

The Uniform Hardcore Lemma via Approximate Bregman Projections*

Boaz Barak[†]

Moritz Hardt[‡]

Satyen Kale[§]

Abstract

We give a simple, more efficient and uniform proof of the hard-core lemma, a fundamental result in complexity theory with applications in machine learning and cryptography. Our result follows from the connection between boosting algorithms and hard-core set constructions discovered by Klivans and Servedio [11]. Informally stated, our result is the following: suppose we fix a family of boolean functions. Assume there is an efficient algorithm which for every input length and every smooth distribution (i.e. one that doesn't assign too much weight to any single input) over the inputs produces a circuit such that the circuit computes the boolean function noticeably better than random. Then, there is an efficient algorithm which for every input length produces a circuit that computes the function correctly on almost all inputs.

Our algorithm significantly simplifies previous proofs of the uniform and the non-uniform hard-core lemma, while matching or improving the previously best known parameters. The algorithm uses a generalized multiplicative update rule combined with a natural notion of approximate Bregman projection. Bregman projections are widely used in convex optimization and machine learning. We present an algorithm which efficiently approximates the Bregman projection onto the set of high density measures when the Kullback-Leibler divergence is used as a distance function. Our algorithm has a logarithmic runtime over any domain from which we can efficiently sample. High density measures correspond to smooth distributions which arise naturally, for instance, in the context of online learning. Hence, our technique may be of independent interest.

*This paper includes and extends previous work by one of the authors [10].

[†]Department of Computer Science, Princeton University, boaz@cs.princeton.edu. Supported by NSF grants CNS-0627526 and CCF-0426582, US-Israel BSF grant 2004288 and Packard and Sloan fellowships.

[‡]Department of Computer Science, Princeton University, mhardt@cs.princeton.edu

[§]Microsoft Research, One Microsoft Way, Redmond, WA 98052, satyen.kale@microsoft.com

1 Introduction

Informally, a hard-core lemma states that if a boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$ is somewhat hard to compute for a class of boolean circuits, then there is a large subset of inputs on which the function is very hard to compute for a slightly smaller class of circuits. Impagliazzo [7] gave the first proof of such a theorem and used it to derive an alternative proof of Yao's XOR lemma [17]. In this context, the hard-core lemma is used to transform a somewhat hard problem into a very hard problem. This method called *hardness amplification* is routinely used in complexity theory and cryptography.

The hard-core lemma itself is actually proven via the contrapositive as follows. We assume that for every large enough set of inputs, there is a small circuit that can match the function with a noticeable advantage over random guessing. From large sets, we move to *high density measures*. These are mappings from $\{0,1\}^n$ to a weight in $[0,1]$ such that the total weight is large. Thus, high density measures are relaxations of indicator functions on large sets. Consequently, we work with the assumption that for every high density measure, there is a small circuit that matches the function with an advantage noticeably better than random guessing when inputs are sampled with probability proportional to their weight in the measure. Note that high density measures correspond to *smooth* probability distributions that do not place too much weight on any single point.

Starting from this assumption, we develop a certain kind of *boosting* algorithm. Boosting is an essential algorithmic technique due to Freund and Schapire [2, 12] widely used in machine learning in order to learn concept classes with high accuracy starting from inaccurate hypotheses. In our context, the boosting algorithm generates a sequence of carefully constructed measures and for each such measure it obtains a small circuit that computes the function noticeably better than random guessing. Finally, it combines these circuits in some manner (typically, by taking a majority vote) to produce a larger circuit that correctly computes the function on almost all inputs.

The connection between boosting algorithms in machine learning and hard-core set constructions was

first observed by Klivans and Servedio [11]. For a boosting algorithm to imply the existence of large hard-core sets, it must satisfy two properties: first, the measures it generates must have high density (we refer to this as the smoothness property). Second, the number of rounds in the boosting algorithm should be small, so that the final circuit size is not too much larger than the circuits obtained in the process.

The hard-core lemmas obtained from such boosting algorithms are typically *non-uniform*: the output circuit is not created by a single efficient algorithm for varying input sizes. It may, in fact, depend on information that is hard to compute. Establishing a uniform hard-core lemma turned out to be more difficult with a first proof due to Holenstein [5] and an earlier weaker variant by Trevisan [14]. There are at least two main reasons for our interest in uniform techniques:

1. Strong methods of hardness amplification are known for non-uniform models of computation, e.g., [17]. Many of these fail in the uniform setting. Developing techniques for uniform hardness amplification has been an active research topic in recent years [14, 15, 8, 9].
2. Uniform techniques are more natural from an algorithmic point of view in areas such as online decision making and expert learning and hence could have potential applications in these areas.

