Electronic Colloquium on Computational Complexity, Report No. 76 (1998)

# Attribute Efficient PAC-learning of DNF with Membership Queries Under the Uniform Distribution

Nader H. Bshouty
Technion

Jeffrey Jackson
Duquesne University

Christino Tamon
Clarkson University

November 13, 1998

**Abstract**

We study attribute efficient learning in the PAC learning model with membership queries. First, we give an *attribute efficient* PAC-learning algorithm for DNF with membership queries under the uniform distribution. Previous algorithms for DNF have sample size polynomial in the number of attributes $n$. Our algorithm is the first attribute efficient learning for DNF, i.e., that runs in polynomial time and uses sample of size polynomial in $\log n$. We also develop lower bound techniques for PAC learning with membership queries under a fixed distribution and show that the sample size of our algorithm for learning DNF under the uniform distribution is almost optimal in terms of $n$. Finally, we present a learning algorithm for DNF that is attribute efficient in its use of random bits.

## 1 Introduction

In the standard PAC learning model [V84], the time and sample complexity of a learning algorithm are allowed to be polynomial in $n$, the number of attributes (variables), as well as in various other parameters including $s$, the size of the target function $f$, and $\epsilon$, the accuracy required of the approximating function $h$ output by the learner. Recently, *attribute efficient* learning algorithms have attracted significant attention [L88, B92, BH96, BHL95, D98, DH94, H88, UTW97]. Attribute efficient learning algorithms are standard PAC algorithms with the additional constraint that the sample complexity of the algorithm must be polynomial in $\log n$ and in the number of attributes that are *relevant* to the target, which we will denote by $r$.

Most efficient PAC learning algorithms in the literature run in polynomial time but have sample size polynomial in $n$, regardless of the number of relevant variables $r$. A current challenge in learning theory is to convert these algorithms to attribute efficient algorithms that use sample size polynomial in $\log n$. Existing PAC algorithms also generally output hypotheses that are similarly of size polynomial in $n$. Thus a further challenge is to find attribute efficient algorithms that output hypotheses of size polynomial in $\log n$ (and the other parameters) and not in $n$. We call the latter *attribute efficient learning with small hypotheses.*

The first attribute efficient learning algorithm was found by Littlestone [L88] for learning threshold functions from equivalence queries only (by a standard argument [A88], this implies an attribute efficient algorithm in the PAC-model as well). However, Littlestone's Winnow algorithm does not output a small hypothesis. A number of attribute efficient learning algorithms have since been found in various learning models [B92, BH96, BHL95, D98, DH94, H88, UTW97].

An attribute efficient result by Haussler [H88] is particularly relevant to the present work. Haussler gave an attribute efficient with small hypothesis algorithm for PAC-learning a single term (conjunction of literals). Using a standard learning reduction, Haussler's result also gives an attribute efficient learning algorithm for DNF expressions having constant number of terms. PAC-learning DNF with non-constant number of terms (even with membership queries) is still one of the outstanding open problems in computational learning theory. Jackson [J97] gave a PAC-learning

ISSN 1433-8092

algorithm for DNF with membership queries under the uniform distribution. The sample size[1] used in Jackson's Harmonic Sieve algorithm is $\tilde{O}(ns^4)$ and the hypothesis size is $O(s^2 \log(1/\epsilon))$.

In this paper we give a new attribute efficient algorithm that uses sample size $\tilde{O}(s^4 \log n)$ while retaining the small hypothesis of size $O(s^2 \log(1/\epsilon))$. Notice that our sample size for learning DNF is independent of the number of relevant variables $r$. This is because large terms in the DNF may be ignored under the uniform distribution and only terms of size $O(\log(s/\epsilon))$ remain. Therefore, in terms of $s$, $r \leq s \log s$.

We also develop a new technique for finding a lower bound on the sample size needed to learn classes in the PAC-learning model with membership queries. We apply this technique to find a lower bound for PAC-learning DNF with membership queries under the uniform distribution. This lower bound will show that our algorithm for PAC-learning DNF with membership queries under the uniform distribution is almost optimal in $n$.

In the appendix, we develop some general derandomization techniques and apply them to obtain a learning algorithm for DNF that is attribute efficient in its use of random bits. Specifically, our algorithm requires $\tilde{O}(s^4 \log^4 n)$ random bits for learning DNF with membership queries under the uniform distribution.

## 2 Definitions and Notation

Let $n$ be some positive integer and let $[n] = \{1, 2, \ldots, n\}$. We consider Boolean functions of the form $f : \{0,1\}^n \to \{-1, +1\}$, the class $\mathcal{C}_n$ of such functions, and the countable union of such classes $\mathcal{C} = \bigcup_{n \geq 0} \mathcal{C}_n$. In this paper we will focus on the class of DNF expressions. A DNF formula is a disjunction of terms, where each term is a conjunction of literals. A literal is either a variable or its negation. The *size* of a DNF $f$ is the number of terms of $f$. The class of DNF formulas on $n$ variables consists of Boolean functions on $n$ inputs that can be written as a DNF formulas of size polynomial in $n$.

For $a \in \{0,1\}^n$, denote the $i$-th bit of $a$ by $a_i$. The Hamming weight of $a$, i.e., the number of ones in $a$, is denoted $wt(a)$. For $i \in [n]$, the unit vector $e_i$ is the vector of all zeros except for the $i$-th bit which is one. For $I \subseteq [n]$, the vector $e_I$ denotes the vector of all zeros except at bit positions indexed by $I$ where they are ones. We denote the bitwise exclusive-or between two vectors $a, b \in \{0,1\}^n$ by $a \oplus b$. The dot product $a \cdot b$ is defined as $\sum_{i=1}^{n} a_i b_i$. When dealing with subsets, we identify them with their characteristic vectors, i.e., subsets of $[n]$ with vectors of $\{0,1\}^n$. So for two subsets $A, B$, the symmetric difference of $A$ and $B$ is denoted by $A \oplus B$.

The asymptotic notation $\tilde{O}(f(n))$ stands for $O(f(n)poly(\log f(n)))$, where $poly(n) = n^{O(1)}$. The function $sign(x)$ returns $+1$ if $x$ is positive and $-1$ is $x$ is negative.

