Boosting and Hard-Core Sets

Adam R. Klivans*
Department of Mathematics
MIT
Cambridge, MA 02139
klivans@math.mit.edu

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

Abstract

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. 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 [15], 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 [15] 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.*

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. Hardness amplification results are a crucial component of recent attempts to derandomize BPP [24, 3, 16]. Perhaps the most famous hardness amplification result is Yao's XOR-lemma [14], 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 [22] can only do this for some $1/2 > \epsilon > 0$. Schapire [25] and then Freund [10, 11] 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 [6, 7, 8, 9, 13, 20, 26]. All known boosting algorithms work by using the weak learning algorithm several times on a sequence of carefully constructed distributions.

*Supported in part by NSF grant CCR-97-01304.
†Supported in part by an NSF Graduate Fellowship, by NSF grant CCR-95-04436 and by ONR grant N00014-96-1-0550.

Reference:	Set size parameter:	Circuit size parameter:
Impagliazzo [15]	ϵ	$O(\gamma^2\epsilon^2)\cdot g$
Nisan [15]	ϵ	$O(\gamma^2(\log(1/\gamma\epsilon))^{-1})\cdot g$
This Paper	$\epsilon/O(\log(1/\epsilon))$	$O(\gamma^2(\log(1/\epsilon))^{-1})\cdot g$

Table 1. Comparison of known hard-core set constructions.

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 [15] 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 2.1; we thus refer to Impagliazzo's method of constructing such a distribution as a hard-core set construction. Schapire [25] was the first to point out that the existence of a boosting algorithm implies the existence of such a distribution.

1.2. Our Results[1]

In this paper 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 [15], Impagliazzo proves the following: 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 loss in the set size parameter. As explained in Section 4.3, we believe our circuit size parameter to be optimal with respect to this class of techniques. Table 1 summarizes our hard-core set construction results.

We also show how to use Impagliazzo's hard-core set construction to obtain a new variant of Jackson's breakthrough Harmonic Sieve algorithm [17] for learning DNF formulae with membership queries under the uniform distribution. Our variant is substantially more efficient than the original algorithm. Jackson's original algorithm runs in time $\tilde{O}(ns^8/\epsilon^{12})$, where n is the number of variables in the DNF formula, s is the number of terms, and ϵ is the accuracy parameter; our variant runs in time $\tilde{O}(ns^8/\epsilon^8)$. (We can further improve the running time to $\tilde{O}(ns^8/\epsilon^6)$ at the cost of learning using a slightly more complicated class of hypotheses).

In recent work Bshouty, Jackson and Tamon [5] have improved the running time of the Harmonic Sieve to $\tilde{O}(rs^4/\epsilon^4)$, where r is the number of distinct variables which appear in the minimal DNF representation of the target formula. Our results improve the running time of their new algorithm to $\tilde{O}(rs^4/\epsilon^2)$ 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] Our measure size parameter can be improved to ϵ. This improvement will appear in the final version.

1.3. Related Work

Boneh and Lipton [4] 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 [15]. 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 DNF formulae. We also mention 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 [15], formalizes the notion of a function which is hard to approximate. (Readers who are familiar with the notation of [15] 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 $\mathcal{U}$ 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 any 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 [15]:

Definition 4 *We say that f is* γ-hard-core on M for size g *if $\Pr_{\mathcal{D}_M}[f(x) = C(x)] \leq 1/2 + \gamma$ 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 [15], is the starting point of all our results:

Theorem 5 [15] *Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0 < \gamma < 1$. Then there is a measure M on $\{0,1\}^n$ with $\mu(M) \geq \epsilon$ such that f is γ-hard-core on M for size $g' = O(\epsilon^2\gamma^2)g$.*

Proof Sketch: 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 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 nonpositive, and intermediate weight to points where the margin is positive but small.