In this paper, we give a new (uniform) algorithm for boosting that enjoys the additional smoothness property necessary to prove the hard-core lemma. This algorithm has the additional desirable property that it has the same number of iterations as the AdaBoost algorithm of Freund and Schapire and hence it is more efficient than previous methods. Our algorithm is also inspired by Warmuth and Kuzmin's [16] technique (which, in turn, uses ideas that originated in the work Herbster and Warmuth [4]) of obtaining smooth distributions from any other distribution by projecting it into the set of smooth distributions using the Kullback-Leibler divergence as a distance function. We transfer this technique to the context of high density measures over the boolean hypercube. The key difference is that the hypercube is a domain of exponential dimension and suitable projections might be difficult to compute efficiently. This problem can be overcome by allowing the hard-core lemma to be non-uniform. In the uniform case, however, we need to develop an efficient approximation algorithm instead.

1.1 Our Result. As mentioned earlier, a measure is a function $M : \{0, 1\}^n \rightarrow [0, 1]$. Define the *density* of the measure to be $2^{-n} \sum_{x \in \{0, 1\}^n} M(x)$.

We will prove the following uniform hardcore lemma:

THEOREM 1.1. *Let $\{f_n\}_{n \in \mathbb{N}}$ denote a family of boolean function and let $\delta, \gamma : \mathbb{N} \rightarrow (0, 1)$. Suppose, there exists an algorithm A which given oracle access to any measure M over $\{0, 1\}^n$ of density $\delta(n)$, returns a circuit C of size at most $s(n)$ such that $\Pr_{x \sim M}[C(x) = f_n(x)] \geq \frac{1}{2} + \gamma(n)$.*

Then there is an algorithm B which for every n and oracle access to f_n with probability $1 - \eta$ (over the internal randomness of B) returns a circuit C' such that C' computes f_n correctly on at least a $1 - \delta(n)$ fraction of all inputs. Furthermore,

1. *the algorithm B works in $O(\frac{1}{\gamma^2} \log \frac{1}{\delta})$ rounds with one call to A in each round and the runtime of B in every round is linear in n , $\log \frac{1}{\eta}$, $\frac{1}{\delta}$, $\frac{1}{\gamma^4}$ and the cost of simulating A ,*
2. *the circuit C' is the majority of $O(\frac{1}{\gamma^2} \log \frac{1}{\delta})$ circuits of size $s(n)$.*

Note that the algorithm assumes the existence of an algorithm which returns small circuits that perform noticeably better than random guessing when supplied with an oracle for any high density measures rather than an oracle for any large subset of inputs. However, an easy application of Chernoff bounds shows that if a set is chosen from a high density measure by choosing each input independently with probability equal to its weight in the measure, then such a set has large size and any small circuit has very similar performance when inputs are chosen uniformly from such a set and when they are chosen according to the measure (see [5] for details). Thus, Theorem 1.1 implies a similar theorem where the assumption on oracle access to a measure of density at least $\delta(n)$ is replaced by a similar assumption on oracle access to a subset of inputs of size at least $\delta(n)2^n$.

The main virtue of our result is efficiency and simplicity. We significantly simplify both the best known non-uniform construction due to Klivans and Servedio [11], as well as the uniform lemma due to Holenstein [5]. At the same time we match the best known parameters in terms of circuit size and obtain additional improvements in runtime.

Compared to the work of Klivans and Servedio [11], our algorithm is simpler for three reasons: (a) It obtains the best known parameters for hard-core set constructions directly by applying the boosting algorithm, rather than building it up incrementally by accumulating small hard-core sets, and (b) it obviates the necessity of composing two different boosting algorithms, with separate analyses of the distributions they produce,

and, consequently, (c) the final circuit found in our construction is a simple majority over circuits found by the boosting algorithm, rather than a majority of majorities over circuits.

Finally, the improved efficiency of our boosting algorithm has ramifications for an application in Klivans and Servedio's paper: it enables us to shave off a factor of $O(\log^6 \frac{1}{\delta})$ from the running time of the fastest known algorithm to learn DNF formulas with membership queries under the uniform distribution, where δ is the error parameter (i.e. to output a hypothesis that matches the unknown DNF formula on a $1-\delta$ fraction of inputs). Also, the final hypothesis is a simple majority-of-parities circuit, instead of a majority-of-majorities-of-parities circuit. Our boosting algorithm should be of independent interest also for the applications in Servedio's paper [13] on learning in the presence of malicious noise.

