

MQ ALGORITHM FOR LEARNING DNF WITH RESPECT TO THE UNIFORM

Guy Grebla (grebla@cs.technion.ac.il)

Ido Feldman (idofeld@cs.technion.ac.il)

1. INTRODUCTION

We describe an algorithm for learning DNF with respect to the uniform distribution. The algorithm presented here is a boosting algorithm which receives a weak learning algorithm for DNF as an input, runs it on different distributions, and combines the results into a unified hypothesis. Similar to previous algorithm, we will use majority vote to combine the weak hypothesis into a strong one. The algorithm presented is close to optimal in terms of number of examples and time complexity. This summary is heavily based on [7, 8].

2. BACKGROUND

2.1. Previous Boosting Algorithms. 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.

2.2. Freund's Boosting Algorithm. The DNF learning algorithm presented here is based on one of Freund's boosting algorithms. 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 polynomialtime 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.

We now discuss the F1 boosting algorithm in detail. As input, F1 is given positive ϵ, δ, 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.

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

where $B(j; n; p)$ 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

$$(2.1) \quad D_i(x) = \frac{D(x)\alpha_{r_i(x)}^i}{\sum_y D(y)\alpha_{r_i(y)}^i}.$$

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 $\frac{1}{2}\gamma^{-2} \ln(\varepsilon^{-1})$ 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 (2.1) is the probability that F1 accepts an example while simulating the oracle $EX(f, D_i)$. Thus if at some stage i this denominator, $\sum_y D(y)\alpha_{r_i(y)}^i$, 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 $\sum_y D(y)\alpha_{r_i(y)}^i$ 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 2.1 The following lemma concerning the functionality and running time of F1 is due to Freund [4]; the previously unpublished proof of one part of Freund’s argument is presented in [8].

Lemma 2.1. *For all iterations $0 \leq i \leq k - 2$ of F1,*

$$\alpha_{max}^i \leq \sqrt{8/3\pi(k - i - 1)}e^{1/12}$$

Proof. We can rewrite the definition of α_r^i as follows (ignoring the choices of r that give $\alpha_r^i = 0$ for the purpose of the upper bound):

$$\alpha_r^i = \binom{x}{(1/2 - \mu)x} \left(\frac{1}{2} - \frac{\gamma}{2}\right)^{x(1/2 - \mu)} \left(\frac{1}{2} + \frac{\gamma}{2}\right)^{x(1/2 + \mu)},$$

where $x = k - i - 1$ and $\mu = 1/2 - ([k/2] - r)/(k - i - 1)$.

It is well known that for any real numbers $x \geq 1$ and $0 \leq \mu < 1/2$,

$$e^{-\frac{1}{3(1-4\mu^2)x}} < \frac{\binom{x}{(1/2-\mu)x}}{\sqrt{2/\pi x} e^{xH(1/2-\mu)}} < \frac{e^{1/12x}}{\sqrt{1-4\mu^2}},$$

Where $H(y) = -y \ln y - (1-y) \ln(1-y)$ is the entropy function, and the extension of the binomial function to the reals is based on the extension of the factorial to the gamma function $x! = \Gamma(x+1)$.

Using this upper bound, we bound the last expression for any value of μ

$$\begin{aligned} & \binom{x}{(1/2-\mu)x} \left(\frac{1}{2} - \frac{\gamma}{2}\right)^{x(1/2-\mu)} \left(\frac{1}{2} + \frac{\gamma}{2}\right)^{x(1/2+\mu)} < \\ & \sqrt{\frac{2}{\pi x}} \frac{e^{1/12x}}{\sqrt{1-\gamma^2}} e^{xH(\frac{1}{2}-\mu) + x(\frac{1}{2}-\mu) \ln(\frac{1}{2}-\frac{\gamma}{2}) + x(\frac{1}{2}+\mu) \ln(\frac{1}{2}+\frac{\gamma}{2})} \end{aligned}$$

But a basic inequality is that for any $-1/2 \leq \mu \leq 1/2$,

$$H\left(\frac{1}{2} - \mu\right) \leq -\left(\frac{1}{2} - \mu\right) \ln\left(\frac{1}{2} - \frac{\gamma}{2}\right) - \left(\frac{1}{2} + \mu\right) \ln\left(\frac{1}{2} + \frac{\gamma}{2}\right)$$

