Boosting and Hard-Core Set Construction

Adam R. Klivans* (klivans@theory.lcs.mit.edu)
Laboratory for Computer Science, MIT, Cambridge, MA 02139, U.S.A.

Rocco A. Servedio† (rocco@deas.harvard.edu)
Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, U.S.A.

Abstract. This paper connects *hard-core set construction,* a type of hardness amplification from computational complexity, and *boosting*, a technique from computational learning theory. Using this connection we give fruitful applications of complexity-theoretic techniques to learning theory and vice versa. We show that the hard-core set construction of Impagliazzo (Impagliazzo, 1995), which establishes the existence of distributions under which boolean functions are highly inapproximable, may be viewed as a boosting algorithm. Using alternate boosting methods we give an improved bound for hard-core set construction which matches known lower bounds from boosting and thus is optimal within this class of techniques. We then show how to apply techniques from (Impagliazzo, 1995) to give a new version of Jackson's celebrated Harmonic Sieve algorithm for learning DNF formulae under the uniform distribution using membership queries. Our new version has a significant asymptotic improvement in running time. Critical to our arguments is a careful analysis of the distributions which are employed in both boosting and hard-core set constructions.

Keywords: boosting, hard core set construction, computational complexity

1. Introduction

1.1. Boosting and Hard-Core Sets

This paper connects two fundamental ideas from theoretical computer science: *hard-core set construction,* a type of hardness amplification from computational complexity, and *boosting*, a technique from computational learning theory.

We refer to a *hardness amplification* as a result of the following form: given a Boolean function f that is mildly inapproximable by circuits of some bounded size g, construct from f a new function f' that is highly inapproximable by all circuits of size closely related to g. Here "mildly inapproximable" means roughly that no circuit can agree with f on a fraction of inputs very close to 1, while "highly inapproximable" means that no circuit can agree with f on a fraction of

* Supported in part by NSF grant CCR-97-01304.
† Supported in part by NSF grant CCR-95-04436 and by NSF grant CCR-98-77049.

Table I. Comparison of known hard-core set constructions.

Reference	Set size parameter	Circuit size parameter
Impagliazzo (Impagliazzo, 1995)	$\Omega(\epsilon)$	$O(\gamma^2\epsilon^2)g$
Nisan (Impagliazzo, 1995)	$\Omega(\epsilon)$	$O(\gamma^2(\log(1/(\gamma\epsilon)))^{-1})g$
This paper	$\Omega(\epsilon)$	$O(\gamma^2(\log(1/\epsilon))^{-1})g$

inputs significantly greater than $1/2$. Hardness amplification results are a crucial component of recent attempts to derandomize the complexity class BPP (Babai et al., 1993; Impagliazzo and Widgerson, 1997; Nisan and Wigderson, 1994). Perhaps the most famous hardness amplification result is Yao's XOR-lemma (Goldreich et al., 1995), which states that if a Boolean function f is mildly inapproximable by circuits of size g then the XOR of several independent copies of f is highly inapproximable for circuits of size closely related to g.

While the goal of hardness amplification is to amplify some small initial "hardness" of a boolean function, the goal of boosting is to "boost" some small initial advantage over random guessing that a learner can achieve in Valiant's PAC (Probabilistically Approximately Correct) model of learning. Roughly speaking, a *strong* learning algorithm in this model is an algorithm which, given access to random labelled examples $\langle x, f(x)\rangle$ drawn from any distribution $\mathcal{D}$, can generate a hypothesis h such that $\Pr_{x\in\mathcal{D}}[f(x) = h(x)] \geq 1-\epsilon$ for any $\epsilon > 0$, while a *weak* learning algorithm (Kearns and Valiant, 1994) can only do this for some $1/2 > \epsilon > 0$. Schapire (Schapire, 1990) and then Freund (Freund, 1990; Freund, 1992) gave boosting algorithms which convert weak learners into strong learners, thus proving the equivalence of weak and strong learnability. Since then, boosting has been applied in a wide variety of contexts and continues to be an active area of research, see e.g. (Drucker and Cortes, 1996; Drucker et al., 1994; Drucker et al., 1993b; Drucker et al., 1993a; Freund and Schapire, 1997; Jackson and Craven, 1996; Schapire and Singer, 1998). All known boosting algorithms work by using the weak learning algorithm several times on a sequence of carefully constructed distributions.

Superficially, boosting and hardness amplification seem to have opposite goals— boosting constructs a hypothesis which closely approximates a function f while hardness amplification results prove that certain functions are hard to approximate. The proof techniques employed in both areas, however, have a similar structure. All known hardness amplification results go by contradiction: assuming there exists a circuit C capable of mildly approximating f', one proves the

existence of a slightly larger circuit which closely approximates f. From this perspective, a hardness amplification proof resembles a type of boosting procedure: circuits which mildly approximate a function f' (these correspond to the hypotheses output by the weak learner) are combined to form a new circuit computing f on a large fraction of inputs.

In an important paper, Impagliazzo (Impagliazzo, 1995) reduces the problem of amplifying the hardness of a function f to the problem of constructing a distribution $\mathcal{D}$ such that f is highly inapproximable by small circuits for inputs chosen according to $\mathcal{D}$. He then constructs such a distribution and uses it to prove an XOR lemma. Impagliazzo also shows that the existence of such a distribution implies the existence of a "hard-core set" as defined in Section 2; we thus refer to Impagliazzo's method of constructing such a distribution as a *hard-core set construction*. Schapire (Schapire, 1990) was the first to point out that the existence of a boosting algorithm implies the existence of such a distribution.

1.2. Our Results

We give an explicit correspondence between the distributions that arise in Impagliazzo's hard-core set construction and the distributions constructed by boosting algorithms. This observation allows us to prove that the hard-core set construction of Impagliazzo *is* a boosting algorithm when the initial distribution is uniform. As we will show, there are two important parameters which boosting and hard-core set constructions share: the number of "stages" required and the "boundedness" of the distributions which are constructed. Interestingly, the procedures which have been used for hard-core set construction have better "boundedness" and can be used to improve algorithms in computational learning theory, while boosting algorithms require fewer "stages" and can be used to improve hard-core set construction.

We first show how to use known boosting algorithms to obtain new hard-core set constructions. In (Impagliazzo, 1995) Impagliazzo proves the following theorem: given a function f such that no circuit of size less than g correctly computes f on more than $(1-\epsilon)2^n$ inputs, then for any $\gamma < 1/2$ there exists a set S of size $\epsilon 2^n$ such that no circuit of size $O(\gamma^2\epsilon^2)g$ can correctly compute f on more than a $(1/2+\gamma)$ fraction of the inputs in S. By letting known boosting algorithms dictate the construction of the distributions in Impagliazzo's proof, we improve on previous results with respect to the circuit size parameter with only a small constant factor loss in the set size parameter. As explained in Section 4.5, we believe our parameters to be optimal up to constant

factors with respect to this class of techniques. Table 1 summarizes our hard-core set construction results.

We then show how to use Impagliazzo's hard-core set construction to obtain a more efficient version of Jackson's Harmonic Sieve algorithm (Jackson, 1997) for learning DNF formulae under the uniform distribution using membership queries. Jackson's original algorithm learns using the hypothesis class of threshold-of-parity functions and runs in time essentially $\tilde{O}(ns^{10}/\epsilon^{12})$, where n is the number of variables in the DNF formula, s is the number of terms, and ϵ is the accuracy parameter.[1] Our variant uses the same hypothesis class and runs in time $\tilde{O}(ns^{10}/\epsilon^{10})$. We can further improve the running time to $\tilde{O}(ns^{10}/\epsilon^{6})$ at the cost of learning using a more complex class of hypotheses.

In recent work Bshouty, Jackson and Tamon (Bshouty et al., 1999) have improved the running time of the Harmonic Sieve to $\tilde{O}(ns^6/\epsilon^4)$.[2] Our results improve the running time of their new algorithm to $\tilde{O}(ns^6/\epsilon^2)$ time steps, which is the fastest known algorithm for PAC learning DNF with membership queries under the uniform distribution.

Our main technical contribution is a careful analysis of the distributions constructed during the boosting process. We show that boosting procedures which construct distributions with high minimum entropy are desirable for good hard-core set constructions.

1.3. Related Work

Boneh and Lipton (Boneh and Lipton, 1993) have applied Yao's XOR-lemma to prove the equivalence of weak and strong learnability for certain types of concept classes under the uniform distribution. Their result applies to concept classes closed under a polynomial number of XOR operations.

1.4. Organization

In Section 2 we give an overview of the hard-core set construction found in (Impagliazzo, 1995). In Section 3 we outline the structure of all known boosting algorithms. In Section 4 we give an explicit connection between the constructions detailed in Sections 2 and 3 and show how to apply boosting techniques to obtain new hard-core set constructions. In Section 5 we show how the techniques described in section 2 can be used to improve the running time of Jackson's algorithm for learning

[1] In (Jackson, 1997) a running time of $\tilde{O}(ns^8/\epsilon^{12})$ is claimed but this is in error as described in Section 5.2 (Jackson, 2002).

[2] In (Bshouty et al., 1999) a running time of $\tilde{O}(ns^4/\epsilon^4)$ is claimed but this is in error as described in Section 5.4 (Jackson, 2002).

DNF formulae. We also describe related algorithms in learning theory where our techniques can be applied.

