Polynomial time Prediction Strategy with almost Optimal Mistake Probability

Nader H. Bshouty *
Department of Computer Science
Technion, 32000
Haifa, Israel
bshouty@cs.technion.ac.il

Abstract

We give the first *polynomial time* prediction strategy for any PAC-learnable class C that probabilistically predicts the target with mistake probability

$$\frac{poly(\log(t))}{t} = \tilde{O}\left(\frac{1}{t}\right)$$

where t is the number of trials. The lower bound for the mistake probability is [HLW94]

$$\Omega\left(\frac{1}{t}\right),$$

so our algorithm is almost optimal. [1]

1 Introduction

In the Probabilistic Prediction model [HLW94] a teacher chooses a boolean function $f : X \to \{0,1\}$ from some class of functions C and a distribution D on X. At trial t the learner receives from the teacher a point x_t chosen from X according to the distribution D and is asked to predict $f(x_t)$. The learner uses some strategy S (algorithm), predicts $S(x_t)$ and sends it to the teacher. The teacher then answers "correct" if the prediction is correct, i.e. if $S(x_t) = f(x_t)$ and answers "mistake" otherwise. The goal of the learner is to run in polynomial time at each trial (polynomial in $\log t$ and some measures of the class and the target) minimize the worst case (over all $f \in C$ and D) probability of mistake in predicting $f(x_t)$.

Haussler et. al. in [HLW94] gave a double exponential time prediction strategy (exponential in the number of trials t) that achieves mistake probability $V_C/t = O(1/t)$ where V_C is

*This research was supported by the fund for promotion of research at the Technion. Research no. 120-025.

[1] The lower bound proved in [HLW94] is $\omega(V_C/t)$ where V_C is the VC-dimension of the class C. In our case V_C is constant with respect to t

the VC-dimension of the class C. They also show a lower bound of $\Omega(V_C/t)$ for the mistake probability. They then gave an exponential time algorithm (polynomial in t) that achieves mistakes probability $(V_C/t)\log(t/V_C) = O(\log t/t)$ assuming that C is PAC-learnable in polynomial time. Since learning in the probabilistic model implies learning in the PAC model, the requirement that C is efficiently PAC-learnable is necessary for efficient probabilistic prediction. The results from [BG02] gives a randomized strategy that achieves mistake probability exponentially small with the number of mistakes for prediction all $f(x_i)$. It is not clear what is the error of this algorithm as a function of number of trials t.

In this paper we give a deterministic prediction strategy. we show that if C is PAC-learnable then there is deterministic prediction strategy that runs in polynomial time and achieve mistake probability at most

$$\frac{poly(\log t)}{t} = \tilde{O}\left(\frac{1}{t}\right).$$

The first part of the paper gives a deterministic PAExact-learning algorithm for any PAC-learnable class that achieves exponentially small error in the number of equivalence queries. In the second part we show how to turn this algorithm to a deterministic prediction strategy that achieves the required mistake probability. While a randomized version of the first part was known in [BG02], it wasn't clear how to achieve the second part. The contribution of the paper is in building a deterministic PAExact-learning algorithm and in turning this algorithm to almost optimal strategy.

In section 2 and 3 we build a new deterministic booster for the PAExact-model and then in section 4 we show how to change the PAExact-learning algorithm to a prediction strategy that achieves the above bound.

2 Learning Models and Definitions

Let C be a class of functions $f : X \to \{0,1\}$. The domain X can be finite, countable infinite, or $\mathcal{R}^n$ for some $n \geq 1$. In learning, a *teacher* has a *target function* $f \in C$ and a *probability distribution* D on X. The *learner* knows C but does not know the probability distribution D nor the function f.

The *problem size* I_f that we will use in this paper depends on X, C and f and it can be different in different settings. The term "polynomial" means polynomial in the problem size I_f. For example, for Boolean functions with $X = \{0,1\}^n$, C is a set of formulas (e.g. DNF, Decision tree, etc.). The problem size is $I_f = n + size_C(f)$ where $size_C(f)$ is the minimal size of a formula in C that is equivalent to f. Then "polynomial" means $poly(I_f) = poly(n, size_C(f))$. For infinite domains X the parameter n is usually replaced by the VC-dimension of the class V_C and $I_f = V_C + size_C(f)$. Then "polynomial" in this case is $poly(V_C, size_C(f))$.

The learner can ask the teacher *queries* about the target. The teacher can be regarded as an adversary with unlimited computational power that must answer honestly but also wants to fail the learner from learning quickly. The queries we consider in this paper are:

Example Query according to D ($\mathbf{Ex}_D$) [V84] For the example query the teacher chooses

$x \in X$ according to the probability distribution D and returns $(x, f(x))$ to the learner.

We say that the hypothesis h_r *ε-approximates* f with respect to distribution D if

$$E_r[\Pr_D[f(x) \neq h_r(x)]] \leq \varepsilon$$

where here and elsewhere $\Pr_D$ denotes $\Pr_{x \in_D X}$.

The learning models we will consider in this paper are

PAC (Probably Approximately Correct)[V84] In the PAC learning model we say that an algorithm $\mathcal{A}$ of the learner *PAC-learns* the class C if for any $f \in C$, any probability distribution D and any $\varepsilon, \delta > 0$ the algorithm $\mathcal{A}(\varepsilon, \delta)$ asks example queries according to D, Ex_D, and with probability at least $1 - \delta$, outputs a polynomial size circuit h that ε-approximates f with respect to D. That is $\Pr_D[X_{f \Delta h}] \leq \varepsilon$. We say that C is *PAC-learnable* if there is an algorithm that PAC-learns C in time $poly(1/\varepsilon, \log(1/\delta), I_f)$.

In the online learning model [L88] the teacher at each trial sends a point $x \in X$ to the learner and the learner has to predict $f(x)$. The learner returns to the teacher the prediction y. If $f(x) \neq y$ then the teacher returns "mistake" to the learner. The goal of the learner is to minimize the number of prediction mistakes.

Online [L88] In the online model we say that algorithm $\mathcal{A}$ of the learner *Online-learns* the class C if for any $f \in C$ and for any δ, algorithm $\mathcal{A}(\delta)$ with probability at least $1 - \delta$ makes bounded number of mistakes. We say that C is *Online-learnable* if the number of mistakes and the running time of the learner for each prediction is $poly(\log(1/\delta), I_f)$.

