On the Fourier Spectrum of Monotone Functions

NADER H. BSHOUTY AND CHRISTINO TAMON

University of Calgary, Calgary, AB Canada

Abstract. In this paper, monotone Boolean functions are studied using harmonic analysis on the cube. The main result is that *any* monotone Boolean function has most of its power spectrum on its Fourier coefficients of "degree" at most $O(\sqrt{n})$ under *any* product distribution. This is similar to a result of Linial et al. [1993], which showed that AC^0 functions have almost all of their power spectrum on the coefficients of degree, at most $(\log n)^{O(1)}$, under the uniform distribution. As a consequence of the main result, the following two corollaries are obtained:

—For any $\epsilon > 0$, monotone Boolean functions are PAC learnable with error ϵ under product distributions in time $2^{\tilde{O}((1/\epsilon)\sqrt{n})}$.

—Any monotone Boolean function can be approximated within error ϵ under product distributions by a non-monotone Boolean circuit of size $2^{\tilde{O}(1/\epsilon\sqrt{n})}$ and depth $\tilde{O}(1/\epsilon \sqrt{n})$.

The learning algorithm runs in time subexponential as long as the required error is $\Omega(1/(\sqrt{n} \log n))$. It is shown that this is tight in the sense that for any subexponential time algorithm there is a monotone Boolean function for which this algorithm cannot approximate with error better than $\tilde{O}(1/\sqrt{n})$.

The main result is also applied to other problems in learning and complexity theory. In learning theory, several polynomial-time algorithms for learning some classes of monotone Boolean functions, such as Boolean functions with $O(\log^2 n/\log \log n)$ relevant variables, are presented. In complexity theory, some questions regarding monotone NP-complete problems are addressed.

Categories and Subject Descriptors: F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*probabilistic computation*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computation of transforms, computation on polynomials*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*probabilistic algorithms*; I.2.6 [**Artificial Intelligence**]: Learning—*concept learning*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Approximation, circuits, complexity, Fourier transform, functions, harmonic analysis learning, monotone Boolean, monotone circuits

1. *Introduction*

In recent years, harmonic analysis on the cube or the discrete Fourier transform of Boolean functions has emerged as one of the most versatile tools in theoreti-

Authors' address: Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, AB Canada T2N 1N4; e-mail: {bshouty,tamon}@cpsc.ucalgary.ca.

Journal of the ACM, Vol. 43, No. 4, July 1996, pp. 747–770.

cal computer science. It has found various applications in areas such as circuit complexity [Kahn et al. 1988; Bruck and Smolensky 1992], computational learning theory[1], cryptography [Coppersmith et al. 1993] and others.

Although harmonic analysis on the cube was originally introduced by Kahn, Kalai, and Linial [1988] for the purpose of studying the "influence" of variables on Boolean functions, the first paper that introduced the Fourier transform to learning theory was the beautiful paper of Linial, Mansour, and Nisan [1993]. In the latter paper, they proved that AC^0 functions have almost all of their power spectrum on the Fourier coefficients of Hamming weight $(\log n)^{O(1)}$. This result led to the PAC-learnability of AC^0 functions in time $n^{poly(\log n)}$ under the uniform distribution. The impact of the Fourier transform on learning theory cannot be underestimated judging from the sequence of papers that followed the paper [Linial et al. 1993]. This technique alone has made some outstanding results possible in recent years culminating in Jackson's result [Jackson 1994] on the PAC learnability of DNF formulas under the uniform distribution with membership queries.

In this paper, we study monotone Boolean functions using harmonic analysis on the cube. Our main result is that *any* monotone Boolean function has most of its power spectrum on its Fourier coefficients of *degree* at most $O(\sqrt{n})$ under *any* product distribution. Based on our main result we derive two important implications in learning theory and circuit complexity.

—Given any $\epsilon > 0$, the class of monotone Boolean functions is PAC learnable with error ϵ under product distributions in time $2^{\tilde{O}(1/\epsilon\sqrt{n})}$.

—Given any $\epsilon > 0$ and any monotone Boolean function f, there exists a nonmonotone Boolean circuit of size $2^{\tilde{O}(\sqrt{n})}$ and depth $\tilde{O}(1/\epsilon \sqrt{n})$ that approximates f with error ϵ under product distributions.

The time complexity of our learning algorithm is subexponential as long as $\epsilon = \Omega(1/(\sqrt{n} \log n))$. We will show that this is the best possible error rate for any subexponential time learning algorithm. To the best of our knowledge, the above learning result is the first subexponential PAC learning algorithm for monotone Boolean functions (even for monotone Boolean functions which require an exponential circuit size) and the second result is the first approximation result for monotone Boolean functions using nonmonotone Boolean circuit of subexponential size and sublinear depth.

We also introduce and study a new measure of complexity for probability distributions called the *convex dimension* of a distribution. We prove that if a concept class is approximable under a collection of distributions then it is also approximable under any distribution that belongs to the convex combination of that collection. The convex dimension $cdim(D)$ of a distribution D is the minimal number of product distributions such that D is in their convex combination. We note that our previous results hold also for any distribution D modulo a complexity factor of $cdim(D)$ and in particular, our algorithm is subexponential for any distribution D with a subexponential convex dimension.

[1] See Linial et al. [1993], Furst et al. [1991], Kushilevitz and Mansour [1993], Aiello and Mihail [1991], Bellare [1992], Mansour [1992], Blum et al. [1994], and Jackson [1994].

Further applications of the main result include some improvements on two efficient, that is, polynomial time, learning algorithms for monotone Boolean functions. Kearns and Valiant [1989] proved that any monotone Boolean function is efficiently weakly PAC learnable with error $(1/2) - (1/2n)$ under the uniform distribution. We improve their result by showing that there is an efficient weak PAC learning algorithm under the uniform distribution with error $(1/2) - \Omega((\log^2 n)/n)$. Our result is based on a direct application of a result due to Kahn et al. [1988]. Furthermore, under any product distribution, we show that there is an efficient weak PAC learning algorithm with error $(1/2) - (c/n)$, for any constant c.