2. Hard-Core Set Construction Overview

2.1. Definitions

Our first definition, taken from (Impagliazzo, 1995), formalizes the notion of a function which is hard to approximate. (Readers who are familiar with the notation of (Impagliazzo, 1995) will notice that we are using different variables; the reasons for this will become clear in Section 4.)

Definition 1. Let f be a boolean function on $\{0,1\}^n$ and $\mathcal{D}$ a distribution on $\{0,1\}^n$. Let $0 < \epsilon < 1/2$ and let $n \leq g \leq 2^n/n$. We say that f is *ϵ-hard for size g under $\mathcal{D}$* if for any boolean circuit C with at most g gates, we have $\Pr_{\mathcal{D}}[f(x) = C(x)] \leq 1 - \epsilon$.

In other words, any circuit of size at most g must disagree with f with probability at least ϵ for x drawn according to $\mathcal{D}$. (Throughout the paper we use "circuit" to refer to a fanin-2 circuit composed of AND, OR and NOT gates. Also, throughout the paper we use $\mathcal{U}$ to denote the uniform distribution on $\{0,1\}^n$.)

Definition 2. A *measure* on $\{0,1\}^n$ is a function $M : \{0,1\}^n \to [0,1]$. The *absolute size* of a measure M is denoted by $|M|$ and equals $\sum_x M(x)$; the *relative size* of M is denoted $\mu(M)$ and equals $|M|/2^n$.

Definition 3. For a real valued function ξ, $L_\infty(\xi)$ denotes $\max_x |\xi(x)|$.

The quantity $\log(L_\infty(\mathcal{D})^{-1})$ is often referred to as the *minimum entropy* of $\mathcal{D}$. There is a natural correspondence between measures and distributions: the distribution $\mathcal{D}_M$ induced by a measure M is defined by $\mathcal{D}_M(x) = M(x)/|M|$. Conversely, if $\mathcal{D}$ is a distribution then the measure $M_{\mathcal{D}}$ induced by $\mathcal{D}$ is defined by $M_{\mathcal{D}}(x) = \mathcal{D}(x)/L_\infty(\mathcal{D})$. Thus $M_{\mathcal{D}}$ is the largest measure which is a constant-multiple rescaling of $\mathcal{D}$ (note that $\mathcal{D}$ itself is a measure, though typically one which has much smaller size than $M_{\mathcal{D}}$). It is clear that $|M_{\mathcal{D}}| = 1/L_\infty(\mathcal{D})$ and $\mu(M_{\mathcal{D}}) = 1/L_\infty(2^n\mathcal{D})$. Thus, large measures correspond to distributions which do not assign large weight to any point (i.e., have high minimum entropy).

The next definition is also from (Impagliazzo, 1995):

Input: real numbers $0 < \epsilon < 1$, $0 < \gamma < 1/2$
Boolean function f
Output: circuit h such that $\Pr_{\mathcal{U}}[h(x) = f(x)] > 1 - \epsilon$

1. **set** $i \leftarrow 0$
2. **set** $M_0(x) \equiv 1$
3. **until** $\mu(M_i) < \epsilon$ **do**
4. let C_i be a circuit of size g' with $\Pr_{\mathcal{D}_{M_i}}[C(x) = f(x)] \geq 1/2 + \gamma$
5. **set** $R_{C_i}(x) \equiv 1$ if $f(x) = C_i(x)$, $R_{C_i}(x) \equiv -1$ otherwise
6. **set** $N_i(x) \equiv \sum_{0 \leq j \leq i} R_{C_j}(x)$
7. **set** $M_{i+1}(x) \equiv \begin{cases} 1 & \text{if } N_i(x) < 0 \\ 1 - \epsilon\gamma N_i(x) & \text{if } 0 \leq N_i(x) \leq 1/(\epsilon\gamma) \\ 0 & \text{if } N_i(x) > 1/(\epsilon\gamma) \end{cases}$
8. **set** $i \leftarrow i + 1$
9. **return** $h \equiv MAJ(C_0, C_1, \ldots, C_{i-1})$

Figure 1. The `IHA` algorithm.

Definition 4. We say that f is *γ-hard-core on M for size g* if we have $\Pr_{\mathcal{D}_M}[f(x) = C(x)] \leq 1/2 + \gamma$ for every circuit C of size at most g. For $S \subseteq \{0,1\}^n$ we say that f *is γ-hard-core on S for size g* if f is γ-hard-core on M_S for size g, where $M_S(x)$ is the characteristic function of S.

2.2. Existence of Hard-Core Measures

The following theorem due to Impagliazzo is the starting point of all our results:

Theorem 5. (Impagliazzo, 1995) Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0 < \gamma < 1/2$. Then there is a measure M with $\mu(M) \geq \epsilon$ such that f is γ-hard-core on M for size $g' = O(\epsilon^2\gamma^2)g$.

Proof: Assume by way of contradiction that for every measure M with $\mu(M) \geq \epsilon$ there is a circuit C_M of size at most g' such that $\Pr_{\mathcal{D}_M}[f(x) = C_M(x)] > 1/2 + \gamma$. Now consider the algorithm `IHA` which is given in Figure 2.1. This algorithm iteratively modifies M until its relative size is less than ϵ. After each modification we obtain a circuit C_M as above. Once the relative size of M becomes less than ϵ we combine the circuits obtained during the process to contradict the original assumption. The following easily verifiable claims are useful for understanding how `IHA` works:

- $N_i(x)$ is the margin by which the majority vote of $C_0, \ldots, C_i$ correctly predicts the value of $f(x)$.
- The measure M_{i+1} assigns weight 0 to points where the margin of correctness is large, weight 1 to points where the margin is negative, and intermediate weight to points where the margin is positive but small.

Impagliazzo proves the following claim in (Impagliazzo, 1995):

Claim 6. After at most $i_0 = O(1/(\epsilon^2\gamma^2))$ cycles through the loop in `IHA`, $\mu(M_i)$ must be less than ϵ.

Once this happens and we exit the loop, it is easy to see that $h \equiv MAJ(C_0, \ldots, C_{i-1})$ agrees with f on all inputs except those which have $N_i(x) \leq 0$ and hence $M_i(x) = 1$. Since $\mu(M_i) < \epsilon$, this implies that $\Pr_{\mathcal{U}}[f(x) = h(x)] \geq 1 - \mu(M_i) > 1 - \epsilon$. But h is a majority circuit over at most i_0 circuits each of size at most g', and majority over i_0 inputs can be computed by a circuit of size $O(i_0)$ (see e.g. (Muller and Preparata, 1975)). It follows that h has at most $g'i_0 + O(i_0) \leq g$ gates, which contradicts the original assumption that f is ϵ-hard for circuits of size g under $\mathcal{U}$. □

Using a non-constructive proof technique, Nisan has established a similar result which is reported in (Impagliazzo, 1995). In Nisan's theorem the circuit size parameter is slightly worse as a function of γ but substantially better as a function of ϵ :

Theorem 7. (Impagliazzo, 1995) Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0 < \gamma < 1/2$. Then there is a measure M with $\mu(M) \geq \epsilon$ such that f is γ-hard-core on M for size $g' = O(\gamma^2(\log(2/\gamma\epsilon))^{-1})g$.

In Section 4.2 we will establish results of this type which have a better circuit size parameter than either Theorem 5 or Theorem 7.

We note that Theorems 5 and 7 assert the existence of a large measure, not a large set as was promised in Section 1. Using a probabilistic argument which we give in Section 4.4 Impagliazzo has shown that the existence of a large measure M on which f is hard-core implies the existence of a large set S on which f is also hard-core (with slightly different parameters).

3. Boosting Overview

In this section we define the learning model, weak and strong learning, and *boosting*, which converts a weak learner to a strong one.

3.1. Definitions

We take as our learning framework Valiant's widely studied PAC (Probably Approximately Correct) model of concept learning (Valiant, 1984). In this model a *concept class* is a collection $C = \cup_{n \geq 1} C_n$ of boolean functions where each $f \in C_n$ is a boolean function on $\{0,1\}^n$. For example, we might have C_n as the class of all boolean conjunctions on n variables. If f and h are two boolean functions on $\{0,1\}^n$ and $\mathcal{D}$ is a distribution on $\{0,1\}^n$, we say that h is an *ϵ-approximator for f under $\mathcal{D}$* if $\Pr_{\mathcal{D}}[f(x) = h(x)] \geq 1 - \epsilon$. The learner has access to an *example oracle* $\mathrm{EX}(f, \mathcal{D})$ which, when queried, provides a labelled example $\langle x, f(x) \rangle$ where x is drawn from $\{0,1\}^n$ according to the distribution $\mathcal{D}$ and $f \in C_n$ is the unknown target concept which the algorithm is trying to learn. The goal of the learner is to generate an ϵ-approximator for f under $\mathcal{D}$. We thus have the following definition:

Definition 8. An algorithm $\mathtt{A}$ is a *strong PAC learning algorithm for a concept class C* if the following condition holds: for any $n \geq 1$, any $f \in C_n$, any distribution $\mathcal{D}$ on $\{0,1\}^n$, and any $0 < \epsilon, \delta < 1$, if $\mathtt{A}$ is given access to n, ϵ, δ and $\mathrm{EX}(f, \mathcal{D})$, then $\mathtt{A}$ runs in time polynomial in n, ϵ^{-1}, δ^{-1}, and $size(f)$, and with probability at least $1 - \delta$ algorithm $\mathtt{A}$ outputs an ϵ-approximator for f under $\mathcal{D}$.

