality that can be achieved by boosting.

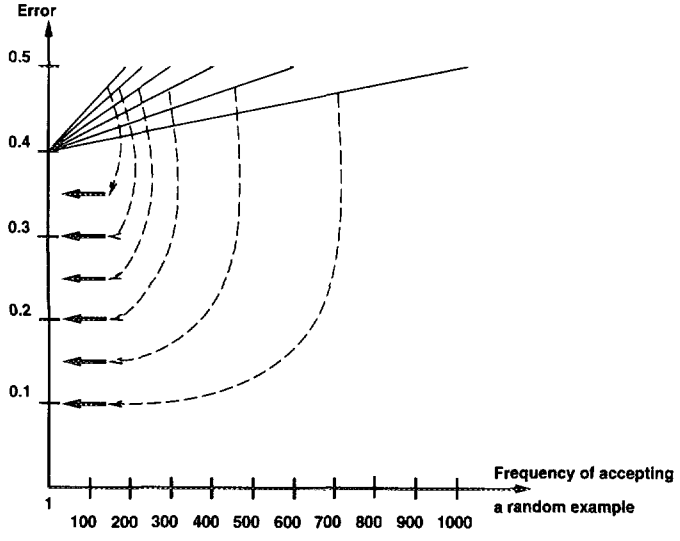

FIG. 5. The accuracy that can be achieved using boosting for a learner whose accuracy depends on the distribution. The horizontal line denotes $1/t$, or the expected number of examples that have to be filtered per accepted example. The origin denotes an acceptance rate of 1, i.e., every example is accepted, which means that the weak learner is observing the original distribution. The vertical axis denotes the error of the hypotheses. Each sloped line denotes a requirement on the maximal error of the weak learner as a function of the divergence from the target distribution. Each such bound guarantees a different accuracy of the final hypothesis, which is described by the bold arrow on the error axis.

We start by simplifying Eq. (25). By choosing $k = (4/\gamma^2)\ln(1/\varepsilon)$, we get an upper bound on the error of h_i as a function of γ and t_i:

$$\hat{\gamma}_i \geq \gamma\left(1 - \frac{\gamma\varepsilon(1-\varepsilon)}{4t_i \ln(1/\varepsilon)}\right).$$

Different choices for γ generate different lower bounds on $\hat{\gamma}_i$ as a function of t_i. An illustration of these lower bounds is given in Fig. 5.

In order to separate the requirements for **WeakLearn** from the particulars of our boosting algorithm, we need to upper bound the value of t_i using a measure of the discrepancy between the target distribution and the filtered distribution. We shall now define such a measure of discrepancy, show that this measure is closely related to the Kullback–Leibler divergence, and give a stronger version of Theorem 3.6 based on this measure.

DEFINITION 1. Let $\mathcal{P}$ and $\mathcal{Q}$ be two distributions defined over the same space X and σ-algebra Σ. The maximal-ratio divergence between $\mathcal{Q}$ and $\mathcal{P}$, denoted $D_M(\mathcal{Q} \| \mathcal{P})$, is defined to be

$$D_M(\mathcal{Q} \| \mathcal{P}) \doteq \ln\left(\sup_{A \in \Sigma,\, \mathcal{Q}(A) > 0} \frac{\mathcal{Q}(A)}{\mathcal{P}(A)}\right).$$

We now lower bound the maximal ratio divergence using the well-known Kullback–Leibler divergence.

LEMMA 4.2. *For any two distributions* $\mathcal{Q}$ *and* $\mathcal{P}$, *defined on the same measure space,*

$$D_M(\mathcal{Q} \| \mathcal{P}) \geq D_{\text{KL}}(\mathcal{Q} \| \mathcal{P}),$$

where $D_{\text{KL}}(\mathcal{Q} \| \mathcal{P})$ *is the Kullback–Leibler divergence, which is defined as*

$$D_{\text{KL}}(\mathcal{Q} \| \mathcal{P}) \doteq E_{x \in \mathcal{Q}}\left(\ln \frac{\mathcal{Q}(x)}{\mathcal{P}(x)}\right).$$

Proof. If $E_{\mathcal{Q}}(\ln(\mathcal{Q}(x)/\mathcal{P}(x))) \geq a$ then there exists a set A such that $\mathcal{Q}(A) > 0$ and $\ln(\mathcal{Q}(A)/\mathcal{P}(A)) \geq a$, which implies that $D_M(\mathcal{Q} \| \mathcal{P}) \geq a$. ∎

Note that there is no similar inequality relating the two measures of divergence in the other way. That is because there might be a set A such that $Q(A)$ is very small, so that the contribution of this set to $D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P})$ is negligible, but on the other hand $\mathcal{Q}(A)/\mathcal{P}(A)$ is extremely large.