This inequality is strict unless $\mu = \gamma/2$. From this we get that

$$\binom{x}{(1/2-\mu)x} \left(\frac{1}{2} - \frac{\gamma}{2}\right)^{x(1/2-\mu)} \left(\frac{1}{2} + \frac{\gamma}{2}\right)^{x(1/2+\mu)} < \sqrt{\frac{2}{\pi x}} \frac{e^{1/12x}}{\sqrt{1-\gamma^2}}$$

As $\gamma \leq 1/2$ and $x \geq 1$, we obtain the statement of the lemma. □

Lemma 2.2. *If at any stage i of F1*

$$\sum_y D(y) \alpha_{r_i(y)}^i \leq \frac{\epsilon^3}{57}$$

then the hypothesis h formed by taking a majority vote over the weak hypotheses w_0, w_1, \dots, w_{i-1} is an ϵ -approximator for f with respect to D .

Before proving this lemma, we note that the 0.39 bound on $\bar{\gamma}$ in F1 was chosen to keep the statement of the above stopping criterion relatively simple. As will become apparent in the proof, other choices of the bound can be used to derive alternative stopping criteria for F1 having exponents of ϵ arbitrarily close to 2 and smaller constants.

Proof. For $0 \leq i < k$ and $0 \leq r < i$, define

$$D(r) = \sum_{x: r_i(x)=r} D(x)$$

(recall that $r_i(x)$ counts the number of weak hypotheses found before stage i that are “right” on x). Our goal will be to show that at every stage i of F1,

$$(2.2) \quad \sum_y D(y) \alpha_{r_i(y)}^i \leq \frac{\epsilon^3}{57} \Rightarrow \sum_{r \leq i/2} D(r) \leq \epsilon$$

Given this, a majority vote over the weak hypotheses found prior to stage i is an ϵ -approximator to the target function f .

The argument for (2.2) consists of two parts. First, we show that it is always the case that at any stage i ,

$$(2.3) \quad \sum_{r=0}^{\lfloor i/2 - (k-i)\bar{\gamma} \rfloor} D(r) \leq \epsilon/2$$

Then we will show that given the antecedent of (2.2) it is also true that

$$(2.4) \quad \sum_{r=\lfloor i/2 - (k-i)\bar{\gamma} \rfloor}^{\lfloor i/2 \rfloor} D(r) \leq \epsilon/2$$

To show (2.3) we need a fundamental result from Freund [4]. Define for each stage i of the F1 boosting process the function β_r^i , $0 \leq r \leq i$, as

$$\beta_r^i = \begin{cases} 0 & \text{if } r > \frac{k}{2} \\ \sum_{j=0}^{\lfloor \frac{k}{2} \rfloor - r} B(j; k-i, \frac{1}{2} + \bar{\gamma}) & \text{if } i - \lfloor \frac{k}{2} \rfloor \leq r \leq \frac{k}{2} \\ 1 & \text{otherwise} \end{cases}$$

(recall that $B(\cdot; \cdot, \cdot)$ represents the binomial formula). Then Freund [4, 5] shows that if F1 is run for $k \geq \frac{1}{2}\delta^{-2} \ln(4/\epsilon)$ stages on a $(\frac{1}{2} - \gamma)$ -approximate weak learner, at each stage i we will have

$$\sum_{r=0}^i \beta_r^i D_i(r) \leq \frac{\epsilon}{4}.$$

Note also that $\bar{\gamma} \leq \gamma$ and we take $k = \frac{1}{2}\bar{\gamma}^{-2} \ln(4/\epsilon)$ then $k \geq \frac{1}{2}\gamma^{-2} \ln(4/\epsilon)$. Now the median of a binomial distribution $B(n, p)$ is either $\lfloor np \rfloor$ or $\lceil np \rceil$. Because β_r^i represents a cumulative binomial distribution, where the underlying binomial distribution has mean $\mu = (k-i)(\frac{1}{2} + \bar{\gamma})$, it follows that for r such that $\lfloor k/2 \rfloor - r \geq \mu$, $\beta_r^i \geq \frac{1}{2}$. In particular, for $r \leq \lfloor i/2 - (k-i)\bar{\gamma} \rfloor$, β_r^i is thus lower bounded, which gives us

$$\frac{\epsilon}{4} \geq \sum_{r=0}^{\lfloor i/2 - (k-i)\bar{\gamma} \rfloor} \beta_r^i D_i(r) \geq \frac{1}{2} \sum_{r=0}^{\lfloor i/2 - (k-i)\bar{\gamma} \rfloor} D_i(r)$$