Our second improvement is on Sakai and Maruoka's [1994] PAC learnability result of monotone $O(\log n)$-term DNF under the uniform distribution. We improve their result in two ways. First, we prove that their result extends to any constant-bounded product distribution, and second, we give an extension to a more general concept class. In particular, let $A(k)$ be the class of Boolean functions of the form $f(T_1, \ldots, T_k)$ where f is a monotone Boolean function on k inputs and each T_i is a monotone conjunction or disjunction. Our other results state that there are polynomial time PAC learning algorithms for $A(\log n)$, for $A(O(\log^2 n/(\log \log n)^3))$, and for monotone Boolean functions that depend on $O(\log^2 n/\log \log n)$ relevant variables (the last two results require the error ϵ to be constant).

Finally, we apply our results to monotone graph properties. We show that there is a Boolean circuit of size $2^{\tilde{O}(n/\epsilon\sqrt{r(n)})}$ and depth $\tilde{O}((n/\epsilon)\sqrt{r(n)})$ that approximates within error ϵ any monotone graph property with a threshold function of $r(n)$. For example, there is a Boolean circuit of size $n^{O(1/\epsilon\sqrt{\log n})}$ that approximates to within error ϵ the Hamiltonian property on random graphs $G(n, p)$, for any p. We also discuss the connection of this work with a related result [Bollobás et al. 1987].

All of the algorithms considered in this papers fall in the category of a statistical query algorithm, and hence, by the result of Kearns [1993], are noise-tolerant.

The paper is organized as follows: Section 2 is devoted to a description of the learning models considered in this paper. Section 3 describes notations and the basic theory of the Fourier transform for Boolean functions. It also contains some basic facts that will be required in proving some of the later results. In Section 4, we prove our main spectral characterization of Boolean functions that is based on influence and the average sensitivity. The next section, Section 5, is devoted to learning monotone Boolean functions. This section also contains some lower bound results that shows near optimality of the learning results. Section 6 describes the application of the main result to the circuit approximation of monotone Boolean functions and Section 7 considers the approximation of monotone graph properties. Finally, Section 8 discusses applications of the main result in deriving some efficient learning algorithms.

2. *The Learning Model*

The learning model considered in this paper is the *Probably Approximately Correct* (PAC) learning model introduced by Valiant [1984] and its weak variant introduced by Kearns and Valiant [1989]. Let C_n be a class of Boolean functions over n variables, let D be a probability distribution over $\{0, 1\}^n$, and let $f \in C_n$ be a target function. The learning algorithm has access to an example oracle $EX(D, f)$ which generates random labeled examples $(a, f(a))$, where $a \in \{0, 1\}^n$ is drawn according to distribution D. Given any positive ϵ and δ, after observing some random examples, the learning algorithm must output a hypothesis h that satisfies

$$\Pr[D(h\Delta f) \le \epsilon] \ge 1 - \delta,$$

where $h\Delta f = \{x : h(x) \ne f(x)\}$. The above probability is taken with respect to the random examples seen by the algorithm and some internal randomization in the algorithm. The running time of the learner will depend on n, $1/\epsilon$, $1/\delta$, and the size $s(f)$ of f (under some predetermined representation). If there is such a learning algorithm that succeeds for all $f \in C_n$ then C_n is PAC learnable under distribution D. We sometimes refer to such a learning algorithm as an (ϵ, δ) PAC learning algorithm. We say that C_n is weakly PAC learnable under distribution D if there is a fixed polynomial p and a learning algorithm that succeeds for an error $\epsilon = (1/2) - (1/p(n, s(f)))$.

3. *Preliminaries*

In this section we review some notation and some standard facts about the discrete Fourier transform of Boolean functions.

When we write log we mean $\log_2$. We will use the shorthand $[n]$ for the set $\{1, 2, \ldots, n\}$ and the Iversonian I[statement] to mean 1 if the statement is true and 0 otherwise. For $a \in \{0, 1\}^n$, let a_i denote the ith bit of a. The vector $e_i \in \{0, 1\}^n$ denotes the vector with all zeros except for the ith bit which is 1. The Hamming weight of a, that is, the number of ones in a, is denoted by $|a|$.

Let $f: \{0, 1\}^n \to \{-1, +1\}$ be a Boolean function. Let D be a product distribution over $\{0, 1\}^n$ with $\Pr[x_i = 1] = \mu_i$. Thus for $a \in \{0, 1\}^n$ we have the distribution of a is $D(a) = \Pi_{a_i=1} \mu_i \, \Pi_{a_i=0} (1 - \mu_i)$. The distribution D is called *constant bounded* if there exists a constant $c \in (0, 1)$ independent of n such that for all i we have $\mu_i \in [c, 1 - c]$. The standard deviation of x_i is defined as

$$\sigma_i = \sqrt{\mu_i(1 - \mu_i)}.$$

The *influence* of variable x_i on f (see Kahn et al. [1988] and Hancock and Mansour [1991]) over a product distribution D is defined as the probability that $f(x)$ differs from $f(x \oplus e_i)$ when x is chosen according to D. Here $x \oplus e_i$ means x with its ith bit flipped. We will use the notation $I_{D,i}(f)$ to denote the above probability. Often we will use the restriction notation of functions, $f_0 = f|_{x_i \leftarrow 0}$ and $f_1 = f|_{x_i \leftarrow 1}$. With this notation we have that for any Boolean function f

$$I_{D,i}(f) = E_D[I[f(x) \ne f(x \oplus e_i)]] = \frac{1}{2} E_D[|f_1 - f_0|] = \frac{1}{4} E_D[(f_1 - f_0)^2].$$