Compared to Holenstein's construction we reduce the dependence in the circuit size from $1/\delta^{O(1)}$ to $O(\log \frac{1}{\delta})$. The runtime of the algorithm B is better by polynomial factors. In the context of Holenstein's applications in cryptography [5, 6], these quantitative improvements result in more efficient proofs of security. The main technical difficulty in Holenstein's construction is a procedure to ensure that any distribution produced at intermediate steps has high enough density. This step is necessary in order to meet the assumption of the lemma. The proof of termination for this procedure is rather complicated. In contrast, in our proof this step is replaced by a natural notion of *approximate Bregman projection*. Bregman projections are widely used in convex optimization and machine learning to make solutions more well-behaved. Hence, our technique of computing approximate Bregman projections may be of independent interest. For this reason, we will give an informal overview of our techniques in the next section.

2 Overview of the proof

The structure of our boosting algorithm is based on the same technology as Freund and Schapire's well known *AdaBoost* algorithm [3] and that is the Multiplicative Weights Update method. Specifically, our algorithm creates throughout several round a sequence of (implicitly represented) probability measures $M^{(1)}, \dots, M^{(t)}$ over the boolean hypercube. Every measure is obtained from the previous one by applying a multiplicative update rule based on a certain penalty function. In the t -th round of our algorithm, we pass the measure $M^{(t)}$ to the weak learning algorithm given in the assumption of our theorem. This algorithm produces a circuit $C^{(t)}$. The penalty function used at the end of the t -th round

is based on the performance of this circuit when inputs are drawn from the measure $M^{(t)}$. In this fashion, we create a family of circuits $C^{(1)}, \dots, C^{(T)}$. These circuits we finally combine into a single majority circuit.

The difficulty with this approach is to make sure *efficiently* that every measure $M^{(t)}$ is of high enough density. This requirement is a priori violated by the Multiplicative Weights Update. To achieve this property we need to augment our algorithm with the technique of *Bregman projections* (as used in [4, 16] for a similar purpose).

2.1 Efficient Bregman projections. The advantage and the size of the circuit we output all depend on the performance of the multiplicative weights algorithm which is analyzed using a potential argument. Abstractly, in this potential argument we fix a measure M and track the *distance* from M to each point $M^{(t)}$. Our notion of distance is the Kullback-Leibler divergence $D(M \parallel M^{(t)})$. The Kullback-Leibler divergence is an example of a *Bregman divergence*. Therefore, Bregman's Theorem implies that the distance $D(M \parallel M^{(t)})$ does *not* increase even when we project $M^{(t)}$ to some convex set Γ that contains M . Here, the projection of $M^{(t)}$ onto Γ is defined as the measure M' that minimizes the distance $D(M' \parallel M^{(t)})$. Naturally, in our case the convex set Γ is polytope of high density measures over the boolean hypercube. Notice, the Bregman projection is defined by a convex program and can thus be computed efficiently if the dimension of problem is polynomially bounded. This, unfortunately, is not the case when we work over the boolean hypercube $\{0, 1\}^n$ and aim for a runtime that is polynomial in n . To overcome this problem we introduce the natural notion of an approximate Bregman projection for which the increase in distance is bounded by some parameter. We then show that the guarantees of the multiplicative weights algorithm are roughly preserved when working with good enough approximate projections. More importantly, we give an efficient randomized algorithm for computing approximate Bregman projections onto the set of high density measures. This algorithm uses an efficient implicit characterization of the projection, as well as the fact that the density parameter can be approximated efficiently from Chernoff bounds.

3 Probability Measures and Bregman Projections

Let X denote a finite set. A discrete *measure* on the set X is a function $M: X \rightarrow [0, 1]$. We let $|M| = \sum_{x \in X} M(x)$ denote the *weight* of M and $\mu(M) = |M|/|X|$ denotes its *density*.

The *Kullback-Leibler divergence* between two mea-

asures M, N is defined as

$$(3.1) \quad D(M \parallel N) = \sum_x M(x) \log \frac{M(x)}{N(x)} + N(x) - M(x).$$

Further, let $\Gamma \subseteq \mathbb{R}^{|X|}$ be a non-empty closed convex set of measures. Then, the *Bregman projection* of N onto the set Γ is defined as

$$(3.2) \quad P_\Gamma N = \arg \min_{M \in \Gamma} D(M \parallel N).$$

One can show that for every measure N , the minimum in (3.2) exists and is unique. Furthermore, we have the following theorem.

THEOREM 3.1. (BREGMAN, 1967) *Let N, M be measures such that $M \in \Gamma$. Then,*

$$(3.3) \quad D(M \parallel P_\Gamma N) + D(P_\Gamma N \parallel N) \leq D(M \parallel N).$$

In particular,

$$(3.4) \quad D(M \parallel P_\Gamma N) \leq D(M \parallel N).$$

We refer the reader to the text book by Censor and Zenios [1] for a proof of Bregman's theorem and further information on the subject.

3.1 High Density Measures.

$$\Gamma_\delta = \{M \mid \mu(M) \geq \delta\}$$

as the set of measures having density at least δ . When δ is clear from the context, we may omit it. For every $\delta \in [0, 1]$, the set Γ_δ is closed and convex.

