Electronic Colloquium on Computational Complexity, Report No. 127 (2012)

An Explicit VC-Theorem for Low-Degree Polynomials

Eshan Chattopadhyay
eshanc@cs.utexas.edu

Pravesh Kothari
kothari@cs.utexas.edu

Adam Klivans
klivans@cs.utexas.edu

October 2, 2012

Abstract

Let $X \subseteq \mathbb{R}^n$ and let $\mathcal{C}$ be a class of functions mapping $\mathbb{R}^n \to \{-1, 1\}$. The famous VC-Theorem states that a random subset S of X of size $O(\frac{d}{\epsilon^2} \log \frac{d}{\epsilon})$, where d is the VC-Dimension of $\mathcal{C}$, is (with constant probability) an ϵ-approximation for $\mathcal{C}$ with respect to the uniform distribution on X. In this work, we revisit the problem of constructing S explicitly. We show that for any $X \subseteq \mathbb{R}^n$ and any Boolean function class $\mathcal{C}$ that is uniformly approximated by degree k, low-weight polynomials, an ϵ-approximation S can be be constructed deterministically in time $\mathsf{poly}(n^k, 1/\epsilon, |X|)$. Previous work due to Chazelle and Matoušek suffers an $d^{O(d)}$ factor in the running time and results in superpolynomial-time algorithms, even in the case where $k = O(1)$.

We also give the first hardness result for this problem and show that the existence of a $\mathsf{poly}(n^k, |X|, 1/\epsilon)$-time algorithm for deterministically constructing ϵ-approximations for circuits of size n^k for every k would imply that $\mathsf{P} = \mathsf{BPP}$. This indicates that in order to construct explicit ϵ-approximations for a function class $\mathcal{C}$, we should not focus solely on $\mathcal{C}$'s VC-dimension.

Our techniques use deterministic algorithms for discrepancy minimization to construct hard functions for Boolean function classes over *arbitrary* domains (in contrast to the usual results in pseudorandomness where the target distribution is uniform over the hypercube).

ISSN 1433-8092

1 Introduction

The VC-Theorem is one of the most important results in Statistics and Machine Learning and gives a quantitative bound on the rate of convergence of an empirical estimate of the bias of every function in a Boolean concept class to its true bias. The rate depends on the well-known *VC-dimension* of a function class (we refer the reader to Vapnik [Vap] for background material), and for the purposes of this paper we state the theorem as follows:

Theorem 1.1 (VC Theorem [VC71], See also [Vap]). *Let $\mathcal{C}$ be a Boolean function class mapping $\mathbb{R}^n \to \{-1, 1\}$ and let d denote its VC-dimension. Let $X \subseteq \mathbb{R}^n$ be finite and let $\mathcal{U}_X$ be the uniform distribution on X. Let S be the set of points obtained by taking $O(\frac{d}{\epsilon^2} \log \frac{d}{\epsilon})$ random draws from $\mathcal{U}_X$ and let $\mathcal{U}_S$ be the uniform distribution over points in S. Then with probability at least $1/2$ over the choice of S, for every $c \in \mathcal{C}$*

$$|\Pr_{\mathcal{U}_S}[c(x) = 1] - \Pr_{\mathcal{U}_X}[c(x) = 1]| \leq \epsilon.$$

This bound on the sample size was improved to $O(\frac{d}{\epsilon^2})$ in [LLS01].

In this paper we are concerned with the following question: given as input the set X and a bound on the VC-dimension d of the class $\mathcal{C}$, can we construct S in deterministic polynomial time in $n, 1/\epsilon, |X|$, and d? The set S is commonly referred to as an ϵ-approximation.

From the perspective of computational complexity (and in particular pseudorandomness), it is natural to try to understand the complexity of explicitly constructing ϵ-approximations. In computational geometry, this problem has been studied before, most notably in work due to Chazelle and Matoušek [CM96]. The parameters d and n, however, were considered constants, and the main goal was to obtain algorithms with run-time linear in $|X|$. Their work has had applications for finding deterministic algorithms for solving low-dimensional linear programs.

In an impressive new paper by Feldman and Langberg [FL11], the authors prove that ϵ-approximations for the function class of *halfspaces* in n dimensions give new algorithms for constructing core-sets and subsequently yield new applications for a host of clustering problems. Their paper inspires the following challenging open problem:

Open Problem 1. *Given a finite set of points X in $\mathbb{R}^n$, construct an ϵ-approximation for the class of halfspaces in deterministic time* $\mathsf{poly}(n, |X|, 1/\epsilon)$.

Previous work in computational geometry on explicit constructions of ϵ-approximations requires an enumeration of all possible labelings induced by $\mathcal{C}$ on the set X and suffers a $d^{\Omega(d)}$ factor in the running time where d is the VC-dimension of the function class. Since halfspaces in n dimensions have VC-dimension $n + 1$, these results run in time $n^{\Omega(n)}$.

1.1 Our Contributions

Unfortunately, we were unable to make progress on the above open problem. It turns out, however, that it is challenging to explicitly construct ϵ-approximations even for very simple classes of Boolean functions such as conjunctions and constant-depth decision trees. Here we give the first explicit constructions of ϵ-approximations for these classes that run in time subexponential in the VC-dimension. For the case of constant-depth decision trees, we give a polynomial-time deterministic construction. Our most general positive result is the following:

Theorem 1.2 (ϵ-Approximation For Functions Approximated by Low Degree Polynomials)**.** *Let $\mathcal{C}$ be any class of Boolean functions on n inputs on any finite set $X \subseteq [-1,1]^n$ such that $\mathcal{C}$ is δ-uniformly approximated on X by polynomials of degree at most d and weight at most W. Then, there is an algorithm that constructs an $(\epsilon+\delta\log|X|)$-approximation for $\mathcal{C}$ of size $\mathsf{poly}(W,k,\log n,\frac{1}{\epsilon})$ and runs in time $\mathsf{poly}(W,n^k,\frac{1}{\epsilon})\cdot|X|$ whenever $\epsilon=\Omega\left(W\sqrt{\frac{k\log n}{|X|}}\right)$. (The exact dependence on the parameters can be found in Corollary 4.1.)*

Remark 1.1. *As in Chazelle and Matoušek [CM96], we require a lower bound on ϵ in order to achieve non-trivial bounds on the size of the ϵ-approximation. Roughly speaking, if ϵ is chosen to be too small, then we allow ourselves to output the entire set X as the approximation.*

Combining Theorem 1.2 with known results on uniformly approximation Boolean functions by polynomials, we obtain the following corollary for Boolean conjunctions:

Corollary 1.3 (Informal Statement; see Section 4 for precise bounds)**.** *Let $X \subseteq \{-1,1\}^n$. Then there is a deterministic algorithm that constructs an ϵ-approximation for the class of Boolean conjunctions on n variables of size $n^{\tilde{O}(\sqrt{n})}/\epsilon^2$ with running time $n^{O(\sqrt{n})}/\epsilon^2\cdot|X|$.*

Recall that the class of conjunctions on n literals has VC-dimension n, so previous work due to Chazelle and Matoušek [CM96] would require time $\Omega(2^n)$ to produce such an approximation. We point out here, however, that the *size* of the ϵ-approximations output by Chazelle and Matoušek will be much better than ours (in fact, they achieve the optimal $O(\frac{d}{\epsilon^2}\log\frac{d}{\epsilon})$ bound on the size of S).

For the class of constant-depth decision trees, we can obtain a polynomial-time, deterministic algorithm:

Corollary 1.4 (Informal Statement; see Section 4 for precise bounds)**.** *Let $X \subseteq \{-1,1\}^n$ and let $\mathcal{C}$ be the class of depth k decision trees. Then there is a deterministic algorithm that constructs an ϵ-approximation for $\mathcal{C}$ of size $n^{O(k)}/\epsilon^2$ with running time $\frac{n^{O(k)}}{\epsilon^2}\cdot|X|$.*

Constant-depth decision trees have VC-dimension $\Omega(\log n)$ (for completeness we include a proof in Section C), so previous work will require time $\Omega(n^{\log\log n})$ to produce ϵ-approximations.

