An Efficient Membership-Query Algorithm for Learning DNF with Respect to the Uniform Distribution

Jeffrey C. Jackson*
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

*This research was sponsored by the National Science Foundation under Grant No. CCR-9119319. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. Author's current address: Dept. of Mathematics and Computer Science, Duquesne University, Pittsburgh, PA 15282. E-mail: jackson@mathcs.duq.edu

Running head: Learning DNF

Abstract

We present a membership-query algorithm for efficiently learning DNF with respect to the uniform distribution. In fact, the algorithm properly learns with respect to uniform the class TOP of Boolean functions expressed as a majority vote over parity functions. We also describe extensions of this algorithm for learning DNF over certain nonuniform distributions and for learning a class of geometric concepts that generalizes DNF. Furthermore, we show that DNF is weakly learnable with respect to uniform from noisy examples. Our strong learning algorithm utilizes one of Freund's boosting techniques and relies on the fact that boosting does not require a completely distribution-independent weak learner. The boosted weak learner is a nonuniform extension of a parity-finding algorithm discovered by Goldreich and Levin.

1 Introduction

Consider the following 20-questions-like game between two players, Bob and Alice. Bob has a Disjunctive Normal Form (DNF) expression f in mind. Alice is allowed to ask about the value of f on input assignments of her choosing, *i.e.*, to ask *membership queries*. She is allowed to ask a number of membership queries bounded by a polynomial in the number of terms in f, and the total processing time she is allowed is similarly bounded. Alice wins the game if she discovers a function h such that f and h produce the same output value on at least 99% of the possible input assignments (such an h is called a *strong approximator for f with respect to the uniform distribution*).

Is there a strategy that Alice can follow such that, regardless of the DNF expression chosen by Bob, Alice wins the game? Ever since Valiant introduced the Probably Approximately Correct (PAC) learning framework [46], there has been a great deal of interest in developing strategies for winning games similar to the one above. The "games" vary in two basic ways: we can vary the set of functions that Bob can choose from (the *function class*) and we can vary the rules of the game, such as the kinds of questions Alice is allowed to ask (the *learning model*).

The specific question of whether or not there is a winning strategy (a *learning algorithm*) for the DNF concept class in some reasonable learning model has been especially well-studied. While a number of algorithms have been developed for learning various subclasses of DNF in a variety of models [46, 34, 7, 3, 30, 4, 2, 11, 38, 18], many of the results for unrestricted DNF have been negative. Angluin, who introduced the model of exact learning with proper equivalence queries [5] (models mentioned here are defined in the next section), showed that DNF is not efficiently learnable in this model [6]. Subsequently, Angluin and Kharitonov [8] showed that, given standard cryptographic assumptions, if DNF is learnable with membership queries then it is learnable without membership queries with respect to the class of distributions computable by polynomial-size circuits. This result provides substantial evidence that membership queries, which have proved useful in a number of learning algorithms, may not simplify distribution-independent DNF learning. Recently, Kharitonov [35] again used cryptographic assumptions to show that the class AC^0 of constant-depth, polynomial-size $\{\wedge, \vee, \neg\}$-circuits is not even weakly learnable using membership queries with respect to the uniform distribution. Roughly speaking, this says that Alice cannot win the game above when Bob is allowed to choose an arbitrary AC^0 function f, even if Alice's function h is only required to agree with f on 51% of the input assignments (such an h is a *weak approximator to f with respect to uniform*). While this hardness result does not immediately apply to subclasses of AC^0 such as the class of functions representable as a polynomial-size DNF, it provided at least circumstantial evidence that polynomial-size DNF might also be hard to learn—even weakly—with respect to uniform.

In sharp contrast, we prove a positive DNF learning result: DNF is strongly learnable with respect to the uniform distribution using membership queries. In other words, Alice *does* have a winning (probabilistic) strategy for the original game above. This improves on two previous DNF learning algorithms in the same uniform-with-membership model: Mansour's quasi-polynomial-time algorithm [43] and a polynomial-time weak learner due to Blum *et al.* [10].

Our algorithm for DNF learning is largely the combination of two powerful tools. One of these is a beautiful technique due to Goldreich and Levin [29]. Given a black box that will answer membership queries for a Boolean function f over $\{0,1\}^n$, their technique efficiently locates a subset A of the inputs of f such that the parity χ_A of the bits in A is a weak approximator for f with respect to uniform, if such an A exists. This technique later formed the basis for a well-known learning theoretic result by Kushilevitz and Mansour [37] and has generally been referred to as the `KM` algorithm in the learning-theoretic community (see, *e.g.*, [10]). However, we will refer to the algorithm in this paper as the weak parity algorithm, or `WP` for short.

A useful connection between DNF learning and the `WP` algorithm was first noted by Blum *et al.* [10]. They showed that for every DNF f, there is a parity function χ_A that weakly approximates f with respect to uniform. Because `WP` will find such a parity function as long as it exists, this implies that `WP` is a weak learning algorithm for DNF with respect to uniform. While this is an interesting result theoretically, the weak model is clearly not as intuitive a notion of learning as the stronger model presented above.

To obtain our strong learning result for DNF, we show that for every DNF f *and* for every probability distribution D over f's inputs there is some χ_A that is a weak approximator for f with respect to D. We also show that the Goldreich/Levin technique can be extended to efficiently find such a χ_A for many (but not necessarily all) probability distributions.

The other tool we use is hypothesis boosting [44, 21, 22, 24]. In particular, we show how to apply one of Freund's boosting algorithms [21] to boost our weak learner into a strong learner for DNF with respect to the uniform distribution. Boneh and Lipton [14] have previously proved a distribution-dependent boosting result, but their boosting mechanism was somewhat specialized. Our algorithm appears to be the first to obtain a distribution-dependent strong learning algorithm by applying a standard boosting technique.

The hypothesis output by our algorithm is in TOP, the class of functions expressed as a threshold of parity functions (this class is also known as PT_1 [15]). In fact, we show that the DNF learning algorithm can also be used to properly learn TOP with respect to uniform. This is a particularly interesting result because of the expressiveness of TOP. For example, every Boolean function f has a TOP representation that is at most polynomially larger than the smallest representation of f as either a CNF or a DNF [36] or as a decision tree with parity functions at the internal nodes [37]. Several more examples of classes subsumed by TOP are presented later in the paper. However, results of Bruck [15] and Bruck and Smolensky [16] show that our algorithm does not readily extend to either the class of depth-2 threshold circuits or the class of depth-3 $\{\wedge, \vee, \neg\}$-circuits.

We then extend the basic algorithm in several ways. First, we generalize the algorithm to learn geometric concepts defined over non-Boolean domains, an area that has attracted substantial interest recently ([20] contains a nice summary of research in this area). Among other results, we show that the class UBOX—unions of axis-parallel rectangles defined over the domain $\{0, \ldots, b-1\}^n$—is efficiently learnable with respect to uniform for any constant b. This complements previous positive results for UBOX, in stronger learning models, that either restrict the class in some way or assume that n is constant [13, 42, 26, 19, 20]. We also show that DNF is learnable not only with respect to uniform but also over a variety of other distributions, including the constant-bounded product distributions. Finally, we show that DNF is, with high probability, weakly learnable with respect to uniform even if the target function is corrupted by random but persistent classification noise. While this weak learning result is not an extension of our strong learning algorithm, it does generalize the weak parity algorithm.

The remainder of the paper is organized as follows. In the next section we provide definitions and several well-known facts that are used later in the paper. Section 3 describes Freund's boosting algorithm and `WP` in some detail. Our main result, showing that DNF is strongly learnable, is in Section 4, and Section 5 describes our results concerning TOP. The remainder of the paper details the above variations on the basic learning algorithm and presents some directions for further research.

2 Definitions and notation

2.1 Functions and function classes

We will be interested in the learnability of sets (*classes*) of Boolean functions. The Boolean functions we consider are, unless otherwise noted, of the type $f : \{0,1\}^n \to \{-1,+1\}$ for fixed positive values of n, where an output of $+1$ denotes `true` and -1 `false`. We call $\{0,1\}^n$ the *instance space* of f, an element x in the instance space an *instance*, and the pair $\langle x, f(x)\rangle$ an *example* of f. We denote by x_i the ith bit of instance x. Following standard practice, for any complex-valued function g we define the norms $L_\infty(g) = \max_x\{|g(x)|\}$, $L_1(g) = \sum_x |g(x)|$, and $L_2(g) = \sqrt{\sum_x |g(x)|^2}$.

Intuitively, a learning algorithm should be allowed to run in time polynomial in the complexity of the function f to be learned; we will use the *size* of a function as a measure of its complexity. The size measure will depend on the function class to be learned. In particular, each function class $\mathcal{F}$ that we study implicitly defines a natural class $\mathcal{R}_\mathcal{F}$ of representations of the functions in $\mathcal{F}$. We define the *size of a function* $f \in \mathcal{F}$ as the minimum, over all $r \in \mathcal{R}_\mathcal{F}$ such that r represents f, of the size of r, and we define below the size measure for the two representation classes of primary interest in this paper, DNF and TOP. Arbitrary Boolean functions can be represented by these classes; we will refer to such function classes as *universal* classes. Thus, for an arbitrary Boolean function f we mean by the *DNF-size* (*TOP-size*) of f the minimum size of any DNF (TOP) representation of f.

A DNF expression is a disjunction of terms, where each term is a conjunction of literals and a literal is either a variable or its negation. An example DNF expression is $(\bar{x}_1 \wedge x_2) \vee (\bar{x}_2 \wedge x_3)$. The *size of a DNF expression* r is the number of terms in r.

A TOP expression is a majority vote over a fixed collection of parity functions, *i.e.*, a depth-2 circuit with a majority gate at the root and parity gates at the bottom level. We assume that the majority function outputs `true` if and only if over half of its inputs are `true`. By a *parity function* we mean a function that computes parity over a fixed subset of the (unnegated) inputs or the negation of such a function. Parity over the empty set is by definition the constant function `true`. Also, a parity function may occur multiple times in a TOP expression. Thus, an example of a TOP expression is $MAJ(\neg(x_1 \oplus x_4), x_2, x_2, 1)$, where we have used 1 to represent the always-true parity function. The *size of a TOP expression* r is the number of parity functions in r. Equivalently, we can think of a TOP as an integer-weighted vote over distinct parity functions; in this case, the TOP-size of a representation is the sum of the magnitudes of the weights. The TOP-size of the example expression above is 4 in either case, although it has only three distinct parity functions.

2.2 Learning models

Before defining the learning models we will consider, we define several supporting concepts. Given a function f and probability distribution D on the instance space of f, we say that function h is an *ϵ-approximator for f with respect to D* if $\mathbf{Pr}_D[h = f] \geq 1 - \epsilon$. An *example oracle for f with respect to D* ($EX(f, D)$) is an oracle that on request draws an instance x at random according to probability distribution D and returns the example $\langle x, f(x)\rangle$. A *membership oracle for f* ($MEM(f)$) is an oracle that given any instance x returns the value $f(x)$. Let $\mathcal{D}_n$ denote a nonempty set of probability distributions on $\{0,1\}^n$. Any set $\mathcal{D} = \cup_n \mathcal{D}_n$ is called a *distribution class*. For some distributions that have obvious generalizations to distribution classes (such as the uniform distribution on $\{0,1\}^n$) we blur the distinction between distribution and distribution class.

Now we formally define the Probably Approximately Correct (PAC) model of learnability [46]. Let ϵ and δ be positive values (called the *accuracy* and *confidence* of the learning procedure, respectively). Then we say that the function class $\mathcal{F}$ is *(strongly) PAC-learnable* if there is an algorithm $\mathcal{A}$ such that for any ϵ and δ, any $f \in \mathcal{F}$ (the *target function*), and any distribution D on the instance space of f (the *target distribution*), with probability at least $1 - \delta$ algorithm $\mathcal{A}(EX(f, D), \epsilon, \delta)$ produces an ϵ-approximation for f with respect to D in time polynomial in the number of input bits n, the size of f, $1/\epsilon$, and $1/\delta$. We generally drop the "PAC" from "PAC-learnable" when the model of learning is clear from context.

We will consider a number of variations on the basic PAC model. Let $\mathcal{M}$ be any model of learning (e.g., PAC). If $\mathcal{F}$ is $\mathcal{M}$-learnable by an algorithm $\mathcal{A}$ that requires a membership oracle then $\mathcal{F}$ is *$\mathcal{M}$-learnable using membership queries*. If $\mathcal{F}$ is $\mathcal{M}$-learnable for $\epsilon = 1/2 - 1/p(n, s)$, where p is a fixed polynomial and s is the size of f, then $\mathcal{F}$ is *weakly $\mathcal{M}$-learnable*. We say that *$\mathcal{F}$ is $\mathcal{M}$-learnable by $\mathcal{H}$* if $\mathcal{F}$ is $\mathcal{M}$-learnable by an algorithm $\mathcal{A}$ that always outputs a function $h \in \mathcal{H}$. If $\mathcal{F}$ is $\mathcal{M}$-learnable by $\mathcal{F}$ then we say that $\mathcal{F}$ is *properly* $\mathcal{M}$-learnable. Finally, note that the PAC model places no restriction on the example distribution D, *i.e.*, it is *distribution-independent*. If $\mathcal{F}$ is $\mathcal{M}$-learnable for all distributions D in distribution class $\mathcal{D}$ then $\mathcal{F}$ is *$\mathcal{M}$-learnable with respect to $\mathcal{D}$*.

We also at times refer to models of exact learning [5]; we now define several concepts related to these models. An *equivalence oracle for* f ($EQ(f)$) is an oracle that, given a hypothesis function h, returns an instance x such that $f(x) \neq h(x)$ if such an x exists; otherwise, the oracle returns "equivalent." A function class $\mathcal{F}$ is *exactly learnable* if there is an algorithm $\mathcal{A}$ such that for all $f \in \mathcal{F}$ and for all possible equivalence oracles $EQ(f)$, $\mathcal{A}(EQ(f))$ runs in time polynomial in n and the size of f and returns a hypothesis h equivalent to f. Function class $\mathcal{F}$ is *exactly learnable with proper equivalence queries* if $\mathcal{F}$ is exactly learnable by some algorithm $\mathcal{A}$ such that for all of the hypotheses h posed by $\mathcal{A}$ to the equivalence oracle, $h \in \mathcal{F}$.

2.3 The Fourier transform

For each set $A \subseteq \{1, \ldots, n\}$ we define the function $\chi_A : \{0,1\}^n \to \{-1, +1\}$ as

$$\chi_A(x) = (-1)^{\sum_{i \in A} x_i} = 1 - 2\left(\sum_{i \in A} x_i \bmod 2\right).$$

That is, $\chi_A(x)$ is the Boolean function that is 1 when the parity of the bits in x indexed by A is even and is -1 otherwise. Also define the inner product $\langle f, g \rangle = \mathbf{E}_x[f(x) \cdot g(x)]$ and define the norm as usual by $\|f\| = \sqrt{\mathbf{E}_x[f^2(x)]}$. Expectations and probabilities here and elsewhere are with respect to the uniform distribution over the instance space unless otherwise indicated, and we will typically omit explicit reference to the instance variable x.

Given the above definitions, $\{\chi_A \mid A \subseteq \{1, \ldots, n\}\}$ is an orthonormal set of functions, *i.e.*, $\langle \chi_A, \chi_B \rangle = 0$ if $A \neq B$ and the inner product is 1 otherwise. To see the $A \neq B$ case, choose any i that is in one set and not the other (say $i \in A$ and $i \notin B$) and split the sum defining the inner product into two sums, one with $x_i = 0$ and the other with $x_i = 1$. These two sums are negatives of one another.

Furthermore, because the vector space of real-valued functions over $\{0,1\}^n$ has dimension 2^n, the set of 2^n parity functions forms an orthonormal basis for this vector space. As a result, every function $f : \{0,1\}^n \to \mathbf{R}$ can be uniquely expressed as a linear combination of parity functions: $f = \sum_A \hat{f}(A) \cdot \chi_A$, where $\hat{f}(A) = \langle f, \chi_A \rangle = \mathbf{E}[f \cdot \chi_A]$. We call the vector of coefficients $\hat{f}$ the *Fourier transform* of f. Note that for Boolean ($\{-1, +1\}$-valued) f, $\hat{f}(A)$ represents the correlation of f and χ_A with respect to the uniform distribution. Also note that $\hat{f}(\emptyset) = \mathbf{E}[f\chi_\emptyset] = \mathbf{E}[f]$, since $\chi_\emptyset$ is the constant function $+1$.

Example: Let $f = x_1 \wedge x_2$ and let $n = 2$. Then the Fourier coefficients of f are $\hat{f}(\emptyset) = \hat{f}(\{1\}) = \hat{f}(\{2\}) = -\frac{1}{2}$

and $\hat{f}(\{1,2\}) = \frac{1}{2}$, *i.e.*

$$f = -\frac{1}{2}\left(\chi_{\emptyset} + \chi_{\{1\}} + \chi_{\{2\}} - \chi_{\{1,2\}}\right).$$

Of course, f can also be represented this way for $n > 2$. In general, if a function depends only on the variables indexed by a set S, then for all sets A such that $A - S \neq \emptyset$, $\hat{f}(A) = 0$. $\square$

Note that for any f and g mapping $\{0,1\}^n$ into the reals,

$$\begin{aligned}\mathbf{E}[fg] &= \mathbf{E}_x\left[\left(\sum_A \hat{f}(A)\chi_A(x)\right)\cdot\left(\sum_B \hat{g}(B)\chi_B(x)\right)\right] \\ &= \sum_{A,B} \hat{f}(A)\hat{g}(B)\mathbf{E}_x\left[\chi_A(x)\chi_B(x)\right].\end{aligned}$$

By the orthonormality of the χ's, this gives that $\mathbf{E}[fg] = \sum_A \hat{f}(A)\hat{g}(A)$, and note that a similar result holds for any orthonormal basis. As a corollary of this we have Parseval's identity: $\mathbf{E}[f^2] = \sum_A \hat{f}^2(A)$. For Boolean f it follows that $\sum_A \hat{f}^2(A) = 1$.

2.4 Estimating expected values

We will frequently want to estimate the expected value of random variables. The following three lemmas allow us to compute the sample size needed to ensure that with high probability the sample mean closely approximates the true mean of a random variable. The lemmas differ in the restrictions placed on the random variable and in how the closeness of approximation is measured (additively or multiplicatively).