If f is a monotone Boolean function, that is, $f_0 \le f_1$ always, then this simplifies to

$$I_{D,i}(f) = \frac{1}{2} E_D[f_1 - f_0].$$

To facilitate stating some of our results we introduce the following notion of *influence norm* $I_D(f)$ of f with respect to a product distribution D:

$$I_D(f) = \sqrt{\sum_{i=1}^{n} (2\sigma_i I_{D,i}(f))^2}.$$

The *sensitivity* of f at a point $a \in \{0, 1\}^n$, denoted by $s_a(f)$, is defined as the number of neighbors of a (in the standard n-cube ordering) whose f-values differ from $f(a)$. More formally,

$$s_a(f) = |\{i \in [n] : f(a \oplus e_i) \neq f(a)\}| = \sum_{i=1}^{n} I[f(a) \neq f(a \oplus e_i)].$$

The *average sensitivity* of f with respect to a product distribution D, denoted $s_D(f)$, is defined as

$$s_D(f) = E_D[s_x(f)].$$

It is well known that the average sensitivity is equivalent to the sum of the influences, as seen from the following simple derivation:

$$s_D(f) = E_D[s_x(f)] = \sum_{i=1}^{n} E_D[I[f(x) \neq f(x \oplus e_i)]] = \sum_{i=1}^{n} I_{D,i}(f).$$

The Fourier transform of Boolean functions over product distribution is defined as follows [Furst et al. 1991]. First we define the inner product over the 2^n-dimensional vector space of all real-valued functions over $\{0, 1\}^n$ as follows:

$$(f, g)_D = \sum_{x} D(x) f(x) g(x) = E_D[fg].$$

Now let $z_i(x) = (\mu_i - x_i)/\sigma_i$. Note that z_i has mean zero and variance one (i.e., it is standard normal). Next we define the basis function

$$\phi_a(x) = \prod_{a_i=1} z_i(x).$$

These basis functions satisfy the following properties.

(1) *decomposable*. $\phi_{ab}(xy) = \phi_a(x)\phi_b(y)$, where xy is the concatenation of strings x and y (possibly of different lengths).

(2) *orthonormal*

$$(\phi_a, \phi_b)_D = \begin{cases} 1 & \text{if} \quad a \neq b \\ 0 & \text{otherwise}. \end{cases}$$

Given the orthonormality of these ϕ_as we get the Fourier representation of any Boolean function f as

$$f = \sum_a \tilde{f}(a)\phi_a,$$

where $\tilde{f}(a) = (f, \phi_a)_D = E_D[f\phi_a]$. Also because of orthonormality we have Parseval's equation:

$$1 = E_D[f^2] = \sum_a \tilde{f}^2(a).$$

Finally, note $\tilde{f}(0_n) = E_D[f]$, where 0_n is the n-bit all-zero vector.

For the case of D being the uniform distribution, the following notations are used: χ_a in place of ϕ_a and $\hat{f}(a)$ in place of $\tilde{f}(a)$. Note that in this case, $\mu_i = \sigma_i = 1/2$, for all $i \in [n]$.

In this paper we will deal with two different notions of *approximation* to a Boolean function. For the first type, we say that a Boolean function h approximates another Boolean function f to within error ϵ in the *Boolean sense* if

$$\Pr[h(x) \neq f(x)] \leq \epsilon.$$

On the other hand, we say that a real-valued function h approximates another function f to within error ϵ in the *mean square sense* if

$$E[(h(x) - f(x))^2] \leq \epsilon.$$

Unless otherwise stated, we will use the *Boolean sense* as the default case.

In most cases, we will appeal to the following version of Chernoff–Hoeffding bounds [Hagerup and Rub 1989; Mansour 1995].

THEOREM 3.1. (CHERNOFF–HOEFFDING BOUNDS) *Let $x_1, \ldots, x_m$ be independent identically distributed random variables with $E[x_i] = p$, $|x_i| \leq B$, and let $s_m = x_1 + \cdots + x_m$. Then*

$$m \geq \frac{2B^2}{\epsilon^2} \ln \frac{2}{\delta} \text{ implies } Pr\left[\left|\frac{s_m}{m} - p\right| > \epsilon\right] \leq \delta.$$

4. *Spectral Lemmas*

We are now ready for a lemma which relates the influence and the Fourier transform of Boolean functions. The next lemma is a folklore result whose proof we include for completeness.

LEMMA 4.1. *For any Boolean function f, for any product distribution D and for any $i \in [n]$,*

$$4\sigma_i^2 I_{D,i}(f) = \sum_{a:a_i=1} \tilde{f}^2(a).$$

PROOF. Without loss of generality, let $i = 1$. First recall that $I_{D,1}(f) = 1/4E_D[(f_0 - f_1)^2]$. Applying Parseval to the right hand side gives

$$I_{D,1}(f) = \frac{1}{4} \sum_{b\in\{0,1\}^{n-1}} (\tilde{f}_0 - \tilde{f}_1)^2(b) = \frac{1}{4} \sum_{b\in\{0,1\}^{n-1}} (\tilde{f}_0(b) - \tilde{f}_1(b))^2.$$

We now find some relation for $\tilde{f}_0$ and $\tilde{f}_1$. Recall that

$$f(x) = \sum_a \tilde{f}(a)\phi_a(x) = \sum_{0b} \tilde{f}(0b)\phi_b(y) + \sum_{1b} \tilde{f}(1b)\phi_b(y)\,\frac{\mu_1 - x_1}{\sigma_1}.$$

The last step uses the decomposable property of ϕ_a. From this we will get

$$f_0 \equiv f|_{x_1 \leftarrow 0} = \sum_b \left(\tilde{f}(0b) + \frac{\mu_1}{\sigma_1}\tilde{f}(1b)\right)\phi_b(y),$$

and

$$f_1 \equiv f|_{x_1 \leftarrow 1} = \sum_b \left(\tilde{f}(0b) - \frac{(1-\mu_1)}{\sigma_1}\tilde{f}(1b)\right)\phi_b(y).$$

This implies

$$\tilde{f}_0(b) = \tilde{f}(0b) + \frac{\mu_1}{\sigma_1}\tilde{f}(1b), \qquad \text{and} \qquad \tilde{f}_1(b) = \tilde{f}(0b) - \frac{(1-\mu_1)}{\sigma_1}\tilde{f}(1b).$$

