Learning DNF Expressions from Fourier Spectrum

Vitaly Feldman VITALY@POST.HARVARD.EDU
IBM Almaden Research Center
San Jose, CA, USA

Editor: Shie Mannor, Nathan Srebro, Robert C. Williamson

Abstract

Since its introduction by Valiant in 1984, PAC learning of DNF expressions remains one of the central problems in learning theory. We consider this problem in the setting where the underlying distribution is uniform, or more generally, a product distribution. Kalai, Samorodnitsky, and Teng (2009b) showed that in this setting a DNF expression can be efficiently approximated from its "heavy" low-degree Fourier coefficients alone. This is in contrast to previous approaches where boosting was used and thus Fourier coefficients of the target function modified by various distributions were needed. This property is crucial for learning of DNF expressions over smoothed product distributions, a learning model introduced by Kalai et al. (2009b) and inspired by the seminal smoothed analysis model of Spielman and Teng (2004).

We introduce a new approach to learning (or approximating) a polynomial threshold functions which is based on creating a function with range $[-1, 1]$ that approximately agrees with the unknown function on low-degree Fourier coefficients. We then describe conditions under which this is sufficient for learning polynomial threshold functions. Our approach yields a new, simple algorithm for approximating any polynomial-size DNF expression from its "heavy" low-degree Fourier coefficients alone. This algorithm greatly simplifies the proof of learnability of DNF expressions over smoothed product distributions and is simpler than all previous algorithm for PAC learning of DNF expression using membership queries. We also describe an application of our algorithm to learning monotone DNF expressions over product distributions. Building on the work of Servedio (2004), we give an algorithm that runs in time $\mathrm{poly}((s \cdot \log{(s/\epsilon)})^{\log{(s/\epsilon)}}, n)$, where s is the size of the DNF expression and ϵ is the accuracy. This improves on $\mathrm{poly}((s \cdot \log{(ns/\epsilon)})^{\log{(s/\epsilon)} \cdot \log{(1/\epsilon)}}, n)$ bound of Servedio (2004).

Keywords: PAC learning, polynomial threshold function, smoothed analysis, monotone DNF

1. Introduction

PAC learning of DNF expressions (or formulae) is the problem posed by Valiant (1984) in his seminal work that introduced the PAC model. The original problem asks whether polynomial-size DNF expressions are learnable from random examples on points sampled from an unknown distribution. Despite efforts by numerous researchers, the problem still remains open, with the best algorithm taking $2^{\tilde{O}(\sqrt[3]{n})}$ time (Klivans and Servedio, 2004). In the course of this work, a number of restricted versions of the problem were introduced and studied. One such assumption is that the distribution over the domain (which is the n-dimensional hypercube $\{-1, 1\}^n$) is uniform, or more generally, a product distribution. In this setting a simple quasi-polynomial $n^{O(\log n)}$ algorithm for learning DNF expressions was found by Verbeurgt (1990). However, no substantially better algorithms are known so far even for much simpler classes such as functions of at most $\log n$-variables ($\log n$-juntas).

Another natural restriction commonly considered is monotone DNF (MDNF) expressions, i.e. those without negated variables. Without restrictions on the distribution, the problem is no easier than the original one (Kearns et al., 1987) but appears to be easier for product distributions. Sakai and Maruoka (2000) gave a polynomial-time algorithm for $\log n$-term MDNF learning and Bshouty and Tamon (1996) gave an algorithm for learning a class of functions which includes $O(\log^2 n/\log\log n)$-term MDNFs. Most recently, Servedio (2004) proved a substantially stronger result: s-term MDNFs are learnable to accuracy ϵ in time polynomial in $(s{\cdot}\log{(ns/\epsilon)})^{\log{(s/\epsilon)}\cdot\log{(1/\epsilon)}}$ and n. In particular, his result implies that $O(2^{\sqrt{\log n}})$-term MDNFs are learnable in polynomial time to any constant accuracy. Numerous other restrictions of the original problem were considered. We refer the interested reader to Servedio's paper (2004) for a more detailed overview.

Several works also considered the problem in the stronger *membership query* (MQ) model. In this model the learner can ask for a value of the unknown function at any point in the domain. Valiant (1984) gave an efficient MQ learning algorithm for MDNFs of polynomial size. In a celebrated result, Jackson (1997) gave a polynomial time MQ learning algorithm for DNFs over product distributions. Jackson's algorithm uses the Fourier transform-based learning technique (Linial et al., 1993) and combines the Kushilevitz-Mansour algorithm for finding a "heavy" Fourier coefficient of a boolean function (Goldreich and Levin, 1989; Kushilevitz and Mansour, 1993) with the Boosting-by-Majority algorithm of Freund (1995). A similar approach was used in the subsequent improvements to Jackson's algorithm (Klivans and Servedio, 2003; Bshouty et al., 2004; Feldman, 2007).

The access to membership queries is clearly a very strong assumption and is unrealistic in most learning applications. Several works give DNF learning algorithms which relax this requirement: the learning algorithm of Bshouty and Feldman (2002) uses random examples from product distributions chosen by the algorithm and the algorithm of Bshouty et al. (2005) uses only examples produced by a random walk on the hypercube. Another approach is to relax the requirement that the PAC algorithm succeeds on all polynomial-size DNF formulae and require it to succeed on a randomly chosen expression generated from some simple distribution over the formulae (Aizenstein and Pitt, 1995). Strong results of this form were achieved recently by Jackson et al. (2011) and Sellie (2009).

A new way to avoid the worst-case hardness of learning DNF was recently proposed by Kalai et al. (2009b). Their model is inspired by the seminal model of smoothed analysis introduced in the context of optimization and numerical analysis by Spielman and Teng (2004). Smoothed analysis is based on the insight that, in practice, real-valued inputs or parameters of the problem are a result of noisy and imprecise measurements. Therefore the complexity of a problem is measured not on the worst-case values but on a random perturbation of those values. In the work of Kalai et al. (2009b) the perturbed parameters are the expectations of each of the coordinates of a product distribution over $\{-1,1\}^n$. In a surprising result they showed that DNF formulae are learnable efficiently in this model (and that decision trees are even learnable agnostically).

A crucial and the most involved component of the DNF learning algorithm of Kalai et al. (2009b) is the algorithm that – given all "heavy" (here this refers to those of inverse-polynomial magnitude), low-degree (logarithmic in the learning parameters) Fourier coefficients of the target DNF f to inverse-polynomial accuracy – finds a function that is ϵ-close to f. Such an algorithm is necessary since, in the boosting-based approach of Jackson (1997), the weak learner needs to learn with respect to distributions which depend on previous weak hypotheses. When learning over a smoothed product distribution, the first weak hypothesis depends on the specific perturbation and therefore in the subsequent boosting stages, the parameters of the product distribution can no longer be thought of as perturbed randomly. Kalai et al. (2009b) show that this is not only a matter of complications in the analysis but an actual limitation of the boosting-based approach. Therefore they used an algorithm that first collects all the "heavy" low-degree Fourier coefficients and then relies solely on this information to approximate the target function.

1.1. Our Results

We describe a new approach to the problem of learning a polynomial threshold function (PTF) from approximations of its "heavy" low-degree Fourier coefficients, a problem we believe is interesting in its own right. The approach exploits a generalization of a simple structural result about any s-term DNF f: for every function $g : \{-1,1\}^n \to [-1,1]$, the error of g on f (measured as $\mathbf{E}_{\mathcal{U}}[|f(x) - g(x)|]$) is at most $\gamma \cdot (2s+1)$, where γ is the magnitude of the largest difference between two corresponding Fourier coefficients of f and g (Kalai et al., 2009b). We use $\hat{f}$ to denote the vector of Fourier coefficients of f and so this difference can be expressed as $\|\hat{f} - \hat{g}\|_\infty$. Hence to find a function ϵ-close to f it is sufficient to find a function g such that $\|\hat{f} - \hat{g}\|_\infty \le \epsilon/(2s+1)$, in other words, g that has approximately (in the infinity norm) the same Fourier spectrum as f. We give a new, simple algorithm (Th. 13) that constructs a function (with range in $[-1,1]$) which has approximately the desired Fourier spectrum.

Our algorithm builds g in a fairly straightforward way: starting with a constant $g_0 \equiv 0$ function we iteratively correct each coefficient to the desired value (by adding the difference in the coefficients multiplied by the corresponding basis function). After each such step the new function g_t might have values outside of $[-1,1]$. We correct this by "cutting-off" values outside of $[-1,1]$ (in other words, project them to $[-1,1]$). A simple argument shows that both of these operations reduce $\|f - g_t\|_2^2 = \mathbf{E}_{\mathcal{U}}[f(x) - g_t(x))^2]$. The coefficient correction procedure reduces this squared distance measure significantly and implies the convergence of the algorithm. In addition, through a slightly more complicated potential argument we show that there is no need to perform the projection after each coefficient update; a single projection after all updates suffices (Th. 21). This implies that the function we construct via this algorithm is itself a polynomial threshold function (PTF).