Let $f, h$ be Boolean functions. We say that $h$ is an *$\epsilon$-approximator* for $f$ under distribution $D$ if $\Pr_D[f \neq h] < \epsilon$. We also use the notation $f \triangle h$ to denote the symmetric difference between $f$ and $h$, i.e., $\{x : f(x) \neq h(x)\}$. The example oracle for $f$ with respect to $D$ is denoted by $EX(f, D)$. This oracle returns the pair $(x, f(x))$ where $x$ is drawn from $\{0,1\}^n$ according to distribution $D$. The membership oracle for $f$ is denoted by $MEM(f)$. On input $x \in \{0,1\}^n$, this membership oracle returns $f(x)$. The *Probably Approximately Correct* (PAC) learning model [V84] is defined as follows. A class $\mathcal{C}$ of Boolean functions is called *PAC-learnable* if there is an algorithm $\mathcal{A}$ such that for any positive $\epsilon$ (accuracy) and $\delta$ (confidence), for any $f \in \mathcal{C}$, and for any distribution $D$, with probability at least $1 - \delta$, the algorithm $\mathcal{A}(EX(f, D), \epsilon, \delta)$ produces an $\epsilon$-approximator for $f$ with respect to $D$ in time polynomial in the size $s$ of $f$, $n$, $1/\epsilon$ and $1/\delta$. We call a concept class *weakly PAC-learnable* if it is PAC-learnable with $\epsilon = 1/2 - 1/poly(n, s)$. The $\epsilon$-approximator for $f$ in this case is called a *weak hypothesis* for $f$. If $\mathcal{C}$ is PAC-learnable by an algorithm $\mathcal{A}$ that uses the membership oracle then $\mathcal{C}$ is said to be *PAC-learnable with membership queries.*

A variable or input $x_i$ to a function $f$ is relevant if $f(a) \neq f(a \oplus e_i)$, for some $a \in \{0,1\}^n$. If there exists a function $\iota(n) = o(n)$ so that $\mathcal{C}$ is PAC-learnable by an algorithm $\mathcal{A}$ that asks $poly(r, s)\iota(n)$ examples and queries, where $r$ is the number of relevant variables of the target function $f \in \mathcal{C}$,

[1] Here and elsewhere, we focus in our sample complexity bounds on $n$ and $s$, although the bounds are also polynomial in all other relevant parameters.

then $\mathcal{C}$ is said to be $\iota(n)$-*attribute efficient PAC-learnable* with membership queries. A class is *attribute-efficient PAC-learnable* if it is $\log n$-attribute efficient PAC-learnable. Moreover, if the algorithm $\mathcal{A}$ outputs an $\epsilon$-approximator $h$ of size $poly(r,s)\iota(n)$, then $\mathcal{C}$ is said to be PAC-learnable attribute efficiently with *small hypothesis*.

The Fourier transform of a Boolean function $f$ is defined as follows. Let $\hat{f}(a) = E_x[f(x)\chi_a(x)]$ be the *Fourier coefficient* of $f$ at $a \in \{0,1\}^n$, where $\chi_a(x) = (-1)^{a \cdot x}$ and the expectation is taken with respect to the uniform distribution over $\{0,1\}^n$. It is a well-known fact that any Boolean function $f$ can be represented as $f(x) = \sum_a \hat{f}(a)\chi_a(x)$, since the functions $\chi_a$, $a \in \{0,1\}^n$, form an orthonormal basis of real-valued functions over $\{0,1\}^n$. When dealing with a real-valued function $g : \{0,1\}^n \to R$, the notation $|g|$ denotes $\max\{|g(x)| : x \in \{0,1\}^n\}$. For a positive $\theta$, a Fourier coefficient $a \in \{0,1\}^n$ is called $\theta$-heavy if $|\hat{f}(a)| \geq \theta$.

Next we define some notation from coding theory [vL]. Let $\Sigma$ be a finite alphabet of size $m$. A *code* $L$ of *word length* $n$ is a subset of $\Sigma^n$. The distance between two codewords $x, y \in L$ is defined as $d(x,y) = |\{i \in [n] : x_i \neq y_i\}|$ and the *minimum distance* of $L$ is defined as

$$d(L) = \min\{d(x,y) | x, y \in L, x \neq y\}.$$

A $m$-ary $(n,d)$-code is a code over an alphabet of size $m$ of word length $n$ and minimum distance $d$.

# 3 Learning DNF

In this section we give the attribute efficient algorithm for learning DNF under the uniform distribution. Our algorithm is based on Jackson's Harmonic Sieve for learning DNF expressions [J97]. The Harmonic Sieve has sample and time complexity $poly(n,s)$. Our goal in this section is to give an algorithm that has sample complexity $poly(\log n, s)$ and that runs in polynomial time.

The Sieve itself is based on one of Freund's boosting algorithms [F90, F93]. This boosting algorithm runs in stages. In each stage it creates a new distribution and assumes that the learner can find a weak hypothesis for the target (one that $(1/2 - \gamma)$-approximates the target) under this distribution. The algorithm performs $O(\gamma^{-2}\log(1/\epsilon))$ stages.

Jackson showed that there is an algorithm that finds a weak parity hypothesis in each stage of the boosting algorithm that is a $(1/2 - 1/O(s))$-approximation of the target. He also showed that finding a weak parity hypothesis in each stage is equivalent to finding a heavy Fourier coefficient of a function $g$ that satisfies $|g| < poly(1/\epsilon)$. In the original Sieve, finding the heavy coefficient of such function is performed by an algorithm due to Goldreich and Levin [GL89] that was first applied to learning problems by Kushilevitz and Mansour[KM93]. This algorithm runs in polynomial time with sample size $poly(|g|, n, s)$.

Since the hypothesis found is $(1/2 - 1/O(s))$-approximation of the target, Freund's boosting algorithm should be run in $T = O(s^2 \log(1/\epsilon))$ stages. So the sample complexity is at most $T \cdot poly(|g|, n, s)$ (it can be made asymptotically smaller by using a single sample for all calls to the Goldreich-Levin algorithm).

Notice that to convert the above algorithm to one that is attribute efficient we simply need an algorithm that finds weak parity hypotheses with sample complexity $poly(|g|, \log n, s)$ instead of $poly(|g|, n, s)$.

In the following subsections, we begin with an algorithm by Levin [L93] for finding a heavy coefficient that has sample and time complexity $poly(|g|, n, s)$. We then show how to reduce its time complexity from an $n^2$ to an $n$ dependence. This modified algorithm is then converted to one that has sample complexity $poly(|g|, \log n, s)$, which then gives us a similar result for DNF formulas. For simplicity of exposition, we will assume in the following sections that $g$ is boolean function and therefore $|g| = 1$. The reader can easily check that the complexity remains polynomial in $|g|$ when $g$ is not boolean.

