

A Booster for the PAExact model

Nader H. Bshouty *

Department of Computer Science

Technion, 32000

Haifa, Israel

bshouty@cs.technion.ac.il

Abstract

We give a new booster that changes any PAC-learning algorithm to PAExact-learning algorithm that achieves exponentially small error. This booster is much more simpler and more efficient than previous ones and the first that uses deterministic hypotheses.

In particular, we show that if a class C is PAC-learnable in polynomial time with constant error and sample size V (for most learnable classes C , V is the VC-dimension of the class C) then C is PAExact-learnable in polynomial time with an algorithm that after d equivalence queries achieves error

$$\eta = 2^{-\tilde{O}\left(\left(\frac{d}{V}\right)^{\frac{1}{3}}\right)}.$$

We then give, for any integer V , a class C of VC-dimension V such that any PAExact-learning algorithm with unlimited computational power that PAExact-learns C and uses d equivalence queries will, with high probability, output a hypothesis with error at least

$$\eta \geq 2^{-\Omega\left(\frac{d}{V}\right)}$$

1 Introduction

In this paper we study two learning models that lie between the PAC and Exact models. The Probably Exact model (PEExact) model, introduced by Bshouty [B97], is the Exact model (learning *exactly* the target function from equivalence queries) in which each counterexample to an equivalence query EQ_D is drawn according to a distribution D rather than maliciously chosen. When the concept class is infinite it is impossible to achieve an exact learning of the target. In this case, the Probably Almost Exact model (PAExact), introduced by Bshouty, Jackson and Tamon [BJT02], is the same as the PExact model but requires that the hypothesis produced by the learning algorithm has negligible ($1/\omega(\text{poly})$) error.

In [BJT02], Bshouty et al. showed that Exact-learnable \Rightarrow PExact-learnable \Rightarrow PAExact-learnable \Rightarrow PAC-learnable. Based on Blum construction [B94], they also showed that under the standard cryptographic assumption that one-way functions exist, PExact-learnable \neq

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

PAC-learnable. Recently, Osherson [O03] proved that PExact-learnable = Exact-learnable. He showed that any PExact-learning algorithm that learns C under any distribution can be changed to an Exact-learning algorithm that learns C . In [BG02] Bshouty and Gavinsky showed that PAC-learnable = PAExact-learnable. They gave two constructions. The first construction is similar to Schapire boosting algorithm [S90]. It starts from a polynomial time PAC-learning algorithm that learns C with error $\varepsilon = 1/\text{poly}$ and constructs a polynomial time PAExact-learning algorithm for C that achieve error $1/\text{poly}^{O(\text{poly} \log)} = 1/\omega(\text{poly})$. The second construction is based on Mansour and McAllester [MA00] boosting algorithm and uses equivalence queries with randomized hypotheses. The construction takes a polynomial time PAC-learning algorithm that learns with error $1/\text{poly}$ and constructs a PAExact-learning algorithm that uses equivalence queries with randomized hypothesis and achieves error $1/2^{\text{poly}}$. It follows from the latter result that if a class C is PAC-learnable in polynomial time then it is learnable in the Probabilistic Prediction model from examples with an algorithm that runs in polynomial time for each prediction (polynomial in $\log(\text{the number of trials})$) and that after polynomial number of mistakes achieves a hypothesis that predicts the target with probability $1 - 1/2^{\text{poly}}$.

In particular, it is shown in [BG02] that if C is PAC-learnable in polynomial time with constant error and sample of size V , then C is PAExact-learnable (respectively, Probabilistic Predictable) in polynomial time that after d equivalence queries with randomized hypotheses (respectively, mistakes) achieves a hypothesis with error

$$\eta = \frac{1}{2^{\min\left(d^{\frac{1}{6}}, \left(\frac{d}{V}\right)^{\frac{1}{5}}\right)}}.$$

In this paper we give a new and simple booster that uses equivalence queries with *deterministic* hypotheses and substantially improves the error of the prediction. We show that after d mistakes it achieves a hypothesis with error

$$\eta = \frac{1}{2^{\tilde{O}\left(\left(\frac{d}{V}\right)^{\frac{1}{3}}\right)}}.$$

For bounded poly-bit distribution (a distribution D such that $D(x) = 0$ or $D(x) \geq 1/2^{\text{poly}}$ for every $x \in X$) our booster not only PAExact-learns the target but also PExact-learn it in polynomial time using equivalence queries with deterministic hypothesis.

In the Probabilistic Prediction model, Haussler et. al. in [HLW94] gave a double exponential time algorithm (in d) that achieves error $\eta \leq \frac{1}{2^{O(d/V_C)}}$ where V_C is the VC-dimension of the class C . They also gave an exponential time algorithm that achieves error $\eta \leq \frac{1}{2^{O((d/V_C)^{1/2})}}$ assuming that the consistent hypothesis problem for C (find a hypothesis $h \in C$ that is consistent with the sample) can be solved in polynomial time. It follows from our result that if the consistent hypothesis problem for C can be solved in polynomial time then there is a Probabilistic Prediction learning algorithm that runs in polynomial time and achieve error

$$\eta \leq \frac{1}{2^{\tilde{O}\left(\left(\frac{d}{V_C}\right)^{1/3}\right)}}.$$

We then give a lower bound. We show that for every integer k there is a class C with $V_C = k$ such that any PAExact-learning algorithm with unlimited computational power that learns C over the uniform distribution and uses d equivalence queries will, with probability at least $1/2$, achieve error at least

$$\eta \geq \frac{1}{2^{\Omega\left(\frac{d}{\sqrt{C}}\right)}}.$$

We also show that this lower bound is not true for any class C .