To generalize our approach to product distributions, we strengthen the structural lemma about DNF expressions to measure the error in terms of the largest difference between corresponding low-degree Fourier coefficients and extend it to product distributions (Th. 11). The algorithm itself uses the Fourier basis for the given product distribution but otherwise remains essentially unchanged. We also give a more general condition on PTFs that is sufficient for bounding $\mathbf{E}_{\mathcal{U}}[|f(x) - g(x)|]$ in terms of largest difference between corresponding low-degree Fourier coefficients of f and g. The general condition implies that our algorithm can also be used to learn any integer-weight linear threshold of terms as long as the sum of the magnitudes of weights (or the *total weight*) is polynomial.

We give several applications of our approach. The most immediate one is to obtain a simple algorithm for learning DNF expressions over product distributions with membership queries (Cor. 15). Given access to membership queries, the Fourier spectrum of any function can be approximated using the well-known Kushilevitz-Mansour algorithm and its generalization to product distributions (Goldreich and Levin, 1989; Kushilevitz and Mansour, 1993). We can then apply our approximation algorithm to get a hypothesis which is ϵ close to the target function. While technically our iterative algorithm is similar to boosting, the resulting algorithm for learning DNF is simpler and more self-contained than previous boosting-based algorithms.

The second application of our approximation algorithm and the motivation for this work is its use in the context of smoothed analysis of learning DNF over product distributions (Th. 18) where the problem was originally formulated and solved by Kalai et al. (2009b). The approximation algorithm of Kalai et al. (2009b) is based on an elaborate combination of the *positive-reliable* DNF learning algorithm of Kalai et al. (2009a) and the agnostic learning algorithm for decisions trees of Gopalan et al. (2008). In contrast, our algorithm gives a natural solution to the problem which is significantly simpler technically and is more general. We also note that the algorithm of Kalai et al. (2009b) does not construct a function with Fourier transform close to that of f and is not based on the structural results we use.

In another application of our approach we give a new algorithm for learning MDNF expressions over product distributions. Our algorithm is based on Servedio's algorithm for learning MDNFs (Servedio, 2004). The main idea of his algorithm is to restrict the target function to influential variables, those that can change the value of the target function with significant probability. For any monotone function, influential variables can be easily identified. Then all the Fourier coefficients of low degree and restricted to influential variables are estimated individually from random examples. The sign of the resulting low-degree polynomial is used as a hypothesis. The degree for which such an approximation method is known to work is $20 \cdot \log{(s/\epsilon)} \cdot \log{(1/\epsilon)})$ (Mansour, 1995). Using our simple structural result about DNF and our algorithm for constructing a function with desired Fourier coefficients, we show (Th. 19) that to achieve ϵ-accuracy coefficients of degree at most $O(\log{(s/\epsilon)})$ are sufficient. This results in $\mathrm{poly}((s \cdot \log{(s/\epsilon)})^{\log{(s/\epsilon)}}, n)$ time algorithm improving on $\mathrm{poly}((s \cdot \log{(ns/\epsilon)})^{\log{(s/\epsilon)} \cdot \log{(1/\epsilon)}}, n)$ bound of Servedio (2004).

Related work. A closely related problem of finding a function with specified correlations with a given set of functions was considered by Trevisan et al. (2009) and their solution is based on a similar algorithm (with a more involved analysis). Our setting differs in that the set of functions with which correlations are specified has a superpolynomial size and the functions are not necessarily boolean (when the distribution is non-uniform).

In the Chow Parameter problem the goal is to find an approximation to a linear threshold function (LTF) f from its degree-1 and degree-0 Fourier coefficients (the Chow parameters). O'Donnell and Servedio (2011) gave the first algorithm for the problem which is based on finding a function whose Chow parameters are close in Euclidean distance to those of f (as opposed to $\|\cdot\|_\infty$ distance in our problem). Then they used an intricate structural result about LTFs to derive an approximation bound. Their algorithm is based on a brute-force search of some of the Chow parameters. A very recent, doubly exponential improvement to the solution of the problem was obtained using a new, stronger structural result and a new algorithm for constructing a linear threshold function from approximations of Chow parameters (De et al., 2012). As in our applications, the algorithm of De et al. (2012) constructs a bounded function with the given degree-1 Fourier spectrum. However the update step of their algorithm is optimized for minimizing the Euclidean distance of the Chow parameters of the obtained function to the given ones.

Organization. Structural results required for approximating DNF expressions and PTFs are given in Section 3. In Section 4 we describe our main algorithm for constructing a function with the desired Fourier spectrum. In Section A we give applications of our approach. Several missing proofs appear in the full version of the paper available on arXiv (Feldman, 2012).

2. Preliminaries

For an integer k, let $[k]$ denote the set $\{1, 2, \ldots, k\}$. For a vector $v \in \mathbb{R}^k$, we use the following notation for several standard quantities: $\|v\|_0 = |\{i \in [k] \mid v_i \neq 0\}|$, $\|v\|_1 = \sum_{i \in [k]} |v_i|$, $\|v\|_\infty = \max_{i \in [k]}\{|v_i|\}$ and $\|v\|_2 = \sqrt{\sum_{i \in [k]} v_i^2}$. For a real value α, we denote its projection to $[-1, 1]$ by $P_1(\alpha)$. That is, $P_1(\alpha) = \alpha$ if $|\alpha| \leq 1$ and $P_1(\alpha) = \mathsf{sign}(\alpha)$, otherwise.

We refer to real-valued functions with range in $[-1, 1]$ as *bounded*. Let $B_d = \{a \in \{0, 1\}^n \mid \|a\|_0 \leq d\}$. For $a \in \{0, 1\}^n$ let $\chi_a(x)$ denote the function $\prod_{a_i = 1} x_i$. It is a monomial and also a parity function over variables with indices in $\{i \leq n \mid a_i = 1\}$. A degree-$d$ polynomial threshold function is a function representable as $\mathsf{sign}(\sum_{a \in B_d} w(a)\chi_a(x))$ for some vector of weights $w \in \mathbb{R}^{B_d}$. When the representing vector w is sparse we can describe it by listing all the non-zero coefficients only. We refer to this as being *succinctly represented*.

PAC learning. Our learning model is Valiant's (1984) well-known PAC model. In this model, for a concept f and distribution D over $\{-1, 1\}^n$, an *example oracle* $\mathrm{EX}(f, D)$ is an oracle that, upon request, returns an example $(x, f(x))$ where x is chosen randomly with respect to D, independently of any previous examples. A *membership query* (MQ) learning algorithm is an algorithm that has oracle access to the target function f in addition to $\mathrm{EX}(f, D)$, namely it can, for every point $x \in \{-1, 1\}^n$ obtain the value $f(x)$. For $\epsilon \geq 0$, we say that function g is ϵ-close to function f relative to distribution D if $\mathbf{Pr}_D[f(x) = g(x)] \geq 1 - \epsilon$. For a concept class C, we say that an algorithm $\mathcal{A}$ *efficiently* learns C over distribution D, if for every $\epsilon > 0$, n, $f \in C$, $\mathcal{A}$ outputs, with probability at least $1/2$ and in time polynomial in n/ϵ, a hypothesis h that is ϵ-close to f relative to D. Learning of DNF expressions is commonly parameterized by the size s (i.e. the number of terms) of the smallest-size DNF representation of f. In this case the running time of the efficient learning algorithm is also allowed to depend polynomially on s. For $k \in [n]$ an s-term k-DNF expression is a DNF expression with s terms of length at most k.

Fourier transform. A number of methods for learning over the uniform distribution $\mathcal{U}$ are based on the Fourier transform technique. The technique relies on the fact that the set of all parity functions $\{\chi_a(x)\}_{a \in \{0,1\}^n}$ forms an orthonormal basis of the linear space of real-valued function over $\{-1, 1\}^n$ with inner product defined as $\langle f, g \rangle_{\mathcal{U}} = \mathbf{E}_{\mathcal{U}}[f(x)g(x)]$. This fact implies that any real-valued function f over $\{-1, 1\}^n$ can be uniquely represented as a linear combination of parities, that is $f(x) = \sum_{a \in \{0,1\}^n} \hat{f}(a)\chi_a(x)$. The coefficient $\hat{f}(a)$ is called Fourier coefficient of f on a and equals $\mathbf{E}_{\mathcal{U}}[f(x)\chi_a(x)]$; $\|a\|_0$ is called the *degree* of $\hat{f}(a)$. For a set $S \subseteq \{0, 1\}^n$ we use $\hat{f}(S)$ to denote the vector of all coefficients with indices in S and $\hat{f}$ to denote the vector of all the Fourier coefficients of f. The vector of all degree-$(\leq d)$ Fourier coefficients of f can then be expressed as $\hat{f}(B_d)$. We also use a similar notation for vectors of estimates of Fourier coefficients. Namely, for $S \subseteq \{0, 1\}^n$ we use $\tilde{f}(S)$ to denote a vector in $\mathbb{R}^S$ indexed by vectors in S. We denote by $\tilde{f}(a)$ the a-th element of $\tilde{f}(S)$. Whenever appropriate, we use succinct representations for vectors of Fourier coefficients (i.e. listing only the non-zero coefficients).