Lemma 1 (Chernoff) *Let X_i be independent random variables all with mean μ such that for all i, $X_i \in \{0,1\}$. Then for any $0 \leq \lambda \leq 1$,*

$$\mathbf{Pr}\left[\frac{1}{m}\sum_{i=1}^{m} X_i \geq (1+\lambda)\mu\right] \leq e^{-\lambda^2 m\mu/3}$$

and

$$\mathbf{Pr}\left[\frac{1}{m}\sum_{i=1}^{m} X_i \leq (1-\lambda)\mu\right] \leq e^{-\lambda^2 m\mu/2}.$$

Lemma 2 (Hoeffding) *Let X_i be independent random variables all with mean μ such that for all i, $a \leq X_i \leq b$. Then for any $\lambda > 0$,*

$$\mathbf{Pr}\left[\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| \geq \lambda\right] \leq 2e^{-2\lambda^2 m/(b-a)^2}.$$

Lemma 3 (Bienaymé-Chebyschev) *Let X_i be pairwise independent random variables all with mean μ and variance σ^2. Then for any $\lambda > 0$,*

$$\mathbf{Pr}\left[\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| \geq \lambda\right] \leq \frac{\sigma^2}{m\lambda^2}.$$

Hoeffding's inequality will be particularly useful to us. For later reference, we state the following immediate corollary of Lemma 2:

Corollary 4 *Let X represent a random variable that produces values in the range $[a, b]$. Then there is an algorithm* $\texttt{AMEAN}(X, b-a, \lambda, \delta)$ *that for any such X and any $\lambda > 0$ produces a value μ' such that $|\mathbf{E}[X] - \mu'| \leq \lambda$ with probability at least $1 - \delta$. Furthermore,* AMEAN *runs in time polynomial in $b - a$, λ^{-1}, and $\log \delta^{-1}$, specifically $O((b-a)^2 \log \delta^{-1}/\lambda^2)$.*

Proof: Take $m = (b-a)^2 \ln(2/\delta)/(2\lambda^2)$ and let S represent the sum of m draws from the random variable X. Then $\mu' = S/m$ satisfies the requirements of the corollary. □

The Chernoff bound can be used to obtain a similar algorithm for estimating the mean of a $\{0, 1\}$-valued random variable within a multiplicative factor. At times we will want to similarly estimate the mean of an $[a, b]$-valued random variable within a multiplicative factor, rather than within an additive factor as above. The following corollary shows that under certain conditions this estimate can be performed efficiently using a standard guess-and-double technique, a technique that is also used several times elsewhere in this paper.

Corollary 5 *Let X represent a random variable that produces values in the range $[a, b]$, and let $\mathbf{E}[X] = \mu \neq 0$. Then there is an algorithm* $\texttt{MMEAN}(X, b-a, \delta)$ *that for any such X produces a value μ' such that $|\mu - \mu'| \leq |\mu|/2$ with probability at least $1 - \delta$. Furthermore, when* MMEAN *is successful (i.e., with probability at least $1 - \delta$) it runs in time polynomial in $|\mu|^{-1}$, $b - a$, and $\log \delta^{-1}$, specifically $O((b-a)^2 \log(\delta\mu)^{-1}/\mu^2)$.*

Proof: The idea behind MMEAN is the following. By Hoeffding, drawing $m \geq 8(b-a)^2 \ln(2/\delta)/(\mu^2)$ random examples of X is sufficient to produce μ' such that $|\mu' - \mu| \leq \mu/4$ with probability at least $1 - \delta$ (for readability, we will not use absolute value bars on μ when it is clear from context that we are referring to its magnitude). Of course, we do not know μ and therefore we cannot directly compute the required sample size m. However, if we compute m using a value $\mu_g < \mu$ then we will still get, with high probability, an estimate μ' within $\mu/4$ of the true mean. On the other hand, if we happen to use a value μ_g for μ that is much larger than μ—say $\mu_g = 4\mu$—and attempt the same estimation using μ_g to select the number of samples, then with high probability we will produce an estimate μ' that is much smaller than $3\mu_g/4$. Such an event is evidence that our "guess" μ_g is much larger than μ; therefore, we can refine our guess and try again. In this way we can start with a very poor guess μ_g for μ and yet quickly converge on an adequate guess, which will lead to a good estimate μ' for μ.

Specifically, we begin by taking $\mu_g = \frac{1}{2}$ and computing $\mu' = \mathtt{AMEAN}(X, b-a, \mu_g/4, \delta/2)$. If $|\mu'| \geq 3\mu_g/4$ then $\mu \geq \mu_g/2$ and therefore $|\mu' - \mu| \leq \mu/2$ with probability at least $1 - \delta/2$. However, if $|\mu'| < 3\mu_g/4$ then we halve both μ_g and δ and again estimate μ'. By the reasoning above, this procedure will, with probability at least $1-\delta$, terminate after at most $\lceil \log(\mu^{-1}) \rceil$ steps and will produce μ' such that $|\mu' - \mu| \leq \mu/2$. □

3 Two tools

Our DNF learning algorithm is largely the combination of two powerful tools: an algorithm for finding the large Fourier coefficients of a function [29, 37] and a method for "boosting" weak hypotheses into stronger ones [44, 21, 22]. These algorithms form the basis for our algorithm, and it will also be necessary to tailor them somewhat to our needs, so we discuss each of them in some detail.

3.1 Hypothesis boosting

Schapire [44] first discovered the surprising fact that every representation class that can be weakly learned (in a distribution-independent model) can be strongly learned. Subsequently, several improved boosting algorithms have been developed [21, 22, 23, 24]. In addition to certain efficiency advantages, the subsequent boosters tend to be much simpler than Schapire's original algorithm. As we will see, this simplicity will be exploited in a novel way by our Harmonic Sieve algorithm.

All of the currently-known boosting algorithms are similar in certain respects. In each case, we assume that the boosting algorithm is given a weak learner; in fact, we typically assume that the weak learner is distribution-independent and produces $(\frac{1}{2} - \gamma)$-approximators for the target representation class, where γ is known by the boosting algorithm. All of the boosters run the given weak learner multiple times and combine the resulting weak hypotheses in some manner to produce a strong hypothesis. Boosting occurs as a result of running the weak learner on different (simulated) example oracles $EX(f, D_i)$ each time, thus producing weak hypotheses that perform well on different regions of the instance space.

The boosters differ in how they define the particular simulated distributions D_i against which the weak learner is run and in how they combine the weak hypotheses generated into the final strong hypothesis. The DNF learning algorithm presented in this paper is based on one of Freund's boosting algorithms [21]; we will call this algorithm `F1`. An important feature of `F1` for our purposes is that all of the distributions D_i simulated by `F1` when boosting with respect to uniform are polynomial-time computable functions, *i.e.*, the weight assigned to x by D_i can be efficiently computed for all x. This means that in addition to supplying the weak learner with example oracle $EX(f, D_i)$ at each boosting step, we can also give the learner an oracle for the distribution D_i. This ability to provide the weak learner with a such a distribution oracle is crucial, as our weak algorithm, unlike other learning algorithms that we are familiar with, requires such an oracle.

Another noteworthy aspect of Freund's `F1` algorithm is that the final hypothesis is formed in a simple way: apply the majority function MAJ to the weak hypotheses produced during the boosting process. As the weak hypotheses generated by the Harmonic Sieve are relatively simple (they are parity functions), our final hypothesis is also quite simple. This may have implications for potential practical applications of the Harmonic Sieve and related algorithms, as discussed briefly in the concluding section.

We now discuss the `F1` boosting algorithm in detail. As input, `F1` is given positive ϵ, δ, and γ, a weak learner `WL` that produces $(\frac{1}{2}-\gamma)$-approximate hypotheses for functions in a function class $\mathcal{F}$, and an example oracle $EX(f, D)$ for some $f \in \mathcal{F}$. The weak learner `WL` is assumed to take an example oracle and confidence parameter δ' as inputs and produce a $(\frac{1}{2}-\gamma)$-approximate hypothesis with probability at least $1-\delta'$.

Given these inputs, the `F1` algorithm steps sequentially through k stages (an appropriate value for k is given below). At each stage i, $0 \leq i \leq k-1$, `F1` performs one run of `WL`, which produces (with high probability) a $(\frac{1}{2}-\gamma)$-approximate hypothesis w_i. During the first stage, `F1` runs `WL` on the given example oracle $EX(f, D)$. During each of the succeeding stages $i > 0$, `F1` runs `WL` on a simulated example oracle $EX(f, D_i)$, where distribution D_i focuses weight on those instances x such that slightly fewer than half of the i weak hypotheses generated during previous stages are correct on x and slightly more than half are incorrect. Recalling that the final hypothesis of `F1` is a majority vote over the weak hypotheses it produces, this choice of distribution makes some sense: the distribution focuses the weak learner on those instances that would be misclassified if a majority vote of the current weak hypotheses was taken, but which are also close to receiving a correct vote. It is clearly a good idea to focus the weak learner at the next stage on such instances.

On the other hand, this choice of distribution may appear to have certain problems. Specifically, such a D_i puts very little weight on instances that are very wrong (*i.e.*, instances x such that almost all of the previous hypotheses label x incorrectly), and thus these instances will very likely not be correctly labeled by the final hypothesis. While this is true, the distributions D_i begin (for small i) rather "flat" with respect to variation from the target distribution D and become more focused gradually as i increases. Thus it is not until i is fairly large that the booster begins to effectively ignore the very wrong instances, and Freund has shown that for the specific D_i's defined below there are in fact very few of these ignored instances at any stage of boosting. Thus the fact that these instances will likely be wrong in the final hypothesis is not a concern.

Before defining the distributions D_i precisely and explaining how the algorithm simulates $EX(f, D_i)$, we need some notation. Let

$$\tilde{\alpha}_r^i = \begin{cases} B(\lfloor \frac{k}{2} \rfloor - r; k-i-1, \frac{1}{2}+\gamma) & \text{if } i - \frac{k}{2} < r \leq \frac{k}{2} \\ 0 & \text{otherwise} \end{cases}$$

where $B(j; n, p) = \binom{n}{j} p^j (1-p)^{n-j}$ is the binomial formula. Also, let $\alpha_r^i = \tilde{\alpha}_r^i / \max_r \{\tilde{\alpha}_r^i\}$. Finally, let $r_i(x)$ represent the number of weak hypotheses w_j among those hypotheses produced before stage i that are "right" on x, *i.e.*, $r_i(x) = |\{0 \leq j < i \mid w_j(x) = f(x)\}|$.

Now we are ready to describe the simulation of $EX(f, D_i)$. During stage $i > 0$, when the weak learner requests an example from the simulated example oracle $EX(f, D_i)$, `F1` queries the example oracle $EX(f, D)$ and receives an example $\langle x, f(x) \rangle$. With probability $\alpha_{r_i(x)}^i$, `F1` accepts this example. If `F1` does not accept then it queries for another example, repeating this process until it does accept. Finally, `F1` passes the accepted example to `WL`. Thus the distribution D_i simulated by `F1` at stage i is

$$D_i(x) = \frac{D(x)\alpha_{r_i(x)}^i}{\sum_y D(y)\alpha_{r_i(y)}^i}. \qquad (1)$$

Stage i is completed when `WL` outputs its hypothesis w_i. After all k stages have completed, `F1` outputs as its hypothesis the majority function applied to the k weak hypotheses w_i.

It can be shown that if the number of stages k is chosen to be at least $\frac{1}{2}\gamma^{-2}\ln(\epsilon^{-1})$ and the weak learner successfully produces a $(\frac{1}{2} - \gamma)$-approximator at each stage, then `F1` will produce an ϵ-approximator as its final hypothesis. However, there is a potential computational difficulty with this approach to boosting. Notice that in general it may require many samples of the example oracle $EX(f,D)$ to simulate the example oracle $EX(f,D_i)$. In fact, it can be seen that the denominator of (1) is the probability that `F1` accepts an example while simulating the oracle $EX(f,D_i)$. Thus if at some stage i this denominator, $\sum_y D(y)\alpha^i_{r_i(y)}$, becomes quite small then to simulate the example oracle $EX(f,D_i)$ could require a very long (perhaps superpolynomial) time.

To account for this possibility, the algorithm estimates $\sum_y D(y)\alpha^i_{r_i(y)}$ at each stage i. If the estimate is below a threshold then `F1` terminates, outputting as its hypothesis h a majority vote over the weak hypotheses discovered before stage i. Intuitively, this is a reasonable stopping condition for the algorithm for the following reasons. As explained earlier, at every stage i very few instances are "very" wrong, *i.e.*, are incorrectly classified by well over half of the weak hypotheses. Also, if the probability of acceptance while simulating $EX(f,D_i)$ is very small then there must be few instances in the region focused on by D_i, that is, few instances on which slightly more than half of the weak hypotheses vote incorrectly. Putting these facts together means that few instances are incorrectly classified by h as defined above.

An implementation of `F1` that incorporates these ideas and takes into account several smaller issues is given in Figure 1. The following lemma concerning the functionality and running time of `F1` is due to Freund [21]; the previously unpublished proof of one part of Freund's argument is presented in [32].

Lemma 6 (Freund) *Algorithm* `F1`*, given positive* ϵ*,* δ*, and* γ*, a* $(\frac{1}{2}-\gamma)$*-approximate PAC learner for representation class* $\mathcal{F}$*, and example oracle* $EX(f,D)$ *for some* $f \in \mathcal{F}$ *and any distribution* D*, runs in time polynomial in* n*,* s*,* γ^{-1}*,* ϵ^{-1}*, and* $\log(\delta^{-1})$ *and produces, with probability at least* $1-\delta$*, an* ϵ*-approximation for* f *with respect to* D*.*

The specific implementation of `F1` given in Figure 1 runs in time $\tilde{O}(\gamma^{-2}(\gamma^{-2}+\epsilon^{-6}+T(\mathtt{WL})))$, where the soft-O notation $\tilde{O}$ is the same as the big-O notation but suppresses logarithmic factors, and $T(\mathtt{WL})$ denotes the maximum time required by any call to `WL`. By changing the bound on γ in line 1 of the algorithm, it is possible to move the contribution of ϵ to the algorithm's time bound arbitrarily close to ϵ^{-4} [32], but we have chosen simplicity over efficiency here for expositional purposes.

`F1` can also be used to boost a weak membership-query learner into a strong membership-query learner. The

Invocation: $h \leftarrow \mathtt{F1}(EX(f,D), \gamma, \mathtt{WL}, \epsilon, \delta)$
Input: Example oracle $EX(f,D)$ for target f and distribution D; $0 < \gamma \leq \frac{1}{2}$; $(\frac{1}{2} - \gamma)$-approximate weak learner $\mathtt{WL}(EX, \delta)$ which succeeds with probability at least $1 - \delta$; $\epsilon > 0$; $\delta > 0$
Output: h such that, with probability at least $1 - \delta$, $\mathbf{Pr}_D[f = h] \geq 1 - \epsilon$

$\bar{\gamma} \equiv \min(\gamma, 0.39)$
$k \leftarrow \frac{1}{2}\bar{\gamma}^{-2}\ln(4/\epsilon)$
$w_0 \leftarrow \mathtt{WL}(EX(f,D), \delta/2k)$
for $i \leftarrow 1, \ldots, k-1$ **do**
  $r_i(x) \equiv |\{0 \leq j < i \mid w_j(x) = f(x)\}|$
  $B(j; n, p) \equiv \binom{n}{j} p^j (1-p)^{n-j}$
  $\tilde{\alpha}^i_r \equiv B(\lfloor k/2 \rfloor - r; k - i - 1, 1/2 + \bar{\gamma})$ if $i - k/2 < r \leq k/2$, $\tilde{\alpha}^i_r \equiv 0$ otherwise
  $\alpha^i_r \equiv \tilde{\alpha}^i_r / \max_{r=0,\ldots,i-1}\{\tilde{\alpha}^i_r\}$.
  $\Theta \equiv \epsilon^3/57$
  $X \equiv$ draw example $\langle x, f(x) \rangle$ from $EX(f,D)$ and compute $\alpha^i_{r_i(x)}$
  $E_\alpha \leftarrow \mathtt{AMEAN}(X, b - a = 1, \frac{1}{3}\Theta, \delta/2k)$
  if $E_\alpha \leq \frac{2}{3}\Theta$ **then**
    $k \leftarrow i$
    break do
  endif
  $D_i(x) \equiv \frac{D(x)\alpha^i_{r_i(x)}}{\sum_y D(y)\alpha^i_{r_i(y)}}$
  $w_i \leftarrow \mathtt{WL}(EX(f, D_i), \delta/2k)$
enddo
$h(x) \equiv MAJ(w_0(x), w_1(x), \ldots, w_{k-1}(x))$
return h

Figure 1: The $\mathtt{F1}$ hypothesis boosting algorithm. $\mathtt{AMEAN}$ is described in Corollary 4. X is a random variable used to estimate $\mathbf{E}_D[\alpha^i_{r_i(x)}]$. $EX(f, D_i)$ is a simulated example oracle.

only change to `F1` is that it accepts the membership oracle $MEM(f)$ as an argument and includes this oracle as an argument in the calls to `WL`. The Harmonic Sieve will be based on the membership-query version of `F1`.

3.2 Finding weak-approximating parity functions

To produce the weak hypotheses that will be boosted, we utilize a slightly modified version of an algorithm discovered by Goldreich and Levin as part of their proof that parity functions are hard-core predicates for one-way functions [29]. As noted in the introduction, this membership-query algorithm finds a parity function which weakly approximates the target with respect to the uniform distribution over inputs, assuming such a weak approximator exists. We therefore call this the weak parity, or `WP`, algorithm (it has also been called the `KM` algorithm [10]). Kushilevitz and Mansour showed that the `WP` algorithm could be used to learn parity decision trees with respect to the uniform distribution [37], and it has since been the basis of other learning algorithms as well [43, 10].

Actually, the `WP` algorithm is somewhat more general than we have described thus far. The basic `WP` algorithm finds, with probability at least $1 - \delta$, close approximations to all of the large magnitude Fourier coefficients of a Boolean function f on $\{0,1\}^n$. Since on this domain a Fourier coefficient $\hat{f}(A)$ represents the correlation between a parity χ_A and the target function f, `WP` can be used to find *all* parities that correlate well with f. By "large" coefficients we mean coefficients of magnitude exceeding some threshold θ; the algorithm runs in time polynomial in n, $\log(\delta^{-1})$, and θ^{-1}. `WP` makes membership queries for f, but otherwise f is treated as a black box.