### 3.1 Levin's Algorithm

We describe Levin's algorithm [L93] for solving the following problem: Given an unknown target $f : \{0,1\}^n \to \{-1,+1\}$, a positive value $\theta$, and a membership oracle $MEM(f)$, find the index (frequency) of a *heavy* Fourier coefficient of $f$, i.e., a coefficient $a$ such that $|\hat{f}(a)| > \theta$. In the case of DNF, $\theta = 1/O(s)$, where $s$ is a size measure of the target.

Assume for the moment that we have guessed that $\hat{f}(a)$ is a heavy coefficient and we want to verify our guess. Typically, we would draw a sample $X$ of $x \in \{0,1\}^n$ uniformly at random and compute $\sum_{x \in X} f(x)\chi_a(x)/|X|$. By Chernoff bounds, for $|X|$ on the order of $\hat{f}^{-2}(a)$ we will get a good estimate of $\hat{f}(a)$ with high probability. However, notice that we do not need a completely uniform distribution to produce a coarse estimate with reasonably high probability. In particular, if we draw the examples $X$ from any pairwise independent distribution, then we can apply Chebyshev bounds and get that for $|X| \geq 2n/\hat{f}^2(a)$, with probability at least $1-1/2n$, $\mathrm{sign}(\sum_{x \in X} f(x)\chi_a(x)) = \mathrm{sign}(\hat{f}(a))$. Thus a polynomial-size sample suffices to find the sign of the coefficient with reasonably high probability, even using a distribution which is only pairwise rather than mutually independent.

Now one way to generate a pairwise independent distribution is by choosing a random $n$-by-$k$ 0-1 matrix $R$ and forming the set $Y = \{R \cdot p \mid p \in \{0,1\}^k - \{0^k\}\}$ (the arithmetic in the matrix multiplication is performed modulo 2). This set $Y$ is pairwise independent because each $n$-bit vector in $Y$ is a linear combination of random vectors, so knowing any one vector $Rp$ gives no information about what any of the remaining vectors might be, even if $p$ is known. Thus if we take $k = \lceil \log_2(2n/\hat{f}^2(a)) \rceil + 1$ and form the set $Y$ as above then with probability at least $1 - 1/2n$ over the random draw of $R$, $\mathrm{sign}(\sum_{x \in Y} f(x)\chi_a(x)) = \mathrm{sign}(\hat{f}(a))$. Now note that

$$\hat{f}(a) = E_x[f(x \oplus e_i)\chi_a(x \oplus e_i)] = (-1)^{a_i} E_x[f(x \oplus e_i)\chi_a(x)].$$

The first equality follows from the definition of Fourier coefficients by a change of variables. The second equality follows from $\chi_a(x \oplus e_i) = \chi_a(x)\chi_a(e_i) = (-1)^{a_i}\chi_a(x)$.

Now assume for the moment that $\hat{f}(a) > 0$. Then for $Y$ as above, we have that for a given $i$, $\mathrm{sign}(\sum_{x \in Y} f(x \oplus e_i)\chi_a(x)) = (-1)^{a_i}$ with probability at least $1 - 1/2n$, and thus with probability at least $1/2$ this holds simultaneously for all $i$.

We need one more observation. Instead of summing over $x$ above, we could rewrite this as a sum over $p \in P$, where $P = \{0,1\}^k - \{0^k\}$. With probability at least $1/2$, for all $i$ we have

$$\begin{aligned}
(-1)^{a_i} &= \mathrm{sign}\left(\sum_{p \in P} f((Rp) \oplus e_i)\chi_a(Rp)\right) \\
&= \mathrm{sign}\left(\sum_{p \in P} f_{R,i}(p)(-1)^{a^T \cdot R \cdot p}\right) \\
&= \mathrm{sign}\left(\sum_{p \in P} f_{R,i}(p)\chi_{a^T R}(p)\right),
\end{aligned}$$

where $f_{R,i}(p) \equiv f((Rp) \oplus e_i)$.

Now fix $z \in \{0,1\}^k$ and notice that $\widehat{f_{R,i}}(z) = 2^{-k}\sum_{p \in \{0,1\}^k} f_{R,i}(p)\chi_z(p)$. Also, adding the zero vector to the earlier sum giving $(-1)^{a_i}$ will not affect the sign of the sum if it is sufficiently bounded away from zero, which follows from our choice of $k$ (recall that it was 1 larger than apparently needed, which covers both the fact that we did not have the zero vector in $Y$ and this consideration). Therefore, for $z = a^T R$, with probability at least $1/2$ over the choice of $R$, $(-1)^{a_i} = \mathrm{sign}(\widehat{f_{R,i}}(z))$ holds for all $i$.

Of course, the problem is to find $a$, so we don't know which $z$ has this property. But because there are only $2^k \leq 8n/\hat{f}^2(a)$—a polynomial—many $z$'s, we can simply try them all. For given $R$ and $i$ we can use the Fast Fourier Transform to compute all $2^k$ of the Fourier coefficients of

**Input:** Membership oracle $MEM(f)(x,i)$ that given $x$ and $i$ returns $f(x \oplus e_i)$; number of input bits $n$; threshold $0 < \theta < 1$ such that there is at least one Fourier coefficient $\hat{f}(a)$ such that $|\hat{f}(a)| \geq \theta$
**Output:** With probability at least $1/2$ return set containing $n$-vector $a$ such that $|\hat{f}(a)| \geq \theta$

1. Define $k \equiv \lceil \log_2(2n/\theta^2) \rceil + 1$
2. Choose $n$ by $k$ matrix $R$ by uniformly choosing from $\{0,1\}$ for each entry of $R$
3. Generate the set $Y$ of $n$-vectors $\{R \cdot p \mid p \in \{0,1\}^k\}$
4. **for each** $i \in [n]$ **do**
5. **for each** $R \cdot p \in Y$ **do**
6. Call $MEM(f)(R \cdot p, i)$ to compute $f((R \cdot p) \oplus e_i) = f_{R,i}(p)$
7. **end do**
8. Compute (using FFT) $\widehat{f_{R,i}}$
9. **for each** $z \in \{0,1\}^k$
10. Compute $a_{z,i} = \text{sign}(\widehat{f_{R,i}}(z))$
11. **end do**
12. **end do**
13. For $z \in \{0,1\}^k$ define $a_z \equiv (a_{z,1}, a_{z,2}, \ldots, a_{z,n})$
14. **return** $\{a_z, -a_z \mid z \in \{0,1\}^k\}$

Figure 1: Levin's algorithm.