We will make use of Parseval's identity which states that for every real-valued function f over $\{-1, 1\}^n$, $\mathbf{E}_{\mathcal{U}}[f^2] = \sum_a \hat{f}(a)^2 = \|\hat{f}\|_2^2$. Given oracle access to a function f (i.e. membership queries), the Fourier transform of a function can be approximated using the KM algorithm (Goldreich and Levin, 1989; Kushilevitz and Mansour, 1993)

Theorem 1 (KM algorithm) *There exists an algorithm that for any real-valued function $f : \{-1, 1\}^n \to [-1, 1]$, given parameters $\theta > 0$, $\delta > 0$ and oracle access to f, with probability at least $1 - \delta$, re-*

turns a succinctly represented vector $\tilde{f}$, such that $\|\hat{f} - \tilde{f}\|_\infty \leq \theta$ and $\|\tilde{f}\|_0 \leq 4/\theta^2$. The algorithm runs in $\tilde{O}(n^2 \cdot \theta^{-6} \cdot \log(1/\delta))$ time and makes $\tilde{O}(n \cdot \theta^{-6} \cdot \log(1/\delta))$ queries to f.

Product distributions. We consider learning over product distributions on $\{-1,1\}^n$. For a vector $\mu \in (-1,1)^n$ let D_μ denote the product distribution over $\{-1,1\}^n$ such that $\mathbf{E}_{x\sim D_\mu}[x_i] = \mu_i$ for every $i \in [n]$. For each $i \in [n]$, $x_i = 1$ with probability $(1+\mu_i)/2$. For $c \in (0,1]$ the distribution D_μ is said to be c-bounded if $\mu \in [-1+c, 1-c]^n$. The uniform distribution is then equivalent to $D_{\bar{0}}$, where $\bar{0}$ is the all-zero vector, and is 1-bounded. We use $\mathbf{E}_\mu[\cdot]$ to denote $\mathbf{E}_{x\sim D_\mu}[\cdot]$ and $\mathbf{E}[\cdot]$ to denote $\mathbf{E}_{x\sim\mathcal{U}}[\cdot]$ and similarly for $\mathbf{Pr}$.

The Fourier transform technique extends naturally to product distributions (Furst et al., 1991). For $\mu \in (-1,1)^n$ the inner product is defined as $\langle f,g\rangle_\mu = \mathbf{E}_\mu[f(x)g(x)]$. The corresponding orthonormal basis of functions over D_μ is given by the set of functions $\{\phi_{\mu,a} \mid a \in \{0,1\}^n\}$, where $\phi_{\mu,a}(x) = \prod_{a_i=1} \frac{x_i-\mu_i}{\sqrt{1-\mu_i^2}}$. Every function $f : \{-1,1\}^n \to \mathbb{R}$ can be uniquely represented as $f(x) = \sum_{a\in\{0,1\}^n} \hat{f}_\mu(a)\phi_{\mu,a}(x)$, where the μ-Fourier coefficient $\hat{f}_\mu(a)$ equals $\mathbf{E}_\mu[f(x)\phi_{\mu,a}(x)]$. We extend our uniform-distribution notation for vectors of Fourier coefficients to product distributions analogously. For any product distribution μ, a degree-d polynomial $p(x)$ has no non-zero μ-Fourier coefficients of degree greater than d.

The KM algorithm has been extended to product distributions by Bellare (1991) (see also Jackson, 1997). Below we describe a more efficient version given by Kalai et al. (2009b) (referred to as the EKM algorithm) which is efficient for all product distributions.

Theorem 2 (EKM algorithm) *There exists an algorithm that for any real-valued function $f : \{-1,1\}^n \to [-1,1]$, given parameters $\theta > 0$, $\delta > 0$, $\mu \in (-1,1)^n$, and oracle access to f, with probability at least $1-\delta$, returns a succinctly represented vector $\tilde{f}_\mu$, such that $\|\hat{f}_\mu - \tilde{f}_\mu\|_\infty \leq \theta$ and $\|\tilde{f}_\mu\|_0 \leq 4/\theta^2$. The algorithm runs in time polynomial in n, $1/\theta$ and $\log(1/\delta)$.*

When learning relative to distribution D_μ we can assume that μ is known to the learning algorithm. For our purposes a sufficiently-close approximation to μ can always be obtained by estimating μ_i for each i using random samples from D_μ.

Without oracle access to f, but given examples of f on points drawn randomly from D_μ one can estimate the Fourier coefficients up to degree d by estimating each coefficient individually in a straightforward way (that is, by using the empirical estimates). A naïve way of analyzing the number of samples required to achieve certain accuracy requires a number of samples that depends on μ and the degree of the estimated coefficient (since $|\phi_{\mu,a}(x)|$ depends on them). Kalai et al. (2009b) gave a more refined analysis which eliminates the dependence on d and μ and implies the following theorem.

Theorem 3 (Low Degree Algorithm) *There exists an algorithm that for any real-valued function $f : \{-1,1\}^n \to [-1,1]$ and $\mu \in (-1,1)^n$, given parameters $d \in [n]$, $\theta > 0$, $\delta > 0$, and access to EX(f, D_μ), with probability at least $1-\delta$, returns a succinctly-represented vector $\tilde{f}_\mu$, such that $\|\hat{f}_\mu(B_d) - \tilde{f}_\mu(B_d)\|_\infty \leq \theta$ and $\|\tilde{f}_\mu\|_0 \leq 4/\theta^2$. The algorithm runs in time $n^d \cdot$ poly$(n \cdot \theta^{-1} \cdot \log(1/\delta))$.*

3. Structural Conditions for Approximation

In this section we prove several connections relating the L_1 distance of a low-degree PTF f to a bounded function g (i.e. $\mathbf{E}[|f(x) - g(x)|]$) and the maximum distance between the low-degree portions of the Fourier spectrum of f and g (i.e. $\|\hat{f}(B_d) - \hat{g}(B_d)\|_\infty$). A special case of such a connection was proved by Kalai et al. (2009b). Another special case, for linear threshold functions, was given by Birkendorf et al. (1998). Our version yields strong bounds for every PTF $f(x) = \mathsf{sign}(p(x))$ where polynomial $p(x)$ satisfies $|p(x)| \geq 1$ for all x and $p(x)$ is close to a low-degree polynomial $p'(x)$ of small $\|\cdot\|_1$ norm. In particular, it applies to any function representable as an integer-weight low-degree PTF of polynomial total weight and to any integer-weight linear threshold of terms (ANDs) of polynomial total weight (which includes polynomial size DNF expressions). We start by defining two simple and known measures of complexity of a degree-d PTF.

Definition 4 *For $\lambda > 0$, we say that a polynomial $p(x)$, λ-*sign-represents *a boolean function $f(x)$ if for all $x \in \{-1,1\}^n$, $f(x) = \mathsf{sign}(p(x))$ and $|p(x)| \geq \lambda$. For a degree-d PTF f, let $W_1^d(f)$ denote*

$$\min\{\|\hat{p}\|_1 \mid p \text{ 1-sign-represents } f\}.$$

The degree-d total integer weight *of f is*

$$TW^d(f) = \min\{\|\hat{p}\|_1 \mid \hat{p} \text{ is integer and } f = \mathsf{sign}(p)\}.$$

Remark 5 *We briefly remark that $W_1^d(f)$ is exactly the inverse of the* advantage *of a degree-d PTF defined by Krause and Pudlák (1997) as the largest λ for which there exists a polynomial $p(x)$ such that p λ-sign-represents f and $\|\hat{p}\|_1 = 1$). In addition, linear programming duality implies that the advantage of f equals α if and only if α is the smallest value such that for every distribution D over $\{-1,1\}^n$ there exists a monomial $\chi_a(x)$ of degree at most d such that $|\mathbf{E}_D[f(x) \cdot \chi_a(x)]| \geq \alpha$ (see Nisan's proof in (Impagliazzo, 1995)). Finally, clearly $W_1^d(f) \leq TW^d(f)$. The characterization of advantage using the LP duality together with the boosting algorithm by Freund (1995) imply that $TW^d(f) = O(n \cdot W_1^d(f)^2)$.*

We first prove a simpler special case of our bound when the representing polynomial $p(x)$ and the approximating polynomial $p'(x)$ are the same.

Lemma 6 *Let $p(x)$ be a degree-d polynomial that 1-sign-represents a PTF $f(x)$. For every $\mu \in (-1,1)^n$ and bounded function $g(x) : \{-1,1\}^n \to [-1,1]$,*

$$\mathbf{E}_\mu[|f(x) - g(x)|] \leq \|\hat{f}_\mu(B_d) - \hat{g}_\mu(B_d)\|_\infty \cdot \|\hat{p}_\mu(B_d)\|_1.$$

Proof First note that for every x, the values $f(x), f(x) - g(x)$ and $p(x)$ have the same sign. Therefore $\mathbf{E}_\mu[|f(x) - g(x)|] = \mathbf{E}_\mu[f(x)(f(x) - g(x))] \leq \mathbf{E}_\mu[p(x)(f(x) - g(x))]$. From here we immediately get that

$$\mathbf{E}_\mu[p(x)(f(x) - g(x))] = \sum_{a \in B_d} \hat{p}_\mu(a)\mathbf{E}_\mu[(f(x) - g(x))\phi_{\mu,a}(x)] = \sum_{a \in B_d} \hat{p}_\mu(a)(\hat{f}_\mu(a) - \hat{g}_\mu(a))$$
$$\leq \|\hat{f}_\mu(B_d) - \hat{g}_\mu(B_d)\|_\infty \cdot \|\hat{p}_\mu(B_d)\|_1 .$$

■

To apply our bound to functions which are close (but not equal) to a degree-d PTF we also give the following approximate version of Lemma 6.