`WP` is a recursive algorithm that is given as input a membership oracle $MEM(f)$, threshold θ, confidence δ, and the specification of a collection of Fourier coefficients. A collection of coefficients is specified by two parameters, an integer $k \in [0, n]$ and a set $A \subseteq \{1, \ldots, k\}$; these parameters define the collection $C_{A,k}$ consisting of all the coefficients $\hat{f}(A \cup B)$ such that $B \subseteq \{k+1, \ldots, n\}$. Initially, `WP` is run on the collection of all Fourier coefficients of f, $C_{\emptyset,0}$.

Each time `WP` is called, it begins by defining a partition of the input collection $C_{A,k}$ into the two equal-sized collections $C_{A,k+1}$ and $C_{A\cup\{k+1\},k+1}$. For notational convenience we let C_i denote either of the collections in the partition. After partitioning the input collection, `WP` next tests to see whether or not a large coefficient can possibly be a member of one or both of the collections. It (conceptually) does this by computing the sum of squares of the Fourier coefficients in each collection; we denote the sum for collection C_i by $L_2^2(C_i)$. If $L_2^2(C_i) < \theta^2$ for either C_i

then certainly none of the coefficients in that C_i exceeds the threshold θ, and the coefficients in that C_i can be ignored in subsequent processing. On the other hand, if $L_2^2(C_i) \geq \theta^2$ and $|C_i| > 1$ then WP recurses on C_i. The above-threshold singleton collections remaining at the end of this process are the desired large coefficients. Because the sum of squares of the Fourier coefficients of a Boolean function is 1 (by Parseval), the algorithm will recurse on at most θ^{-2} subcollections at any level of the recursion. Thus this algorithm runs in time polynomial in the depth n of the recursion, in θ^{-1}, and in the maximum time required to compute $L_2^2(C_i)$ for any C_i.

Of course, the time required to compute $L_2^2(C_i)$ could be exponentially large. The WP algorithm gets around this difficulty by estimating $L_2^2(C_i)$ for each C_i in an ingenious way: for every function $f : \{0,1\}^n \to \mathbf{R}$, every $k \in [0, n]$, and every $A \subseteq \{1, \ldots, k\}$,

$$L_2^2(C_{A,k}) = \mathbf{E}_{x,y,z}[f(yx)f(zx)\chi_A(y \oplus z)], \tag{2}$$

where the expectation is uniform over $x \in \{0,1\}^{n-k}$, $y \in \{0,1\}^k$, and $z \in \{0,1\}^k$, and yx represents the concatenation of y and x.

Here we briefly present a derivation of Equation 2 intended to give some intuition for this relationship; a formal proof a of a more general equality is also given in Section 6. First, we define

$$f_A(x) = \sum_B \hat{f}(A \cup B)\chi_B(x),$$

where $x \in \{0,1\}^{n-k}$ and $A \subseteq \{1, \ldots, k\}$ as above and $B \subseteq \{k+1, \ldots, n\}$. Now by Parseval, $\mathbf{E}_x[f_A^2(x)] = \sum_B \hat{f}_A^2(B)$, and by our definition of f_A this is exactly $L_2^2(C_{A,k})$. Therefore, to complete the derivation, we only need an appropriate expression for $f_A(x)$. A key observation is that if we let f_x represent the function f restricted so that the last $n-k$ inputs to f are fixed to the values given by x, then for all $A \subseteq \{1, \ldots, k\}$,

$$\sum_B \hat{f}(A \cup B)\chi_B(x) = \widehat{f_x}(A),$$

which follows easily from the definition of the Fourier transform. Finally, if we expand $\widehat{f_x}(A)$ by applying the definition of a Fourier coefficient, we get that

$$f_A(x) = \mathbf{E}_y[f(yx)\chi_A(y)],$$

where the expectation is uniform over $y \in \{0,1\}^k$. In short, the fact that there is a strong relationship between the Fourier coefficients of any restriction of a function f and some subset of the Fourier coefficients of f itself allows us to estimate the sum of squares of certain subsets of the coefficients of f by estimating the coefficients of certain restrictions of f.

Thus, by Equation 2, $L_2^2(C_{A,k})$ can be estimated by using membership queries to sample f appropriately. In particular, by Hoeffding's inequality, `WP` can efficiently estimate a value μ' such that

$$\mathbf{Pr}\left[|\mu' - L_2^2(C_{A,k})| \geq \frac{\theta^2}{4}\right] \leq \frac{\delta\theta^2}{2n}.$$

Therefore, with this same probability, if $L_2^2(C_{A,k}) \geq \theta^2$ then $\mu' \geq 3\theta^2/4$, and if $L_2^2(C_{A,k}) < \theta^2/2$ then $\mu' < 3\theta^2/4$. With high probability, then, if we recurse only on collections that have $\mu' \geq 3\theta^2/4$ then we will recurse on all the collections that have $L_2^2(C_{A,k}) \geq \theta^2$ and we will not recurse on any collection such that $L_2^2(C_{A,k}) < \theta^2/2$. By Parseval, this implies that we will recurse on at most $2/\theta^2$ collections for each of the n values of k.

By the earlier argument this will give us an algorithm that with probability at least $1-\delta$ returns a set $S \subseteq 2^{\{1,\ldots,n\}}$ such that

1. For all A, if $|\hat{f}(A)| \geq \theta$ then $A \in S$; and
2. For all $A \in S$, $|\hat{f}(A)| \geq \frac{\theta}{\sqrt{2}}$.

We say that such a set has the *large Fourier coefficient property* for the function f and threshold θ. The running time of the algorithm is bounded asymptotically by the time required to adequately estimate $L_2^2(C_{A,k})$ at most $2n/\theta^2$ times, giving a bound of $O(n \log(n/\delta\theta^2)/\theta^6)$.

Our learning algorithms will also need to find the large coefficients of certain functions that are not Boolean valued. This leads us to extend `WP` slightly:

Lemma 7 *There is an algorithm* $\mathtt{WP}'$ *such that, for any function* $g : \{0,1\}^n \to \mathbf{R}$*, threshold* $\theta > 0$*, and confidence* $\delta > 0$, $\mathtt{WP}'(n, MEM(g), \theta, L_\infty(g), \delta)$ *returns, with probability at least* $1-\delta$*, a set with the large Fourier coefficient property.* $\mathtt{WP}'$ *uses membership queries and runs in time polynomial in* n, θ^{-1}, $\log(\delta^{-1})$*, and* $L_\infty(g)$*, specifically* $O(nL_\infty^6(g)\log(nL_\infty^2(g)/\delta\theta^2)/\theta^6)$.

Proof: The algorithm is shown in Figure 2. For each collection $C_{A,k}$, by Hoeffding's inequality $O(L_\infty^4(g)\log(nL_\infty^2(g)/\delta\theta^2)/\theta^4)$ samples of the random variable X are sufficient for `AMEAN` to estimate μ' such that

$$\mathbf{Pr}\left[|\mu' - L_2^2(C_{A,k})| \geq \frac{\theta^2}{4}\right] \leq \frac{\delta\theta^2}{2nL_\infty^2(g)}.$$

Invocation: $S \leftarrow \texttt{WP}'(n, MEM(g), \theta, L_\infty(g), \delta)$
Input: Number n of inputs to function $g : \{0,1\}^n \to \mathbf{R}$; membership oracle $MEM(g)$; $0 < \theta \leq 1$; $L_\infty(g) > 0$; $\delta > 0$
Output: Set S that, with probability at least $1 - \delta$, has the large Fourier coefficient property for g and θ.

1. $A \leftarrow \emptyset$
2. $k \leftarrow 0$
3. **return** $\texttt{WP}'\texttt{-aux}(A, k, n, MEM(g), \theta, L_\infty(g), \delta)$

Invocation: $S \leftarrow \texttt{WP}'\texttt{-aux}(k, A, n, MEM(g), \theta, L_\infty(g), \delta)$
Input: Integer $k \in [0, n]$; set $A \subseteq \{1, \ldots, k\}$; number n of inputs to function $g : \{0,1\}^n \to \mathbf{R}$; membership oracle $MEM(g)$; $0 < \theta \leq 1$; $L_\infty(g) > 0$; $\delta > 0$
Output: A set S that, with probability at least $1 - \delta$, is the intersection of a set T that has the large Fourier coefficient property for g and θ with the set $\{B \subseteq \{1, \ldots, n\} \mid B \cap \{1, \ldots, k\} = A\}$.

1. $X \equiv$ draw $x \in \{0,1\}^{n-k}$, $y, z \in \{0,1\}^k$ uniformly at random and compute $g(yx)g(zx)\chi_A(y \oplus z)$
2. $\mu' \leftarrow \texttt{AMEAN}(X, 2L_\infty^2(g), \theta^2/4, \delta\theta^2/(2nL_\infty^2(g)))$
3. **if** $\mu' < 3\theta^2/4$ **then**
4.   **return** $\emptyset$
5. **else if** $k = n$ **then**
6.   **return** $\{A\}$
7. **else**
8.   **return** $\texttt{WP}'\texttt{-aux}(k+1, A, n, MEM(g), \theta, L_\infty(g), \delta) \cup \texttt{WP}'\texttt{-aux}(k+1, A \cup \{k+1\}, n, MEM(g), \theta, L_\infty(g), \delta)$

Figure 2: The $\texttt{WP}'$ parity-finding algorithm. $\texttt{AMEAN}$ is described in Corollary 4. X is a random variable used to estimate $\mathbf{E}_{x,y,z}[g(yx)g(zx)\chi_A(y \oplus z)] = L_2^2(C_{A,k})$.

The fact that $\sum_A \hat{g}^2(A) = \mathbf{E}[g^2] \leq L_\infty^2(g)$ means that the number of recursive calls at each of the n levels of the recursion is, with probability at least $1 - \delta$, bounded above by $2L_\infty^2(g)/\theta^2$. The rest of the argument is analogous to that for $\texttt{WP}$. □

4 Learning DNF with respect to uniform

In this section we prove our main result: DNF is learnable with respect to the uniform distribution using membership queries. We begin by extending a result of Blum *et al.*, who showed that for every DNF expression f there is a parity function that weakly approximates f with respect to the uniform distribution [10]. Below we show that for every DNF f *and* for every distribution D there is a parity function that weakly approximates f with respect to D. Next we show how to exploit this fact to produce an algorithm for weakly learning DNF with respect to certain nonuniform distributions. It is then shown that this weak learner can be boosted into a strong learner for DNF with respect to the uniform distribution. We close the section with a brief comparison between the DNF algorithm and earlier Fourier-based algorithms.

We now prove that for every DNF expression f and every probability distribution D over the instance space of f there is a parity that weakly approximates f with respect to D. Before presenting the proof we give some of the intuition behind this result.

First, note that if $\mathbf{E}_D[f(x)]$ deviates noticeably (*i.e.*, inverse polynomially in the size of f) from 0 then the constant parity $\chi_\emptyset$ or its negation is a weak approximator to f with respect to D (recall that we are assuming Boolean functions map to $\{-1, +1\}$). The harder case in proving our claim, then, is showing that the claim holds when D is such that f is unbiased, or very nearly so. In this case, we can conceptually split the instance space into two disjoint sets: those instances that satisfy f (call this set Sat) and those that do not ($Unsat$). Then the correlation of any function $T(x)$ with f is given by

$$
\begin{aligned}
& \mathbf{Pr}_D[x \in Sat] \cdot \mathbf{Pr}_D[T(x) = f(x) \mid x \in Sat] + \mathbf{Pr}_D[x \in Unsat] \cdot \mathbf{Pr}_D[T(x) = f(x) \mid x \in Unsat] \\
\approx \;& \frac{1}{2}\left[\mathbf{Pr}_D[T(x) = f(x) \mid x \in Sat] + \mathbf{Pr}_D[T(x) = f(x) \mid x \in Unsat]\right].
\end{aligned}
$$

Now let s be the number of terms in the DNF expression f and let $T(x)$ be the $\{-1, +1\}$-valued function equivalent to the term in f that is best correlated with f with respect to D (and recall that we take -1 to represent `false` (unsatisfied) and $+1$ to represent `true` (satisfied), although the opposite convention is often used in Fourier work). Since T represents a term of f, $\mathbf{Pr}_D[T(x) = f(x) \mid x \in Unsat] = 1$. Furthermore, since there are only s terms and T is the best correlated term, $\mathbf{Pr}_D[T(x) = f(x) \mid x \in Sat] \geq 1/s$. Thus T agrees with f with probability noticeably greater than $\frac{1}{2}$, or equivalently, $\mathbf{E}_D[f \cdot T]$ is noticeably large.

Next, note that T is a Boolean function and therefore can be expressed as a linear combination of parity functions by applying the Fourier transform. In fact, since T corresponds to a term (which we will also call T) and a term is simply a conjunction, the function T has a very simple Fourier representation. Specifically, call χ_A a *parity in term* T (denoted $\chi_A \in T$) if A is a (possibly empty) subset of the variables in term T, and let $|T|$ represent the number of variables in term T. Then it can be shown that the Fourier representation of the $\{-1, +1\}$-valued function corresponding to the term T is

$$T(x) = -1 + 2 \cdot \left(\frac{1}{2^{|T|}} \sum_{\chi_A \in T} \pm\chi_A(x)\right),$$

where the sense of a parity (negated or not) in the sum depends on the senses of the variables in the term. This implies that

$$|\mathbf{E}_{x\in_R D}[f(x)\cdot T(x)]| \leq |\mathbf{E}_D[f(x)]| + 2\mathbf{E}_{\chi_A\in T}\left[|\mathbf{E}_D[f(x)\chi_A(x)]|\right],$$

where $\mathbf{E}_{\chi_A\in T}[\cdot]$ represents the expected value over uniform random choice of $\chi_A \in T$.

But since we showed above that the function $T(x)$ is noticeably well-correlated with an essentially unbiased function f—in other words, that $\mathbf{E}_D[f\cdot T]$ is noticeably large while $\mathbf{E}_D[f]$ is very small—the above relation implies that the "average" of the parity functions in T is noticeably well-correlated with f. Therefore, at least one of these parity functions must be well-correlated with f, which gives us our claim.

Our original proof formalized these ideas. Subsequently, Bshouty [17] refined our approach and developed the proof that we present here, which gives somewhat better constants.

Fact 8 *For every DNF expression f with s terms and for every distribution D on the instance space of f there exist a term T in f and a $\chi_A \in T$ such that $|\mathbf{E}_D[f\chi_A]| \geq 1/(2s+1)$.*

Proof: Let $p = \mathbf{Pr}_{x\in_R D}[x \text{ satisfies } f]$. There is at least one term T in f such that $\mathbf{Pr}_D[x \text{ satisfies } T] \geq p/s$. Let $T(x)$ be the Boolean function represented by T, *i.e.*, $T(x) = 1$ when x satisfies T and $T(x) = -1$ otherwise. Also, let $T'(x) = (T(x)+1)/2$; that is, $T'(x)$ is the $\{0,1\}$-valued version of $T(x)$. Furthermore, assume without loss of generality that none of the literals in T are negated and let V represent the set indices i of the variables x_i appearing in T. Then for all x, the definition of the parity function χ_A gives us immediately that

$$T'(x) = \prod_{v\in V} \frac{1-\chi_{\{v\}}(x)}{2}.$$

It also follows from the definition that if A and B are disjoint subsets of $\{1,\ldots,n\}$ then $\chi_A(x)\cdot\chi_B(x) = \chi_{A\cup B}(x)$. Thus it is readily seen that the product above—and therefore $T'(x)$—is equivalent to $\mathbf{E}_{A\subseteq V}[(-1)^{|A|}\chi_A(x)]$, where the expectation is uniform over the subsets $A \subseteq V$.

Now consider the quantity $\mathbf{E}_{x\in_R D}[f(x)\cdot T'(x)]$. Substituting the expected value above for T' and rearranging gives

$$\mathbf{E}_D[fT'] = \mathbf{E}_{A\subseteq V}[(-1)^{|A|}\mathbf{E}_D[f\chi_A]] \leq \mathbf{E}_{A\subseteq V}[|\mathbf{E}_D[f\chi_A]|].$$

Also, since T is a term of f, for any x, $f(x) = -1$ implies $T'(x) = 0$, and $T'(x) = 1$ implies $f(x) = 1$. Thus

$$\mathbf{E}_D[fT'] = \mathbf{E}_D[T'] = \mathbf{Pr}_D[T=1] \geq p/s.$$

Combining these two relationships for $\mathbf{E}_D[fT']$, we see that there is some $\chi_A \in T$ such that $|\mathbf{E}_D[f\chi_A]| \geq p/s$. Also, since f is $\{-1,+1\}$-valued, it is easy to show that $p = (\mathbf{E}_D[f]+1)/2$. Thus if $\mathbf{E}_D[f] \geq -1/(2s+1)$ then there is some $\chi_A \in T$ such that $|\mathbf{E}_D[f\chi_A]| \geq 1/(2s+1)$. On the other hand, if $\mathbf{E}_D[f] \leq -1/(2s+1)$ then $|\mathbf{E}_D[f\chi_\emptyset]| \geq 1/(2s+1)$, since $\chi_\emptyset \equiv 1$. Because for all T, $\chi_\emptyset \in T$, this completes the proof. □

4.1 Nonuniform Weak DNF Learning

Fact 8 says that for every DNF f and every distribution D there exists a parity weakly approximating f with respect to D. This suggests that we may be able to strongly learn DNF by boosting a weak DNF learner that produces parity functions as the weak hypotheses. The question, of course, is whether or not there is an efficient algorithm for finding appropriate parity functions.

We already know the answer to this question when the problem is restricted to finding a weakly-approximating parity with respect to uniform: the `WP` algorithm can efficiently solve this problem. Thus the `WP` algorithm is a natural basis for the more general weak learner we desire. In fact, it is known that certain uniform-distribution learnability results based on `WP` can be generalized to learn with respect to product distributions [9] (product distributions are defined in Section 7). However, it is not clear that an algorithm for weakly learning with respect to this distribution class can be boosted into a strong learner; certainly a richer class of distributions is required if we plan to use an existing boosting algorithm such as `F1`.