The remainder of the paper is organized as follows: In section 2 we define the learning models. In section 3 we give previous results and compare them with ours. In section 4 we give the new booster. In section 5 we prove the lower bound and in section 6 we give some open problems.

2 Learning Models and Definitions

Let C be a class of functions $f : X \rightarrow \{0, 1\}$. 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 $I_f = n + \text{size}_C(f)$ where $\text{size}_C(f)$ is the minimal size of a formula in C that is equivalent to f . Then “polynomial” means $\text{poly}(I_f) = \text{poly}(n, \text{size}_C(f))$. For infinite domains X the parameter n is usually replaced by the VC-dimension of the class V_C and $I_f = V_C + \text{size}_C(f)$. Then “polynomial” in this case is $\text{poly}(V_C, \text{size}_C(f))$.

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 (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.

Equivalence Query (EQ) [A88] In the equivalence query the learner asks EQ(h) for some polynomial size circuit h . The teacher chooses $y \in X_{f\Delta h} \triangleq \{x \in X \mid f(x) \neq h(x)\}$ and returns y . If $X_{f\Delta h}$ is empty, the teacher answers “YES”, indicating that h is equivalent to f .

Equivalence Query according to D (EQ $_D$)[B97] For the equivalence query according to distribution D the learner asks EQ $_D(h)$ for some polynomial size circuit ¹ h . The teacher chooses $y \in X_{f\Delta h}$ according to the induced distribution of D on $X_{f\Delta h}$ and returns $(y, f(y))$. If $\Pr_D[X_{f\Delta h}] = 0$, the teacher answers “YES”.

¹For infinite domains X , the definition of “circuit” depends on the setting in which the elements of C are represented. The hypothesis h must have polynomial size in this setting. E.g., if $X = \mathcal{R}^n$ we may ask of h to be a polynomial size *arithmetic* circuit

The EQ_D can be extended to also handle random hypothesis [BG02]. A random hypothesis $h_r : X \times R \rightarrow \{0, 1\}$ is a polynomial size circuit where for an input $x_0 \in X$ it randomly uniformly chooses $r_0 \in R$ and returns $h_{r_0}(x_0)$. In that case, $\text{EQ}_D(h_r(x))$ returns an output of the following procedure

1. Randomly choose $x_0 \in X$ according to distribution D .
2. Randomly uniformly choose $r_0 \in R$.
3. if $f(x_0) \neq h_{r_0}(x_0)$ then output $(x_0, f(x_0))$ else goto (1).

This procedure returns the first example x_0 that h_r err on.

When X is finite or countable infinite, each x_0 is received with probability

$$\frac{D(x_0) \Pr_r[h_r(x_0) \neq f(x_0)]}{\sum_x D(x) \Pr_r[h_r(x) \neq f(x)]}.$$

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 $\text{poly}(1/\varepsilon, \log(1/\delta), I_f)$.

Exact (Exactly Correct) [A88] In the Exact-learning model we say that an algorithm \mathcal{A} of the learner *Exact-learns* the class C if for any $f \in C$ and any δ the algorithm $\mathcal{A}(\delta)$ asks equivalence queries and with probability at least $1 - \delta$ outputs a polynomial size circuit h that is equivalent to f . We say that C is *Exact-learnable* if there is an algorithm that Exact-learns C in time $\text{poly}(\log(1/\delta), I_f)$.

PEXact (Probably Exactly Correct) [B97] In the Probably Exact model we say that an algorithm \mathcal{A} of the learner *PEXact-learns* the class C if for any $f \in C$ and any δ the algorithm $\mathcal{A}(\delta)$ asks equivalence queries with respect to D , EQ_D , and with probability at least $1 - \delta$ outputs a polynomial size circuit h that satisfies $\Pr_D[X_{f \Delta h}] = 0$. We say that C is *PEXact-learnable* if there is an algorithm that Probably Exact-learns C in time $\text{poly}(\log(1/\delta), I_f)$.

If the hypothesis in the equivalence query can also be randomized then we refer to this model as PEXact_R .

PAExact (Probably Almost Exactly Correct) [BJT02] In the Almost Exact model we say that an algorithm \mathcal{A} of the learner *PAExact-learns* the class C if for any $f \in C$ and any δ the algorithm $\mathcal{A}(\delta)$ asks equivalence queries with respect to D , EQ_D , and with probability

at least $1 - \delta$ outputs a polynomial size circuit h that satisfies $\Pr_D[X_{f\Delta h}] = 1/\omega(\text{poly}(I_f))$. We say that C is *PAExact-learnable* if there is an algorithm that PAExact-learns C in time $\text{poly}(\log(1/\delta), I_f)$.

If the hypothesis in the equivalence query can also be randomized then we refer to this model as PAExact_R .