Lemma 7 *Let $p(x)$ be a polynomial that 1-sign-represents a PTF $f(x)$ and let $p'(x)$ be any degree-d polynomial. For every $\mu \in (-1,1)^n$ and a bounded function $g(x) : \{-1,1\}^n \to [-1,1]$,*

$$\mathbf{E}_\mu[|f(x) - g(x)|] \leq \|\hat{f}_\mu(B_d) - \hat{g}_\mu(B_d)\|_\infty \cdot \|\widehat{p'}_\mu(B_d)\|_1 + 2\mathbf{E}_\mu[|p'(x) - p(x)|].$$

Proof Following the proof of Lemma 7, we get

$$\begin{aligned}\mathbf{E}_\mu[|f(x) - g(x)|] &\leq \mathbf{E}_\mu[p(x)(f(x) - g(x))] \\ &= \mathbf{E}_\mu[p'(x)(f(x) - g(x))] + \mathbf{E}_\mu[(p(x) - p'(x))(f(x) - g(x))] \\ &\leq \|\hat{f}_\mu(B_d) - \hat{g}_\mu(B_d)\|_\infty \cdot \|\widehat{p'}_\mu(B_d)\|_1 \ + \mathbf{E}_\mu[2|p'(x) - p(x)|].\end{aligned}$$

■

We now give bounds on such representations of DNF expressions. As a warm-up we start with the uniform distribution case which is implicit in (Kalai et al., 2009b).

Lemma 8 *For any s-term DNF f, $W_1^n(f) \leq 2s + 1$.*

Proof Let $t_1(x), t_2(x), \ldots, t_s(x)$ denote the $\{0,1\}$ versions of each of the terms of f. For each $i \in [s]$ let T_i denote the set of the indices of all the variables in the term t_i. Then, $t_i = \prod_{j \in T_i} \frac{1 \pm x_j}{2}$, where the sign of each variable x_j is determined by whether it is negated or not in t_i. As is well-known (e.g. Blum et al., 1994), this implies that $\|\hat{t_i}\|_1 = 1$. Now, let $p(x) = 2\sum_{i \in [s]} t_i(x) - 1$. It is easy to see that, $|p(x)| \geq 1$, $f(x) = \mathsf{sign}(p(x))$, $p(x)$ and

$$\|\hat{p}\|_1 \leq 2 \sum_{i \in [s]} \|\hat{t_i}\|_1 + 1 \leq 2s + 1 \ .$$

■

An immediate corollary of Lemma 6 and Lemma 8 is the following bound given by Kalai et al. (2009b).

Corollary 9 *Let f be an s-term DNF expression. For every bounded function $g(x)$, $\mathbf{E}[|f(x) - g(x)|] \leq (2s+1) \cdot \|\hat{f} - \hat{g}\|_\infty$.*

As can be seen from of Lemma 8, bounding $W_1^n(f)$ is based on bounding $\|\hat{t_i}\|_1$ for every term t_i of a DNF expression. Therefore we next prove a product distribution bound on $\|\hat{t_i}\|_1$.

Lemma 10 *Let $t(x)$ be a $\{0,1\}$ AND of d boolean literals, that is, for a set of d literals $T \subseteq \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n\}$, $t(x) = 1$ when all literals in T are set to 1 in x and 0 otherwise. For any constant $c \in (0,1]$ and $\mu \in [-1+c, 1-c]^n$,*

$$\|\hat{t}_\mu\|_1 = \|\hat{t}_\mu(B_d)\|_1 \leq (2-c)^{d/2}.$$

Proof Let S denote the set of all vectors in $\{0,1\}^n$ corresponding to subsets of T, that is

$$S = \{a \mid \forall i \in [n],\ (a_i = 0 \bigvee \{x_i, \bar{x}_i\} \cap T \neq \emptyset)\}.$$

Clearly, $\|\hat{t}_\mu\|_1 = \|\hat{t}_\mu(B_d)\|_1 = \|\hat{t}_\mu(S)\|_1$. In addition, by Parseval's identity

$$\|\hat{t}_\mu\|_2^2 = \mathbf{E}_\mu[t(x)^2] = \mathbf{Pr}_\mu[t(x) = 1] \leq (1 - c/2)^d \ .$$

Now, by the Cauchy-Schwartz inequality,

$$\|\hat{t}_\mu(S)\|_1 \leq 2^{d/2} \cdot \|\hat{t}_\mu\|_2 = 2^{d/2} \cdot (1 - c/2)^{d/2} = (2 - c)^{d/2} \ ,$$

giving us the desired bound. ■

We now use Lemmas 7 and 10 to give a bound for all product distributions.

Theorem 11 *Let $c \in (0, 1]$ be a constant, μ be a c-bounded distribution and $\epsilon > 0$. For an integer $s > 0$ let f be an s-term DNF. For $d = \lfloor \log(s/\epsilon)/\log(2/(2 - c)) \rfloor$ and every bounded function $g : \{-1, 1\}^n \to [-1, 1]$,*

$$\mathbf{E}_\mu[|f(x) - g(x)|] \leq (2 \cdot (2 - c)^{d/2} \cdot s + 1) \cdot \|\hat{f}_\mu(B_d) - \hat{g}_\mu(B_d)\|_\infty + 4\epsilon.$$

Proof As in the proof of Lemma 8, let $t_1(x), t_2(x), \ldots, t_s(x)$ denote the $\{0, 1\}$ versions of each of the terms of f and let $p(x) = 2\sum_{i\in[s]} t_i(x) - 1$ be a polynomial that 1-sign-represents f. Now let $M \subseteq [s]$ denote the set of indices of f's terms which have length $\geq d+1 \geq \log(s/\epsilon)/\log(2/(2 - c))$ and let $p'(x) = 2\sum_{i\notin M} t_i(x) - 1$. In other words, p' is p with contributions of long terms removed and, in particular, is a degree-d polynomial.

For each $i \in M$, $\mathbf{E}_\mu[t_i(x)] = \mathbf{Pr}_\mu[t_i(x) = 1] \leq (1 - c/2)^{d+1} \leq \epsilon/s$. This implies that

$$\mathbf{E}_\mu[|p'(x) - p(x)|] \leq \sum_{i\in M} \mathbf{E}_\mu[2|t_i(x)|] \leq 2\epsilon \ . \tag{1}$$

Using Lemma 10, we get

$$\|\widehat{p'}_\mu(B_d)\|_1 \leq 2\sum_{i\notin M} \|\hat{t_i}_\mu(B_d)\|_1 + 1 \leq 2 \cdot (2 - c)^{d/2} \cdot s + 1. \tag{2}$$

We can now apply Lemma 7 and equations (1, 2) to obtain

$$\begin{aligned}\mathbf{E}_\mu[|f(x) - g(x)|] &\leq \|\hat{f}_\mu(B_d) - \hat{g}_\mu(B_d)\|_\infty \cdot \|\widehat{p'}_\mu(B_d)\|_1 \ + 2\mathbf{E}_\mu[|p'(x) - p(x)|] \\ &\leq (2 \cdot (2 - c)^{d/2} \cdot s + 1) \cdot \|\hat{f}_\mu(B_d) - \hat{h}_\mu(B_d)\|_\infty + 4\epsilon.\end{aligned}$$

■

It is easy to see that Theorem 11 generalizes to any function that can be expressed as low-weight linear threshold of terms. Specifically, we prove the following generalization (the proof appears in the full version).

Theorem 12 *Let $c \in (0, 1]$ be a constant, μ be a c-bounded distribution and $\epsilon > 0$. For an integer $s > 0$ let $f = h(u_1, u_2, \ldots, u_s)$, where h is an LTF over $\{-1, 1\}^s$ and u_i's are terms. For $d = \lfloor \log(W_1^1(h)/\epsilon)/\log(2/(2 - c)) \rfloor$ and every bounded function $g : \{-1, 1\}^n \to [-1, 1]$,*

$$\mathbf{E}_\mu[|f(x) - g(x)|] \leq (2 \cdot (2 - c)^{d/2} + 1) \cdot W_1^1(h) \cdot \|\hat{f}_\mu(B_d) - \hat{g}_\mu(B_d)\|_\infty + 4\epsilon.$$

For $c = 1$, $(2 - c)^{d/2} = 1$ and for $c \in (0, 1)$, $(2 - c)^{d/2} \leq (W_1^1(h)/\epsilon)^{(1/\log(2/(2-c)) - 1)/2}$.

4. Construction of a Fourier Spectrum Approximating Function

As follows from Corollary 9 (and Th. 11), to ϵ-approximate a DNF expression over a product distribution, it is sufficient to find a bounded function g such that g has approximately the same Fourier spectrum as f. In this section we show how this can be done by giving an algorithm which constructs a function with the desired Fourier spectrum or the low-degree part thereof.