Thus we seek an efficient mechanism for finding weakly-approximating parity functions with respect to a fairly broad class of distributions. However, there is relatively strong circumstantial evidence that DNF may not be weakly learnable with respect to arbitrary distributions [10]. What we are seeking, then, is an algorithm that is apparently the first of its kind: an algorithm that learns efficiently with respect to a rather general distribution class (in particular, one with no independence assumptions), but also an algorithm that does not necessarily efficiently learn over arbitrary distributions. This presents quite a puzzle!

The solution to this puzzle begins with the following simple but critical observations. What we would like is an algorithm that generalizes `WP`, which recall is given a threshold θ and uses membership queries on the target f to find the index A of a Fourier coefficient $\hat{f}(A)$ such that $|\hat{f}(A)| \geq \theta/\sqrt{2}$. Because a Fourier coefficient $\hat{f}(A)$ is by definition $\mathbf{E}[f \cdot \chi_A]$, finding the index A of a large Fourier coefficient $\hat{f}(A)$ leads us to a parity χ_A that weakly approximates f with respect to uniform. We would like a more general algorithm that, given both θ and (in a form to be determined) a distribution D, finds a χ_A such that, say, $|\mathbf{E}_D[f \cdot \chi_A]| \geq \theta/2$. While the expected value in this relation cannot be viewed as a Fourier coefficient of f as was the case when D was uniform, notice that

$$\mathbf{E}_D[f \cdot \chi_A] = \sum_x f(x)\chi_A(x)D(x) = \frac{1}{2^n}\sum_x 2^n f(x)D(x)\chi_A(x).$$

Thus if we take $g(x) = 2^n f(x)D(x)$ then we have that for all A, $\mathbf{E}_D[f \cdot \chi_A] = \hat{g}(A)$. That is, finding a large Fourier coefficient $\hat{g}(A)$ of g will lead us to a parity χ_A that weakly approximates f with respect to D.

Therefore, we have reduced the problem of efficiently finding a well-correlated parity with respect to arbitrary distributions D to efficiently finding a large Fourier coefficient of a function g which is essentially the product of the target f and the distribution D. If we had a membership oracle for g and if g was Boolean then we could apply `WP` directly to the problem of finding a large coefficient of g. In fact, if we are given an oracle for distribution D (a function that given input x returns the weight assigned to x by D) along with a membership oracle for f then it is a simple matter to simulate an oracle for $g = 2^n fD$. Also, as shown in Section 3.2, a modified version of `WP` can find the large Fourier coefficients of non-Boolean g in time polynomial in $L_\infty(g)$ as well as in the normal parameters of `WP`. For g of the form we are interested in, this means that the algorithm runs in time polynomial in $L_\infty(2^n D)$. These observations form the basis of an algorithm `WDNF` for efficiently learning DNF (in a weak sense) with respect to a broad class of distributions:

Lemma 9 *Let D represent both a probability distribution over $\{0,1\}^n$ and the corresponding distribution oracle. There is an algorithm* `WDNF` *such that for any function $f : \{0,1\}^n \to \{-1,+1\}$, for any probability distribution D on the instance space of f, and for any positive δ,* `WDNF`$(EX(f,D), MEM(f), D, \delta)$ *finds, with probability at least $1-\delta$, a Boolean function h such that $\mathbf{E}_D[fh] = \Omega(s^{-1})$, where s is the DNF-size of f. The probability of success is taken over the random choices made by the* `WDNF` *algorithm and by the oracle $EX(f,D)$. The algorithm, when it succeeds, runs in time polynomial in n, s, $\log(\delta^{-1})$, and $L_\infty(2^n D)$. Specifically, let $K = max\{s, L_\infty(2^n D)\}$. Then the algorithm runs in time $\tilde{O}(nK^{12})$. The weak hypothesis h is a parity function (possibly negated).*

Proof: Let $g(x) = 2^n f(x)D(x)$. Then by Fact 8 and the above argument there is some χ_A such that $|\mathbf{E}_D[f\chi_A]| = |\hat{g}(A)| \geq 1/(2s+1)$. Thus by Lemma 7, `WP'`$(n, MEM(g), \theta = 1/(2s+1), L_\infty(g) = L_\infty(2^n D), \delta)$ will, with probability

Invocation: $h \leftarrow \texttt{WDNF}(EX(f,D), MEM(f), D, \delta)$
Input: Example oracle $EX(f,D)$ for target f and distribution D; membership oracle $MEM(f)$ for f; distribution oracle D; $\delta > 0$
Output: h such that, with probability at least $1-\delta$, $\mathbf{Pr}_D[f = h] \geq \frac{1}{2} + \frac{1}{8s+4}$, where s is the DNF-size of f.

$s \leftarrow 1$; $L \leftarrow 1$; $\delta' \leftarrow \delta/2$; $M \leftarrow 0$
repeat
    $MEM(g)(x) \equiv 2^n \cdot MEM(f)(x) \cdot D(x)$
    $\theta \equiv 1/(2s+1)$
    $S \leftarrow \texttt{WP}'(n, MEM(g), \theta, L, \delta'/2)$
    if $|S| \leq 2L^2/\theta^2$ **then**
        for each $B \in S$ **do**
            $X \equiv$ sample $EX(f,D)$ producing $\langle x, \ell \rangle$ and compute $\ell \cdot \chi_B(x)$
            $\mu' \leftarrow \texttt{AMEAN}(X, 2, \theta/4, \delta'\theta^2/(4L^2))$
            if $|\mu'| \geq 3\theta/4$ and $|\mu'| > |M|$ **then** $M \leftarrow \mu'$; $A \leftarrow B$
        enddo
    endif
    $s \leftarrow 2s$; $L \leftarrow 2L$; $\delta' \leftarrow \delta'/2$
until $M \neq 0$
return $h \equiv \text{sign}(M) \cdot \chi_A$

Figure 3: The weak DNF learning algorithm WDNF. The notation "$MEM(g)(x)$" represents the value returned by the (simulated) membership oracle for g on input x.

at least $1-\delta$, find a χ_A such that $|\mathbf{E}_D[f\chi_A]| = \Omega(s^{-1})$. Furthermore, the $\texttt{WP}'$ algorithm when given these parameters runs in time polynomial in n, s, $\log(\delta^{-1})$, and $L_\infty(2^n D)$.

Note, however, that WDNF is not given the values of s or of $L_\infty(2^n D)$. We circumvent this difficulty as illustrated in Figure 3. Specifically, note that if $\texttt{WP}'$ is called using values s' and $L'_\infty(2^n D)$ that are larger than their respective true values, $\texttt{WP}'$ will still succeed (with high probability) at finding an appropriate set of large Fourier coefficients of g. However, the algorithm may run longer than it would have run had the smaller values been used. Therefore, we use a simple guess-and-double technique that quickly converges on parameter values that are larger than necessary and yet small enough to maintain the desired performance guarantees. To assure that the overall algorithm succeeds with the desired confidence, we require successively smaller probabilities of failure δ' of $\texttt{WP}'$ as each new set of parameter guesses is used.

There is still another difficulty: because we are guessing the value of $L_\infty(2^n D)$, we do not have any guarantee that a nonempty set S returned by $\texttt{WP}'$ has the large Fourier coefficient property ($\texttt{WP}'$ may be using a number of examples insufficient to accurately estimate the required Fourier coefficients). However, we can use calls to the example oracle $EX(f,D)$ to estimate $\mathbf{E}_D[f\chi_B]$ for each $B \in S$. By Hoeffding (Lemma 2), if we use $m = \lceil 8 \ln(8L^2/\delta'\theta^2)/\theta^2 \rceil$

examples then, with probability at least $1 - \delta'\theta^2/4L^2$, our estimate μ' is within $\theta/4$ of $\mathbf{E}_D[f\chi_B]$. Therefore, if $|\mu'| \geq 3\theta/4$ then $|\mathbf{E}_D[f\chi_B]| \geq \theta/2$. Furthermore, since the parity χ_A returned as the hypothesis of `WDNF` corresponds to an $A \in S$ which gives a maximal estimate μ', and since there is some parity which has true correlation with f of at least $1/(2s+1)$, the returned hypothesis h has correlation $\mathbf{E}_D[fh] \geq 1/(4s+2)$.

Also note that if a sufficiently large value of L is used in a call to `WP`$'$ then by the proof of Lemma 7, $|S| \leq 2L^2/\theta^2$ with high probability. Therefore, if the algorithm detects that the size of a set S returned by `WP`$'$ exceeds this bound, it takes this as evidence that the guessed parameter L was too small and does no further processing on S. Thus by allowing `WP`$'$ to fail with probability at most $\delta'/2$ and each of the at most $2L^2/\theta^2$ estimates of Fourier coefficients $\hat{g}(B)$ to fail with probability at most $\delta'\theta^2/4L^2$, we have that the overall failure of each pass through the main loop is at most δ'. Furthermore, our choices for δ' assure that the overall failure probability of `WDNF` is at most δ.

Finally, note that the algorithm terminates (with the required probability) after $O(\log(s)+\log(L_\infty(2^n D)))$ steps, where the time required for each step is dominated by the call to `WP`$'$, which requires time (ignoring log factors) $\tilde{O}(nL^6/\theta^6)$. The above bound on the number of steps and the relationship between θ and s gives the claimed time bound. □

Note that the uniform distribution assigns weight 2^{-n} to all inputs. Thus an immediate corollary of the above lemma is that DNF is weakly learnable (given example, membership, and distribution oracles) with respect to any distribution D such that for all $x \in \{0,1\}^n$, the weight that D assigns to x is at most $p(n, s, \epsilon^{-1}, \delta^{-1})/2^n$ for some fixed polynomial p. In other words, `WDNF` weakly learns DNF with respect to any distribution D that puts only polynomially more weight on its inputs than does the uniform distribution. We will refer to such distributions as *polynomially-near uniform.*

Thus we now have, as desired, a weak DNF algorithm that efficiently learns with respect to a rather broad class of distributions (given certain oracles, including the nonstandard distribution oracle). The learning algorithm also has the expected property that it does not guarantee efficient learning with respect to arbitrary distributions. This is a promising start; what remains is to show how `WDNF` can be integrated with Freund's `F1` hypothesis boosting algorithm to produce a strong learning algorithm for DNF.

4.2 Strongly Learning DNF

If the weak learner WDNF is to be efficient, two properties are required of the target distribution D: the distribution must be polynomially-near uniform, and an oracle for the distribution must be provided to the learner. Now consider the target distributions D_i generated at each stage of Freund's F1 boosting algorithm when the booster's goal is to produce an ϵ-approximating hypothesis with respect to uniform. As we will see below, these target distributions D_i are polynomially-near uniform. In fact, this is true of all boost-by-filtering algorithms. However, a second property of F1's distributions D_i is not true of the distributions of some other boosting algorithms: each of F1's distributions is defined—modulo a scale factor—by a polynomial-time computable function. The scale factor can also be estimated efficiently. Therefore, an approximate distribution oracle can be provided to the weak learner for each target distribution D_i generated during boosting. Furthermore, we show that the weak learner does not require an exact distribution oracle; this approximate oracle suffices.

Putting this all together gives an algorithm for strongly learning DNF with respect to uniform. We call our algorithm the Harmonic Sieve (HS) because conceptually it repeatedly finds a dominant harmonic (parity function that correlates well with the target) and then damps out its influence (shifts the distribution) so that another harmonic may become dominant.

Theorem 10 *DNF is learnable with respect to the uniform distribution using membership queries.*

Proof: We will actually prove a somewhat more general result: the Harmonic Sieve algorithm, given an oracle for any probability distribution D along with the usual parameters, learns DNF with respect to D in time polynomial in n, the DNF-size s of the target f, ϵ^{-1}, $\log(\delta^{-1})$, and $L_\infty(2^n D)$. As indicated above, the algorithm is essentially an application of F1 to boosting the weak learner WDNF developed in the preceding section. Figure 4 describes the algorithm. For simplicity, this version of the algorithm assumes that the DNF-size s of the target f is provided; this assumption is easily removed by modifying the algorithm to use a guess-and-double technique similar to that used in WDNF. Note also that in most respects the main procedure of the Harmonic Sieve is identical to F1. The major difference is that HS simulates an approximate distribution oracle D'_i at every stage of boosting and provides this simulated oracle to the weak learner.

To see that this algorithm satisfies the requirements of the theorem, first note that if WDNF succeeds at producing a

Invocation: $h \leftarrow \mathtt{HS}(EX(f,D), MEM(f), D, s, \epsilon, \delta)$
Input: Example oracle $EX(f,D)$ for target f and distribution D; membership oracle $MEM(f)$; distribution oracle D; DNF-size s of f; $\epsilon > 0$; $\delta > 0$
Output: h such that, with probability at least $1-\delta$, $\mathbf{Pr}_D[f = h] \geq 1 - \epsilon$

$\gamma \leftarrow 1/(8s+4)$
$k \leftarrow \frac{1}{2}\gamma^{-2}\ln(4/\epsilon)$
$w_0 \leftarrow \mathtt{WDNF}(EX(f,D), MEM(f), D, \delta/2k)$
for $i \leftarrow 1, \ldots, k-1$ **do**
  $r_i(x) \equiv |\{0 \leq j < i \mid w_j(x) = f(x)\}|$
  $B(j;n,p) \equiv \binom{n}{j} p^j (1-p)^{n-j}$
  $\tilde{\alpha}^i_r \equiv B(\lfloor k/2 \rfloor - r; k-i-1, 1/2+\bar{\gamma})$ if $i - k/2 < r \leq k/2$, $\tilde{\alpha}^i_r \equiv 0$ otherwise
  $\alpha^i_r \equiv \tilde{\alpha}^i_r / \max_{r=0,\ldots,i-1}\{\tilde{\alpha}^i_r\}$.
  $\Theta \equiv \epsilon^3/57$
  $X \equiv$ draw example $\langle x, f(x)\rangle$ from $EX(f,D)$ and compute $\alpha^i_{r_i(x)}$
  $E_\alpha \leftarrow \mathtt{AMEAN}(X, b-a=1, \frac{1}{3}\Theta, \delta/2k)$
  if $E_\alpha \leq \frac{2}{3}\Theta$ **then**
    $k \leftarrow i$
    break do
  endif
  $D_i(x) \equiv \frac{D(x)\alpha^i_{r_i(x)}}{\sum_y D(y)\alpha^i_{r_i(y)}}$
  $D'_i(x) \equiv (D(x)\alpha^i_{r_i(x)})/E_\alpha$
  $w_i \leftarrow \mathtt{WDNF}(EX(f,D_i), MEM(f), D'_i, \delta/2k)$
enddo
$h(x) \equiv MAJ(w_0(x), w_1(x), \ldots, w_{k-1}(x))$
return h

Figure 4: The HS algorithm for efficiently learning DNF. AMEAN is described in Corollary 4. X is a random variable used to estimate $\mathbf{E}_D[\alpha^i_{r_i(x)}]$. $EX(f,D_i)$ is a simulated example oracle. D'_i is a simulated distribution oracle.

$(\frac{1}{2} - \frac{1}{8s+4})$-approximate weak hypothesis with respect to D_i at each stage i then Lemma 6 shows that the hypothesis produced by HS will be an ϵ-approximator to the target f with respect to the target distribution D. Furthermore, by Lemma 9, WDNF will produce such a hypothesis (with high probability) given an oracle for D_i. This presents the following difficulty: to simulate an exact oracle for D_i (line 16 of Figure 4) requires computing exactly the exponentially large sum $\sum_y D(y)\alpha^i_{r_i(y)}$.

To circumvent this difficulty, HS provides WDNF with the approximate oracle D'_i mentioned above (line 17). Note from line 11 that HS, like F1, uses random sampling of the example oracle $EX(f, D)$ to estimate an approximation E_α to the sum $\sum_y D(y)\alpha^i_{r_i(y)}$. This estimate is, with high probability, within an additive factor $\Theta/3$ of the true value. Because the test at line 12 assures (with high probability) that every D'_i is computed using a value of E_α that exceeds $2\Theta/3$, $\frac{2}{3}\sum_y D(y)\alpha^i_{r_i(y)} \leq E_\alpha \leq 2\sum_y D(y)\alpha^i_{r_i(y)}$. This implies that there is a constant $c_i \in [1/2, 3/2]$ such that for all x, $D'_i(x) = c_i D_i(x)$.

Now consider the functional impact of supplying this approximate oracle rather than the true oracle to WDNF (we consider the impact on running time below). WDNF uses its given distribution oracle (call it D_W) for exactly one purpose: to simulate a membership oracle for the function $g = 2^n f D_W$. Thus the only impact of multiplying D_W by a constant c is to multiply the function g by the same constant. Furthermore, the Fourier transform is a linear operator, and thus $\widehat{c \cdot g}(A) = c \cdot \hat{g}(A)$ for all A. In summary, multiplying WDNF's distribution oracle by a constant has the effect of multiplying all of the Fourier coefficients of the induced function g by the same constant. Because the relative sizes of the coefficients are unchanged, multiplying g by a constant will not adversely affect the ability of WP$'$ to find the large Fourier coefficients of g.

Thus, changing the distribution oracle by a constant factor has no expected impact on the functionality of WDNF. Therefore, by the earlier argument, HS will return an ϵ-approximator to the target f.

The only remaining concern is with the running time of the algorithm. First, recalling that HS is very similar to the polynomial-time algorithm F1, note that HS also runs in time polynomial in the appropriate parameters if two conditions are met: both the number of boosting stages k of HS and the running time of WDNF must be bounded by polynomials. Clearly the number of boosting stages is polynomially bounded. Thus all that remains is to bound the running time of WDNF.