So continuing with $I_{D,1}(f)$.

$$\begin{aligned} I_{D,1}(f) &= \frac{1}{4}\sum_b (\tilde{f}_0(b) - \tilde{f}_1(b))^2 \\ &= \frac{1}{4}\sum_b \left(\tilde{f}(0b) + \frac{\mu_1}{\sigma_1}\tilde{f}(1b) - \tilde{f}(0b) + \frac{(1-\mu_1)}{\sigma_1}\tilde{f}(1b)\right)^2 \\ &= \frac{1}{4}\sum_b \left(\frac{\mu_1 + (1-\mu_1)}{\sigma_1}\right)^2 \tilde{f}^2(1b) = \sum_b \left(\frac{\tilde{f}(1b)}{2\sigma_1}\right)^2. \end{aligned}$$

□

The following definition extends the definition of Hamming weight of a Boolean vector to the case of product distributions.

Definition 4.2. (*Hamming weight*). Let D be a product distribution. We define the weight of $a \in \{0, 1\}^n$ under D to be

$$\|a\|_D = \log \prod_{a_i=1} \frac{1}{\sigma_i}.$$

Note that $\|a\|_D \geq |a|$ for any product distribution, and equality is attained precisely when D is the uniform distribution. When the context is clear we will drop the subscript D from $\|a\|_D$ and just write $\|a\|$.

THEOREM 4.3. (MAIN THEOREM). *For any product distribution D, for any Boolean function f, for all positive integer k,*

$$\sum_{\|a\|\ge k} \tilde{f}^2(a) \le \frac{2}{k} I_D(f) \sqrt{\sum_{i=1}^{n} \left(\sigma_i \, log \, \frac{1}{\sigma_i}\right)^2} \le 1.062 \, \frac{\sqrt{n}}{k} I_D(f).$$

PROOF. Note that from Lemma 4.1

$$\begin{aligned}\sum_{i=1}^{n} 4\sigma_i^2 I_{D,i}(f) \log \sigma_i^{-1} &= \sum_{i=1}^{n} \sum_{a:a_i=1} \log \sigma_i^{-1} \tilde{f}^2(a) \\ &= \sum_{a\in\{0,1\}^n} \tilde{f}^2(a) \sum_{i:a_i=1} \log \sigma_i^{-1} \\ &= \sum_{a\in\{0,1\}^n} \|a\| \tilde{f}^2(a).\end{aligned}$$

Recall that the Cauchy–Schwartz inequality is

$$\left(\sum_{i=1}^{n} a_i b_i\right)^2 \le \left(\sum_{i=1}^{n} a_i^2\right)\left(\sum_{i=1}^{n} b_i^2\right).$$

Now we let $a_i = 2\sigma_i I_{D,i}(f)$ and $b_i = 2\sigma_i \log \sigma_i^{-1}$ to get

$$\begin{aligned}I_D(f)^2 &= \sum_{i=1}^{n} 4\sigma_i^2 I_{D,i}(f)^2 \\ &\ge \frac{(\sum_{i=1}^{n} 4\sigma_i^2 I_{D,i}(f) \log \sigma_i^{-1})^2}{4 \sum_{i=1}^{n} (\sigma_i \log \sigma_i^{-1})^2}, \quad \text{by Cauchy–Schwartz} \\ &= \frac{1}{4 \sum_{i=1}^{n} (\sigma_i \log \sigma_i^{-1})^2} \left(\sum_a \|a\| \tilde{f}^2(a)\right)^2 \\ &\ge \frac{1}{4 \sum_{i=1}^{n} (\sigma_i \log \sigma_i^{-1})^2} \left(k \sum_{\|a\|\ge k} \tilde{f}^2(a)\right)^2\end{aligned}$$

for any positive integer k, which proves the first inequality. The second inequality can be seen using simple calculus since $(x \log x^{-1})^2 \le e^{-2} \log^2 e \le 0.2817$, for all $x \in [0, 1/2]$ (simply note that $2\sqrt{0.2817} = 1.062$). □

An alternative way of viewing and deriving the above result is via the notion of average sensitivity. On occasion, we will drop the subscript D from $s_D(f)$ when the product distribution D is clear from context.

THEOREM 4.4. (ALTERNATIVE MAIN THEOREM). *For any Boolean function f and any product distribution D*

$$\sum_a \left\{ \tilde{f}(a)^2 : \|a\|_D \ge \left(\frac{6}{5}\right) \frac{s_D(f)}{\epsilon} \right\} \le \epsilon.$$

PROOF. We start with the identity

$$\sum_a \|a\|\tilde{f}(a)^2 = \sum_i 4\sigma_i\left(\sigma_i \log \frac{1}{\sigma_i}\right) I_{D,i}(f),$$

and use the fact $\sigma \le 1/2$ and $x \log(1/x) \le 3/5$ (not the best), for $x \in [0, 1]$, to get

$$\sum_a \|a\|\tilde{f}(a)^2 \le \frac{6}{5} s(f)$$

(because $s(f) = \quad {}_i I_{D,i}(f)$). But the left-hand side is bounded from below by k ${}_{a:\|a\|\ge k} \tilde{f}(a)^2$ for any k. So in particular choose $k = (6/5)s(f)/\epsilon$. □

The following lemma states a key link between the Fourier spectrum and the influences in monotone Boolean functions.