Our algorithm is based on the following idea: given a bounded function g such that for some a, $|\hat{f}(a) - \hat{g}(a)| \geq \gamma$ we show how to obtain a bounded function g_1 which is closer in L_2 distance squared to f than g. Parseval's identity states that $\mathbf{E}[(f-g)^2] = \sum_b (\hat{f}(b) - \hat{g}(b))^2$. Therefore to improve the distance to f we do the simplest imaginable update: define $g' = g + (\hat{f}(a) - \hat{g}(a))\chi_a$. In other words g' is the same as g but with a's Fourier coefficient set to $\hat{f}(a)$. Clearly,

$$\mathbf{E}[(f-g')^2] = \sum_{b \neq a} (\hat{f}(b) - \hat{g}(b))^2 = \mathbf{E}[(f-g)^2] - (\hat{f}(a) - \hat{g}(a))^2 \leq \mathbf{E}[(f-g)^2] - \gamma^2.$$

The only problem with this approach is that g' is not necessarily a function with values bounded in $[-1,1]$. However, following the idea from (Feldman, 2009), we can we convert g' to a bounded function g_1 by cutting-off all values outside of $[-1,1]$ (which is achieved by applying the projection function P_1). The target function f is boolean and therefore this step can only decrease the L_2 distance squared to f. This simple argument implies that starting with $g \equiv 0$ we can update it iteratively until we reach a bounded function g_t such that for all a, $|\hat{f}(a) - \hat{g}(a)| \leq \gamma$. The decrease in the L_2 distance squared at every step implies that the total number of steps cannot exceed $1/\gamma^2$. Also note that for running this algorithm the only thing we need are (the approximate values of) the Fourier coefficients of f.

We now state and prove the claim formally. The input to our algorithm is a vector $\tilde{f}(B_d) \in \mathbb{R}^{B_d}$ of desired coefficients up to degree d given to some accuracy γ. Further, in our applications we will only use vectors with at most $O(1/\gamma^2)$ non-zero coefficients since for every Boolean function at most $1/\gamma^2$ of its Fourier coefficients are of magnitude greater than γ and smaller coefficients are approximated by 0.

Theorem 13 *There exists a randomized algorithm* `PTFapprox` *that for every boolean function* $f : \{-1,1\}^n \to \{-1,1\}$*, given* $\gamma > 0, \delta > 0$ *a degree bound* d *and a succinctly-represented vector of coefficients* $\tilde{f}(B_d) \in \mathbb{R}^{B_d}$ *such that* $\|\hat{f}(B_d) - \tilde{f}(B_d)\|_\infty \leq \gamma$ *and* $\|\tilde{f}(B_d)\|_0 = O(1/\gamma^2)$*, with probability at least* $1-\delta$*, outputs a bounded function* $g : \{-1,1\}^n \to [-1,1]$ *such that* $\|\hat{f}(B_d) - \hat{g}(B_d)\|_\infty \leq 5\gamma$*. The algorithm runs in time polynomial in* n*,* $1/\gamma$ *and* $\log(1/\delta)$*.*

Proof We build g via the following iterative process. Let $g_0 \equiv 0$. At step t, given g_t, we run the KM algorithm (Th. 1) to compute all the Fourier coefficients of g_t which are of degree at most d to accuracy $\gamma/2$. Let $\widetilde{g_t}(B_d) \in \mathbb{R}^{B_d}$ denote the vector of estimates output by the algorithm. By Theorem 1, there are at most $16/\gamma^2$ non-zero coefficients in $\widetilde{g_t}(B_d)$. For now let's assume that the output of the KM is always correct; we will deal with the confidence bounds later in the standard manner.

If $\|\widetilde{g_t}(B_d) - \tilde{f}(B_d)\|_\infty \leq 7\gamma/2$, then we stop and output g_t. By triangle inequality,

$$\begin{aligned}\|\hat{f}(B_d) - \widehat{g_t}(B_d)\|_\infty &\leq \|\hat{f}(B_d) - \tilde{f}(B_d)\|_\infty + \|\tilde{f}(B_d) - \widetilde{g_t}(B_d)\|_\infty + \|\widetilde{g_t}(B_d) - \widehat{g_t}(B_d)\|_\infty \\ &\leq \gamma + 7\gamma/2 + \gamma/2 = 5\gamma\ ,\end{aligned}$$

in other words g_t satisfies the claimed condition.

Otherwise, there exists $a \in B_d$ such that $|\widetilde{g}_t(a) - \tilde{f}(a)| > 7\gamma/2$. We note that using the succinct representation of $\tilde{f}(B_d)$ and $\widehat{g}_t(B_d)$ such a can be found in $O(n(\|\widetilde{g}_t\|_0 + \|\tilde{f}\|_0)) = O(n/\gamma^2)$ time. First observe that, by triangle inequality,

$$|\widehat{g}_t(a) - \hat{f}(a)| \geq |\widetilde{g}_t(a) - \tilde{f}(a)| - |\tilde{f}(a) - \hat{f}(a)| - |\widehat{g}_t(a) - \widetilde{g}_t(a)| \leq 7\gamma/2 - \gamma - \gamma/2 = 2\gamma.$$

Let $g'_{t+1} = g_t + (\tilde{f}(a) - \widetilde{g}_t(a))\chi_a$. The Fourier spectrums of g_t and g'_{t+1} differ only on a. Therefore, by using Parseval's identity, we obtain that

$$\begin{aligned} \mathbf{E}[(f - g_t)^2] - \mathbf{E}[(f - g'_{t+1})^2] &= (\hat{f}(a) - \widehat{g}_t(a))^2 - (\hat{f}(a) - \tilde{f}(a) + \widetilde{g}_t(a) - \hat{g}(a))^2 \\ &\geq (2\gamma)^2 - (3\gamma/2)^2 = 7\gamma^2/4\,. \end{aligned} \quad (3)$$

Now let $g_{t+1} = P_1(g_t)$. For every x, $(f(x) - g_{t+1}(x))^2 \leq (f(x) - g'_{t+1}(x))^2$. Together with equation (3) this implies that $\mathbf{E}[(f - g_{t+1})^2] \leq \mathbf{E}[(f - g_t)^2] - 7\gamma^2/4$. At step 0 we have $\mathbf{E}[(f - g_0)^2] = 1$ and therefore the process will terminate after at most $4/(7\gamma^2)$ steps.

We note that in order to make sure that the success probability is at leat $1 - \delta$ it is sufficient to run the KM algorithm with confidence parameter $4\delta/(7\gamma^2)$. At step t evaluating g_t on any point x takes $O(t \cdot n)$ time and therefore each invocation of the KM algorithm takes $\tilde{O}(n^2 \cdot \gamma^{-8} \cdot \log(1/\delta))$ time. Overall this implies that the running time of PTFapprox is $\tilde{O}(n^2 \cdot \gamma^{-10} \cdot \log(1/\delta))$. ■

A simple observation about PTFapprox is that it does not rely on the update step being a multiple of a boolean function. Therefore it would work verbatim for any orthonormal basis and not only parities. Therefore, by using the EKM algorithm in place of KM we can easily extend our algorithm to any product distribution.

Theorem 14 *There exists a randomized algorithm* PTFapproxProd *that for every $\mu \in (-1,1)^n$, boolean function $f : \{-1,1\}^n \to \{-1,1\}$, given $\mu, \gamma > 0, \delta > 0$, a degree bound d and a succinctly-represented vector of coefficients $\tilde{f}_\mu(B_d) \in \mathbb{R}^{B_d}$ such that $\|\hat{f}_\mu(B_d) - \tilde{f}_\mu(B_d)\|_\infty \leq \gamma$ and $\|\tilde{f}_\mu(B_d)\|_0 = O(1/\gamma^2)$, with probability at least $1 - \delta$, outputs a function $g : \{-1,1\}^n \to [-1,1]$ such that $\|\hat{f}_\mu(B_d) - \hat{g}_\mu(B_d)\|_\infty \leq 5\gamma$. The algorithm runs in time polynomial in n, $1/\gamma$ and $\log(1/\delta)$.*

One disadvantage of this construction is that g output by PTFapprox is not a PTF itself. The reason for this is that the projection operation P_1 is applied after every update. We now show that instead of applying the projection step after every update it is sufficient to apply the projection once to all the updates. This idea is based on Impagliazzo's (1995) argument in the context of hardcore set construction, and is also the basis for the algorithm of Trevisan et al. (2009). Impagliazzo's proof uses the same L_2 squared potential function but requires an additional point-wise counting argument to prove that the potential can be used to bound the number of steps. Instead, we augment the potential function in a way that captures the additional counting argument and generalized to non-boolean functions (necessary for the product distribution case). As a result the algorithm will output a function of the form $P_1(\sum_{a \in B_d} \alpha_a \chi_a)$ which is then converted to a PTF by applying the sign function. The modified proof also allows us to easily derive a bound on the total integer weight of the resulting PTF. Further details of the proper version of of Theorem 14 are given in Appendix B.

5. Applications

In Appendix A we give several application of our approximating algorithms to the problem of learning DNF expressions in several models of learning. Our first application is a new algorithm for learning DNF expressions using membership queries over any product distribution. In the second application we show a simple algorithm for learning DNF expressions from random examples coming from a smoothed product distribution. In the third application we give a new and faster algorithm for learning MDNF over product distributions (from random examples alone). We describe all the applications for (M)DNF expressions. However, by using the more general Theorem 12 in place of Theorem 11, we immediately get that our algorithms can be also used to learn a broader set of concept classes which includes, for examples, (monotone) majorities of terms. Previous algorithms for the second and third applications rely strongly on the term-combining function being an OR.

