Learning and smoothed analysis

Adam Tauman Kalai
Microsoft Research
New England

Alex Samorodnitsky[*]
The Hebrew University
of Jerusalem

Shang-Hua Teng[†]
Microsoft Research
New England

Abstract— We give a new model of learning motivated by smoothed analysis (Spielman and Teng, 2001). In this model, we analyze two new algorithms, for PAC-learning DNFs and agnostically learning decision trees, from random examples drawn from a constant-bounded product distributions. These two problems had previously been solved using membership queries (Jackson, 1995; Gopalan *et al*, 2005). Our analysis demonstrates that the "heavy" Fourier coefficients of a DNF suffice to recover the DNF. We also show that a structural property of the Fourier spectrum of any boolean function over "typical" product distributions.

In a second model, we consider a simple new distribution over the boolean hypercube, one which is symmetric but is not the uniform distribution, from which we can learn $O(\log n)$-depth decision trees in polynomial time.

1. Introduction

The core machine learning task of efficient binary classification from random training examples was crisply formulated in Valiant's PAC model [15] and follow-up models such as Agnostic learning [9]. Yet polynomial-time PAC and agnostic learning of simple Boolean concepts have defied the best efforts of researchers in computational learning theory, even for simple functions $f : \{-1,1\}^n \to \{0,1\}$ such as Juntas, functions that depend on a few, e.g. $\log\log\log n$, bits (let alone decision trees or DNFs), and even when the input is assumed to be uniform over $\{-1,1\}^n$. Nonetheless, children and small animals are capable of learning concepts, such as classifying images of cats and dogs, that seem much more advanced than DNFs.

In a stronger interactive model, Jackson [6] showed how to learn DNFs over product distributions using *membership queries*, black-box evaluations of the *target function* f at polynomially many arbitrary inputs x, chosen by the algorithm. However, in many real-world situations, one would like to learn from random examples alone.

The basic setup for learning from random examples is as follows. An algorithm is given polynomially many *training examples* $\langle (x^i, f(x^i))\rangle_{i=1}^m$ for some unknown *target* function[1] $f : \{-1,1\}^n \to \{0,1\}$, where the examples x^i are drawn independently from some distribution $\mathcal{D}$ on $\{-1,1\}^n$. The goal is to output a hypothesis $h : \{-1,1\}^n \to \{0,1\}$ with low *error* $\mathrm{err}(h) = \Pr_{x\sim\mathcal{D}}[h(x) \neq f(x)]$ on future examples from the same distribution. Learning is with respect to a *concept class* $\mathcal{C}$ of $g : \{-1,1\}^n \to \{0,1\}$. Define $\mathrm{opt} = \min_{g\in\mathcal{C}} \mathrm{err}(g)$. A polytime algorithm *agnostically [9] learns* $\mathcal{C}$ over $\mathcal{D}$ if for any $f : \{-1,1\}^n \to \{0,1\}$, with high probability over $\mathrm{poly}(1/\epsilon)$ many training examples, it outputs h with $\mathrm{err}(h) \le \mathrm{opt} + \epsilon$. If the algorithm succeeds only under the further assumption that $\mathrm{opt} = 0$ (i.e., assuming $f \in \mathcal{C}$), then it *PAC learns* $\mathcal{C}$ over $\mathcal{D}$ [15].

In the original distribution-free formulation of PAC and agnostic learning, learners must succeed for any distribution $\mathcal{D}$. However, a natural simplifying assumption is that the bits of x are independent. Let us require that the distribution over $x \in \{-1,1\}^n$ is a (constant-bounded) product distribution $\mathcal{D}_\mu$ with parameter $\mu \in [c-1, 1-c]^n$ i.e., the individual bits x_i are independent and $\mu_i = \mathrm{E}_{x\sim\mathcal{D}_\mu}[x_i] \in [c-1, 1-c]$ for some constant $c > 0$.

Since learning theory lacks efficient algorithms that learn interesting classes of functions over product distributions, it is natural to try to relax these assumptions somehow. Using special properties that hold for random decision trees, with high probability, Jackson and Servedio [7] show how to PAC-learn random log-depth decision trees over the uniform distribution. We achieve stronger results regarding arbitrary target functions, by considering nonuniform product distributions.

1.1. Smoothed product distributions

Motivated by *smoothed analysis* [13], we define learning $\mathcal{C}$ with respect to *smoothed product distributions* as follows. Again an arbitrary function $f : \{-1,1\}^n \to \{0,1\}$ is chosen, but a product distribution is chosen whose parameters are specified only up to a proscribed accuracy. Formally, for some constant c, μ is chosen from uniformly at random from a cube of side $2c$, $\mu \in \bar{\mu} + [-c, c]^n$, where $\bar{\mu}$ may be arbitrary. The algorithm must succeed for any $(f, \bar{\mu})$ (in the case of PAC learning, it is further assumed $f \in \mathcal{C}$), with high probability over the chosen μ and polynomially many i.i.d. samples from $\mathcal{D}_\mu$. Section 1.5 provides formal definitions.

Unfortunately, learning with respect to arbitrary (f, μ) requires learning with respect to *adversarial* pairs, as well. Since many real-world learning problems are not actually adversarial, it is arguably reasonable to assume that the parties selecting f and μ are not completely coordinated – they may be correlated but not to high precision.[2] Put another way, for

[*]This work was performed while visiting Microsoft Research New England

[†]Starting in September 2009, the author is at the University of Southern California.

[1]We implicitly assume that multiple occurrences of the same example x will share the same label. However, this is a simplifying assumption and any algorithm which agnostically learns in this model can be generalized to learn from joint distributions over $x \times \{0,1\}$ (see, e.g., [5]).

[2]In fact, it is common in machine learning to assume a friendly coordination between f and $\mathcal{D}$ via "margin" assumptions that state that there is no data near the boundary between positive and negative examples.

any f the set of "hard" distributions $\mathcal{D}_\mu$, or at least those where our algorithms fail, are few and far between in the sense that there cannot be too many of them on the whole or even many concentrated in any small region. We give two polynomial-time algorithms for learning over smoothed product distributions, one that PAC learns DNFs (Theorem 7) and one that agnostically learns decision trees (Theorem 9).

1.2. Overview of the approach

For any product distribution $\mu \in (-1,1)^n$, every function $f : \{-1,1\}^n \to \mathbb{R}$ can be written *uniquely* as,

$$f(x) = \sum_S \hat{f}_\mu(S)\chi_{S,\mu}(x), \text{ where } \chi_{S,\mu}(x) = \prod_{i\in S} \frac{x_i - \mu_i}{\sqrt{1-\mu_i^2}}.$$

With standardized coordinates, $z_i = (x_i - \mu)(1-\mu_i^2)^{-1/2}$ (mean 0 and variance 1), $\hat{f}_\mu(S)$ is simply the coefficient of $\prod_{i\in S} z_i$ in multilinear polynomial $f(x) = \sum_S \hat{f}_\mu(S)\prod_{i\in S} z_i$. An appealing property of this "Fourier" representation is that $\hat{f}_\mu(x) = \mathrm{E}_{x\sim\mathcal{D}_\mu}[f(x)\chi_{S,\mu}(x)]$. The first challenge is finding the important or so-called "heavy" coefficients of the target function, namely the sets S such that $|\hat{f}_\mu(S)|$ is large. This is the standard first step in learning DNFs and decision trees, usually performed by the Kushilevitz-Mansour algorithm [10] that employs membership queries. We analyze a simple feature construction algorithm showing that it will succeed in finding these heavy coefficients (at least on sets $|S| = O(\log n)$), for any bounded f, for most product distributions.

For some types of functions, such as polynomial-sized decision trees, it is known that the coefficients of magnitude $|\hat{f}_\mu(S)| \geq \mathrm{poly}(\epsilon/n)$ and size $|S| < O(\log(n/\epsilon))$ suffice to ϵ-approximate f. However, for more complex functions such as DNFs or agnostically learn decision trees, the heavy coefficients are only *weak learners* and some time of boosting is employed. Unfortunately, boosting is problematic in the smoothed product distribution setting because the first weakly accurate hypothesis h_1 that is learned would depend on μ, and further attempts to generate weakly accurate hypotheses would fail to satisfy the independence between the new target function and distribution.[3]

Instead, we show a new property about PAC learning of DNFs and agnostic learning of decision trees. In particular, the heavy coefficients of a DNF f are enough to recover a good approximation to f directly (without further access to f) and similarly, the heavy coefficients of any Boolean function f suffice to match the error of the most accurate decision tree approximation to f.

Finding heavy coefficients. As a simple example, consider the polynomial $f(x) = \sum_{i\in T} x_i \pmod 2 = \frac{1}{2} - \frac{1}{2}\prod_{i\in S}(-x_i)$, the parity of some unknown set of bits, T. Under the uniform distribution $\mathcal{D}_0$, there is only one nonzero coefficient, $|\hat{f}_0(T)| = \frac{1}{2}$ (aside from the constant coefficient $\hat{f}_0(\varnothing)$). On the other hand, under a nonuniform product distribution, for instance say each $\mu_i \in \{-1/\sqrt{2}, 1/\sqrt{2}\}$, then $|\hat{f}_\mu(S)| = 2^{-|T|/2}$ for each $S \subseteq T$ and $\hat{f}_\mu(S) = 0$ for each $S \nsubseteq T$. By estimating the coefficients of singleton sets $S = \{x_i\}$, it is easy to recover T in polynomial time, for $|T| = O(\log n)$-sized parities.

[3]This is the general case and not pathological, otherwise every DNF could be written as a majority of individual attributes, since boosting produces a majority of weak hypotheses and our analysis shows that there is almost always a weakly correlated bit x_i.