By Lemma 9, if WDNF was invoked with a distribution oracle for D_i then the running time of WDNF would be appropriately bounded, since $L_\infty(D_i) \leq 3L_\infty(D)/\Theta$ and Θ is polynomial in ϵ. Also notice that if the distribution oracle is multiplied by a constant factor between 1 and 1.5 then `WDNF` may be forced to execute the body of its repeat-until loop one more time than otherwise in order to "guess" a value of L large enough for the successful operation of `WP'`. The only other impact on the running time of `WDNF` comes from changes to the running time of the call to `WP'`. There are two potential effects on the running time of `WP'` resulting from multiplication of D_W by c. First, because this multiplication may reduce the magnitude of the Fourier coefficients of g, it may be necessary to use a smaller threshold value θ in the call to `WP'` in order to find a large Fourier coefficient. Second, because the running time of `WP'` depends on $L_\infty(2^n D_W)$, this multiplication may directly increase the time bound. However, in both cases the running time is increased by at most a small constant if the multiplicative factor c is near 1, as we guaranteed earlier. Thus the running time of `WDNF`, and therefore of `HS`, is bounded by a polynomial in the desired parameters. □

Taking $K = O(s + L_\infty(2^n D)/\epsilon^3)$, it can be seen that `HS` runs in time $\tilde{O}(s^2(nK^{12} + \epsilon^{-6}))$ for learning DNF with respect to arbitrary distribution D, where the factor of nK^{12} is associated with `WDNF` and the ϵ factor with `AMEAN`. For the special case of learning with respect to uniform, we can modify the algorithm slightly and obtain a better bound. Specifically, in this case the `WDNF` algorithm can fix $L = (3 \cdot 57)/\epsilon^3$ rather than "guessing" the value of L. This modified `WDNF` runs in time $\tilde{O}(ns^6/\epsilon^{18})$, giving an overall bound for learning DNF with respect to uniform of $\tilde{O}(ns^8/\epsilon^{18})$. In fact, the dependence of L on ϵ can be reduced to nearly ϵ^{-2} (cf. Section 3.1), giving a bound of $\tilde{O}(ns^8/\epsilon^{12+c})$ for arbitrarily small constant c.

4.3 Implementation considerations

While the Harmonic Sieve is a polynomial-time algorithm, there is clearly much room for improvement of its time bound. So far, we have concentrated on presenting a conceptually simple and readily extensible algorithm rather than on attempting to optimize its computational complexity. In this section we make several brief observations that might be helpful in implementing a version of the algorithm.

First, there are alternatives to the boosting algorithm we described. In particular, Freund has also developed a similar and more efficient (in the distribution-independent setting) boosting algorithm [22, 23] that can also be used to learn DNF. We have chosen to detail `F1` for two reasons. First, the later algorithm (call it `F2`) is slightly more involved in that it may choose not to call the weak learner at all at some stages of the boosting process; therefore, clarity of exposition favors F1. Second, F2 produces distributions D_i which may deviate from uniform more than the distributions produced by F1. In turn, this deviation from uniform plays a role in the time bound on the Harmonic Sieve, as shown above. Therefore, the F2 algorithm does not offer a clear performance advantage over F1 in the context of the Harmonic Sieve. However, if efficient implementation of the Harmonic Sieve for a particular application is an issue, both boosting algorithms should be considered. On the other hand, Schapire's original boosting algorithm [44] is probably not an appropriate basis for HS, as it is not clear how to efficiently simulate oracles for the distributions defined by this algorithm.

Recently, Freund and Schapire have developed yet another boosting algorithm, AdaBoost [24]. AdaBoost is designed specifically for "boosting-by-sampling" as opposed to the "boosting-by-filtering" technique employed by F1. A boosting-by-filtering algorithm defines each probability distribution D_i by filtering examples received from the oracle $EX(f, D)$. In contrast, in boosting-by-sampling a set S of examples is drawn by the boosting algorithm from $EX(f, D)$ once at the beginning of learning. Given an appropriate weak learner, the boosting algorithm then efficiently finds a hypothesis h that well-approximates the target f with respect to the distribution D' defined to be uniform over the set S and having zero weight on all instances not represented in S. For sufficiently large S, this hypothesis will with high probability also well-approximate f with respect to the true target distribution D.

In practice, the notion of finding a hypothesis consistent with a given data set plays an important role in many learning algorithms. Thus from an applied point of view, it could be useful to base the Harmonic Sieve on a boosting-by-sampling algorithm. In fact, any boosting-by-filtering algorithm can be converted to a boosting-by-sampling algorithm (the basic idea is to simulate an example oracle $EX(f, D')$). However, AdaBoost, which is designed with boosting-by-sampling in mind, has potential practical advantages over earlier boosting algorithms. Primarily, AdaBoost is *adaptive* (this is what the "Ada" in AdaBoost stands for). Specifically, the distributions D_i created by AdaBoost depend on (adapt to) how well the weak hypotheses produced before stage i approximate the function f. Furthermore, AdaBoost uses a measure of how well each weak hypothesis approximates the target to assign weight to that weak hypothesis in the final strong hypothesis. As a result, we expect AdaBoost to require fewer boosting stages in general than would be required by any of the earlier boosting algorithms.

Unfortunately, we currently do not know how to obtain positive DNF learning-theoretic results using a boosting-by-sampling algorithm. The key problem is that the distributions D_i defined by `AdaBoost` are typically extremely nonuniform, and as noted before, the running time of `HS` depends directly on the nonuniformity of the target distribution. In particular, it appears to be very difficult in general to find a well-correlated parity with respect to very non-uniform distributions. However, in some applications it may be that n is so small that instead of using an approximation algorithm like `WP`$'$ to find the best-correlated parity, we could use exhaustive search. In fact, use of a multidimensional version of the discrete fast Fourier transform (see *e.g.* [1]) to compute all of the Fourier coefficients of a function makes this option practical for larger n than would be possible by a brute force enumeration of the coefficients. If `AdaBoost` is then used to boost the weak hypotheses found in this way, the size of the resulting strong hypothesis will depend polynomially on the size of the smallest DNF consistent with the data and only logarithmically on the size of the sample. Therefore, by a standard argument [12], given a sufficiently large (but polynomial-size) sample of the target DNF, the algorithm will with high probability produce a good approximator to the target with respect to the target distribution.

Another change that can be made is to the weak learner. Levin [39] has developed a clever randomized algorithm that given a membership oracle for any Boolean function f finds a Fourier coefficient of f of magnitude at least θ (if such a coefficient exists) in time $\tilde{O}(n^2/\theta^4)$ (the "soft-O" notation is the same as "big-O" notation except log factors are suppressed). Depending on the values of n and θ, this may be an improvement over `WP`, which runs in time $\tilde{O}(n/\theta^6)$. Like `WP`, this algorithm can be extended to find parity functions that correlate well with a target function f with respect to near-uniform distributions D; both algorithms have running times similarly dependent on the deviation of D from uniform. Furthermore, Levin was primarily interested in optimizing the query complexity of the algorithm, not its running time; it seems likely that the running time of the algorithm can be further improved.

4.4 Comparison with previous Fourier-based algorithms

We close this section by briefly comparing the Harmonic Sieve with previous Fourier-based learning algorithms (*e.g.*, [40, 37, 43]). All of these algorithms are similar in that they produce the same final hypotheses: thresholds of weighted parity functions (our hypotheses are weighted in that a particular parity χ_A may be chosen as the weak hypothesis at several different stages). However, in at least one aspect `HS` differs significantly from previous work. Typically, given appropriate oracles for a target function f and an accuracy ϵ, the goal of a Fourier-based learning algorithm has been to find a function h such that $L_2(f-h) \leq \epsilon$. An argument of Linial *et al.* then gives that sign(h) is an ϵ-approximator for f with respect to uniform.

More specifically, the standard approach has been to find a set S containing the largest Fourier coefficients of the target f and to let

$$h = \sum_{A \in S} \hat{f}(A)\chi_A.$$

However, Mansour [43] demonstrated that for DNF the set S must be superpolynomially large (in ϵ) in order to have $L_2(f-h) \leq \epsilon$, even for reasonably simple DNF expressions f. This provided some evidence that DNF might not be learnable by Fourier techniques that rely solely on finding parities that correlate well with the target function with respect to the uniform distribution on inputs.

The Harmonic Sieve escapes this potential difficulty by introducing a new approach to Fourier-based learning: find large Fourier coefficients of functions other than the target (the functions $g_i = 2^n f D_i$ discussed earlier). Alternatively, as pointed out above, this approach can be viewed as finding parity functions that correlate well with the target with respect to a variety of distributions. Although the distributions considered by HS are polynomially-near uniform in the sense defined earlier, even relatively small deviations from uniform can significantly affect how well a given parity approximates the target. The fact that HS finds parities that correlate well with the target over a range of distributions means that the algorithm can potentially find "important" parity functions that might be missed by earlier Fourier-based learning algorithms. Furthermore, even if all of the parity functions included in the final HS hypothesis have large Fourier coefficients, the weights of the parities in the HS hypothesis are not necessarily close estimates of Fourier coefficients, as was the case with previous algorithms.

However, it is possible that, say, the Goldreich-Levin algorithm run with an appropriate (inverse-polynomial) threshold could learn the same classes learnable by HS. That is, it may be that our DNF learnability result has more to do with the analytical approach taken than the HS algorithm itself. Whether or not this is the case is an interesting question for further research.

5 Learning Other Representation Classes

In this section we attempt to generalize our DNF learning result to other representation classes with Boolean inputs; our results are both positive and negative. Our primary positive result is that the representation class TOP of thresholds of parity functions is learnable by the Harmonic Sieve. We then discuss several classes that are restricted versions of TOP and that therefore are also learnable with respect to uniform. Finally, we consider two other representation classes—depth-2 threshold circuits and depth-3 Boolean circuits—that cannot be learned efficiently by the Harmonic Sieve.

5.1 Learning TOP

TOP, like DNF, is a universal class. Thus the learnability result we would like to show is that arbitrary Boolean functions are learnable in time polynomial in their TOP-size (recall that the TOP-size of a function is the number of parity functions in the smallest unweighted TOP representing the target function, or equivalently the minimal sum of the weights in any integer-weighted TOP circuit representing the function). To prove this, we first prove that all TOP's have a certain property: for every distribution D, every TOP f is weakly approximated with respect to D by some parity function. The analogous property for DNF (Fact 8) was the only property of DNF used in the proof that DNF is learnable. Thus, given this property for TOP we can immediately apply the proof of Theorem 10 to show that TOP is learnable with respect to uniform by `HS`. Moreover, TOP is properly learnable, since `HS` produces a TOP as its hypothesis.

The TOP property we seek is actually a corollary of a theorem of Goldmann, Hastad, and Razborov [28]. We give a relatively simple alternate proof similar to one of Blum *et al.* [10] for a more specific TOP property.

Fact 11 *For every f the majority of s parity functions (not necessarily distinct) and for every distribution D on the instance space of f, one of the parity functions χ_A defining f satisfies $|\mathbf{E}_D[f\chi_A]| \geq 1/(2s+1)$.*

Proof: We will think of each parity function in f as a function χ_A that outputs a value in $\{-1, +1\}$ based on the input to f. Thus, for all inputs x except those on which the parity "voters" are evenly split, $f(x)$ is simply the sign of the sum of the values of the parity functions on x. Furthermore, the voters can be evenly split only if s is even. But in this case it is straightforward to create a new circuit with $2s+1$ parity gates computing the same function (each of the initial parity gates is replicated and the constant parity $\chi_\emptyset$ is added).

Thus for any f representable as a majority of s parity functions there is some $F = \sum_A \hat{F}(A)\chi_A$ such that $f = \mathrm{sign}(F)$ (where sign(0) is undefined), $L_1(\hat{F}) \leq 2s+1$, and all the coefficients $\hat{F}(A)$ of F are integers. Specifically, this F is simply the sum of the parity functions defining f (or defining the equivalent circuit with an odd number of parity functions constructed above if s is even). Note that $|F(x)| \geq 1$ for all x since F has integer coefficients and is always nonzero, and since $f = \mathrm{sign}(F)$, we have

$$\begin{aligned} 1 &\leq \mathbf{E}_D[|F|] = \mathbf{E}_D[fF] = \mathbf{E}_D[f \sum_A \hat{F}(A)\chi_A] = \sum_A \hat{F}(A)\mathbf{E}_D[f\chi_A] \\ &\leq \max_A\{|\mathbf{E}_D[f\chi_A]|\} \sum_A |\hat{F}(A)|. \end{aligned}$$

□

Theorem 12 *TOP is properly learnable with respect to uniform using membership queries.*

5.2 Some specialized TOP classes

TOP is a fairly expressive class of representations. For example, a corollary of a result of Krause and Pudlák is that for every Boolean function f, the TOP-size of f is at most polynomially larger than the DNF-size of f [36]. Actually, we get as a corollary of our learning result an alternate proof of this relationship (this method of obtaining complexity results from boosting results has been noted before [23]). Our bound on the size of the resulting TOP is the same as that obtained by Krause and Pudlák.

Corollary 13 *Every DNF f with s terms can be computed as the majority of $O(ns^2)$ parity functions.*

Proof: Given a DNF function f, imagine running `HS` with a simulated membership oracle for f, with $\epsilon = 2^{-(n+1)}$, with $\delta < 1$, and with D the uniform distribution. While `HS` may not run efficiently in n, note that by the proof of Theorem 10 the algorithm will with nonzero probability produce a hypothesis h with the following properties:

1. h is the majority of $k = O(ns^2)$ parities.

2. $\mathbf{Pr}[h \neq f] < 2^{-n}$.

For the second property to hold, it must be that $h \equiv f$. Since the algorithm succeeds—with positive probability—at finding an h with these properties, such an h must exist. □

Before going further, we need some notation. In particular, we have previously used the notion of a subclass of a representation class; now we generalize this notion somewhat. We will view a representation class $\mathcal{F}$ as a *restriction* (or a *specialization*) of a class $\mathcal{H}$ if there exists a fixed polynomial p such that for every n and every Boolean f over $\{0,1\}^n$, $\mathcal{H}$-size of f is at most $p(n,s)$, where s is the $\mathcal{F}$-size of f. Given such an $\mathcal{F}$ and $\mathcal{H}$, learnability of $\mathcal{H}$ implies non-proper learnability of $\mathcal{F}$. In other words, the restricted class, while it may be capable of representing arbitrary Boolean functions, is from a learning perspective an easier class. We will also say that $\mathcal{H}$ *generalizes* $\mathcal{F}$ if $\mathcal{F}$ is a restriction of $\mathcal{H}$. If $\mathcal{H}$ generalizes $\mathcal{F}$ but not vice versa, then $\mathcal{H}$ *strictly* generalizes $\mathcal{F}$.

Thus what we have shown above is that DNF is a restricted class of TOP. In fact, the restriction is strict, as n-bit parity has only an exponentially large DNF representation. In addition to DNF, TOP strictly generalizes another universal class, this class having the property that the size of a function f is measured not in terms of how f is expressed but instead in terms of a Fourier property of the function f itself. In particular, define the L_1*-size* of a Boolean function f to be $L_1(\hat{f})$, that is, the sum of the magnitudes of the Fourier coefficients of f, and let CL_1 be the class of arbitrary Boolean functions with this size measure. Bruck and Smolensky [16] have shown that TOP strictly generalizes CL_1. In fact, CL_1 is the class that Kushilevitz and Mansour showed to be efficiently learnable with respect to uniform using membership queries; the fact that the class of parity decision trees is similarly learnable is a corollary of the fact that this class specializes CL_1 [37]. Because CL_1 in turn specializes TOP, `HS` can efficiently learn—with respect to uniform and using membership queries—every class previously shown learnable by Kushilevitz and Mansour.

As another example of a restricted TOP class (and therefore a class learnable with respect to uniform by the Harmonic Sieve), consider the class of functions which can be represented as a majority vote of arbitrary Boolean functions each of fanin $O(\log n)$. We define the size of a representation in this class to be the number of (unweighted) functions voting. To see that this class specializes TOP, consider the Fourier transform of a Boolean function g defined over a subset S of the input bits. It follows from the definition of the Fourier transform that the value of every Fourier coefficient $\hat{g}(A)$ of such a g can be expressed exactly by a rational number with denominator $2^{|S|}$, and that a specific coefficient $\hat{g}(A)$ is non-zero only if $A \subseteq S$ (cf. the example of Section 2.3). This latter fact in turn implies that g has at most $2^{|S|}$ non-zero Fourier coefficients. So, for example, if g is a function over three input bits and χ_A is a parity function over any four input bits then $\hat{g}(A)$ will be zero.

Putting the above facts together, it can be seen that the integer-valued function $2^{|S|}g$ is expressible exactly as the (unweighted) sum of $O(2^{2|S|})$ parity functions χ_A, at most $O(2^{|S|})$ of which are distinct. So given a threshold of s Boolean functions g_i each on $O(\log n)$ inputs, we can create for each g_i a polynomial-size (in n) sum of parity functions computing $2^c g_i$, where c is the maximum fanin of any g_i. Taking a threshold over these sums gives a threshold over polynomially many (in n and s) parity functions that computes the original function.

As a final example of a class that specializes TOP, consider the class parity-DNF of representations that are depth-3 circuits with an OR at the top level, AND's in the middle, and parity functions at the bottom. Parity-DNF also specializes TOP. The high level idea behind this observation is the following. The Fourier representation of a conjunction of parities is very similar to a conjunction of literals: it is roughly an "average" of a collection of parity functions plus a constant. Thus we can prove, much as we did for DNF, that for every parity-DNF f and every distribution D there is a parity that weakly approximates f with respect to D. Therefore, the proof that DNF is learnable as TOP can readily be generalized to show that parity-DNF is also learnable as TOP. Then the proof of Corollary 13 shows that parity-DNF is a restriction of TOP.

5.3 Two classes not generalized by TOP

The Harmonic Sieve algorithm does not readily extend to learning two representation classes that are seemingly small generalizations of DNF and TOP. In particular, Bruck [15] has shown that there is a polynomial-size (in n) depth-2 threshold circuit f_1 such that $L_\infty(\hat{f}_1) = 2^{-n/2}$, and an argument of Bruck and Smolensky [16] can be used to construct a polynomial-size depth-3 $\{\wedge, \vee, \neg\}$-circuit f_2 such that $L_\infty(\hat{f}_2) = n^{-\log n}$. Let s be the size of a target function f according to the size measure of either of these two representation classes. Then the Fourier characterizations above say that for each of these classes there are functions f such that *all* of the Fourier coefficients of f are superpolynomially (in s) small, or equivalently, that *no* parity is correlated inverse-polynomially in s with f with respect to the uniform distribution. Thus the $\mathtt{WP}'$ procedure invoked by $\mathtt{HS}$ to find the largest Fourier coefficient of a target function will need an inverse superpolynomially small threshold value θ. The running time of $\mathtt{WP}'$, in turn, is bounded inverse-polynomially in θ, and therefore is not polynomially bounded for these classes. Furthermore, because the maximum magnitude of any Fourier coefficient $\hat{f}(A)$ of a function f upper bounds the inverse of the TOP-size of f (Fact 11), these classes are not restrictions of TOP.