The argument for (2.4) is somewhat more involved. First, since the binomial distribution $B(n, p)$ reaches a maximum at $\lfloor p(n+1) \rfloor$, we have that α_r^i is maximized for some $r \leq \lfloor i/2 - (k-i)\bar{\gamma} \rfloor + 1$. For larger values of r , α_r^i is monotonically decreasing. Therefore, in the region $\lfloor i/2 - (k-i)\bar{\gamma} \rfloor < r < i/2$, α_r^i reaches a minimum α_{min} at $r = \lfloor i/2 \rfloor$. We will show that $\alpha_{min} \geq \epsilon^2/28.5$, and therefore that

$$\begin{aligned} & \sum_{r=\lfloor i/2 - (k-i)\bar{\gamma} \rfloor + 1}^{\lfloor i/2 \rfloor} D(r) \leq \\ & \sum_{y: i/2 - (k-i)\bar{\gamma} < r(y) \leq i/2} D(y) \frac{\alpha_{r_i(y)}^i}{\alpha_{min}} \leq \frac{\epsilon^3}{57\alpha_{min}} \leq \frac{\epsilon}{2} \end{aligned}$$

as desired.

The lower bound argument for $\alpha_{min} = \alpha_{\lfloor i/2 \rfloor}^i$ also consists of two parts. In particular, recall that $\alpha_r^i \equiv \tilde{\alpha}_r^i / \max\{\tilde{\alpha}_r^i\}$. The denominator of this expression is upper bounded by $\sqrt{8/(3\pi(k-i-1))}e^{1/12}$. Thus we only need an appropriate lower bound on $\tilde{\alpha}_{min} = \tilde{\alpha}_{\lfloor i/2 \rfloor}^i$ to complete the proof.

To show the required bound on $\tilde{\alpha}_{min}$ we first consider several cases. If k is odd and i is even then

$$(2.5) \quad \tilde{\alpha}_{min} = \binom{k-i-1}{\frac{k-i-1}{2}} \left(\frac{1}{2} + \bar{\gamma}\right)^{\frac{k-i-1}{2}} \left(\frac{1}{2} - \bar{\gamma}\right)^{\frac{k-i-1}{2}}$$

We use $\tilde{\alpha}_{1/2}$ to denote the righthand side of (2.5). If k is even and i is odd we have

$$\tilde{\alpha}_{min} = \tilde{\alpha}_{\lfloor i/2 \rfloor}^i \geq \tilde{\alpha}_{\lfloor i/2 \rfloor}^i = \tilde{\alpha}_{1/2}$$

For the other two cases (k and i both even or both odd), we have

$$\tilde{\alpha}_{min} = \tilde{\alpha}_{\lfloor i/2 \rfloor}^i \geq \tilde{\alpha}_{\lfloor i/2 \rfloor + 1/2}^i = \tilde{\alpha}_{1/2}$$

where we have generalized the definition of the binomial coefficient to the reals using the $\Gamma(\cdot)$ function.

To lower bound $\tilde{\alpha}_{1/2}$, we first note that using Stirling's approximation it can be shown that for any real $x \geq 1$,

$$\binom{x}{\frac{x}{2}} > \sqrt{\frac{2}{\pi x}} 2^x e^{1/3}$$

Also,

$$\left(\frac{1}{2} + \bar{\gamma}\right)^{x/2} \left(\frac{1}{2} - \bar{\gamma}\right)^{x/2} = 2^{-x} (1 - 4\bar{\gamma}^2)^{x/2}.$$