More generally, we show the following structural Fourier property of arbitrary bounded functions under smoothed product distributions. For any $f : \{-1,1\}^n \to [-1,1]$, and any $\bar{\mu} \in (2c-1, 1-2c)^n$, with high probability over uniformly random $\mu \in \bar{\mu} + [-c,c]^n$, for each large coefficient $|\hat{f}_\mu(T)| \geq \beta$, every $S \subseteq T$ is large, $|\hat{f}_\mu(S)| \geq \alpha$, as well. Here $\beta > \alpha$ and both are of order $c^{-O(|T|)}$, see Lemma 3. This gives a simple method of finding all the heavy coefficients: starting with $\mathcal{S} = \{\varnothing\}$, for each $S \in \mathcal{S}$ and $i \notin S$, if $|\hat{f}_\mu(S \cup \{i\})| \geq \alpha$, then add $S \cup \{i\}$ to the collection $\mathcal{S}$. This process repeats until no further sets are added to $\mathcal{S}$.

1.2.1. Learning from the heavy coefficients alone: Let us first give some intuition about why the heavy coefficients information-theoretically suffice, and then roughly describe the efficient learning algorithms. For simplicity, consider the uniform distribution $\hat{f}_0(S) = \hat{f}(S)$ and $\chi_S(x) = \prod_{i\in S} x_i$. Further, suppose we are given explicitly all coefficients whose magnitude is at least ϵ, i.e., we are given $f_{>\epsilon} = \sum_{S:|\hat{f}(S)|>\epsilon} \hat{f}(S)\chi_S(x)$. By Parseval's inequality, there are at most $1/\epsilon^2$ such coefficients and hence $|f_{>\epsilon}(x)| \leq 1/\epsilon$ for any x. Of course, we may not be able to estimate any coefficient exactly, but we can estimate it to arbitrary precision. (The actual property we will use is that the coefficients of the estimate are within ϵ of the true coefficient, since $\|\hat{f} - \hat{f}_{>\epsilon}\|_\infty = \max_S |\hat{f}(S) - \hat{f}_{>\epsilon}(S)| \leq \epsilon$.)

It is well-known that if C is a conjunction, such as $x_1 \wedge \neg x_3 \wedge x_7 = \frac{1+x_1}{2}\frac{1-x_3}{2}\frac{1+x_7}{2}$, then $\|\hat{C}\|_1 = \sum_S |\hat{C}(S)| = 1$ and that if g is a decision tree with t leaves (which can be written as the sum of at most t conjunctions), $\|\hat{g}\|_1 \leq t$. Let $f, g : \{-1,1\}^n \to \{0,1\}$ be any binary functions. A simple but useful observations is that one can approximate $\mathrm{err}(g) = \Pr[f \neq g]$ using g and $f_{>\epsilon}$ alone, without access to f:

$$\begin{aligned}\mathrm{err}(g) &= \mathrm{E}[(1-f)g + (1-g)f] \\ &= \mathrm{E}[f + g - 2fg] \\ &= \mathrm{E}[f_{>\epsilon} + g - 2f_{>\epsilon}g] + \mathrm{E}[f_{\leq\epsilon}(1-2g)],\end{aligned}$$

where $f_{\leq\epsilon} = f - f_{>\epsilon}$. Note that, $|\hat{f}_{\leq\epsilon}(S)| \leq \epsilon$ for any S, $\mathrm{E}[f_{\leq\epsilon}(x)] = \hat{f}_{\leq\epsilon}(\varnothing)$, and,

$$|\mathrm{E}[f_{\leq\epsilon}g]| = \left|\sum_S \hat{f}_{\leq\epsilon}(S)\hat{g}(S)\right| \leq \sum_S \epsilon|\hat{g}(S)| = \epsilon\|\hat{g}\|_1.$$

Since $1+2g$ has L_1 norm $\leq 1+2\|\hat{g}\|_1$, by the triangle inequality $\mathrm{err}(g)$ is within an additive $\epsilon(1+2\|\hat{g}\|_1)$ of $\mathrm{E}[g + f_{>\epsilon} - 2f_{>\epsilon}g]$, a quantity which is possible to compute only from g and $f_{>\epsilon}$.

Lemma 1. *Let f be a t-term DNF. Let $\psi = C_1 \vee C_2 \vee \ldots \vee C_t$ be a DNF which minimizes,*

$$\mathrm{E}[f_{>\epsilon} - \psi] + 2\sum_{i=1}^t \mathrm{E}[(1-f_{>\epsilon})C_i].$$

Then $\mathrm{err}(\psi) \leq 6\epsilon s$.

In the analysis, we will need more refined lemmas due to the fact that we are working over product distributions, we only have estimates of coefficients, and we only can find coefficients of log-degree terms. Most importantly, we need an efficient algorithm as well. But the proof of the above lemma is simple and also sheds light on the algorithm.

Proof: For any candidate DNF $g = \gamma_1(x) \vee \ldots \vee \gamma_t(x)$,

$$\begin{aligned}\mathrm{err}(g) &= \mathrm{E}[(1-f)g + (1-g)f] \\ &= \mathrm{E}[f-g] + 2\,\mathrm{E}[(1-f)g] \\ &\le \mathrm{E}[f-g] + 2\sum_{i=1}^{t}\mathrm{E}[(1-f)\gamma_i],\end{aligned}$$

where in the last step we have used the fact that the false positive rate of g is at most the sum of the false positive rates of the γ_i's. Next, define proxy errors e_1, e_2, by,

$$\begin{aligned}e_1(g) &= \mathrm{E}[f-g] + 2\sum_{i=1}^{t}\mathrm{E}[(1-f)\gamma_i] \\ e_2(g) &= \mathrm{E}[f_{>\epsilon}-g] + 2\sum_{i=1}^{t}\mathrm{E}[(1-f_{>\epsilon})\gamma_i] \\ |e_1(g)-e_2(g)| &= \left|\mathrm{E}[(f-f_{>\epsilon})(1+2\textstyle\sum\gamma_i)]\right| \\ &\le \|f-f_{>\epsilon}\|_\infty \|\widehat{1+2\textstyle\sum\gamma_i}\|_1 \le \epsilon(1+2s)\end{aligned}$$

In the above we have used the fact that $u \cdot v \le \|u\|_\infty \|v\|_1$ and that $\|\gamma_i\|_1 = 1$. We also note that $\mathrm{err}(g) \le e_1(g)$, $e_1(f) = 0$, and $|e_1(g) - e_2(g)| \le \epsilon(1+2s)$ for all s-term DNFs. These together imply that $\mathrm{err}(\psi) \le e_1(\psi) \le 2\epsilon(1+2s) \le 6\epsilon s$. ■

A similar statement shows that, for any Boolean function f, the best decision tree can be approximated from its heavy Fourier coefficients.

Efficient approximation from heavy coefficients. Gopalan *et al* apply a gradient projection method for optimization over functions with low L_1 norm, a relaxation of decision trees and conjunctions. Such functions can always be approximated by sparse polynomials and hence succinctly represented. We employ the same approach here. For learning DNFs, we combine the optimization with the "reliable" DNF learning approach of Kalai *et al* [8]. The idea is to do a relaxation to a convex set of functions. Consider the set of functions,

$$\mathcal{G} = \left\{g : \{-1,1\}^n \to [0,1] \mid \|\hat{g}\|_1 \le t\right\}.$$

Now, in the case of decision trees, the goal will be to minimize, $\mathrm{E}[f_{>\epsilon} + g - 2f_{>\epsilon}g]$ over $\mathcal{G}$. The key properties of such an optimization problem are (1) the objective function is a convex function of g (in fact it is linear), (2) the set $\mathcal{G}$ is a convex set, and (3) (approximate) membership in $\mathcal{G}$ can be determined efficiently. This last point is somewhat subtle. Given an explicit sparse polynomial represented by its list of nonzero coefficients, it is easy to check if $\sum_S |\hat{g}(S)| \le t$. It is more difficult to check that g is bounded in $[0,1]$. However, for learning, it suffices that g is nearly bounded which can be verified in polynomial time. In analogy with the fact that convex functions can often be efficiently minimized over convex sets, the convex objective function (of functions) can be approximately minimized over something approximating $\mathcal{G}$. Interestingly, the reliable approach to learning DNF resembles recent work in complexity theory on fooling DNF [2].

1.3. Part II: Learning from diversity

Many distributions have dependencies among the bits resulting from an underlying "diversity" in a population. For example, consider a medical problem such as predicting whether someone will get diabetes from an attribute vector, including, say, age, height, and weight. It is clear that an individual's attributes will be correlated – children tend to be younger, shorter and lighter than adults. As a second example, consider classifying email as SPAM or not based on a $\{0,1\}^n$ vector which indicates the absence or presence of n different words in an email. There is a large variance in email length and the number of distinct words in an email. On the other hand, if the data were coming from the uniform distribution, most examples would have a $1/2 \pm n^{-1/2}$ fraction of 1's. Hence, in many situations there is an underlying diversity in the population which may be quantified by a single parameter, e.g., age or size, and this diversity leads to dependencies between attributes.

As a simplified model of this phenomenon, consider the following distribution ρ_c on $x \in \{0,1\}^n$, for any constant $c \in (0, 1/2]$.

$$\rho_c(x) = \frac{1}{2c}\int_{\frac{1}{2}-c}^{\frac{1}{2}+c} p^{|x|}(1-p)^{n-|x|}dp, \quad |x| = \sum x_i.$$

To generate an example from this type of distribution, first a $p \in [1/2 - c, 1/2 + c]$ is chosen uniformly at random. Then an example $x \in \{0,1\}^n$ is chosen from the *p-biased product distribution* ν_p (the product distribution in which $\mathrm{E}_{x\sim\nu_p}[x_i] = p$ for each i). We give an algorithm that PAC-learns depth-$O(\log n)$ trees over ρ_c. Interestingly, the distribution $\rho_{1/2}$, a distribution with many appealing mathematical properties, has recently been used to simplify the proof of the density Hales-Jewett Theorem [11].