Using these measures of divergence, we can lower bound t_i by functions of the divergence between the target distribution and the ith filtered distribution:

LEMMA 4.3. *If $\mathcal{D}$ is the target distribution and F_i is the distribution generated by* **FiltEX** *during the ith iteration, then*

$$t_i \leqslant e^{-D_M(F_i\|D)} \leqslant e^{-D_{\mathrm{KL}}(F_i\|D)}.$$

Proof. The second inequality follows from Lemma 4.2. To prove the first inequality, assume that $D_M(F_i\|\mathcal{D}) \geqslant a$. Then there exists a set $A \in \Sigma$ such that $F_i(A)/\mathcal{D}(A) \geqslant e^a$. Using the definition of the measure generated by filtering in Eq. (3) we get

$$e^a \leqslant \frac{\sum_{r=0}^{i} \mathcal{D}(A \cap X_r^i)\,\alpha_r^i/Z_i}{\sum_{r=0}^{i} \mathcal{D}(A \cap X_r^i)} \leqslant \frac{\sum_{r=0}^{i} \mathcal{D}(A \cap X_r^i)/Z_i}{\sum_{r=0}^{i} \mathcal{D}(A \cap X_r^i)} = \frac{1}{Z_i}.$$

The inequality holds because $\alpha_r^i \leqslant 1$ always.[15] Here $Z_i = \sum_{r=0}^{i} \mathcal{D}(X_r^i)\,\alpha_r^i = t_i$, from which we get $1/t_i \geqslant e^a$, which proves the lemma. ▮

We now combine the results of Lemmas 4.1, 4.2, and 4.3 to arrive at the following stronger version of Theorem 3.6.

THEOREM 4.4. *Fix a target distribution $\mathcal{D}$ and real-valued parameters $\gamma, \varepsilon, \delta > 0$. If* **WeakLearn** *is a learning algorithm that for any distribution $\mathcal{P}$ over the sample space X and any $c \in \mathbf{C}$ generates a hypothesis whose error, with respect to $\mathcal{P}$, is smaller than*

$$\frac{1}{2} - \gamma\left(1 - \frac{\varepsilon(1-\varepsilon)\,\gamma}{4\ln(1/\varepsilon)}\,e^{D_{\mathrm{KL}}(\mathcal{P}\|\mathcal{Q})}\right),$$

then, with probability at least $1-\delta$, the algorithm $\mathbf{B}_{\mathrm{Filt}}$, given the parameters, generates a hypothesis whose error, with respect to $\mathcal{D}$, is smaller than ε.

Proof. The algorithm uses $k = (4/\gamma^2)\ln(1/\varepsilon)$ as given in statement 1 of Algorithm $\mathbf{B}_{\mathrm{Filt}}$ (Fig. 4). From Lemma 4.1 we get that it is enough if the error in the ith iteration is smaller than

$$\frac{1}{2} - \gamma\left(1 - \frac{\varepsilon(1-\varepsilon)\,\gamma}{4t_i\ln(1/\varepsilon)}\right).$$

[15] Note that a tighter bound can be proved using the bound on $\alpha_{\max}^i = \max_{0 \leqslant i \leqslant r} \alpha_r^i$ given in Lemma 3.9. However, here we avoid using this tighter bound because we want the bound to be independent of i.

Combining Lemmas 4.3 and 4.2, we get that $t_i \leqslant e^{-D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{F})}$, which proves the theorem. ▮

Note that Theorem 4.4 assumes that the weak learner is completely reliable, i.e., that it has probability 1 of generating a hypothesis with the desired accuracy. The algorithm can be used for less reliable weak learning algorithms, but there is a subtle point that needs to be addressed in that case. The point is that the number of examples required by $\mathbf{B}_{\mathrm{Rel}}$ in order to increase the reliability is $\Omega(1/\gamma^2)$. Thus if the error of the hypothesis has to be just very slightly smaller than 1/2, the number of examples that are required to test if the hypothesis is good increases without bounds. To avoid this problem the required error has to be set to a smaller value, thus making the detection of a good hypothesis easier. We omit the details of this variant of the boosting algorithm.

4.2. *Boosting Multiple Valued Concepts*

As was noted by Schapire [Sch91], the generalization of the equivalence between strong and weak learning to concepts with more than two labels does not enjoy the same tightness as the two-label case. In the two-label case an ability to predict correctly with probability slightly better than that of random guessing is equivalent to strong learning. In the j-label case the probability that a random guess is correct is equal to $1/j$, while the minimal requirement for weak learning to be equivalent to strong learning is still to predict correctly with a probability slightly better than *one-half*.[16] As any j-valued decision rule can be replaced by $j-1$ binary decision rules of the type "is the label equal to i," the binary boosting algorithm can be used $j-1$ times to generate the desired hypothesis. However, it is possible to perform the boosting process in one pass, generating a simple j-valued hypothesis and eliminating the dependence of the complexity on j. The combination rule that is used is simply the j-valued plurality, i.e., the strong hypothesis labels the input with the label given by the largest number of weak hypotheses. The algorithm and its analysis are almost identical to the binary case; the only difference is that the definition of the filtering factor is based on one more parameter, denoted by t, that is the number of incorrect hypotheses whose output is not equal to the incorrect label with the largest number of votes. For example, suppose the labels are the 10 digits. Assume the correct label for some example is "0" and the incorrect label that got the largest number of votes is "9" (irrespective of whether the number of votes "9" got is larger than the number of votes "0" got). Then t is the number of votes that the