Acknowledgments

I thank Sasha Sherstov for pointing out the connection of our $W_1^d(f)$ measure of a PTF f to the definition of advantage by Krause and Pudlák (1997).

References

H. Aizenstein and L. Pitt. On the learnability of disjunctive normal form formulas. *Machine Learning*, 19(3):183–208, 1995.

M. Bellare. The spectral norm of finite functions. Technical Report TR-495, MIT, 1991.

A. Birkendorf, E. Dichterman, J. Jackson, N. Klasner, and H.-U. Simon. On restricted-focus-of-attention learnability of boolean functions. *Machine Learning*, 30(1):89–123, 1998.

A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of STOC*, pages 253–262, 1994.

N. Bshouty and V. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research*, 2, 2002.

N. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996.

N. Bshouty, J. Jackson, and C. Tamon. More efficient PAC-learning of DNF with membership queries under the uniform distribution. *Journal of Computer and System Sciences*, 68(1):205–234, 2004.

N. Bshouty, E. Mossel, R. O'Donnell, and R. Servedio. Learning DNF from random walks. *Journal of Computer and System Sciences*, 71(3):250–265, 2005.

A. De, I. Diakonikolas, V. Feldman, and R. Servedio. Nearly optimal solutions for the Chow Parameters Problem and low-weight approximation of halfspaces. Manuscript, to appear in STOC 2012, 2012.

V. Feldman. Attribute efficient and non-adaptive learning of parities and DNF expressions. *Journal of Machine Learning Research*, (8):1431–1460, 2007.

V. Feldman. A complete characterization of statistical query learning with applications to evolvability. In *Proceedings of FOCS*, pages 375–384, 2009.

V. Feldman. Learning DNF expressions from Fourier spectrum. *CoRR*, abs/1203.0594, 2012.

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

M. Furst, J. Jackson, and S. Smith. Improved learning of AC^0 functions. In *Proceedings of COLT*, pages 317–325, 1991.

O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of STOC*, pages 25–32, 1989.

P. Gopalan, A. Kalai, and A. Klivans. Agnostically learning decision trees. In *Proceedings of STOC*, pages 527–536, 2008.

R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of FOCS*, pages 538–545, 1995.

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

J. Jackson, H. Lee, R. Servedio, and A. Wan. Learning random monotone DNF. *Discrete Applied Mathematics*, 159(5):259–271, 2011.

J. Kahn, G. Kalai, and N. Linial. The influence of variables on Boolean functions. In *Proceedings of FOCS*, pages 68–80, 1988.

A. Kalai, V. Kanade, and Y. Mansour. Reliable agnostic learning. In *Proceedings of COLT*, 2009a.

A. Kalai, A. Samorodnitsky, and S.-H. Teng. Learning and smoothed analysis. In *Proceedings of FOCS*, pages 395–404, 2009b.

M. Kearns, M. Li, L. Pitt, and L. Valiant. On the learnability of Boolean formulae. In *Proceedings of STOC*, pages 285–295, 1987.

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

A. Klivans and R. Servedio. Learning DNF in time $2^{\tilde{\mathrm{O}}(\mathrm{n}^{1/3})}$. *Journal of Computer and System Sciences*, 68(2):303–318, 2004.

M. Krause and P. Pudlák. On the computational power of depth-2 circuits with threshold and modulo gates. *Theor. Comput. Sci.*, 174(1-2):137–156, 1997.

E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

Y. Mansour. An $O(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution. *Journal of Computer and System Sciences*, 50:543–550, 1995.

R. O'Donnell and R. Servedio. The chow parameters problem. *SIAM Journal on Computing*, 40 (1):165–199, 2011.

Y. Sakai and A. Maruoka. Learning monotone log-term DNF formulas under the uniform distribution. *Theory of Computing Systems*, 33:17–33, 2000.

L. Sellie. Exact learning of random DNF over the uniform distribution. In *Proceedings of STOC*, pages 45–54, 2009.

R. Servedio. On learning monotone DNF under product distributions. *Information and Computation*, 193(1):57–74, 2004.

D. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of ACM*, 51(3):385–463, 2004.

L. Trevisan, M. Tulsiani, and S. Vadhan. Regularity, boosting, and efficiently simulating every high-entropy distribution. In *Proceeding of IEEE Conference on Computational Complexity*, pages 126–136, 2009.

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

K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of COLT*, pages 314–326, 1990.

Appendix A. Applications to Learning DNF Expressions

A.1. Learning with Membership Queries

An immediate application of Theorem 14 together with the bound in Theorem 11 and the EKM algorithm (Th. 2) is a simple algorithm for learning DNF over any constant-bounded product distribution.

Corollary 15 *Let $c \in (0, 1]$ be a constant. There exists a membership query algorithm* `DNFLearnMQProd` *that for every c-bounded μ, efficiently PAC learns DNF expressions over D_μ.*

Proof Let $\epsilon' = \epsilon/9$ and, as defined in Th. 11, let $d = \lfloor \log(s/\epsilon')/\log(2/(2-c)) \rfloor$ and

$$\gamma = \epsilon'/(2(2-c)^{d/2}s + 1) = \Omega\left((\epsilon/s)^{(1/\log(2/(2-c))+1)/2}\right).$$

`DNFLearnMQProd` consists of two phases:

1. **Collect γ-approximations to all degree-d μ-Fourier coefficients**. In this step we run the EKM algorithm for f with parameters, $\theta = \gamma$, $\delta = 1/4$ and μ to obtain a succinctly-represented $\tilde{f}_\mu(B_d)$ such that $\|\tilde{f}_\mu(B_d) - \tilde{g}_\mu(B_d)\|_\infty \le \gamma$ (EKM returns the complete $\tilde{f}_\mu$ but we discard coefficients with degree higher than d).

2. **Construct a bounded g with the given μ-Fourier spectrum**. In this step we run `PTFapproxProd` on $\tilde{f}_\mu(B_d)$ with parameters d, γ, μ and $\delta = 1/4$ to construct a bounded function g such that $\|\hat{f}_\mu(B_d) - \hat{g}_\mu(B_d)\|_\infty \leq 5\gamma = 5\epsilon'/(2(2-c)^{d/2}s+1)$. Note that this step requires no access to membership queries or random examples of f.

We return $\mathsf{sign}(g(x))$ as our hypothesis. Overall, if both steps are successful (which happens with probability at least $1/2$) then, according to Theorem 11,

$$\mathbf{E}_\mu[|f-g|] \leq \|\hat{f}_\mu(B_d)-\hat{g}_\mu(B_d)\|_\infty\cdot(2(2-c)^{d/2}s+1)+4\epsilon' = 5\gamma\cdot(2(2-c)^{d/2}s+1)+4\epsilon' = 9\epsilon' = \epsilon.$$

This implies $\mathbf{Pr}_\mu[f \neq \mathsf{sign}(g)] \leq \mathbf{E}_\mu[|f-g|] \leq \epsilon$.

The running time of both phases of `DNFLearnMQProd` is polynomial in n, and $1/\gamma$, which for any constant $c \in (0, 1]$, is polynomial in $n \cdot s/\epsilon$. ■

As noted in the proof, the only part of our algorithm that uses membership queries is the phase that collects Fourier coefficients of logarithmic degree. This step can also be performed using weaker forms of access to the target function, such as extended statistical queries of Bshouty and Feldman (2002) or examples coming from a random walk on a hypercube Bshouty et al. (2005). Hence our algorithm can be adapted to those models in a straightforward way.

A.2. Smoothed Analysis of Learning DNF over Product Distributions

We now describe how `PTFapproxProd` can be used in the context of smoothed analysis of learning DNF over product distributions introduced by Kalai et al. (2009b). We start with a brief description of the model.

A.2.1. LEARNING FROM SMOOTHED PRODUCT DISTRIBUTIONS

Motivated by the seminal model of smoothed analysis by Spielman and Teng (2004), Kalai et al. (2009b) defined learning a concept class C with respect to smoothed product distributions as follows. The model measures the complexity of a learning algorithm with respect to a product distribution D_μ where μ is "perturbed" randomly. More formally, μ is chosen uniformly at random from a cube $\bar{\mu} + [-c, c]^n$ for a $2c$-bounded $\bar{\mu}$. A learning algorithm in this model must, for every $\bar{\mu}$ and $f \in C$, PAC learn f over D_μ with high probability over the choice of μ.