In the above definition $size(f)$ measures the complexity of the function f under some fixed reasonable encoding scheme. For the concept class DNF which we will consider in Section 5, $size(f)$ is the minimum number of terms in any disjunctive normal form representation of f.

If the algorithm $\mathtt{A}$ is only guaranteed to find a $(1/2 - \gamma)$-approximator for some $\gamma > 0$ with probability $1 - \delta$, then we say that $\mathtt{A}$ is a $(1/2 - \gamma)$-*approximate* learning algorithm; if $\gamma = \Omega(1/p(n, size(f)))$ for some polynomial p, we say that $\mathtt{A}$ is a *weak* learning algorithm (The notion of weak learning was introduced by Kearns and Valiant in (Kearns and Valiant, 1994)). We will abuse notation and say that $\mathtt{A}$ is a $(1/2 - \gamma)$-approximate learning algorithm for f if $\mathtt{A}$ is a $(1/2 - \gamma)$-approximate learning algorithm for the concept class C which consists of the single function f. In a series of important results, Schapire (Schapire, 1990) and subsequently Freund (Freund, 1990; Freund, 1992) have shown that if $\mathtt{A}$ is a weak learning algorithm for a concept class C, then there exists a strong learning algorithm for C. Their proofs are highly constructive

in that they give explicit *boosting* algorithms which transform weak learning algorithms into strong ones. We now formally define boosting algorithms (a related definition can be found in (Freund, 1995)):

Definition 9. An algorithm B is said to be a *boosting algorithm* if it satisfies the following condition: for any boolean function f and any distribution $\mathcal{D}$, if B is given $0 < \epsilon, \delta < 1$, $0 < \gamma \leq 1/2$, an example oracle $\mathrm{EX}(f, \mathcal{D})$, and a $(1/2 - \gamma)$-approximate learning algorithm WL for f, then algorithm B runs in time polynomial in $n, size(f), \gamma^{-1}, \epsilon^{-1}$, and δ^{-1}, and with probability at least $1 - \delta$ algorithm B outputs an ϵ-approximator for f under $\mathcal{D}$.

3.2. Structure of Boosting Algorithms

All known boosting algorithms rely crucially on the fact that the weak learning algorithm WL can find a $(1/2 - \gamma)$-approximator for f under $\mathcal{D}'$ for *any* distribution $\mathcal{D}'$, as long as WL is given access to the example oracle $\mathrm{EX}(f, \mathcal{D}')$. We give the following high-level definition:

Definition 10. A *canonical booster* is a boosting algorithm which has the following iterative structure:

- At stage 0 the algorithm starts with $\mathcal{D}_0 = \mathcal{D}$ and uses WL to generate an approximator h_0 for f under $\mathcal{D}_0$.
- At stage i the boosting algorithm does two things: (1) constructs a distribution $\mathcal{D}_i$ which favors points where the previous hypotheses $h_0, \ldots, h_{i-1}$ do poorly at predicting the value of f, and (2) simulates the example oracle $\mathrm{EX}(f, \mathcal{D}_i)$ and lets WL access this simulated example oracle to produce a hypothesis h_i which is an approximator for f under $\mathcal{D}_i$.
- Finally, after doing this repeatedly for several stages, the boosting algorithm combines the hypotheses $h_0, \ldots, h_{i-1}$ in some way to obtain a final hypothesis h which satisfies the following property: if each hypothesis h_i is a $(1/2 - \gamma)$-approximator for f under $\mathcal{D}_i$, then h is an ϵ-approximator for f under $\mathcal{D}$.

We feel that this definition captures the essence of known boosting algorithms.

4. Hard-Core Set Construction from Boosting

4.1. A Structural Similarity

From the descriptions of the hard-core set construction of Section 2 and the canonical boosting algorithm of Section 3, one can see a close structural resemblance between the IHA algorithm and the canonical boosting algorithm outlined above. To be more specific, just as IHA assumes that at each stage there is a circuit C_i for which $\Pr_{\mathcal{D}_{M_i}}[f(x) \neq C_i(x)] \leq 1/2 - \gamma$, the canonical boosting algorithm assumes that WL can generate at each stage a hypothesis h_i for which $\Pr_{\mathcal{D}_i}[f(x) \neq h_i(x)] \leq 1/2 - \gamma$. The induced distributions $\mathcal{D}_{M_i}$ of IHA correspond precisely to the distributions $\mathcal{D}_i$ of the canonical boosting algorithm (note that IHA starts off with the measure $M_0 = 1$ which corresponds to the uniform distribution $\mathcal{U} = \mathcal{D}_0$). Finally, just as the canonical boosting algorithm combines the hypotheses $h_0, \ldots, h_{i-1}$ in some fashion to obtain a final hypothesis h which has $\Pr_{\mathcal{U}}[f(x) = h(x)] \geq 1 - \epsilon$, the IHA algorithm combines the circuits $C_0, \ldots, C_{i-1}$ by taking majority to obtain a circuit h such that $\Pr_{\mathcal{U}}[f(x) = h(x)] \geq 1 - \epsilon$.

We conclude that IHA is an algorithm which succeeds in boosting *provided that the starting distribution is the uniform distribution* $\mathcal{U}$. Since boosting algorithms from computational learning theory will work for *any* starting distribution, *a priori* it seems as if it should be possible to use any boosting algorithm in place of IHA and obtain a hard-core set construction. In the next section we prove a theorem which formalizes this idea and emphasizes the parameters which are important to obtain a good hard-core set construction.

4.2. A General Hard-Core Set Construction

Definition 11. Let $\mathcal{D}$ be a distribution over $\{0,1\}^n$. For $d \geq 1$ we say that $\mathcal{D}$ is *d-smooth* if $L_\infty(2^n\mathcal{D}) \leq d$.

As an immediate consequence of Definitions 2 and 11, we have

Observation 12. If distribution $\mathcal{D}$ is d-smooth then $\mu(M_\mathcal{D}) \geq 1/d$.

Definition 13. Let B be a canonical boosting algorithm which takes as input ϵ, δ, γ, an example oracle $\mathrm{EX}(f, \mathcal{D})$, and a $(1/2 - \gamma)$-approximate learning algorithm WL for f.

1. We say that B is a *$k(\epsilon, \gamma)$-stage* boosting algorithm if the following holds: For all example oracles $\mathrm{EX}(f, \mathcal{D})$ and $(1/2 - \gamma)$-approximate learners WL for f, algorithm B simulates at most $k = k(\epsilon, \gamma)$ distributions $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{k-1}$ for WL and uses WL to generate at most k hypotheses $h_0, \ldots, h_{k-1}$.

2. We say that B is a $d(\epsilon,\gamma)$-*smooth* boosting algorithm if the following holds: For all functions f and $(1/2-\gamma)$-approximate learners WL, when B is given $\mathrm{EX}(f,\mathcal{U})$ and WL, with probability $1-\delta$ both of the following events occur: (i) the simulated distributions $\mathcal{D}_0,\ldots,\mathcal{D}_{k-1}$ are each $d(\epsilon,\gamma)$-smooth, and (ii) the hypothesis h which B outputs satisfies $\Pr_{\mathcal{U}}[f(x)=h(x)] \geq 1-\epsilon$.

The property of the distributions $\mathcal{D}_i$ described in part 2 of the above definition is similar to Levin's notion of "dominated" distributions (Levin, 1986).

Now we can state the following theorem which generalizes Impagliazzo's hard-core set construction.

Theorem 14. Let B be a $k(\epsilon,\gamma)$-stage, $d(\epsilon,\gamma)$-smooth boosting algorithm which outputs as its final hypothesis a circuit of size r over inputs $h_0,\ldots,h_{k-1}$. Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0<\gamma<1/2$. Then there is a measure M on $\{0,1\}^n$ with $\mu(M) \geq 1/d(\epsilon,\gamma)$ such that f is γ-hard-core on M for size $g'=(g-r)/k(\epsilon,\gamma)$.

Proof: The proof is analogous to the proof of Theorem 5. Assume by way of contradiction that for every measure M with $\mu(M) \geq 1/d(\epsilon,\gamma)$ there is a circuit C_M of size at most g' such that $\Pr_{\mathcal{D}_M}[f(x)=C_M(x)] \geq 1/2+\gamma$. By Observation 12, this implies that for every $d(\epsilon,\gamma)$-smooth distribution $\mathcal{D}$ there is a circuit $C_{\mathcal{D}}$ of size at most g' for which we have $\Pr_{\mathcal{D}}[f(x)=C_{\mathcal{D}}(x)] \geq 1/2+\gamma$.

Now run the boosting algorithm B on inputs ϵ,δ,γ, and $\mathrm{EX}(f,\mathcal{U})$. Since B is $d(\epsilon,\gamma)$-smooth, with nonzero probability we have that (i) every distribution $\mathcal{D}_i$ which B simulates will be $d(\epsilon,\gamma)$-smooth, and (ii) the final hypothesis which B outputs is an ϵ-approximator to f under the original distribution $\mathcal{U}$. By (i), there must exist a circuit C_i of at most g' gates which is a $(1/2-\gamma)$-approximator for f under $\mathcal{D}_i$. Give B this circuit when it calls WL on distribution $\mathcal{D}_i$. Now by (ii), the final hypothesis which B outputs must be an ϵ-approximator to f under the original distribution $\mathcal{U}$. But since B is $k(\epsilon,\gamma)$-stage, this final hypothesis is a circuit of size at most $r+g'k(\epsilon,\gamma) \leq g$, which contradicts the original assumption that f is ϵ-hard for circuits of size g under $\mathcal{U}$. □

4.3. New Hard-Core Set Constructions

Here we apply Theorem 14 to obtain new hard-core set constructions from known boosting algorithms. We proceed in stages. First, we show how two different boosting algorithms yield different hard-core set constructions. Next, we combine these boosting algorithms to achieve a

new hard-core set construction which improves on results of Impagliazzo and Nisan in the circuit size parameter but has a set size parameter worse than their results by a logarithmic factor. In Section 4.5 we improve the set size parameter to within a constant factor of the Impagliazzo/Nisan results.