[16] To realize this, consider a 3-label concept such that for any example there are only *two* possible labels (over the whole concept class). In this case, using a random coin flip to choose one of the two possible labels will give a correct answer half of the time, but the concept class might still be unlearnable [Sch91].

digits "1" to "8" got. The change in Formula (1) is that k is replaced by $k-t$:

$$\alpha^i_{r,t} = \begin{cases} 0 & \text{if } r \leqslant i - \dfrac{k-t}{2} \\ \binom{k-t-i-1}{\lfloor (k-t)/2 \rfloor - r} \left(\dfrac{1}{2}+\gamma\right)^{\lfloor (k-t)/2 \rfloor - r} & \\ \quad \times \left(\dfrac{1}{2}-\gamma\right)^{\lceil (k-t)/2 \rceil - i + 1 + r} & \\ & \text{if } i - \dfrac{k-t}{2} < r \leqslant \dfrac{k-t}{2} \\ 0 & \text{if } r > \dfrac{k-t}{2}. \end{cases} \tag{26}$$

It is interesting to note that the resources required are completely independent of j, the number of possible labels. This is even true if j is different for different n and s, or if j is infinite, even uncountable! However, the requirement of weak learning for concepts with uncountable ranges is unreasonably hard. The hypothesis must generate the *exact* correct output for more than half the inputs (in probability). In this case the result described in the next section might be more relevant.

4.3. *Boosting Real Valued Concepts*

A modification of the boosting algorithm can be used for boosting learning algorithms for concept classes whose range is a real number (for a review of algorithms for learning real-valued functions, see Chap. 5 in [Nat91]). This variant of the boosting algorithm transforms learning algorithms that generate hypotheses whose expected error is small (with respect to the input distribution) into algorithms that generate hypotheses whose error is small for most of the input domain.

Assume **C** is a set of functions from R to R and **WeakLearn** is a learning algorithm for **C**. Let p be any density function over R, and let $(x_1, f(x_1))$, $(x_2, f(x_2))$, ..., $(x_n, f(x_n))$ be a set of examples drawn independently at random according to p and labeled according to some $f \in \mathbf{C}$. Then A, upon observing this sample, generates a hypothesis function g such that with probability larger than $1-\delta$,

$$\int_{-\infty}^{+\infty} |f(x) - g(x)|\, dp(x) < d. \tag{27}$$

We shall sketch how the boosting algorithm can be used to generate a function h such that with high probability

$$P_p\left(|f(x) - h(x)| > \frac{d}{1/2-\gamma}\right) < \varepsilon,$$

where P_p is the probability according to the density p and $\gamma, \varepsilon > 0$ are polynomial fractions.

Using the Markov inequality and setting $\Delta = d/(1/2-\gamma)$ we get, from Eq. (27), that

$$P_p(|f(x) - g(x)| > \Delta) < \tfrac{1}{2} - \gamma.$$

We extend the notion of *agreement* between a concept and a hypothesis on an example x to concepts defined on the reals by saying that f and g "Δ-agree" on x if $|f(x) - g(x)| < \Delta$. Using the extended definition of agreement we can say that **WeakLearn** is a weak-learner for the concept class **C**. If we replace all the places in the boosting algorithm in which it refers to "agree" or "correct" by corresponding references to "Δ-agree" or "Δ-agrees with the true function," we get a boosting algorithm for real-valued functions.

Suppose, for simplicity, that we are using algorithm $\mathbf{B}_{\text{Samp}}$. Then the result of running the boosting algorithm over the weak learning algorithm is k real-valued functions $h_1(x), ..., h_k(x)$ such that for any point in the sample more than $k/2$ of the functions are within Δ of the correct value. It is interesting to observe that the results of Theorems 3.2 and 3.10 hold without change for the real-valued case. Thus, by choosing the size of the sample large enough, we are guaranteed that, with probability at least $1-\delta$, more than half of the hypotheses are Δ-correct on all but ε of the points of the *whole domain*.

Observe that if more than half of the functions Δ-agree with f on a point x then the median of the functions Δ-agrees with f. From this we get that the median is the natural generalization of the majority for this case. By taking the median of the k weak hypotheses we get

$$P_p(\text{Median}(h_1, h_2, ..., h_k)\ \Delta\text{-agrees with } f) > 1 - \varepsilon.$$

4.4. *Parallelizing PAC Learning*

The fact that the boosting by filtering algorithm, $\mathbf{B}_{\text{Filt}}$, accepts only a small fraction of the examples with which it is presented has an interesting implication on the possibility of achieving optimal speed-up when parallelizing learning algorithms.

Observe that the time complexity of $\mathbf{B}_{\text{Filt}}$ is dominated by the time that is spent by the procedure **FiltEX** on checking examples that are eventually rejected. Observe also the probability that any given example is accepted during the generation of the ith hypothesis is constant. In other words, it is independent of whether or not any other example is tested or accepted during the ith stage.