Probabilistic Prediction (PP) [HLW94] In the Probabilistic Prediction model the points sent to the learner are chosen from X according to some distribution D. We say an algorithm $\mathcal{A}$ of the learner *η-PP-learns* the class C if for any $f \in C$ and for any δ the algorithm $\mathcal{A}(\delta)$ with probability at least $1 - \delta$ after bounded number of mistakes can predict the answer with probability greater than $1 - \eta$. We say that C is *η-PP-learnable* if the number of mistakes and the running time of the learner at each trial is $poly(\log(1/\delta), I_f)$.

3 The New Algorithm

In this section we give our new booster and prove its correctness. In Subsection 3.1 we show how to start from a hypothesis that approximates the target function f and refine it to get a better one. In Subsection 3.2 we give the main algorithm and prove its correctness.

3.1 Refining The Hypothesis

We will first give a booster that takes a hypothesis that η-approximates the target and builds a new hypothesis that $\eta/2$-approximates the target.

Let $\mathcal{A}$ be a PAC-learning algorithm that learns the class C in polynomial time from $m_{\mathcal{A}}(\varepsilon, \delta, I_f)$ examples. Let h_0 be a hypothesis such that

$$\Pr_D[f \neq h_0] \leq \eta. \tag{1}$$

Our booster learns a sequence of hypotheses $\mathcal{H} = h_1, h_2, h_3, \ldots, h_k$ and then uses this sequence to build the refined hypothesis.

We start with the following notation. Let

$$H_j^{\wedge} = \begin{cases} 1 & j = 1 \\ h_1 \wedge \cdots \wedge h_{j-1} & j > 1 \end{cases} \text{ and } H_j^{\vee} = \begin{cases} 0 & j = 1 \\ h_1 \vee \cdots \vee h_{j-1} & j > 1 \end{cases}.$$

Let

$$H_j = \overline{h_0} H_j^{\wedge} \vee h_0 H_j^{\vee} \text{ and } G_j = \overline{h_0} H_j^{\wedge} \vee h_0 \overline{H_j^{\vee}}.$$

Notice that

$$H_j = \begin{cases} H_j^{\wedge} & h_0 = 0 \\ H_j^{\vee} & h_0 = 1 \end{cases} \text{ and } G_j = \begin{cases} H_j^{\wedge} & h_0 = 0 \\ \overline{H_j^{\vee}} & h_0 = 1 \end{cases}. \tag{2}$$

Now we show how the booster learns h_j from $h_1, h_2, \ldots, h_{j-1}$. The booster runs the procedure **Learnh**(j, ε, δ). See Figure 1. This procedure either returns a refined hypothesis h (see steps 10 and 11 in **Learnh**) or returns the next hypothesis h_j in the sequence $\mathcal{H}$ (see step 14 in **Learnh**). In the former case h_j =NULL indicating that h_{j-1} is the last function in the sequence $\mathcal{H}$ and then $\mathcal{H} = h_1, h_2, \ldots, h_k$ for $k = j - 1$. In the latter case a new function h_j is generated in $\mathcal{H}$. We will show that for some $\varepsilon = 1/poly(\log(1/\eta))$ and $k = poly(\log(1/\eta))$, either h_0 or H_k or H_{k+1} (this depends where the algorithm returns in the last call for **Learnh**. In step 10, 11 or 14, respectively) is an $\eta/2$-approximation of f.

Before we give a formal proof, we give some intuition. The hypotheses that the booster generates are used to build a branching program $B_{\mathcal{H}}$. This branching program is described in Figure 2.

The branching program $B_{\mathcal{H}}$ has internal nodes labeled with the hypothesis h_i and unlabeled leaves. The labels $z_{i,j}$ and $z'_{i,j}$ are names assigned to the internal nodes and leaves, respectively. Each $x \in X$ corresponds to a *computation path* in $B_{\mathcal{H}}$. The computation path start from the root z_0 and for each node z that it arrives to that is labeled with h, it follows the edge that is labeled with 0 if $h(x) = 0$ and follows the edge that is labeled with 1 otherwise. Each computation path ends in some leaf. For each node z we define X_z the set of points in X that arrive to z. That is, its computational path contains the node z. Define

$$X_z^{(\xi)} = \{x \in X_z \mid f(x) = \xi\}.$$

It is easy to see that

P1. If we label the leaves of $B_{\mathcal{H}}$ in the following way we get H_{j+1}: For $i = 2, \ldots, j+1$, assign the label 0 to $z'_{i,0}$ and $z_{j+1,1}$ and 1 to $z'_{i,1}$ and $z_{j+1,0}$.

P2. If we label the leaves of $B_{\mathcal{H}}$ in the following way we get h_0: For $i = 2, \ldots, j+1$, assign the label 0 to $z'_{i,0}$ and $z_{j+1,0}$ and 1 to $z'_{i,1}$ and $z_{j+1,1}$.

P3. We have $G_{j+1}(x) = 1$ if and only if the computation path of x ends at $z_{j+1,0}$ or $z_{j+1,1}$.

Learnh (j, ε, δ)

1) **Set** $m \leftarrow m_{\mathcal{A}}(\varepsilon/2, \delta, I_f)$; $r_0 \leftarrow 0$; $r_j \leftarrow 0$.
2) **For** $i \leftarrow 1$ to m
3) **Flip** a fair coin →result
4) **If** result="Head" **Then**
5) **Repeat** $\mathrm{EQ}_D(h_0) \rightarrow (x_i, f(x_i))$; $r_0 \leftarrow r_0 + 1$;
6) **Until** $G_j(x_i) = 1$ or $r_0 = 4m$.
7) **Else**
8) **Repeat** $\mathrm{EQ}_D(H_j) \rightarrow (x_i, f(x_i))$; $r_j \leftarrow r_j + 1$;
9) **Until** $G_j(x_i) = 1$ or $r_j = 4m$.
10) **If** $r_0 = 4m$ **Then Return**($h_j \equiv$NULL, $h \equiv h_0$)
11) **If** $r_j = 4m$ **Then Return**($h_j \equiv$NULL, $h \equiv H_j$)
12) $S \leftarrow S \cup \{(x_i, f(x_i))\}$.
13) **Run** $\mathcal{A}$ with examples $S \rightarrow h_j$
14) **Return**$(h_j, h \equiv$NULL)

Figure 1: *The algorithm* **Learnh**(j, ε, δ) *learns the* j*th function in the sequence* $\mathcal{H}$.