REMARK 3.1. *We will not need this fact, but we point out that high density measures correspond to smooth probability distributions. Indeed, the measure M has density at least δ if and only if the probability distribution $\frac{1}{|M|}M$ satisfies $\frac{1}{|M|}M(x) \leq \frac{1}{\delta|X|}$ for all $x \in X$.*

We will denote the Bregman projection operator onto the set Γ_δ by P_δ . The projection operator P_δ has the following very useful characterization.

LEMMA 3.1. *Let N be a measure with support at least $\delta 2^n$ and let $c \geq 1$ be the smallest constant such that the measure $M^* = \min(1, c \cdot N)$ has density δ . Then, $P_\delta N = M^*$.*

Proof. Consider the function $f(M) = D(M \parallel N)$ defined over the polytope Γ_δ . Since f is convex and differentiable in every variable we have

$$f(M) \geq f(M^*) + \nabla f(M^*)^T (M - M^*)$$

for every $M \in \Gamma_\delta$. Hence, in order to show that M^* minimizes the function f , it suffices to verify the optimality condition

$$\nabla f(M^*)^T (M - M^*) \geq 0$$

for all $M \in \Gamma_\delta$. It is easy to check that the gradient $\nabla f(M^*)$ is given by

$$\nabla f(M^*) = \left(\log \frac{M^*(x)}{N(x)} : x \in X \right).$$

Whenever x is such that $M^*(x) = 1$, we must have $N(x) \geq \frac{1}{c}$ and hence $\log(M^*(x)/N(x)) = \log \frac{1}{N(x)} \leq \log c$. But, $M(x) - M^*(x) = M(x) - 1 \leq 0$. Thus,

$$\log \frac{M^*(x)}{N(x)} \cdot (M(x) - M^*(x)) \geq \log c \cdot (M(x) - M^*(x)).$$

On the other hand, if $M^*(x) < 1$, then $\log(M^*(x)/N(x)) = \log(cN(x)/N(x)) = \log c$. Hence,

$$\begin{aligned} \sum_x \log \frac{M^*(x)}{N(x)} \cdot (M(x) - M^*(x)) \\ \geq \log c \cdot \sum_x M(x) - M^*(x). \end{aligned}$$

Finally, $\log c \geq \log 1 = 0$ and

$$\sum_{x \in X} M(x) \geq \sum_{x \in X} M^*(x),$$

since otherwise M would have density less than δ .

3.2 Approximate Bregman Projections. Computing the Bregman projection exactly requires time $|X|$ in general, since we may need to examine the weight of every point in X . We will be interested in computing approximations of the following kind more efficiently.

DEFINITION 3.1. *We call a measure \tilde{N} an α -approximation of $P_\Gamma N$ if for all $M \in \Gamma$ we have*

1. $\tilde{N} \in \Gamma$, and,
2. $D(M \parallel \tilde{N}) \leq D(M \parallel P_\Gamma N) + \alpha$.

Whether or not we can compute such an approximation more efficiently also depends upon how efficiently we can represent the resulting measure. In the case of P_δ we have an efficient implicit representation by virtue of the characterization in Lemma 3.1. In fact, we will now show how to compute the projection P_δ approximately. The proof also uses the fact that we can estimate the density parameter of a measure efficiently using Chernoff bounds. Notice, however, if N is a measure

of density $\delta_0 < \delta$, then the right scaling factor c may not simply be $c_0 = \delta/\delta_0$. This is because the weight of some points might get clipped at the value 1. In fact, in general c can be arbitrarily large. To see this consider the measure supported on a δ fraction of the inputs which is 1 at every point on its support except a single point where it may be arbitrarily small but positive. On the other hand, when given a bound on c , we can iteratively search for the right scaling factor.

LEMMA 3.2. *Let N be a measure such that $P_\delta N = \min(1, c \cdot N)$ for $c \leq 1 + \gamma$ and suppose we have oracle access to the measure N . Further, assume we can sample uniform elements from the domain X in time t . Then, we can compute an implicitly represented $\epsilon\delta|X|$ -approximation of $P_\delta N$ in time*

$$O\left(\frac{t}{\delta\epsilon^2}\left(\log\log\frac{\gamma}{\epsilon} + \log\frac{1}{\eta}\right)\right)$$

with probability $1 - \eta$.

Proof. By Lemma 3.1, the projection of N onto the set of density- δ measures is computed by finding the smallest factor $c \geq 1$ such that the measure $P_\delta N = \min(1, c \cdot N)$ has density δ .

CLAIM 1. *Suppose \tilde{N} is a measure such that*

1. $\tilde{N}(x) \geq P_\delta N(x)$ for all $x \in X$,
2. $|\tilde{N}| - |P_\delta N| \leq \epsilon\delta|X|$.