Assume now that we use one of the standard parallel-computation paradigms, such as the PRAM model, and

that we have a computer with p processors at our disposal. Then we can parallelize the procedure **FiltEX** in the following way. Each of the p processors runs the procedure **FiltEX** independently, each making separate calls to **EX**, so that they test different random examples.[17] When one of the p processors accepts an example, all the other processors are halted and their results are ignored.[18] The accepted example is then returned to **WeakLearn** as usual. Recall that out of the $O(1/\varepsilon(\ln 1/\varepsilon)^{3/2})$ examples that are needed for learning only $O(\ln 1/\varepsilon)$ examples have to be accepted and returned to **WeakLearn**. If the number of processors is $O(1/\varepsilon \sqrt{\ln 1/\varepsilon})$ then the search for an acceptable example takes expected constant time, so that the expected running time of the boosting algorithm becomes $O(\ln 1/\varepsilon)$. If p is smaller, then a p-fold speedup over the serial execution is achieved. We summarize this observation in the following theorem.

THEOREM 4.5. *If* **C** *is a polynomially PAC-learnable concept class then there exists a parallel learning algorithm for* **C** *that runs on a PRAM machine with* $O(1/\varepsilon)$ *processors whose time complexity dependence on the accuracy is* $O(\log 1/\varepsilon)$.

5. SUMMARY AND OPEN PROBLEMS

The algorithms we have described in this paper give the best upper bounds currently known on the resources required for polynomial PAC learning. While these bounds are in some respects close to optimal, further improvement might still be possible in the dependence of the sample and time complexity on the parameters ε and γ.

One undesired property of our boosting algorithm is that it requires prior knowledge of a distribution-independent bound on the accuracy of the hypotheses that **WeakLearn** generates. While guessing a bound is a theoretically feasible solution, it is expensive in practical applications [Dru93]. Recently, Freund and Schapire [FS95] have developed a boosting algorithm which does not require such prior knowledge. The number of weak hypotheses that need to be combined to reach a given level of accuracy is almost as small as the number achieved here.

A deeper problem is that the assumption of distribution-independent bounds for learning algorithms often seems to be unreasonable. The results in [FS95] and Theorem 4.4 are encouraging in this respect because they show that boosting can be achieved even without uniform bounds. This might be a sign that a richer and maybe more realistic theory of learning can be developed in which performance bounds are distribution dependent.

In this paper we have shown that the boosting algorithm can be generalized to multiple-valued concept classes as well as real-valued concept classes. However, the results regarding real-valued concept classes are still rather weak, and one would hope that stronger types of boosting can be achieved in that context. The use of boosting in the context of p-concepts [KS90] is another long-standing open problem. Some progress on the problem of boosting in the context of independent label noise has been achieved in a recent work by Aslam and Decatur [AD93] about boosting learning algorithms in the statistical query model introduced by Kearns [Kea93].

Last but not least, boosting has been successfully applied to some practical machine learning problems [DSS93]. Further experimentation with boosting methods will hopefully achieve even better results. Such experiments are also important for discovering interesting new problems for theoretical research.

APPENDIX A: SUMMARY OF NOTATION

A.1. *Concept Learning Notation*

The sample space is denoted X, the concept class is denoted **C**, and the class of hypotheses is denoted H. Typical elements of these spaces are denoted x, c, and h respectively. The distribution over X, according to which examples are generated, is denoted by $\mathcal{D}$. We denote by $S = \{(x_1, c(x_1)), \ldots, (x_m, c(x_m))\}$ a sample of m examples, labeled according to $c \in \mathbf{C}$. The accuracy parameter is denoted ε, and the reliability parameter is denoted δ. The sample, time, and space required for the learning algorithm under discussion to achieve accuracy ε with reliability δ are denoted $m(\varepsilon, \delta)$, $s(\varepsilon, \delta)$, and $t(\varepsilon, \delta)$ respectively.

A.2. *Notation for Describing Boosting*

We denote a generic weak learning algorithm by **WeakLearn**. We use ε_0 and δ_0 to denote the accuracy and the reliability of **WeakLearn**. Usually ε_0 is close to $1/2$ (the accuracy of a random guess) and δ_0 is close to 1 (probability zero of generating an ε_0-accurate hypothesis). For this reason we define $\gamma = 1/2 - \varepsilon_0$ and $\lambda = 1 - \delta_0$. The number of examples, time and space required by the weak learner to achieve its fixed goals are denoted m_0, t_0, and s_0 respectively. We denote the hypothesis generated by the boosting algorithm by h_M, and the set of all such hypotheses by H_M.

[17] We either assume that the running time of **EX** is negligible or that **EX** can generate many examples at the same time.

[18] We assume that halting all processors can be done in unit time.