We say that C is η -*PAExact-learnable* if it is PAExact -learnable and it achieves error at most η . That is, $\Pr_D[X_{f\Delta h}] \leq \eta$. When randomized hypotheses are used in the equivalence query then we say that C is η -*PAExact_R-learnable*.

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 $\text{poly}(\log(1/\delta), I_f)$.

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 $\text{poly}(\log(1/\delta), I_f)$.

For a countable domain X , we define

$$m_D = \inf\{x \in X \mid D(x) \neq 0\}.$$

A distribution is called *bounded poly-bit* distribution if $m_D > 1/2^{\text{poly}(I_f)}$.

3 Previous and New results

In this section we give some previous results and compare them with our results in this paper

3.1 PAC and PAExact

It is known that learnability (in polynomial time) in the Exact model implies learnability in the PAC model [A88, L88]. Blum showed in [B94] that under the standard cryptographic assumption that one-way functions exist PAC-learnable \neq Exact-learnable. This means that there is class C that is PAC-learnable in polynomial time but is not Exact-learnable in polynomial time assuming one-way functions exist. Bshouty et. al. showed in [BC+96] that if $\text{P}=\text{NP}$ then the PAC model is equivalent to the Exact model.

In [BJT02], Bshouty et al. showed that Exact-learnable \Rightarrow PExact-learnable \Rightarrow PAExact-learnable \Rightarrow PAC-learnable. Based on Blum construction [B94], they also showed that under

the standard cryptographic assumption that one-way functions exist, PExact-learnable \neq PAC-learnable. In [BG02] Bshouty and Gavinsky showed that PAC-learnable = PExact-learnable. They gave two constructions. The first construction is similar to Schapire boosting algorithm [S90]. It starts from a polynomial time PAC-learning algorithm that learns with error $\varepsilon = 1/\text{poly}(I_f)$ and construct a polynomial time PExact-learning algorithm that learns with error $1/I_f^{O(\log I_f)} = 1/\omega(\text{poly}(I_f))$. This implies that PAC-learnable = $1/I_f^{O(\log I_f)}$ -PExact-learnable. The second construction is based on Mansour and McAllester [MA00] boosting algorithm and uses equivalence queries with randomized hypotheses. The construction takes a polynomial time PAC-learning algorithm that learns with error $1/\text{poly}(I_f)$ and construct a PExact-learning algorithm that uses equivalence queries with randomized hypothesis and achieves error $1/2^{\text{poly}(I_f)}$. This implies that PAC= $1/2^{\text{poly}(I_f)}$ -PExact_R.

In this paper we show that PAC= $1/2^{\text{poly}(I_f)}$ -PExact. That is, any PAC learning algorithm that runs in polynomial time can be changed to a PExact-learning algorithm that runs in polynomial time, asks equivalence queries with a deterministic hypothesis and outputs a hypothesis that achieves error $1/2^{\text{poly}(I_f)}$.

Our algorithm uses a new booster. Our booster starts from a deterministic hypothesis h that η approximates the target f and after polynomial number of equivalence queries builds a new deterministic hypothesis h' that $\eta/2$ approximates the target.

3.2 PAC and PExact

The result in [BG02] implies that if C is PAC-learnable then it is PExact_R-learnable under any bounded poly-bit distribution. Our result in this paper shows that if the class C is PAC-learnable then it is PExact-learnable under any bounded poly-bit distribution.

In [BJT02] it is shown that if the learner is deterministic then PExact-learnable is equivalent to Exact-learnable. A recent result of Owshanko in [O03] shows that PExact-learnable is equivalent to Exact-learnable even with randomized learner. The proof is based on building a set of distributions that are not bounded poly-bit and showing that any PExact learning algorithm that learns under any distribution from this set can be turned to Exact learning algorithm with the same complexity. This shows that our result (that is mentioned in the previous paragraph) cannot be extended to nonbounded poly-bit distributions.

3.3 Error rate in PExact and Lower bound

It is shown in [BG02] that if C is PAC-learnable (in polynomial time) with constant error and sample of size V , then C is PExact_R-learnable (in polynomial time) with an algorithm that after d equivalence queries achieves a hypothesis with error

$$\eta = \frac{1}{2^{\min\left(d^{\frac{1}{6}}, \left(\frac{d}{V}\right)^{\frac{1}{5}}\right)}}.$$

In this paper we show that C is PExact-learnable (in polynomial time) with an algorithm

that uses deterministic hypotheses and achieves a hypothesis with error

$$\eta = \frac{1}{2^{\tilde{O}\left(\left(\frac{d}{V}\right)^{\frac{1}{3}}\right)}}.$$

We then give a lower bound. We show that for every integer k there is a class C with VC-dimension $V_C = k$ such that any PAExact-learning algorithm with unlimited computational power that learns C over the uniform distribution and uses d equivalence queries will with probability at least $1/2$ achieves error at least

$$\eta \geq \frac{1}{2^{\Theta\left(\frac{d}{V_C}\right)}}.$$

A challenging open problem is to close the gap between the upper and the lower bound.