Impagliazzo proves that after at most $i_0 = O(1/(\epsilon^2\gamma^2))$ cycles through the loop, $\mu(M_i)$ 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

Input: $\epsilon > 0, \gamma > 0$, boolean function f
Output: a circuit h such that $\Pr_{\mathcal{U}}[h(x) = f(x)] \geq 1 - \epsilon$

set $i \leftarrow 0$
$M_0(x) \equiv 1$
until $\mu(M_i) < \epsilon$ **do**
    let C_i be a circuit of size at most g' with $\Pr_{\mathcal{D}_{M_i}}[C(x) = f(x)] \geq 1/2 + \gamma$
    $R_{C_i}(x) \equiv 1$ if $f(x) = C_i(x)$, $R_{C_i}(x) \equiv -1$ otherwise
    $N_i(x) \equiv \sum_{0 \leq j \leq i} R_{C_j}(x)$
    $M_{i+1}(x) \equiv 0$ if $N_i(x) \geq 1/\epsilon\gamma$, $M_{i+1}(x) \equiv 1$ if $N_i(x) \leq 0$, $M_{i+1}(x) \equiv 1 - \epsilon\gamma N_i(x)$ otherwise
    set $i \leftarrow i + 1$
$h \equiv MAJ(C_0, C_1, \ldots, C_{i-1})$
return h

Figure 1. The `IHA` **algorithm.**

i_0 circuits each of size at most g', and majority over i_0 inputs can be computed by a circuit of size $O(i_0)$. 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 [15]. 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 6 [15] *Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0 < \gamma < 1$. Then there is a measure M on $\{0,1\}^n$ 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 (but a slightly smaller measure) than either Theorem 5 or Theorem 6.

(We note that Theorems 5 and 6, as well as the theorems which we will prove later, assert the existence of a large measure, not a large set as was promised in Section 1. Using a straightforward probabilistic argument, Impagliazzo has shown in [15] that if f is γ-hard-core on M for size g' with $\mu(M) \geq \epsilon$, then there is a set $S \subseteq \{0,1\}^n$ with $|S| \geq \epsilon 2^n$ such that f is 2γ-hard-core on S for size g'.)

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 [27]. 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 7 *An algorithm* `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* `A` *is given access to n, ϵ, δ and EX$(f, \mathcal{D})$, then* `A` *runs in time polynomial in n, ϵ^{-1}, δ^{-1}, and $size(f)$, and with probability at least $1 - \delta$ algorithm* `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 `A` is only guaranteed to find a $(1/2 - \gamma)$-approximator for some $\gamma > 0$, then we say that `A` is a $(1/2 - \gamma)$-*approximate* learning algorithm; if $\gamma = \Omega(1/p(n, size(f)))$ for some polynomial p, we say that `A` is a *weak* learning algorithm (The notion of weak learning was introduced by Kearns and Valiant in [22]). We will abuse notation and say that `A` is a $(1/2 - \gamma)$-approximate learning algorithm for f if `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 [25] and subsequently Freund [10, 11] have shown that if `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 [12]):

Definition 8 *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 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 EX$(f, \mathcal{D}')$. We give the following high-level definition:

Definition 9 *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 a $(1/2-\gamma)$-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 EX$(f, \mathcal{D}_i)$ and lets* WL *access this simulated example oracle to produce a hypothesis h_i which is a $(1/2-\gamma)$-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 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 $\Pr_{\mathcal{D}_{M_i}}[f(x) \neq C(x)] \leq 1/2-\gamma$, 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 10 *Let $\mathcal{D}$ be a distribution over $\{0,1\}^n$. For $d \geq 1$, we say that $\mathcal{D}$ is d-*bounded *if $L_\infty(2^n\mathcal{D}) \leq d$.*

As an immediate consequence of Definitions 2 and 10, we have

Observation 11 *A distribution $\mathcal{D}$ is d-bounded iff $\mu(M_{\mathcal{D}}) \geq 1/d$.*

Definition 12 *Let* B *be a canonical boosting algorithm which takes as input ϵ, δ, γ, an example oracle 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 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 $d(\epsilon, \gamma)$-*bounded *boosting algorithm if the following holds: For all functions f and $(1/2-\gamma)$-approximate learners* WL*, when* B *is given EX$(f, \mathcal{U})$ and* WL*, with nonzero probability both of the following events occur: (i) the simulated distributions $\mathcal{D}_0, \ldots, \mathcal{D}_{k-1}$ are each $d(\epsilon, \gamma)$-bounded, and (ii) the hypothesis h which* B *outputs satisfies $\Pr_{\mathcal{D}}[f(x) = h(x)] \geq 1-\epsilon$.*

Note that the property of the distributions $\mathcal{D}_i$ described in part 2 of the above definition is similar (but not identical) to Levin's notion of "dominated" distributions [23].

Now we can state the following theorem which generalizes Impagliazzo's hard-core set construction from [15].