We first consider Freund's boost-by-majority algorithm from (Freund, 1990) which, following (Jackson, 1995), we refer to as F1. Algorithm F1 is a $k = O(\gamma^{-2}\log(1/\epsilon))$-stage boosting algorithm which combines its k hypotheses using the majority function. Jackson's analysis ((Jackson, 1995), pp. 57–59) yields the following fact about F1:

Fact 15. If F1 is given inputs ϵ, δ, γ, and access to $\mathrm{EX}(f, \mathcal{D})$ and to a $(1/2 - \gamma)$-approximate weak learner WL for f, then with probability $1 - \delta$ each distribution $\mathcal{D}'$ which F1 simulates for WL satisfies

$$L_\infty(\mathcal{D}') = O(1/\epsilon^3) \cdot L_\infty(\mathcal{D}).$$

This immediately implies that F1 is $O(1/\epsilon^3)$-smooth.[3] We thus obtain the following hard-core set construction:

Theorem 16. Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0 < \gamma < 1/2$. Then there is a measure M on $\{0,1\}^n$ with $\mu(M) = \Omega(\epsilon^3)$ such that f is γ-hard-core on M for size $g' = O(\gamma^2(\log(1/\epsilon))^{-1})g$.

Next, we consider Freund's later $\mathtt{B}_{\mathrm{Filt}}$ algorithm from (Freund, 1995) (the name comes from the fact that the algorithm "filters" examples from the original distribution to simulate new distributions). Like F1, algorithm $\mathtt{B}_{\mathrm{Filt}}$ is a k-stage boosting algorithm for $k = O(\gamma^{-2}(\log 1/\epsilon))$. $\mathtt{B}_{\mathrm{Filt}}$ combines its $(1/2-\gamma)$-approximators to obtain an ϵ-approximator for f by using a majority function on k inputs which may have some random inputs. A straightforward argument shows that some circuit of size $O(k)$ is an ϵ-approximator for f. To analyze the smoothness of $\mathtt{B}_{\mathrm{Filt}}$, we use the following fact which follows from Lemma 3.4 and Lemma 3.9 of (Freund, 1995):

Fact 17. If $\mathtt{B}_{\mathrm{Filt}}$ is given inputs ϵ, δ, γ, $\mathrm{EX}(f, \mathcal{D})$ and a $(1/2 - \gamma)$-approximate weak learner WL for f, then with probability $1 - \delta$ each distribution $\mathcal{D}'$ which $\mathtt{B}_{\mathrm{Filt}}$ simulates for WL satisfies

$$L_\infty(\mathcal{D}') = O(\log(1/\epsilon)/(\epsilon\gamma)) \cdot L_\infty(\mathcal{D}).$$

Since Fact 17 implies that $\mathtt{B}_{\mathrm{Filt}}$ is $O(\log(1/\epsilon)/(\epsilon\gamma))$-smooth, we obtain

[3] In (Freund, 1990) Freund states that the F1 algorithm can be shown to be $O(1/\epsilon^2)$-smooth. Jackson proves that F1 is $O(1/\epsilon^3)$-smooth and states that variants of F1 can be shown to be $O(1/\epsilon^{2+\rho})$-smooth for arbitrarily small values $\rho > 0$.

Theorem 18. Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0 < \gamma < 1/2$. Then there is a measure M on $\{0,1\}^n$ with $\mu(M) = \Omega(\epsilon\gamma(\log(1/\epsilon))^{-1})$ such that f is γ-hard-core on M for circuits of size $g' = O(\gamma^2(\log(1/\epsilon))^{-1}g)$.

Finally we establish a stronger hard-core set construction by combining the previous two approaches. In (Freund, 1992) Freund describes a two-level boosting algorithm which works as follows: algorithm `F1` is used to boost from accuracy $(1/2 - \gamma)$ to accuracy $1/4$, and algorithm $\mathtt{B}_{\mathrm{Filt}}$ boosts from accuracy $1/4$ to accuracy ϵ by taking `F1` as its weak learner. We call this combined algorithm $\mathtt{B}_{\mathrm{Comb}}$.

Lemma 19. $\mathtt{B}_{\mathrm{Comb}}$ is an $O(\gamma^{-2}\log(1/\epsilon))$-stage boosting algorithm.

Proof: The top level of $\mathtt{B}_{\mathrm{Comb}}$, which uses algorithm $\mathtt{B}_{\mathrm{Filt}}$, takes $O(\log(1/\epsilon))$ stages since the weak learner which it uses is `F1` which provides $(1/2 - \gamma')$-accurate hypotheses with $\gamma' = 1/4$. The bottom level, which uses algorithm `F1`, takes $O(\gamma^{-2})$ stages since it boosts a $(1/2 - \gamma)$-approximate learner to accuracy $1/4$. Consequently the combined algorithm $\mathtt{B}_{\mathrm{Comb}}$ uses the claimed number of stages. □

Lemma 20. $\mathtt{B}_{\mathrm{Comb}}$ is an $O(\log(1/\epsilon)/\epsilon)$-smooth boosting algorithm.

Proof: Since $\mathtt{B}_{\mathrm{Filt}}$ is boosting from accuracy $1/4$ to accuracy ϵ using `F1` as its weak learner, Fact 17 implies that each distribution $\mathcal{D}'$ which $\mathtt{B}_{\mathrm{Filt}}$ passes to `F1` satisfies

$$L_\infty(\mathcal{D}') = O(\log(1/\epsilon)/\epsilon) \cdot L_\infty(\mathcal{D}).$$

Since `F1` is boosting from accuracy $(1/2 - \gamma)$ to accuracy $1/4$, Fact 15 implies that if $\mathcal{D}''$ is the distribution which `F1` passes to `WL`, then

$$L_\infty(\mathcal{D}'') = O(1) \cdot L_\infty(\mathcal{D}').$$

Combining these two equations, we find that

$$L_\infty(\mathcal{D}'') = O(\log(1/\epsilon)/\epsilon) \cdot L_\infty(\mathcal{D}).$$

□

Finally, we note that the final hypothesis which $\mathtt{B}_{\mathrm{Comb}}$ outputs is a depth 2 majority circuit over the weak hypotheses h_i, since both `F1` and $\mathtt{B}_{\mathrm{Filt}}$ combine their hypotheses using the majority function. A straightforward linear bound on the size of this majority circuit (see (Muller and Preparata, 1975)) yields the following hard-core set construction:

Theorem 21. Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0 < \gamma < 1/2$. Then there is a measure M on $\{0,1\}^n$ with $\mu(M) = \Omega(\epsilon(\log(1/\epsilon))^{-1})$ such that f is γ-hard-core on M for circuits of size $g' = O(\gamma^2(\log(1/\epsilon))^{-1}g)$.

4.4. From Hard-Core Measures to Hard-Core Sets

While we have been referring to our results thus far as hard-core set constructions, in fact Theorems 16, 18 and 21 establish the existence of hard-core measures rather than hard-core sets. We now show how hard-core measure constructions such as Theorems 16, 18 and 21 imply corresponding hard-core set constructions. This conversion from hard-core measures to hard-core sets is based on an argument from (Impagliazzo, 1995).

We will use the following crude lemma:

Lemma 22. The number of Boolean circuits of size g is at most $((n+5)g^2)^g$.

Proof: To specify a circuit it suffices to specify, for each of g gates, the two inputs and the label of the gate. For a given gate there are at most g^2 choices for the inputs and at most $n+5$ choices $(x_1, \ldots, x_n, \neg, \vee, \wedge, TRUE, FALSE)$ for the label. □

The following easy fact follows from the definition of $|M|$.

Fact 23. Let C be a circuit and view C and f as taking values in $\{-1,1\}$. Then $\Pr_{\mathcal{D}_M}[C(x) = f(x)] = \frac{1}{2} + \rho$ if and only if

$$\sum_{x \in \{0,1\}^n} M(x)C(x)f(x) = 2\rho|M|.$$

Now we state and prove our conversion from hard-core measures to hard-core sets. Constant factors have not been optimized in the following lemma.

Lemma 24. Let f be a Boolean function on $\{0,1\}^n$ and M a measure such that (i) $\mu(M) \geq \tau$ and (ii) f is γ-hard-core on M for size g where $g \leq \frac{1}{2n} \cdot 2^n\gamma^2\tau^2$. Then there exists a set S with $|S| \geq \frac{\tau}{2}2^n$ such that f is 4γ-hard-core on S for size g.