Definition 16 (Kalai et al. 2009b) *Let C be a concept class. An algorithm $\mathcal{A}$ is said to learn C over smoothed product distributions if for every constant $c \in (0, 1/2]$, $f \in C$, $\epsilon, \delta > 0$, and any $2c$-bounded $\bar{\mu}$, given access to $EX(f, D_\mu)$ for a randomly and uniformly chosen $\mu \in \bar{\mu} + [-c, c]^n$, with probability at least $1-\delta$, $\mathcal{A}$ outputs a hypothesis h, ϵ-close to f relative to D_μ. The probability here is taken with respect to the random choice of μ, choice of random samples from D_μ and any internal randomization of $\mathcal{A}$. $\mathcal{A}$ is said to learn efficiently if its running time is upper-bounded by a polynomial in $n/(\epsilon\cdot\delta)$ (and the size s of f if C is parameterized) where the degree of the polynomial is allowed to depend on c.*

Feature Finding Algorithm. A key insight in the results of Kalai et al. (2009b) is that if a bounded function f has a low-degree significant $\bar{\mu}$-Fourier coefficient $\hat{f}_{\bar{\mu}}(a)$, then after the perturbation f will have significant μ-Fourier coefficients for all $b \leq a$ (here $b \leq a$ means $b_i \leq a_i$ for all $i \in [n]$). This insight leads to a simple method for finding all the significant μ-Fourier coefficients of degree d in time polynomial in 2^d instead of n^d required by the Low Degree algorithm.

Theorem 17 (Greedy Feature Construction (GFC)(Kalai et al., 2009b)) *Let $c \in (0, 1/2]$ be a constant. There exists an algorithm that for every $f : \{-1,1\}^n \to [-1,1]$, $d \in [n]$, $\theta, \delta > 0$, $2c$-bounded $\bar{\mu}$, given access to $EX(f, D_\mu)$ for a randomly and uniformly chosen $\mu \in \bar{\mu} + [-c, c]^n$, with probability at least $1-\delta$, outputs a succinctly-represented vector $\tilde{f}(B_d)$ such that $\|\hat{f}_\mu(B_d) - \tilde{f}_\mu(B_d)\|_\infty \leq \theta$ and $\|\tilde{f}_\mu(B_d)\|_0 \leq 4/\theta^2$. The algorithm runs in time $O((n \cdot 2^d/(\theta \cdot \delta))^{k(c)})$ for some constant $k(c)$ which depends only on c.*

A.2.2. APPLICATION OF PTFapproxProd

The Greedy Feature Construction algorithm gives an efficient algorithm for collecting μ-Fourier coefficients of logarithmic degree. The application of PTFapproxProd in this setting is now straightforward. All that needs to be done is to replace the EKM algorithm in the coefficient collection phase of DNFLearnMQProd (Cor. 15) with the GFC algorithm. The coefficient collection phase of DNFLearnMQProd requires only coefficients of logarithmic degree in the learning parameters and therefore the resulting combination runs in polynomial time (the approximator construction phase is unchanged and still uses the EKM algorithm). Thereby we obtain a new simple proof of the following theorem from (Kalai et al., 2009b).

Theorem 18 (Kalai et al. 2009b) *DNF expressions are PAC learnable efficiently over smoothed product distributions.*

A.3. Learning Monotone DNF

We now describe our algorithm for learning monotone s-term DNF from random examples alone. For simplicity, we describe it for the uniform distribution, but all the ingredients that we use have their product distribution versions and hence the generalization is straightforward (we describe it in Appendix **??**). As pointed out earlier, our algorithm is based on Servedio's algorithm for learning monotone DNF (Servedio, 2004). The main idea of his algorithm is to restrict learning to influential variables alone (which for a monotone function can be efficiently identified) and then run the Low Degree algorithm 3 to approximate all the Fourier coefficients of low degree on influential variables. The sign of the resulting low-degree polynomial $p(x)$ is then used as a hypothesis. The degree that is known to be sufficient for such approximation to work was derived using a Fourier concentration bound by Mansour (1995) and Linial et al. (1993) and equals $20 \cdot \log(s/\epsilon) \cdot \log(1/\epsilon)$.

In our algorithm, instead of just taking the sign of $p(x)$ as the hypothesis, we use PTFapprox to produce a bounded function with the same Fourier coefficients as $p(x)$. The advantage of this approach is that the degree bound required to achieve ϵ-accuracy using our approach is reduced to $\log(s/\epsilon) + O(1)$ (and is also significantly easier to prove than the Switching Lemma-based bound of Mansour (1995)). Further, the accuracy estimation in our algorithm does not depend on n the number of sufficiently influential variables does not depend on n. As a consequence our algorithm is attribute-efficient.

Following Servedio (2004), we rely on a well-known connection between the influence of a variable and Fourier coefficients that include that variable. Formally, for a function $f : \{-1,1\}^n \to \{-1,1\}$ and $i \in [n]$ let $f_{i,1}(x)$ and $f_{i,-1}(x)$ denote $f(x)$ with bit i of the input set to 1 and -1, respectively. The influence of variable i over distribution D is defined as $I_{D,i}(f) = \mathbf{Pr}_D[f_{i,1}(x) \neq f_{i,-1}(x)]$. We use $I_i(f)$ to denote the influence over the uniform distribution. Let $S_i = \{a \in \{0,1\}^n \mid a_i = 1\}$. Kahn et al. (1988) have shown that for every $i \in [n]$,

$$I_i(f) = \sum_{a \in S_i} \hat{f}(a)^2 = \|\hat{f}(S_i)\|_2^2. \tag{4}$$

The crucial use of monotonicity is that for any monotone f, $I_{D,i}(f) = (\mathbf{E}_D[f_{i,1}(x)] - \mathbf{E}_D[f_{i,-1}(x)])/2$ and hence one can estimate $\|\hat{f}(S_i)\|_2^2$ using random uniform examples of f. We now describe our algorithm for learning monotone DNF over the uniform distribution more formally.

Theorem 19 *There exists an algorithm that PAC learns s-term monotone DNF expressions over the uniform distribution to accuracy ϵ in time $\tilde{O}(n \cdot (s \cdot \log{(s/\epsilon)})^{O(\log{(s/\epsilon)})})$.*

Proof Our algorithm is based on the same two phases as `DNFLearnMQProd` in Corollary 15. Hence we set $\epsilon' = \epsilon/9$, $d = \lfloor \log{(s/\epsilon')} \rfloor$ and $\gamma = \epsilon'/(2s+1)$.

The goal of the first phase of the algorithm is to collect γ-approximations to degree-d Fourier coefficients of f. We do this by first finding the influential variables and then using a low-degree algorithm restricted to the influential variables.

Using equation (4), we can conclude that if for some variable i, $I_i(f) = \|\hat{f}(S_i)\|_2^2 \leq \gamma^2$, then there are no Fourier coefficients of f, that include variable i and are greater in their magnitude than γ. We can therefore eliminate variable i, that is approximate all of Fourier coefficients in S_i by 0. Also, as we mentioned before, $I_i(f)$ can be estimated from random examples of f. We will use an estimate to accuracy $\gamma^2/3$ and exclude variable i if the estimate is lower than $2\gamma^2/3$ (the straightforward details of the required confidence bounds appear in the more detailed and general proof of Theorem 20).

We argue that this process will eliminate all but at most $s \cdot \log{(3s/\gamma^2)}$ variables. This, follows from the fact that if a variable i appears only in terms of length greater than $\log{(3s/\gamma^2)}$ then it cannot be influential enough to survive the elimination condition. Over the uniform distribution, each term of length greater than $\log{(3s/\gamma^2)}$ equals 1 with probability at most $\gamma^2/(3s)$. The value $f_{i,1}(x)$ differs from $f_{i,-1}(x)$ only if x is accepted by a term that includes variable i. There are at most s terms and therefore (for a variable i that appears only in terms of length $\log{(3s/\gamma^2)}$)

$$(\mathbf{E}[f_{i,1}(x)] - \mathbf{E}[f_{i,-1}(x)])/2 < s \cdot \gamma^2/(3s) = \gamma^2/3.$$

Consequently, the influence of such variable i cannot be within $\gamma^2/3$ of $3\gamma^2/3$ (required to survive the elimination). Therefore at the end of the first step we will end up with variables only from terms of length at most $\log{(3s/\gamma^2)}$. Hence there will be at most $s \cdot \log{(3s/\gamma^2)}$ variables left. Let M denote the set of the remaining (influential) variables.

In the second step of this phase we run the low-degree algorithm for degree d and $\theta = \gamma = \epsilon'/(2s+1)$ restricted to the variables in M, and let $\tilde{f}(B_d)$ be the resulting vector of approximate Fourier coefficients (the coefficients with variables outside of M are 0). By Theorem 3 and the property of our influential variables $\|\hat{f}(B_d) - \tilde{f}(B_d)\|_\infty \leq \gamma$.

We can now construct an approximating function in the same way as we did in `DNFLearnMQProd` (Cor. 15). Namely, in the third step of the algorithm we run `PTFapprox` on $\tilde{f}(B_d)$ to obtain a bounded function g such that $\|\hat{f}(B_d) - \hat{g}(B_d)\|_\infty \leq 5\gamma = 5\epsilon'/(2s+1)$. Then, by Theorem 11,

$$\mathbf{E}[|f-g|] \leq (2s+1)\|\hat{f}(B_d) - \hat{g}(B_d)\|_\infty + 4\epsilon' \leq (2s+1) \cdot 5\epsilon'/(2s+1) + 4\epsilon' = 9\epsilon' = \epsilon.$$

Hence $\mathbf{Pr}[\mathsf{sign}(g) \neq f] \leq \epsilon$.