LEMMA 4.5. *For any monotone Boolean function f, for any product distribution D, and for any $i \in [n]$*

$$\tilde{f}(e_i) = -2\sigma_i I_{D,i}(f).$$

PROOF. Let D_i be the induced distribution over all the variables except x_i. We have the following derivation.

$$\begin{aligned}
\tilde{f}(e_i) &= E_D[f\phi_{e_i}] \\
&= E_{D_i}E_{x_i}[fz_i] \\
&= E_{D_i}\left[(1-\mu_i)\frac{\mu_i}{\sigma_i}f_0 + \mu_i\frac{\mu_i - 1}{\sigma_i}f_1\right] \\
&= E_{D_i}\left[\frac{\mu_i(1-\mu_i)}{\sigma_i}(f_0 - f_1)\right] \\
&= \sigma_i E_{D_i}[f_0 - f_1].
\end{aligned}$$

Now recall that for monotone Boolean functions $I_{D,i}(f) = 1/2E_{D_i}[f_1 - f_0]$. □

5. *Learning Monotone Boolean Functions*

The main results of this section are a subexponential time PAC learning algorithm for any monotone Boolean function and some nearly matching lower bounds on the error rate and some other parameters. But first we review the connections between Fourier transform and PAC learning that were first given in Lineal et al. [1993] and Blum et al. [1994] (see also Mansour's excellent survey [Mansour 1994]).

Fact 1 (*PAC Learning and Fourier Spectrum*). Let D be a product distribution, let $f \in \{-1, +1\}$ be a Boolean function, and let $A \subset \{0, 1\}^n$ be a set of assignments. The real-valued function $g(x) = \quad {}_{a\in A} \tilde{f}(a)\phi_a(x)$ satisfies

$$\Pr_D[f \neq sgn(g)] \leq E_D[(f-g)^2] = \sum_{a \notin A} \tilde{f}^2(a),$$

where $sgn(g)(x) = (-1)^{I[g(x)<0]}$ returns the sign of $g(x)$.

Given g, the randomized Boolean function h defined as [Blum et al. 1994]:

$$h(x) = \begin{cases} -1 & \text{with probability } p(x) \\ +1 & \text{with probability } 1 - p(x) \end{cases}$$

where

$$p(x) = \frac{(1-g(x))^2}{2(1+g^2(x))},$$

satisfies a slightly better error bound

$$\Pr_D[f \neq h] \leq \frac{1}{2} E_D[(f-g)^2] = \frac{1}{2} \sum_{a \notin A} \tilde{f}^2(a).$$

Using this fact, Boolean functions can be learned by collecting the Fourier coefficients $\tilde{f}(a)$ for all $a \in A$. This can be done by finding an approximation h_a of $E_D[f\phi_a] = \tilde{f}(a)$ that satisfies

$$|h_a - \tilde{f}(a)| \leq \sqrt{\epsilon/(2|A|)}$$

with confidence $1 - \delta/|A|$, for every $a \in A$. Then we define the hypothesis $h = {}_{a \in A} h_a \phi_a$. This hypothesis will be an approximation of f that satisfies

$$\Pr_D[f(x) \neq h(x)] \leq \sum_{a \notin A} \tilde{f}^2(a) + \frac{\epsilon}{2}$$

with probability at least $1 - \delta$. Now if ${}_{a \notin A} \tilde{f}^2(a) \leq \epsilon/2$, then the hypothesis h will satisfy

$$\Pr_D[f(x) \neq h(x)] \leq \epsilon.$$

To approximate the Fourier coefficients, we use sampling to find $E_D[f\phi_a]$ for all $a \in A$. By the Chernoff–Hoeffding bounds (Theorem 3.1), if $|f\phi_a| = |\phi_a| \leq B$, then we will need a sample of size at least

$$\frac{4B^2|A|}{\epsilon} \ln \frac{|A|}{\delta}. \qquad (1)$$

When A is the set of all assignments of Hamming weight less than or equal to k, the above algorithm is called the *k-lowdegree Fourier algorithm*. In this case the number of coefficients that the algorithm must estimate is

$$|A| = \sum_{i=0}^{k} \binom{n}{i} \le \left(\frac{ne}{k}\right)^{k} = 2^{k\log(ne/k)}. \tag{2}$$

So $\exp(k\ \log(ne/k))$ will dictate the running time of the k-lowdegree Fourier algorithm.

Note that, in the above discussion, an ϵ-approximator h in the *mean squared sense* is also an ϵ-approximator in the *Boolean sense.*

5.1. A SUBEXPONENTIAL TIME ALGORITHM. The following theorems will show that monotone Boolean functions are PAC learnable under product distributions in subexponential time. For ease of analysis we will assume that the learner knows the product distribution, that is, the parameters μ_i are exactly known, for all $i \in [n]$. Later in a separate subsection we will discuss the case when the learner does not know the distribution and show that this only incurs a log n blow up in the exponent of the time complexity. As in most cases, we deal with $k = \omega(\log n)$, the time complexity is still $2^{\tilde{O}(k)}$.

THEOREM 5.1.1. *For any* ϵ, $\delta > 0$, *any monotone Boolean function is PAC learnable under any product distribution with error* ϵ *and confidence* $1 - \delta$ *in time*

$$2^{O(1/\epsilon\sqrt{n}\log(\epsilon\sqrt{n}))}\, log\, \frac{1}{\delta}.$$

PROOF. We will use the k-lowdegree Fourier algorithm with

$$k = 1.062 I_D(f)\, \frac{\sqrt{n}}{(\epsilon/2)}$$

and with the hypothesis set to $h = {}_{\|a\|\le k}\, h_a \phi_a$. By Theorem 4.3, h is an $(\epsilon/2)$-approximation of f. As $\|a\| \ge |a|$, we have that

$$\{a : \|a\| \le k\} \subseteq \{a : |a| \le k\}$$

and hence the k-lowdegree Fourier algorithm only need to collect (estimate) the Fourier coefficients of Hamming weight at most k. From the definitions of $\|a\|$ and ϕ_a, if $\|a\| \le k$ we get that

$$|\phi_a| = \left| \prod_{a_i=1} \frac{\mu_i - x_i}{\sigma_i} \right| = 2^{\|a\|} \left| \prod_{a_i=1} (\mu_i - x_i) \right| \le 2^k,$$

since $|\mu_i - x_i| \le 1$. Now by (1), (2), and the above, the algorithm outputs a hypothesis that is an approximation of f to within error ϵ with sample size and time complexity of

$$2^{O(\sqrt{n} I_D(f)/\epsilon(\ln(\sqrt{n}\epsilon/I_D(f))))} \log \frac{1}{\delta}. \tag{3}$$