Proof: Let C be any circuit of size at most g. Consider the following randomized construction of a set $S \subseteq \{0,1\}^n$: for each $x \in \{0,1\}^n$ put x in S with probability $M(x)$. Let M_S be the characteristic function of

S. For each $x \in \{0,1\}^n$ the expected value of $M_S(x)$ is $M(x)$, and thus by linearity of expectation we have

$$E\left[\sum_{x\in\{0,1\}^n} M_S(x)C(x)f(x)\right] = \sum_{x\in\{0,1\}^n} M(x)C(x)f(x) \leq 2\gamma|M|$$

where the inequality follows from Fact 23 and f being γ-hard-core on M for size g. For each value of x the quantity $M_S(x)C(x)f(x)$ is a random variable in the interval $[-1,1]$. Hoeffding's tail bound now implies that

$$\begin{aligned}\Pr\left[\frac{1}{2^n}\sum_x M_S(x)C(x)f(x) \geq 4\gamma\mu(M)\right] &\leq \exp\left(\frac{-2\cdot 2^n(2\gamma\mu(M))^2}{4}\right)\\ &\leq \exp(-2\cdot 2^n\gamma^2\tau^2)\end{aligned}$$

where the second inequality is because $\mu(M) \geq \tau$. By Lemma 22 and the bound on g we have that the number of circuits of size at most g is at most $((n+5)g^2)^g \ll \frac{1}{10}\exp(2\cdot 2^n\gamma^2\tau^2)$. Thus the probability that there exists a C with $|C| \leq g$ such that $\sum_{x\in\{0,1\}^n} M_S(x)C(x)f(x) \geq 4\gamma|M|$ is less than $\frac{1}{10}$.

Meanwhile, we also have that $E[|S|] = |M|$ and $|S|$ is a sum of 2^n independent random variables each with range $\{0,1\}$. Thus Hoeffding's bound implies that

$$\begin{aligned}\Pr\left[\frac{|S|}{2^n} < \frac{\mu(M)}{2}\right] &\leq \exp(-2\cdot 2^n(\mu(M)/2)^2)\\ &\leq \exp(-2^n\tau^2/2).\end{aligned}$$

If $\tau < \frac{1}{2^{n/2}}$ then the bound on g in the statement of the lemma is less than $\frac{1}{2n}$ and the lemma is trivially true. Thus we assume that $\tau \geq \frac{1}{2^{n/2}}$ which by the above inequality implies that $\Pr[|S| \geq \frac{|M|}{2}] > \frac{1}{10}$.

These two probability bounds together imply that there exists some S with $|S| \geq \frac{|M|}{2} \geq \frac{\tau}{2}2^n$ such that for every circuit C with at most g gates

$$\sum_{x\in\{0,1\}^n} M_S(x)C(x)f(x) \leq 4\gamma|M| \leq 8\gamma|S| = 8\gamma|M_S|.$$

By Fact 23 we have that $\Pr_{x\in S}[C(x) = f(x)] \leq \frac{1}{2} + 4\gamma$ for every such circuit C, and thus f is 4γ-hard-core on S. □

Combining this lemma with Theorem 21 we obtain the following hard-core set construction:

Theorem 25. Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ where $g = O\left(\frac{2^n}{n} \cdot \frac{\epsilon^2}{\log 1/\epsilon}\right)$ and let $0 < \gamma < 1/2$. Then there is a set $S \subseteq \{0,1\}^n$ with $|S| = \Omega(\epsilon(\log 1/\epsilon)^{-1})2^n$ such that f is 4γ-hard-core on S for size $g' = O(\gamma^2(\log(1/\epsilon))^{-1}g)$.

4.5. Optimal Hard-Core Set Construction

Theorem 25 improves the circuit size parameter in Impagliazzo's construction from $O(\gamma^2\epsilon^2)g$ to $O(\gamma^2)(\log(1/\epsilon))^{-1})g$ but has a set size parameter of $\Omega(\epsilon(\log(1/\epsilon))^{-1})$ which is smaller, and hence worse, than Impagliazzo's bound of $\Omega(\epsilon)$. Having a large set size parameter is important for certain applications of hard-core set construction in derandomization such as constructing efficient pseudorandom generators (Sudan et al., 2001).

In this section we show how the set size parameter of Theorem 25 can be improved from $\Omega(\epsilon(\log 1/\epsilon)^{-1})$ to $\Omega(\epsilon)$. The basic idea is quite simple and was suggested by Avi Wigderson (Wigderson, 1999). Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let S be the hard-core set whose existence is asserted by Theorem 25. If $|S| = o(\epsilon 2^n)$ then we can take away the points in this hard-core set and intuitively f should still be almost ϵ-hard for circuits of size g under the uniform distribution on the remaining points. We can then reapply Theorem 25 on the remaining points using "almost ϵ" in place of ϵ to obtain a new hard-core set. This procedure can be repeated until the total size of all the hard-core sets is $\Omega(\epsilon 2^n)$.

To make this argument precise we will need to consider measures which are defined over proper subsets of $\{0,1\}^n$. If $X \subseteq \{0,1\}^n$ and $M : X \to [0,1]$ is a measure we write $|M|_X$ to denote $\sum_{x \in X} M(x)$ and $\mu_X(M)$ to denote $\frac{|M|_X}{|X|}$. We write $\mathcal{U}_X$ to denote the uniform distribution on X.

The following generalization of Theorem 25 is easily seen to follow from the arguments of Section 4.3.

Theorem 26. Fix $X \subseteq \{0,1\}^n$ such that $|X| \geq 2^n/2$. Let f be ϵ-hard for circuits of size g under $\mathcal{U}_X$ where $g = O\left(\frac{2^n}{n} \cdot \frac{\epsilon^2}{\log 1/\epsilon}\right)$ and let $0 < \gamma < 1/2$. Then there is a set $S \subset X$ with $|S| = \Omega(\epsilon(\log(1/\epsilon))^{-1})|X|$ such that f is 4γ-hard-core on S for size $g' = O(\gamma^2(\log(1/\epsilon))^{-1}g)$.

Proof: Identical to the proof of Theorem 25 with X in place of $\{0,1\}^n$ and $|X|$ in place of 2^n. Note that there is enough slack in the Hoeffding tail bounds of Lemma 24 for them to still go through as long as $|X| \geq 2^n/2$. □

We need the following easy lemma:

Lemma 27. If $X \subseteq \{0,1\}^n$ satisfies $|X| \geq (1-\frac{\epsilon}{2})2^n$ and f is ϵ-hard for circuits of size g under $\mathcal{U}$ then f is $\frac{\epsilon}{2}$-hard for circuits of size g under $\mathcal{U}_X$.

Proof: For any circuit C with $|C| \leq g$ we have that $|\{x : C(x) = f(x)\}| \leq (1-\epsilon)2^n$. Thus $\Pr_{\mathcal{U}_X}[C(x) = f(x)] \leq \frac{1-\epsilon}{1-\epsilon/2} < 1 - \frac{\epsilon}{2}$. □

Now we prove our strongest hard-core set construction.

Theorem 28. Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ where $g = O\left(\frac{2^n}{n} \cdot \frac{\epsilon^2}{\log 1/\epsilon}\right)$ and let $0 < \gamma < 1/2$. Then there is a set $S \subseteq \{0,1\}^n$ with $|S| \geq \frac{\epsilon}{2}2^n$ such that f is γ-hard-core on S for size $g' = O(\gamma^2(\log(1/\epsilon))^{-1}g)$.

Proof: Theorem 25 implies that there is a set S_0 with $|S_0| = \Omega(\epsilon(\log 1/\epsilon)^{-1})2^n$ such that f is γ-hard-core on S_0 for size $g' = O(\gamma^2(\log 1/\epsilon)^{-1}g)$. If $|S_0| \geq \frac{\epsilon}{2}2^n$ we are done, so we assume that $|S_0| < \frac{\epsilon}{2}2^n$ and let $X_1 = \{0,1\}^n \setminus S_0$. Lemma 27 implies that f is $\frac{\epsilon}{2}$-hard for circuits of size g under $\mathcal{U}_{X_1}$. Theorem 26 now implies the existence of a set $S_1 \subset X_1$ with $|S_1| = \Omega(\epsilon(\log 1/\epsilon)^{-1})|X_1|$ such that f is γ-hard-core on S_1 for size $g' = O(\gamma^2(\log(1/\epsilon))^{-1}g)$. Now let $X_2 = X_1 \setminus S_1$. Continue in this fashion, obtaining S_i from X_i as above, until $\sum |S_i| \geq \frac{\epsilon}{2}2^n$.

To see that this works, notice that until $\sum |S_i| \geq \frac{\epsilon}{2}2^n$ each set X_i satisfies $|X_i| > (1-\frac{\epsilon}{2})2^n$. Thus by Lemma 27 at each stage we have that f is $\frac{\epsilon}{2}$-hard for size g under $\mathcal{U}_{X_i}$, so we can apply Theorem 26 each time with hardness parameter $\epsilon/2$. Furthermore, we have that each $|S_i| = \Omega(\epsilon(\log(1/\epsilon))^{-1})|X_i|$ where $|X_i| > (1-\frac{\epsilon}{2})2^n$, so we must achieve $\sum |S_i| > \frac{\epsilon}{2}2^n$ after at most $O(\log 1/\epsilon)$ stages.