1.2 Our Contributions: Hardness Result

Since we are allowed to run in time polynomial in $|X|$, it may seem that the problem of constructing ϵ-approximations is easier or at least incomparable to the problem of building pseudorandom generators for particular target distributions (such as $\{-1,1\}^n$). We prove, however, that giving explicit constructions of ϵ-approximations in time polynomial in n and the VC-dimension d is at least as hard as proving $\mathsf{P}=\mathsf{BPP}$ (recall that polynomial-size circuits have polynomial VC-dimension):

Theorem 1.5. *Let $\mathcal{C}_k^m$ be the class of all Boolean circuits of size at most m^k on m inputs (each such circuit computes a function from $\{-1,1\}^m\to\{-1,1\}$). Suppose, there exists an algorithm $\mathcal{B}^k$ for some $k>1$, which takes as input $X\subseteq\{-1,1\}^m$ and computes an $\frac{1}{3}$-approximation for $\mathcal{C}_m^k$ on X of size at most $\frac{|X|}{2}$ in time $\mathsf{poly}(|X|,m^k)$. Then $\mathsf{P}=\mathsf{BPP}$.*

In fact, we can show that constructing ϵ-approximations even for *linear*-size circuits deterministically in polynomial-time would imply $\mathsf{P}=\mathsf{BPP}$. As far as we are aware, this is the first hardness

result for the general problem of deterministically constructing ϵ-approximations and explains why Chazelle and Matoušek's work suffers an $d^{O(d)}$ factor in the running time. We conclude that we must consider additional properties of a concept class for which we wish to build ϵ-approximations other than its VC-dimension.

Our hardness result does *not*, however, rule out polynomial-time, deterministic constructions of ϵ-approximations for restricted function classes such as halfspaces (in fact, the relationship between Open Problem 1 and the problem of constructing pseudorandom generators for halfspaces with respect to $\{-1,1\}^n$ remains unclear).

1.3 Our Approach

Previous approaches for constructing ϵ-approximations over general domains $X \subseteq \mathbb{R}^n$ due to Chazelle and Matoušek [CM96] involve an enumeration of all possible labelings induced on X with respect to a concept class $\mathcal{C}$. Naively this results in an algorithm with time complexity $|X|^{O(d)}$ where d is the VC-dimension of $\mathcal{C}$. Chazelle and Matoušek instead only enumerate labelings on small subsets of X and use an elaborate method of partitioning and merging these subsets to reduce the time complexity to $|X| \cdot d^{O(d)}$ while maintaining an optimal size ϵ-approximation.[1] We wish to avoid this sort of enumeration altogether and run in time subexponential or even polynomial in the VC-dimension (although one drawback of our results is that we do not achieve optimal size ϵ-approximations).

Inspired by results on pseudorandom generators, our approach generates *average-case hard functions* with respect to arbitrary domains for a fixed concept class $\mathcal{C}$. In many cases in the pseudorandomness literature, an average-case hard function with respect to the uniform distribution on $\{-1,1\}^n$ can be used to build a pseudorandom generator. Here we show how to directly translate average-case hard functions with respect $\mathcal{U}_{\mathcal{X}}$ for an arbitrary X in order to generate an ϵ-approximation for X.

One advantage of this approach is that once we have generated an average-case hard function for a concept class $\mathcal{C}$, we also obtain hard functions for any class $\mathcal{C}'$ approximated in ℓ_1 by $\mathcal{C}$. As such, we focus on generating hard functions for low-degree polynomials, as they can approximate several interesting Boolean function classes.

The question remains– how do we deterministically obtain average-case hard functions with respect to arbitrary domains? Although this problem is well-understood in the complexity literature for the case of the uniform distribution over $\{-1,1\}^n$, very little is known for other domains (e.g., arbitrary subsets of the hypercube). In retrospect, it is not difficult to see that a suitably constructive proof of an upper-bound on the VC dimension of a concept class $\mathcal{C}$ will yield a worst-case hard function with respect to any domain.

Still, constructive proofs of upper-bounds on the VC-dimension of interesting function classes are hard to come by, and we require average-case, not worst-case hardness. To this end, we turn to well-studied tools for generating *low-discrepancy* colorings. We use a derandomized version of a low-discrepancy coloring for the class of monomials, and this coloring will correspond to an average-case hard function for polynomials.

Finally, we can combine our techniques with the work of Chazelle and Matoušek to obtain deterministic algorithms with a run-time that is linear in $|X|$.

[1] To reduce the time complexity to $d^{O(d)}$ Chazelle and Matoušek require a *subsystem oracle* for the class $\mathcal{C}$. The existence of such an oracle is dependent on the concept class.

2 Preliminaries

We will deal with uniform distributions on arbitrary finite sets X and bounded functions from X to $[-1,1]$. This normalization of the range is without loss of generality and just fixes the scale for our parameters. We will refer to such functions as just bounded functions and the corresponding classes as bounded function classes. The uniform distribution on X, denoted by $\mathcal{U}_X$ fixes our notions of correlations (inner products) and norms. We will also encounter classes of Boolean valued functions (Boolean functions from now) defined on *arbitrary* finite domains X that range over $\{-1,1\}$.

Definition 2.1 (Inner Product and Norms)**.** *For the uniform distribution $\mathcal{U}_X$ on X, and any two functions $f,g : X \to [-1,1]$, the inner product of f and g is given by: $\langle f,g\rangle_{\mathcal{U}_X} = \mathbb{E}_{x\sim\mathcal{U}_X}[f(x)\cdot g(x)]$. The ℓ_1, ℓ_2 norms of a function f on X are defined in the standard way as $\mathbb{E}_{x\sim\mathcal{U}_X}[|f(x)|]$, $\mathbb{E}_{x\sim\mathcal{U}_X}[f^2(x)]$ respectively.*

For a function class with respect to the distribution $\mathcal{U}_X$ on its domain X, we define a hard function (which is always Boolean) for the class as a function which has low correlation with every member function of the class.

Definition 2.2 (Hard Function)**.** *Let $\mathcal{C}$ be bounded function class on X. A function $f : X \to \{-1,1\}$ is said to be ϵ-hard for the class $\mathcal{C}$ on X if for every $c \in \mathcal{C}$, $|\langle f,c\rangle_{\mathcal{U}_X}| \le \epsilon$.*

We will encounter two kinds of approximators.

Definition 2.3 (ℓ_1-Approximators)**.** *A class $\mathcal{C}$ of bounded functions on X is said to be δ-approximated in ℓ_1-norm by another class $\mathbb{F}$ on X if for every $c \in \mathcal{C}$, there exists an $f \in \mathbb{F}$ such that $\mathbb{E}_{x\sim\mathcal{U}_X}[|c(x)-f(x)|] \le \delta$.*

Definition 2.4 (Uniform Approximation)**.** *A class of bounded functions $\mathcal{C}$ is said to be δ-uniformly approximated on X by another class $\mathbb{F}$ if for every $c \in \mathcal{C}$, there exists a $f \in \mathbb{F}$ such that for every $x \in X$, $|c(x)-f(x)| \le \delta$.*

We will also need the following definition of the weight of a polynomial.

Definition 2.5 (Weight of a Polynomial)**.** *Let $p : \mathbb{R}^n \to \mathbb{R}$ with $p(x) = \sum_{i=1}^{t} a_i \cdot m_i$ be a polynomial of degree k with monomials m_i each of degree at most k. We define the weight of p as $W(p) = \sum_{i=0}^{t} |a_i|$.*

2.1 Discrepancy

We now provide the basic definitions from discrepancy theory. We will only concern ourselves with *combinatorial discrepancy* (discrepancy from now) here. For further details, the reader may consult [Cha01, Mat99, PA95].

Definition 2.6 (Discrepancy of a Set System)**.** *Let $(X,\mathbb{S})$ be a set system with $\mathbb{S} = \{S_1, S_2, ..., S_m\}$ and $|X| = n$. Let $\chi : X \to \{-1,1\}$ be a coloring of X. The discrepancy of χ with respect to any set S is defined as $\chi(S) = \sum_{x\in S}\chi(x)$. The discrepancy of the coloring χ with respect to the set system $(X,\mathbb{S})$ is defined as $disc[X,\mathbb{S}](\chi) = max_{S\in\mathbb{S}}|\chi(S)|$. The discrepancy of the set system $(X,\mathbb{S})$ is defined as $disc(X,\mathbb{S}) = \min_{\chi:X\to\{-1,1\}} disc[X,\mathbb{S}](\chi)$.*

Intuitively, a low discrepancy coloring two-colors the domain X such that all the subsets of X in the set system receive approximately equal numbers of the 2 colors. A set system can be viewed as a Boolean function class by associating to every set in the set system, its incidence function on X. On the other hand, if $\mathcal{C}$ is a class of Boolean functions on the domain X, then every function $c \in \mathcal{C}$ is associated with the set whose indicator function on X is c, i.e. $S_c = \{x \mid x \in X, c(x) = 1\}$. Thus the class $\mathcal{C}$ along with the domain X is equivalent to a set system $(X, \mathbb{S}_{\mathcal{C}})$ where $\mathbb{S}_{\mathcal{C}} = \{S_c \mid c \in \mathcal{C}\}$.