The distribution ρ_c is not completely realistic, but it captures one aspect of real (nonuniform) distributions. We start with this simple distribution, but extensions to other related distributions (e.g., not centered around bias 1/2) are likely possible. The main result here is the following.

Theorem 2. *Fix any constant $c > 0$. Then there is a polynomial M such that, for any $\delta \in (0,1)$, $n, d \ge 1$, and any depth-d decision tree f, for $m \ge M(2^d n \log 1/\delta)$ examples $(x^i, f(x^i))$ where each x^i is chosen independently from ρ_c, with probability $\ge 1 - \delta$, the algorithm described in Section 2.4 outputs a polynomial exactly equivalent to f and runs in time poly(m).*

The first step in our algorithm is to reduce this model to a related model suggested earlier and independently by Arpe and Mossel [1], in which it is assumed that one has access to k different example oracles representing samples from different p-biased distributions. If one reinterprets their results in our setting, then in polynomial time one can learn $k = O\left(\frac{\log n}{\log\log n}\right)$-Juntas, i.e., arbitrary functions that depend on only k relevant bits. Note that $O(\log n)$-depth decision trees include $O(\log n)$-Juntas as a special case. More generally, our algorithm learns sparse, low degree integer polynomials.

1.4. Organization

We first focus on learning from smoothed product distributions. Section 1.5 gives preliminaries for this problem. Section 1.7 gives an algorithm for finding the "heavy" Fourier coefficients in the smoothed product distribution model. Section 1.8 gives an algorithm for approximating a DNF from its heavy coefficients. Section 1.9 gives an algorithm for approximating any function as well as the best decision tree, i.e., agnostically learning decision trees, from its heavy Fourier coefficients. Note that these latter two sections are not specific to any smoothed analysis – they simply show how to learn from heavy coefficients alone. For example, it could be used to replace boosting in Jackson's DNF learning algorithm (though our algorithm is not simpler).

Section 2 discusses the model of learning from diversity and is self-contained.

1.5. Preliminaries

Let $N = \{1, 2, \ldots, n\}$. We consider examples (x, y) with $x \in \{-1, 1\}^n$ and $y \in \{0, 1\}$. A product distribution $\mathcal{D}_\mu$ over $\{-1, 1\}^n$ is parameterized by its mean vector $\mu \in [-1, 1]^n$, where $\mu_i = \mathrm{E}_{x \sim \mathcal{D}_\mu}[x_i]$ and the bits are independent. The uniform distribution is $\mathcal{D}_0$. We say $\mathcal{D}_\mu$ is c-bounded if $\mu_i \in [c-1, 1-c]$ for all i.

We denote $\Pr_{x \sim \mathcal{D}_\mu}$ by $\Pr_\mu$ and $\mathrm{E}_{x \sim \mathcal{D}_\mu}$ by E_μ for brevity. Let $\chi_{S,\mu}(x) = \prod_{i \in s}(x_i - \mu_i)/\sqrt{1 - \mu_i^2}$. This normalization gives $\mathrm{E}_\mu[\chi_{\{i\},\mu}(x)] = 0$ and $\mathrm{E}[\chi^2_{\{i\},\mu}(x)] = 1$, and hence by independence $\mathrm{E}[\chi_{S,\mu}(x)] = 0$ and $\mathrm{E}[\chi^2_{S,\mu}(x)] = 1$ for $S \neq \emptyset$. When μ is understood from context, we write $\chi_S(x)$.

Define the inner product $\langle f, g \rangle_\mu = \mathrm{E}_\mu[f(x)g(x)]$. By independence $\langle \chi_{S,\mu}, \chi_{T,\mu} \rangle_\mu = 0$ for $S \neq T$ and $\langle \chi_{S,\mu}, \chi_{S,\mu} \rangle_\mu = 1$. Hence, the 2^n different χ_S's form an orthonormal basis for the set of real-valued functions on $\{-1, 1\}^n$ with respect to $\langle\rangle_\mu$. We define the Fourier coefficient (relative to μ), for any $S \subseteq N$,

$$\hat{f}_\mu(S) = \mathrm{E}_\mu[f(x)\chi_{S,\mu}(x)]. \qquad (1)$$

Also observe that $\hat{f}_0(S)$ is the standard Fourier coefficient over the uniform distribution, and that, for any $\mu \in [-1, 1]^n$,

$$f(x) = \sum_{S \subseteq N} \hat{f}_\mu(S)\chi_{S,\mu}(x).$$

When μ is understood from context we write simply $\hat{f} = \hat{f}_\mu$.

Henceforth we write $\sum_S$ to denote $\sum_{S \subseteq N}$ and $\sum_{|S|=d}$ to denote the sum over $S \subseteq N$ such that $|S| = d$. Similarly for $\sum_{|S|>d}$, and so forth. It can be shown that $\langle f, g \rangle_\mu = \sum_{S \subseteq N} \hat{f}(S)\hat{g}_\mu(S)$, and Parseval's equality,

$$\langle f, f \rangle_\mu = \sum_{S \subseteq N} \hat{f}^2_\mu(S) = \mathrm{E}_\mu[f^2(x)].$$

This implies that for any $f : \{-1, 1\}^n \to [-1, 1]$, $\sum_S \hat{f}^2_\mu(S) \leq 1$. It is also useful for bounding $\mathrm{E}_\mu[(f(x) - g(x))^2] = \sum_S(\hat{f}(S) - \hat{g}_\mu(S))^2$.

It will also be helpful to think of $\hat{f} \in \mathbb{R}^{2^n}$ as a vector in 2^n-dimensional Euclidean space, and we will use the following quantities: $\|\hat{f}\|_2 = \sqrt{\sum_S \hat{f}^2(S)}$, $\|\hat{f}\|_1 = \sum_S |\hat{f}(S)|$, $\|\hat{f}\|_\infty = \max_S |\hat{f}(S)|$, and $\|\hat{f}\|_0 = |\{S \mid \hat{f}(S) \neq 0\}|$.

Fix any constant $c \in (0, 1/2)$. We assume we have some fixed $2c$-bounded product distribution $\bar{\mu} \in [2c-1, 1-2c]^n$ and that a *perturbation* $\Delta \in [-c, c]^n$ is chosen uniformly at random and the resulting product distribution has $\mu = \bar{\mu} + \Delta$. Note that $\mathcal{D}_\mu$ is c-bounded.

A *disjunctive normal form* (DNF) formula is an OR of ANDs, e.g., $f(x) = (x_1 \wedge \neg x_3) \vee (x_2 \wedge x_3 \wedge x_10) \vee x_4$. The negation of a DNF is a *conjunctive normal form* (CNF) formula, e.g.,$(\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_10) \wedge \neg x_4$. For the definition of a binary decision trees, see, e.g., [10]. The size of a decision tree is defined to be the number of leaves.

1.6. Smoothed product distributions: Fourier structure

The following lemmas show that, with high probability, for every coefficient $\hat{f}_\mu(S)$ that is sufficiently large, say $|\hat{f}(S)| > \beta$, it is very likely that all subterms $T \subseteq S$ have $|\hat{f}(T)| > \alpha$, for some $\alpha < \beta$. In other words, with high probability, all sub-coefficients of large $\hat{f}(S)$ will be pretty large.

Lemma 3. *Let $f : \{-1, 1\}^n \to [-1, 1]$. Let $\alpha, \beta \geq 0$, $d \in \mathbb{N}$. Let $c \in (0, 1/2)$, $\bar{\mu} \in [2c-1, 1-2c]^n$, and $\mu = \bar{\mu} + \Delta$ where $\Delta \in [-c, c]^n$ is chosen uniformly at random. Then, with probability at most $\alpha^{1/2}\beta^{-5/2}(2/c)^{2d}$, there exists $T \subseteq U \subseteq N$ such that $|U| \leq d \wedge |\hat{f}_\mu(T)| \leq \alpha \wedge |\hat{f}_\mu(U)| \geq \beta$.*

The proof of this lemma is omitted due to space constraints. In order to prove it, we give a continuous variant of Schwartz-Zippel lemma. This lemma states that a nonzero degree-d multilinear function cannot be too close to 0 (or any other value) too often over $x \in [-1, 1]^n$. In particular, this is a *non*concentration bound saying that a nonzero multilinear polynomial cannot be concentrated near 0 (or it's mean or any real value).

Lemma 4. *Let $g : \mathbb{R}^n \to \mathbb{R}$ be a degree-d multilinear polynomial, $g(x) = \sum_{|S| \leq d} \hat{g}(S) \prod_{i \in S} x_i$. Suppose that there exists $S \subseteq N$ with $|S| = d$ and $|\hat{g}(S)| \geq 1$. Then for a uniformly chosen random $x \in [-1, 1]^n$, and for any $\epsilon > 0$,*

$$\Pr_{x \in [-1,1]^n} [\ |g(x)| \leq \epsilon\ ] \leq 2^d\sqrt{\epsilon}.$$

Proof: WLOG let say $\hat{g}(D) = 1$ for $D = \{1, 2, \ldots, d\}$ for we can always permute the terms and rescale the polynomial so that this coefficient is exactly 1. We first establish that,

$$\Pr_{x \in [-1,1]^n}[|g(x)| \leq \epsilon] \leq \Pr_{x \in [-1,1]^n}\left[\left|\prod_{i \in D} x_i\right| \leq \epsilon\right]. \qquad (2)$$