Now let $S = \cup_i S_i$. Since the S_i are disjoint we have $|S| \geq \frac{\epsilon}{2}2^n$. Moreover, since f is γ-hard-core on each $|S_i|$ for size g', for any circuit C of size at most g' we have

$$\Pr_{\mathcal{U}_S}[C(x) = f(x)] \;=\; \sum_i \left(\frac{|S_i|}{|S|} \Pr_{\mathcal{U}_{S_i}}[C(x) = f(x)]\right) \leq \frac{1}{2} + \gamma.$$

Thus f is γ-hard-core on S for size g' and the theorem is proved. □

Remark 1. The reader may have noticed that the two arguments used to go from a hard-core measure to a hard-core set and to increase the size of the hard-core set do not depend in any essential way on the fact that the initial hard-core measure was of size $\mu(M) = \Omega(\epsilon(\log 1/\epsilon)^{-1})$. We could also have obtained the same final result – a hard-core set

construction with $\epsilon/2$ as the set size parameter and $O(\gamma^2(\log 1/\epsilon)^{-1})$ as the circuit size parameter – by using the boosting algorithm F1 which implies the existence of a measure of size $\Omega(\epsilon^3)$ (Theorem 16). We introduced and analyzed the $\mathtt{B}_{\mathrm{Comb}}$ algorithm in part because this boosting algorithm will play an important role in the results of Section 5.

Remark 2. Freund has shown that any algorithm which boosts a $(1/2-\gamma)$-approximate weak learner to accuracy $1-\epsilon$ must combine at least $\Omega(\gamma^{-2}\log(1/\epsilon))$ weak hypotheses in the worst case (Freund, 1995). Thus, for any hard-core set construction falling within this framework our circuit size parameter is optimal up to constant factors. Furthermore, it is easy to see that any general hard-core set construction such as those we have given must have a set size parameter of at most $O(\epsilon)$, since the original ϵ-hard function f might be very easy to compute (e.g. constant; see (Shaltiel, 2001)) on a $1-O(\epsilon)$ fraction of inputs. Thus we believe that the hard-core set construction given by Theorem 28 is optimal up to constant factors with respect to both the circuit size and set size parameters.

4.6. A Boosting Algorithm from IHA

We have not yet described just how boosting algorithms manage to simulate the different distributions $\mathcal{D}_i$ for the example oracles $\mathrm{EX}(f,\mathcal{D}_i)$ which are required by the weak learning algorithm at each boosting stage. There are two different types of boosting algorithms, known as *boosting-by-filtering* and *boosting-by-sampling*, which handle this issue in different ways. In boosting-by-filtering the distribution $\mathcal{D}_i$ is simulated from $\mathcal{D}$ by filtering examples received from $\mathrm{EX}(f,\mathcal{D})$. If the filtering process accepts example x with probability $\alpha(x)$ and rejects x (i.e. discards x and makes another call to $\mathrm{EX}(f,\mathcal{D})$) with probability $1-\alpha(x)$, then it is easy to see that this filtering process defines a new distribution $\mathcal{D}'$ where

$$\mathcal{D}'(x) = \frac{\alpha(x)\mathcal{D}(x)}{\sum_y \alpha(y)\mathcal{D}(y)}. \tag{1}$$

The boosting algorithms F1, $\mathtt{B}_{\mathrm{Filt}}$ and $\mathtt{B}_{\mathrm{Comb}}$ all are boosting-by-filtering algorithms.

In boosting-by-sampling, on the other hand, a set S of examples is drawn from $\mathrm{EX}(f,\mathcal{D})$ once and for all at the beginning of the boosting process and the initial distribution $\mathcal{D}_0$ is taken to be the uniform distribution over S. Subsequent distributions $\mathcal{D}_i$ are nonzero only on the points of S, and the final hypothesis h generated by boosting has high accuracy with respect to the uniform distribution over S. Well-known

results on generalization error (Blumer et al., 1989) imply that if h belongs to a concept class with bounded Vapnik-Chervonenkis dimension, then for sufficiently large S any hypothesis h which is correct on all of S will with high probability have low error under $\mathcal{D}$. The `AdaBoost` algorithm of Freund and Schapire (Freund and Schapire, 1997) is an $O(\log(1/\epsilon)/\gamma^2)$-stage boosting-by-sampling algorithm; however, as discussed by Jackson in (Jackson, 1997), Adaboost does not seem to have any nontrivial smoothness properties.

Impagliazzo's proof shows that it is possible to use `IHA` directly as a $O(1/(\epsilon^2\gamma^2))$-stage, $1/\epsilon$-smooth boosting by sampling algorithm. Building on this algorithm, Servedio has given an improved boosting-by-sampling algorithm which is also $1/\epsilon$-smooth but uses only $O(1/(\epsilon\gamma^2))$ stages (Servedio,). In the next section we discuss using `IHA` as a $O(1/(\epsilon^2\gamma^2))$-stage, $O(1/\epsilon)$-smooth boosting by filtering algorithm in the case where the initial distribution $\mathcal{D}$ is uniform over $\{0,1\}^n$.

5. Faster Algorithms for Learning DNF

We have seen that boosting algorithms can be used to improve on previous complexity-theoretic hard-core set constructions. Now we go in the opposite direction and use ideas from hard-core set construction to establish new results in learning theory. We show that the uniform distribution boosting algorithm which is implicit in `IHA` can be used to significantly improve the asymptotic running time of Jackson's Harmonic Sieve algorithm for learning DNF under the uniform distribution using membership queries. This algorithm is widely viewed as one of the most important results in computational learning theory. We also show how a different modification inspired by our analysis in Section 4.3 can improve the running time even further at the cost of learning using more complex hypotheses.

Bshouty, Jackson and Tamon (Bshouty et al., 1999) have recently given a variant of the Harmonic Sieve which runs substantially faster than the original algorithm. Their improvement is obtained by speeding up a weak learning algorithm which is a component of the Harmonic Sieve, and is orthogonal to our improvements of the boosting component of the Sieve. As described below, by combining our techniques with their improvements we obtain the fastest known algorithm for learning DNF under the uniform distribution with membership queries.

5.1. The DNF Learning Problem

A *disjunctive normal form* (DNF) expression is a disjunction of terms where a term is a conjunction of Boolean literals. Since every Boolean function can be expressed as a DNF, the concept class DNF is the class of all Boolean functions over $\{0,1\}^n$. The *DNF-size* of a function f is the minimum number of terms in any DNF expression for f. Thus an efficient learning algorithm for the concept class DNF must be able to learn any Boolean function in time polynomial in the number of terms in its smallest DNF representation.

In his seminal paper Valiant posed the question of whether there is an efficient PAC algorithm for learning DNF under an arbitrary probability distribution on labeled examples (Valiant, 1984). A recent result (Klivans and Servedio,) shows that there is a PAC learning algorithm which learns to accuracy ϵ and runs in time $2^{O(n^{1/3}\log n \log s)}/\epsilon$ for target concepts of DNF-size s, but it is not yet known whether there is an algorithm which runs in time $\mathrm{poly}(n, s, 1/\epsilon)$. If the learning scenario is suitably modified, though, then efficient learning of DNF becomes possible. In a breakthrough result several years ago Jackson gave the Harmonic Sieve algorithm which uses *membership queries* to learn DNF to accuracy ϵ under the uniform distribution in $\mathrm{poly}(n, s, 1/\epsilon)$ time steps (Jackson, 1997). A membership query is an oracle query in which the learner specifies a point $x \in \{0,1\}^n$ and the membership oracle $\mathrm{MEM}(f)$ returns the value $f(x)$.

Although the Harmonic Sieve runs in polynomial time, it is not considered to be computationally practical due to the high degree of the polynomial time bound. We show how to substantially improve the algorithm's time dependence on the error parameter ϵ, thus making progress towards a more efficient implementation.

5.2. The Harmonic Sieve

The main result of (Jackson, 1997) is the following theorem:

Theorem 29. (Jackson, 1997) Let f be a Boolean function on $\{0,1\}^n$ of DNF-size s and let $0 < \epsilon, \delta < 1$. For any constant $\rho > 0$, given access to a membership oracle $MEM(f)$ the Harmonic Sieve algorithm runs in time $\tilde{O}(ns^{10}/\epsilon^{12+\rho})$ and with probability $1-\delta$ outputs a hypothesis h such that $\Pr_{\mathcal{U}}[h(x) \neq f(x)] \leq \epsilon$.[4]

At the heart of Jackson's Harmonic Sieve algorithm is a procedure `WDNF` which was first studied in (Blum et al., 1994). The `WDNF`

[4] In (Jackson, 1997) a running time of $\tilde{O}(ns^8/\epsilon^{12+\rho})$ is reported but this was in error as explained below (Jackson, 2002).

algorithm takes as input an example oracle $\mathrm{EX}(f,\mathcal{D})$, a membership oracle $\mathrm{MEM}(f)$, a distribution oracle $\mathrm{DIST}(\mathcal{D})$ and a value $\delta > 0$. A distribution oracle $\mathrm{DIST}(\mathcal{D})$ is an oracle which, when queried with a point x in the domain of $\mathcal{D}$, returns the value of $\mathcal{D}(x)$. With probability at least $1-\delta$ the `WDNF` algorithm outputs a parity function which is a $(1/2 - \Omega(1/s))$-approximator for f under $\mathcal{D}$, where s is the DNF-size of f.[5]

The Harmonic Sieve algorithm works by running Freund's boosting algorithm `F1` with `WDNF` as the weak learning algorithm. At the i-th stage of boosting the `F1` boosting algorithm simulates some distribution $\mathcal{D}_i$ and uses the simulated oracles $EX(f,\mathcal{D}_i)$ and $\mathrm{DIST}'(\mathcal{D}_i)$ for `WDNF`; here $\mathrm{DIST}'(\mathcal{D}_i)$ is a constant-factor approximation of $\mathrm{DIST}(\mathcal{D}_i)$[6]. The following lemma is a direct consequence of Jackson's Lemma 9 and the analysis used in its proof:

Lemma 30. (Jackson, 1997) Let f be any Boolean function of DNF-size s over $\{0,1\}^n$ and let $\mathcal{D}$ be any distribution over $\{0,1\}^n$. If `WDNF` is run using $\mathrm{EX}(f,\mathcal{D})$, $\mathrm{MEM}(f)$ and $\mathrm{DIST}'(\mathcal{D})$ as its oracles, then `WDNF` runs in

$$\tilde{O}(\mathrm{time}(\mathrm{DIST}'(\mathcal{D})) \cdot ns^6(L_\infty(2^n\mathcal{D}))^6 \; + \\ \mathrm{time}(\mathrm{EX}(f,\mathcal{D})) \cdot s^4(L_\infty(2^n\mathcal{D}))^2)$$

time steps and with probability at least $1-\delta$ outputs a parity function which is a $(1/2 - \Omega(1/s))$-approximator to f under $\mathcal{D}$.