$f_{R,i}$ in time $k2^k$. We do this for all $n$ values of $i$ and take signs of the coefficients, giving us $2^k$ $\{-1,+1\}$-valued vectors of length $n$. Now with probability at least $1/2$ one of these vectors or the negation (depending on the sign of $\hat{f}(a)$) of one of the vectors represents $a$, the frequency of a heavy Fourier coefficient.

Figure 1 shows Levin's algorithm for finding a set containing a heavy Fourier coefficient with probability at least $1/2$. By running this algorithm multiple times, with high probability a list of indices containing a heavy coefficient is produced. If we want a single heavy coefficient, a testing phase can be performed. The overall algorithm then has sample complexity $\tilde{O}(n^2\theta^{-2})$.

Notice also that *oblivious* sampling is used by this algorithm. That is, this algorithm chooses the sample without regard to the target function. Furthermore, recall that the boosting algorithm will run this heavy-coefficient algorithm on non-boolean functions $g(x)$. It turns out that the membership oracle that returns the value $g(x)$ for a given input $x$ is simulated using the value $f(x)$. Therefore, after we have drawn a sample to learn a heavy coefficient for $f(x)$, no further sampling of $f$ is required for subsequent boosting stages. The only concern is that we draw a sufficiently large sample in the first stage so that we have sufficient probability that not only the first round, but all rounds, succeed. By standard Chernoff analysis and the fact that there are $\tilde{O}(s^2)$ boosting stages, this only increases the required sample size by a multiplicative factor that is dominated by a $\log s$ term.

Summarizing, Levin's algorithm can be used to learn DNF with sample complexity $\tilde{O}(n^2s^2)$ (in comparison, Goldreich-Levin has a bound of $\tilde{O}(ns^4)$). In the next subsection we improve on this algorithm, obtaining sample complexity $\tilde{O}(ns^2)$. This improved algorithm is in turn the basis for our final algorithm, which is attribute efficient.

### 3.2 First Improvement on Levin's algorithm

An idea for improving on Levin's algorithm is to extend his observation about the relationship between flipping a single bit of the input and the effect on the sign of the heavy Fourier coefficient. For example, notice that if flipping both $x_1$ and $x_2$ results in a sign change in the heavy coefficient and flipping $x_2$ alone does not, then this would be an indication that $a_1$ is set in the heavy index,

independent of a direct test obtained by flipping $x_1$ alone. In general, for any fixed $i, j \in [n]$,

$$\begin{aligned}\hat{f}(a) &= E_x[f(x \oplus e_{i,j})\chi_a(x \oplus e_{\{i,j\}})] \\ &= (-1)^{a_i}(-1)^{a_j} E_x[f(x \oplus e_{\{i,j\}})\chi_a(x)].\end{aligned}$$

Now assuming that $\hat{f}(a) > 0$, for pairwise independent $Y$ as before we have that for a given $i$,

$$\text{sign}(\sum_{x \in Y} f(x \oplus e_{\{i,j\}})\chi_a(x)) = (-1)^{a_i}(-1)^{a_j}$$

with probability at least $1 - 1/2n$. We also have that

$$\text{sign}(\sum_{x \in Y} f(x \oplus e_j)\chi_a(x)) = (-1)^{a_j}$$

with the same probability. Thus with probability $1 - 1/n$ both of these sums have the correct sign and can be used to solve for the value of $a_i$ (the sign of the product of the two sums gives $(-1)^{a_i}$). This gives a different way to arrive at $a_i$ than Levin's original way. We can do a similar thing using $a_k \neq a_j$ to get yet another way to compute $a_i$.

If there was sufficient independence between these different ways of arriving at values of $a_i$, then we could use many such calculations and take their majority vote to arrive at a good estimate of the value of $a_i$. We could then get by with a smaller probability of success for each of the individual calculations, say more like constant rather than $1 - 1/2n$. And if $n$ was not required in this probability, then working back we see that it could be removed from $k$ as well.

This observation forms the basis for the algorithm shown in Figure 2. First, notice that for the algorithm's choice of $k$ we have that, for $a$ such that $|\hat{f}(a)| > \theta$, with probability at least $1 - c$ over the choice of $R$

$$\text{sign}(\sum_{x \in Y} f(x)\chi_a(x)) = \text{sign}(\hat{f}(a)).$$

Furthermore, for any fixed $I \subseteq [n]$, we can similarly apply Chebyshev to $f(x \oplus e_I)$ and get that with the same probability $1 - c$ over choice of $R$,

$$\text{sign}(\sum_{x \in Y} f(x \oplus e_I)\chi_a(x)) = \text{sign}(\hat{f}(a)) \prod_{j \in I}(-1)^{a_j}. \tag{1}$$

This means that the expected fraction of $I$'s which fail to satisfy (1) for uniform random choice of $R$ is at most $c$. Therefore, by Markov's inequality, the probability of choosing an $R$ such that a $2c$ or greater fraction of the $I$'s fail to satisfy (1) is at most $1/2$. We will call such an $R$ "bad" for $a$ and all other $R$'s "good" for $a$. Furthermore, for any fixed $i \in [n]$ and for any $R$ that is good for $a$, at most a $2c$ fraction of the $I$'s fail to satisfy the following equality:

$$\text{sign}(\sum_{x \in Y} f(x \oplus e_I \oplus e_i)\chi_a(x)) = \text{sign}(\hat{f}(a)) \prod_{j \in I \oplus \{i\}}(-1)^{a_j}. \tag{2}$$

This follows because there is a one-to-one correspondence between the $I$'s that fail to satisfy this equality and those that fail to satisfy (1). Therefore, combining (1) and (2), the probability is also at most $1/2$ that for a $4c$ or greater fraction of the $I$'s, either of the following conditions holds (each condition is a conjunction of two equalities):

$$\text{sign}(\sum_{x \in Y} f(x \oplus e_I)\chi_a(x)) \neq \text{sign}(\hat{f}(a)) \prod_{j \in I}(-1)^{a_j}$$

and

$$\text{sign} \sum_{x \in Y} f(x \oplus e_I \oplus e_i)\chi_a(x)) = \text{sign}(\hat{f}(a)) \prod_{j \in I \oplus \{i\}}(-1)^{a_j},$$

**Input:** Membership oracle $MEM(f)(x, I)$ that given $x$ and $I$ returns $f(x \oplus e_I)$; number of input bits $n$; threshold $0 < \theta < 1$ such that there is at least one Fourier coefficient $\hat{f}(a)$ such that $|\hat{f}(a)| \geq \theta$