3.4 The PP model

It is known that the Online and the Exact learning models are equivalent [A88, L88]. In the same way as in [A88, L88] one can prove that learning in the PAExact model implies learning in the PP model. In particular, if a class C is η -PAExact-learnable in polynomial time then C is η -PP-learnable in polynomial time. It is not known whether η -PP-learnable implies η -PAExact-learnable. The reason is that the learner in the PP model may take advantage of the correct predictions (that are not counted in the complexity) to improve the hypothesis after each trial. This feature is missing in the PAExact model.

It follows from the latter result that if a class is PAC-learnable in polynomial time then it is learnable in the Probabilistic Prediction model from examples with an algorithm that runs in polynomial time for each prediction (polynomial in $\log(\text{the number of trials } t)$ or polynomial in the number of mistakes d) and that after polynomial number of mistakes achieves a hypothesis that predicts the target with probability $1 - 1/2^{\text{poly}(I_f)}$.

In the Probabilistic Prediction model, Haussler et. al. in [HLW94] gave a double exponential time algorithm (in d) for any class C that achieves error

$$\eta \leq \frac{1}{2^{\mathcal{O}\left(\frac{d}{V_C}\right)}}.$$

They also gave an exponential time algorithm that achieves error

$$\eta \leq \frac{1}{2^{\mathcal{O}\left(\left(\frac{d}{V_C}\right)^{1/2}\right)}}$$

assuming that the consistent hypothesis problem for C (find a hypothesis $h \in C$ that is consistent with the sample) can be solved in polynomial time. It follows from [BG02] that

if the consistent hypothesis problem for C can be solved in polynomial time then there is a Probabilistic Prediction learning algorithm that runs in polynomial time and achieves error

$$\eta = \frac{1}{2^{\min\left(d^{\frac{1}{6}}, \left(\frac{d}{\sqrt{C}}\right)^{\frac{1}{5}}\right)}}.$$

Our PAExact-learning algorithm in this paper gives a PP-learning algorithm that runs in polynomial time (assuming that the consistent hypothesis problem for C can be solved in polynomial time) and achieves error

$$\eta \leq \frac{1}{2^{\tilde{O}\left(\left(\frac{d}{\sqrt{C}}\right)^{1/3}\right)}}.$$

4 The New Booster

In this section we give our new booster and prove its correctness. In Subsection 4.1 we show how to start from a hypothesis that approximates the target function f and refine it to get a better one. In Subsection 4.2 we give the main algorithm and prove its correctness.

4.1 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 $\mathcal{H} = h_1, h_2, h_3, \dots, h_k$ and then uses this sequence to build the refined hypothesis.

We start with the following notation. Let

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

Let

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

Notice that

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

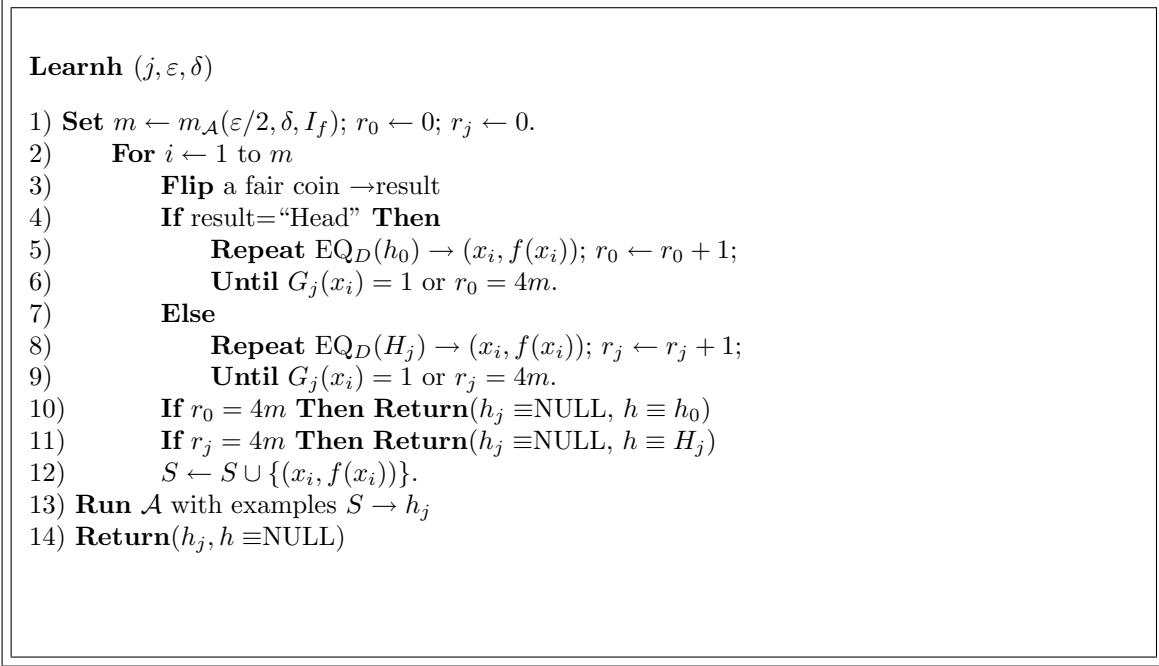


Figure 1: The algorithm **Learnh**(j, ε, δ) learns the j th function in the sequence \mathcal{H} .