It can also be shown that for $0 < y < 1$, $(1-y)^{\frac{1}{y}-1} \geq e^{-1}$. Furthermore, for $\bar{\gamma} \leq 0.39$, $1 - 4\bar{\gamma}^2 \geq e^{-1}$. Therefore, for $x = k - i - 1 \leq k = \frac{1}{2}\bar{\gamma}^{-2} \ln(4/\epsilon)$, we have

$$\left(\frac{1}{2} + \bar{\gamma}\right)^{\frac{k-i-1}{2}} \left(\frac{1}{2} - \bar{\gamma}\right)^{\frac{k-i-1}{2}} \geq 2^{-(k-i+3)} \epsilon^2$$

which means that

$$\tilde{\alpha}_{min} \geq \sqrt{\frac{2}{\pi(k-i-1)}} 2^{k-i-1} e^{-1/3} 2^{-(k-i+3)} \epsilon^2,$$

and combining this with the earlier upper bound on $\max\{\tilde{\alpha}_r^i\}$ gives

$$\alpha_{min} \geq \frac{\sqrt{3}}{32} e^{-5/12} \epsilon^2 \geq \frac{\epsilon^2}{28.5}.$$

□

Lemma 2.3. *Algorithm F1, given positive ϵ, δ, γ , a $(\frac{1}{2} - \gamma)$ -approximate PAC learner for representational class \mathcal{F} , and example oracle $EX(f, D)$ for some $f \in \mathcal{F}$ and any distribution D , runs in time polynomial in $n, s, \gamma^{-1}, \epsilon^{-1}$, and $\log(\delta^{-1})$ and produces, with probability at least $1 - \delta$ and ϵ -approximation for f with respect to D .*

Proof. If the algorithm goes through all k stages, then by the result of the majority vote game [4] the lemma holds. Otherwise, by using union bound and the definition of *AMEAN* it can be seen that the condition of lemma 2.2 are satisfied with probability of at least $1 - \delta$, and therefore the lemma holds.

□

FIGURE 2.1. The F1 hypothesis boosting algorithm.

Invocation: $h \leftarrow F1(EX(f, D), \gamma, WL, \varepsilon, \delta)$

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 $WL(EX, \delta)$ which succeeds with probability at least $1 - \delta$; $\varepsilon > 0$; $\delta > 0$

Output: h such that, with probability at least $1 - \delta$, $\Pr_D[f = h] \geq 1 - \varepsilon$

- (1) $\bar{\gamma} \equiv \min(\gamma, 0.39)$
- (2) $k \leftarrow \frac{1}{2}\bar{\gamma}^{-2} \ln(4/\varepsilon)$
- (3) $w_0 \leftarrow WL(EX(f, D), \delta/2k)$
- (4) **for** $i \leftarrow 1, \dots, k-1$ **do**
 - (a) $r_i(x) \equiv |\{0 \leq j < i \mid w_j(x) = f(x)\}|$
 - (b) $B(j; n, p) \equiv \binom{n}{j} p^j (1-p)^{n-j}$
 - (c) $\tilde{\alpha}_r^i \equiv B(\lfloor k/2 \rfloor - r; k-i-1; 1/2 + \bar{\gamma})$ if $i - k/2 < r \leq k/2$, $\tilde{\alpha}_r^i \equiv 0$ otherwise
 - (d) $\alpha_r^i \equiv \tilde{\alpha}_r^i / \max_{r=0, \dots, i-1} \{\tilde{\alpha}_r^i\}$.
 - (e) $\Theta \equiv \varepsilon^3/57$
 - (f) $X \equiv$ draw examples $\langle x, f(x) \rangle$ from $EX(f, D)$ and compute $\alpha_{r_i}^i(x)$
 - (g) $E_\alpha \leftarrow AMEAN(X, b-a=1, \frac{1}{3}\Theta, \delta/2k)$
 - (h) **if** $E_\alpha \leq \frac{2}{3}\Theta$ **then**
 - (i) $k \leftarrow i$
 - (ii) **break do**
 - (i) $D_i(x) \equiv \frac{D(x)\alpha_{r_i}^i(x)}{\sum_y D(y)\alpha_{r_i}^i(y)}$
 - (j) $w_i \leftarrow WL(EX(f, D_i), \delta/2k)$
- (5) **end do**
- (6) $h(x) \equiv MAJ(w_0(x), w_1(x), \dots, w_{k-1}(x))$
- (7) **return** h

The specific implementation of $F1$ given in Figure 2.1 runs in time $\tilde{O}(\gamma^{-2}(\gamma^{-2} + \varepsilon^{-6} + T(WL)))$, where the softO notation \tilde{O} is the same as the big-O notation but suppresses logarithmic factors, and $T(WL)$ 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} [8], 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 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 $F1$.