**Output:** With probability at least 1/4 return set containing $n$-vector $a_z$ such that $|\hat{f}(a_z)| \geq \theta$

1. Choose constant $0 < c < 1/8$ (different choices will give different performance for different problems)
2. Define $t \equiv 2\lceil \ln(4n)/(1-8c)^2 \rceil + 1$
3. Choose set $T$ consisting of $t$ uniform random $n$-vectors over $\{0,1\}$
4. Define $k \equiv \lceil -\log_2(c\theta^2) \rceil + 1$
5. Choose $n$ by $k$ matrix $R$ by uniformly choosing from $\{0,1\}$ for each entry of $R$
6. Generate the set $Y$ of $n$-vectors $\{R \cdot p \mid p \in \{0,1\}^k\}$
7. **for each** $I \in T$ **do**
8. **for each** $R \cdot p \in Y$ **do**
9. Call $MEM(f)(R \cdot p, I)$ to compute $f((R \cdot p) \oplus e_I) = f_{R,I}(p)$
10. **end do**
11. Compute (using FFT) $\widehat{f_{R,I}}$
12. **for each** $i \in [n]$ **do**
13. Compute $f_{R,I \oplus \{i\}}(p)$ as in line 9.
14. Compute (using FFT) $\widehat{f_{R,I \oplus \{i\}}}$
15. **end do**
16. **end do**
17. **for each** $z \in \{0,1\}^k$ **do**
18. **for each** $i \in [n]$
19. Compute $a_{z,i} = \text{sign}(\sum_{I \in T} \text{sign}(\widehat{f_{R,I}}(z) \cdot \widehat{f_{R,I \oplus \{i\}}}(z)))$
20. **end do**
21. **end do**
22. **return** $\{a_z \mid z \in \{0,1\}^k\}$

Figure 2: Improved Levin algorithm.

or

$$\text{sign}(\sum_{x \in Y} f(x \oplus e_I)\chi_a(x)) = \text{sign}(\hat{f}(a)) \prod_{j \in I} (-1)^{a_j}$$

and

$$\text{sign} \sum_{x \in Y} f(x \oplus e_I \oplus e_i)\chi_a(x)) \neq \text{sign}(\hat{f}(a)) \prod_{j \in I \oplus \{i\}} (-1)^{a_j}.$$

This in turn implies that for fixed $i$,

$$\Pr_R[\Pr_I[\text{sign}(\sum_{x \in Y} f(x \oplus e_I)\chi_a(x) \cdot \sum_{x \in Y} f(x \oplus e_I \oplus e_i)\chi_a(x)) \neq (-1)^{a_i}] \geq 4c] \leq \frac{1}{2}.$$

So, for good $R$'s, a random choice of $I$ has probability at least $1 - 4c$ of giving the correct sign for $a_i$ and probability at most $4c$ of giving the incorrect sign. If the correct sign is $+1$, then for a good $R$ and any $i$,

$$\mathbf{E}_I[\text{sign}(\sum_{x \in Y} f(x \oplus e_I)\chi_a(x) \cdot \sum_{x \in Y} f(x \oplus e_I \oplus e_i)\chi_a(x))] \geq 1 - 8c.$$

Similarly, a correct sign of $-1$ gives expected value bounded by $-1 + 8c$. So by Hoeffding, if we estimate this expected value by taking a sum over $t$ randomly chosen $I$'s, for

$$t \geq \frac{2 \ln 4n}{(1-8c)^2}, \tag{3}$$

then the sign of the estimate will be $(-1)^{a_i}$ with probability at least $1 - 1/2n$. Therefore, with probability at least $1/2$ this holds simultaneously for all $n$ possible values of $i$. In this case, we will discover all $n$ bits of the index $a$ of the heavy Fourier coefficient $\hat{f}(a)$. So overall we succeed with probability at least $1/4$ for a given choice of $R$.

This algorithm has sample complexity $\tilde{O}(n\theta^{-2})$. Once again, oblivious sampling is used, which implies a sample complexity of $\tilde{O}(ns^2)$ for DNF learning. In the next subsection we show how to change this algorithm to an attribute efficient algorithm for learning DNF.

### 3.3 Attribute efficient learning DNF

In this subsection we show that if the target have a heavy coefficient $\hat{f}(a)$ with weight $wt(a) = k$ then $a$ can be found in $poly(k, \hat{f}^{-1}(a))$ time and space complexity. We then show that the algorithm for learning DNF finds heavy coefficients with $wt(a) = O(\log s)$. With those two results we get our attribute efficient learning algorithm.

Notice first that for any $b \in \{0,1\}^n$ we have

$$\hat{f}(a) = E_x[f(x \oplus b)\chi_a(x \oplus b)] = \chi_a(b)E_x[f(x \oplus b)\chi_a(x)].$$

As in the previous section we choose $|Y| = 2^{\lceil -\log_2(c\theta^2)\rceil + 1}$ and we have, for $R$ good for $a$, with probability at least $1 - 2c$

$$\text{sign}\left(\sum_{x \in Y} f(x \oplus b)\chi_a(x)\right) = \text{sign}(\hat{f}(a))\chi_a(b)$$

and

$$\text{sign}\left(\sum_{x \in Y} f(x \oplus b \oplus d)\chi_a(x)\right) = \text{sign}(\hat{f}(a))\chi_a(b)\chi_a(d)$$

for any fixed $d \in \{0,1\}^n$. Then taking (similar to Subsection 3.2, equation 3)

$$t \geq \frac{4\ln(4(wt(a)\log n))}{(1-8c)^2}$$

we have that for good $R$ and fixed $d$, with probability at least $1 - 1/(wt(a)\log n)^2$ the estimation of

$$\mathbf{E}_b\left[\text{sign}\left(\left(\sum_{x \in Y} f(x \oplus b)\chi_a(x)\right)\left(\sum_{x \in Y} f(x \oplus b \oplus d)\chi_a(x)\right)\right)\right]$$

is equal to $\chi_a(d)$. This provides a membership oracle for $\chi_a(d)$ that *for every* $d \in \{0,1\}^n$ gives with probability at least $1 - 1/(wt(a)\log n)^2$ the true value of $\chi_a(d)$. It is known (see for example [BH96]) that there is an algorithm that runs in polynomial time, asks $O(wt(a)\log n)$ membership queries of $MEM(\chi_a)$, and with high probability finds $a$. (This algorithm is easily modified to terminate after this many queries even if the membership oracle does not represent a parity function $\chi_a$.) Since the number of membership queries of this algorithm is $O(wt(a)\log n)$ and the probability that the above approach fails to correctly answer one membership query is $1/(wt(a)\log n)^2$, the probability of failing to correctly answer any membership queries of the algorithm is $1/(wt(a)\log n) = o(1)$. Using this membership query algorithm for each of the $\tilde{O}(\theta^{-2})$ values of $z$, we will with probability at least $1/4$ find $a$. This algorithm requires sample size of $\tilde{O}((wt(a)\log n)\theta^{-2})$, where now the $\tilde{O}$ does not suppress factors of $\log n$ but does suppress sublogarithmic factors of $n$ as well as logarithmic factors of other parameters.