By Lemma 4.5, we note that $I_D(f) \le 1$, for any monotone function f, because

$$I_D(f)^2 = \sum_i (2\sigma_i I_{D,i}(f))^2 = \sum_i \tilde{f}(e_i)^2 \leq 1$$

by Parseval's. Thus, we have $I_D(f)\log(1/I_D(f)) \leq 1$, and therefore

$$I_D(f)\log\frac{\epsilon\sqrt{n}}{I_D(f)} = I_D(f)\log\frac{1}{I_D(f)} + I_D(f)\log(\epsilon\sqrt{n}) = O(\log(\epsilon\sqrt{n})).$$

This analysis proves the time complexity stated in the theorem. □

Remark 5.1.2. Note that using the above algorithm with subexponential time, the best achievable error rate is $\epsilon = 1/\sqrt{n}$. In Section 5.3, we show that this is the best possible error rate up to a $O(\log n)$ factor.

We can alternatively derive the above theorem using the average sensitivity instead of the influence norm, that is, apply Theorem 4.4 instead of Theorem 4.3. Note that in this case, $s_D(f) \leq \sqrt{n}$, for any monotone Boolean function f.

5.2. LEARNING WHEN THE PRODUCT DISTRIBUTION IS UNKNOWN. In this section, we address the issue of learning when the parameters of the product distribution, that is, the μ_is, are unknown. First, we argue that we may ignore all μ_is that are less than n^{-2} or greater than $1 - n^{-2}$ as this will add only an additive factor of n^{-1} to the final error. Formally, let $A \subseteq \{0, 1\}^n$ be the set of all *good* assignments x, that is, ones that satisfy $x_i = 0$, for all $\mu_i < n^{-2}$, and $x_i = 1$, for all $\mu_i > 1 - n^{-2}$. Note that the probability of an example x being good is at least $1 - n^{-1}$. Suppose that h is a hypothesis that approximates the target f quite well on examples from A. Then

$$\Pr_D[h(x) \neq f(x)] \leq \Pr[h(x) \neq f(x) | x \text{ good}] + \Pr[x \text{ not good}] \leq \epsilon + n^{-1}.$$

So we may assume that $n^{-2} < \mu_i < 1 - n^{-2}$, for all $i \in [n]$.

We will estimate each μ_i up to an error of $M^{-(\log n + 4)}$ (using Chernoff-Hoeffding bounds this takes only poly($M^{\log n}$) time), where setting $k = O(\sqrt{n}/\epsilon)$,

$$M = 2^k = 2^{O(\sqrt{n}/\epsilon)}.$$

We will consider three quantities

$$h_A = \sum_{a \in S} E_D[f\phi'_a]\phi'_a,$$

$$h_B = \sum_{a \in S} E_{D'}[f\phi'_a]\phi'_a,$$

$$h_C = \sum_{a \in R} E_D[f\phi_a]\phi_a,$$

where D' is the estimated distribution (using the estimated μ_is) and ϕ'_as are the basis functions according to D', and R and S are the sets of assignments for which the algorithm needs to estimate the ϕ_as and ϕ'_as, respectively. Notice that from the proof of Theorem 5.1.1. and as $I_D(f) \leq 1$, for all monotone Boolean function f, we have

$$|\phi'_a| \le M.$$

We wanted to learn h_C but because only an approximation of μ_i can be found, we will try to learn h_B. Now, as the example oracle gives the examples according to the distribution D and not D' we will instead learn h_A. As the learning parameters (the number of coefficients) are done for D' we have

$$E_{D'}[(h_B - f)^2] \le \epsilon.$$

Suppose $\mu_i + \tau_i$ is the estimation for μ_i, where $|\tau_i| < M^{-(\log n + 4)}$. Notice that

$$\begin{aligned}\frac{D'(x)}{D(x)} &= \prod_{x_i=1}\left(1 + \frac{\tau_i}{\mu_i}\right)\prod_{x_i=0}\left(1 - \frac{\tau_i}{1 - \mu_i}\right)\\ &= \left(1 + O\left(\frac{n^2}{M^{\log n + 4}}\right)\right)^n\\ &= 1 + O\left(\frac{1}{M^{\log n + 3}}\right).\end{aligned}$$

We also have the following bound on $D(x)/D'(x)$.

$$\begin{aligned}\frac{D(x)}{D'(x)} &= \prod_{x_i=1}\left(1 - \frac{\tau_i}{\mu_i + \tau_i}\right)\prod_{x_i=0}\left(1 + \frac{\tau_i}{1 - \mu_i - \tau_i}\right)\\ &\le \prod_{x_i=0}\left(1 + \frac{\tau_i}{1 - \mu_i - \tau_i}\right)\\ &= \left(1 + O\left(\frac{2n^2}{M^{\log n + 4}}\right)\right)^n\\ &= 1 + O\left(\frac{1}{M^{\log n + 3}}\right)\end{aligned}$$

because $1 - \mu_i - \tau_i \ge n^{-2} - M^{-(\log n + 4)} \ge n^{-2}/2$, for large enough n.