2.3. 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 [6]. 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 [2]). Kushilevitz and Mansour showed that the WP algorithm could be used to learn parity decision trees with respect to the uniform distribution [9], and it has since been the basis of other learning algorithms as well [2, 10].

Actually, the WP algorithm is somewhat more general than we have described thus far. The basic WP algorithm finds, with probability at least $1 - \gamma$, 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 $\text{MEM}(f)$, 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 $A \subseteq \{1, \dots, k\}$; these parameters define the collection $C_{A,k}$ consisting of all the coefficients $\hat{f}(A \cup B)$ such that $B \subseteq \{k+1, \dots, n\}$. Initially, WP is run on the collection of all Fourier coefficients of f , $C_{\phi,0}$.

Each time WP is called, it begins by defining a partition of the input collection $C_{A,k}$ into the two equalized 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 abovethreshold 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 $f : \{0, 1\}^n \rightarrow R$, every $k \in [0, n]$, and every $A \subseteq \{1, \dots, k\}$,

$$(2.6) \quad L_2^2(C_{A,k}) = E_{x,y,z} [f(yx)f(zx)\chi_A(y \oplus z)]$$

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 2.6 intended to give some intuition for this relationship; First, we define

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

where $x \in \{0, 1\}^{n-k}$ and $A \subseteq \{1, \dots, k\}$ as above and $B \subseteq \{k+1, \dots, n\}$. Now by Parseval, $E_x [f_A^2(x)] = \sum_B \hat{f}_A^2(B)$, 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 $A \subseteq \{1, \dots, k\}$:

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

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

$$f_A(x) = E_y [f(yx) \chi_A(y)],$$

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

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

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 $1 - \gamma$ returns a set $S \subseteq 2^{\{1, \dots, n\}}$ such that:

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

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 $O(n \log(n/\delta\theta^2)/\theta^6)$.

FIGURE 2.2. The WP' parity-finding algorithm

Invocation: $s \leftarrow WP'(n, MEM(g), \theta, L_\infty(g), \delta)$

Input: Number n of inputs to function $g : \{0, 1\}^n \rightarrow \mathbf{R}$; membership oracle $MEM(g)$; $0 < \theta \leq 1$; $L_\infty(g) > 0$; $\delta > 0$

Output: Set S hat, with probability at least $1 - \delta$, has the large Fourier coefficient property for g and θ .

- (1) $A \leftarrow \phi$
- (2) $k \leftarrow 0$
- (3) return $WP^{\text{'}}\text{-aux}(A, k, n, MEM(g), \theta, L_\infty(g), \delta)$

Invocation: $S \leftarrow WP' - \text{aux}(k, A, n, MEM(g), \theta, L_\infty(g), \delta)$

Input: Integer $k \in [0, n]$; Set $A \subseteq \{1, \dots, k\}$; number n of inputs to function $g : \{0, 1\}^n \rightarrow \mathbf{R}$; 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 $\{B \subseteq \{1, \dots, n\} \mid B \cap \{1, \dots, k\} = A\}$.

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

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 2.4. *There is an algorithm WP' such that, for any function $g : \{0, 1\}^n \rightarrow \mathbf{R}$, threshold $\theta > 0$, and confidence $\delta > 0$, $WP'(n, MEM(g), \theta, L_\infty(g), \delta)$ returns, with probability at least $1 - \delta$, a set with the large Fourier coefficient property. 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 2.2. For each collection $C_{A,k}$, by Hoeffding's inequality $O(nL_\infty^4(g) \log(nL_\infty^2(g)/\delta\theta^2)/\theta^4)$ samples of the random variable X are sufficient for $AMEAN$ to estimate μ' such that