To analyze the running time of our algorithm we note that both the first and the third steps can be done in $\tilde{O}(n)\cdot \mathrm{poly}(s/\epsilon)$ time. According to Theorem 3, the second step can be done in $n\cdot |M|^d \cdot \mathrm{poly}(|M|/\gamma) = n\cdot (s\cdot \log{(s/\epsilon)})^{O(\log{(s/\epsilon)})}$ time steps. Altogether, we obtain the claimed bound on the running time. ■

A corollary of our running time bound is that for s and ϵ such that $s/\epsilon = 2^{\sqrt{\log n}}$, s-term monotone DNF are learnable to accuracy ϵ in polynomial time. Servedio's algorithm is only guaranteed to efficiently learn $2^{\sqrt{\log n}}$-term MDNF to constant accuracy.

We remark that the bound on running time can be simplified for monotone s-term k-DNF expressions. Specifically, we will obtain an algorithm running in $(s\cdot k)^{O(k)}\cdot (n/\epsilon)^{O(1)}$ time. This algorithm can be used to obtain fully-polynomial learning algorithms for monotone $2^{\sqrt{\log n}}$-term $\sqrt{\log n}$-DNF and other subclasses of MDNF expressions for which no fully-polynomial learning algorithms were known.

The following straightforward generalization of our learning algorithm to product distributions is proved in the full version.

Theorem 20 *For any constant $c \in (0,1]$ there exists an algorithm* `MDNFLearnProd` *that PAC learns s-term monotone DNF expressions over all c-bounded product distributions to accuracy ϵ in time $\tilde{O}(n\cdot (s\cdot \log{(s/\epsilon)})^{O(\log{(s/\epsilon)})})$.*

Appendix B. A Proper PTF Construction Algorithm

Theorem 21 *There exists a randomized algorithm* `PTFconstructProd` *that for every $\mu \in (-1,1)^n$, boolean function $f : \{-1,1\}^n \to \{-1,1\}$, given $\mu, \gamma > 0, \delta > 0$, a degree bound d and a succinctly-represented vector of coefficients $\tilde{f}_\mu(B_d) \in \mathbb{R}^{B_d}$ such that $\|\hat{f}_\mu(B_d) - \tilde{f}_\mu(B_d)\|_\infty \le \gamma$ and $\|\tilde{f}_\mu(B_d)\|_0 = O(1/\gamma^2)$, with probability at least $1-\delta$, outputs a bounded function $g : \{-1,1\}^n \to [-1,1]$ such that $\|\hat{f}_\mu(B_d) - \hat{g}_\mu(B_d)\|_\infty \le 5\gamma$. The algorithm runs in time polynomial in n, $1/\gamma$ and $\log{(1/\delta)}$. In addition, $g(x) = P_1(g'(x))$ for a degree-d polynomial such that $\widehat{g'}_\mu = \gamma\cdot \hat{p}_\mu$ where $\hat{p}_\mu$ is a vector of integers and $\|\hat{p}_\mu\|_1 \le 1/(2\gamma^2)$.*

Proof As in the proof of Theorem 13, we build g via an iterative process starting from $g'_0 \equiv 0$ and $g_0 = P_1(g'_0)$. We use the EKM algorithm (Th. 2) to compute $\widetilde{g_t}_\mu(B_d)$ and stop and return g_t if $\|\widetilde{g_t}_\mu(B_d) - \tilde{f}_\mu(B_d)\|_\infty \le 7\gamma/2$. Otherwise (there exists $a \in B_d$ such that $|\widetilde{g_t}_\mu(a) - \tilde{f}_\mu(a)| > 7\gamma/2$ and $|\widehat{g_t}_\mu(a) - \hat{f}_\mu(a)| > 2\gamma$), we let $\gamma' = \gamma\cdot \mathsf{sign}(\tilde{f}_\mu(a) - \widetilde{g_t}_\mu(a))$, $g'_{t+1} = g'_t + \gamma'\chi_{a,\mu}$ and $g_{t+1} = P_1(g'_{t+1})$.

We prove a bound on the total number of steps using the following potential function:

$$E(t) = \mathbf{E}_\mu[(f-g_t)^2] + 2\mathbf{E}_\mu[(f-g_t)(g_t - g'_t)] = \mathbf{E}_\mu[(f-g_t)(f-2g'_t+g_t)].$$

The key claim of this proof is that $E(t) - E(t+1) \ge \gamma^2$. First,

$$\begin{aligned} E(t) - E(t+1) &= \mathbf{E}_\mu[(f-g_t)(f-2g'_t+g_t)] - \mathbf{E}_\mu[(f-g_{t+1})(f-2g'_{t+1}+g_{t+1})] \\ &= \mathbf{E}_\mu\left[(f-g_t)(2g'_{t+1}-2g'_t) - (g_{t+1}-g_t)(2g'_{t+1}-g_t-g_{t+1})\right] \\ &= \mathbf{E}_\mu[2(f-g_t)\gamma'\chi_{a,\mu}] - \mathbf{E}_\mu\left[(g_{t+1}-g_t)(2g'_{t+1}-g_t-g_{t+1})\right] \qquad (5) \end{aligned}$$

We observe that $\mathbf{E}_\mu[2(f - g_t)\gamma'\chi_{a,\mu}] = 2\gamma'(\hat{f}_\mu(a) - \widehat{g_t}_\mu(a))$ and that $\mathsf{sign}(\hat{f}_\mu(a) - \widehat{g_t}_\mu(a)) =$ $\mathsf{sign}(\tilde{f}_\mu(a) - \widetilde{g_t}_\mu(a))$. Therefore, we get

$$\mathbf{E}_\mu[2(f - g_t)\gamma'\chi_a] \geq 2\gamma|\hat{g}_{t,\mu}(a) - \hat{f}_\mu(a)| \geq 4\gamma^2 \ . \tag{6}$$

To upper-bound the expression $\mathbf{E}_\mu\left[(g_{t+1} - g_t)(2g'_{t+1} - g_t - g_{t+1})\right]$ we prove that for every point $x \in \{-1,1\}^n$,

$$(g_{t+1}(x) - g_t(x))(2g'_{t+1}(x) - g_t(x) - g_{t+1}(x)) \leq 2\gamma^2\chi_{a,\mu}(x)^2.$$

We first observe that $|g_{t+1}(x) - g_t(x)| = |P_1(g'_t(x) + \gamma'\chi_{a,\mu}(x)) - P_1(g'_t(x))| \leq |\gamma'\chi_{a,\mu}(x)| =$ $|\gamma\chi_{a,\mu}(x)|$ (a projection operation does not increase the distance). Now

$$|2g'_{t+1}(x) - g_t(x) - g_{t+1}(x)| \leq |g'_{t+1}(x) - g_t(x)| + |(g'_{t+1}(x) - g_{t+1}(x)|.$$

The first part $|g'_{t+1}(x) - g_t(x)| = |\gamma'\chi_{a,\mu}(x) + g'_t(x) - g_t(x)| \leq |\gamma'\chi_{a,\mu}(x)|$ unless $g'_t(x) - g_t(x) \neq 0$ and $g'_t(x) - g_t(x)$ has the same sign as $\gamma'\chi_{a,\mu}(x)$. However, in this case $g_{t+1}(x) = g_t(x)$ and as a result $(g_{t+1}(x) - g_t(x))(2g'_{t+1}(x) - g_t(x) - g_{t+1}(x)) = 0$. Similarly, $|g'_{t+1}(x) - g_{t+1}(x)| \leq$ $|\gamma'\chi_{a,\mu}(x)|$ unless $g_{t+1}(x) = g_t(x)$. Altogether we obtain that

$$(g_{t+1}(x) - g_t(x))(2g'_{t+1}(x) - g_t(x) - g_{t+1}(x)) \leq \max\{0, |\gamma\chi_{a,\mu}(x)|(|\gamma'\chi_{a,\mu}(x)| + |\gamma'\chi_{a,\mu}(x)|)\} = 2\gamma^2\chi_{a,\mu}(x)^2.$$

This implies that

$$\mathbf{E}_\mu\left[(g_{t+1} - g_t)(2g'_{t+1} - g_t - g_{t+1})\right] \leq 2\gamma^2\mathbf{E}_\mu[\chi_{a,\mu}(x)^2] = 2\gamma^2. \tag{7}$$

By substituting equations (6) and (7) into equation (5), we obtain the claimed decrease in the potential function

$$E(t) - E(t+1) \geq 4\gamma^2 - 2\gamma^2 = 2\gamma^2.$$

We now observe that $E(t) = \mathbf{E}_\mu[(f - g_t)^2] + 2\mathbf{E}_\mu[(f - g_t)(g_t - g'_t)] \geq 0$ for all t. This follows from noting that for every x and $f(x) \in \{-1,1\}$, either $f(x) - P_1(g'_t(x))$ and $P_1(g'_t(x)) - g'_t(x))$ have the same sign or one of them equals zero. Therefore $\mathbf{E}_\mu[(f - g_t)(g_t - g'_t)] \geq 0$ (and, naturally, $\mathbf{E}_\mu[(f - g_t)^2] \geq 0$). It is easy to see that $E(0) = 1$ and therefore this process will stop after at most $1/(2\gamma^2)$ steps.

The claim on the representation of g_t output by the algorithm follows immediately from the definition of $g_t = P_1(g'_t)$ and g'_t being a sum of t μ-Fourier basis functions multiplied by $\pm\gamma$. ■