In our setting, we will deal with discrepancies of arbitrary bounded functions. This definition is just a simple generalization of the preceding definition of discrepancy of a set system. Computing a coloring required in this generalization is sometimes referred to as the *lattice approximation problem* (see for example [Siv02]).

Definition 2.7 (Discrepancy of a Function Class)**.** *Let $\mathcal{C}$ be a set of functions $c : X \to [-1, 1]$ and let $\chi : X \to \{-1, 1\}$ be a coloring of X. The discrepancy of χ with respect to the function c is defined as $\chi(c) = \sum_{x:c(x)\geq 0} \chi(x) \cdot c(x)$. The discrepancy of χ with respect to the class $\mathcal{C}$ on X is defined as $disc[X, \mathcal{C}](\chi) = \max_{c\in\mathcal{C}} |\chi(c)|$.*

Note that when the function class is the set of indicator functions of a set system $(X, \mathbb{S})$ we recover the definition of the discrepancy of a set system as in Definition 2.6.

Quite expectedly, a uniformly random coloring turns out to be a low discrepancy coloring. We include a short proof of this fact in Section A.1 for completeness.

Lemma 2.8 (Discrepancy of Random Coloring)**.** *Let $\mathcal{C}$ be a bounded function class of m functions on domain $X \subseteq \mathbb{R}^n$. Let $\chi : X \to \{-1, 1\}$ be chosen uniformly and independent at random for every $x \in X$, i.e. $\Pr[\chi(x) = 1] = \frac{1}{2}$ for every $x \in X$. Then with probability at least $\frac{1}{2}$, $disc[X, \mathcal{C}](\chi) \leq O(\sqrt{n \log m})$.*

For the case of set systems a simple derandomization by conditional expectations of the random coloring method described above yields a deterministic construction of a low discrepancy coloring. For bounded function classes, we can use Nisan's deterministic simulation [Nis92a, Nis92b] to compute such a coloring deterministically (see [Siv02]) in $\mathsf{poly}(m, n)$ time.

Lemma 2.9 (Deterministic Construction of Low Discrepancy Coloring [Cha01, Siv02])**.** *Let $\mathcal{C}$ be a bounded function class of m functions on X. There exists a deterministic algorithm running in time $\mathsf{poly}(m, |X|)$ that produces a coloring with discrepancy at most $O(\sqrt{|X| \log m})$.*

Spencer, in his much celebrated paper [Spe85], showed that there exists a coloring for every set system that actually *beats* the discrepancy bound of random coloring. In a breakthrough result, Bansal [Ban10] solved the long open problem of constructing such a coloring by giving a randomized algorithm and subsequently a deterministic one along with Spencer [SB11]. Very recently, an arguably simpler proof of this result was found [LM12]. Since the improvement for our purposes by using the recent deterministic constructions is minor and our techniques use these results as a black box, we skip the analysis of the difference in our constructions caused by choosing between these algorithms.

2.2 Definition of ϵ-Approximations for Boolean Function Classes

We now define the idea of an ϵ-approximation discussed in the introduction. Note that this definition is only for the case of *Boolean* function classes.

Definition 2.10 (ϵ-approximation)**.** *Let $\mathcal{C}$ be a* Boolean *function class on the domain X. An ϵ-approximation for $\mathcal{C}$ on X is a set $Y \subseteq X$ such that for every $c \in \mathcal{C}$,*

$$|\Pr_{x \sim \mathcal{U}_Y}[c(x) = 1] - \Pr_{x \sim \mathcal{U}_X}[c(x) = 1]| \leq \epsilon.$$

As we noted in Theorem 1.1, the famous VC Theorem shows that, for a class $\mathcal{C}$ on a finite domain X, a random subset of X of size $O(\frac{d}{\epsilon^2} \log \frac{d}{\epsilon})$ is an ϵ-approximation with constant probability.

2.3 Hard Functions and Discrepancy

Here we describe a simple connection between the idea of low discrepancy colorings and hard functions. This connection is almost an equivalence when the underlying class consists of Boolean functions. In the case of classes of arbitrary bounded functions, however, the connection is slightly more complicated. As we shall see, this translation both simplifies several proofs and facilitates our polynomial-approximation based approach.

We first show how an algorithm for constructing a low discrepancy coloring for a Boolean function class yields a hard function for the class. This translation is immediate and just results in loss of a constant factor in the parameters.

Proposition 2.1 (Low Discrepancy $\Rightarrow$ Hard Function)**.** *Let $\mathcal{C}$ be a class of bounded functions from X to $[-1, 1]$. Let $-C = \{-c : c \in C\}$ denote the class of all negated functions from $\mathcal{C}$. If $\chi : X \to \{-1, 1\}$ is a coloring of X with discrepancy at most $\epsilon|X|$ with respect to $\mathcal{C} \cup -\mathcal{C}$ then χ is 2ϵ-hard for $\mathcal{C}$ on X.*

Proof. Let $c \in \mathcal{C}$ be any member function of the class $\mathcal{C}$. Since χ has discrepancy at most $\epsilon|X|$ with respect to c and $-c$, we have: $|\sum_{\substack{x \in X \\ c(x) \geq 0}} \chi(x) \cdot c(x)| \leq \epsilon|X|$ and $|\sum_{\substack{x \in X \\ c(x) \leq 0}} \chi(x) \cdot c(x)| \leq \epsilon|X|$. Thus,

$$|\langle \chi, c \rangle_{\mathcal{U}_X}| = |\mathop{\mathbb{E}}_{x \sim \mathcal{U}_X}[\chi(x) \cdot c(x)]| \leq \frac{1}{|X|} \left(|\sum_{\substack{x \in X \\ c(x) \geq 0}} \chi(x) \cdot c(x)| + |\sum_{\substack{x \in X \\ c(x) \leq 0}} \chi(x) \cdot c(x)| \right) \leq \frac{1}{|X|} (\epsilon|X| + \epsilon|X|)$$
$$= 2\epsilon$$

□

Remark 2.1. *If χ is a low discrepancy coloring for $\mathcal{C} \cup \mathbb{1}$ where $\mathcal{C}$ is Boolean function class and $\mathbb{1}$ is the constant function taking the value 1 everywhere, then χ is a low discrepancy coloring for $\mathcal{C} \cup -\mathcal{C}$ (with a loss a factor of 2). In other words, for Boolean function classes we only need χ to be a low discrepancy coloring for $\mathcal{C} \cup \mathbb{1}$.*

It is now easy to see that for a class of m Boolean functions on X, the algorithm for Lemma 2.9 yields a $2\sqrt{\frac{\log m}{|X|}}$-hard function for the class on X and runs in time $\mathsf{poly}(m, |X|)$.

To complete the connection, we need to show how a function which is hard for a class $\mathcal{C}$ on X is a low discrepancy coloring for $\mathcal{C}$. When $\mathcal{C}$ is Boolean this is true whenever $\mathcal{C}$ contains the constant function $\mathbb{1}$. However, when $\mathcal{C}$ is an arbitrary class of functions bounded in $[-1, 1]$ we require χ to be hard for a related class too. We first define this class of absolute values of all functions in a given class.