In other words, the worst case is a monomial. To see this, write,

$$g(x) = x_1 g_1(x_2, x_3, \ldots, x_n) + g_2(x_2, x_3, \ldots, x_n).$$

Now, by independence imagine picking x by first picking $x_2, x_3, \ldots, x_n$ (later we will pick x_1). Let $\gamma_i = g_i(x_2, \ldots, x_n)$ for $i = 1, 2$. Then, consider the two sets $I_1 = \{x_1 \in \mathbb{R} : |x_1\gamma_1 + \gamma_2| \leq \epsilon\}$ and $I_2 = \{x_1 \in \mathbb{R} : |x_1\gamma_1| \leq \epsilon\}$. These are both intervals, and they are of equal width. However, I_2 is centered at the origin. Hence, since x_1 is chosen uniformly from $[-1, 1]$, we have that for any fixed γ_1, γ_2, $\Pr_{x_1 \in [-1,1]}[x_1 \in I_1] \leq \Pr_{x_1 \in [-1,1]}[x_1 \in I_2]$, because $I_2 \cap [-1, 1]$ is at least as wide as $I_1 \cap [-1, 1]$. Hence it suffices to prove the lemma for those functions where $\hat{g}(S) = 0$ for all S for which $1 \notin S$. (In fact, this is the worst case.) By symmetry, it suffices to prove the lemma for those functions where $\hat{g}(S) = 0$ for all S for which $i \notin S$, for $i = 1, 2, \ldots, d$. After removing all terms S that do not contain D we are left with the function x_D, establishing (2). Now, for a loose bound, one can use Markov's inequality:[4]

$$\Pr[|x_D| \leq \epsilon] = \Pr\left[|x_D|^{-\frac{1}{2}} \geq \epsilon^{-\frac{1}{2}}\right] \leq \frac{\mathrm{E}[|\prod_D x_i|^{-\frac{1}{2}}]}{\epsilon^{-\frac{1}{2}}} = \epsilon^{\frac{1}{2}} 2^d.$$

[4]A tight bound, $\Pr[|x_1 \ldots x_d| \leq \epsilon] = \epsilon \sum_{i=0}^{d-1} \log^i \frac{1}{\epsilon}$, follows from $\Pr[|x_1 x_2 \ldots x_{i+1}| \leq \epsilon] = \int_0^1 \Pr[|x_1 x_2 \ldots x_i| \leq \frac{\epsilon}{t}]dt$ and induction.

In the last step, $\mathrm{E}[|\prod_D x_i|^{-\frac{1}{2}}] = \mathrm{E}[|x_1|^{-\frac{1}{2}}]^d$ by independence and symmetry, and a simple calculation based on the fact that $|x_1|$ is uniform from $[0,1]$ gives $\mathrm{E}[|x_1|^{-\frac{1}{2}}] = 2$. ■

An interesting property of this bound is that it does not hold for inputs chosen over the discrete hypercube $\{-1,1\}^n$. For example, the function $f(x) = 1 + x_1$ is 0 on half of the discrete hypercube but 0 on a measure-0 fraction of the solid cube. This lemma is also a bit stronger than what holds for (non-multilinear) polynomials [3], [4] – here one can see that the polynomial x_1^d is too concentrated for our purposes.

1.7. Finding the heavy coefficients

For simplicity, we suppose that the algorithms have exact knowledge of μ. In general, these parameters can be estimated to any desired inverse-polynomial accuracy in polynomial time. The algorithm is below.

```
Algorithm **Greedy feature construction**.
Inputs: $(x^1,y^1),(x^2,y^2),\dots,(x^m,y^m) \in \mathbb{R}^n \times \{-1,1\}$, degree $d \ge 1$, and $\mu \in (-1,1)^n$.
1. Let $\mathcal{S}_0 := \{\varnothing\}$.
2. For $k = 1,2,\dots,d$ :
  1) Let
     $$\mathcal{S}_k := \mathcal{S}_{k-1} \cup \left\{ S \cup \{i\} \;\middle|\; S \in \mathcal{S}_{k-1} \wedge \wedge \left| \frac{1}{m}\sum_{j=1}^m y^j \chi_{S\cup\{i\},\mu}(x^j) \right| \ge m^{-1/3} \right\}.$$
  2) If $|\mathcal{S}_k| > m$ then abort and output FAIL.
3. Output the following polynomial $p : \{-1,1\}^n \to \mathbb{R}$,
   $$p(x) = \sum_{S \subseteq \mathcal{S}_n} \left( \frac{1}{m}\sum_{j=1}^m y^j \chi_{S\cup\{i\},\mu}(x^j) \right) \chi_{S,\mu}(x).$$
```

A "heavy" coefficient is simply one with large magnitude $|\hat{f}(S)|$. A "large" set is one for which $|S|$ is large, and a small set has $|S|$ small. We now claim (proof omitted) that the GREEDY FEATURE CONSTRUCTION (GFC) algorithm finds all heavy coefficients on small sets S.

Lemma 5. *For any constant $c > 0$, there exists a univariate polynomial u, such that for any $\epsilon, \delta > 0$, $n, d \ge 1$, $\bar{\mu} \in [2c-1, 1-2c]$, and any $f : \{-1,1\}^n \to [-1,1]$, the GFC algorithm run with $m = u(\log(n)2^d/\epsilon\delta)$ samples, with probability $\ge 1-\delta$, outputs degree-d polynomial $p(x)$ with $|\hat{p}_\mu(S) - \hat{f}_\mu(S)| \le \epsilon$ for each S with $|S| \le d$, and such that $\hat{p}_\mu(S) = 0$ for each S with $|\hat{f}_\mu(S)| \le \epsilon/2$. GFC is a polynomial-time algorithm.*

1.8. Learning CNF from heavy coefficients

In this section, fix a constant-bounded product distribution $\mu \in [c-1, 1-c]^n$. It will be slightly easier to describe the algorithm in terms of learning CNFs, $f(x) = D_1(x) \wedge \dots \wedge D_t(x)$, where each $D_i(x)$ is a disjunction, e.g., $x_3 \vee \neg x_7$. Since the negation of a DNF is a CNF of the same size, learning CNFs and learning DNFs are equivalent problems. The algorithm for learning CNF from heavy coefficients is given below.

We define a penalty function for being outside of the range $[0,1]$, $\Phi : \mathbb{R} \to \mathbb{R}$,

$$\Phi(x) = \begin{cases} x-1 & \text{if } x > 1 \\ 0 & \text{if } x \in [0,1] \\ -x & \text{if } x < 0 \end{cases} \quad \phi(x) = \begin{cases} 1 & \text{if } x > 1 \\ 0 & \text{if } x \in [0,1] \\ -1 & \text{if } x < 0 \end{cases}.$$

It will be helpful to try to find a function $h : \{-1,1\}^n \to [0,1]$, and this penalty will be useful in approximately achieving this goal. Note that Φ is convex. It will also be helpful to consider the ϕ. While Φ is not differentiable, it is easy to verify that $\phi(x) \in \nabla\Phi(x)$ is a *subgradient* of Φ, which formally means,

$$\Phi(x) - \Phi(x_0) \ge \phi(x_0)(x - x_0), \tag{3}$$

for any $x_0, x \in \mathbb{R}$.

Let *target* function $f(x) = D_1(x)D_2(x)\dots D_s(x)$, where s is known to the algorithm[5] and each $D_i(x) \in \{0,1\}$ is a disjunction, e.g., $(x_3 \vee \neg x_7 \vee x_9)$. Our goal is to find a function $h : \{-1,1\}^n \to \{0,1\}$ such that $\Pr_\mu[h(x) \ne f(x)] \le \epsilon$.

For this algorithm, we assume that we begin with an approximation of the heavy coefficients of f. In particular, we suppose that we start with a polynomial p such that $\max_{S:|S|\le d} |\hat{p}(S) - \hat{f}(S)|_\infty \le \tau$ and such that $\|\hat{p}\|_0 \le 8/\tau^2$, which the previous section explains how to find. It turns out that this will be enough and we will not need direct access to f, however one must consider f for the purposes of analysis.

Define $K_d = \{g : \{-1,1\} \to \mathbb{R} \mid \deg(g) \le d \text{ and } \|\hat{g}\|_1 \le 1\}$. In Section 1.11, we give the projection algorithm which computes $\mathrm{proj}_{\mu,K_d}(g) = \arg\min_{h \in K_d} \|\hat{h}_\mu - \hat{g}_\mu\|^2$.

```
Algorithm **CNF Appx**.
Input: $n, d, T, R, \Lambda_1, \Lambda_2 \ge 1, \eta, \tau, G > 0$, $\mu \in (-1,1)^n$, black-box access to polynomial $p : \{-1,1\}^n \to \mathbb{R}$.

For $i = 1,2,\dots,R$:
1) Let $H_i = h_1 h_2 \cdots h_{i-1}$   $(H_1 = 1)$
2) Let $g_i^1 = 0$.
3) For $j = 1,2,\dots,T$:
   $$g_i^{j+1} = \mathrm{proj}_{\mu,K_d}\Big(\mathrm{EKM}_\mu\big(g_i^j - \eta(H_i - \Lambda_1 p + \Lambda_2\phi(g_i^j)), 1 + \eta G, \tau, \delta/(RT)\big)\Big).$$
4) Let function $h_i$ be $h_i(x) = \mathrm{I}[\frac{1}{T}\sum_{j=1}^T g_i^j(x) \ge \frac{1}{2}]$.
Output hypothesis $h(x) = h_1 h_2 \cdots h_R$.
```

Theorem 6. *Let $c \in (0,1)$ be a constant. Let $\mu \in [c-1, 1-c]^n$. Let $f : \{-1,1\}^n \to \{0,1\}$ compute an s-term DNF. Let $\epsilon, \delta, B > 0$. Take $R = 6s/\epsilon$, $\Lambda_1 = 36R/\epsilon$, $\Lambda_2 = 40\Lambda_1^2 R/\epsilon$, $d = \log(20s/\epsilon)/c$, $\epsilon_0 = \epsilon/(20s\Lambda_1) = \epsilon^3/(4320s^2)$, $G = 1 + \Lambda_1 B + \Lambda_2$, $\tau = (\epsilon_0\Lambda_1/16)^2$, $T = (4G\epsilon_0\Lambda_1)^2$, and $\eta = \sqrt{T}/G$. Let $p : \{-1,1\}^n \to [-B,B]$ be such that $|\hat{f}_\mu(S) - \hat{p}_\mu(S)| \le \epsilon_0$ for all sets of size $|S| \le d$ and $\hat{p}(S) = 0$ for $|S| > d$. Then with probability $\ge 1-\delta$, the CNF Appx algorithm outputs h with $\Pr_\mu[h(x) \ne f(x)] \le \epsilon$. The runtime of the algorithm is polynomial in $nB\log(1/\delta)/\epsilon$ times the amount of time to evaluate p.*

[5]A standard "doubling trick" can be applied to generalize to the case when s is not known.