It follows from Jackson's analysis that the `F1` boosting algorithm constructs each distribution $\mathcal{D}_i$ in such a way that after a "one-time" initial cost of $\tilde{O}(L_\infty(2^n\mathcal{D}_i)^2)$ time steps (to estimate the scaling factor in the denominator of Equation 1) for each $\mathcal{D}_i$, it is possible to simulate a constant-factor approximation $\mathrm{DIST}'(\mathcal{D}_i)$ of $\mathrm{DIST}(\mathcal{D}_i)$ in $\tilde{O}(i)$ time steps per call.[7] Jackson's analysis also implies that each call to $\mathrm{EX}(f,\mathcal{D}_i)$ made by `WDNF` can be simulated in time $\tilde{O}(iL_\infty(2^n\mathcal{D}_i))$ with high probability. Thus by Lemma 30, the time required for the i-th execution of `WDNF` on distribution $\mathcal{D}_i$ is bounded by $\tilde{O}(nis^6(L_\infty(2^n\mathcal{D}))^6)$.

[5] For $A \subseteq \{1,\ldots,n\}$ the parity function $\chi_A : \{0,1\}^n \to \{0,1\}$ is $\chi_A(x) = \sum_{i\in A} x_i \bmod 2$.

[6] To be more precise, the oracle $\mathrm{DIST}'(\mathcal{D}_i)$ is such that $\mathrm{DIST}'(\mathcal{D}_i)(x) = \alpha\cdot\mathrm{DIST}(\mathcal{D}_i)(x)$ for all x where $\alpha \in [\frac{1}{2},\frac{3}{2}]$ is some fixed constant. Jackson shows that access to DIST' is sufficient for `WDNF` to work successfully.

[7] In (Jackson, 1997) it is implicitly claimed that only constant time is required to simulate $\mathrm{DIST}'(\mathcal{D}_i)$ per call, but in fact $\tilde{O}(i)$ time is required since at the i-th stage of boosting there are i weak hypotheses which must be evaluated per call (Jackson, 2002).

As noted in Section 4.3, for any $\rho > 0$ algorithm F1 can be shown to be an $O(\gamma^{-2}\log(1/\epsilon))$-stage, $O(1/\epsilon^{2+\rho})$-smooth boosting algorithm, so $L_\infty(2^n\mathcal{D}_i) = O(1/\epsilon^{2+\rho})$ for every distribution $\mathcal{D}_i$ which WDNF uses. Moreover, $\gamma = \Omega(1/s)$ for the WDNF algorithm as stated earlier. It is clear that the running time of WDNF for each $\mathcal{D}_i$ dominates the "one-time cost" mentioned earlier which is incurred for each $\mathcal{D}_i$. Recalling the number of stages and smoothness of Freund's F1 booster, we have that the overall Harmonic Sieve algorithm runs in time $\tilde{O}(ns^{10}/\epsilon^{12+\rho})$ for any $\rho > 0$. The hypotheses output by the Harmonic Sieve are majority-of-parity circuits since each weak hypothesis is a parity circuit and F1's hypothesis is a majority circuit over weak hypotheses.

5.3. A Faster Version of the Harmonic Sieve

As described above, the Harmonic Sieve algorithm works by boosting under the uniform distribution and its running time strongly depends on the smoothness of the boosting algorithm. The following observation follows directly from the discussion of IHA in Section 2:

Observation 31. For each measure M_i constructed in the execution of IHA the distribution $\mathcal{D}_{M_i}$ is $1/\epsilon$-smooth.

This is substantially better than the distributions constructed by F1 which are guaranteed only to be $(1/\epsilon^{2+\rho})$-smooth. Thus, it appears that we can use the better boundedness of the IHA algorithm to obtain a faster version of the Harmonic Sieve, and indeed this turns out to be the case.

One detail which needs to be addressed, though, is that since the running time of WDNF depends on the quantity $L_\infty(2^n\mathcal{D}_i)$, where $\mathcal{D}_i$ is a distribution over $\{0,1\}^n$, we need a version of IHA which works efficiently over the entire domain $\{0,1\}^n$. Another way of saying this is that we need a boost-by-filtering version of IHA. Doing an exact computation of $\mu(M_i)$ in line 3 of IHA would take exponential time for the domain $\{0,1\}^n$, so our boost-by-sampling version of IHA instead estimates the value of $\mu(M_i)$ by using a sample average. More precisely, the algorithm draws a collection of uniformly distributed x's from $\{0,1\}^n$ and uses the observed average value of $M_i(x)$ on this sample as its estimate for $\mu(M_i)$. It is easy to see that $\mu(M_i)$ is the expected value of $M_i(x)$ for uniformly chosen x; standard bounds on sampling (e.g. Corollary 2.2 of (Jackson, 1995)) show that with very high probability an estimate $\mu'(M_i)$ satisfying $\frac{1}{2}\mu(M_i) \leq \mu'(M_i) \leq \frac{3}{2}\mu(M_i)$ can be found using $\tilde{O}(1/\mu(M_i)^2)$ samples. Thus in the boost-by-sampling version of IHA the test in line 3 can be approximately performed in $\tilde{O}(1/\epsilon^2)$ time steps, and the resulting algorithm will be $2/\epsilon$-smooth with high probability.

(We note that the $\tilde{O}(1/\mu(M_i)^2)$ time steps required to perform this estimation for each measure M_i corresponds exactly to the "one-time cost" of $\tilde{O}(L_\infty(2^n\mathcal{D}_i)^2)$ for estimating the denominator of Equation 1 which was required for each distribution $\mathcal{D}_i$ simulated by F1.) Thus IHA can be translated into a $O(1/(\gamma^2\epsilon^2))$-stage, $O(1/\epsilon)$-smooth boost-by-filtering algorithm under the uniform distribution.

As was the case with F1, the boost-by-sampling version of IHA will construct each distribution $\mathcal{D}_i$ in such a way that it is possible to simulate a constant-factor approximation $\mathrm{DIST}'(\mathcal{D}_i)$ of $\mathrm{DIST}(\mathcal{D}_i)$ in $\tilde{O}(i)$ time. Furthermore, as with F1 it is possible to simulate each call to $\mathrm{EX}(f,\mathcal{D}_i)$ in time $\tilde{O}(iL_\infty(2^n\mathcal{D}_i))$ with high probability. Thus by Lemma 30, the time required for the i-th execution of WDNF on distribution $\mathcal{D}_i$ is bounded by $\tilde{O}(nis^6(L_\infty(2^n\mathcal{D}))^6)$.

We refer to the modified Harmonic Sieve algorithm which uses IHA in place of F1 as HS′. Putting all the pieces together, we see that although HS′ requires a factor of $\tilde{\Omega}(1/\epsilon^2)$ more boosting stages than the original Sieve, this disadvantage is more than offset by the improved runtime of WDNF. We obtain the following:

Theorem 32. There is a membership-query algorithm HS′ for learning DNF under the uniform distribution which runs in time $\tilde{O}(ns^{10}/\epsilon^{10})$. The algorithm outputs as its final hypothesis a majority-of-parity circuit.

We can achieve an even faster variant of the Harmonic Sieve, at the price of using more complex hypotheses, by using the $\mathtt{B}_{\mathrm{comb}}$ boosting algorithm instead of the boost-by-filtering IHA algorithm. As noted in Section 4.2, $\mathtt{B}_{\mathrm{Comb}}$ is an $O(\gamma^{-2}\log(1/\epsilon))$-stage, $O(\log(1/\epsilon)/\epsilon)$-smooth boost-by-filtering algorithm. The smoothness of $\mathtt{B}_{\mathrm{Comb}}$ implies that the "one-time cost" of computing the scaling factor for each distribution $\mathcal{D}_i$ is now $\tilde{O}(L_\infty(2^n\mathcal{D}_i)^2) = \tilde{O}(1/\epsilon^2)$. As before, the time required to simulate $\mathrm{DIST}'(\mathcal{D}_i)$ is $\tilde{O}(i)$, and the time required to simulate $\mathrm{EX}(f,\mathcal{D}_i)$ is $\tilde{O}(iL_\infty(2^n\mathcal{D}_i))$. ¿From Lemma 30 we find that if $\mathtt{B}_{\mathrm{Comb}}$ is used as the boosting algorithm in the Sieve, then the total running time of each boosting stage will be at most $\tilde{O}(ns^8/\epsilon^6)$. Since we boost for at most $O(s^2\log(1/\epsilon))$ stages under $\mathtt{B}_{\mathrm{Comb}}$, we have the following theorem:

Theorem 33. There is a membership-query algorithm for learning s-term DNF formulae under the uniform distribution on $\{0,1\}^n$ which runs in time $\tilde{O}(ns^{10}/\epsilon^6)$. The algorithm outputs as its final hypothesis a majority-of-majority-of-parity circuit.