A.3. *Meaning of Common Notation in Different Sections*

symbol	Meaning in Majority-Vote Game	Meaning in analysis of $\mathbf{B}_{\text{Samp}}$	Meaning in analysis of $\mathbf{B}_{\text{Filt}}$
k	The total number of iterations in the game.	The total number of weak hypotheses combined by the boosting algorithm.	
$i = 0 \ldots k$	The number of iterations played so far.	The number of weak hypotheses generated so far.	
$r = 0 \ldots i$	The number of marks.	The number of weak hypotheses that are correct	
X_r^i	The points that have been marked r times in the first i iterations	The points in the sample on which r out of the first i weak hypotheses are correct	The points in X on which r out of the first i weak hypotheses are correct
$V(A)$	The value of the set A	The number of sample points in A	The probability of A according to the distribution $\mathcal{D}$
$W_i(A)$	The weight of the set A in the ith iteration	The sum of the weights assigned to the sample points in A using hypotheses $h_1, \ldots, h_{i-1}$	The probability of A according to the distribution filtered using hypotheses $h_1, \ldots, h_{i-1}$
α_r^i	The weight assigned to points in X_r^i defined in Equation 1		$\alpha_r^i/\alpha_{\max}^i$ is the probability of accepting an example from X_r^i during the ith iteration, where $\alpha_{\max}^i = \max_{0 \le r \le i} \alpha_r^i$
β_r^i	The potential of the points in X_r^i, defined in Equation 6		
$q_r^i = V(X_r^i)$	The value of X_r^i	The number of sample points in X_r^i	The probability of X_r^i according to the distribution $\mathcal{D}$
$x_r^i = \frac{V(X_r^i \cap X_{r+1}^{i+1})}{V(X_r^i)}$	The fraction, in terms of value) of X_r^i that is marked in the ith iteration	The fraction of the points of X_r^i on which h_{i+1} is correct	The fraction (in terms of the distribution $\mathcal{D}$) of X_r^i on which h_{i+1} is correct
L The loss set	The set of points marked less than $k/2$ times in the k iterations	The sample points on which the majority vote is incorrect i.e. the empty set	The set of points in X on which the majority vote is incorrect, i.e. the set of points on which h_M is incorrect.

A.4. *Special Notation*

- $t_i = \sum_{r=0}^{i} q_r^i \alpha_r^i$, the expected weight of a random example in the ith iteration. This notation is used in the analysis of $\mathbf{B}_{\text{Filt}}$ because $t_i/\alpha_{\max}^i$ is the probability of accepting a random example during the ith iteration.
- $\hat{\gamma}_i$, the actual edge of the ith weak hypothesis, h_i. In other words, the error of h_i, with respect to the ith filtered distribution, is $1/2 - \hat{\gamma}_i$.
- m_R, denotes the number of examples required by $\mathbf{B}_{\text{Rel}}$ for generating a weak hypothesis with the desired reliability.

APPENDIX B: BOOSTING THE RELIABILITY OF A LEARNING ALGORITHM

We present the boosting algorithm, $\mathbf{B}_{\text{Rel}}$, in Fig. 6 and prove its performance.

Proof of Lemma 3.5. The bound on the number of examples is immediate from the definition of the algorithm. To prove that the algorithm is correct, we bound the probability that the resulting hypothesis has error larger than $1/2 - \gamma/2$. There are two events that might cause this. The first is that all of the r hypotheses generated by **WeakLearn**

Algorithm $\mathbf{B}_{\text{Rel}}$
Input: **EX**, **WeakLearn**, γ, λ, δ
Output: A hypothesis h_M, that has error smaller than $1/2 - \gamma/2$ with probability at least $1 - \delta$.

1. Call **WeakLearn** $r = \frac{\ln(2/\delta)}{\lambda}$ times, each time on a different set of random examples. Store the resulting hypotheses as $h_1, \ldots, h_r$.
2. Count the number of mistakes made by each of the r hypotheses on a random sample of size $m = (8/\gamma^2)\ln(2r/\delta)$.
3. Return the hypothesis that makes the smallest number of mistakes on the sample.

FIG. 6. A description of the algorithm for boosting the reliability of an algorithm.

have error larger than $1/2-\gamma$. The second is that a hypothesis that has error larger than $1/2-\gamma/2$ makes fewer mistakes, on the test sample, than a hypothesis that has error smaller than $1/2-\gamma$. It is easy to bound the probability of each of those events by $\delta/2$. This proves the lemma.

As we know that each call to **WeakLearn** has probability of at least λ of generating a hypothesis with error smaller than $1/2-\gamma$ at each trial, the probability of not generating an accurate enough hypothesis is at most

$$(1-\lambda)^r=(1-\lambda)^{1/\lambda \ln(2/\delta)}\leqslant e^{-\ln(2/\delta)}=\delta/2.$$

In order for the second event to happen, given that one of the hypotheses has error smaller than $1/2-\gamma$, there has to be a bad hypothesis whose estimated error is larger than that of the good hypothesis. For this to happen, the gap between the actual error and the estimated error for at least one of the r hypotheses has to be at least $\gamma/4$. Using Hoeffding bounds we get that this probability is at most

$$\begin{aligned} re^{-2m(\gamma/4)^2} &= r\exp(-2(8/\gamma^2)\ln(2r/\delta)(\gamma/4)^2) \\ &= re^{-\ln(2r/\delta)}=\delta/2, \end{aligned}$$

which proves the lemma. ∎

APPENDIX C: PROOF OF LEMMA 2.5

First observe that as we are interested only in the ratio of the weight and value of G to those of D, we can assume without loss of generality that $V(D)=W(D)=1$.