Definition 2.11 (Absolute Value Class)**.** *For any function $c : X \to [-1, 1]$, we define $|c| : X \to [0, 1]$ as the absolute value of c on X. That is, $|c|(x) = |c(x)|$ for every $x \in X$. For a class $\mathcal{C}$ of bounded functions on a finite set X, we denote by $\mathcal{C}^{abs}$ the class of functions defined as $\mathcal{C}^{abs} = \{|c| \mid c \in \mathcal{C}\}$.*

Proposition 2.2 (Hard Function $\Rightarrow$ Low discrepancy)**.** *Let $\mathcal{C}$ be a class of bounded functions on X. Suppose $\chi : X \to \{-1, 1\}$ is ϵ-hard for the class $\mathcal{C} \cup \mathcal{C}^{abs}$ on X. Then $\chi : X \to \{-1, 1\}$ is a coloring of X for $\mathcal{C}$ with discrepancy at most $\epsilon|X|$.*

Proof. Let $c \in \mathcal{C}$. We know that χ is ϵ hard for c and $|c|$ on X. Thus, we have,
$|\langle \chi, c \rangle_{\mathcal{U}_X}| \leq \epsilon \Rightarrow |\sum_{\substack{x \in X \\ c(x) \geq 0}} \chi(x) \cdot c(x) + \sum_{\substack{x \in X \\ c(x) < 0}} \chi(x) \cdot c(x)| \leq \epsilon |X|$ and
$|\sum_{x \in X} \chi(x) \cdot |c(x)|| \leq \epsilon |X| \Rightarrow |\sum_{\substack{x \in X \\ c(x) \geq 0}} \chi(x) \cdot c(x) - \sum_{\substack{x \in X \\ c(x) < 0}} \chi(x)| \leq \epsilon |X|$

From the two equations above, we get:

$$|\sum_{\substack{x \in X \\ c(x) \geq 0}} \chi(x) \cdot c(x)| \leq \max \left\{ |\sum_{\substack{x \in X \\ c(x) \geq 0}} \chi(x) + \sum_{x \in X : c(x) < 0} \chi(x)|, |\sum_{x \in X : c(x) \geq 0} \chi(x) - \sum_{\substack{x \in X \\ c(x) < 0}} \chi(x)| \right\}$$
$$\leq \epsilon |X|$$

□

Note. *Note that the class $\mathcal{C}^{abs}$ is just the constant function $\mathbb{1}$ whenever $\mathcal{C}$ is a class of Boolean functions. Thus when we deal with a Boolean function class that contains the constant function $\mathbb{1}$ a hard function is a low discrepancy coloring.*

3 Constructing Hard Functions on Arbitrary Domains

In this section, we describe our constructions of hard functions for Boolean function classes on arbitrary domains that are approximated by low weight linear combinations of functions from another small class. We then apply these results to obtain hard functions for the well studied Boolean function classes of conjunctions (and more generally linear sized formulas) and decision trees. We first show that if a class $\mathcal{C}$ has good approximations as low weight linear combinations of functions from a class $\mathbb{F}$, then one can construct a hard function for $\mathcal{C}$ on X by constructing a hard function for the class $\mathbb{F}$ on X. (Using Proposition 2.2 from Section 2.3 will give us a low discrepancy coloring from these hard functions). We start by introducing the notion of a (W, δ)-approximating class.

Definition 3.1 ((W, δ)-approximating class)**.** *Let $\mathcal{C}$ be a class of bounded functions on X. A class of bounded functions $\mathbb{F}$ on X is a (W, δ)-approximating class for $\mathcal{C}$ if for every $c \in \mathcal{C}$, there exist reals α_i for $1 \leq i \leq r$ satisfying $\sum_{i=1}^{r} |\alpha_i| \leq W$ and r functions, $f_1, f_2, .., f_r \in \mathbb{F}$ such that $\mathbb{E}_{x \sim \mathcal{U}_X}[|c(x) - \sum_{i=1}^{r} \alpha_i f_i(x)|] \leq \delta$.*

We now show that if a class $\mathcal{C}$ has a (W, δ) approximating class $\mathbb{F}$ on X, then a hard function for $\mathbb{F}$ is a hard function for $\mathcal{C}$ on X.

Theorem 3.2 (Hard Functions Through Low Weight Approximators)**.** *Let $\mathcal{C}$ be a class of Boolean functions on an arbitrary finite set X. Suppose a class of bounded functions $\mathbb{F}$ is a (W,δ)-approximating class for $\mathcal{C}$ on X. If $\chi : X \to \{-1,1\}$ is $\frac{\epsilon}{W}$-hard for $\mathbb{F} \cup \mathbb{F}^{abs}$ on X then χ is $(\epsilon+\delta)$-hard for $\mathcal{C}$ on X.*

Remark 3.1. *Note that for a hard function χ for a bounded function class $\mathcal{C}$ to be a low discrepancy coloring for the class, we require that χ be a hard function for the class $\mathcal{C}^{abs}$ also (see Section 2.3, Proposition 2.2). However, when $\mathcal{C}$ is a class of Boolean functions, $\mathcal{C}^{abs}$ is just the constant function $\mathbb{1}$. Thus the above theorem gives us both a hard function and a low discrepancy coloring for the class $\mathcal{C}$.*

We use the following simple lemma which shows the relationship between hard functions for a bounded function and its approximator.

Lemma 3.3 (Hard Functions through ℓ_1 Approximation)**.** *Let $c : X \to [-1,1]$ and $p : X \to \mathbb{R}$ be such that $\mathbb{E}_{x\sim\mathcal{U}_X}|p(x)-c(x)| \le \delta$. Suppose $\chi : X \to \{-1,1\}$ satisfies $|\langle \chi, p\rangle_{\mathcal{U}_X}| \le \epsilon$. Then, $|\langle \chi, c(x)\rangle_{\mathcal{U}_X}| \le \epsilon+\delta$.*

Proof. Define $e : X \to \mathbb{R}$ such that $e(x) = c(x) - p(x)$ for every $x \in X$. We have:

$$\begin{aligned}
|\mathop{\mathbb{E}}_{x\sim\mathcal{U}_X}[\chi(x)\cdot c(x)]| &= |\mathop{\mathbb{E}}_{x\sim\mathcal{U}_X}[\chi(x)\cdot(p(x)+e(x))]| \\
&\le |\mathop{\mathbb{E}}_{x\sim\mathcal{U}_X}[\chi(x)\cdot p(x)]| + |\mathop{\mathbb{E}}_{x\sim\mathcal{U}_X}[\chi(x)\cdot e(x)]| \\
&\le \epsilon+\delta
\end{aligned}$$

□

We now complete the proof of the Theorem 3.2.

Proof of Theorem 3.2. First we show the claim for $\mathcal{C}$. Let $c \in \mathcal{C}$ be an arbitrary member of $\mathcal{C}$. Since $\mathbb{F}$ is a (W,δ)-approximating class for $\mathcal{C}$, there exist α_i for $1 \le i \le r$ and r functions $f_1, f_2, \cdots, f_r$ all in $\mathbb{F}$, such that $\mathbb{E}_{x\sim\mathcal{U}_X}[|c(x) - \sum_{i=1}^r \alpha_i f_i(x)|] \le \delta$. Thus, for $c \in \mathcal{C}$:

$$\begin{aligned}
|\langle \sum_{i=1}^r \alpha_i f_i, \chi\rangle_{\mathcal{U}_X}| = |\mathop{\mathbb{E}}_{x\sim\mathcal{U}_X}[\chi(x)\cdot\sum_{i=1}^r \alpha_i f_i(x)]| &= |\sum_{i=1}^r \alpha_i \mathop{\mathbb{E}}_{x\sim\mathcal{U}_X}[\chi(x)\cdot f_i(x)]| \\
&\le |\sum_{i=1}^r \alpha_i \frac{\epsilon}{W}| \le \left(\sum_{i=1}^r |\alpha_i|\right)\frac{\epsilon}{W} \\
&\le \epsilon \qquad (1)
\end{aligned}$$

Note that by construction χ is ϵ-hard for $\mathbb{1}$ on $\mathcal{U}_X$. Thus, using Lemma 3.3 and (1) we obtain:

$$|\langle c, \chi\rangle_{\mathcal{U}_X}| \le \epsilon+\delta.$$

□

Fix any finite set $X \subseteq [-1,1]^n$. Consider the class of all monomials of degree at most k in n variables as functions on X (denoted by $\mathcal{M}_k$). These monomials form a class of size $O(n^k)$ of bounded functions. We can construct a hard function for the class of all monomials of degree at most k using the algorithm from Lemma 2.9. Thus we have the following result.

Lemma 3.4 (Hard Functions for Monomials)**.** *Let $\mathcal{M}_k$ be the class of all monomials of degree at most k in n variables on $X \subseteq [-1,1]^n$. Then the algorithm from Lemma 2.9 runs in time $O(n^k|X|)$ and produces a function $\chi : X \to \{-1,1\}$ such that for every monomial $m \in \mathcal{M}_k$, $|\langle m, \chi\rangle| = O\left(\sqrt{\frac{k \log n}{|X|}}\right)$.*

Now, using Theorem 3.2 and Lemma 3.4 we can translate the hard function for monomials to a hard function for functions approximated by polynomials on arbitrary finite domains.