P4. For points that satisfy $G_{j+1}(x) = 1$ we have $H_{j+1}(x) = \overline{h_0(x)}$.

To get to the intuition we promised, we added another figure.

In Figure 3 for each node z we assigned two values $(p_z^{(0)}, p_z^{(1)})$. Our goal will be to show that with high probability,

$$p_z^{(0)} \geq \Pr_D[X_z^{(0)}] \text{ and } p_z^{(1)} \geq \Pr_D[X_z^{(1)}]. \tag{3}$$

Notice now that when we label a leaf z in $B_{\mathcal{H}}$ with constant $\xi \in \{0, 1\}$ it will contribute error $\Pr_D[X_z^{\bar{\xi}}] \leq p_z^{(\bar{\xi})}$ to the final hypothesis. Therefore, assuming (3) is true, if we label $z'_{i,0}$ with 0, $z'_{i,1}$ with 1, $z_{j+1,0}$ with 1 and $z_{j+1,1}$ with 0, we get error at most $2(j\varepsilon\eta + \varepsilon^j)$. By property (P1) this labeling gives H_{j+1}. Therefore [2], if (3) is true, then

$$\Pr_D[f \neq H_{j+1}] \leq 2(j\varepsilon\eta + \varepsilon^j).$$

Now, for appropriate $1/\varepsilon$ and j that are polynomial in $\log(1/\eta)$ this error is less than $\eta/2$.

The question remains: How to guarantee such pair of values $(p_z^{(0)}, p_z^{(1)})$? After learning $h_1, \ldots, h_{j-1}$, the algorithm tries to learn the target function over the domain of all points that

[2] Our proof uses slightly different approach and achieve error $j\varepsilon\eta + \varepsilon^j$.

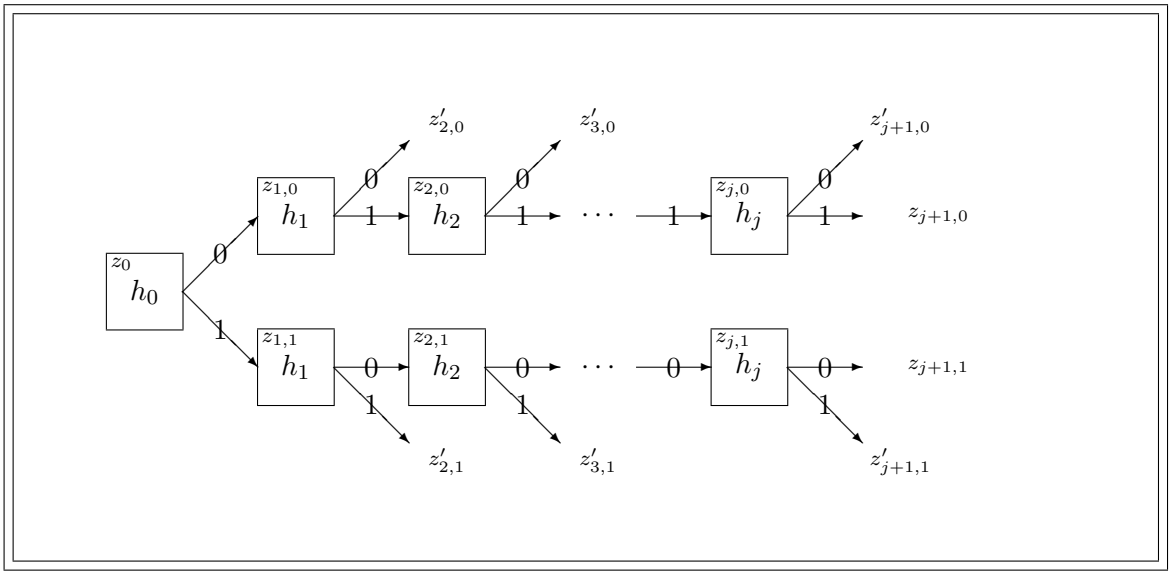

Figure 2: *The branching program $B_{\mathcal{H}}$ that* **Learnh** *builds*

arrives at nodes $z_{j,0}$ and $z_{j,1}$ with the same distribution D projected on those points and that is normalized to give equal weight (weight 1/2) to the set of such points x with $h_0(x) = 1$ and the set of such points with $h_0(x) = 0$. As we mentioned in property (P3), points that arrives at nodes $z_{j,0}$ and $z_{j,1}$ are exactly the points that satisfies $G_j(x) = 1$. Therefore **Learnh** accepts only counterexamples that satisfy $G_j(x) = 1$ (see steps 6 and 9). To achieve equal probability distribution weight for points with $h_0(x) = 0$ and $h_0(x) = 1$ the algorithm with probability 1/2 asks $\mathrm{EQ}_D(H_j)$, which is by property (P4) equivalent to $\mathrm{EQ}(\overline{h_0})$ for points x that satisfy $G_j(x) = 1$, and with probability 1/2, asks $\mathrm{EQ}_D(h_0)$. If the algorithm succeeds to learn h_j, then this hypothesis (as we will show in the proof) will refine the bounds $(p^{(0)}_{z_{j,0}}, p^{(1)}_{z_{j,0}})$ to $(\varepsilon p^{(0)}_{z_{j,0}}, p^{(1)}_{z_{j,0}})$ and $(p^{(0)}_{z_{j,0}}, \varepsilon p^{(1)}_{z_{j,0}})$ in the children nodes.

One final question remains to be answered: What happen if the booster cannot receive enough points that ends in $z_{j,0}$ and $z_{j,1}$, i.e., points that satisfy $G_j(x) = 1$? This happens when all the counterexamples comes from the other leaves. In this case we know that the weight of the points that arrive at leaves $z_{j,0}$ and $z_{j,1}$ is less than the weight of the counterexamples in the other leaves which is at most $2j\varepsilon\eta$. Then the total error of this hypothesis, whether it is H_j in step 8 or h_0 in step 5, is at most $4j\varepsilon\eta$ which is again less than $\eta/2$ for $j, 1/\varepsilon = poly(1/\eta)$.

So either we learn h_{j+1} and then H_{j+1} achieves error at most $2(j\varepsilon\eta + \varepsilon^j)$ or no more examples can be achieved and then H_j or h_0 (depends in which step the algorithm cannot get more examples) achieves error at most $4j\varepsilon\eta$. For $1/\varepsilon, j = poly(1/\eta)$ the error in both cases is less than $\eta/2$.