Now we show how the booster learns h_j from h_1, h_2, \dots, 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 $h_j = \text{NULL}$ indicating that h_{j-1} is the last function in the sequence \mathcal{H} and then $\mathcal{H} = h_1, h_2, \dots, h_k$ for $k = j - 1$. In the latter case a new function h_j is generated in \mathcal{H} . We will show that for some $\varepsilon = 1/\text{poly}(\log(1/\eta))$ and $k = \text{poly}(\log(1/\eta))$, 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

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

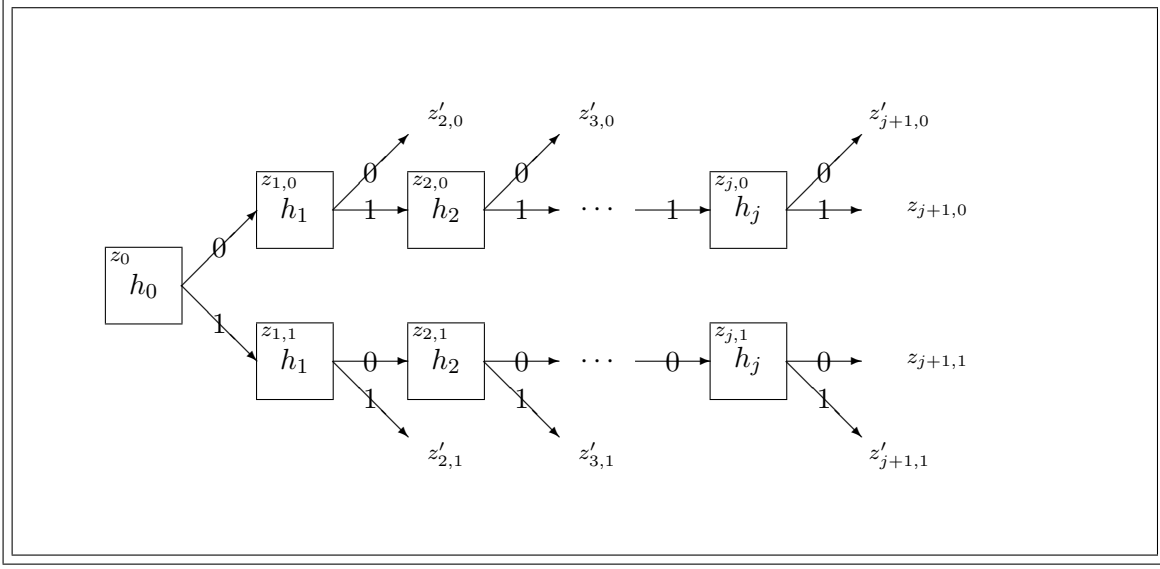


Figure 2: *The branching program $B_{\mathcal{H}}$ that **Learnh** builds*

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 $i = 2, \dots, j+1$, 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 $i = 2, \dots, j+1$, 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 $z_{j+1,1}$.
- 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,

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

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 $2(j\epsilon\eta + \epsilon^j)$. By property (P1) this labeling gives H_{j+1} . Therefore ², if (3) is true, then

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

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

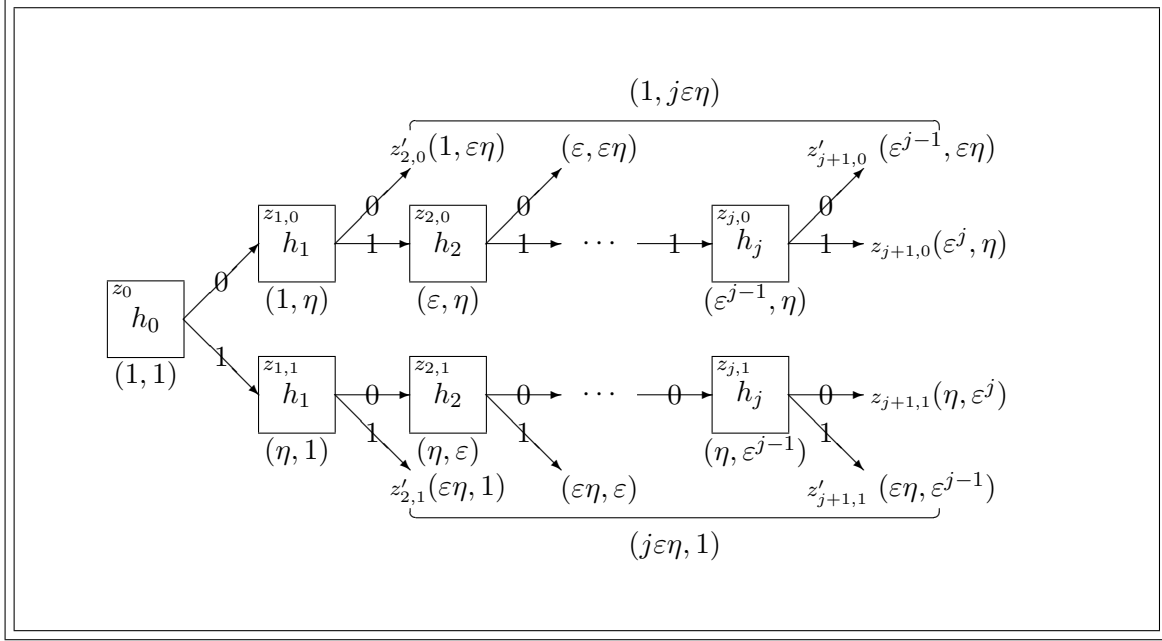


Figure 3: For every node z we have a pair $(p_z^{(0)}, p_z^{(1)})$ where $p_z^{(0)}$ (respectively, $p_z^{(1)}$) is the weight of the set of positive (respectively, negative) points that arrives at node z .