Corollary 3.5 (Hard Functions for Functions Approximated by Low Degree Polynomials)**.** *Let $\mathcal{C}$ be a class of Boolean functions on $[-1,1]^n$ and $X \subseteq [-1,1]^n$ a finite set such that for each $c \in \mathcal{C}$ there is a polynomial $p : \mathbb{R}^n \to \mathbb{R}$, of degree at most k and weight at most W which δ-approximates c in ℓ_1-norm on X. Then, there exists an algorithm that runs in time $\mathsf{poly}(n^k) \cdot |X|$ that constructs a function $\chi : X \to \{-1,1\}$ that is $(\epsilon + \delta)$-hard for $\mathcal{C}$ on X where $\epsilon = O\left(W\sqrt{\frac{k \log n}{|X|}}\right)$.*

Hard Functions for Conjunctions. To obtain our result for the class of all conjunctions (and disjunctions), we use the following result which shows that these functions have uniformly approximating polynomials of low degree and weight. Although [OS03] only talks about the degree of their approximating polynomial, a simple inspection of their proof (which uses Chebyshev polynomials of the first kind) shows the weight bound noted below.

Theorem 3.6 (Uniform Approximation on $\{0,1\}^n$ [OS03])**.** *Let $f : \{-1,1\}^n \to \{-1,1\}$ be any Boolean conjunction (or disjunction). Then, for every $\delta > 0$, there exists a polynomial $p : \mathbb{R}^n \to \mathbb{R}$ of degree $O(\sqrt{n} \cdot (\log n + \log \frac{1}{\delta}))$ such that for all $x \in \{-1,1\}^n$, we have $|f(x) - p(x)| \leq \delta$. Further the weight of this polynomial is bounded by $n^{O(\sqrt{n} \cdot (\log n + \log \frac{1}{\delta}))}$.*

As a result of Theorem 3.6 and Corollary 3.5 we obtain the following construction of hard functions for the class of conjunctions which generalizes to all linear sized formulas of constant depth.

Corollary 3.7. *Let $\mathcal{C}$ denote the class of Boolean conjunctions. There exists an algorithm that runs in time $n^{O(\sqrt{n} \cdot (\log n + \log \frac{1}{\epsilon}))}|X|$ and computes a ϵ-hard function for the class $\mathcal{C}$ for any $\epsilon = \frac{1}{\sqrt{|X|}} \cdot n^{\Omega(\sqrt{n} \log n)}$.*

Remark 3.2. *The uniform approximation theorem in [OS03] shows that for the more general class of formulas F of size s and depth k, there is a polynomial that uniformly δ-approximates F on $\{-1,1\}^n$ of degree $O(\sqrt{s} \cdot (\log^{2k} s + \log^k \frac{1}{\delta}))$ and weight $n^{O(\sqrt{s} \cdot (\log^{2k} s + \log^{2k} \frac{1}{\delta}))}$. Our result for constructing hard functions directly translates to this more general class.*

Hard Functions for Decision Trees. We now use our technique to construct hard functions for the class of bounded depth decision trees.

Constant depth decision trees are computed exactly by low degree polynomials. This enables us to use Theorem 3.5 to obtain a polynomial time algorithm for finding a hard function for this class.

Theorem 3.8. *Let $f : \{-1,1\}^n \to \{-1,1\}$ be a Boolean function computed by a decision tree of depth k. Then f is* exactly *computed by a real valued polynomial of degree k and weight at most 2^k.*

This result is well known. We include a proof in Section C of the appendix for completeness. As a corollary of Theorem 3.8 and Theorem 3.5 we obtain a hard function for the class of all bounded depth decision trees.

Corollary 3.9. *Let DT^k denote the class of all Boolean functions on n inputs computed by depth k decision trees. Then for any $X \subseteq \{-1,1\}^n$, there exists a deterministic algorithm which runs in time $\mathsf{poly}(n^k) \cdot |X|$ and outputs an ϵ-hard function for the class DT^k where $\epsilon = O(2^k \sqrt{\frac{k \log n}{|X|}})$.*

4 ϵ-Approximation from Hard Functions

In this section we show how to construct ϵ-approximations from hard functions combining our techniques with those of Chazelle-Matoušek's [CM96].

All such algorithms are based on the simple observation that an ϵ-hard function for the class $\mathcal{C} \cup \mathbb{1}$ on X can be easily used to obtain a 2ϵ-approximation of size $\frac{|X|}{2}$ for $\mathcal{C}$ on X. Specifically, if $\chi : X \to \{-1,1\}$ is ϵ-hard for the class $\mathcal{C}$ on X then an arbitrary subset of size $\frac{|X|}{2}$ of the larger of the two sets $\chi^{-1}(-1)$ and $\chi^{-1}(1)$ is a 2ϵ-approximation for $\mathcal{C}$ on X. This observation itself can be repeatedly used to obtain an ϵ-approximation. However, the running time of this algorithm grows exponentially in $|X|$. The reader can find the details of the observation above and a simple analysis of the repeated halving algorithm in Section B of the appendix.

4.1 ϵ-Approximation from Hard Functions in Time Linear in $|X|$

In this section we show how to combine our techniques of constructing hard functions with those of Chazelle and Matoušek's to obtain an algorithm to construct ϵ-approximation that runs in time linear in $|X|$.

Using the repeated halving algorithm with our hard function constructions for functions approximated by polynomials of degree at most k and weight at most W, we obtain an algorithm that runs in time $\mathsf{poly}(W, n^k, \frac{1}{\epsilon}, |X|)$. However in many applications in computational geometry, such as range counting [AHS07], the dimension n is small and the number of points in the domain X is large. In such situations, we want the running time of the algorithms to depend linearly on $|X|$ as opposed to a fixed polynomial in $|X|$ guaranteed by the repeated halving method.

We adapt the technique of [CM96] here to work with our hard function construction to produce ϵ-approximations in time bound that depends linearly in $|X|$ (Algorithm 1). Our algorithm for constructing hard functions for various classes produces a function that is $g'(n,\mathcal{C})\sqrt{\frac{\log |X|}{|X|}}$-hard for $\mathcal{C}$ on X. Here $g'(n,\mathcal{C})$ depends on the class $\mathcal{C}$ at hand. Using Lemma B.1 we can obtain a $2g'(n,\mathcal{C})\sqrt{\frac{\log |X|}{|X|}} = g(n,\mathcal{C})\sqrt{\frac{\log |X|}{|X|}}$-approximation for $\mathcal{C}$ of size $\frac{|X|}{2}$. We use this procedure as a blackbox and call it algorithm $\mathcal{A}$ (the halving algorithm) in the following description. Thus, given input X, $\mathcal{A}$ returns an $g(n,\mathcal{C})\sqrt{\frac{\log |X|}{|X|}}$-approximation of size $\frac{|X|}{2}$ for class $\mathcal{C}$.

4.1.1 Algorithm

The algorithm partitions X into disjoint subsets of size K. It then pairs these subsets and replaces them with their union (*merge step*). Now, it constructs an ϵ-approximation of size K for each such subset by first constructing a hard function and then using Lemma B.1 (*halving step*). It is easy to see that repeating these steps will produce an ϵ-approximation. However, to optimize the running

Algorithm 1 Compute ϵ-approximation using Hard Function Construction for class $\mathcal{C}$

Require: Algorithm $\mathcal{A}$ (the halving algorithm), a finite subset X of the domain of $\mathcal{C}$ and a parameter $\epsilon > 0$.

1: Partition X into arbitrary subsets of size $K = c_1 \frac{g(n,\mathcal{C})^6}{\epsilon^2} \log \frac{g(n,\mathcal{C})}{\epsilon}$ for a large enough constant c_1.
2: **while** there is more than one set in the current partition: **do**
3: **for** $R = g(n,\mathcal{C})^2 + 1$ rounds, whenever there is more than one set remaining **do**
4: Arbitrarily group the sets into pairs such that each set appears in exactly one pair.
5: Replace every pair by their union.
6: For every set S that remains, run $\mathcal{A}$ on S for the class $\mathcal{C} \cup \mathbb{1}$ and replace it by the set returned by $\mathcal{A}$.
7: **end for**
8: Arbitrarily group the sets into pairs such that each set appears in exactly one pair and replace every pair by their union.
9: **end while**
10: To the remaining set, repeatedly apply $\mathcal{A}$ till a set of size $(c_2 \frac{g(n,\mathcal{C})^2}{\epsilon^2} \log \frac{g(n,\mathcal{C})}{\epsilon})$ (for some large enough c_2) remains.
11: Return the resulting set.

time, the algorithm uses a clever combination of these merge and halving steps instead of the naive repetition. For details of the technique, the reader may consult [CM96] and [Cha01].