Now

$$\begin{aligned}E_D[(h_B - f)^2] &= E_{D'}\left[\frac{D}{D'}(h_B - f)^2\right]\\ &\le E_{D'}[(h_B - f)^2]\max\frac{D}{D'}\\ &\le \epsilon\left(1 + O\left(\frac{1}{M^{\log n + 3}}\right)\right)\\ &\le \epsilon + O\left(\frac{\epsilon}{M^{\log n + 3}}\right).\end{aligned}$$

Therefore, h_B is also a good approximation of f with respect to the distribution D. Now we show that h_A is good enough. We have

$$\begin{aligned}|h_A - h_B| &= \left| \sum_{a \in S} E_D[f\phi'_a]\phi'_a - \sum_{a \in S} E_{D'}[f\phi'_a]\phi'_a \right| \\ &= \left| \sum_{a \in S} E_D\left[f\phi'_a\left(1 - \frac{D'}{D}\right)\right]\phi'_a \right| \\ &\le |S|M^2\left| 1 - \min\frac{D'}{D}\right| \\ &\le O\left(\frac{1}{M}\right),\end{aligned}$$

because

$$|S| \le \sum_{i=0}^{k} \binom{n}{i} \le n^k = 2^{k \log n} = M^{\log n}.$$

Therefore,

$$\begin{aligned}E_D[(h_A - f)^2] &\le 2E_D[(h_A - h_B)^2 + (h_B - f)^2] \\ &= 2E_D[(h_A - h_B)^2] + 2E_D[(h_B - f)^2] \\ &\le 2\epsilon + O\left(\frac{1}{M}\right).\end{aligned}$$

This completes the analysis for learning when the product distribution is unknown.

5.3. LOWER BOUNDS. In this section, we give several lower bounds showing that our algorithm is nearly optimal in terms of running time complexity and error rate. The following theorem shows that the error rate achieved in our algorithm is the best possible for a subexponential time algorithm.

THEOREM 5.3.1. *Any learning algorithm which PAC-learns any monotone Boolean function under the uniform distribution and which runs in subexponential time* (*even with time* 2^{cn}, *for any* $c < 1$) *will output an approximation with an error of at least* $\Omega(1/\sqrt{n}\ \log n)$.

PROOF. There are at least $m(n) = 2^{\binom{n/2}{n}} \ge 2^{d2^n/\sqrt{n}}$ monotone Boolean functions over n variables, for some constant $d < 1$ (see Wegener [1987]). Suppose A is the ϵ-approximation algorithm for any monotone Boolean function. If A outputs a hypothesis h then h can ϵ-approximate at most

$$k(n) = \sum_{i \le \epsilon 2^n} \binom{2^n}{i} \le 2^{2^n \epsilon \log(e/\epsilon)}$$

Boolean functions. Assuming A runs in time 2^{cn}, for some constant $c < 1$, then A can output at most $2^{2^{cn}}$ possible hypotheses. Therefore, we must have $2^{2^{cn}}k(n) \geq m(n)$ which implies

$$2^{cn} + \epsilon 2^n \log \frac{e}{\epsilon} \geq \frac{d2^n}{\sqrt{n}}.$$

This implies $\epsilon = \Omega(1/\sqrt{n} \log n)$. □

The next corollary gives a lower bound for the error rate of any learning algorithm that runs in time bounded by $2^{\sqrt{n}}$.

COROLLARY 5.3.2. *Any learning algorithm for monotone Boolean functions under the uniform distribution with a running time bounded by* $2^{\sqrt{n}}$ *cannot achieve an error smaller than*

$$\Omega\left(\frac{1}{n^{1/4} \, log \, n}\right).$$

PROOF. Let $\mathcal{A}$ be an algorithm that runs in time $2^{\sqrt{n}}$ and achieves error $\epsilon(n)$, for any n. We will construct an algorithm $\mathcal{B}$ that learns any monotone Boolean function over n variables in time 2^{cn}, for some $c < 1$, and achieves an error of $2\epsilon((cn)^2)$. Now because by Theorem 5.3.1.

$$2\epsilon((cn)^2) = \Omega\left(\frac{1}{\sqrt{n} \log n}\right)$$

we get the result of the corollary.

The algorithm $\mathcal{B}$ with input $f(x_1, \ldots, x_n)$ will *pretend* that the input is over $(cn)^2$ variables and runs the algorithm $\mathcal{A}$ to learn the function f. The error rate achievable by this algorithm is $\epsilon = \epsilon((cn)^2)$. This algorithm outputs a hypothesis h that satisfies

$$E_{x_1, \ldots, x_{(cn)^2}}[f \neq h] < \epsilon.$$

This does not imply that h is a good approximation to f under the original domain $\{x_1, \ldots, x_n\}$. We now proceed by randomly and uniformly choosing values $x^0_{n+1}, \ldots, x^0_{(cn)^2}$ and return the hypothesis

$$h\left(x_1, \ldots, x_n, x^0_{n+1}, \ldots, x^0_{(cn)^2}\right).$$

As the expectation is over the uniform distribution, we have

$$E_{x_1, \ldots, x_{(cn)^2}}[f \neq h] = E_{x_{n+1}, \ldots, x_{(cn)^2}} E_{x_1, \ldots, x_n}[f \neq h] < \epsilon,$$

and therefore with probability at least 1/2 (by Markov's inequality) a random $x^0_{n+1}, \ldots, x^0_{(cn)^2}$ gives a $2\epsilon = 2\epsilon((cn)^2)$ approximation to f. □

We now investigate the best error rate of the low-degree algorithm. The first theorem shows that there exists a monotone Boolean function such that to approximate this function within error ϵ using its low-degree Fourier coefficients, for $\epsilon = n^{-1/2}$, we need to collect all coefficients of weight less or equal to cn, for some constant $c < 1$. The second theorem shows that to approximate the majority function with the same error (in the mean square sense) we need to collect all of its Fourier coefficients of weight up to $O(\sqrt{n})$.

THEOREM 5.3.3. *For any constant $c < 1$ there is a monotone Boolean function f that satisfies*

$$\sum_{|a|\geq cn} \hat{f}^2(a) \geq \Omega\left(\frac{1}{\sqrt{n}\,\log n}\right).$$

PROOF. Assume for contradiction that there is some constant $c < 1$ such that for any monotone function f

$$\sum_{|a|\geq cn} \hat{f}^2(a) \leq O\left(\frac{1}{\sqrt{n}\,\log n}\right).$$

But this implies that the low-degree algorithm that searches all coefficients of degree at most cn will approximate f within an error of $O(1/(\sqrt{n}\,\log n))$. This contradicts Theorem 5.3.1 modulo constant factors. □

THEOREM 5.3.4. *The majority function f satisfies* ${}_{|a|\geq\Omega(\sqrt{n})}\,\hat{f}^2(a) \geq \Omega(1/\sqrt{n})$.