Then, \tilde{N} is an $\epsilon\delta|X|$ -approximation of $P_\delta N$

Proof. Let $M \in \Gamma_\delta$. Then,

$$\begin{aligned} & D(M || \tilde{N}) - D(M || P_\delta N) \\ &= \sum_x M(x) \log \frac{P_\delta N(x)}{\tilde{N}(x)} + |\tilde{N}| - |P_\delta N| \end{aligned}$$

(by 1) $\leq |\tilde{N}| - |P_\delta N|$

(by 2) $\leq \epsilon\delta|X|$.

Now, suppose we can compute a factor $\tilde{c} \in [1, 1 + \gamma]$ such that the measure $\tilde{N} = \min(1, \tilde{c} \cdot N)$ satisfies

$$\delta \leq \mu(\tilde{N}) \leq (1 + \epsilon)\delta.$$

Then it is easy to check that \tilde{N} will satisfy the assumptions of Claim 1. Indeed, we have $\tilde{c} \geq c$ and hence the first requirement. The second one follows since $|P_\delta N| \geq \delta|X|$.

We will compute \tilde{c} using binary search over the interval $[1, 1 + \gamma]$. At each step, we estimate the weight of our candidate measure to within a multiplicative

error of $1 + \Omega(\epsilon)$ using random samples. We can relate the number of samples to the error probability using Hoeffding's bound. As soon as the weight of our candidate lies in an appropriately small interval, we terminate. At every step of binary search we proceed as follows:

1. Let c^* be the current candidate value for \tilde{c} and let $N^* = \min(1, c^* \cdot N)$.
2. Compute an estimate $\hat{\mu}$ of the density $\mu(N^*)$: Sample s points $x_1, \dots, x_s \in \{0, 1\}^n$ uniformly at random and compute the average weight

$$\hat{\mu} = \frac{1}{s} \sum_{i=1}^s N^*(x_i).$$

Notice, $N^*(x_i)$ are independent random variables in the interval $[0, 1]$ such that $E[N^*(x_i)] = \mu(N^*)$. Hence, by Chernoff bounds

$$\Pr(|\hat{\mu} - \mu(N^*)| > \frac{\epsilon\delta}{3}) \leq \exp(-\Omega(s\epsilon^2\delta)).$$

3. If $\hat{\mu} \in [\delta + \frac{\epsilon\delta}{3}, (1 + \epsilon)\delta - \frac{\epsilon\delta}{3}]$, terminate. Otherwise, we continue with binary search.

Binary search will perform at most $O(\log\frac{\gamma}{\epsilon})$ steps in the worst case. If in every step our density estimate is correct, then the algorithm terminates with an $\epsilon\delta|X|$ -approximation.

To apply a union bound over the number of steps and further achieve the desired over all error probability of η , we want that

$$O(\log(\gamma/\epsilon)e^{-\Omega(s\epsilon^2\delta)}) \leq \eta,$$

which is true for

$$s = O\left(\frac{1}{\delta\epsilon^2}\left(\log\log\frac{\gamma}{\epsilon} + \log\frac{1}{\eta}\right)\right).$$

REMARK 3.2. *Our algorithm to compute the approximate projection is essentially optimal up to the doubly-logarithmic factor $\log\log(\gamma/\epsilon)$, since at least $\Omega(\frac{1}{\delta\epsilon^2}\log\frac{1}{\eta})$ samples are required to estimate the density parameter to within the multiplicative error of $1 + \epsilon$ and error probability bounded by η .*

4 Online Learning with Approximate Bregman Projections

In this section, we consider the following standard model of online learning. In every round $t = 1 \dots T$, we commit to a measure $M^{(t)}$ over some set X . Next, every point $x \in X$ is associated adversarially with a penalty $m_x^{(t)} \in [0, 1]$, the t -th component of the

penalty vector $\mathbf{m}^{(t)}$. We incur the loss defined as $L(M^{(t)}, \mathbf{m}^{(t)}) = \sum_{x \in \{0,1\}^n} M^{(t)}(x) m_x^{(t)}$. We may think of the loss as proportional to the expected penalty when inputs are sampled according to the measure $M^{(t)}$. Based on all information from previous rounds, we compute an updated measure $M^{(t+1)}$. Ultimately, our goal is to compute a sequence of measures such that the total loss $\sum_{t=1}^T L(M^{(t)}, \mathbf{m}^{(t)})$ is not much larger than the loss of every fixed measure M in hindsight, i.e., $\sum_{t=1}^T L(M, \mathbf{m}^{(t)})$. An additional constraint in our setup is that we are given a closed convex set Γ and demand that every measure $M^{(t)} \in \Gamma$. We will enforce this constraint by replacing every measure $M^{(t)}$ by its approximate projection onto the set Γ .