We now formalize all the above with a slight modification. For the analysis of the algorithm

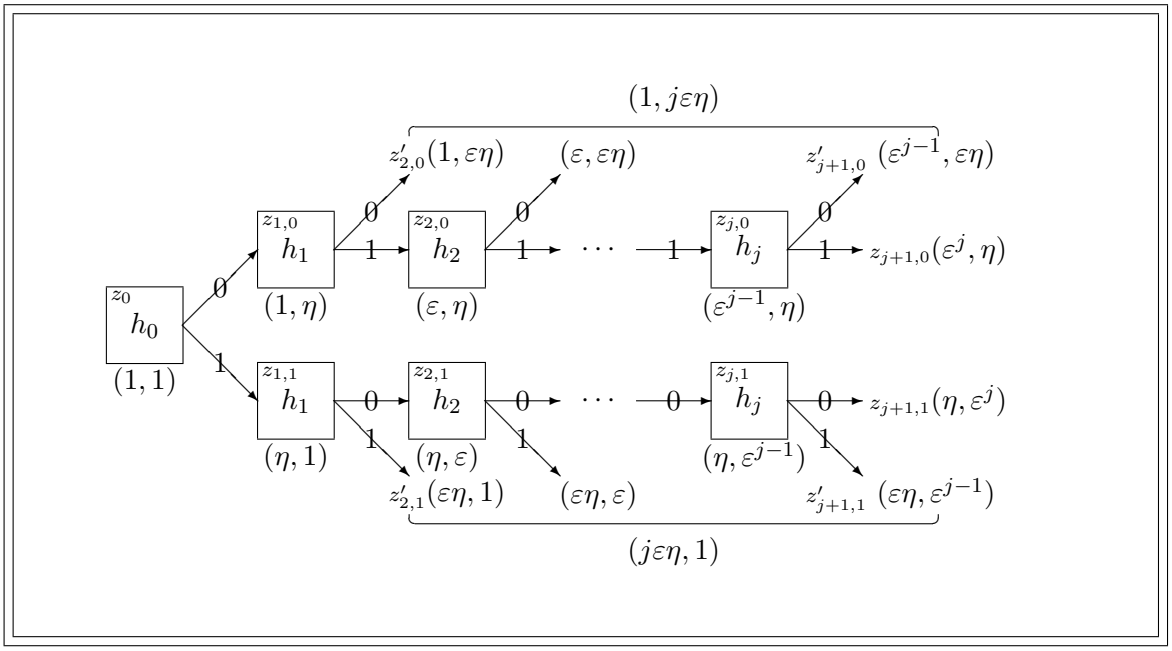

Figure 3: *For every node z we have a pair $(p_z^{(0)}, p_z^{(1)})$ where $p_z^{(0)}$ (respectively, $p_z^{(1)}$) is the weight of the set of positive (respectively, negative) points that arrives at node z.*

we define three values: For $j \leq k$

$$\begin{aligned} w_j &= \Pr_D[\overline{h_0} H_j^{\wedge} \overline{h_j} = 1, f = 1] + \Pr_D[h_0 \overline{H_j^{\vee}} h_j = 1, f = 0] \\ u_j &= \Pr_D[\overline{h_0} H_{j+1}^{\wedge} = 1, f = 0] + \Pr_D[h_0 \overline{H_{j+1}^{\vee}} = 1, f = 1] \\ v_j &= \Pr_D[\overline{h_0}\ \overline{H_{j+1}^{\wedge}} = 1, f = 1] + \Pr_D[h_0 H_{j+1}^{\vee} = 1, f = 0] \end{aligned}$$

Notice that

$$w_j = \Pr_D\left[X^{(1)}_{z'_{j+1,0}} \cup X^{(0)}_{z'_{j+1,1}}\right], u_j = \Pr_D\left[X^{(0)}_{z_{j+1,0}} \cup X^{(1)}_{z_{j+1,1}}\right]$$

and

$$v_j = \Pr_D\left[\bigcup_{i=2}^{j+1} X^{(1)}_{z'_{i,0}} \cup \bigcup_{i=2}^{j+1} X^{(0)}_{z'_{i,1}}\right].$$

We start by proving the following

Property 1 *We have*

1. $u_0 \leq 1$.
2. $v_j = \sum_{i=1}^{j} w_i$.

3. $\Pr_D[f \neq H_{j+1}] = u_j + v_j$.

Proof. We have $u_0 = \Pr_D[f = h_0] < 1$ which follows *1.* Now

$$\begin{aligned} v_j &= \Pr_D[\overline{h_0}\ \overline{H^{\wedge}_{j+1}} = 1, f = 1] + \Pr_D[h_0 H^{\vee}_{j+1} = 1, f = 0] \\ &= \sum_{i=1}^{j} (\Pr_D[\overline{h_0} H_i^{\wedge} \overline{h_i} = 1, f = 1] + \Pr[h_0 \overline{H_i^{\vee}} h_i = 1, f = 0]) \\ &= \sum_{i=1}^{j} w_i. \end{aligned}$$

This follows *2.* Finally we have: By (2)

$$\begin{aligned} \Pr_D[f \neq H_{j+1}] &= \Pr_D[f \neq H_{j+1}, h_0 = 0, f = 0] + \Pr_D[f \neq H_{j+1}, h_0 = 1, f = 1] + \\ &\quad \Pr_D[f \neq H_{j+1}, h_0 = 0, f = 1] + \Pr_D[f \neq H_{j+1}, h_0 = 1, f = 0] \\ &= \Pr_D[\overline{h_0} H^{\wedge}_{j+1} = 1, f = 0] + \Pr_D[h_0 \overline{H^{\vee}_{j+1}} = 1, f = 1] + \\ &\quad \Pr_D[\overline{h_0}\ \overline{H^{\wedge}_{j+1}} = 1, f = 1] + \Pr_D[h_0 H^{\vee}_{j+1} = 1, f = 0] \\ &= u_j + v_j. \end{aligned}$$

and this follows *3.*□

We now prove the following Claims

Claim 3.1 *For every j, with probability at least $1 - \delta$ we have*

$$w_j \leq \varepsilon\eta \text{ and } u_j \leq \varepsilon u_{j-1}.$$

Claim 3.2 *With probability at least $1 - j\delta$ we have: For all $i \leq j$,*

$$w_i \leq \varepsilon\eta,\ u_i \leq \varepsilon^i,\ v_i \leq i\varepsilon\eta$$

and

$$\Pr_D[f \neq H_{i+1}] \leq i\varepsilon\eta + \varepsilon^i.$$

Claim 3.3 *If $\Pr_D[f \neq h_0] > 2(j-1)\varepsilon\eta$ then the probability that* **Learnh**(j, ε, δ) *returns $h \equiv h_0$ is less than $j\delta$.*

Claim 3.4 *If $\Pr_D[f \neq H_j] > 2(j-1)\varepsilon\eta$ then the probability that* **Learnh**(j, ε, δ) *returns $h \equiv H_j$ is less than $j\delta$.*

The first and the second Claim give bounds for w_i, u_i and v_i and show that with high probability the error of the hypothesis H_{j+1} is less than $j\varepsilon\eta + \varepsilon^j$. The other two claims show that if the algorithm stops in steps 10 or 11 then with high probability the hypothesis h_0 or

H_j, respectively, achieves error at most $2(j-1)\varepsilon\eta$. In the next subsection we will choose j and ε such that those errors are less than $\eta/2$.