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, \dots, h_{j-1} , the algorithm tries to learn the target function over the domain of all points that 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 **Learn** 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 $\text{EQ}_D(H_j)$, which is by property (P4) equivalent to $\text{EQ}(\overline{h_0})$ for points x that satisfy $G_j(x) = 1$, and with probability $1/2$, asks $\text{EQ}_D(h_0)$. If the algorithm succeeds to learn h_j , then this hypothesis (as we will show in the proof) will refine the bounds $(p_{z_{j,0}}^{(0)}, p_{z_{j,0}}^{(1)})$ to $(\varepsilon p_{z_{j,0}}^{(0)}, p_{z_{j,0}}^{(1)})$ and $(p_{z_{j,0}}^{(0)}, \varepsilon p_{z_{j,0}}^{(1)})$ 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 $j, 1/\varepsilon = \text{poly}(1/\eta)$.

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 $1/\varepsilon, j = \text{poly}(1/\eta)$ 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 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

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

and

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

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

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

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

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

and this follows β . \square

We now prove the following Claims

Claim 4.1 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 4.2 With probability at least $1 - j\delta$ we have: For all $i \leq j$,

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

and

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

Claim 4.3 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 4.4 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 4.1. When **Learnh** learns h_j it asks with probability $1/2$, $\text{EQ}_D(h_0)$ and with probability $1/2$, $\text{EQ}_D(H_j)$ 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

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

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

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

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

Now the proof of Claim 4.2 follows from Property 1 and Claim 4.1.

Proof of Claim 4.3. We have

$$\begin{aligned}
\Pr_D[G_j = 0 | f \neq h_0] &= \frac{\Pr_D[G_j = 0, f \neq h_0]}{\Pr_D[f \neq h_0]} \\
&= \frac{\Pr_D[\overline{h_0}H_j^\wedge = 1, f = 1] + \Pr_D[h_0H_j^\vee = 1, f = 0]}{\Pr_D[f \neq h_0]} \\
&\leq \frac{v_{j-1}}{2(j-1)\varepsilon\eta}
\end{aligned}$$

By Claim 4.2, 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 $\text{EQ}_D(h_0)$, $4m$ times. Let X_r be a random variable that is equal to 1 if the r th call of $\text{EQ}_D(h_0)$ 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

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

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

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

Figure 4: A PAExact-learning algorithm that refine h_0

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 $j\delta$. \square

Proof of Claim 4.4: 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 4.3. \square

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} .

We now prove

Lemma 1 Suppose $\Pr_D[f \neq h_0] \leq \eta$ and $h = \mathbf{Refine}(h_0, k, \varepsilon, \delta)$. 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 $\mathcal{H} = h_1, h_2, \dots, h_t$, $t \leq k$ 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 happens if and only if one of the following events happens:

[A₁] For some $j = t \leq k$, **Learnh**(j, ε, δ') outputs $h \equiv h_0$ and $\Pr_D[f \neq h_0] \geq \eta'$.

[A₂] For some $j = t \leq k$, **Learnh**(j, ε, δ') outputs $h \equiv H_j$ and $\Pr_D[f \neq H_j] \geq \eta'$.

[A₃] We have $t = k$ and $\Pr_D[f \neq H_{k+1}] \geq \eta'$.

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

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

by Claim 4.3

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

In the same way one can prove

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

Now since

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

by Claim 4.2

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

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

4.2 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

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

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

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) $h_0 \leftarrow \mathbf{Refine}(h_0, k, \varepsilon, \delta')$
- 5) **Output**(h_0)

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

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

Proof. Let $h_0^{(0)}, h_0^{(1)}, \dots, h_0^{(t)}$, $t = \lceil \log(1/\eta) \rceil$ 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. \square

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. \square

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

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

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

equivalence queries, for some constants c_1 , c_2 and c_3 . \square

We now prove our main result

Corollary 1 *If C is PAC-learnable with error $\frac{1}{2} - \gamma$ and confidence λ with sample size V_0 , then for some constants c and c' , C is PAExact-learnable (respectively, PP-learnable) with an algorithm that with probability at least*

$$1 - d^{-c' \frac{V_0}{\gamma^2} \log d}$$

after d equivalence queries (respectively, mistakes) outputs a deterministic hypothesis that achieves error at most (respectively, fails to predict the target with probability at most)

$$\eta = 2^{-c \left(\frac{\gamma^2 d}{V_0 \log^2 d} \right)^{\frac{1}{3}} \log^{\frac{2}{3}} \left(\frac{\gamma^2 d}{V_0 \log^2 d} \right)}.$$

In particular, for $\frac{d}{\log^{2\alpha} d} > \left(\frac{V_0}{\gamma} \right)^\alpha$, $\alpha > 1$,

$$\eta \leq 2^{-c_\alpha \left(\frac{\gamma^2 d}{V_0} \right)^{\frac{1}{3}}},$$

for some constant c_α that depends on α .