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

The fact that $\sum_A \hat{g}^2(A) = E[g^2] \leq L_\infty^2(g)$ 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 WP . \square

3. STRONGLY LEARNING DNF

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 [2]. 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 Fourierbased 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 $E_D[f(x)]$ deviates noticeably (i.e., inverse polynomially in the size of f) from 0 then the constant parity χ_\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

$$\begin{aligned} & \Pr_D[x \in Sat] \cdot \Pr_D[T(x) = f(x) | x \in Sat] \\ & + \Pr_D[x \in Unsat] \cdot \Pr_D[T(x) = f(x) | x \in Unsat] \\ & \approx \frac{1}{2} [\Pr_D[T(x) = f(x) | x \in Sat] + \Pr_D[T(x) = f(x) | x \in Unsat]] \end{aligned}$$

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 , $\Pr_D[T(x) = f(x) | x \in Unsat] = 1$. Furthermore, since there are only s terms and T is the best correlated term, $\Pr_D[T(x) = f(x) | x \in Sat] \geq 1/s$. Thus T agrees with f with probability noticeably greater than $\frac{1}{2}$, or equivalently, $E_D[f \cdot T]$ 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

$$|E_{x \in_R D}[f(x) \cdot T(x)]| \leq |E_D[f(x)]| + 2E_{\chi_A \in T}[|E_D[f(x)\chi_A(x)]|],$$

where $E_{\chi_A \in T}[\cdot]$ 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 $E_D[f \cdot T]$ is noticeably large while $E_D[f]$ is very small—the above relation implies that the “average” of the parity functions in T is noticeably wellcorrelated with f . Therefore, at least one of these parity functions must be wellcorrelated with f , which gives us our claim.

Our original proof formalized these ideas. Subsequently, Bshouty [3] refined our approach and developed the proof that we present here, which gives somewhat better constants.

Fact 3.1. *For every DNF expression f with s terms and for every distribution D on the instance space of f there exists a term T in f and a $\chi_A \in T$ such that $|E_D[f\chi_A]| \geq 1/(2s + 1)$.*

Proof. Let $p = \Pr_{x \in_R D}[x \text{ satisfies } T]$. There is at least one term T in f such that $\Pr_D[x \text{ satisfies } T] \geq p/s$. 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 $\{1, \dots, n\}$ 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 $E_{A \subseteq V}[(-1)^{|A|} \chi_A(x)]$, where the expectation is uniform over the subsets $A \subseteq V$.

Now consider the quantity $E_D[f(x) \cdot T'(x)]$. Substituting the expected value above for T' and rearranging gives

$$E_D[fT'] = E_{A \subseteq V}[(-1)^{|A|} E_D[f\chi_A]] \leq E_{A \subseteq V}[|E_D[f\chi_A]|].$$

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,

$$E_D[fT'] = E_D[T'] = \Pr_D[T = 1] \geq p/s$$

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

3.1. Nonuniform Weak DNF Learning. Fact 3.1 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 [1]. 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 [2]. 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 we 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 $E[f \cdot \chi_A]$, 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, $|E_D[f \cdot \chi_A]| \geq \theta/2$. 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

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

Thus if we take $g(x) = 2^n f(x)D(x)$ then we have that for all A , $E[f \cdot \chi_A] = \hat{g}(A)$. 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 wellcorrelated 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 nonBoolean 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 3.2. *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 $f : \{0, 1\}^n \rightarrow \{-1, +1\}$, 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 $E_D[fh] = \Omega(s^{-1})$, where s is the DNFsize 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 3.1 and the above argument there is some χ_A such that $|E_D[f\chi_A]| = |\hat{g}(A)| \geq 1/(2s + 1)$. Thus by Lemma 7, $WP'(n, MEM(g), \theta = 1/(2s + 1), L_\infty(g) = L_\infty(2^n D), \delta)$ will, with probability at least $1 - \delta$ find a χ_A such that $|E_D[f\chi_A]| \geq \Omega(s^{-1})$. Furthermore, the 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 3.2. Specifically, note that if WP' is called using values s' and $L'_\infty(2^n D)$ that are larger than their respective true values, 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 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 WP' has the large Fourier coefficient property (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 $E_D(f\chi_B)$ for each $B \in S$. By Hoeffding (Lemma 2), if we use $\lceil 8 \ln(8L^2/\delta'\theta^2)/\theta^2 \rceil$ examples then, with probability at least $1 - \delta'\theta^2/4L^2$, our estimate μ' is within $\theta/4$ of $E_D[f\chi_B]$. Therefore, if $|\mu'| \geq 3\theta/4$ then $|E_D[f\chi_B]| \geq \theta/2$. 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 $E_D[fh] \geq 1/(4s + 2)$.