We now prove the claims

Proof of Claim 3.1. When **Learnh** learns h_j it asks with probability 1/2, $\mathrm{EQ}_D(h_0)$ and with probability 1/2, $\mathrm{EQ}_D(H_j)$ and takes only points x_i that satisfies $G_j(x_i) = 1$ (see steps 5-6 and 8-9 in **Learnh**). Let D_j be the probability distribution of x_i. Since the events $f \neq h_0, G_j = 1$ and $f \neq H_j, G_j = 1$ are disjoint (take two cases $f = 0$ and $f = 1$ and use (2) or see property P4) and since the algorithm takes $m_{\mathcal{A}}(\varepsilon/2, \delta, I_f)$ examples to learn h_j, with probability at least $1 - \delta$ we have

$$\begin{aligned}
\frac{\varepsilon}{2} &\geq \Pr_{D_j}[f \neq h_j] \\
&= \frac{1}{2}\Pr_{D}[f \neq h_j | f \neq h_0, G_j = 1] + \frac{1}{2}\Pr_{D}[f \neq h_j | f \neq H_j, G_j = 1] \qquad (4)
\end{aligned}$$

By (1) (2) and (4) we have

$$\begin{aligned}
\varepsilon &\geq \Pr_{D}[f \neq h_j | f \neq h_0, G_j = 1] \\
&= \frac{\Pr_D[f \neq h_j, f \neq h_0, G_j = 1]}{\Pr_D[f \neq h_0, G_j = 1]} \\
&\geq \frac{\Pr_D[\overline{h_0} H_j^{\wedge} \overline{h_j} = 1, f = 1] + \Pr_D[h_0 \overline{H_j^{\vee}} h_j = 1, f = 0]}{\eta} = \frac{w_j}{\eta}.
\end{aligned}$$

Therefore $w_j \leq \varepsilon\eta$.

By (4) and (2) we have

$$\begin{aligned}
\varepsilon &\geq \Pr_{D}[f \neq h_j | f \neq H_j, G_j = 1] \\
&= \frac{\Pr_D[f \neq h_j, f \neq H_j, G_j = 1]}{\Pr_D[f \neq H_j, G_j = 1]} \\
&= \frac{\Pr_D[\overline{h_0} H_{j+1}^{\wedge} = 1, f = 0] + \Pr_D[h_0 \overline{H_{j+1}^{\vee}} = 1, f = 1]}{\Pr_D[\overline{h_0} H_j^{\wedge} = 1, f = 0] + \Pr_D[h_0 \overline{H_j^{\vee}} = 1, f = 1]} = \frac{u_j}{u_{j-1}}.
\end{aligned}$$

Therefore $u_j \leq \varepsilon u_{j-1}$. □

Now the proof of Claim 3.2 follows from Property 1 and Claim 3.1.

Proof of Claim 3.3. We have

$$\begin{aligned}
\Pr_{D}[G_j = 0 | f \neq h_0] &= \frac{\Pr_D[G_j = 0, f \neq h_0]}{\Pr_D[f \neq h_0]} \\
&= \frac{\Pr_D[\overline{h_0}\ \overline{H_j^{\wedge}} = 1, f = 1] + \Pr_D[h_0 H_j^{\vee} = 1, f = 0]}{\Pr_D[f \neq h_0]} \\
&\leq \frac{v_{j-1}}{2(j-1)\varepsilon\eta}
\end{aligned}$$

By Claim 3.2, with probability at least $1-(j-1)\delta$

$$\Pr_D[G_j = 0 | f \neq h_0] \leq \frac{1}{2}.$$

Suppose **Learnh** calls the equivalence query $\mathrm{EQ}_D(h_0)$, $4m$ times. Let X_r be a random variable that is equal to 1 if the rth call of $\mathrm{EQ}_D(h_0)$ returns a counterexample x' such that $G_j(x') = 0$ and $X_r = 0$ otherwise. Then

$$E[X_r] = \Pr_D[G_j = 0 | f \neq h_0] \leq \frac{1}{2}.$$

If **Learnh**(j, ε, δ) outputs h_0 then since the algorithm makes at most m coin flips (see steps 2-3 in **Learnh**) we have

$$\sum_{i=1}^{4m} X_i \geq 3m.$$

Now given that $\{X_r\}_r$ are independent random variables and $E[X_r] \leq 1/2$ and using Chernoff bound we have

$$\begin{aligned}
\Pr[\textbf{Learnh}(j, \varepsilon, \delta) \text{ outputs } h_0] &= \Pr\left[\sum_{i=1}^{4m} X_i > 3m\right] \\
&\leq \Pr\left[\frac{\sum_{i=1}^{4m} X_i}{4m} \geq E[X_r] + \frac{1}{4}\right] \\
&\leq e^{-(m/4)} \leq \delta
\end{aligned}$$

The later inequality follows because $m \geq 4\ln(1/\delta)$. Therefore, the probability that **Learnh**(j, ε, δ) outputs h_0 is at most $j\delta$.□

Proof of Claim 3.4: We have

$$\begin{aligned}
\Pr_D[G_j = 0 | f \neq H_j] &= \frac{\Pr_D[G_j = 0, f \neq H_j]}{\Pr_D[f \neq H_j]} \\
&= \frac{\Pr_D[h_0 H_j^{\vee} = 1, f = 0] + \Pr_D[\overline{h_0}\ \overline{H_j^{\wedge}} = 1, f = 1]}{\Pr_D[f \neq H_j]} \\
&\leq \frac{v_{j-1}}{2(j-1)\varepsilon\eta}.
\end{aligned}$$

Then the proof is exactly the same as the proof of Claim 3.3.□

We now can build the procedure that refines the function h_0. In Figure 4 the procedure **Refine** runs **Learnh** at most k times. It stops running **Learnh** and output a refined hypothesis if one of the following happen:

1. The function h_j is equal to NULL and then it outputs either h_0 or H_j (depends what is h).
2. We get $j = k$ and then it outputs H_{k+1}.