Comparison with Chazelle-Matoušek : In [CM96] the hard function for $\mathcal{C}$ is constructed using Lemma 2.9 directly for the class $\mathcal{C}$. The parameters K and R are fixed based on the VC dimension of the class $\mathcal{C}$. This is because the algorithm in Lemma 2.9, when used directly on $\mathcal{C}$ runs in time $|X|^{O(d)}$ where d is the VC dimension of $\mathcal{C}$. In contrast, our construction of hard functions are faster for the classes we deal with and depend on the parameters defining the class of uniformly approximating polynomials.

We also note that in [CM96] , to compute hard functions (low discrepancy colorings) for $\mathcal{C}$ on any subset of the input set X, the algorithm needs to efficiently enumerate all the distinct configurations produced by the class $\mathcal{C}$ on any subset of the domain $Y \subseteq X$. The authors call such a procedure a *subsystem oracle* for $\mathcal{C}$ and the algorithm functions *assuming* the existence of such an oracle that runs in time $|Y|^{d+1}$ where d is the VC-Dimension of the class $\mathcal{C}$. (This time bound is fixed because Sauer's Lemma [Sau72, She72, VC71] guarantees that the number of configurations that the class $\mathcal{C}$ produces on any X is bounded by $|X|^d$.) Such an oracle can be shown to exist for the class of halfspaces. In our method of constructing hard functions, we do *not* need any such assumption about the existence of a *subsystem oracle*.

Corollary 4.1 (ϵ-Approximation For Functions Approximated by Low Degree Polynomials)**.** *Let $\mathcal{C}$ be any class of Boolean functions on n inputs and $X \subseteq [-1,1]^n$ such that $\mathcal{C}$ is δ-uniformly approximated by the class real valued polynomials of degree at most k and weight at most W on X. Then, Algorithm 1 combined with the hard function construction from Theorem 3.5, constructs an $(\epsilon + \delta \log |X|)$-approximation for $\mathcal{C}$ of size $O(\frac{W^2 k}{\epsilon^2} \log \frac{Wk}{\epsilon}) \cdot \mathsf{poly}(n^k)$ and runs in time $\mathsf{poly}(n^k) \cdot |X| \cdot \frac{W^6 k^3}{\epsilon^2} \log \frac{Wk}{\epsilon}$ for any $\epsilon = \Omega\left(W \sqrt{\frac{k \log n}{|X|}}\right)$.*

As decision trees are exactly computed by polynomials of degree k and weight 2^k (Theorem 3.8), using the above result we have:

Corollary 4.2 (ϵ-Approximation For Decision Trees)**.** *Let DT^k be the class of all Boolean functions on n inputs computed by decision trees of depth at most k and $X \subseteq \{-1,1\}^n$. Then, Algorithm 1 combined with the hard function construction from Corollary 3.9, constructs an ϵ-approximation for DT^k of size $O(\frac{2^{2k}k^2}{\epsilon^2}\log\frac{k}{\epsilon})\mathsf{poly}(n^k)$ and runs in time $O(\frac{2^{6k}k^4}{\epsilon^2}\log\frac{k}{\epsilon})\cdot\mathsf{poly}(n^k)\cdot|X|$ for any $\epsilon = \Omega\left(2^k\sqrt{\frac{k\log n}{|X|}}\right)$. Note that the size of the ϵ-approximation produced is $n^{O(k)}$ and time complexity is $n^{O(k)}|X|$ for this range of ϵ.*

Similarly, combining the construction in Corollary 4.1 with Corollary 3.7 we obtain the following construction ϵ-approximation for the class of Boolean conjunctions.

Corollary 4.3 (ϵ-Approximations for Conjunctions)**.** *Let $\mathcal{C}$ be the class of all Boolean conjunctions on n inputs. Then, Algorithm 1 combined with the hard function construction from Corollary 3.7, constructs an ϵ-approximation for $\mathcal{C}$ of size $\frac{n^{O(\sqrt{n}\cdot\log n)}}{\epsilon^2}$ and runs in time $n^{O(\sqrt{n}\cdot\log n)}\cdot|X|$ whenever $\epsilon \geq n^{\Omega(\sqrt{n}\cdot\log n)}\cdot\sqrt{\frac{\log n}{|X|}}$.*

5 Hardness of Constructing ϵ-Approximations

In this section we show that an efficient deterministic algorithm to compute an ϵ-approximation (or a hard function) for the class of polynomial size circuits on arbitrary inputs $X \subseteq \{-1,1\}^n$ implies $\mathsf{P} = \mathsf{BPP}$.

We show that if we have such an efficient algorithm to construct an ϵ-approximation for the class of polynomial size circuits, then we can construct a function which is computable in E but cannot be computed by any circuit of size at most $2^{\delta n}$ for a fixed $\delta > 0$. The idea to construct this hard function is to use the characteristic function of the efficiently computed ϵ approximation. We then use the following celebrated theorem of Impagliazzo and Wigderson to complete our proof.

Theorem 5.1 (Impagliazzo-Wigderson [IW96])**.** *Suppose there is a function computable in time $2^{O(n)}$ but cannot be computed by any circuit of size at most $2^{\delta n}$ for some fixed $\delta > 0$. Then $\mathsf{P} = \mathsf{BPP}$.*

We first show the hardness of constructing a hard function.

Theorem 5.2 (Hardness of Constructing a Hard Function)**.** *For any $k \geq 1$, let $\mathcal{C}_k^m$ be the class of all Boolean circuits of size at most m^k on m inputs (each such circuit computes a function from $\{-1,1\}^m \to \{-1,1\}$). Suppose $\mathcal{A}^k$ is an algorithm that takes as input any $X \subseteq \{-1,1\}^m$ and returns a function $f : X \to \{-1,1\}$ such that for every $c \in \mathcal{C}_k^m$, $|\langle f, c\rangle| < 1$ and runs in time $\mathsf{poly}(|X|, \frac{1}{\epsilon}, m^k)$. Then there exists a function $L : \{-1,1\}^n \to \{-1,1\}$ computable in time $2^{O(n)}$ such that for any $0 < \delta < \frac{1}{k}$, no circuit on n inputs of size at most $2^{k\delta n}$ ($= 2^{(1-\beta)n}$ for some $\beta > 0$) can compute L.*

Proof. Fix $m = 2^{\delta n}$. Our argument involves constructing a hard function for $\mathcal{C}_k^m$ using $\mathcal{A}^k$ on an appropriate set $X \subseteq \{-1,1\}^m$. Let $X = \{y \cdot 0^{m-n} \mid y \in \{-1,1\}^n\}$ be the set of all possible binary strings of length n padded with $m-n$ 0s. Notice that $\mathcal{C}_k^m$ is a polynomial size circuit family on m

inputs. Construct a hard function $f : X \to \{-1,1\}$ for $\mathcal{C}_k^m$ on X using $\mathcal{A}^k$. Thus, for every $c \in \mathcal{C}_k^m$, there exists a point $x \in X$ such that $f(x) \neq c(x)$.

We now define a function $L : \{-1,1\}^n \to \{-1,1\}$, using the function f constructed above, such that L is computable in deterministic time $2^{O(n)}$ but is worst-case hard to compute for every circuit of size at most $2^{\delta n}$. This will complete our proof.

For $x \in \{-1,1\}^n$, define $L(x) = f(x \cdot 0^{m-n})$. Notice that $x0^{m-n} \in X$ for every $x \in \{-1,1\}^n$. We claim that L is worst case hard for every circuit on n inputs of size at most $2^{k\delta n}$. Suppose, if possible, there is a circuit c on n inputs of size at most $2^{k\delta n}$ such that $c(x) = L(x)$ for every $x \in \{-1,1\}^n$. Construct a circuit c' on m inputs such that $c'(x \cdot 0^{m-n}) = c(x)$ for every $x \in \{-1,1\}^n$. Notice that $c' \in \mathcal{C}_k^m$ since it has size $2^{k\delta n}$ and computes f correctly on all of X. This is a contradiction since f is worst-case hard for $\mathcal{C}_k^m$ on X by construction. □

Remark 5.1. *Observe that by the argument above, an efficient algorithm for constructing a* worst-case *hard function on arbitrary domains for linear sized* circuits *would result in* $\mathsf{P} = \mathsf{BPP}$ *(using Impagliazzo-Wigderson[IW96]). Contrast this with our subexponential-time construction of hard functions for the class of linear-sized* formulas *in Corollary 3.7 (and Remark 3.2). This hints at the inherent difficulty in constructing hard functions and ϵ-approximations for expressive concept classes (see Theorem 5.3 below).*

We now show that an algorithm for computing an ϵ-approximation of any non-trivial size for polynomial sized circuits over arbitrary domain X implies the existence of the algorithm $\mathcal{A}^k$ as in the statement of Theorem 5.2. Combined with Theorem 5.2 and 5.1 we have the required result.