We now show that in the Harmonic Sieve algorithm for learning DNF, at every stage there is a parity $\chi_a(x)$ with $wt(a) = O(\log(s/\epsilon))$ that correlates well with the target DNF. First, all of the

distributions generated by the boosting portion of the algorithm are polynomially bounded away from the uniform distribution. Specifically, the distribution $D$ at stage $k$ satisfies

$$D(x) \leq \frac{poly(1/\epsilon)}{2^n}.$$

Any term $T$ of size at least $|T| \geq c \log(s/\epsilon)$ over this distribution satisfies

$$\Pr[T = 1] \leq \frac{poly(1/\epsilon)}{2^{|T|}} \leq \frac{1}{poly(s/\epsilon)}$$

for some constant $c$. Therefore, if the DNF, with respect to sampling according to $D$, is not very nearly always false, then there is a term in the DNF of size less than $c \log(s/\epsilon)$ that correlates well with the DNF. By the proof of a fact in [J97] it can be shown that there is a parity that depends on less than $c \log(s/\epsilon)$ variables that correlates well with the DNF. On the other hand, if the DNF is very nearly always false (expected value near $-1$ when viewed as a $\{-1, +1\}$ function), we just take $-1$ as the weak hypothesis.

This algorithm for finding a heavy coefficient therefore requires sample size $\tilde{O}(\theta^{-2} \log n)$. However, this algorithm does not use oblivious sampling. Therefore, when used as a subprogram of the Harmonic Sieve, the sampling will have to be repeated $\tilde{O}(s^2)$ times, giving an overall sample complexity for DNF learning of $\tilde{O}(s^4 \log n)$. We have therefore achieved an attribute efficient algorithm, although at some cost in terms of dependence on $s$.

# 4 Lower Bounds

In this section we give some lower bounds on the sample complexity for learning classes of boolean functions under fixed distributions. We prove the following results.

**Theorem 1** *Let $C$ be a class of boolean functions and $0 < \epsilon < 1$ be fixed. Also, let $C_\epsilon \subseteq C$ be such that for every $f_1, f_2 \in C_\epsilon$ we have $\Pr_D[f_1 \neq f_2] \geq 2\epsilon$. Any PAC(MQ)-learning algorithm that learns $C$ under the distribution $D$ with accuracy $\epsilon$ and with confidence $1 - \delta$ uses at least*

$$l = \left\lfloor \log |C_\epsilon| - \log \frac{1}{1-\delta} \right\rfloor - 1$$

*queries.*

**Proof:** Let $\mathcal{A}^f_{r,s}$ be a randomized algorithm that uses a sequence of random bits $r$, is given a set $s$ of $m_1$ random examples, and asks $m_2$ membership queries to $f$, where $m_1 + m_2 < l$. Suppose for every $f \in C_\epsilon$ we have

$$\Pr_{s,r}[D(\mathcal{A}^f_{r,s} \Delta f) \leq \epsilon] \geq 1 - \delta.$$

This implies that there is a specific sequence $r_0$ of bits and specific set $s_0$ of $m_1$ examples such that

$$D(\mathcal{A}^f_{r_0,s_0} \Delta f) \leq \epsilon \tag{4}$$

for at least $(1-\delta)|C_\epsilon|$ of the functions in $C_\epsilon$. Since $\mathcal{A}_{r_0,s_0}$ asks less than $l$ queries, each query gives a response in $\{0, 1\}$, and $2^l < (1-\delta)|C_\epsilon|$, there must be two functions $f_1$ and $f_2$ in $C_\epsilon$ and satisfying (4) for which the algorithm outputs the same hypothesis $\mathcal{A}^{f_1}_{r_0,s_0} = \mathcal{A}^{f_2}_{r_0,s_0} = h$. Now

$$2\epsilon < D(f_1 \Delta f_2) \leq D(h \Delta f_1) + D(h \Delta f_2),$$

and therefore there is an $i$ such that $D(h \Delta f_i) > \epsilon$. This is a contradiction. □

The following lemma is an easy consequence of the Varshamov-Gilbert bound from coding theory (see [vL], Chapter 5).

**Lemma 2** *Let $\Sigma$ be an alphabet with $|\Sigma| = m$ symbols. For $m > 2$ there is a code $L \subseteq \Sigma^n$ of minimum distance $cn$ and size*

$$|L| \geq \left(\frac{m^{1-c}}{2}\right)^n.$$

We now show that to learn a DNF of size $s$ (and $\epsilon < 1/4$) we need sample size nearly $s \log n$.

**Theorem 3** *Learning the class of DNF expressions of size $s$ under the uniform distribution requires sample size $\Omega(s \log(n - \log s))$.*

**Proof:** Let $r = n - \log s$ and $t = \log s$, and let $x_1, \ldots, x_t$ and $y_1, \ldots, y_r$ be the $n$ variables of the DNF. For $a \in \{0,1\}^t$ we define $x^a = x_1^{a_1} \cdots x_t^{a_t}$ where $x_i^{a_i} = x_i$ if $a_i = 1$ and $x_i^{a_i} = \bar{x}_i$ if $a_i = 0$. Let $\Sigma = \{0, 1, y_1, \ldots, y_r, \bar{y}_1, \ldots, \bar{y}_r\}$. Now define the set of DNF formulas

$$C' = \left\{ \bigvee_{a \in \{0,1\}^t} x^a y_a \;\middle|\; (y_a)_{a \in \{0,1\}^t} \in \Sigma^{2^t} \right\}.$$

That is, each DNF expression in $C'$ has $2^t$ terms, each containing the $t$ $x$ variables plus one symbol from $\Sigma$ (either a $y$ variable or a constant). Furthermore, each term in one of these expressions sets the senses (positive or negated) of the $x$ variables differently, and all possible senses are represented. Thus the number of terms in each DNF in $C'$ is $2^t = s$. By Lemma 3 there is $L \subset \Sigma^s$ of minimum distance $cs$ and size

$$|L| \geq \left(\frac{(2r+2)^{1-c}}{2}\right)^s.$$

Fix such an $L$ and define a new set $C$ that is a subset of $C'$:

$$C = \left\{ \bigvee_{a \in \{0,1\}^t} x^a y_a \;\middle|\; (y_a)_{a \in \{0,1\}^t} \in L \right\}.$$

For $y \in L$ we write $f_y = \bigvee_{a \in \{0,1\}^t} x^a y_a$. That is, $f_y$ represents the DNF expression in which the $i$th symbol in the string $y$ is the value of the $y_a$ variable in the $i$th term of the DNF.