Define the following series of partitions of D.

- $\mathscr{P}_0=\{D\}$.
- $\mathscr{P}_1=\{D_0^1, D_1^1\}$ where the sets are disjoint and $V(D_0^1)=V(D_1^1)=\frac{1}{2}$.
- Construct $\mathscr{P}_{i+1}$ from $\mathscr{P}_i$ by splitting each set in $\mathscr{P}_i$ into two disjoint equal valued parts so that the value of each part is exactly 2^{-i}.

We shall now use the partitions $\mathscr{P}_0, \mathscr{P}_1, \mathscr{P}_2, \ldots$ to construct a series of sets $G_0, G_1, G_2, \ldots$ that will provide better and better approximations of the target set G. Assume that the binary expansion of $1/2+\gamma$ is

$$\tfrac{1}{2}+\gamma=\sum_{j=0}^{\infty} b_j 2^{-j}$$

(note that $b_0=0, b_1=1$) and construct the sets G_i according to the following inductive procedure:

$$G_0=\varnothing$$

$$\Delta_i=\text{the set with the largest weight in } \mathscr{P}_i \text{ that is not a subset of } G_i$$

$$G_{i+1}=\begin{cases}\text{if } b_i=0, & G_i \\ \text{if } b_i=1, & G_i\cup\Delta_i\end{cases}$$

It is clear that G_i is a monotonically increasing series of sets and that $\lim_{i\to\infty} V(G_i)=1/2+\gamma$.

Note that all of the parts in $\mathscr{P}$ have equal value. Thus the ratio between the value of the complement of G_i and the value of Δ_i is equal to the number of parts that are outside G_i. On the other hand, the weight of the set Δ_i is the largest among the parts outside G_i. Thus the ratio of the weight of the complement of G_i to the weight of Δ_i is *at least* the number of parts. This fact can be written as follows:

$$\frac{W(\Delta_i)}{1-W(G_i)}\geqslant\frac{V(\Delta_i)}{1-V(G_i)}.$$

We shall now prove by induction on i that $\forall i\geqslant 0$, $W(G_i)\geqslant V(G_i)$.

- For $i=0$, $G_0=\varnothing$ so the claim holds trivially.
- For $i\geqslant 1$, if $b_i=0$ then $G_{i+1}=G_i$ so the induction holds trivially. Else, $b_i=1$ and thus $G_{i+1}=G_i\cup\Delta_i$, and we obtain

$$\begin{aligned} W(G_{i+1}) &= W(G_i)+W(\Delta_i) \\ &= V(G_i)+(W(G_i)-V(G_i))+W(\Delta_i) \\ &\geqslant V(G_i)+(W(G_i)-V(G_i)) \\ &\quad + V(\Delta_i)\frac{1-V(G_i)-(W(G_i)-V(G_i))}{1-V(G_i)} \\ &= V(G_i)+V(\Delta_i) \\ &\quad +(W(G_i)-V(G_i))\left[1-\frac{V(\Delta_i)}{1-V(G_i)}\right]. \end{aligned}$$

The first two terms sum to $V(G_{i+1})$, and the last term is positive because $\Delta_i\subset\bar{G}_i$ implies that $V(\Delta_i)\leqslant 1-V(G_i)$ and because from the induction hypothesis $W(G_i)-V(G_i)\geqslant 0$. The induction hypothesis is thus proved.

Define $G=\bigcup_{j=1}^{\infty} G_i$. As all Δ_i are in the σ-algebra Σ then so is G. Also, $V(G)=\lim_{i\to\infty} V(G_i)=1/2+\gamma$. Similarly, $W(G)=\lim_{i\to\infty} W(G_i)\geqslant 1/2+\gamma$ and because for all i we have $W(G_i)\geqslant V(G_i)$ we also get an inequality at the limit $W(G)\geqslant V(G)=1/2+\gamma$, which proves the lemma. ∎

APPENDIX D: PROOF OF LEMMA 3.10

In order to prove the lemma, we use the following technical lemma:

LEMMA D.1. *For any real numbers $x\geqslant 1$ and $0\leqslant\mu<1/2$,*

$$\exp\left(-\frac{1}{3(1-4\mu^2)x}\right)<\frac{\binom{x}{x(1/2-\mu)}}{\sqrt{2/\pi x}\,e^{xH(1/2-\mu)}}<\frac{\exp(1/12x)}{\sqrt{1-4\mu^2}}, \tag{28}$$

where $H(y) = -y \ln y - (1-y)\ln(1-y)$ *is the entropy function, and the extension of the binomial function to the reals is based on the extension of the factorial to the gamma function* $x! = \Gamma(x+1)$.

Proof. The proof of this lemma is based on the Stirling approximation. Note that as $x \to \infty$ the lower bound converges to 1 while the upper bound converges to $(1-4\mu^2)^{-1/2}$. In other words, for large values of x the binomial $\binom{x}{x(1/2-\mu)}$ is related to the exponential function in the denominator by a small factor.