Refine $(h_0, k, \varepsilon, \delta)$

1) $j \leftarrow 0$
2) **Repeat**
3) $\quad j \leftarrow j + 1$
4) $\quad$ **Learnh**$(j, \varepsilon, \delta/(3k^2)) \rightarrow (h_j, h)$
5) **Until** $j = k$ or h_j =NULL
6) **If** $j = k$ **Then** $h \leftarrow H_{k+1}$
7) **Return**(h).

Figure 4: *A PAExact-learning algorithm that refine* h_0

We now prove

Lemma 1 *Suppose* $\Pr_D[f \neq h_0] \leq \eta$ *and* $h =$ **Refine**$(h_0, k, \varepsilon, \delta)$. *Then with probability at least* $1 - \delta$ *we have*

$$\Pr_D[f \neq h] \leq \max(k\varepsilon\eta + \varepsilon^k, 2k\varepsilon\eta).$$

Proof. Let $\mathcal{H} = h_1, h_2, \ldots, h_t$, $t \leq k$ be the sequence of hypotheses generated by **Learnh**. Let $\delta' = \delta/(3k^2)$. We want to measure the probability that the algorithm fails to output a hypothesis h that η'-approximates f where $\eta' = \max(k\varepsilon\eta + \varepsilon^k, 2k\varepsilon\eta)$. This happen if and only if one of the following events happen:

$[A_1]$ For some $j = t \leq k$, **Learnh**$(j, \varepsilon, \delta')$ outputs $h \equiv h_0$ and $\Pr_D[f \neq h_0] \geq \eta'$.

$[A_2]$ For some $j = t \leq k$, **Learnh**$(j, \varepsilon, \delta')$ outputs $h \equiv H_j$ and $\Pr_D[f \neq H_j] \geq \eta'$.

$[A_3]$ We have $t = k$ and $\Pr_D[f \neq H_{k+1}] \geq \eta'$.

Now since for $j = 1, \ldots, k$ we have

$$2(j-1)\varepsilon\eta \leq 2k\varepsilon\eta \leq \eta',$$

by Claim 3.3

$$\begin{aligned} \Pr[A_1] &\leq \Pr[\exists\ 1 \leq j \leq k :\ \textbf{Learnh}(j, \varepsilon, \delta') \text{ outputs } h \equiv h_0 \text{ and } \Pr_D[f \neq h_0] \geq 2(j-1)\varepsilon\eta] \\ &\leq \sum_{j=1}^{k} j\delta' \leq k^2\delta' = \frac{\delta}{3} \end{aligned}$$

In the same way one can prove

$$\Pr[A_2] \leq \frac{\delta}{3}.$$

PAExact-Learn (η, δ)

1) **Set** $\varepsilon \leftarrow \frac{\log\log\frac{1}{\eta}}{16\log\frac{1}{\eta}}$, $\delta' \leftarrow \frac{\delta}{6\log\frac{1}{\eta}}$, $k \leftarrow \left\lceil \frac{2\log\frac{1}{\eta}}{\log\log\frac{1}{\eta}} \right\rceil$.
2) **Run** $\mathcal{A}$ with $m_{\mathcal{A}}(\varepsilon, \delta', I_f)$ examples $\rightarrow h_0$
3) **For** $t \leftarrow 1$ **to** $\left\lceil \log\frac{1}{\eta} \right\rceil$
4) $\quad h_0 \leftarrow$ **Refine**$(h_0, k, \varepsilon, \delta')$
5) **Output**(h_0)

Figure 5: *An PAExact-learning algorithm that learns the class C with error η and confidence δ.*

Now since

$$k\varepsilon\eta + \varepsilon^k \leq \eta',$$

by Claim 3.2

$$\begin{aligned} \Pr[A_3] &\leq \Pr[\Pr_D[f \neq H_{k+1}] > k\varepsilon\eta + \varepsilon^k] \\ &\leq k\delta' \leq \frac{\delta}{3}. \end{aligned}$$

Therefore, the probability that the algorithm fails to output a hypothesis that η' approximates f is less than δ.□

3.2 The Algorithm and its Analysis

We are now ready to give the PAExact-learning algorithm. We will first give the algorithm and prove its correctness. Then we give the analysis of the algorithm's complexity.

Let $\mathcal{A}$ be a PAC-learning algorithm that learns C in polynomial time and $m_{\mathcal{A}}(\varepsilon, \delta, I_f)$ examples. In Figure 5, the algorithm **PAExact-Learn**(η, δ) defines

$$\varepsilon = \frac{\log\log\frac{1}{\eta}}{16\log\frac{1}{\eta}},\ \delta' = \frac{\delta}{6\log\frac{1}{\eta}} \text{ and } k = \left\lceil \frac{2\log\frac{1}{\eta}}{\log\log\frac{1}{\eta}} \right\rceil. \tag{5}$$

The algorithm first runs $\mathcal{A}$ to get some hypothesis h_0. Then it runs **Refine** $\lceil \log(1/\eta) \rceil$ times. We will prove the following

Theorem 1 *(***Correctness***) Algorithm* **PAExact-Learn**(η, δ) *learns with probability at least* $1 - \delta$ *a hypothesis that* η*-approximates* f*.*

Proof. Let $h_0^{(0)}, h_0^{(1)}, \ldots, h_0^{(t)}$, $t = \lceil \log(1/\eta) \rceil$ be the functions learned in line 4 of the algorithm. Here $h_0^{(0)} = h_0$ is the hypothesis that is learned in line 2 of the algorithm. We have with probability at least $1 - \delta'$

$$\Pr[f \neq h_0^{(0)}] \leq \varepsilon$$

and by Lemma 1 with probability at least $1 - \delta'$ we have

$$\Pr[f \neq h_0^{(k)}] \leq \max(k\varepsilon\eta_0 + \varepsilon^k, 2k\varepsilon\eta_0)$$

where $\Pr_D[f \neq h_0^{(k-1)}] = \eta_0$. Now since

$$k\varepsilon\eta_0 + \varepsilon^k \leq \frac{\eta_0 + \eta}{4}$$

and

$$2k\varepsilon\eta_0 \leq \frac{\eta_0}{2}$$

and since

$$\max\left(\frac{\eta_0 + \eta}{4}, \frac{\eta_0}{2}\right) \leq \max\left(\frac{\eta_0}{2}, \eta\right),$$

we have

$$\Pr_D[f \neq h_0^{(k)}] \leq \max\left(\frac{\Pr_D[f \neq h_0^{(k-1)}]}{2}, \eta\right).$$

Therefore, with probability at least $1 - \delta$ we have

$$\Pr_D[f \neq h_0^{\lceil \log(1/\eta) \rceil}] \leq \eta.$$

This completes the proof of the Theorem.□

For the analysis of the algorithm we first give a very general Theorem and then apply it to different settings.