For classes that are PAC-learnable with constant γ and $O(V_C)$ examples, where V_C is the VC-dimension of the class, if for some constant $\alpha > 1$ $d > V_C^\alpha$ then

$$\eta \leq 2^{-O\left(\left(\frac{d}{V_C}\right)^{\frac{1}{3}}\right)}.$$

Proof. We use Theorem 3 with

$$d = c_2 \frac{V_0 w}{\gamma^2} \log^2 \frac{V_0 w}{\gamma^2} \text{ and } d = c_1 w \log \frac{1}{\delta}, \tag{6}$$

for some constants c_1 and c_2 . Now the first equation gives

$$\frac{32 \log^3 \frac{1}{\eta}}{\log \log \frac{1}{\eta}} = w = c_3 \frac{\gamma^2 d}{V_0 \log^2 d}$$

for some constant c_3 and therefore the error is

$$\eta \leq 2^{-c \left(\frac{\gamma^2 d}{V_0 \log^2 d} \right)^{\frac{1}{3}} \log^{\frac{2}{3}} \left(\frac{\gamma^2 d}{V_0 \log^2 d} \right)},$$

for some constant c . The second equation in (6) gives

$$\delta = d^{-c' \frac{V_0}{\gamma^2} \log d},$$

for some constant c' . This completes the proof of the first part of the Corollary.

Now suppose $\frac{d}{\log^{2\alpha} d} > \left(\frac{V_0}{\gamma}\right)^\alpha$. Then

$$\begin{aligned} c \left(\frac{\gamma^2 d}{V_0 \log^2 d} \right)^{\frac{1}{3}} \log^{\frac{2}{3}} \left(\frac{\gamma^2 d}{V_0 \log^2 d} \right) &= c \left(\frac{\gamma^2 d}{V_0} \right)^{\frac{1}{3}} \left(1 - \frac{\log \frac{V_0 \log^2 d}{\gamma}}{\log d} \right)^{\frac{2}{3}} \\ &\geq c \left(\frac{\gamma^2 d}{V_0} \right)^{\frac{1}{3}} (1 - \alpha^{-1})^{\frac{2}{3}}. \square \end{aligned}$$

5 Lower Bound

In this section we prove the following lower bound

Theorem 4 *There is a class C of VC-dimension V_C such that any PAExact-learning algorithm that uses at most d equivalence queries with randomized hypotheses will, with probability at least $1/3$, outputs a hypothesis that its error is at least*

$$\eta_d = 2^{-c \left(\frac{d}{V_C} \right)}$$

for some constant c .

First note that this result is not true for all classes of VC-dimension V . In particular, let F_V be the class of all the functions $f : \{1, 2, \dots, V\} \rightarrow \{0, 1\}$. The VC-dimension of this class is V . For this class it is easy to see that the algorithm that builds a table for $f(1), f(2), \dots, f(V)$, under any distribution after $d < V$ equivalence queries achieves error (with high probability) at most $(V - d)/V$ and after V equivalence query achieves error 0 (with probability 1).

We now prove the Theorem

Proof of the Theorem: To prove the Theorem we give an adversary strategy for answering the equivalence queries such that for any strategy of the learner, with probability at least $1/2$, after d randomized equivalence queries there are still two hypotheses h_1 and h_2 consistent with all the answers and

$$\Pr[h_1 \neq h_2] \geq 2\eta_d.$$

This shows that for any output hypothesis h one of the possible target hypotheses h_1 or h_2 have error at least η_d .

We will first prove the Theorem for a class of VC-dimension $V_C = 1$. The class is defined as follows: Let $X = [0, 1] = \{x \mid 0 \leq x \leq 1\}$ and D the uniform distribution over $[0, 1]$. The concept class is

$$Ray = \{f_a \mid a \in [0, 1]\}$$

where $f_a(x) = [x > a]$. I.e., $f_a(b) = 1$ if $b \geq a$ and 0 otherwise.

At any stage of the algorithm the adversary holds a set of all the previous counterexamples $N \cup P$ where N is the set of negative counterexamples and P is the set of positive counterexamples. Let

$$m_N = \max_{(q,0) \in N} q \text{ and } m_P = \min_{(p,1) \in P} p.$$

Now the adversary strategy is as follows: For the next equivalence query $EQ(h_r)$ the adversary run the following procedure:

1. While (True) do
2. Randomly uniformly choose $x' \in [0, 1]$.
3. Randomly uniformly choose r .
4. $y' \leftarrow h_r(x')$.
5. If $y' = 1$ and $x' \leq m_N$ then return($x', 0$).
6. If $y' = 0$ and $x' \geq m_P$ then return($x', 1$).
7. If $m_N < x' < m_P$ then return(x', \bar{y}').

In this procedure the adversary chooses x' and r randomly uniformly to compute $h_r(x')$. Any consistent hypothesis f_a with $N \cup P$ is 0 for $x \leq m_N$ and 1 for $x \geq m_P$. Therefore, if $y' = 1$ and $x' \leq m_N$ then the adversary must return $(x', 0)$ (step 5) and if $y' = 0$ and $x' \geq m_P$ then the adversary must return $(x', 1)$ (step 6). In both cases the learner does not gain any new information about the target. If $m_N < x' < m_P$ then the adversary returns (x', \bar{y}') .