The proof of this theorem is omitted, but, using it, we are now able to analyze our DNF learning algorithm.

Theorem 7. *For any constant $c > 0$, there is a univariate polynomial u such that, for any DNF $f : \{-1,1\}^n \to \{0,1\}$ of size s terms, any $\epsilon, \delta > 0$, and any $\bar{\mu} \in [2c-1, 1-2c]^n$, there is an algorithm that takes at most $u(ns/(\epsilon\delta))$ examples from $\mathcal{D}_\mu$ with uniformly random $\mu \in \bar{\mu} + [-c, c]^n$, runs in time $u(ns/(\epsilon\delta))$, and, with probability $\geq 1-\delta$, outputs a hypothesis h with $\Pr_\mu[h(x) \neq f(x)] \leq \epsilon$. The probability here is taken over the random choice of μ and m i.i.d. samples from product distribution $\mathcal{D}_\mu$.*

Proof: We describe an algorithm for learning a CNF. The reduction is trivial – replace f and h with $1-f$ and $1-h$, respectively.

Let $\epsilon_0 = \epsilon^3/(4320s^2)$, $\delta_0 = \delta/2$. The algorithm first calls the Greedy Feature Construction algorithm with degree $d = \log(20s/\epsilon)/c$ and $m = \text{poly}(\log(n)2^d/(\epsilon_0\delta_0))$, so that, with probability $\geq 1-\delta_0$, we get an estimate p such that $|\hat{p}_\mu(S) - \hat{f}_\mu(S)| \leq \epsilon_0$ for each S with $|S| \leq d$, and such that $\hat{p}(S) = 0$ for each S with $|\hat{f}(S)| \leq \epsilon_0/2$. By Parseval, there can be at most $4/\epsilon_0^2$ different coefficients of magnitude greater than $|\hat{f}(S)| > \epsilon/2$. For each of these $|\hat{p}(S)| \leq 1 + \epsilon_0$. Hence,

$$|p(x)| \leq \sum_{|S| \leq d} |\hat{p}_\mu(S)| \cdot |\chi_{\mu,S}(x)| \leq \frac{4}{\epsilon_0^2}(1+\epsilon_0)\left(\frac{2}{c}\right)^d.$$

In the above, we have used $|\chi_{S,\mu}(x)| \leq (2/c)^{|S|}$, which follows from the fact that $|\chi_{\{i\},\mu}(x)| \leq \frac{2-c}{\sqrt{1-(1-c)^2}} \leq 2/c$ for any $i \in N$, and $x \in \{-1,1\}^n$, by the definition of χ. Let $B = \frac{4}{\epsilon_0^2}(1+\epsilon_0)\left(\frac{2}{c}\right)^d = \text{poly}(s/\epsilon)$. Next we run the CNF Appx algorithm on p with the parameters ϵ, δ_0, B and those given in Theorem 6. With probability $\geq 1-\delta/2$, this will succeed in outputting a hypothesis with error at most ϵ. Both the Greedy Feature Construction and the CNF Appx algorithms run in polynomial time. ■

1.9. Agnostically learning decision trees from heavy coefficients

At this point, it will be helpful to define K_{dt},

$$K_{dt} = \left\{g : \{-1,1\} \to \mathbb{R} \mid \deg(g) \leq d \text{ and } \|\hat{g}\|_1 \leq t\right\}.$$

Note that $K_d = K_{d1}$, for our earlier definition of K_d.

```
Algorithm **DT Appx**.
Input: $n, d, t, T, \Lambda \geq 1, \eta, \tau, G > 0$, $\mu \in (-1,1)^n$, black-box access to polynomial $p : \{-1,1\}^n \to \mathbb{R}$.

1) Let $g^1 = 0$.
2) For $j = 1, 2, \ldots, T$:
    $g^{j+1} = \text{proj}_{\mu, K_{dt}}\Big(\text{EKM}_\mu\big(g^j - \eta(\Lambda\phi(g_i^j)) - p, 1 + \eta G, \tau, \delta/T\big)\Big)$.
3) Let $g = \frac{1}{T}\sum_{j=1}^T g^j$.
4) Draw $m$ samples $x^1, x^2, \ldots, x^m$ from $\mathcal{D}_\mu$.
5) Choose $\theta \in [0,1]$ so as to minimize $\sum_{i=1}^m \big(\mathrm{I}[g(x^i) \geq \theta](1 - p(x^i)) + \mathrm{I}[g(x^i) < \theta]p(x^i)\big)$.
6) Output hypothesis $h(x) = \mathrm{I}[g(x) \geq \theta]$.
```

Theorem 8. *Let $c \in (0,1)$ be a constant. Let $s, n \geq 1, \epsilon, \delta, B > 0$, and $\mu \in [c-1, 1-c]^n$. Let $f : \{-1,1\}^n \to \{0,1\}$ be a binary function. Take $d = \frac{2}{c}\log\frac{8s}{\epsilon}$, $t = 4^d$, $\Lambda = \frac{33}{\epsilon}$, $G = 1+2B+\Lambda$, $\eta = G^{-1}T^{-1/2}$, $\epsilon_0 = \frac{\epsilon}{60t}$, $T = \frac{16G^2}{\epsilon_0^2}$, $\tau = \frac{\epsilon_0^2}{256t}$, and $m = \frac{8}{\epsilon^3}\log^2\frac{1}{\delta}$. Let $p : \{-1,1\}^n \to [-B, B]$ be such that $|\hat{f}_\mu(S) - \hat{p}_\mu(S)| \leq \epsilon_0$ for all sets of size $|S| \leq d$ and $\hat{p}(S) = 0$ for $|S| > d$. Then with probability $\geq 1-\delta$, the CNF Appx algorithm outputs h with* $\text{err}(h) \leq \text{opt} + \epsilon$*. The runtime of the algorithm is polynomial in $nB\log(1/\delta)/\epsilon$ times the amount of time to evaluate p.*

The proof of this theorem is omitted due to space limitations. However, using it, we are now able to analyze our agnostic decision tree learning algorithm.

Theorem 9. *For any constant $c > 0$, there is a univariate polynomial u such that, for any $f : \{-1,1\}^n \to \{0,1\}$ and any $s \geq 1, \epsilon, \delta > 0$, and any $\bar{\mu} \in [2c-1, 1-2c]^n$, there is an algorithm that takes at most $u(ns/(\epsilon\delta))$ examples from $\mathcal{D}_\mu$ with uniformly random $\mu \in \bar{\mu} + [-c, c]^n$, runs in time $u(ns/(\epsilon\delta))$, and, with probability $\geq 1-\delta$, outputs a hypothesis h with $\text{err}(h) \leq \text{opt} + \epsilon$. The probability here is taken over the random choice of μ and m i.i.d. samples from product distribution $\mathcal{D}_\mu$.*

Proof: Let $\epsilon_0 = \frac{\epsilon}{60t}$, $\delta_0 = \delta/2$. The algorithm first calls the Greedy Feature Construction algorithm with degree $d = \frac{2}{c}\log\frac{8s}{\epsilon}$ and $m = \text{poly}(\log(n)2^d/(\epsilon_0\delta_0))$, so that, with probability $\geq 1-\delta_0$, we get an estimate p such that $|\hat{p}_\mu(S) - \hat{f}_\mu(S)| \leq \epsilon_0$ for each S with $|S| \leq d$, and such that $\hat{p}(S) = 0$ for each S with $|\hat{f}(S)| \leq \epsilon_0/2$. Exactly as in the proof of Theorem 7, $|p(x)| \leq \frac{4}{\epsilon_0^2}(1+\epsilon_0)\left(\frac{2}{c}\right)^d$. Let $B = \frac{4}{\epsilon_0^2}(1+\epsilon_0)\left(\frac{2}{c}\right)^d = \text{poly}(s/\epsilon)$. Next we run the DT Appx algorithm on p with the parameters ϵ, δ_0, B and those given in Theorem 6. With probability $\geq 1-\delta/2$, this will succeed in outputting a hypothesis with error at most ϵ. Both the Greedy Feature Construction and the CNF Appx algorithms run in polynomial time. ■

1.10. Fourier gradient descent

Both our DNF and agnostic decision tree learners can be viewed in a common framework as a general Fourier "gradient descent" algorithm of a convex *loss* function $\mathcal{L}(f)$ over an arbitrary fixed product distribution $\mathcal{D}_\mu$, which is a generalization of the algorithm of Gopalan *et al* [5]. Let $\mathbb{R}^{\{-1,1\}^n}$ denote the set of functions from $\{-1,1\}^n$ to $\mathbb{R}$. Again note that $K_d = K_{d1}$, for our earlier definition of K_d. Note that $0 \in K_{dt}$ and $\|\hat{f}\|_2 \leq \|\hat{f}_\mu\|_1 \leq t$ for each $f \in K_{dt}$. We also suppose that the product distribution parameters μ have been fixed.