The additional circuit complexity comes from the fact that the hypothesis output by $\mathtt{B}_{\mathrm{Comb}}$ is a depth 2 majority circuit over its inputs.

5.4. Extensions

Throughout this section we have only discussed using the Harmonic Sieve to learn DNF formulae under the uniform distribution. Jackson generalizes the algorithm to several other concept classes including TOP (polynomial-size depth-2 circuits where the top gate computes majority and the bottom gates compute parities) and unions of axis-parallel rectangles over $\{0, 1, \ldots, b\}^n$ for constant b. In each case our approach can be used to improve the running time of Jackson's algorithms.

In recent work Bshouty, Jackson and Tamon (Bshouty et al., 1999) have given a new version of the Harmonic Sieve for learning DNF under the uniform distribution. The new algorithm differs from the original Harmonic Sieve in that it uses a faster version of the WDNF algorithm. Implicit in their analysis is the following lemma:[8]

Lemma 34. (Bshouty et al., 1999) Let f be any Boolean function of DNF-size s over $\{0,1\}^n$ and let $\mathcal{D}$ be any distribution over $\{0,1\}^n$. There is an algorithm WDNF$'$ which takes as input an example oracle $\mathrm{EX}(f, \mathcal{D})$, a membership oracle $\mathrm{MEM}(f)$, a distribution oracle $\mathrm{DIST}'(\mathcal{D})$, and a value $\delta > 0$. In the context of the Harmonic Sieve, each invocation of WDNF$'$ with the i-th distribution $\mathcal{D}_i$ generated by the boosting algorithm takes $\tilde{O}(nis^2(L_\infty(2^n\mathcal{D}_i))^2)$ time steps and and with probability at least $1 - \delta$ outputs a parity function which is a $(1/2 - \Omega(1/s))$-approximator to f under $\mathcal{D}_i$.

The Harmonic Sieve variant described in (Bshouty et al., 1999) runs the original $O(1/\epsilon^{2+\rho})$-smooth F1 boosting algorithm for $\tilde{O}(s^2)$ stages, using the new WDNF$'$ algorithm as the weak learner, to obtain an overall running time of essentially $\tilde{O}(ns^6/\epsilon^4)$. By instead using the $O(\log(1/\epsilon)/\gamma^2)$-stage, $O(\log(1/\epsilon)/\epsilon)$-smooth boosting algorithm $\mathtt{B}_{\mathrm{comb}}$ as in Section 5.3, we obtain the following result, which is the fastest known algorithm for learning DNF under the uniform distribution using membership queries:

Theorem 35. There is a membership-query algorithm for learning s-term DNF formulae over $\{0,1\}^n$ under the uniform distribution which runs in time $\tilde{O}(ns^6/\epsilon^2)$. The algorithm outputs as its final hypothesis a majority-of-majority-of-parity circuit.

[8] The error mentioned in Section 5.2 in Jackson's analysis of the Harmonic Sieve also appears in the recent work of Bshouty, Jackson and Tamon. Lemma 34 and Theorem 35 incorporate a correction: the WDNF$'$ algorithm takes time $\tilde{O}(nis^2(L_\infty(2^n\mathcal{D}_i))^2)$ as opposed to $\tilde{O}(ns^2(L_\infty(2^n\mathcal{D}_i))^2)$ as is stated in (Bshouty et al., 1999).

6. Acknowledgements

We thank Jeff Jackson for useful conversations concerning the sectiontion on learning DNF formulae. We thank Salil Vadhan for insights on (Impagliazzo, 1995). We would like to thank Amos Beimel, Venkatesan Guruswami, Salil Vadhan and Les Valiant for helpful comments on an early version of this paper.

References

Babai, L., L. Fortnow, N. Nisan, and A.Wigderson: 1993, 'BPP has subexponential time simulations unless EXPTIME has publishable proofs'. *Computational Complexity* **3**, 307–318.

Blum, A., M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich: 1994, 'Weakly learning DNF and characterizing statistical query learning using Fourier analysis'. In: *Proceedings of the Twenty-Sixth Annual Symposium on Theory of Computing*. pp. 253–262.

Blumer, A., A. Ehrenfeucht, D. Haussler, and M. Warmuth: 1989, 'Learnability and the Vapnik-Chervonenkis dimension'. *Journal of the ACM* **36**(4), 929–965.

Boneh, D. and R. Lipton: 1993, 'Amplification of weak learning over the uniform distribution'. In: *Proceedings of the Sixth Annual Workshop on Computational Learning Theory*. pp. 347–351.

Bshouty, N., J. Jackson, and C. Tamon: 1999, 'More efficient PAC learning of DNF with membership queries under the uniform distribution'. In: *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*. pp. 286–295.

Drucker, H. and C. Cortes: 1996, 'Boosting decision trees'. In: *Advances in Neural Information Processing Systems 8*. pp. 479–485.

Drucker, H., C. Cortes, L. D. Jackel, Y. Lecun, and V. Vapnik: 1994, 'Boosting and other ensemble methods'. *Neural Computation* **6**(6), 1289–1301.

Drucker, H., R. Schapire, and P. Simard: 1993a, 'Boosting performance in neural networks'. *International Journal of Pattern Recognition and Machine Intelligence* **7**(4), 705–719.

Drucker, H., R. Schapire, and P. Simard: 1993b, 'Improving performance in neural networks using a boosting algorithm'. In: *Advances in Neural Information Processing Systems 5*. pp. 42–49.

Freund, Y.: 1990, 'Boosting a weak learning algorithm by majority'. In: *Proceedings of the Third Annual Workshop on Computational Learning Theory*. pp. 202–216.

Freund, Y.: 1992, 'An improved boosting algorithm and its implications on learning complexity'. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. pp. 391–398.

Freund, Y.: 1995, 'Boosting a weak learning algorithm by majority'. *Information and Computation* **121**(2), 256–285.

Freund, Y. and R. Schapire: 1997, 'A decision-theoretic generalization of on-line learning and an application to boosting'. *Journal of Computer and System Sciences* **55**(1), 119–139.

Goldreich, O., N. Nisan, and A. Wigderson: 1995, 'On Yao's XOR-Lemma'. Electronic Colloquium on Computational Complexity TR95-050.

Impagliazzo, R.: 1995, 'Hard-core distributions for somewhat hard problems'. In: *Proceedings of the Thirty-Sixth Annual Symposium on Foundations of Computer Science*. pp. 538–545.

Impagliazzo, R. and A. Widgerson: 1997, 'P = BPP unless E has subexponential circuits: derandomizing the XOR lemma'. In: *Proceedings of the Twenty-Ninth Annual Symposium on Theory of Computing*. pp. 220–229.

Jackson, J.: 1995, 'The Harmonic sieve: a novel application of Fourier analysis to machine learning theory and practice'. Ph.D. thesis, Carnegie Mellon University.

Jackson, J.: 1997, 'An efficient membership-query algorithm for learning DNF with respect to the uniform distribution'. *Journal of Computer and System Sciences* **55**, 414–440.

Jackson, J.: 2002. Personal communication.

Jackson, J. and M. Craven: 1996, 'Learning sparse perceptrons'. In: *Advances in Neural Information Processing Systems 8*. pp. 654–660.

Kearns, M. and L. Valiant: 1994, 'Cryptographic limitations on learning Boolean formulae and finite automata'. *Journal of the ACM* **41**(1), 67–95.

Klivans, A. and R. Servedio, 'Learning DNF in time $2^{\tilde{O}(n^{1/3})}$'. In: *Proceedings of the Twenty-Sixth Annual Symposium on Theory of Computing*.

Levin, L.: 1986, 'Average case complete problems'. *SIAM Journal on Computing* **15**(1), 285–286.

Muller, D. and F. Preparata: 1975, 'Bounds to complexities of networks for sorting and for switching'. *Journal of the ACM* **22**(2), 195–201.

Nisan, N. and A. Wigderson: 1994, 'Hardness versus randomness'. *Journal of Computer and System Sciences* **49**, 149–167.

Schapire, R.: 1990, 'The strength of weak learnability'. *Machine Learning* **5**(2), 197–227.

Schapire, R. and Y. Singer: 1998, 'Improved boosting algorithms using confidence-rated predictions'. In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. pp. 80–91.

Servedio, R., 'Smooth boosting and learning with malicious noise'.

Shaltiel, R.: 2001, 'Towards proving strong direct product theorems'. In: *Proceedings of the Sixteenth Conference on Computational Complexity*. pp. 107–117.

Sudan, M., L. Trevisan, and S. Vadhan: 2001, 'Pseudorandom generators without the XOR lemma'. *Journal of Computer and System Sciences* **62**(2), 236–266.

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

Wigderson, A.: 1999. Personal communication.