Theorem 13 *Let B be a $k(\epsilon,\gamma)$-stage, $d(\epsilon,\gamma)$-bounded 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 $0<\gamma<1$. 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 11, this implies that for every $d(\epsilon,\gamma)$-bounded distribution $\mathcal{D}$ there is a circuit $C_{\mathcal{D}}$ of size at most g' such that $\Pr_{\mathcal{D}}[f(x)=C_{\mathcal{D}}(x)]\geq 1/2+\gamma$.

Now run the boosting algorithm `B` on inputs ϵ,δ,γ, and $\mathrm{EX}(f,\mathcal{U})$. Since `B` is $d(\epsilon,\gamma)$-bounded, with nonzero probability we have that (i) every distribution $\mathcal{D}_i$ which `B` simulates will be $d(\epsilon,\gamma)$-bounded, 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 13 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 subsumes and improves results of Impagliazzo and Nisan in the circuit size parameter and has a slightly worse measure size parameter.

We first consider Freund's boost-by-majority algorithm from [10] which, following Jackson [18], we refer to as `F1`. Algorithm `F1` simulates at most $k=O(\gamma^{-2}\log(1/\epsilon))$ distributions $\mathcal{D}_i$ and combines its k hypotheses using the majority function. Jackson's analysis ([18], pp. 57–59) yields the following fact about `F1`:

Fact 14 *If `F1` is given inputs ϵ,δ,γ, $EX(f,\mathcal{D})$ and a $(1/2-\gamma)$-approximate weak learner `WL` for f, then with high probability each distribution $\mathcal{D}'$ which `F1` simulates for `WL` satisfies*

$$L_\infty(\mathcal{D}')=O(1/\epsilon^2)\cdot L_\infty(\mathcal{D}).$$

This immediately implies that `F1` is $O(1/\epsilon^2)$-bounded. We thus obtain the following hard-core set construction:

Theorem 15 *Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0<\gamma<1$. Then there is a measure M on $\{0,1\}^n$ with $\mu(M)=\Omega(\epsilon^2)$ 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 $\mathtt{B}_{\text{Filt}}$ algorithm from [12] (the name comes from the fact that the algorithm "filters" examples from the original distribution to simulate new distributions). Like `F1`, algorithm $\mathtt{B}_{\text{Filt}}$ is a k-stage boosting algorithm for $k=O(\gamma^{-2}\log(1/\epsilon))$. $\mathtt{B}_{\text{Filt}}$ 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 a ϵ-approximator for f. To analyze the boundedness of $\mathtt{B}_{\text{Filt}}$, we use the following fact which is implicit in [12]:

Fact 16 *If $\mathtt{B}_{\text{Filt}}$ is given inputs ϵ,δ,γ, $EX(f,\mathcal{D})$ and a $(1/2-\gamma)$-approximate weak learner `WL` for f, then with high probability each distribution $\mathcal{D}'$ which $\mathtt{B}_{\text{Filt}}$ simulates for `WL` satisfies*

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

Since Fact 16 implies that $\mathtt{B}_{\text{Filt}}$ is $O(\log(1/\epsilon)/(\epsilon\gamma))$-bounded, we obtain

Theorem 17 *Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0<\gamma<1$. 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 size $g'=O(\gamma^2(\log(1/\epsilon))^{-1}g)$.*

Finally we establish our strongest hard-core set construction by combining the previous two approaches. In [11], 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 $\mathtt{B}_{\text{Filt}}$ boosts from accuracy $1/4$ to accuracy ϵ by taking `F1` as its weak learner. We call this combined algorithm $\mathtt{B}_{\text{Comb}}$.

Lemma 18 *The boosting algorithm $\mathtt{B}_{\text{Comb}}$ is an $O(\gamma^{-2}\log(1/\epsilon))$-stage boosting algorithm.*

Proof: The top level of $\mathtt{B}_{\text{Comb}}$, which uses algorithm $\mathtt{B}_{\text{Filt}}$, 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 $\mathtt{B}_{\text{Comb}}$ uses the claimed number of stages. ■

Lemma 19 *$\mathtt{B}_{Comb}$ is an $O(\log(1/\epsilon)/\epsilon)$-bounded boosting algorithm.*

Proof: Since $\mathtt{B}_{\text{Filt}}$ is boosting from accuracy $1/4$ to accuracy ϵ using F1 as its weak learner, Fact 16 implies that each distribution $\mathcal{D}'$ which $\mathtt{B}_{\text{Filt}}$ 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 14 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 $\mathtt{B}_{\text{Comb}}$ outputs is a depth 2 majority circuit over the weak hypotheses h_i, since both F1 and $\mathtt{B}_{\text{Filt}}$ combine their hypotheses using the majority function. A straighforward bound on the size of this majority circuit yields our strongest hard-core set construction:

Theorem 20 *Let f be ϵ-hard for circuits of size g under $\mathcal{U}$ and let $0 < \gamma < 1$. 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 size $g' = O(\gamma^2(\log(1/\epsilon))^{-1}g)$.*

Freund [12] has shown that any successful boosting algorithm must combine at least $\Omega(\gamma^{-2}\log(1/\epsilon))$ weak hypotheses to achieve error less than ϵ (this matches the upper bound given in Lemma 18). Thus, for any hard-core set construction falling within this framework our circuit size parameter is optimal.