The stirling approximation to the factorial can be written in the following way:[19]

$$\forall x \geq 1, \qquad x \ln x - x + \frac{\ln x}{2} + \ln\sqrt{2\pi} < \ln(x!) < x \ln x - x + \frac{\ln x}{2} + \ln\sqrt{2\pi} + \frac{1}{12x}.$$

From this we obtain the lower bound as follows:

$$\begin{aligned}
&\ln\binom{x}{x(1/2-\mu)} \\
&\quad = \ln x! - \ln((1/2-\mu)x)! - \ln((1/2+\mu)x)! \\
&\quad > x\ln x - x + \frac{\ln x}{2} + \ln\sqrt{2\pi} \\
&\qquad - x(1/2-\mu)\ln(x(1/2-\mu)) + x(1/2-\mu) \\
&\qquad - \frac{\ln(x(1/2-\mu))}{2} - \ln\sqrt{2\pi} - \frac{1}{12(1/2-\mu)x} \\
&\qquad - x(1/2+\mu)\ln(x(1/2+\mu)) + x(1/2+\mu) \\
&\qquad - \frac{\ln(x(1/2+\mu))}{2} - \ln\sqrt{2\pi} - \frac{1}{12(1/2+\mu)x} \\
&\quad = xH(1/2-\mu) - \ln\sqrt{x} - \ln\sqrt{\frac{1}{4}-\mu^2} \\
&\qquad - \ln\sqrt{2\pi} - \frac{1}{12(1/4-\mu^2)x} \\
&\quad \geq xH(1/2-\mu) - \ln\sqrt{2\pi x} + \ln 2 - \frac{1}{3(1-4\mu^2)x}.
\end{aligned}$$

And we obtain the upper bound as follows:

$$\begin{aligned}
&\ln\binom{x}{x(1/2-\mu)} \\
&\quad = \ln x! - \ln((1/2-\mu)x)! - \ln((1/2+\mu)x)! \\
&\quad < x\ln x - x + \frac{\ln x}{2} + \ln\sqrt{2\pi} + \frac{1}{12x} \\
&\qquad - x(1/2-\mu)\ln(x(1/2-\mu)) + x(1/2-\mu) \\
&\qquad - \frac{\ln(x(1/2-\mu))}{2} - \ln\sqrt{2\pi} \\
&\qquad - x(1/2+\mu)\ln(x(1/2+\mu)) + x(1/2+\mu) \\
&\qquad - \frac{\ln(x(1/2+\mu))}{2} - \ln\sqrt{2\pi} \\
&\quad = xH(1/2-\mu) - \ln\sqrt{x} - \ln\sqrt{\frac{1}{4}-\mu^2} - \ln\sqrt{2\pi} + \frac{1}{12x} \\
&\quad = xH(1/2-\mu) - \ln\sqrt{2\pi x} + \ln 2 \\
&\qquad - \ln\sqrt{1-4\mu^2} + \frac{1}{12x}. \quad \blacksquare
\end{aligned}$$

[19] See, for example, Eq. (9.91) in [GKP91].

Proof of Lemma 3.9. We can rewrite the definition of α_r^i from Fig. 4 as follows (ignoring the choices of r that give $\alpha_r^i = 0$ for the purpose of the upper bound):

$$\alpha_i^r = \binom{x}{(1/2-\mu)x}\left(\frac{1}{2}-\frac{\gamma}{2}\right)^{x(1/2-\mu)}\left(\frac{1}{2}+\frac{\gamma}{2}\right)^{x(1/2+\mu)},$$

where $x = k-i-1$ and $\mu = 1/2 - (\lfloor k/2 \rfloor - r)/(k-i-1)$. Using the upper bound given in Lemma D.1, bound the last expression for any value of μ

$$\begin{aligned}
&\binom{x}{(1/2-\mu)x}\left(\frac{1}{2}-\frac{\gamma}{2}\right)^{x(1/2-\mu)}\left(\frac{1}{2}+\frac{\gamma}{2}\right)^{x(1/2+\mu)} \\
&\quad < \sqrt{\frac{2}{\pi x}}\frac{e^{1/12x}}{\sqrt{1-\gamma^2}}\exp\left(xH\left(\frac{1}{2}-\mu\right)\right. \\
&\qquad \left. + x\left(\frac{1}{2}-\mu\right)\ln\left(\frac{1}{2}-\frac{\gamma}{2}\right) + x\left(\frac{1}{2}+\mu\right)\ln\left(\frac{1}{2}+\frac{\gamma}{2}\right)\right).
\end{aligned}$$

But a basic inequality is that for any $-1/2 \leq \mu \leq 1/2$,

$$H\left(\frac{1}{2}-\mu\right) \leq -\left(\frac{1}{2}-\mu\right)\ln\left(\frac{1}{2}-\frac{\gamma}{2}\right) - \left(\frac{1}{2}+\mu\right)\ln\left(\frac{1}{2}+\frac{\gamma}{2}\right).$$