We have the following lemma.

Input: Closed convex set of measures Γ
Initial measure $M^{(1)} \in \Gamma$
Parameters $\gamma \in (0, \frac{1}{2})$, $\alpha \geq 0$ and $T \in \mathbb{N}$

For $t = 1, 2, \dots, T$ rounds:

1. Let $\mathbf{m}^{(t)}$ be an arbitrary penalty.
2. Define $N^{(t+1)}$ coordinate-wise using the following update rule

$$N^{(t+1)}(x) = (1 - \gamma)^{m_x^{(t)}} M^{(t)}(x).$$

3. Let $M^{(t+1)}$ be an α -approximation of the measure $P_\Gamma N^{(t+1)}$.

Output: The sequence of measures $M^{(1)}, M^{(2)}, \dots, M^{(T)}$.

Figure 1: Multiplicative weights update using approximate Bregman projections

LEMMA 4.1. (TOTAL LOSS) *For every measure $M \in \Gamma$, the algorithm depicted in Figure 4 achieves the following bound in the above game,*

$$\sum_{t=1}^T L(M^{(t)}, \mathbf{m}^{(t)}) - \frac{\alpha}{\gamma} T \leq (1 + \gamma) \sum_{t=1}^T L(M, \mathbf{m}^{(t)}) + D(M \parallel M^{(1)}).$$

Proof. By definition,

$$\begin{aligned} & D(M \parallel N^{(t+1)}) - D(M \parallel M^{(t)}) \\ &= \sum_x M(x) \log \frac{M^t(x)}{N^{(t+1)}(x)} + |N^{(t+1)}| - |M^{(t)}|. \end{aligned}$$

Also,

$$\begin{aligned} \sum_x M(x) \log \frac{M^t(x)}{N^{(t+1)}} &= \sum_x M(x) \log((1 - \gamma)^{-m_x^{(t)}}) \\ &\leq \gamma(1 + \gamma) \sum_x M(x) m_x^{(t)} \\ &= \gamma(1 + \gamma) L(M, \mathbf{m}^{(t)}), \end{aligned}$$

where we used that

$$-\ln(1 - \gamma) \leq \gamma(1 + \gamma)$$

for $\gamma \leq \frac{1}{2}$. On the other hand,

$$\begin{aligned} |N^{(t+1)}| &= \sum_x (1 - \gamma)^{m_x^{(t)}} M^{(t)}(x) \\ &\leq \sum_x (1 - \gamma m_x^{(t)}) M^{(t)}(x) \\ &= |M^{(t)}| - \gamma \sum_x M^{(t)}(x) m_x^{(t)} \\ &= |M^{(t)}| - \gamma L(M^{(t)}, \mathbf{m}^{(t)}). \end{aligned}$$

Hence,

$$\begin{aligned} & D(M \parallel N^{(t+1)}) - D(M \parallel M^{(t)}) \\ &= \gamma(1 + \gamma) L(M, \mathbf{m}^{(t)}) - \gamma L(M^{(t)}, \mathbf{m}^{(t)}). \end{aligned}$$

Furthermore, by Bregman's Theorem,

$$\begin{aligned} D(M \parallel N^{(t+1)}) &\geq D(M \parallel P_\Gamma N^{(t+1)}) \\ &\geq D(M \parallel M^{(t+1)}) - \alpha, \end{aligned}$$

since $M^{(t+1)}$ is an α -approximation of $P_\Gamma N^{(t+1)}$. Hence, we have established the bound

$$\begin{aligned} & D(M \parallel M^{(t+1)}) - D(M \parallel M^{(t)}) \\ &= \gamma(1 + \gamma) \sum_x M(x) m_x^{(t)} \\ &\quad - \gamma \sum_x M^{(t)}(x) m_x^{(t)} + \alpha. \end{aligned} \tag{4.5}$$

To conclude the proof, we sum up (4.5) from $t = 1$ to T and simplify the telescoping sum.

5 A Uniform Hard-Core Lemma

In this section, we prove our main theorem which we restate below.

THEOREM 5.1. (THEOREM 1.1 RESTATED) *Let $\{f_n\}_{n \in \mathbb{N}}$ denote a family of boolean function and let $\delta, \gamma: \mathbb{N} \rightarrow (0, 1)$. Suppose, there exists an algorithm A which given oracle access to any measure M over $\{0, 1\}^n$ of density $\delta(n)$, returns a circuit C of size at most $s(n)$ such that $\Pr_{x \sim M}[C(x) = f_n(x)] \geq \frac{1}{2} + \gamma(n)$.*

Then there is an algorithm B which for every n and oracle access to f_n with probability $1 - \eta$ (over the internal randomness of B) returns a circuit C' such that C' computes f_n correctly on at least a $1 - \delta(n)$ fraction of all inputs. Furthermore,

1. the algorithm B works in $O(\frac{1}{\gamma^2} \log \frac{1}{\delta})$ rounds with one call to A in each round and the runtime of B in every round is linear in n , $\log \frac{1}{\eta}$, $\frac{1}{\delta}$, $\frac{1}{\gamma^4}$ and the cost of simulating A ,
2. the circuit C' is the majority of $O(\frac{1}{\gamma^2} \log \frac{1}{\delta})$ circuits of size $s(n)$.