4.4. A Boosting Algorithm from IHA

We note that Impagliazzo's proof shows that IHA is a boosting algorithm under the uniform distribution, not under an arbitrary initial distribution. In fact, it is possible to extend IHA to obtain a true boosting algorithm which can be used under any initial distribution. This is done by using standard boost-by-sampling techniques [11, 13]; the basic idea is to draw a sample S of examples from the initial distribution $\mathcal{D}$ and then run IHA on the uniform distribution over S to obtain a hypothesis h which is correct on all points in S. Well-known results on PAC learning and the Vapnik-Chervonenkis dimension [2] imply that if h belongs to a concept class which is "sufficiently simple" (has low Vapnik-Chervonenkis dimension), then for sufficiently large S any hypothesis h which is correct on all points of S will with high probability have low error under $\mathcal{D}$.

5. Faster Algorithms for Learning DNF

In the previous section we saw how boosting algorithms can be used to obtain new hard-core set constructions. In this section we go in the opposite direction and establish new results in learning theory based on Impagliazzo's hard-core set construction. We show that the uniform distribution boosting algorithm which is implicit in IHA can be used significantly improve the running time of Jackson's Harmonic Sieve algorithm for learning DNF under the uniform distribution using membership queries, which 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.

Very recently Bshouty, Jackson and Tamon [5] have given a variant of the Harmonic Sieve algorithm 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. 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 each term is a conjunction of boolean literals. Since every boolean function can be expressed in this form, 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, a 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 1984 paper [27], Valiant posed the question of whether there is a strong PAC learning algorithm for DNF. The lack of progress on this question led researchers to consider weaker learning models (giving more power to the learning algorithm or relaxing the criteria for successful learning) in the hope of proving some positive result. One way of giving more power to the learning algorithm is by allowing it to make *membership queries*. A membership query is an oracle query in which the learner specifies a point x and the membership oracle MEM(f) returns the value $f(x)$ of the unknown target function on x. Another relaxation of the PAC model is to require that the learning algorithm succeed not for an arbitrary distribution but only under the uniform distribution.

In a breakthrough result ten years after Valiant's paper, Jackson [17] gave an algorithm, the Harmonic Sieve, which uses membership queries to learn DNF in polynomial time under the uniform distribution. Although his algorithm runs in polynomial time, it is not considered to be computationally practical. In this section we show how to substantially improve the algorithm's time dependency on the error parameter ϵ, thus making progress towards a more efficient implementation.

5.2. An Overview of the Harmonic Sieve

Jackson proves the following theorem:

Theorem 21 **[19]** *The class of DNF formulae over* $\{0,1\}^n$ *is strongly learnable under the uniform distribution using membership queries in time* $\tilde{O}(ns^8/\epsilon^{12})$ *where* s *is the DNF-size of the target function* f *and* ϵ *is the accuracy parameter. The algorithm outputs as its final hypothesis a majority-of-parity circuit.*

At the heart of Jackson's Harmonic Sieve algorithm is a procedure `WDNF` [1] which uses queries to MEM(f) as well as calls to the example oracle EX$(f,\mathcal{D})$ for weakly learning DNF (see Appendix A for a more detailed description of the `WDNF` algorithm). Jackson proves the following:

Lemma 22 **[19]** *For any boolean function* f *of DNF-size* s *over* $\{0,1\}^n$ *and any distribution* $\mathcal{D}$, *algorithm* `WDNF` *runs in time* $\tilde{O}(ns^6(L_\infty(2^n\mathcal{D}))^6)$ *and outputs a parity function which is a* $(1/2-\Omega(1/s))$*-approximator to* f *under* $\mathcal{D}$.