Theorem 5.3. *Let $\mathcal{C}_k^m$ be the class of all Boolean circuits of size at most m^k on m inputs (each such circuit computes a function from $\{-1,1\}^m \to \{-1,1\}$). Suppose, there exists an algorithm $\mathcal{B}^k$ for some $k > 0$, which takes as input $X \subseteq \{-1,1\}^m$ and computes an $\frac{1}{3}$-approximation for $\mathcal{C}_m^k$ on X of size at most $\frac{|X|}{2}$ in time $\mathsf{poly}(|X|, m^k)$. Then $\mathsf{P} = \mathsf{BPP}$.*

Proof. We first use $\mathcal{B}^k$ to design an algorithm $\mathcal{A}^k$ that constructs a hard function for $\mathcal{C}_m^k$ on any given input $X \subseteq \{-1,1\}^m$. For any $X \subseteq \{-1,1\}^m$, algorithm $\mathcal{A}^k$ uses the algorithm $\mathcal{B}^k$ with input X to obtain a $\frac{1}{3}$-approximation $Y \subseteq X$ for the class C_k^m and outputs the function $\mathbb{1}_Y : X \to \{-1,1\}$ where $\mathbb{1}_Y(x) = 1$ iff $x \in Y$ otherwise -1.

We first argue that $\mathcal{A}^k$ indeed constructs a function $\mathbb{1}_Y$ such that $|\langle c, \mathbb{1}_Y \rangle| < 1$. Suppose if possible, $\mathbb{1}_Y(x) = c(x)$ for every $x \in X$ for some $c \in C_d^m$. (The case where $\mathbb{1}_Y(x) = -c(x)$ for every $x \in X$ can be handled similarly.) Since Y is a $\frac{1}{3}$-approximation for c, we must have $|\Pr_{x \sim \mathcal{U}_Y}[c(x) = 1] - \Pr_{x \sim \mathcal{U}_X}[c(x) = 1]| \leq \frac{1}{3}$.

But since $c = \mathbb{1}_Y$ and $|Y| \leq \frac{|X|}{2}$, we have $\Pr_{x \sim \mathcal{U}_X}[c(x) = 1] = \frac{|Y|}{|X|} \leq 1/2$ and $\Pr_{x \sim \mathcal{U}_Y}[c(x) = 1] = 1$. This implies that $|\Pr_{x \sim \mathcal{U}_Y}[c(x) = 1] - \Pr_{x \sim \mathcal{U}_X}[c(x) = 1]| \geq \frac{1}{2}$, which contradicts the fact that Y is a $\frac{1}{3}$-approximation.

Now using Theorem 5.2 with $\mathcal{A}^k$ given by the above construction we obtain, for every n, a function L computable in $2^{O(n)}$ which is not computed by any circuit of size $2^{\delta n}$. Using Theorem 5.1 we have the result. □

References

[AHS07] Boris Aronov, Sariel Har-Peled, and Micha Sharir. On approximate halfspace range counting and relative epsilon-approximations. In *Proceedings of the Twenty-Third An-*

nual Symposium on Computational Geometry, SCG '07, pages 327–336, New York, NY, USA, 2007. ACM.

[Ban10] Nikhil Bansal. Constructive algorithms for discrepancy minimization. In *FOCS*, pages 3–10, 2010.

[CM96] Bernard Chazelle and Jirí Matousek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *J. Algorithms*, 21(3):579–597, 1996.

[Cha01] Bernard Chazelle. *The discrepancy method: randomness and complexity*. Cambridge University Press, New York, NY, USA, 2000.

[FL11] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *STOC*, pages 569–578, 2011.

[IW96] Russell Impagliazzo and Avi Wigderson. P = BPP unless E has sub-exponential circuits: Derandomizing the XOR lemma (preliminary version). In *Proceedings of the 29th STOC*, pages 220–229. ACM Press, 1996.

[LLS01] Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62:2001, 2000.

[LM12] S. Lovett and R. Meka. Constructive Discrepancy Minimization by Walking on The Edges. *ArXiv e-prints*, March 2012.

[Mat99] Jiri Matousek. *Geometric Discrepancy: An Illustrated Guide (Algorithms and Combinatorics)*. Springer, 1 edition, 1999.

[Nis92a] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[Nis92b] Noam Nisan. RL $\subseteq$ SC. In *STOC*, pages 619–623, 1992.

[OS03] Ryan O'Donnell and Rocco A. Servedio. New degree bounds for polynomial threshold functions. In *STOC*, pages 325–334, 2003.

[PA95] Janos Pach and Pankaj Agrawal. *Combinatorial Geometry*. Wiley-Interscience, October 1995.

[SB11] Camil Demetrescu and Magnús M. Halldórsson, editors. *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, volume 6942 of *Lecture Notes in Computer Science*. Springer, 2011.

[Sau72] Norbert Sauer. On the density of families of sets. *J. Comb. Theory, Ser. A*, 13(1):145–147, 1972.

[She72] Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41:247–261, 1972.

[Siv02] D. Sivakumar. Algorithmic derandomization via complexity theory. In *IEEE Conference on Computational Complexity*, page 10, 2002.

[Spe85] Joel Spencer. Six standard deviations suffice. *Transactions of the American Mathematical Society*, 289(2):pp. 679–706, 1985.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[Vap] Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998.

A Discrepancy

A.1 Discrepancy of Random Coloring

Proof of Lemma 2.8. The proof is a standard application of Chernoff-Hoeffding Bounds. Note that for every $c \in \mathcal{C}$, $|\sum_{x \in X} c(x)| \leq n$, thus, by Chernoff-Hoeffding Bound, for any $c \in \mathcal{C}$,

$$\Pr[|\chi(c)| \geq r\sqrt{n}] \leq 2e^{-\frac{1}{4}r^2}.$$

for any $r \leq \sqrt{n}$. Using $r = \sqrt{4 \log 4m}$, we get,

$$\Pr[|\chi(c)| \geq \sqrt{4n \log 4m}] \leq \frac{1}{2m}.$$

Using the union bound, we obtain the result. □

B The Repeated Halving Algorithm

We first give a proof of the observation that hard functions yield approximations of half the size of the domain. We then give a simple analysis of the repeated halving algorithm.

Lemma B.1. *Let $\mathcal{C}$ be a class of Boolean functions on X. Suppose $\chi : X \to \{-1, 1\}$ is ϵ-hard for $\mathcal{C} \cup \mathbb{1}$ on X. Let $\chi^{-1}(z)$ be the larger set between $\chi^{-1}(-1)$ and $\chi^{-1}(1)$. Construct Y by taking any subset of $\chi^{-1}(z)$ of size $\frac{|X|}{2}$. Then Y is a 2ϵ-approximation for $\mathcal{C}$ on X.*

Proof. Let $T = \chi^{-1}(z)$ and Y be the set obtained by choosing any subset of size $\frac{|X|}{2}$ from T. Let $c \in \mathcal{C}$ be any member function in the class $\mathcal{C}$. Since χ is ϵ-hard for $\mathbf{1} \in \mathcal{C}$ on X, we must have $|\chi^{-1}(z)| - |\chi^{-1}(-z)| \leq \epsilon|X|$. Thus $|\chi^{-1}(z)| \leq \frac{|X|}{2} + \frac{\epsilon|X|}{2}$. Let $S_c = \{x \in X \mid c(x) = 1\}$ denote the set of points in X where c takes the value 1.

$$\begin{aligned}
|\Pr_{y \sim \mathcal{U}_Y}[c(y) = 1] - \Pr_{x \sim \mathcal{U}_X}[c(x) = 1]| &= \left|\frac{|S_c \cap Y|}{|Y|} - \frac{|S_c|}{|X|}\right| = \frac{1}{|X|}\left|2|S_c \cap Y| - |S_c|\right| \\
&\leq \frac{1}{|X|}\left|2|S_c \cap Y| - 2|S_c \cap \chi^{-1}(z)|\right| \\
&+ \frac{1}{|X|}\left|2|S_c \cap \chi^{-1}(z)| - |S_c|\right| \leq \frac{\epsilon|X|}{|X|} + \frac{\epsilon|X|}{|X|} \qquad (2) \\
&\leq 2\epsilon
\end{aligned}$$

where in equation (2) the first term comes from the bound on the difference in size between $|\chi^{-1}(z)|$ and $|Y| = \frac{|X|}{2}$ and the second term comes from χ being ϵ hard for $\mathcal{C} \cup \mathbb{1}$ on X. □