Input: Oracle access to a boolean function f_n
Parameters $\delta > 0$, $\gamma \in (0, \frac{1}{2})$ and $T \in \mathbb{N}$
Algorithm A satisfying the assumption of Theorem 1.1

For $t = 1, 2, \dots, T = \frac{4}{\gamma^2} \log \frac{1}{\delta} + 1$ rounds:

1. Run algorithm A with oracle access to $M^{(t)}$ so as to obtain a circuit $C^{(t)}$.

Here, $M^{(1)}$ denotes the measure that is δ at every point.

2. Define $\mathbf{m}^{(t)}$ by putting $m_x^{(t)} = 1$ if $C^{(t)}(x) = f_n(x)$ and 0 otherwise.
3. Define $N^{(t+1)}$ coordinate-wise using the following update rule

$$N^{(t+1)}(x) = (1 - \gamma)^{m_x^{(t)}} M^{(t)}(x).$$

4. Let $M^{(t+1)}$ be an $(\gamma^2 \delta 2^n / 4)$ -approximation of $P_\delta N^{(t+1)}$.

Output: The circuit
 $C' = \text{MAJORITY}(C^{(1)}, C^{(2)}, \dots, C^{(T)})$

Figure 2: Outline of the smooth Boosting algorithm used in the proof of the hard-core lemma.

5.1 Proof of Theorem 1.1. In Figure 2, we outline the algorithm stated in the conclusion of our theorem. We will first analyze the error of the circuit that the algorithm outputs assuming all steps in our algorithm can be computed efficiently. Second, we will analyze the runtime of our algorithm (specifically Step 4) using Lemma 3.2.

5.1.1 Error Bound. We claim that the circuit C' computes f_n correctly on a $1 - \delta$ fraction of the inputs.

To argue this point we will appeal to Lemma 4.1. Let $E = \{x \in \{0, 1\}^n \mid C'(x) \neq f(x)\}$, i.e., those points on which the majority circuit C' errs. Further, suppose $|E| = \delta 2^n$. Let $W = \delta 2^n$.

First notice, it follows from our assumption that

$$\begin{aligned} & \sum_{t=1}^T L(M^{(t)}, \mathbf{m}^{(t)}) \\ &= \sum_{t=1}^T |M^{(t)}| \Pr_{x \sim M^{(t)}} [C^{(t)}(x) = f(x)] \\ (5.6) \quad & \geq \left(\frac{1}{2} + \gamma\right) TW, \end{aligned}$$

where we used that $|M^{(t)}| \geq W$.

Further, notice that $\sum_{t=1}^T m_x^{(t)}$ is equal to the number of circuits $C^{(t)}$ which correctly compute f_n on input x . Since C' is a majority circuit, we have $\sum_{t=1}^T m_x^{(t)} \leq \frac{1}{2} T$ whenever $x \in E$. Let U_E denote the uniform measure over E , i.e., U_E is equal to 1 on every point in E and 0 otherwise. We have $|U_E| = \delta 2^n$. Then,

$$(5.7) \quad \sum_{t=1}^T L(U_E, \mathbf{m}^{(t)}) = \sum_{x \in E} \sum_{t=1}^T m_x^{(t)} \leq \frac{1}{2} TW.$$

Moreover,

$$\begin{aligned} D(U_E \parallel M^{(1)}) &= \sum_{x \in E} \log \frac{1}{M^{(1)}(x)} + |M^{(1)}| - |U_E| \\ &= W \log \frac{1}{\delta}, \end{aligned}$$

since $M^{(1)}(x) = \delta$ and $|M^{(1)}| = |U_E|$. Thus, we can apply Lemma 4.1 to (5.6) and (5.7) with $M = U_E$ to conclude

$$(5.8) \quad \left(\frac{1}{2} + \gamma\right) TW - \frac{\alpha}{\gamma} T \leq \left(\frac{1}{2} + \frac{\gamma}{2}\right) TW + \frac{W}{\gamma} \log \frac{1}{\delta}.$$

That is,

$$\frac{\gamma}{2} T - \frac{\alpha}{\gamma W} T \leq \frac{1}{\gamma} \log \frac{1}{\delta}.$$

For $\alpha \leq \frac{\gamma^2 W}{4}$, the LHS is at least $\frac{\gamma}{4} T$ and hence,

$$T \leq \frac{4}{\gamma^2} \log \frac{1}{\delta}.$$

So, if we run our algorithm for $T \geq \frac{4}{\gamma^2} \log \frac{1}{\delta} + 1$ rounds, then we can be sure that $|E|/2^n < \delta$.