Let $\mathcal{L} : \mathbb{R}^{\{-1,1\}^n} \to \mathbb{R}$ denote a convex *loss* function, meaning that for any $\lambda \in [0,1]$ and $g, h : \{-1,1\}^n \to \mathbb{R}$, $\mathcal{L}(\lambda g + (1-\lambda)h) \geq \lambda\mathcal{L}(g) + (1-\lambda)\mathcal{L}(h)$. The goal is to (approximately) minimize the loss over K_{dt}, $\min_{f \in K_{dt}} \mathcal{L}(f)$. Since we do not assume that $\mathcal{L}$ is differentiable, we consider a *subgradient* descent type of algorithm. We suppose we have access to two things. First, we assume we have black-box access to a bounded "sugradient" function $\Gamma : \mathbb{R}^{\{-1,1\}^n} \times \{0,1\}^n \to [-G, G]$, for some $G \geq 0$. By subgradient, we mean:

$$\forall f,g:\{-1,1\}^n \to \mathbb{R} \quad \mathcal{L}(g) \geq \mathcal{L}(f) + \mathrm{E}_\mu\big[\Gamma(f,x)(g(x)-f(x))\big]. \tag{4}$$

This is similar to the gradient bound for convex differentiable u on Euclidean space, where $u(x') \geq u(x) + \nabla u(x)\cdot(x'-x)$. Let $\Gamma_f(x) = \Gamma(f,x)$. This connection can be made precise when one considers $\hat{f} \in \mathbb{R}^{2^n}$ as a vector in Euclidean space and Γ_f as the gradient of $\mathcal{L}(\hat{f})$. More generally, $\mathcal{L}$ may not be differentiable and any *subgradient* (tangent plane lying below $\mathcal{L}$) will do.

Second, we assume we have access to a projection oracle, which when given a function f, finds the closest $g \in K_{dt}$ to f,

$$\mathrm{proj}_{\mu,K_{dt}}(f) = \arg\min_{g\in K_{dt}} \|\hat{g} - \hat{f}\|_2,$$

which returns the closest function in K_{dt} to f. The projection routine is described in Section 1.11. It is probably easiest to first understand the algorithm at its conceptual level, ignoring runtime and efficient representation. One may even think of the functions being represented by their 2^n different Fourier coefficients. However, we will shortly describe how to implement it efficiently.

The gradient projection method [12] (sometimes called the *projected subgradient method*) in this context, chooses a sequence of functions, starting with an arbitrary $f^1 \in K_{dt}$ and then taking $f^{(i+1)} = \mathrm{proj}_{\mu,K_{dt}}(f^i - \eta\Gamma_{f^i})$, where $\eta > 0$ is a *step size*. However, in order to be efficient, we will need an explicit sparse representation of f^i and Γ_{f^i}. In particular, the f^i's are represented by a list of nonzero Fourier coefficients. As we will see, the projection operation never increases the number of nonzero coefficients, i.e., $\|\mathrm{proj}_{\mu,K_{dt}}(f)\|_0 \leq \|\hat{f}\|_0$. The projection operation is described in Section 1.11. Finally, in order to represent Γ_{f^i} succinctly, we will use an extension of the Kushilev-Mansour routine for extracting heavy coefficients of a function. The extension, omitted due to space limitations, handles product distributions.

```
Algorithm **Fourier gradient descent**.
Inputs: $T \geq 1, \epsilon, \delta, \eta, G > 0$, black-box $\Gamma : \mathbb{R}^{\{-1,1\}^n} \to [-G,G]$, black box $\mathrm{proj}_K : \mathbb{R}^{\{-1,1\}^n} \to K$. Output: $h \in K$.
  1) Let $f^1 = 0$
  2) For $i = 1,2,\ldots,T$ :
         $f^{i+1} = \mathrm{proj}_{\mu,K_{dt}}\left(\mathrm{EKM}_\mu(f^i - \eta\Gamma_{f^i}, t+\eta G, \epsilon, \delta)\right)$
  3) Output $h = \frac{1}{T}\sum_{i=1}^T f^i$
```

Lemma 10. *Let $\mu \in [-1,1]^n, \delta, G, t \geq 0, T \geq 1$. Let loss $\mathcal{L} : \mathbb{R}^{\{-1,1\}^n} \to \mathbb{R}$ and subgradient $\Gamma : K_{dt} \to [-G,G]$ satisfy (4). Take $\eta = G^{-1}T^{-1/2}$. Then, with probability $\geq 1 - T\delta$, the Fourier gradient descent algorithm outputs $h \in K$ with*

$$\mathcal{L}(h) \leq \min_{f\in K_{dt}} \mathcal{L}(f) + 2\frac{tG}{\sqrt{T}} + 8\epsilon^{\frac{1}{2}}t^{\frac{3}{2}}.$$

This Lemma is a more general presentation of the approach used by Gopalan *et al*, which was based on Zinkevich's analysis of a general gradient projection algorithm [16]. We give a proof in the full version of the paper.

The definition and analysis of the EKM algorithm is omitted, but satisfies the following.

Lemma 11. *For any $n \geq 1$, $B, \epsilon, \delta > 0$, $\mu \in (-1,1)^n$, $f : \{-1,1\}^n \to [-B,B]$, given $m = \mathrm{poly}(n, B/\epsilon, \log(1/\delta))$ calls to f, with probability $\geq 1-\delta$, the Extended Kushilevitz-Mansour $\mathrm{EKM}_\mu(f,B,\epsilon,\delta)$ algorithm outputs a polynomial $p : \{-1,1\}^n \to \mathbb{R}$ such that,*

$$\|\hat{p}_\mu - \hat{f}_\mu\|_\infty \leq \epsilon,$$

and $\|\hat{p}\|_0 \leq 8B^2/\epsilon^2$. The runtime of EKM is polynomial in m.

We now generalize a procedure used by Gopalan *et al* [5] to keep the coefficients of a polynomial bounded in L_1 norm.

1.11. Projection

The projection operation is defined with respect to a product distribution μ, which determines the Fourier basis. (Alternatively, it could be defined simply for vectors in $\mathbb{R}^{2^n}$.) Consider the following function.

Definition 1. *Given a function f and $\ell \geq 0$, define* $\text{soft-threshold}(f,\mu,d,\ell)$ *as the function g where*

$$\hat{g}_\mu(S) = \begin{cases} \hat{f}_\mu(S) - \ell, & \textit{if } \hat{f}_\mu(S) \textit{ and } |S| \leq d \geq \ell \\ \hat{f}_\mu(S) + \ell, & \textit{if } \hat{f}_\mu(S) \leq -\ell \textit{ and } |S| \leq d \\ 0, & \textit{otherwise.} \end{cases} \tag{5}$$

This procedure is sometimes referred to as *soft thresholding* in practice. As we will show, $\mathrm{proj}_{\mu,K_{dt}}(f) = \text{soft-threshold}(f,\mu,d,\ell)$ for the smallest $\ell \geq 0$ such that $\|\text{soft-threshold}(f,\ell))\|_1 \leq t$. This is equivalent to the following continuous procedure. If $\|\hat{f}_\mu\|_1 \leq t$ output f. Otherwise,

a) Start decreasing the magnitudes of all nonzero Fourier coefficients of f by equal amounts.
b) If some coefficient reaches 0, it then stays at 0.
c) Continue this till we reach a g where $\|\hat{g}\|_1 = t$.

Lemma 12. *If f is represented by a list of nonzero coefficients, $\mathrm{proj}_{\mu,K_{dt}}(f)$ can be computed in time $O(\|\hat{f}\|_0 \log \|\hat{f}\|_0)$ [5].*

Proof: We first argue that $\mathrm{proj}_{\mu,K_{dt}}(f) = \text{soft-threshold}(f,\mu,d,\ell)$ for the smallest $\ell \geq 0$ such that $\|\text{soft-threshold}(f,\mu,d,\ell)\|_1 \leq t$. We then argue that this can be computed efficiently.

Let $f : \{-1,1\}^n \to \mathbb{R}$ and let $g = \mathrm{proj}_{\mu,K_{dt}}(f)$. By compactness of K_{dt}, and by strict convexity of $\|\hat{f} - \hat{g}\|^2$, g exists and is unique. By definition of K_{dt}, $\hat{g}(S) = 0$ for all S of size $|S| > d$. Hence, $\|\hat{f} - \hat{g}\|_2^2 = \sum_{|S|\leq d}(\hat{f}(S) - \hat{g}(S))^2 + \sum_{|S|>d}\hat{f}^2(S)$. Since the latter sum does not depend on g, WLOG we may assume $\hat{f}(S) = 0$ for all sets of size $|S| > d$. We may also assume WLOG that $\hat{f}(S) \geq 0$ for each S, in which case it is easy to see that $\hat{g}(S) \in [0, \hat{f}(S)]$.

Now, suppose there exist two sets S, T such that $\hat{f}(S) - \hat{g}(S) > \hat{f}(T) - \hat{g}(T)$. Then, because $y = x^2$ is a strictly convex function, for sufficiently small $\epsilon > 0$, the quantity $(\hat{f}(S) - \hat{g}(S))^2 + (\hat{f}(T) - \hat{g}(T))^2$ would strictly decrease if we decreased $\hat{g}(T)$ by ϵ and increased $\hat{g}(S)$ by ϵ. Since g minimizes $\|\hat{f} - \hat{g}\|^2$ over K_{dt}, it must be that this change would cause g to no longer be in K_{dt}. However, notice that this decrease/increase by ϵ does not increase $\|\hat{g}\|_1$ unless $\epsilon > \hat{g}(T)$.

Put another way, if $\hat{g}(T) > 0$, then we can modify g by a sufficiently small ϵ to decrease $\|\hat{f}-\hat{g}\|^2$ while keeping $g \in K_{dt}$, which would be a contradiction. Therefore, we conclude that

$$\hat{f}(S) - \hat{g}(S) > \hat{f}(T) - \hat{g}(T) \Rightarrow \hat{g}(T) = 0.$$

This implies that for some $\ell \geq 0$, for all S either $\hat{f}(S) - \hat{g}(S) = \ell$ or $\hat{f}(S) - \hat{g}(S) < \ell$ and $\hat{g}(S) = 0$, which means that $g = \text{soft-threshold}(f, \mu, d, \ell)$.