This inequality is strict unless $\mu = \gamma/2$. From this we get that

$$\begin{aligned}
&\binom{x}{(1/2-\mu)x}\left(\frac{1}{2}-\frac{\gamma}{2}\right)^{x(1/2-\mu)}\left(\frac{1}{2}+\frac{\gamma}{2}\right)^{x(1/2+\mu)} \\
&\quad < \sqrt{\frac{2}{\pi x}}\frac{e^{1/12x}}{\sqrt{1-\gamma^2}}.
\end{aligned}$$

As $\gamma \leq 1/2$ and $x \geq 1$, we obtain the statement of the lemma. $\blacksquare$

ACKNOWLEDGMENTS

I thank Robert Schapire for his many contributions to this paper, which include the use of boosting for compression, the implication of boosting on circuit complexity and the definition of a general boosting algorithm. Most of the work described in this paper was done while I was a student of the University of California at Santa Cruz. I thank my teachers there, Manfred Warmuth, David Haussler, and David Helmbold, for their help in writing this paper. I also thank Eli Shamir for the observation of the implication of boosting on learning in parallel. Finally, I thank the anonymous referees for their many valuable comments.

Received September 23, 1993; final manuscript received February 6, 1995

REFERENCES

[AD93] Aslam, J. A., and Decatur, S. E. (1993), General bounds on statistical query learning and PAC learning with noise via hypothesis boosting, *in* "Proceedings, 35th Annual IEEE Symposium on the Foundations of Computer Science, Nov."

[BEHW87] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1987), Occam's razor, *Inform. Process Lett.* **24**, 377–380, Apr.

[BEHW89] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989), Learnability and the Vapnik–Chervonenkis dimension, *J. Assoc. Comput. Mach.* **36**(4), 929–965.

[Dru93] Drucker, H. (1992–1993), Private correspondence.

[DSS93] Drucker, H., Schapire, R., and Simard, P. (1993), Improving performance in neural networks using a boosting algorithm, *in* "Advances in Neural Informations Processing Systems 5," pp. 42–49, Kaufmann, San Mateo, CA.

[FS95] Freund, Y., and Schapire, R. E. (1995), A decision-theoretic generalization of on-line learning and an application to boosting, *in* "eurocolt95."

[FW93] Floyd, S., and Warmuth, M. (1993), "Sample Compressions, Learnability, and the Vapnik–Chervonenkis Dimension," Tech. Rep. UCSC-CRL-93-13, Department of Computer and Information Sciences, University of California, Santa Cruz.

[GHR92] Goldmann, M., Hastad, L., and Razborov, A. (1992), Majority gates vs general weighted threshold gates, *Comput. Complexity* **2**.

[GKP91] Graham, R. L., Knuth, D. E., and Patashnik, O. (1991), "Concrete Mathematics, a Foundation for Computer Science," Addison–Wesley, Reading, MA.

[HKLW91] Haussler, D., Kearns, M., Littlestone, N., and Warmuth, M. K. (1991), Equivalence of models for polynomial learnability, *Inform. and Comput.* **95**(2), 129–161, Dec.

[HLW88] Haussler, D., Littlestone, N., and Warmuth, M. K. (1988), Predicting $\{0, 1\}$ functions on randomly drawn points, *in* "Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science," pp. 100–109, IEEE Computer Soc. Press, New York.

[Kea93] Kearns, M. (1993), Efficient noise-tolerant learning from statistical queries, *in* "Proceedings, 25th Annual ACM Symposium on the Theory of Computing," pp. 392–401, ACM Press, New York.

[KS90] Kearns, M. J., and Schapire, R. E. (1990), Efficient distribution-free learning of probabilistic concepts, *in* "Proceedings of the 31st Symposium on the Foundations of Computer Science," pp. 382–391. IEEE Computer Soc. Press, Los Alamitos, CA.

[KV88] Kearns, M., and Valiant, L. G. (1988), "Learning Boolean Formulae or Finite Automata Is as Hard as Factoring," Tech. Rep. TR-14-88, Aiken Computation Laboratory, Harvard University, Cambridge, MA.

[KV94] Kearns, M., and Valiant, L. G. (1994), Cryptographic limitations on learning boolean formulae and finite automata, *J. Assoc. Comput. Mach.* **41**(1), 67–95.

[LW86] Littlestone, N., and Warmuth, M. (1986), "Relating Data Compression and Learnability;" this early and hard-to-locate work is referenced and partly rewritten in [FW93].

[Nat91] Natarajan, B. K. (1991), "Machine Learning: A Theoretical Approach," Kaufmann, San Mateo, CA.

[Sch90] Schapire, R. E. (1990), The strength of weak learnability, *Mach. Learning* **5**(2), 197–227.

[Sch91] Schapire, R. E. (1991), "The Design and Analysis of Efficient Learning Algorithms," Ph.D. thesis. M.I.T.

[Sch92] Schapire, R. E. (1992), Private correspondence, Jan.

[Sha92] Shamir, E. (1992), Private correspondence.

[Val94] Valiant, L. G. (1984), A theory of the learnable, *Commun. ACM* **27**(11), 1134–1142, Nov.