Theorem 2 *(***Efficiency***) Algorithm* **PAExact_Learn**(η, δ) *uses*

$$\frac{16 \log^2 \frac{1}{\eta}}{\log \log \frac{1}{\eta}} m_{\mathcal{A}}\left(\frac{\log \log \frac{1}{\eta}}{32 \log \frac{1}{\eta}}, \frac{\left(\log \log \frac{1}{\eta}\right)^2 \delta}{72 \log^3 \frac{1}{\eta}}, I_f\right)$$

equivalence queries.

Proof. We will use the notations in (5). Algorithm **PAExact_Learn**(η, δ) calls the procedure **Refine**$(h_0, k, \varepsilon, \delta')$, $\lceil \log(1/\eta) \rceil$ times. The procedure **Refine** $(h_0, k, \varepsilon, \delta')$ calls the procedure **Learnh** $(j, \varepsilon, \delta'/3k^2)$, k times and the procedure **Learnh**$(j, \varepsilon, \delta'/3k^2)$ calls the example oracle at most $8m_{\mathcal{A}}(\varepsilon/2, \delta'/(3k^2), I_f)$ times. This follows the result.□

It follows from Theorem 2

Theorem 3 *If C is PAC-learnable with error $\frac{1}{2}-\gamma$ and confidence λ with sample of size V_0, then C is PAExact-learnable with*

$$d(\delta,\eta) = O\left(w\log\frac{1}{\delta} + \frac{V_0 w}{\gamma^2}\log^2\frac{V_0 w}{\gamma^2}\right),$$

equivalence queries where

$$w = \frac{32\log^3\frac{1}{\eta}}{\left(\log\log\frac{1}{\eta}\right)^2}$$

and time polynomial in d and $1/\lambda$. In particular, if X is countable and $m_D \neq 0$ then C is PExact-learnable with $d(\delta, m_D)$ equivalence queries and time polynomial in d and $1/\lambda$.

Proof. We use Corollary 3.3 in [F95]. It shows that if C is PAC-learnable with error $\frac{1}{2}-\gamma$ and confidence λ with sample of size V_0 then it is PAC-learnable with error ε and confidence $1-\delta$ with sample of size

$$m_{\mathcal{A}}(\varepsilon,\delta,I_f) = O\left(\frac{1}{\varepsilon}\log\frac{1}{\delta} + \frac{V_0}{\varepsilon\gamma^2}\log^2\frac{V_0}{\varepsilon\gamma^2}\right).$$

Let

$$\varepsilon = \frac{\log\log\frac{1}{\eta}}{32\log\frac{1}{\eta}} \text{ and } \delta' = \frac{\left(\log\log\frac{1}{\eta}\right)^2\delta}{72\log^3\frac{1}{\eta}}.$$

Then by Theorem 2, C is PAExact-learnable with

$$\begin{aligned}
\frac{16\log^2\frac{1}{\eta}}{\log\log\frac{1}{\eta}}c_1\left(\frac{1}{\varepsilon}\log\frac{1}{\delta'} + \frac{V_0}{\varepsilon\gamma^2}\log^2\frac{V_0}{\varepsilon\gamma^2}\right) &= c_2\left(w\log\frac{1}{\delta'} + \frac{V_0 w}{\gamma^2}\log^2\frac{V_0}{\varepsilon\gamma^2}\right)\\
&\leq c_2\left(w\log\frac{1}{\delta'} + \frac{V_0 w}{\gamma^2}\log^2\frac{V_0 w}{\gamma^2}\right)\\
&\leq c_3\left(w\log\frac{1}{\delta} + \frac{V_0 w}{\gamma^2}\log^2\frac{V_0 w}{\gamma^2}\right)
\end{aligned}$$

equivalence queries, for some constants c_1, c_2 and c_3.□

Before we leave this section we give the algorithm that will be used in the next section. We will use $d(\delta,\eta)$ to denote the complexity of the algorithm **PAExact-Learn**(η,δ).

We now prove

Theorem 4 *There is an algorithm* **PAExact-Learn**(δ) *that after d mistakes will be holding a hypothesis that with probability at least $1-\delta$ has error $\eta(d)$ where*

$$d = 2d(\delta,\eta(d)).$$

Proof. Set a constant d_0. We run PAExact-Learn$(\delta,\eta(d_0))$ and after d_0 mistakes we run PAExact-Learn$(\delta,\eta(2d_0))$ and so on. When $d = d_0 + 2d_0 + \cdots + 2^i d_0 \leq 2^{i+1}d_0$ with probability at least $1-\delta$ the final hypothesis has error η where $2^i d_0 = d(\delta,\eta)$. This gives $d = 2d(\delta,\eta(d))$.□

Predict $(x_t, S = ((h_0, t_0), \ldots, (h_d, t_d)))$

1) **Initial** $h_0 = 1; t_0 = 0; S = ((h_0, t_0))$.
2) **Let** $\ell = \arg\min t_i$.
3) **Find** η_0 and η_1 in
   $d = 2d(\eta_0, \eta_0)$ and
   $\frac{t}{d} = \frac{V^\star}{\eta_1} \log \frac{1}{\eta_1} + \frac{1}{\eta_1} \log \frac{t^3}{\eta_1}$
4) **If** $\eta_1 < \eta_0$ **Predict** $h_\ell(x_t)$ **Else Predict** $h_d(x_t)$.
5) **Receive** $f(x_t)$
6) **If** $h_d(x_t) \neq f(x_t)$ **Then**
7) **Send** x_t to **PAExact-Learn** (δ) and
8) **Receive** a new hypothesis h_{d+1}
9) **Add** $S \leftarrow (S, (h_{d+1}, 0))$.
10) **Else**
11) $t_d = t_d + 1$.
12) **Goto** 2.