5.1.2 Computing the approximation. It remains to show how to compute the projection in Step 4. Let $M = M^{(t)}$ for some $t \in \{1, \dots, T\}$ be a measure of density δ and suppose we apply the update rule in Step 3

so as to obtain a measure $N = N^{(t+1)}$. At this point we know that

$$(5.9) \quad \mu(N) \geq (1 - \gamma)\delta.$$

Recall, by Lemma 3.1, the projection is given by $P_\delta N = \min(1, c \cdot N)$ for some particular c . It follows from (5.9) that

$$(5.10) \quad 1 \leq c \leq \frac{1}{1 - \gamma} \leq 1 + 2\gamma.$$

Hence, we can apply Lemma 3.2 where we set $\eta = \frac{\eta'}{T}$ (to achieve the over all error probability of η') and $\epsilon = 4/\gamma^2$. As a result, the projection is computed in time

$$O\left(\frac{1}{\delta\epsilon^2} \left(\log \frac{\eta'}{T} + \log \log \frac{\gamma}{\epsilon}\right)\right) = O\left(\frac{1}{\delta\epsilon^2} \log \frac{\eta'}{T}\right).$$

This concludes the proof of Theorem 1.1. \square

5.2 Improving the success probability. The conclusion of Theorem 1.1 gives us an algorithm B which computes f_n correctly on a $1 - \delta(n)$ fraction of the inputs. Holenstein [5] already showed how this can be improved to $1 - \delta(n)/2$ (which is in fact optimal).

For simplicity we only briefly outline how this is done. Let \mathcal{C} be the collection of T circuits obtained at the end of the algorithm. Let S be any set of size $W = \delta 2^n$, and let U_S be the uniform measure on S . We claim that a randomly chosen circuit computes the value of f on a randomly chosen input from S with probability more than $1/2$. Otherwise, we have

$$\sum_{t=1}^T L(U_S, \mathbf{m}^{(t)}) = \sum_{x \in S} \sum_{t=1}^T m_x^{(t)} \leq \frac{1}{2}TW.$$

Following the argument of Section 5.1.1 with U_S in place of U_E , we get a contradiction for $T > \frac{4}{\gamma^2} \log \frac{1}{\delta}$. Now, the construction of Claim 2.15 of [5] is directly applicable to the circuits in \mathcal{C} , and it yields a circuit which computes f_n correctly on $1 - \delta(n)/2$ fraction of inputs.

Acknowledgments

We would like to thank Thomas Holenstein and David Steurer for helpful discussions on this work. The third author would like to thank Manfred Warmuth for showing him the Bregman projection idea.

References

[1] Yair Censor and Stavros A. Zenios. *Parallel Optimization — Theory, Algorithms, and Applications*. Oxford University Press, 1997.

[2] Yoav Freund. Boosting a weak learning algorithm by majority. In *Proc. 3rd COLT*, 1990.

[3] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[4] Mark Herbster and Manfred K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[5] Thomas Holenstein. Key agreement from weak bit agreement. In *Proc. 37th ACM STOC*, 2005.

[6] Thomas Holenstein. Pseudorandom generators from one-way functions: A simple construction for any hardness. In *Proc. of 3rd TCC*. Springer, 2006.

[7] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proc. 36th IEEE FOCS*, 1995.

[8] Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Approximately list-decoding direct product codes and uniform hardness amplification. In *Proc. 47th IEEE FOCS*, 2006.

[9] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized and derandomized. In *Proc. 40th ACM STOC*, 2008.

[10] Satyen Kale. Boosting and hard-core set constructions: a simplified approach. *Electronic Colloquium on Computational Complexity (ECCC)*, (131), 2007.

[11] Adam R. Klivans and Rocco A. Servedio. Boosting and hard-core set construction. *Machine Learning*, 51(3):217–238, 2003.

[12] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[13] Rocco A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

[14] Luca Trevisan. List-decoding using the XOR lemma. In *Proc. of 44th IEEE FOCS*, 2003.

[15] Luca Trevisan. On uniform amplification of hardness in NP. In *Proc. 37th ACM STOC*, 2005.

[16] Manfred K. Warmuth and Dima Kuzmin. Randomized pca algorithms with regret bounds that are logarithmic in the dimension. In *In Proc. of NIPS*, 2006.

[17] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91. IEEE, 1982.