Now notice that for every $y^{(1)}$ and $y^{(2)}$ in $L$ we have $f_{y^{(1)}} \oplus f_{y^{(2)}} = f_{y^{(1)} \oplus y^{(2)}}$. This follows from the fact that $f_y$ can be also written as $\oplus_{a \in \{0,1\}^t} x^a y_a$. Now

$$\begin{aligned}
\Pr[f_{y^{(1)}} \neq f_{y^{(2)}}] &= \Pr[f_{y^{(1)}} \oplus f_{y^{(1)}} = 1] \\
&= \Pr[f_{y^{(1)} \oplus y^{(2)}} = 1] \\
&= E[f_{y^{(1)} \oplus y^{(2)}}] \\
&= \frac{1}{s} \sum_{a \in \{0,1\}^t} E[y_a^{(1)} \oplus y_a^{(2)}] \\
&\geq \frac{c}{2}.
\end{aligned}$$

The latter is because $y_a^{(1)}$ and $y_a^{(2)}$ are different in at least $cs$ entries and each $y_a^{(1)} \oplus y_a^{(2)}$ that is not zero has expectation at least $1/2$.

By Theorem 1, any PAC(MQ) learning algorithm with $\epsilon = c/4$ for this class needs at least

$$\log \left(\frac{(2r+2)^{1-c}}{2}\right)^s$$

queries for any constant $c < 1$. This gives a sample complexity of $\Omega(s \log(n - \log s))$.

## References


[A88] D. Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319-342, 1988.

[B92] A. Blum. Learning Boolean Functions in an Infinite Attribute Space, *Machine Learning*, 9(4):373-386, 1992.

[BHL95] A. Blum, L. Hellerstein and N. Littlestone. Learning in the Presence of Finitely and Infinitely Many Irrelevant Attributes. *Journal of Computer and System Sciences*, 50(1):32-40, 1995.

[BH96] N. H. Bshouty, L. Hellerstein. Attribute-efficient Learning in Query and Mistake-bound Models. In *Proceeding of the Ninth Annual ACM Workshop on Computational Learning Theory*, 235-243, 1996.

[DH94] A. Dhagat and L. Hellerstein. PAC Learning with Irrelevant Attributes. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, 64-74, 1994.

[D98] P. Damaschke. Adaptive versus Nonadaptive Attribute-efficient Learning. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 590-596, 1998.

[F90] Y. Freund. Boosting a Weak Learning Algorithm by Majority. In *Proceedings of 3nd Annual Workshop on Computational Learning Theory*, 202-216, 1990.

[F93] Y. Freund. An Improved Boosting Algorithm and Its Implications on Learning Complexity. In *Proceedings of the 5th Ann. Workshop on Computational Learning Theory*, 391-398, 1992.

[GL89] O. Goldreich and L. Levin. A Hardcore Predicate for all One-Way Functions. In *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing*, pages 25-32, 1989.

[H88] D. Haussler. Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework. *Artificial Intelligence*, 36(2), 177-222, 1988.

[J97] J. C. Jackson. An Efficient Membership-Query Algorithm for Learning DNF with Respect to the Uniform Distribution. *Journal of Computer and System Sciences*, 55(3):414-440, 1997.

[KM93] E. Kushilevitz and Y. Mansour. Learning Decision Trees using the Fourier Spectrum. *SIAM Journal on Computing*, 22(6): 1331-1348, 1993.

[L88] N. Littlestone. Learning when Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning*, 2(4):285-318, 1988.

[L93] L. Levin. Randomness and Non-determinism. *Journal of Symbolic Logic*, 58(3):1102-1103, 1993.

[M92] Y. Mansour. An $O(n^{\log\log n})$ Learning Algorithm for DNF under the Uniform Distribution. In *Proceedings of Fifth Annual Conference on Computational Learning Theory*, pages 53–61, 1992.

[NN93] J. Naor and M. Naor. Small-Bias Probability Spaces: Efficient Constructions and Applications. *SIAM Journal on Computing*, 22(4):838-856, 1993.

[UTW97] R. Uehara, K. Tsuchida and I. Wegener. Optimal Attribute-efficient Learning of Disjunction, Parity, and Threshold Functions. In *EuroCOLT' 97, LNAI 1208 Springer*, 171-184, 1997.

[V84] L. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134-1142, 1984.

[vL] J.H. van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1982.

# A Towards a derandomization

In this section we sketch an algorithm for learning DNF with respect to uniform that is attribute efficient in terms of its use of random bits (its *randomness complexity*) as well as its sample complexity. The algorithm is actually somewhat simpler conceptually than our previous attribute efficient algorithm, but it does not produce as tight a sample complexity bound.

We begin by describing a technique to slightly derandomize Levin's basic algorithm [L93] and then show how to apply this to DNF learning. It should be noted that Kushilevitz and Mansour [KM93] also showed how to derandomize another algorithm for solving the same parity-finding problem solved by Levin, but they assumed that a quantity called the $L_1$-norm of the target function is polynomially bounded and is known. However, the $L_1$ norm is not polynomially bounded for the class of polynomial-size DNF expressions [M92].

We use the $\lambda$-*bias distributions* studied in [NN93] which is defined as follows. Let $D : \{0,1\}^n \to [0,1]$ be a probability distribution. $D$ is called $\lambda$-bias if for all $a \in \{0,1\}^n - \{0_n\}$ we have $|\hat{D}(a)| \leq \lambda 2^{-n}$ In [NN93], an explicit construction of a $\lambda$-bias probability distribution of size $O((n/\lambda)^2)$ is given. Note that this requires $O(\log(n/\lambda))$ random bits.

Recall that Levin's algorithm constructs the $n \times k$ matrix of Boolean entries by selecting each entry randomly and independently. So it uses $kn$ random bits altogether. We modify this algorithm by choosing $k$ *independent* column vectors according to a $\lambda$-bias distribution $D$ over $\{0,1\}^n$. Note that the number of random bits required is $O(k \log(n/\lambda))$.