As an example of how these hardness results are obtained, we give the depth-3 circuit construction here. First, it can be shown that the Fourier representation of the Boolean function $\wedge(x) = x_1 \wedge x_2$ is

$$\wedge(x) = -\frac{1}{2}\left(1 + \chi_{\{1\}}(x) + \chi_{\{2\}}(x) - \chi_{\{1,2\}}(x)\right).$$

Now consider a circuit that consists of a parity gate that takes a set $\{C_i\}$ of subcircuits as input. We can obtain an arithmetic representation of the function computed by this circuit by taking the product of the Fourier transforms of the subcircuits C_i. Also, note that the product $\chi_A \cdot \chi_B = \chi_{A \Delta B}$, where Δ denotes symmetric difference. Therefore, the parity of $\log n$ 2-bit $\wedge$ functions, where the $\wedge$'s are applied to pairwise-disjoint sets of variables, has a Fourier transform in which every nonzero coefficient has magnitude n^{-1}. On the other hand, this is a function on $2 \log n$ bits, and therefore it can be represented either as a CNF with at most n^2 clauses or as a DNF with at most n^2 terms.

Let g_i represent the parity of $\log n$ 2-bit $\wedge$ functions over pairwise-disjoint variables, and let S_i represent the set of variables in g_i. Also, let g represent the function formed by taking the parity of a set of $\log n$ functions g_i where each g_i is defined over a set of variables S_i that is disjoint from all the other sets. Then, by reasoning similar to the above, the nonzero Fourier coefficients of the resulting function will all have magnitude $n^{-\log n}$. Furthermore, the function can be written as a polynomial-size DNF over inputs that are polynomial-size CNF's, which is a polynomial-size depth-3 circuit.

In the next section we consider learnability of a representation class that generalizes DNF in a different way, moving from functions over $\{0,1\}^n$ to functions over more general hypercubes. We show that `HS` can be extended to learn this more general class.

6 Learning unions of axis-parallel rectangles

In this section we describe a generalization of the `HS` algorithm that learns certain classes of functions on non-Boolean domains. First, note that the positive instances of an s-term DNF f are covered exactly by the union of s subcubes of the Boolean hypercube $\{0,1\}^n$. Based on this view of DNF, we can define a natural generalization of DNF over the domain $[b]^n$ (we use $[b]$ to represent $\{0,\ldots,b-1\}$). For l and u in $[b]^n$ such that $\ell_i \leq u_i$ for all i in $\{1,\ldots,n\}$, the *rectangle* $[\ell, u]$ on $[b]^n$ is the set of instances $\prod_{i=1}^{n}\{\ell_i, \ell_i+1, \ldots, u_i\}$. A UBOX representation r is a union of rectangles and the *size* of a UBOX representation r is the number of rectangles in r. The function

$f : [b]^n \to \{-1, +1\}$ represented by such an r is the function whose positive instances are covered exactly by r. The *UBOX-size* of a function $f : [b]^n \to \{-1, +1\}$ is the size of the smallest UBOX representation of f. It should be noted that the definition given here deviates from standard notation for geometric concepts (see, *e.g.*, [20]). Typically, the dimension n of the instance space is denoted by d, the number of attribute values b is denoted by n, and the number of rectangles s is denoted by m. We have chosen to use this nonstandard notation for consistency with the DNF portion of this paper.

We will show that if b is a constant then UBOX is learnable with respect to the uniform distribution using membership queries in time polynomial in the usual parameters. This complements results in stronger models of learning showing polynomial-time learnability for UBOX when either the dimension n or the number of boxes s is held constant [13, 42, 26, 20].

We begin by showing how to extend the Harmonic Sieve algorithm to learn a subclass of UBOX that generalizes the class k-DNF of DNF expressions where each term has at most k literals. A *k-rectangle* $[\ell, u]$ on $[b]^n$ is a rectangle such that $|\{i \mid \ell_i \neq 0 \text{ or } u_i \neq b-1\}| \leq k$. k-UBOX is the restriction of UBOX to unions of k-rectangles. After showing that k-UBOX is learnable with respect to the uniform distribution for restricted values of k, we show that the same algorithm can learn unrestricted UBOX in time exponential in $b \log \log b$ and thus in polynomial time for constant b.

Before developing these algorithms, we show how to generalize the `WP` algorithm for learning over the domain $[b]^n$.

6.1 Generalizing `WP`

The UBOX algorithm we develop will, like `HS`, build on an extension of `WP`. In particular, the Fourier analysis that underlies the `WP` algorithm can be generalized to produce an algorithm that finds the large Fourier coefficients of a Boolean function on $[b]^n$. The Fourier basis functions are now of the form

$$\chi_a(x) = \omega_b^{\sum_i a_i x_i},$$

where $a \in [b]^n$ and $\omega_b = e^{2\pi\sqrt{-1}/b}$. These functions form a basis for the space of all complex-valued functions on $[b]^n$, and every function $f : [b]^n \to \mathbf{C}$ can be uniquely expressed as a linear combination of the χ functions: $f = \sum_a \hat{f}(a)\chi_a^*$, where χ^* represents the complex conjugate of χ and $\hat{f}(a) = \mathbf{E}[f\chi_a]$. Also, for such functions f

Parseval's identity holds, although now it is expressed in terms of magnitudes:

$$\mathbf{E}_x\left[|f(x)|^2\right] = \sum_a \left|\hat{f}(a)\right|^2.$$

Finally, the basis functions are orthonormal in the sense that

$$\mathbf{E}_x\left[\chi_a(x)\cdot\chi_b^*(x)\right] = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise.} \end{cases}$$

To find the large Fourier coefficients of a function $f : [b]^n \to \{-1, +1\}$, we first define the collection $C_{a,k}$, where k is an integer in $[0, n]$ and a is a vector in $[b]^k$, by generalizing the earlier definition of $C_{A,k}$ (Section 3.2):

$$C_{a,k} = \left\{\hat{f}(ad) \mid d \in [b]^{n-k}\right\},$$

where ad denotes the concatenation of the strings a and d. Then the Fourier analysis underlying the `WP` algorithm can be generalized to show that for every f, k, and a as above,

$$L_2^2(C_{a,k}) = \mathbf{E}_{x,y,z}[\mathrm{Re}(f^*(yx)f(zx)\chi_a(y-z))], \tag{3}$$

where $L_2^2(C_{a,k})$ represents the sum of squares of the magnitudes of the elements of $C_{a,k}$; the expectation is uniform over $x \in [b]^{n-k}$, $y \in [b]^k$, and $z \in [b]^k$; and $y - z$ represents the difference between y and z in $\mathbf{Z}_b^k$.

To derive this equation, fix k, let a be any vector in $[b]^k$, and let x be any vector in $[b]^{n-k}$. Then we define $f_a(x)$ in terms of a subset of the Fourier coefficients of f:

$$f_a(x) = \sum_d \hat{f}(ad)\chi_d^*(x)$$

where the sum is over all vectors $d \in [b]^{n-k}$. Then by Parseval's, $\mathbf{E}_x[|f_a(x)|^2] = L_2^2(C_{a,k})$. Next note that

$$\begin{aligned} \mathbf{E}_{y\in[b]^k}[f(yx)\chi_a^*(y)] &= \mathbf{E}_y\left[\sum_{c\in[b]^k, d\in[b]^{n-k}} \hat{f}(cd)\chi_{cd}^*(yx)\chi_a^*(y)\right] \\ &= \sum_{c,d}\hat{f}(cd)\chi_d^*(x)\mathbf{E}_y\left[\chi_c(y)\chi_a^*(y)\right] = \sum_d \hat{f}(ad)\chi_d^*(x) \\ &= f_a(x). \end{aligned}$$

Equation 3 follows by some algebraic manipulations and simplification.

The generalized algorithm, then, is exactly the same as `WP`, except it uses the relationship above to estimate $L_2^2(C_{a,k})$ for b collections (one for each possible value of a_k) during each recursive call. This generalized `WP` runs in time polynomial in b as well as in the original parameters, specifically $\tilde{O}(nb/\theta^6)$. `WP'` can be similarly generalized.

6.2 Learning k-UBOX

The main result of this section is that k-UBOX is learnable for certain values of k. To obtain this result, we first generalize Fact 8 to the k-UBOX class. Then we will show how to convert the basis functions found by the generalized `WP′` outlined above into Boolean weak hypotheses.

6.2.1 A fact about k-UBOX

Now we show that for every k-UBOX function f (with k appropriately limited) and for every distribution D there is some Fourier basis function χ_a that correlates well with f with respect to D.

Fact 14 *Let $f : [b]^n \to \{-1, +1\}$ be any union of s k-rectangles, and let D be any probability distribution on $[b]^n$. Then there is some χ_a such that $|\mathbf{E}_D[f\chi_a]| = \Omega(s^{-1}\log^{-k} b)$.*

Proof: We consider two cases. The simple case is when $|\mathbf{E}_D[f]| \geq 1/2s$. In this case, the constant function $\chi_{\bar{0}}$ or its negation satisfies the lemma.

Otherwise, let $p = \mathbf{Pr}_D[f = +1]$. Note that for this case of the proof we have $|\mathbf{E}_D[f]| \leq 1/2s$, and thus $p \leq 1/2 + 1/4s$. Also, there is some rectangle R such that $\mathbf{Pr}_D[x \in R] \geq p/s$. Let $R(x)$ be the Boolean function represented by R, *i.e.*, $R(x) = 1$ when x is in rectangle R and $R(x) = -1$ otherwise. Then, since $f = -1$ implies $R = -1$, $\mathbf{Pr}_D[R = f] \geq (1 - p) + p/s$. It follows that $\mathbf{E}_D[fR] \geq 1/2s$. Also, by an argument similar to one in the proof of Fact 11, $\mathbf{E}_D[fR] \leq \max_a\{|\mathbf{E}_D[f\chi_a]|\} \cdot \sum_a |\widehat{R}(a)|$. Thus the proof is complete once we show that $L_1(\widehat{R}) = O(\log^k b)$.

To see that $L_1(\widehat{R})$ is thus bounded, first observe that the function $(R(x) + 1)/2$ can be written as the product of at most k functions $R_i(x)$, where each $R_i(x)$ is a 1-rectangle with range $\{0, 1\}$. Moreover, it can be shown that $L_1(\widehat{R}) \leq 2\prod_i L_1(\widehat{R}_i)$ [37]. Thus $L_1(\widehat{R}) = O((\max L_1(\widehat{R}_i))^k)$, where the maximum is over all possible 1-rectangles R_i producing outputs in $\{0, 1\}$. In fact, this maximum is the same as the maximum over all 1-rectangles $R_i^1 : [b] \to \{0, 1\}$, since the value of a 1-rectangle is determined by a single variable. In the appendix we show that $\max L_1(\widehat{R}_i^1) = O(\log b)$. □

6.2.2 Turning χ_a into a Weak Approximator

Given the above fact and the generalization of WP$'$ outlined earlier, we can also generalize the other components of the HS algorithm to learn k-UBOX efficiently for certain k. A key difference between the original and generalized HS algorithms is in the form of the weak hypothesis produced at each stage of the algorithm. Unlike the original HS, which produced a χ_A that could be used directly as the weak hypothesis, the generalized HS finds a complex-valued function χ_a at each stage of the boosting process. The proof of the following lemma describes an algorithm for converting such a χ_a into a weak approximator for Boolean f.

Lemma 15 *There is an algorithm* TABLE *such that for any function $f : [b]^n \to \{-1, +1\}$, for any probability distribution D over $[b]^n$, for any $a \in [b]^n$, and for any positive δ,* TABLE$(EX(f,D), a, \delta)$ *returns, with probability at least $1-\delta$, a Boolean hypothesis h_a such that $\mathbf{Pr}_D[h_a \neq f] = 1/2 - \Omega(|\mathbf{E}_D[f\chi_a]|)$. The algorithm runs in time polynomial in n, b, $|\mathbf{E}_D[f\chi_a]|^{-1}$, and $\log(\delta^{-1})$.*

Thus the weak learner for k-UBOX consists of running the generalized WP$'$ followed by running TABLE, where the inputs to TABLE are the index a of the function χ_a found by WP$'$ and the simulated oracle $EX(f, D_i)$ defined by stage i of F1.

Proof of Lemma: TABLE, as the name implies, produces a hypothesis h_a that consists of a table of Boolean entries. In particular, for each $j \in [b]$, the table contains a single entry $h_a(j)$ that defines $h_a(x)$ for all x such that $\chi_a(x) = \omega_b^j$. Thus

$$\mathbf{Pr}_{x\in_R D}[h_a(x) \neq f(x)] = \sum_{j=0}^{b-1} \mathbf{Pr}_{x\in_R D}[h_a(x) \neq f(x) \mid \chi_a(x) = \omega_b^j] \cdot \mathbf{Pr}_{x\in_R D}[\chi_a(x) = \omega_b^j].$$

Let D^j be defined by

$$D^j(x) = \begin{cases} 0 & \text{if } \chi_a(x) \neq \omega_b^j \\ \frac{D(x)}{\mathbf{Pr}_D[\chi_a = \omega_b^j]} & \text{otherwise.} \end{cases}$$

Then

$$\begin{aligned} \mathbf{Pr}_{x\in_R D}[h_a(x) \neq f(x)] &= \sum_{j=0}^{b-1} \mathbf{Pr}_{x\in_R D^j}[h_a(j) \neq f(x)] \cdot \mathbf{Pr}_{x\in_R D}[\chi_a(x) = \omega_b^j] \\ &= \sum_{j=0}^{b-1} \left(\frac{1}{2} - \frac{\mathbf{E}_{x\in_R D^j}[h_a(j) \cdot f(x)]}{2} \right) \cdot \mathbf{Pr}_{x\in_R D}[\chi_a(x) = \omega_b^j] \\ &= \frac{1}{2} - \frac{1}{2} \sum_{j=0}^{b-1} h_a(j) \cdot \mathbf{E}_{x\in_R D^j}[f(x)] \cdot \mathbf{Pr}_{x\in_R D}[\chi_a(x) = \omega_b^j]. \end{aligned}$$

Now note that

$$\mathbf{E}_D[f\chi_a] = \sum_{j=0}^{b-1} \mathbf{E}_{D^j}[f] \cdot \omega_b^j \cdot \mathbf{Pr}_{x \in_R D}[\chi_a(x) = \omega_b^j].$$

and therefore

$$|\mathbf{E}_D[f\chi_a]| \le \sum_j |\mathbf{E}_{D^j}[f]| \cdot \mathbf{Pr}_{x \in D}[\chi_a(x) = \omega_b^j].$$

Define $h'_a(j) = \mathrm{sign}(\mathbf{E}_{D^j}[f])$. Then

$$\mathbf{Pr}_D[h'_a \ne f] \le \frac{1}{2} - \frac{1}{2}|\mathbf{E}_D[f\chi_a]|$$

and h'_a is a weak approximator to f for all a such that $|\mathbf{E}_D[f\chi_a]|$ is inverse-polynomially large. Furthermore, for h''_a defined by

$$h''_a(j) = \begin{cases} -h'_a(j) & \text{if } |\mathbf{E}_{D^j}[f]| \le \frac{|\mathbf{E}_D[f\chi_a]|}{8} \text{ or } \mathbf{Pr}_{x \in_R D}[\chi_a(x) = \omega_b^j] \le \frac{|\mathbf{E}_D[f\chi_a]|}{8b} \\ h'_a(j) & \text{otherwise} \end{cases}$$

it follows that

$$\mathbf{Pr}_D[h''_a \ne f] \le \frac{1}{2} - \frac{1}{4}|\mathbf{E}_D[f\chi_a]|$$

This analysis leads to the following algorithm for producing a Boolean approximator h_a from function χ_a. Let $\theta = |\mathbf{E}_D[f\chi_a]|$. Estimate $\mu'_j = |\mathbf{E}_{D^j}[f]|$ for all j such that $|\mathbf{E}_{D^j}[f]| \ge \theta/8$ and such that $\mathbf{Pr}_D[\chi_a = \omega_b^j] \ge \theta/8b$ (call these the *large* j). Define $h_a(j) = \mathrm{sign}(\mu'_j)$ for all j as above and define the value of $h_a(j)$ arbitrarily for the remaining j. Then if the estimates for μ'_j are all within $\theta/8$ of the true values, h_a is at least a $(1/2 - \theta/4)$-approximator to f.

We now briefly outline a method of efficiently finding appropriate μ'_j for all the large j. First, we will estimate θ using calls to $EX(f, D)$. Note that by Corollary 5, we can estimate θ to within a small multiplicative factor with probability at least $1 - \delta$ in time polynomial in θ^{-1} and $\log \delta^{-1}$. Given such a bound on θ, we can by Hoeffding (Lemma 2) compute the number m_1 of examples from a simulated oracle $EX(f, D^j)$ necessary to accurately and with high probability estimate μ'_j for j such that $|\mathbf{E}_{D^j}[f]| \ge \theta/8$. We can also compute a necessary number m_2 of examples from $EX(f, D)$ such that with high probability, for every j such that $\mathbf{Pr}_D[\chi_a = \omega_b^j] \ge \theta/8b$, at least m_1 of the examples $\langle x, f(x) \rangle$ will satisfy $\chi_a(x) = \omega_b^j$. It can be shown that a value of m_2 polynomial in b, θ^{-1}, and $\log \delta^{-1}$ is sufficient to produce sufficiently accurate estimates of μ'_j for all large j. $\square$

6.2.3 Main Result

Putting this all together, we obtain the following:

Theorem 16 *For p any fixed polynomial and $k = O(\log p(n,s)/\log\log b)$, k-UBOX is learnable with respect to the uniform distribution using membership queries. The algorithm runs in time polynomial in b, n, s, ϵ^{-1}, and* $\log(\delta^{-1})$.

6.3 Learning UBOX

In fact, unrestricted UBOX is learnable in time polynomial in all parameters except b, and thus is efficiently learnable for b constant.