Now if X_i is the random variable $m_P - m_N$ after the i -th randomized equivalence query and Y_i is randomly uniformly chosen from $[0, X_i]$ then $X_0 = 1$ and

$$X_{i+1} \geq \min(Y_i, X_i - Y_i).$$

In the next Lemma we show that

$$\Pr[X_d \geq 2^{-3d}] \geq \frac{1}{2}.$$

Therefore with probability at least $1/2$, after d equivalence queries, we have $m_P - m_N \geq 2^{-3d}$. Since f_{m_P} and $f_{m_N + \xi}$ (for any small ξ) are possible hypotheses (consistent with all the counterexamples) and since $\Pr[f_{m_P} \neq f_{m_N + \xi}] = m_P - m_N \geq 2^{-3d}$ the result for VC-dimension 1 follows.

Now we show the lower bound for any VC-dimension. Let

$$\text{Ray}_V = \{f_a \mid a \in [0, 1]^V\}$$

where $f_a : \{1, 2, \dots, V\} \times [0, 1] \rightarrow \{0, 1\}$ and

$$f_a(k, x) = \begin{cases} 1 & x_k > a_k \\ 0 & \text{otherwise} \end{cases}.$$

Obviously, the VC-dimension of Ray_V is V . Let D be the uniform distribution over $\{1, 2, \dots, V\} \times [0, 1]$. Now the adversary uses the same strategy as before for each subdomain $\{i\} \times [0, 1]$. Suppose that after d equivalence queries we have d_i counterexample in the subdomain $R_i = \{i\} \times [0, 1]$. The distribution of each R_i is $1/V$ and the induced distribution of D over R_i is the uniform distribution. Therefore, the error that the i -th interval contributes is, with probability at least $1/2$, at least

$$\eta_i \geq 2^{-3d_i}/V$$

where d_i are the number of counterexample obtained in R_i . We now will assume w.l.o.g that $d_1 \leq d_2 \leq \dots \leq d_V$ and V is even. Since $\sum_i d_i = d$ we have

$$d_1 \leq \frac{2d}{V}, \dots, d_{V/2} \leq \frac{2d}{V}.$$

Therefore for each $1 \leq i \leq V/2$, with probability at least $1/2$

$$\eta_i \geq \frac{2^{-3d_i}}{V} \geq \frac{2^{-6\frac{d}{V}}}{V}. \quad (7)$$

Now take the random variable $E = \sum_{i=1}^{V/2} E_i$ where E_i is 1 if (7) is not true and 0, otherwise. By Markov inequality on E with probability $1/3$, $V/8$ of the η_i s satisfies (7). Therefore, with probability at least $1/3$ the total error is

$$\sum_{i=1}^{V/2} \eta_i \geq \frac{V}{8} \frac{2^{-6\frac{d}{V}}}{V} \geq 2^{-6\frac{d}{V}-3}.$$

This completes the proof for any VC-dimension. \square

It remains to show

Lemma 2 *Let $X_0, Y_0, X_1, Y_1, \dots, X_d, Y_d$ be random variables such that $X_0 = 1$, Y_i is randomly uniformly chosen from the interval $[0, X_i]$ and*

$$X_{i+1} = \min(Y_i, X_i - Y_i).$$

Then there is a constant c such that

$$\Pr \left[X_d \geq 2^{-3d} \right] \geq \frac{1}{2}.$$

Proof. Consider the random variables $Z_i = X_i/X_{i-1}$ and $W_i = Y_i/X_i$. Then W_i is a random uniform value in the interval $[0, 1]$, W_0, \dots, W_d are independent random variables and

$$Z_{i+1} = \frac{X_{i+1}}{X_i} = \min \left(\frac{Y_i}{X_i}, 1 - \frac{Y_i}{X_i} \right) = \min(W_i, 1 - W_i).$$

are also independent. Consider

$$Z = \prod_{i=1}^d Z_i^{-1/2} = X_d^{-1/2}.$$

Then

$$E[Z] = \prod_{i=1}^d E[Z_i^{-1/2}] = \prod_{i=1}^d \int_0^1 \min(w, 1-w)^{-1/2} dw = 2^{1.5d}.$$

By Markov Inequality we have

$$\begin{aligned} \Pr[X_d \leq 2^{2-3d}] &= \Pr[X_d^{-1/2} \geq 2^{-1} 2^{1.5d}] \\ &= \Pr[Z \geq 2^{-1} E[Z]] \\ &\leq \frac{1}{2}. \end{aligned}$$

Therefore $\Pr[X_d \geq 2^{-3d}] > 1/2. \square$

6 Open Problems

In this section we give some open problems

1. For PExact-learnability we showed the following bounds on the error

$$\frac{1}{2^{\Theta\left(\frac{d}{\sqrt{C}}\right)}} \leq \eta \leq \frac{1}{2^{O\left(\left(\frac{d}{\sqrt{C}}\right)^{1/3}\right)}}.$$

Tighten the gap between the lower and upper bounds.

2. Does PP-learnable = PAExact-learnable?
3. Show that DNF is PAExact-learnable under the uniform distribution with membership queries.

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.