The algorithm can be implemented in exactly the same manner as that of Gopalan *et al*, except that we first zero out all $\hat{f}(S)$ for $|S| > d$. After that, if $\|\hat{f}\|_1 \leq t$, the answer is simply $\text{proj}_{\mu,K_{dt}}(f) = f = \text{soft-threshold}(f, \mu, d, 0)$. Otherwise, let $k = \|\hat{f}\|_0$ and sort the sets so that $0 < |\hat{f}(S_1)| \leq |\hat{f}(S_2)| \leq \ldots \leq |\hat{f}(S_k)|$, which can be done in time $O(k \log k)$. For each $i \leq k$, let $a_i = (k-i)|\hat{f}(S_i)| + \sum_{j \leq i} |\hat{f}(S_i)|$. It is easy to see that a_i is nondecreasing, that all k a_i's can be computed in one linear-time pass through the nonzero coefficients of f, and that $a_i = \|\hat{f}\|_1 - \|\text{soft-threshold}(f, \mu, d, \hat{f}(S_i))|_1$. Also, it is easy to see that the desired ℓ satisfies $\|\hat{f}\|_1 - \|\text{soft-threshold}(f, \mu, d, \ell)|_1 = \|\hat{f}\|_1 - t$. Hence, if $a_i \leq \|\hat{f}\|_1 - t \leq a_j$, then the desired ℓ is in $[|\hat{f}(S_i)|, |\hat{f}(S_j)|]$. After finding the range $\ell \in [a_i, a_j)$, the exact value of ℓ is determined by a simple formula. Finally, the $\text{soft-threshold}(f, d, \mu, \ell)$ is computed in linear time. ■

Another useful property shown by Gopalan *et al* is that two functions which are close in L_∞ norm become close in L_2 norm after projection onto the L_1 ball. In our context, we use the following modification for the degree-d constrained L_1 ball.

Lemma 13. *Let $f, g : \{-1,1\}^n \to \mathbb{R}$ be functions such that $\|\hat{f} - \hat{g}\|_\infty \leq \epsilon$. Then,*

$$\|\text{proj}_{\mu,K_{dt}}(f) - \text{proj}_{\mu,K_{dt}}(g)\|_2^2 \leq 4\epsilon t.$$

Proof: Again, WLOG, suppose $\hat{f}(S) = \hat{g}(S) = 0$ for all sets S of size $|S| > d$. Now, suppose that $a = \text{proj}_{\mu,K_{dt}}(f) = \text{soft-threshold}(f, d, \mu, \ell_1)$ and $b = \text{proj}_{\mu,K_{dt}}(g) = \text{soft-threshold}(g, d, \mu, \ell_2)$. WLOG suppose $\ell_1 \leq \ell_2$. Next, let $c = \text{soft-threshold}(g, d, \mu, \ell_1)$. Next, we claim $\|a - c\|_\infty \leq \|\hat{f} - \hat{g}\|_\infty$. This is because, on a term by term basis, moving any two real numbers $\hat{f}(S)$ and $\hat{g}(S)$ both a distance ℓ closer to 0 can only decrease the distance between the two numbers. Notice that $b = \text{soft-threshold}(c, d, \mu, \ell_2 - \ell_1)$. Next, we claim that $\|b - c\|_\infty \leq \epsilon$. The reason is that we know that c is within L_∞ distance ϵ of b, which has L_1 norm at most t. Hence, if we move all coordinates ϵ closer to 0, the resulting function will certainly be within K_{dt}. Finally,

$$\|a-b\|_2^2 \leq \|a-b\|_1 \cdot \|a-b\|_\infty \leq (\|a\|_1 - \|b\|_1) \cdot \|a-b\|_\infty \leq 2t\|a-b\|_\infty.$$

Using $\|a - b\|_\infty \leq \|a - c\|\infty + \|b - c\|_\infty \leq \epsilon + \epsilon$ completes the proof. ■

2. Part II: Learning from diversity

Let us first return to the setting of learning from diversity. We use a different notation more suitable for this part.

2.1. Preliminaries

For $x \in \{0,1\}^n$ and $S \subseteq [n] = \{1, 2, \ldots, n\}$, let $x[S] = \prod_{i \in S} x_i$ denote a conjunction. We consider t-sparse, degree-d, B-bounded, integer multilinear polynomials $f(x) = \sum_{i=1}^{t} b_i x[S_i]$, where the sets $S_i \subseteq [n]$ are distinct, $b_i \in \mathbb{Z}$, $|b_i| \leq B$, and $|S_i| \leq d$. We say f is in *canonical form* if the sets are arranged in order of size, breaking ties lexicographically. The *constant coefficient* is the coefficient in front of the term $x[\varnothing]$, e.g., $17 + 3x_1 + 7x_8 + 9x_1x_{11} + 17x_3x_5$ is in canonical form and the constant coefficient is 17. Let the *mindegree* of the polynomial be $|S_1|$. The *mindegree terms* are those terms whose degree equals $|S_1|$. We similarly define the *mindegree* of a univariate polynomial to be the smallest degree of a nonzero term, e.g., the min-degree of $3x^2 + 17x^4 + x^9$ is 2.

Let $|x| = \sum_i |x_i|$ and the p-biased product distribution be denoted by $\nu_p(x) = p^{|x|}(1-p)^{|x|}$. Let $\rho_c(x) = \frac{1}{2c}\int_{1/2-c}^{1/2+c} \nu_p(x)dp$. We may abuse notation and say that a polynomial is degree-d when it is degree $\leq d$ or t-sparse when it is $\leq t$-sparse.

The size of a decision tree is defined to be the number of leaves. We define the depth of the root of the tree to be 0. Thus a depth-d tree computes a degree-d multilinear polynomial. It is easy to see that a depth-d decision tree $f : \{0,1\}^n \to \{-1,1\}$ computes a degree-d, 3^d-sparse, 2^d-bounded integer multilinear polynomial.

2.2. Intuition

Suppose that the function to be learned was a parity on $\log n$ bits, $f(x) = \prod_{i \in S}(2x_i - 1)$. If we restrict ourselves to examples which have a $1/2 - c$ fraction of 1's, then a simple argument shows that the bits in S will be correlated with f while the other bits will not. More generally, it can be shown that for any $O(\log(n))$-Junta, there will be some $p \in [1/2 - c, 1/2 + c]$ such that among examples with pn 1's, *at least one* of the relevant bits will have an inverse-polynomial correlation. Once one finds a relevant bit, one can recursively solve the Junta problem using divide and conquer. This intuition is misleadingly simple, however, because an actual depth-$O(\log(n))$ tree *can in general depend on all n bits*. Hence, it is not enough to identify the relevant bits.

To illustrate our approach, consider the two functions below.

$$f_1(x) = x_1 - x_2x_3x_4$$
$$f_2(x) = x_1x_2 - x_2x_3 + x_3x_4 - x_4x_1$$

As mentioned, the first step is to use the model of multiple random sources (as in [1]): we can simulate draws from any p-biased distribution we want, for $p \in [1/2-c, 1/2+c]$. This is done by (somewhat carefully) partitioning the examples based on the number of 1's. Now notice that,

$$g_1(p) = \mathrm{E}_{x \sim \nu_p}[f_1(x)] = p - p^3$$
$$g_2(p) = \mathrm{E}_{x \sim \nu_p}[f_2(x)] = 0$$

The above polynomials g_1, g_2 may be estimated by interpolation. In the case of f_1, g_1 reveals that there are degree-1 and degree-3 terms (and perhaps others) in f_1. To find one, we can further look at $\mathrm{E}_{x \sim \nu_p}[f_1(x)|x_i = 1]$ for some i –if we pick a relevant bit i, then the interpolated function will change (for example $i = 1$ gives a conditional expectation of $1 - p^3$). By conditioning on further variables, we can find degree-d terms in time and sample complexity exponential in d. However, f_2 illustrates that the approach just described is not enough, because $\mathrm{E}_{x \sim \nu_p}[f_2(x) | x_i = 1] = 0$ for all i.

The key "trick" is to look at $\mathrm{E}_{x \sim \nu_p}[f^2(x)]$. Note that for any $x \in \{0,1\}^n$ (using $x_i^2 = x_i$), $f_2^2(x) = x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1 - 3x_1x_2x_3 + \ldots$. The point is that now there all degree-2 terms have the same sign, and hence $\mathrm{E}_{x \sim \nu_p}[f_2^2(x)] = 4p^2 + \ldots$, so cancelation cannot make the polynomial 0. Intuition is coming from the fact is if f is nonconstant yet the mean of f is constant across all p-biased distributions, then the variance cannot be constant. In statistics, the term *heteroscedasticity* refers to the fact that the variance of a function may be different on different regions of the input. This is essentially what we are taking advantage of here. Interestingly, the (nonorthogonal) representation of polynomials over $\{0,1\}^n$, e.g., $f(x) = x_1x_2x_3$ as a *monomial*, is used for this part due to certain appealing properties not possessed by the more common Fourier representation.

2.3. Algorithm

The algorithm learns sparse low-degree integral polynomials. For simplicity, we assume that that the algorithm is given all of the relevant parameters, c, n, t, B as input (we take them to be global variables). If c is not known in advance, it may be estimated to any sufficient inverse polynomial accuracy in polynomial time. The assumption that t and B are known may be removed using the doubling trick (run the algorithm starting with a low estimates – each time it fails, double them and restart). We prove the following generalization of Theorem 1.

Theorem 14. *Fix any constant $c > 0$. Then there is a polynomial M such that, for any $\delta \in (0,1)$, $n, d, t, B \geq 1$, and any t-sparse B-bounded degree-d integer polynomial $f : \{0,1\}^n \to \mathbb{Z}$, for $m \geq M(2^d ntB \log 1/\delta)$ examples $(x^i, f(x^i))$ where each x^i is chosen independently from ρ_c, with probability $\geq 1 - \delta$, the algorithm described in section 2.4 outputs a polynomial exactly equivalent to f and runs in time poly(m).*

2.4. Algorithm description and analysis

As mentioned in the introduction, a useful trick in recovering a polynomial over $\{0,1\}^n$ is squaring it, because the mindegree coefficients all are squared.