Proof Sketch of Theorem 21: The Harmonic Sieve algorithm works by applying Freund's booster `F1` to `WDNF`. Since `F1` is an $O(\gamma^{-2}\log(1/\epsilon))$-stage, $O(\epsilon^{-2})$-bounded boosting algorithm, it follows that under the uniform distribution, $L_\infty(2^n\mathcal{D}) = O(\epsilon^{-2})$ for every distribution which `WDNF` will receive. Consequently, the Harmonic Sieve algorithm runs in time $\tilde{O}(ns^8/\epsilon^{12})$. The hypotheses output by the Harmonic Sieve are majority-of-parity circuits since each weak hypothesis is a parity circuit and `F1` takes the majority. ■

5.3. A Faster Version of the Harmonic Sieve

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

Observation 23 *For each measure* M_i *constructed in the execution of* `IHA`*, the distribution* $\mathcal{D}_{M_i}$ *is* $1/\epsilon$*-bounded.*

Since `IHA` is guaranteed to end after $O(\gamma^{-2}\epsilon^{-2})$ cycles through the loop, it follows that the `IHA` algorithm can be translated directly into a a $O(\gamma^{-2}\epsilon^{-2})$-stage, $1/\epsilon$-bounded boosting algorithm under the uniform distribution. Thus, it can be used instead of `F1` in the top layer of the Harmonic Sieve. We call this modified algorithm `HS`$'$. Although `HS`$'$ requires a factor of $\tilde{\Omega}(\epsilon^{-2})$ more boosting stages than `F1`, this is more than made up for by the better boundedness of `HS`$'$, which results in each execution of `WDNF` taking at most $\tilde{O}(ns^6/\epsilon^6)$ time steps. Thus we obtain the following:

Theorem 24 *There is a membership-query algorithm* `HS`$'$ *for learning DNF under the uniform distribution which runs in time* $\tilde{O}(ns^8/\epsilon^8)$. *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 $\mathtt{B}_{\text{Comb}}$ boosting algorithm. As noted in Section 4.2, $\mathtt{B}_{\text{Comb}}$ is an $O(\gamma^{-2}\log(1/\epsilon))$-stage, $O(\log(1/\epsilon)/\epsilon)$-bounded boosting algorithm. Thus, if we use $\mathtt{B}_{\text{Comb}}$ as our boosting algorithm, the runinng time of each execution of `WDNF` will still be at most $\tilde{O}(ns^6/\epsilon^6)$ (here the $\tilde{O}$-notation is hiding a larger polylogarithmic factor). Since we boost for at most $O(s^2\log(1/\epsilon))$ stages, we have the following theorem:

Theorem 25 *There is a membership-query algorithm for learning DNF formulae under the uniform distribution which runs in time* $\tilde{O}(ns^8/\epsilon^6)$. *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 $\mathtt{B}_{\text{Comb}}$ 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 [19] generalizes the algorithm to several other concept classes including TOP (majority-of-parity circuits) and unions of axis-parallel rectangles over $\{0,1,\ldots,b\}^n$. In each case our techniques can be used to improve the running time of his algorithms.

We also note that in recent work, Bshouty, Jackson and Tamon [5] have given a new algorithm 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. This new version of WDNF runs in $\tilde{O}(rs^2(L_\infty(2^n\mathcal{D}))^2)$ time steps, where r is the number of distinct variables which appear in the minimal DNF representation of the target formula. Bshouty, Jackson and Tamon run the original $O(\epsilon^{-2})$-bounded 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 $\tilde{O}(rs^4/\epsilon^4)$ for learning DNF. By applying our techniques as in Section 5.3 (using the $\tilde{O}(\log(1/\epsilon)/\epsilon)$-bounded boosting algorithm B_{Comb}), we can improve the running time of the algorithm to $\tilde{O}(rs^4/\epsilon^2)$.

6. Acknowledgements

We thank Jeff Jackson for useful conversations concerning the section on learning DNF formulae. We thank Salil Vadhan for insights on [15]. We would like to thank Amos Beimel, Venkatesan Guruswami, Salil Vadhan and Les Valiant for helpful comments on an early version of this paper.

A. The WDNF Algorithm

The WDNF algorithm takes as input an example oracle $\text{EX}(f, \mathcal{D})$, a membership oracle $\text{MEM}(f)$, a distribution oracle $\text{DIST}(\mathcal{D})$, and a value $\delta > 0$. A *distribution oracle* $\text{DIST}(\mathcal{D})$ is an oracle which, when queried with a point x in the domain of $\mathcal{D}$, returns $\mathcal{D}(x)$. All of the boosting algorithms F1, B_{Filt}, and B_{Comb}, as well as the uniform-distribution boosting algorithm implicit in IHA, construct their distributions $\mathcal{D}_i$ in such a way that they can efficiently simulate $\text{DIST}(\mathcal{D}_i)$. 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.