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. \square

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 *polynomiallynear 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.

3.2. 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 polynomiallynear 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 polynomiallynear uniform. In fact, this is true of all boostbyfiltering 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 polynomialtime 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 3.3. *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 DNFsize 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 WDF developed in the preceding section. Figure 3.1 describes the algorithm. For simplicity, this version of the algorithm assumes that the DNFsize s of the target f is provided; this assumption is easily removed by modifying the algorithm to use a guessanddouble technique similar to that used in WDF . 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 WDF succeeds at producing a

$(1/2 - 1/(8s + 4))$ approximate weak hypothesis with respect to D_i at each stage i then Lemma 2.3 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 3.2, WDF 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 3.1) requires computing exactly the exponentially large sum $\sum_y D(y) \alpha_{r_i(y)}^i$.

To circumvent this difficulty, HS provides WDF 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 $\sum_y D(y) \alpha_{r_i(y)}^i$. 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 $2\theta/3$, $\frac{2}{3} \sum_y D(y) \alpha_{r_i(y)}^i \leq E_\alpha \leq 2 \sum_y D(y) \alpha_{r_i(y)}^i$. This implies that there is a constant $c \in [\frac{1}{2}, \frac{3}{2}]$ such that for all x , $D'_i(x) = D_i(x)$.

Now consider the functional impact of supplying this approximate oracle rather than the true oracle to WDF (we consider the impact on running time below). WDF 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 $c \cdot \hat{g}(A) = \hat{c \cdot g}(A)$ for all A . In summary, multiplying WDF '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 WDF . 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 polynomialtime 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 WDF 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 WDF .

FIGURE 3.1. The HS algorithm for efficiently learning DNF.

Invocation: $h \leftarrow HS(EX(f, D), MEM(f), D, s, \epsilon, \delta)$

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 $1 - \delta$, $\Pr_D[f = h] \geq 1 - \epsilon$

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

By Lemma 3.2, 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 repeatuntil 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. \square

FIGURE 3.2. 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 .

Invocation: $h \leftarrow WDNF(EX(f, D), MEM(f), D, \delta)$

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 $1 - \delta$, $\Pr_D(f = h) \geq \frac{1}{2} + \frac{1}{8s+4}$, where s is the DNF-size of f .

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

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} , giving a bound of $\tilde{O}(ns^8/\epsilon^{12+c})$ for arbitrarily small constant c .

REFERENCES

- [1] Mihir Bellare. The spectral norm of finite functions. Technical Report TR-495, MIT-LCS, February 1991.
- [2] 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 ACM Symposium on the Theory of Computing. ACM Press, New York, NY, 1994.
- [3] N. Bshouty. Personal communication, November 1994.
- [4] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.
- [5] Yoav Freund. Data Filtering and Distribution Modeling Algorithms for Machine Learning, PhD thesis, University of California at Santa Cruz, Sept. 1993.
- [6] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In Proceedings of the 21st Annual ACM Symposium on Theory of Computing, pages 25–32, 1989.

- [7] Jackson, J., An Efficient Membership-Query Algorithm for Learning DNF with Respect to the Uniform Distribution, *Journal of Computer and System Sciences* 55(3), 1997.
- [8] J. Jackson. The Harmonic Sieve: A Novel Application of Fourier Analysis to Machine Learning Theory and Practice. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1995.
- [9] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. In *Proceedings of STOC '91*, pages 455–464, may 1991.
- [10] Yishay Mansour. An $O(n^{\log\log(n)})$ learning algorithm for DNF under the uniform distribution. In *Fifth Annual Workshop on Computational Learning Theory*, pages 53–61, 1992.