Observation 15. *Let $f(x) = \sum_i a_i x[S_i]$ be a multilinear polynomial in canonical form. Let $f^2(x) = \sum_i b_i x[T_i]$ be the canonical representation of $f^2(x)$. Then $S_1 = T_1$ and $b_i > 0$ for all mindegree terms, i.e., terms where $|S_i| = |S_1|$.*

The above observation follows from the fact that $x_i^2 = x_i$ and hence $x[S]x[T] = x[S \cup T]$.

The algorithm learns the decision tree as a polynomial. Let $f(x) = \sum_{i=1}^t a_i x[S_i]$ be a integer polynomial in canonical form. Say it is degree $\leq d$ and B-bounded. We assume that we are given as input m samples $(x^i, f(x^i))$, for $i = 1, 2, \ldots, m$, where x^i are independently drawn from ρ_c. The goal is to output exactly the same polynomial in canonical form. We will do this by identifying the nonzero coefficients one at a time, in canonical order.

Computing the first coefficient. The first useful fact is that if we are told the first nonzero canonical set, i.e., S_1, then we can compute its coefficient a_1 using samples and time exponential in d. Even this is not obvious (as opposed to the standard Fourier representation). In particular, it is not clear how to do this for polynomial-sized decision trees (as opposed to $O(\log n)$-depth trees). Roughly speaking, the coefficient estimation is done by clustering the examples based on the different fractions of 1's and using interpolation. More precisely, in Section 2.5, we give more general procedure that does what we call T-interpolation.

Definition 2. *For a multilinear polynomial $g(x) = \sum_i a_i x[S_i]$ and $T \subseteq [n]$, let the T-interpolation of g be the polynomial,*

$$g_{\langle T \rangle}(p) = \sum_i a_i p^{|S_i \setminus T|}$$

It is clear that the constant coefficient of $f_{\langle S_1 \rangle}(p)$ is equal to a_1. Hence, given the first set with nonzero coefficient for any function, we can estimate that coefficient. The algorithm for efficiently performing T interpolation is given in Section 2.5, but we state its guarantee here.

Lemma 16. *For any constant $c \in (0, 1/2)$, there is a polynomial M such that, for any $\delta \in (0,1)$, $n, t, d, B \geq 1$, $T \subseteq [n]$ with $|T| \leq d$, and any degree-d t-sparse B-bounded integer multilinear polynomial g, using $m \geq M(2^d tBn \log(1/\delta))$ examples $(x^1, g(x^1)), \ldots, (x^m, g(x^m))$, with probability $\geq 1 - \delta$, algorithm T-INTERPOLATION outputs the T-interpolation polynomial $g_{\langle T \rangle}(p)$.*

Define the jth *residual* $f_j(x) = \sum_{i=j}^t a_i x[S_i]$. By the above, it suffices to identify the sets S_i in canonical order, because we can then estimate a_j as the constant coefficient of $f_{j\langle S_j \rangle}$. Notice that once we have computed (a_i, S_i) for $i = 1, 2, \ldots, j-1$, we can evaluate the jth residual $f_j(x^i) = f(x^i) - \sum_{k=1}^{j-1} a_k x^i[S_k]$ and thus translate samples $(x^i, f(x^i))$ to samples $(x^i, f_j(x^i))$. So it remains to describe how we find the canonically first term in the j residual, i.e., S_j.

Finding the canonically first set. We begin, as suggested by Observation 15, by computing the $\varnothing$-interpolation of f_j^2, $f^2_{j\langle \varnothing \rangle}(p)$, from the data, using algorithm T-INTERPOLATION. The result is a degree $\leq 2d$ integer polynomial in p. If it is identically 0, we output the polynomial f_{j-1} and we are done. Otherwise, let d' be the mindegree of $f^2_{j\langle \varnothing \rangle}$. By Observation 15, we have that d' is equal to the mindegree of f_j^2 and f_j. This follows directly from the fact that all coefficients of mindegree terms of f_j^2 are positive – there is no cancelation when substitute $x_i = p$ for all i. Let $S_j = \{i_1, i_2, \ldots, i_{d'}\}$ with $i_1 < i_2 < \ldots < i_{d'}$. Notice that $i_1 \in [n]$ is the smallest index such that the mindegree of $f^2_{j\langle \{i_1\} \rangle}$ is $d' - 1$, $i_2 \in \{i_1 + 1, i_1 + 2, \ldots, n\}$ is the smallest index such that the mindegree of $f^2_{j\langle \{i_1, i_2\} \rangle}$ is $d' - 2$, and so forth. This gives a means for identifying the set S_j using at most n calls to T-INTERPOLATION.

To complete the description of the algorithm, we need to describe the T-Interpolation algorithm. A formal analysis of runtime and proof of Theorem 14 is omitted due to lack of space.

2.5. T-Interpolation algorithm

```
Algorithm $T$-interpolation.
Input: $T \subseteq [n]$ and $(x^1, y^1), (x^2, y^2), \ldots, (x^m, y^m) \in \{0,1\}^n \times \mathbb{Z}$.
(also assumes knowledge of $n, d \geq 1$, and $c \in (0, 1/2)$)
  1) For $i := 0, 1, 2, \ldots, d$:
    a) Let $p_i := \frac{1}{2} - c + i\frac{2c}{d}$
    b) Let $D_i := \emptyset$.   (* FILTER DATA SUBSET $D_i \subseteq \{1, 2, \ldots, m\}$ *)
    c) For $j = 1, 2, \ldots, m$:
       If $x^j[T] = 1$ then with probability $\frac{\nu_{p_i}(x^j)}{8n\rho_c(x^j)}$, let $D_i := D_i \cup \{j\}$.
    d) Let $y_i := \frac{1}{|D_i|}\sum_{j \in D_i} y^j$.
  2) Lagrange interpolation: Let $r : \mathbb{R} \to \mathbb{R}$ be,
       $$r(p) = \sum_{i=0}^{d} y_i \prod_{j \neq i} \frac{p - p_j}{p_i - p_j}.$$
  3) Collect terms to write $r(p) = \sum_{k=0}^{d} c_k p^k$.
  4) Round each coefficient of $r$ to the nearest integer and output the resulting polynomial.
```

Steps (b) and (c) create a subset of the data, with indices D_i which appears to be drawn from the distribution ν_{p_i} conditioned on the fact that all bits in T are 1. This is done by rejection sampling. In order to see that the algorithm is well-defined, one must verify that $\frac{\nu_{p_i}(x^j)}{8n\rho_c(x^j)} \in [0, 1]$, the proof of which is omitted. Second, we need to explain how one computes this ratio. It is easy to compute $\nu_{p_i}(x^j) = p_i^{\sum_k x_k^j}(1 - p_i)^{\sum_k 1 - x_k^j}$ exactly. Computing $\rho_c(x)$ exactly involves the straightforward expansion and integration of a univariate degree-n polynomial.

3. Conclusions

We have made progress on the problems of learning DNF and decision trees from random examples, by introducing algorithms and new models in which to analyze them. From a practical point of view, perhaps the most limiting assumption from ours and prior work is that the distribution is a product distribution. It would be interesting to see if the smoothed analysis paradigm could be extended beyond product distributions.

Interestingly, the Greedy Feature Construction algorithm is similar to (and could likely be replaced by) such as Feature Construction algorithms [14] that learn sparse polynomials. Our analysis shows that such algorithm PAC-learn decision trees over product distributions, in a smoothed analysis model. We also generalize to DNF and agnostically learning decision trees with more elaborate algorithms.

References

[1] J. Arpe and E. Mossel, "Multiple random oracles are better than one," *CoRR*, vol. abs/0804.3817, 2008.

[2] L. Bazzi, "Polylogarithmic independence can fool dnf formulas," in *FOCS*. IEEE Computer Society, 2007, pp. 63–73.

[3] J. Bourgain, "On the distribution of polynomials on high-dimensional convex sets," in *Geometric aspects of functional analysis (1989–90)*, ser. Lecture Notes in Math. Berlin: Springer, 1991, vol. 1469, pp. 127–137.

[4] A. Carbery and J. Wright, "Distributional and L^q norm inequalities for polynomials over convex bodies in $\mathbb{R}^n$," *Math. Res. Lett.*, vol. 8, no. 3, pp. 233–248, 2001.

[5] P. Gopalan, A. T. Kalai, and A. R. Klivans, "Agnostically learning decision trees," in *Proceedings of the 40th annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 2008, pp. 527–536.

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

[7] J. Jackson and R. Servedio, "Learning random log-depth decision trees under the uniform distribution," in *Proceedings of the 16th Annual Conf. on Computational Learning Theory and 7th Kernel Workshop*, 2003, pp. 610–624.

[8] A. T. Kalai, V. Kanade, and Y. Mansour, "Reliable agnostic learning," in *Proc. Conf. on Learning Theory (COLT'09)*, 2009.

[9] M. Kearns, R. Schapire, and L. Sellie, "Toward Efficient Agnostic Learning," *Machine Learning*, vol. 17, no. 2/3, pp. 115–141, 1994.

[10] E. Kushilevitz and Y. Mansour, "Learning decision trees using the Fourier spectrum," *SIAM Journal of Computing*, vol. 22(6), pp. 1331–1348, 1993.

[11] D. Polymath, "A new proof of the density hales-jewett theorem," 2009, presentation by Ryan O'Donell at Microsoft Research New England.

[12] J. B. Rosen, "The gradient projection method for nonlinear programming. part i. linear constraints," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 1, pp. 181–217, 1960.

[13] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *J. ACM*, vol. 51, no. 3, pp. 385–463, 2004.

[14] R. S. Sutton and C. J. Matheus, "Learning polynomial functions by feature construction," in *ML*, 1991, pp. 208–212.

[15] L. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[16] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20^{th} Intl. Conf. on Machine Learning (ICML'03)*, 2003, pp. 928–936.