Figure 6: *A deterministic prediction strategy.*

4 A Prediction Strategy and its Analysis

In this section we use the algorithm **PAExact-Learn** (η, δ) to give a deterministic prediction strategy. Then give an analysis of its mistake probability.

First we may assume that t is known. This is because we may run our prediction strategy assuming $t = t_0$ and get a prediction strategy with mistake probability $\epsilon(t_0)$. If $t > t_0$ then at trials $t_0 + 1, t_0 + 2, \ldots, 3t_0$ we use the prediction strategy used in trial t_0 and at the same time learn a new prediction strategy (from the last $2t_0$ examples) that has mistake probability $\epsilon(2t_0)$. It is to see that this doubling technique will solve the problem when t is not known.

Second we may assume that t is large enough. When t is polynomial in the other parameters then we can use the PAC-learning algorithm to learn a hypothesis and use this hypothesis for the prediction. This hypothesis will achieve error $\log t/t$.

We also need a bound on the VC-dimension of the class of all possible output hypotheses of **PAExact-Learn** (η, δ). Obviously this cannot be more than the number of examples we use in PAExact algorithm. We denote this by $V^\star$.

The strategy prediction algorithm is described in Figure 6. The procedure saves the hypotheses $h_1, h_2, \ldots$ generated from **PAExact-Learn**(δ) and for each hypothesis h_i it saves d_i the number of examples x_j in which h_i predicted correctly. Notice that the algorithm in line 4 does not necessarily chooses the last hypothesis h_d for the prediction. In some cases, (depends on η_1 and η_2) it chooses the hypothesis that is consistent with the longest sequence

of consecutive examples (see line 2-4 in the algorithm). This choice of hypothesis, which is probably not the best one, will help us to find a simple proof for our main result.

The idea of the proof is very simple. If the number of mistakes d is "large" then the probability of the mistake of the final hypothesis is small. Otherwise, (if d is small) then there is a hypothesis that is consistent with t/d consecutive examples and then this hypothesis will have a small prediction error.

We prove the following Theorem

Theorem 5 *The probability of the prediction error of the strategy* **Predict** *is smaller than*

$$\frac{poly(\log(t))}{t}.$$

Proof Sketch. If $\eta_0 < \eta_1$ then $d = 2d(\eta_0, \eta_0)$ and by Theorem **??** the hypothesis h_d is with probability $1-\eta_0$ has error η_0. Therefore h_d will predict x_{t+1} with mistake probability at most $2\eta_0$.

If $\eta_1 \leq \eta_0$ then since h_ℓ is consistent on at least

$$\frac{t}{d} = \frac{V^\star}{\eta_1}\log\frac{1}{\eta_1} + \frac{1}{\eta_1}\log\frac{t}{\eta_1}$$

consecutive examples, then by OCCAM, with probability at least $1-\eta_1$ the hypothesis h_ℓ has error η_1. Therefore it will predict $f(x_{t+1})$ with probability mistake at most $2\eta_1$. This implies that the probability of the prediction mistake at trial t is at most $\eta = 2\min(\eta_1, \eta_2)$.

Since t is fixed we can consider η_0 and η_1 as functions of d. The error η_0 is monotonically decreasing as a function of d and η_1 is monotonically increasing as a function of d. Therefore, $\eta \leq 2\eta'$ where $\eta' = \eta_1 = \eta_2$. Replacing η_1 and η_2 by η' we get

$$d = O\left(\omega\log\frac{1}{\eta'} + \frac{V_C}{\gamma^2}\log^2\frac{V_C\omega}{\gamma^2}\right), \quad \omega = \frac{32\log^3\frac{1}{\eta'}}{\left(\log\log\frac{1}{\eta_0}\right)^2}$$

and

$$\frac{t}{d} = \frac{V^\star}{\eta'}\log\frac{1}{\eta'} + \frac{1}{\eta'}\log\frac{t^3}{\eta'}.$$

Then

$$\begin{aligned} t &= d\left(\frac{V^\star}{\eta'}\log\frac{1}{\eta'} + \frac{1}{\eta'}\log\frac{t}{\eta'}\right) \\ &= \frac{poly\left(\log\frac{1}{\eta'}\right)poly(\log t)}{\eta'} \end{aligned}$$

Which implies

$$\eta \leq 2\eta' = \frac{poly(\log t)}{t}.\square$$

References

[A88] D. Angluin. newblock Queries and concept learning. *Machine Learning*, 2:319-342, 1987.

[B94] A. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain, *SIAM Journal on Computing 23(5)*,pp. 990-1000,1994.

[B97] N. H. Bshouty, Exact learning of formulas in parallel. *Machine Learning, 26,*pp. 25-41,1997.

[BC+96] N. H. Bshouty, R. Cleve, R. Gavald, S. Kannan, C. Tamon, Oracles and Queries That Are Sufficient for Exact Learning. *Journal of Computer and System Sciences* **52**(3): pp. 421-433 (1996).

[BG02] N. H. Bshouty and D. Gavinsky, PAC=PAExact and other equivalent models in learning Proceedings of the 43rd Ann. Symp. on Foundation of Computer Science. (FOCS) 2002.

[BJT02] N. H. Bshouty, J. Jackson and C. Tamon, Exploring learnability between exact and PAC, Proceedings of the 15th Annual Conference on Computational Learning Theory, 2002.

[F95] Y. Freund, Boosting a weak learning algorithm by majority, *Information and Computation*, **121**, 256-285 (1995).

[HLW94] D. Haussler, N. Littlestone and M. K. Warmuth, Predicting 0,1-functions on randomly drawn points, *Information and Computation, 115,*pp. 248-292,1994.

[KM96] M. Kearns and Y. Mansour, On the Boosting Ability of Top-Down Decision Tree Learning Algorithms, Proceedings of the 28th Symposium on Theory of Computing, pp. 459-468,1996.

[L88] N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[MA00] Y. Mansour and D. McAllester, Boosting using Branching Programs, Proceedings of the 13th Annual Conference on Computational Learning Theory,pp. 220-224,2000.

[O03] A. Owshanko, PExact=Exact learning, manuscript.

[S90] R. E. Schapire, The strength of weak learnability, *Machine Learning, 5(2)*pp. 197-227, 1990.

[V84] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.