References

[1] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. *Weakly learning DNF and characterizing statistical query learning using Fourier analysis.* In "26th Annual Symposium on Theory of Computing," (1994), pp. 253-262.

[2] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. *Learnability and the Vapnik-Chervonenkis Dimension.* J. ACM 36(4) (1989), pp. 929-965.

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

[4] D. Boneh and R. Lipton. *Amplification of weak learning over the uniform distribution.* In "Sixth Annual Workshop on Computational Learning Theory," (1993), pp. 347-351.

[5] N. Bshouty, J. Jackson, and C. Tamon. *More efficient PAC-learning of DNF with membership queries under the uniform distribution.* In "Twelfth Annual Conference on Computational Learning Theory," (1999), pp. 286-295.

[6] H. Drucker and C. Cortes. *Boosting decision trees.* In "Advances in Neural Information Processing Systems 8," 1996.

[7] H. Drucker, C. Cortes, L. Jackel, Y. LeCun, V. Vapnik. *Boosting and other ensemble methods.* Neural Computation 6(6) (1994), pp. 1289-1301.

[8] H. Drucker, R. Schapire, P. Simard. *Boosting performance in neural networks. Int. J. of Pattern Recog. and Machine Intelligence,* 7(4) (1993), pp. 705-719.

[9] H. Drucker, R. Schapire, P. Simard. *Improving performance in neural networks using a boosting algorithm.* In "Advances in Neural Information Processing Systems 5," 1993.

[10] Y. Freund. *Boosting a weak learning algorithm by majority.* In "Third Annual Workshop on Computational Learning Theory," (1990), pp. 202-216.

[11] Y. Freund. *An improved boosting algorithm and its implications on learning complexity.* In "Fifth Annual Workshop on Computational Learning Theory," (1992), pp.391-398.

[12] Y. Freund. *Boosting a weak learning algorithm by majority.* Information and Computation, 121(2) (1995), pp. 256-285.

[13] Y. Freund and R. Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting,* Journal of Comp. and System Sci. 55(1) (1997), pp. 119-139.

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

[15] R. Impagliazzo. *Hard-core distributions for somewhat hard problems.* In "36th Annual Symposium on Foundations of Computer Science," (1995), pp. 538-545.

[16] R. Impagliazzo and A. Widgerson. *P = BPP unless E has subexponential circuits: derandomizing the XOR lemma.* In "29th Annual Symposium on Theory of Computing," (1997), pp. 220-229.

[17] J. Jackson. *An efficient membership-query algorithm for learning DNF with respect to the uniform distribution.* In "35th Annual Symposium on Foundations of Computer Science," (1994), pp. 42-53.

[18] J. Jackson. "The Harmonic Sieve: A Novel Application of Fourier Analysis to Machine Learning Theory and Practice," Ph.D. thesis, Carnegie Mellon University, Aug. 1995; Available as technical report CMU-CS-95-183.

[19] J. Jackson. *An efficient membership-query algorithm for learning DNF with respect to the uniform distribution.* Journal of Computer and System Sciences 55 (1997), pp. 414-440.

[20] J. Jackson and M. Craven. *Learning sparse perceptrons.* In "Advances in Neural Information Processing Systems 8," 1996.

[21] M. Krause and P. Pudlak. *On the computational power of depth 2 circuits with threshold and modulo gates.* In "26th Annual Symposium on Theory of Computing," (1994), pp. 48-57.

[22] M. Kearns and L. Valiant. *Cryptographic limitations on learning boolean formulae and finite automata.* J. ACM 41(1) (1994), pp. 67-95.

[23] L. Levin. *Average case complete problems.* SIAM J. on Comput. 15(1) (1986), pp. 285-286.

[24] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

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

[26] R. Schapire and Y. Singer. *Improved boosting algorithms using confidence-rated predictions.* In "Tenth Annual Workshop on Computational Learning Theory," (1998), pp. 80-91.

[27] L. G. Valiant. *A theory of the learnable.* Comm. ACM, 27(11) (1984), pp. 1134-1142.

[28] A. C. Yao. *Theory and applications of trapdoor functions.* In "23rd Annual Symposium on Foundations of Computer Science," (1982), pp. 80-91.