Corollary 17 *UBOX is learnable with respect to the uniform distribution using membership queries in time polynomial in $(s/\epsilon)^{O(b \log\log b)}$, n, and* $\log(\delta^{-1})$. *Here s represents the UBOX-size of the target function.*

Proof Sketch: First note that for any $t \geq 1$, if $k \geq \log_{b/b-1}(t)$ and R is a k-rectangle, then $\mathbf{Pr}[x \in R] \leq t^{-1}$, where the probability here is uniform over $[b]^n$. Now let $t = 12s/c_1\epsilon^3$, where c_1 is the constant $1/57$ appearing in `HS`. Then for any k-rectangle R such that $k \geq \log_{b/b-1}(t)$, $\mathbf{Pr}[x \in R] \leq c_1\epsilon^3/12s$. Also, for such an R and for any distribution D simulated by `HS` in the process of boosting with respect to uniform (call this an `F1`-uniform distribution), $\mathbf{Pr}_D[x \in R] \leq 1/4s$. Now the rectangle R considered in the proof of Fact 14 was such that $\mathbf{Pr}_D[x \in R] \geq p/s$, where $p \geq 1/2 - 1/4s$ (since $|\mathbf{E}_D[f]| \leq 1/2s$). Thus, if we apply Fact 14 to `F1`-uniform distributions only then the rectangle R in the proof of Fact 14 must be a k rectangle with $k \leq \log_{b/b-1}(t)$. This means that for *any* function $f : [b]^n \to \{-1,+1\}$ and for any `F1`-uniform distribution D there is some χ_a such that $|\mathbf{E}_D[f\chi_a]| = \Omega(s^{-1} \log^{-\log_{b/b-1} t} b) = \Omega(s^{-1} t^{-O(b \log\log b)})$. Thus, applying the `F1` booster to the TABLE algorithm of Lemma 15 produces a learning algorithm for UBOX having the claimed time bound. $\square$

7 Learning with respect to nonuniform distributions

In this section we describe several methods for learning DNF with respect to distributions other than uniform.

7.1 Learning with respect to "nearly" uniform distributions

First, note that we have already laid the groundwork for a certain amount of generalization. Recall that the proof of Theorem 10 actually shows that DNF is learnable with respect to any distribution D such that an oracle for

D is available and $L_\infty(2^n D)$ is bounded by a polynomial in n. Actually, we do not even require an oracle for such a D, because if we learn an $(\epsilon/L_\infty(2^n D))$-approximator h with respect to uniform then h is also an ϵ-approximator with respect to D.

Furthermore, we can relax the need for a polynomial bound on $L_\infty(2^n D)$. Recall that $\mathtt{WP}'$ runs in time polynomial in, among other parameters, $L_\infty(g)$, where g is the (potentially non-Boolean) function for which a large Fourier coefficient is desired. The bound on the magnitude of g is used in $\mathtt{WP}'$ in two ways:

1. The number of samples needed to with high probability accurately estimate

$$\mathbf{E}_{x,y,z}[g(yx)g(zx)\chi_A(y \oplus z)] \tag{4}$$

   can be bounded, using Hoeffding's inequality, by a function that is polynomial in $L_\infty(g)$ along with other parameters.

2. The number of recursive calls made by $\mathtt{WP}'$ is bounded by a polynomial in, among other parameters, $\mathbf{E}[g^2]$. This quantity in turn is bounded by $L_\infty^2(g)$.

A number of samples sufficient to accurately estimate the expectation in (4) can be computed using Chebyschev rather than Hoeffding. To apply Chebyschev, we need to bound the variance σ_p^2 of the function

$$p(x, y, z) = g(yx)g(zx)\chi_A(y \oplus z).$$

We can bound σ_p^2 in terms of the variance σ_g^2 of g, since

$$\sigma_p^2 = \mathbf{E}[p^2] - \mathbf{E}^2[p] \le \mathbf{E}[p^2] = \mathbf{E}_x[\mathbf{E}_y^2[g^2(yx)]] \le \mathbf{E}_{x,y}[g^4(yx)]$$

where the final inequality follows from the fact that for any real-valued function f, $\mathbf{E}^2[f] \le \mathbf{E}[f^2]$. So $\mathbf{E}[g^4]$ upper bounds σ_p^2. Furthermore, $\mathbf{E}[g^4] + 1$ also upper bounds $\mathbf{E}[g^2]$. This means that in both items above a bound on $\mathbf{E}[g^4]$ can be substituted for the bound on $L_\infty(g)$. However, to achieve a confidence of $1 - \delta$ that the estimated value of (4) is sufficiently close to the true value requires a number of examples polynomial in δ^{-1} rather than the $\log \delta^{-1}$ required when Hoeffding was applied.

Recalling that $\mathtt{HS}$ runs $\mathtt{WP}'$ on g's of the form $2^n f D$ where $f \in \{-1, +1\}$, this means that a modified version of the algorithm can efficiently learn DNF with respect to any distribution D for which it is given an oracle and such that $\mathbf{E}[(2^n D)^4]$ is polynomially-bounded in n. Finally, by running the algorithm multiple times with a confidence of $\frac{1}{2}$, the running time can again be made polynomial in $\log \delta^{-1}$.

7.2 Learning with respect to product distributions

We can also learn DNF with respect to certain product distributions. A *product distribution* on $\{0,1\}^n$ is defined by a set $\{\mu_i \in \mathbf{R} \mid 1 \leq i \leq n\}$. Each μ_i represents the probability that input bit i is a 1; this probability is independent of the values of all the other bits. Product distributions are a natural generalization of the uniform distribution, which is the product distribution having $\mu_i = \frac{1}{2}$ for all i. They are also distinct from the distribution classes considered above. For example, $\mathbf{E}[(2^n D_{1/4})^4] > 2^n$ for the product distribution $D_{1/4}$ in which $\mu_i = 1/4$ for all i.

Because of the independence between variables inherent in product distributions, standard Fourier analysis readily generalizes to these distributions [25]. The primary difference is that the functions χ_A are replaced by a generalized Fourier basis consisting of functions ϕ_A, and Fourier coefficients are now defined with respect to this new basis. Specifically, let D be a product distribution with μ_i as above. Then for all $A \subseteq \{1, \ldots, n\}$ we define

$$\phi_A(x) = \prod_{i \in A} \frac{\mu_i - x_i}{\sqrt{\mu_i(1-\mu_i)}}.$$

This ϕ basis is orthonormal with respect to the inner product $\langle f, g \rangle = \mathbf{E}_D[fg]$. Therefore, if for $f : \{0,1\}^n \to \mathbf{R}$ we define $\tilde{f}(A) = \mathbf{E}_D[f\phi_A]$ then

$$f = \sum_A \tilde{f}(A)\phi_A.$$

It is straightforward to verify that when D is the uniform distribution, $\phi_A = \chi_A$ and $\tilde{f}(A) = \hat{f}(A)$ for all A.

Building on this generalized Fourier transform, Bellare has described a modified `WP` algorithm that finds all the large generalized Fourier coefficients of a function with respect to certain product distributions [9]. We can use these ideas to extend `HS` and show the following.

Definition 18 *The* c-bounded product distribution family $\mathcal{PD}_c$ *is the set of all product distributions D such that for every μ_i defining D, $c \leq \mu_i \leq 1 - c$.*

Theorem 19 *For any constant $c \in (0, 1/2]$, DNF is learnable with respect to $\mathcal{PD}_c$ using membership queries.*

Proof Sketch: Let f be the target DNF function and D the target product distribution. We will learn f with respect to D by running `HS` with a modified weak learner. Specifically, we will show that for every D_i defined by `F1` in the process of learning with respect to D we can efficiently and with high probability find a set A such that $|\mathbf{E}_{D_i}[f\phi_A]|$ is inverse polynomially large. It then will follow by a table technique similar to the one outlined in the proof of Lemma 15 that ϕ_A can be converted to a good Boolean weak approximator for f with respect to D_i.

The first step is to show that such a ϕ_A exists. By the proof of Fact 8 we know that for any s-term DNF f either $\mathbf{Pr}_D[f=1] \leq 1/3$, in which case the constant function $\phi_\emptyset$ is a good weak approximator to f, or there exists a term T of f with the following properties:

- $\mathbf{Pr}_D[T \text{ is satisfied}] \geq \frac{1}{3s}$, and
- At least one subset A of the variables in T is such that $|\mathbf{E}_D[f\chi_A]| \geq \frac{1}{2s+1}$.

Furthermore, since D is a c-bounded product distribution, T contains at most $\ln(3s)/c$ variables. Thus there is some parity χ_A over $O(\log(s))$ of the inputs which is a good weak approximator to f with respect to D. Also, for the distributions D_i generated by `F1` in the process of learning against D, $D_i(x) = O(D(x)/\epsilon^2)$ for all x. So for each of these distributions D_i there is some parity over $O(\log(s/\epsilon))$ inputs which is a good weak approximator with respect to D_i.

Now by reasoning similar to the proof of Fact 11, for A such that $|\mathbf{E}_{D_i}[f\chi_A]| \geq 1/(2s+1)$,

$$\frac{1}{2s+1} \leq \max_B |\mathbf{E}_{D_i}[f\phi_B]| \cdot \sum_B |\tilde{\chi}_A(B)|.$$

Because χ_A is a Boolean function and Parseval's identity holds for any orthonormal basis, we have that $\sum_B \tilde{\chi}_A^2(B) = 1$. Furthermore, it can be shown that $\tilde{\chi}_A(B) = 0$ unless $B \subseteq A$. By our earlier bound on the size of A and the relationship between L_1 and L_2 norms, this gives

$$\max_B |\mathbf{E}_{p_i}[f\phi_B]| \geq \left(\frac{\epsilon}{s}\right)^{O(1)}.$$

To find such a ϕ_B we use a modified `WP` algorithm. Define $C_{A,k} = \{\tilde{f}(A \cup E) \mid E \subseteq \{k+1, \ldots, n\}\}$. Then it can be shown that

$$L_2^2(C_{A,k}) = \mathbf{E}_{x,y,z}[f(yx)f(zx)\phi_A(y)\phi_A(z)].$$

For constant-bounded product distribution D and $|A| = O(\log(s/\epsilon))$, $|\phi_A(y)| = (s/\epsilon)^{O(1)}$ for all y. Since we know (modulo our usual guess of s) that there is some A that obeys this bound and such that ϕ_A is well-correlated with f, we modify the `WP` algorithm so that it "cuts off" any branch of the recursion that would consider an A exceeding this bound. Thus at every recursive call to the modified `WP` we can efficiently estimate the above expectation for Boolean f, and therefore we can efficiently find a ϕ_A that correlates well with f with respect to D.

Furthermore, if we define $g(x) = f(x)p_i(x)/p(x)$ then $\mathbf{E}_{p_i}[f\phi_B] = \mathbf{E}_p[g\phi_B]$. For D_i as defined by `F1` when learning with respect to D we have $|g(x)| = O(\epsilon^{-2})$ for all x. Therefore, for each of these p_i's we can efficiently find a ϕ_B that correlates well with f with respect to p_i. □

This result holds whether or not the learner is given an oracle for the target distribution D, since a very good approximation to a product distribution D can be obtained efficiently by using random sampling to estimate the μ_i's defining D.

8 Learning despite persistent classification noise

So far, we have assumed that the membership oracle $MEM(f)$ available to our learning algorithms is completely accurate. In this section we consider learning from membership oracles that randomly misclassify instances. Our main result is that the parity-finding technique of Goldreich and Levin is very tolerant of persistent noise in the classifications. Classification noise is *persistent* if once the membership oracle classifies an instance (either correctly or incorrectly), it consistently gives the same classification in response to all future queries about this instance. We get two interesting corollaries from this result. First, the class of polynomial-size DNF is weakly learnable (with very high probability) with respect to uniform despite substantial noise (a noise rate near $\frac{1}{2}$) in the membership oracle. Second, we generalize work of Kushilevitz and Mansour [37], showing that a minor modification of their algorithm for learning PL_1 (defined below) learns PL_1 from a persistently noisy membership oracle.

Earlier versions of this work [31, 32] contained an erroneous proof that our result could be extended to a strong noise-tolerant algorithm for DNF. However, that proof overlooked the fact that the boosting distributions D_i generated by `F1` are not only dependent on the target function and weak hypotheses produced, but are also noise-dependent. Eli Shamir [45] apparently first discovered this error, and at the time of this writing in joint work with Clara Shwartzman seems near producing a modified noise-tolerant algorithm for learning DNF with respect to uniform.

As indicated above, we focus on learning from a membership oracle that exhibits *persistent* classification noise [27]. In particular, for any Boolean f and noise rate η define the noisy oracle $MEM_{CN}^{\eta}(f)$ as follows. If the oracle has been queried about the instance x previously, it answers with the same value returned before. Otherwise, with probability $1-\eta$ the oracle returns $f(x)$, and with probability η it returns $-f(x)$. Given such a noisy membership oracle, the learning algorithm's goal is to find a hypothesis h that is an ϵ-approximation to the *true* target f, not to the noisy function simulated by the oracle. Note that for $\eta = \frac{1}{2}$, on any input x, $MEM_{CN}^{\eta}(f)$ produces positive and negative outputs with equal probability, independent of f; thus it is information theoretically impossible to learn f in this case. Therefore, we will assume in the sequel that $\eta < \frac{1}{2}$. In fact, following standard practice, we will say that a function class $\mathcal{F}$ is learnable despite persistent classification noise if $\mathcal{F}$ is learnable from an oracle MEM_{CN}^{η} in time depending polynomially on $1/(\frac{1}{2} - \eta) = O(1/(1-2\eta))$ as well as polynomially in the usual parameters.

It should be noted that learning in the presence of persistent noise, while similar to the well-studied model of PAC learning from an example oracle that exhibits classification noise without persistence, also has some marked differences. In the standard classification noise model the example oracle chooses an example at random according to the target distribution as in the noiseless PAC model. However, before returning the example, the oracle flips a biased coin and with probability η reverses the label of the example. Thus, if the oracle chooses the same instance x more than once (which it would do, for example, with high probability if the target distribution put a great deal of weight on x) then the learner might well see both the examples $\langle x, f(x)\rangle$ and $\langle x, -f(x)\rangle$. This will never happen in the persistent noise model.

Now consider learning a Boolean function f from a membership oracle that applies nonpersistent classification noise to examples. We can then ask m membership queries about a single instance x, receiving labels $\ell_1(x)$, $\ell_2(x)$, $\ldots$, $\ell_m(x)$. For sufficiently large m and assuming that the noise rate is less than $\frac{1}{2}$, with high probability $\operatorname{sign}(\sum_i \ell_i(x)) = f(x)$. Furthermore, by a Chernoff bound argument, the value of m required to achieve a probability $1-\delta$ is polynomial in $1/(1-2\eta)$ and $\log \delta^{-1}$. Therefore, if a function class $\mathcal{F}$ is learnable using membership queries, then $\mathcal{F}$ is learnable from a membership oracle that applies nonpersistent classification noise to the examples. Therefore, at least from a theoretical point of view, learning from a membership oracle exhibiting nonpersistent classification noise is essentially equivalent to learning from a noiseless oracle. Thus persistent noise is a more interesting noise model to study in the context of membership query learnability.

Note further that we cannot give PAC-style guarantees in the persistent noise model. Specifically, in the PAC model we are able to achieve any desired level of confidence that our output hypothesis is an ϵ-approximator to the target function, because additional sampling of the given oracles always improves our confidence. This is not the case in the persistent noise model: once a persistently noisy oracle has fixed a value for an example, its choice cannot be changed by further sampling. In an extreme case, the oracle might choose to apply noise to *every* example the learner sees; in this case, it is clearly impossible for the learner to return a hypothesis that will closely approximate the true target, regardless of the amount of sampling the learner performs.

Therefore, our goal will be to show that certain classes are learnable with extremely high (but not arbitrarily high) probability from a membership oracle that exhibits persistent classification noise. Actually, the limitation on the confidence we can achieve leads us to also limit other parameters of the learning algorithm somewhat, including the size of the functions we will consider. For p a fixed polynomial and for all $n \geq 0$, let p-DNF be the class of Boolean functions $f : \{0,1\}^n \to \{-1,+1\}$ expressible as a DNF with at most $p(n)$ terms, and let poly-DNF be the union of the classes p-DNF over all polynomials p. Similarly, let PL_1 be the class of Boolean functions f such that $L_1(\hat{f})$ is bounded by a polynomial in the number of inputs n to f; that is, PL_1 is the polynomial-size version of the class CL_1 defined in Section 5.2. We will show that given modest limitations on the confidence δ and accuracy ϵ input to the learning algorithm, with respect to uniform poly-DNF is weakly learnable and PL_1 is strongly learnable in the persistent classification noise model.

The proof of these facts is based on the following ideas. Since the noise is persistent and the decision about whether or not to apply noise to a given instance is independent of the decisions for all other instances, we can assume that all noise decisions are made before learning commences. That is, we can treat the noisy oracle as if, given f, it chooses a fixed "noisy" function f^η according to the probability distribution D_f^η implied by the noise process described above. Note that, while we refer to f^η as a noisy function, on each execution of the learning algorithm f^η is a Boolean function, and not a random variable as it would be if the noise were not persistent. We will show that, with extremely high probability over the random choice of f^η, *every* Fourier coefficient $\widehat{f^\eta}(A)$ is very near its expected value $(1-2\eta)\hat{f}(A)$, where the expectation is with respect to the choice of f^η according to D_f^η. Note again that for some choices of f^η, some $\widehat{f^\eta}(A)$'s will be very far from $(1-2\eta)\hat{f}(A)$. However, for those (almost all) cases where every $\widehat{f^\eta}(A)$ is near $(1-2\eta)\hat{f}(A)$, running the `WP` algorithm with a threshold of $O(\theta(1-2\eta))$ using the noisy oracle $MEM_{CN}^\eta(f)$ will find essentially the same set of parity functions that would be found running `WP` with threshold θ using the noiseless oracle $MEM(f)$. Because existing algorithms for weakly learning DNF and strongly learning PL_1 with respect to uniform build their hypotheses from the information returned by `WP`, it is then easy to show that these algorithms can be made noise-tolerant.

Thus the key to showing our learnability results is bounding the probability that any $\widehat{f^\eta}(A)$ differs noticeably from $(1-2\eta)\hat{f}(A)$, which we do now. We use $\mathbf{Pr}_\eta[\cdot]$ and $\mathbf{E}_\eta[\cdot]$ to represent probability and expectation, respectively, taken over the choice of f^η randomly according to D_f^η.