First we need to define the notion of pairwise dependent random variables. A sequence of random variables $X_1, \ldots, X_m$ is called *pairwise $\delta$-dependent* if for every $1 \leq i \neq j \leq m$ and for every $a, b$, we have

$$|\Pr[X_i = a, X_j = b] - \Pr[X_i = a]\Pr[X_j = b]| \leq \delta.$$

The following is an easy property of pairwise dependent random variables.

**Claim 4** *Let $X_1, \ldots, X_m \in \{-1, +1\}$ be pairwise $\delta$-dependent random variables. Then*

$$|E[X_i X_j] - E[X_i]E[X_j]| \leq 4\delta.$$

The following gives a version of Chebyshev's inequality for pairwise $\delta$-dependent random variables.

**Lemma 5** *(Chebyshev's inequality) Let $X_1, \ldots, X_m \in \{-1, +1\}$ be pairwise $\delta$-dependent random variables. Suppose that for each $i \in [m]$, $E[X_i] = \mu$ and $Var[X_i] = \sigma^2$. Then*

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

*Proof* The expression on the lefthand side can be bounded from above by

$$\begin{aligned}
\frac{E[(\frac{1}{m}\sum_{i=1}^{m} X_i - \mu)^2]}{\mu^2} &\leq \frac{1}{m^2\mu^2} E[(\sum_i (X_i - \mu))^2] \\
&= \frac{1}{m^2\mu^2}\left(\sum_i E[(X_i - \mu)^2] + \sum_{i \neq j} E[(X_i - \mu)(X_j - \mu)]\right) \\
&= \frac{1}{m^2\mu^2}\left(\sum_i var[X_i] + \sum_{i \neq j}(E[X_i X_j] - \mu^2)\right) \\
&\leq \frac{1}{m^2\mu^2}\left[m\sigma^2 + \binom{m}{2}(4\delta)\right] \\
&\leq \frac{\sigma^2}{m\mu^2} + \frac{2\delta}{\mu^2}.
\end{aligned}$$

□

Setting $\delta = 1/(2m)$ and using the fact that $\sigma^2 = 1 - \mu^2 \leq 1$, yields an upper bound of $\frac{2}{m\mu^2}$ in the above. This is only a factor of 2 larger than the bound obtained from Levin's analysis.

Next, we prove that by choosing $k$ independent column vectors from $\{0,1\}^n$ according to an $\lambda$-bias distribution $D$ the resulting random variables, i.e. $X_i = Rp_i$, $p_i \in \{0,1\}^k \setminus \{0_k\}$, are pairwise $\delta$-dependent, for $\delta = 4\lambda/2^n$. So, if $4\lambda/2^n \leq 1/(2m)$ then the analysis in Subsection 3.1 still holds.

**Claim 6** *Let $R$ be a $n$-by-$k$ matrix with entries from $\{0,1\}$ constructed by selecting $k$ random column vectors from $\{0,1\}^n$ according to a $\lambda$-bias distribution $D$ over $\{0,1\}^n$. Let $X_i = Rp^{(i)}$, for $p^{(i)} \in \{0,1\}^k \setminus \{0_k\}$. Then $X_1, \ldots, X_{2^k-1}$ are pairwise $(4\lambda/2^n)$-dependent random variables.*

*Proof* We observe first that if $D$ is an $\lambda$-bias distribution over $\{0,1\}^n$ then $|D(x) - 2^{-n}| \leq \lambda$. Recall that since for all $a \neq 0_n$ $|\hat{D}(a)| \leq \frac{\lambda}{2^n}$ and $\hat{D}(0_n) = 2^{-n}$,

$$|D(x) - 2^{-n}| = |\sum_{a \neq 0_n} \hat{D}(a)\chi_a(x)| \leq \sum_{a \neq 0_n} |\hat{D}(a)| \leq \lambda.$$

Let $p$ and $q$ be any elements of $\{0,1\}^k \setminus \{0_k\}$. We need to prove that for any $p \neq q$ and any $a, b \in \{0,1\}^n$

$$|\Pr[Rp = a, Rq = b] - \Pr[Rp = a]\Pr[Rq = b]| \leq 4\lambda/2^n$$

Consider first $\Pr[Rp = a]$. Assume without loss of generality that $p_k \neq 0$. Then after choosing the first $k-1$ columns of $R$, the value of the last column is uniquely determine for the equation $Rp = a$ to hold, say the last column must equal to $\alpha \in \{0,1\}^n$. Hence $\Pr[Rp = a] = D(\alpha)$.

Now consider $\Pr[Rp = a, Rq = b]$, with $p \neq q \in \{0,1\}^k$ and $a, b \in \{0,1\}^n$. Assume without loss of generality that the following condition is true.

$$\begin{vmatrix} p_{k-1} & q_{k-1} \\ p_k & q_k \end{vmatrix} \neq 0$$

where the arithmetic is over $GF(2)$. After choosing the first $k-2$ columns, there is a unique solution for the $(k-1)$th and $k$th columns, say $\alpha$ and $\beta$. Then

$$\Pr[Rp = a, Rq = b] = D(\alpha)D(\beta)$$

since we draw independent columns. The difference between two quantities of the form $D(\alpha)D(\beta)$ is at most the difference between $(2^{-n} - \lambda)^2$ and $(2^{-n} + \lambda)^2$ which is at most $4\lambda/2^n$. This completes the proof. □

In addition to changing Levin to use $R$ as described above, we will also change the value of $k$ (number of columns in $R$) to be:

$$k = \left\lceil \log \frac{(wt(a)\log n)^2}{\theta^2} \right\rceil + 1. \tag{5}$$

Then by Chebyshev's inequality, for any $y \in \{0,1\}^n$, the probability that $\chi_a(y)$ is not equal to the sign of

$$(\sum_{x \in Y} f(x)\chi_a(x))(\sum_{x \in Y} f(x \oplus y)\chi_a(x))$$

is at most $1/(wt(a)\log n)^2$. So with $2^{k+1}$ membership queries to $f$ we can simulate a query to $\chi_a(y)$ that succeeds with probability at least $1 - 1/(wt(a)\log n)^2$, and the analysis given in Subsection 3.3 still holds.

The algorithm for determining $a$ from $MEM(\chi_a)$ can also be derandomized to require only $O(\log n)$ random bits. The simulated membership queries above are deterministic for fixed $R$, so require no additional random bits. Since the attribute efficient parity learning algorithm asks $wt(a)\log n$ membership queries for each of $2^k$ vectors $z$, and since there are $\tilde{O}(s^2)$ total boosting iterations to learn DNF in the Harmonic Sieve algorithm, the total number of random bits used is

$$\tilde{O}(s^4 \log^4 n).$$

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'