Lemma B.2 (Compositions of ϵ-Approximations)**.** *Suppose Y is an ϵ-approximation for $\mathcal{C}$ on X and Z is an ϵ'-approximation for $\mathcal{C}$ on Y. Then, Z is an $(\epsilon+\epsilon')$-approximation for $\mathcal{C}$ on X.*

Proof. Y is an ϵ-approximation for $\mathcal{C}$ on X and Z is an ϵ-approximation for $\mathcal{C}$ on Y. This means that for any $c \in \mathcal{C}$,

$$\left|\frac{|\mathbb{1}_c \cap Y|}{|Y|} - \frac{|\mathbb{1}_c|}{|X|}\right| \leq \epsilon \tag{3}$$

and noticing that $(\mathbb{1}_c \cap Y) \cap Z = \mathbb{1}_c \cap Z$,

$$\left|\frac{|\mathbb{1}_c \cap Z|}{|Z|} - \frac{|\mathbb{1}_c \cap Y|}{|Y|}\right| \leq \epsilon' \tag{4}$$

Thus, using (3) and (4),

$$\begin{aligned}\left|\frac{|\mathbb{1}_c \cap Z|}{|Z|} - \frac{|\mathbb{1}_c|}{|X|}\right| &\leq \left|\frac{|\mathbb{1}_c \cap Z|}{|Z|} - \frac{|\mathbb{1}_c \cap Y|}{|Y|}\right| + \left|\frac{|\mathbb{1}_c \cap Y|}{|Y|} - \frac{|\mathbb{1}_c|}{|X|}\right| \\ &\leq \epsilon + \epsilon'\end{aligned}$$

□

In this algorithm we use $\mathcal{A}$ to construct a low discrepancy coloring of the given domain X with respect to the class $\mathcal{C}$. We then take exactly half of the number of points in X from the larger color class to obtain Y of size $|X|/2$ (*halving*). We then treat Y as the new domain and repeat. If we have a procedure to construct an ϵ-hard function for $\mathcal{C}$ on X in time $T(\mathcal{C}, |X|)$, then this procedure takes $O(T(\mathcal{C}, |X|) \cdot \log(t)$ time to construct an $\log t \cdot \epsilon$-approximation of X with respect to the class $\mathcal{C}$ of size $\frac{|X|}{t}$.

Theorem B.3 (ϵ-approximation through Hard Functions)**.** *Let $\mathcal{C}$ be a Boolean function class on a finite subset $X \subseteq \mathbb{R}^n$. Let $\mathcal{A}$ be an algorithm that runs in time $T(\mathcal{C}, |Y|)$ and produces an ϵ-hard function for $\mathcal{C}$ on $Y \subseteq X$. Then there exists an algorithm that runs in time $T(\mathcal{C}, |X|) \cdot \log t$ that produces an $\epsilon \cdot \log t$-approximation of size $\frac{|X|}{t}$ for $\mathcal{C}$ on X.*

Proof. The proof of this theorem proceeds by using the results above to show that a single iteration of the above algorithm yields an ϵ-approximation of half the size of the set we started with and then checking how the errors-in-approximation add up in successive iterations.

We use the following algorithm to construct the ϵ-approximation using the algorithm $\mathcal{A}$ as a black-box. Without loss of generality we assume that $\mathcal{C}$ has the constant function $\mathbb{1}$ in it.

For an analysis of this algorithm first note that the size of the set returned is at most $\frac{|X|}{t}$. Further, note that Lemma B.1 guarantees us an ϵ approximation Y' for the class $\mathcal{C}$ with respect to the set Y in every iteration. By Lemma B.2 we know that in iteration k, Y' is a $k\epsilon$ approximation. Finally observe that after the k^{th} iteration the size of Y is $\frac{|X|}{2^k}$. Thus we have the result. □

Now, for a class $\mathcal{C}$, given an algorithm to construct ϵ-hard function on any subset X of its domain that runs in time $T(\mathcal{C}, |X|)$, there is an easy algorithm to construct a subset $Y \subseteq X$ of size $\frac{|X|}{t}$ that is a $(\epsilon \log t)$-approximation for $\mathcal{C}$ on X using the idea in the result above. This well known algorithm is referred to as the repeated halving method. We also include a simple analysis of this algorithm.

Algorithm 2 Construct ϵ-Approximation for Boolean class $\mathcal{C}$

Require: Set $X \subseteq \mathbb{R}^n$, Algorithm $\mathcal{A}$

$Y \leftarrow X$
while $|Y| > \frac{|X|}{t}$ **do**
  Run $\mathcal{A}$ to construct an ϵ-hard function for $\mathcal{C}$ on Y.
  Let $z \leftarrow \arg\max_{i \in \{-1,1\}} |\chi^{-1}(i)|$
  Take an aribtrary subset of size $\frac{|Y|}{2}$ from $\chi^{-1}(z)$ to produce Y'
  Set $Y \leftarrow Y'$
end while
Return Y

Theorem B.4 (ϵ-Approximations by Repeated Halving)**.** *Let $\mathcal{C}$ be a Boolean function class on a finite subset $X \subseteq \mathbb{R}^n$. Let $\mathcal{A}$ be an algorithm such that given any $Y \subseteq X$, $\mathcal{A}$ produces an ϵ-hard function for $\mathcal{C}$ on Y in time $T(\mathcal{C}, |Y|)$. Then there exists an algorithm that runs in time $O(T(\mathcal{C}, |X|) \cdot \log t \cdot |X|)$ that produces an $\epsilon \cdot \log t$-approximation of size $\frac{|X|}{t}$ for $\mathcal{C}$ on X.*

C Decision Trees

First we provide an easy lower bound on the VC dimension of bounded depth decision trees by proving that the VC dimension of the literals is $\Omega(\log n)$. Then we give the proof of Theorem 3.8.

Lemma C.1. *Let DT^k be the class of Boolean functions computed by depth k decision trees. Then the VC dimension of the class DT^k is $\Omega(\log n)$.*

Proof. First note that x_1,..., x_n are contained in DT^k. We estimate the VC dimension of literals to complete our argument. For this we show a set of $\log n$ points from $\{-1, 1\}^n$ that are shattered by the literals. Let s_1,..., $s_{\log n}$ be the points in $\{-1, 1\}^n$ chosen such that for the coordinate i, the $\log n$-bit string $s_1(i)s_2(i)...s_{\log n}(i)$ is the binary representation of the integer $(i-1)$. It can now be verified that the set of $\log n$ strings defined above are shattered by the class of literals. □

Proof of Theorem 3.8. Let f be a function computed by a depth k decision tree. Consider any leaf i in the decision tree of f. Let l_i be the label of the leaf i (i.e the output on reaching leaf i) and define function f_i as

$$f_i(x) = \begin{cases} 1 & \text{if on input } x \text{ leaf } i \text{ is reached in the decision tree} \\ 0 & \text{otherwise} \end{cases}$$

We claim that $f(x) = \sum_{i \in \text{ leaf}} l_i f_i(x)$. To prove this it is enough to show that for any x, exactly one of the f_i's is 1. Let the variables x_{j_1}, x_{j_2},..., x_{j_k} be present on the path P_i from the root to leaf i and set to 1 or -1 on this path. For any $x_{j_m} \in P_i$ define

$$y_{j_m} = \begin{cases} (\frac{1}{2} + \frac{x_{j_m}}{2}) & \text{if } x_{j_m} \in P_i \text{ and is set to 1 in } P_i \\ (\frac{1}{2} - \frac{x_{j_m}}{2}) & \text{if } x_{j_m} \in P_i \text{ and is set to } -1 \text{ in } P_i \end{cases}$$

It is easy to see that $f_i(x) = y_{j_1} y_{j_2} ... y_{j_k}$. Also note that on the paths to reach any two leaves i, i', there must exist a common node x_k which is set to 1 in order to reach one vertex and -1 to reach

the other. Thus for any input x, exactly one of y_k and y'_k is 0 and the other is 1 which shows that exactly one of f_i and $f_{i'}$ is 1 on any input. This proves our claim.

Thus note that we have a polynomial of degree at most d for each f_i and hence a polynomial of degree at most d exactly computes f. Further since the weight of the polynomial computing f_i is exactly 1 and since there are at most 2^d leaves, the weight of the polynomial computing f is bounded by 2^d. □

ECCC ISSN 1433-8092

http://eccc.hpi-web.de