Lemma 20

$$\mathbf{Pr}_\eta[\exists A \ s.t. \ |\widehat{f^\eta}(A) - (1-2\eta)\hat{f}(A)| > \lambda] \leq 16e^{n-\lambda^2 2^{n-5}}.$$

Proof: First, consider any fixed A, and let $P_{i,j} = \mathbf{Pr}_x[f(x) = i \ \& \ \chi_A(x) = j]$ for $i, j \in \{-1, +1\}$. Then it follows from its definition that $\hat{f}(A) = \sum_{i,j} ijP_{i,j}$. Similarly, for fixed f^η define $P_{i,j,\ell}^\eta = \mathbf{Pr}_x[f(x) = i \ \& \ \chi_A(x) = j \ \& \ f^\eta(x) = \ell]$, and note that

$$\widehat{f^\eta}(A) = \sum_{i,j,\ell} j\ell P_{i,j,\ell}^\eta.$$

Let $E_{i,j,\ell}$ represent $\mathbf{E}_\eta[P_{i,j,\ell}^\eta]$. Taking expectations over random choice of f^η on both sides of the above equality, it follows that if for each i, j, and ℓ, $|P_{i,j,\ell}^\eta - E_{i,j,\ell}| \leq \lambda/8$, then $|\widehat{f^\eta}(A) - (1-2\eta)\hat{f}(A)| \leq \lambda$.

There is a simple relationship between each $E_{i,j,\ell}$ and one of the $P_{i,j}$'s; for example, $E_{1,1,1} = (1-\eta)P_{1,1}$. We will use this relationship to bound the probability that $P_{i,j,\ell}^\eta$ differs from its expected value by more than $\lambda/8$. To illustrate the process we consider a specific example. Let $S_{1,1} = |\{x \mid f(x) = 1 \ \& \ \chi_A(x) = 1\}|$ and $S_{1,1,1}^\eta = |\{x \mid f(x) = 1 \ \& \ \chi_A(x) = 1 \ \& \ f^\eta(x) = 1\}|$. That is, $P_{1,1} = S_{1,1}/2^n$ and $P_{1,1,1}^\eta = S_{1,1,1}^\eta/2^n$. Then

$$\mathbf{Pr}_\eta[|P_{i,j,\ell}^\eta - E_{i,j,\ell}| > \lambda/8] = \mathbf{Pr}_\eta\left[\left|\frac{S_{1,1,1}^\eta}{S_{1,1}} - (1-\eta)\right| > \frac{\lambda}{8P_{1,1}}\right].$$

We can bound the latter probability by Hoeffding. That is, we can conceptually associate a random variable with each of the $S_{1,1}$ x's such that $f(x) = \chi_A(x) = 1$. Each random variable takes on value 1 with probability $1-\eta$ and value 0 otherwise. $S_{1,1,1}^\eta$ then represents the sum of these random variables, which are independent and have the same mean. Therefore,

$$\mathbf{Pr}_\eta[|P^\eta_{i,j,\ell} - E_{i,j,\ell}| > \lambda/8] \le 2e^{-\lambda^2 S_{1,1}/32P^2_{1,1}} \le 2e^{-\lambda^2 2^{2n-5}/S_{1,1}} \le 2e^{-\lambda^2 2^{n-5}}.$$

The lemma follows by the fact that the probability of the union of events is bounded above by the sum of the probabilities of the events. □

Corollary 21 *For every polynomial p there is an $n_0 > 0$ such that for all $n \ge n_0$, all δ such that $\log(\delta^{-1}) \le p(n)$, and all η such that $1/(1-2\eta) \le p(n)$, poly-DNF is weakly learnable from a persistent classification-noise membership oracle $MEM^\eta_{CN}(f)$ with respect to the uniform distribution. The probability of success of the learning algorithm is taken over the randomness of $MEM^\eta_{CN}(f)$ as well as over the internal randomization of the learning algorithm.*

Proof: Based on the earlier discussion, if f^η is such that for all A, $|\widehat{f^\eta}(A) - (1-2\eta)\hat{f}(A)| \le 2^{-n/4}$, and if δ and η meet the restrictions of the theorem, then a guess-and-double version of `WP` will in time polynomial in n and with probability at least $1 - \delta/2$ return a set of weak approximators to f, any one of which can be used as the final hypothesis of the learning algorithm. Furthermore, by the previous lemma, f^η will fail to meet this condition with probability at most $16e^{n-2^{n/2-5}}$. Also, for any fixed polynomial p and all sufficiently large n, this quantity will be smaller than $2^{-p(n)-1}$ and therefore smaller than $\delta/2$. Therefore, with probability at least $1-\delta$ the algorithm succeeds at finding a weak approximator to f in time polynomial in n. □

Corollary 22 *For every polynomial p there is an $n_0 > 0$ such that for all $n \ge n_0$, all ϵ such that $\epsilon^{-1} \le p(n)$, all δ such that $\log(\delta^{-1}) \le p(n)$, and all η such that $1/(1-2\eta) \le p(n)$, PL_1 is learnable from a persistent classification-noise membership oracle $MEM^\eta_{CN}(f)$ with respect to the uniform distribution. The probability of success of the learning algorithm is taken over the randomness of $MEM^\eta_{CN}(f)$ as well as over the internal randomization of the learning algorithm. We assume that the algorithm is told $L_1(\hat{f})$ and the bound $p(n)$ on $1/(1-2\eta)$.*

Proof: Kushilevitz and Mansour [37] showed that given any Boolean function f, invoking the `WP` algorithm with inputs a membership oracle $MEM(f)$, a threshold $\theta = \epsilon/L_1(\hat{f})$, and a confidence $\delta/2$ produces, with probability $1-\delta/2$, a set S such that $\mathrm{sign}(\sum_{A\in S}\hat{f}(A)\chi_A)$ is an ϵ-approximator to f. Recall that the algorithm runs in time polynomial in n, θ, and $\log\delta^{-1}$, and thus—given the conditions of the corollary—in time polynomial in n for the above value of θ. Furthermore, calling `WP` with a smaller threshold θ only improves the accuracy of the resulting hypothesis, although the running time of `WP` increases as well.

As in the preceding proof, we know that for all sufficiently large n and appropriately bounded δ, the noisy membership oracle $MEM^\eta(f)$ will with probability at least $1 - \delta/2$ choose a function f^η such that for all A, $|\widehat{f^\eta}(A) - (1-2\eta)\hat{f}(A)| \le 2^{-n/4}$. Furthermore, for any such f^η, if we invoke `WP` on inputs $MEM(f^\eta)$, $\theta' = \epsilon(1-2\eta)/L_1(\hat{f}) - 2^{-n/4}$, and $\delta/2$, then in time polynomial in n we will get back a set S' essentially the same as the set S we obtained above from the noiseless oracle for f and using the previous θ. Finally, if we use $\theta'' = \epsilon/(L_1(\hat{f})p(n)) - 2^{-n/4}$ then `WP` will still run in polynomial time and returns with probability $1-\delta/2$ a set S'' which produces a hypothesis at least as accurate as that produced using S. □

9 Further Work

While we have shown that DNF is efficiently learnable in a particular model of learning, the question of whether or not DNF is learnable in more general models—particularly the standard PAC model—remains open. A number of interesting intermediate questions also remain unanswered. For example, is it possible to learn DNF (even monotone DNF, in which no variables are negated) with respect to uniform *without* using membership queries?

In Section 4 we briefly compared `HS` with earlier Fourier-based learning algorithms. There we pointed out that `HS` has the *potential* to learn function classes that these previous algorithms might not be able to learn efficiently. Do such classes exist—and in particular, is DNF such a class? Or is it possible that `HS` is no more powerful than earlier algorithms, that is, that the results of this paper are the consequence of an improved analysis rather than an improved algorithm?

A *perceptron* is a weighted threshold over a set of input variables, or more generally over functions of small numbers of input variables. We consider the problem of learning perceptrons with integer weights and define the *size* of such a perceptron to be the sum of the magnitudes of the weights on the inputs to the threshold. A boosting-based algorithm for PAC-learning (without membership queries and distribution-independently) the subclass of perceptrons over constant fanin functions on the domain $\{0,1\}^n$ follows as a corollary of our TOP results. How does this algorithm compare, in theory and in practice, with existing algorithms for PAC-learning this class, such as Winnow [41]? We already know one potential advantage of the new algorithm: if the size of the target perceptron is s then the output hypothesis of the new algorithm will have only $\tilde{O}(s^2)$ relevant variables. Obviously we can have $s^2 \ll n$ for perceptrons with few relevant variables and small weights. In contrast, Winnow generally puts nonzero weight on all n input variables regardless of the number of relevant variables. Some encouraging preliminary results using this boosting-based perceptron learning algorithm are reported in [33].

10 Acknowledgments

Yishay Mansour and Dick Lipton suggested considering the extensions to TOP and UBOX, respectively, and Rob Schapire suggested that parity-DNF should be learnable by `HS`. An insight of Steven Rudich's inspired the initial proof of Fact 8. Yoav Freund graciously provided unpublished details about his boosting algorithm. Avrim Blum and Merrick Furst provided many useful comments on an early draft of this paper. Avrim also simplified the first portion of the proof of Fact 14, and Merrick suggested the algorithm's name. Doug Tygar helped with the proof of Lemma 23. Thanks also to Nader Bshouty, Tino Tamon, and Andrew Tomkins for helpful discussions and comments. Finally, I thank the anonymous referees for their careful reading and many useful suggestions.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] H. Aizenstein, L. Hellerstein, and L. Pitt, *Read-thrice DNF is hard to learn with membership and equivalence queries*, in Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, 1992, pp. 523–532.

[3] H. Aizenstein and L. Pitt, *Exact learning of read-twice DNF formulas*, in Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, 1991, pp. 170–179.

[4] ——, *Exact learning of read-k disjoint DNF and not-so-disjoint DNF*, in Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 71–76.

[5] D. Angluin, *Queries and concept learning*, Machine Learning, 2 (1988), pp. 319–342.

[6] ——, *Negative results for equivalence queries*, Machine Learning, 5 (1990), pp. 121–150.

[7] D. Angluin, M. Frazier, and L. Pitt, *Learning conjunctions of Horn clauses*, in Proceedings of the 30th Annual Symposium on Foundations of Computer Science, 1990, pp. 186–192.

[8] D. Angluin and M. Kharitonov, *When won't membership queries help?*, in Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing, 1991, pp. 444–454.

[9] M. Bellare, *The spectral norm of finite functions*, Tech. Rep. TR-495, MIT-LCS, Feb. 1991.

[10] 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 the 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 253–262.

[11] A. Blum and S. Rudich, *Fast learning of k-term DNF formulas with queries*, in Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992, pp. 382–389.

[12] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, *Occam's razor*, Information Processing Letters, 24 (1987), pp. 377–380.

[13] ——, *Learnability and the Vapnik-Chervonenkis dimension*, J. Assoc. Comput. Mach., 36 (1989), pp. 929–965.

[14] D. Boneh and R. J. Lipton, *Amplification of weak learning under the uniform distribution*, in Proceedings of the Sixth Annual Workshop on Computational Learning Theory, 1993, pp. 347–351.

[15] J. Bruck, *Harmonic analysis of polynomial threshold functions*, SIAM Journal of Discrete Mathematics, 3 (1990), pp. 168–177.

[16] J. Bruck and R. Smolensky, *Polynomial threshold functions, AC^0 functions and spectral norms*, in Proceedings of the 31st Annual Symposium on Foundations of Computer Science, 1990, pp. 632–641.

[17] N. Bshouty. Personal communication.

[18] N. H. Bshouty, *Exact learning via the monotone theory*, in Proceedings of the 34th Annual Symposium on Foundations of Computer Science, 1993, pp. 302–311.

[19] N. H. Bshouty, Z. Chen, and S. Homer, *On learning discretized geometric concepts*, in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 54–63.

[20] N. H. Bshouty, P. W. Goldberg, S. A. Goldman, and H. D. Mathias, *Exact learning of discretized geometric concepts*, Tech. Rep. WUCS-94-19, Washington University, July 1994.

[21] Y. Freund, *Boosting a weak learning algorithm by majority*, in Proceedings of the Third Annual Workshop on Computational Learning Theory, 1990, pp. 202–216.

[22] ——, *An improved boosting algorithm and its implications on learning complexity*, in Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 391–398.

[23] ——, *Data Filtering and Distribution Modeling Algorithms for Machine Learning*, PhD thesis, University of California at Santa Cruz, Sept. 1993. Available as technical report UCSC-CRL-93-37.

[24] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, in Proceedings of the Second Annual European Conference on Computational Learning Theory, 1995.

[25] M. L. Furst, J. C. Jackson, and S. W. Smith, *Improved learning of AC^0 functions*, in Fourth Annual Workshop on Computational Learning Theory, 1991, pp. 317–325.

[26] P. W. Goldberg, S. A. Goldman, and H. D. Mathias, *Learning unions of boxes with membership and equivalence queries*, in Proceedings of the Seventh Annual Workshop on Computational Learning Theory, 1994, pp. 198–207.

[27] S. A. Goldman, M. J. Kearns, and R. E. Schapire, *Exact identification of circuits using fixed points of amplification functions*, SIAM Journal on Computing, 22 (1993), pp. 705–726. Preliminary version appeared in *Proceedings of the 31st Symposium on Foundations of Computer Science*, pages 193–202, 1990.

[28] M. Goldmann, J. Hastad, and A. Razborov, *Majority gates vs. general weighted threshold gates*, in Proceedings of the Seventh IEEE Conference on Structure in Complexity Theory, 1992, pp. 2–13.

[29] O. Goldreich and L. A. Levin, *A hard-core predicate for all one-way functions*, in Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing, 1989, pp. 25–32.

[30] T. R. Hancock, *Learning 2μDNF formulas and kμ decision trees*, in Proceedings of the Fourth Annual Workshop on Computational Learning Theory, 1991, pp. 199–209.

[31] J. Jackson, *An efficient membership-query algorithm for learning DNF with respect to the uniform distribution*, in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 42–53.

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

[33] J. C. Jackson and M. W. Craven, *Learning sparse perceptrons*, in Advances in Neural Information Processing Systems, D. Touretzky, M. Mozer, and M. Hasselmo, eds., vol. 8, MIT Press, Cambridge, MA, 1996, pp. 654–660.

[34] M. Kearns, M. Li, L. Pitt, and L. Valiant, *On the learnability of Boolean formulae*, in Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, 1987, pp. 285–295.

[35] M. Kharitonov, *Cryptographic hardness of distribution-specific learning*, in Proceedings of the 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 372–381.

[36] M. Krause and P. Pudlák, *On the computational power of depth 2 circuits with threshold and modulo gates*, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 48–57.

[37] E. Kushilevitz and Y. Mansour, *Learning decision trees using the Fourier spectrum*, SIAM Journal on Computing, 22 (1993), pp. 1331–1348. Earlier version appeared in *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 455–464, 1991.

[38] E. Kushilevitz and D. Roth, *On learning visual concepts and DNF formulae*, in Proceedings of the Sixth Annual Workshop on Computational Learning Theory, 1993, pp. 317–326.

[39] L. A. Levin, *Randomness and non-determinism*, Journal of Symbolic Logic, 58 (1993), pp. 1102–1103.

[40] N. Linial, Y. Mansour, and N. Nisan, *Constant depth circuits, Fourier transform, and learnability*, J. Assoc. Comput. Mach., 40 (1993), pp. 607–620. Earlier version appeared in *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 574–579, 1989.

[41] N. Littlestone, *Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm*, Machine Learning, 2 (1988), pp. 285–318.

[42] P. M. Long and M. K. Warmuth, *Composite geometric concepts and polynomial predictability*, in Proceedings of the Third Annual Workshop on Computational Learning Theory, 1990, pp. 273–287.

[43] Y. Mansour, *An $O(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution*, in Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 53–61.

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

[45] E. Shamir. Personal communication.

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

Appendix

Here we briefly sketch the proof of a lemma needed for Fact 14.

Lemma 23 *For every* 1*-rectangle* $R : [b] \to \{0,1\}$*,* $L_1(\widehat{R}) = O(\log b)$*.*

Proof Sketch: Let d represent the width of R, that is, $d = |\{x \mid R(x) = 1\}|$. By the definition of the Fourier transform, for all $a \in [b]$, $\widehat{R}(a) = \mathbf{E}_{x\in[b]}[R(x)\cdot\chi_A(x)] = \sum_{x\in[b]} R(x)\cdot\omega_b^{ax}/b$. Clearly $|\widehat{R}(0)| \leq 1$ regardless of the value of d. Furthermore, we may view each root of unity ω_b^c as a 2-dimensional unit vector in the complex plane with tail at the origin and head at the point ω_b^c. Given this view, each of the other Fourier coefficients $\widehat{R}(a)$, $a \in \{1,\ldots,b-1\}$, is a vector sum of a sequence of d 2-dimensional vectors of length $1/b$. Now consider the geometric figure that results if we draw the first vector in the sum defining $\widehat{R}(a)$, then draw the second vector with its tail at the head of the first, the third with its tail at the head of the second, and so on, *i.e.*, graphically represent the vector sum defining $\widehat{R}(a)$. Note that, by the definition of the roots of unity, there is a fixed angle $2\pi a/b$ between each consecutive pair of vectors in the sequence defining $\widehat{R}(a)$. Now a simple geometric argument gives that a circle of diameter diameter $1/b\sin(\pi a/b)$ circumscribes this figure (the circle center is the intersection point of the perpendicular bisectors of the vectors). Because the norm of $\widehat{R}(a)$ is just the length of this vector sum, we have that

$$|\widehat{R}(a)| \leq \frac{1}{b\sin\frac{\pi a}{b}}$$

for $a \in \{1,\ldots,b-1\}$, regardless of the choice of d.

Thus we have that

$$L_1(\widehat{R}) \leq 1 + \frac{1}{b}\sum_{a=1}^{b-1}\frac{1}{\sin\frac{\pi a}{b}}.$$

We can bound the sum on the right-hand side by treating a as a continuous rather than discrete variable and integrating. Specifically, it can be verified that

$$\sum_{a=1}^{b-1}\frac{1}{\sin\frac{\pi a}{b}} \leq 2\left[\frac{1}{\sin\frac{\pi}{b}} + \int_1^{b/2}\csc\frac{\pi a}{b}da\right].$$

Furthermore,

$$\int_1^{b/2}\csc\frac{\pi a}{b}da = \frac{b}{\pi}\left[\ln\tan\frac{\pi}{4} - \ln\tan\frac{\pi}{2b}\right] = \frac{b}{\pi}\ln\cot\frac{\pi}{2b}.$$

Now in the range $0 \leq \theta \leq \pi/2$, $\sin\theta \geq 2\theta/\pi$. Noting that $b \geq 2$, we can apply this fact to the above expressions and simplify to get

$$L_1(\widehat{R}) \leq 2 + \frac{2}{\pi}\ln(b).$$

$\square$