PROOF. Let $f(x) = MAJ(x) = I[\;{}^{n}_{i=1}\, x_i \geq n/2]$. Because f is a symmetric function, the influence of all variables are equal. From the first equality in the proof of Theorem 4.3 we have

$$\sum_{a} |a|\hat{f}^2(a) = \sum_{i=1}^{n} I_i(f) = nI_1(f).$$

To get a bound on $I_1(MAJ)$, note that $I_1(MAJ) \geq 2^{-n}\binom{n}{n/2} \geq c/\sqrt{n}$, for some constant c. Thus

$$\begin{aligned} c\,\sqrt{n} &\leq \sum_{a} |a|\hat{f}(a)^2 = \sum_{|a|\geq c/2\,\sqrt{n}} |a|\hat{f}(a)^2 + \sum_{|a|<c/2\,\sqrt{n}} |a|\hat{f}(a)^2 \\ &\leq n \sum_{|a|\geq c/2\,\sqrt{n}} \hat{f}(a)^2 + \frac{c}{2}\,\sqrt{n}. \end{aligned}$$

This implies that

$$\sum_{|a|\geq c/2\,\sqrt{n}} \hat{f}(a)^2 \geq \frac{c}{2\,\sqrt{n}}. \qquad \square$$

We now use the Vapnik–Chernovenkis dimension to find lower bounds on the sample size. Kearns [1990] had also observed that the VC dimension can be used to prove negative learning results for monotone Boolean functions. Recall that if C is a class of Boolean functions then C shatters $A \subseteq \{0, 1\}^n$ if for every Boolean function $g: A \to \{0, 1\}$ there exists a Boolean function $f \in C$ such that $f|_A = g$. The *Vapnik–Chernovenkis* dimension of C, $VCdim(C)$, is the cardinality of the largest subset A that is shattered by C. Ehrenfeucht et al. [1988] proved a sample complexity lower bound of

$$\Omega\left(\frac{1}{\epsilon}\ln\frac{1}{\delta}+\frac{1}{\epsilon}\,VCdim(C)\right)$$

for PAC learning any class C with error ϵ and confidence δ. It is easy to see that the VC-dimension of monotone functions is at least $\binom{n}{n/2} \sim 2^n/\sqrt{n}$. Hence, we get the following easy corollary:

COROLLARY 5.3.5. *Any* PAC *learning algorithm for all monotone Boolean functions under an arbitrary distribution with error* ϵ *and confidence* δ (*for sufficiently small* ϵ *and* δ) *requires at least* $\Omega(2^n/(\epsilon\sqrt{n}) + (1/\epsilon)\,ln\,(1/\delta))$ *examples.*

6. *Circuit Approximations of Monotone Boolean Functions*

We study the circuit complexity of approximating monotone Boolean functions using Boolean circuits (nonmonotone). We prove that any monotone Boolean function can be approximated by a nonmonotone Boolean circuit of subexponential size and sublinear depth. This result is a consequence of Theorem 4.3.

THEOREM 6.1. *For any monotone Boolean function f on n variables and for any constant* $\epsilon > 0$, *there is a Boolean circuit of size*

$$2^{O(1/\epsilon\,\sqrt{n}log\ n)}$$

and depth

$$O\left(\frac{1}{\epsilon}\sqrt{n}\ log\ n\right)$$

which approximates f to within ϵ *under the uniform distribution.*

PROOF. Note that the low-degree algorithm outputs a hypothesis

$$h(x) = \sum_{|a|\le O(1/\epsilon\,\sqrt{n})} c_a\chi_a(x),$$

where $c_a \sim E[f\chi_a]$. Note that each c_a can be at most $2^{O(1/\epsilon\sqrt{n}\log n)}$ bits. So essentially we need to add m number each being m bits, where $m(n) =$ *test it and again* $2^{O(1/\epsilon\sqrt{n}\log\ n)}$. This problem is known to be in NC^1 (Boolean functions computable by a bounded fan-in, logarithmic depth, and polynomial size Boolean circuit). □

In the next section, we will investigate the circuit approximation of monotone graph properties. For this, we will need the following theorem:

THEOREM 6.2. *For any product distribution D with* $\mu_i = \mu$, *for all i, any monotone Boolean function f on n variables and any constant* $\epsilon > 0$, *there is a Boolean circuit of size*

$$2^{O(\sigma/\epsilon\,\sqrt{n}log(\epsilon\,\sqrt{n}/\sigma))}$$

and depth

$$O\left(\frac{1}{\epsilon}\sigma\sqrt{n}\,log(\epsilon\sqrt{n}/\sigma)\right)$$

which approximates f to within ϵ.

PROOF (SKETCH). We have $\|a\| = |a|\log 1/\sigma$, where $\sigma = \sqrt{\mu(1-\mu)}$. By Theorem 4.3,

$$\sum_{|a|\geq k}\tilde{f}(a)^2 \leq \frac{2\sigma}{k} I_D(f)\sqrt{n}.$$

To get an error ϵ, we need $k \geq 2\sigma/\epsilon\ I_D(f)\sqrt{n}$. Recall that for monotone Boolean functions, $I_D(f) \leq 1$. Notice that we can truncate the μ_is and σ_is while only incurring a polynomial blow-up in error. When these are truncated, $\phi_a(x)$ can be computed in polynomial time and therefore there is a polynomial size circuit that computes them. Now we proceed as in Theorem 6.1. □

6.1. CIRCUIT APPROXIMATION OF MONOTONE GRAPH PROPERTIES. We consider some monotone graph properties on the random graph $G(n, p)$, where n is the number of vertices of G and p is the edge existence probability. Some well-known graph properties are monotone: the clique function $CLIQUE_k^n$ which is one if and only if the graph has a clique of size at least k, the hamiltonicity function HAM_n which is one if and only if the graph has a Hamiltonian cycle, the planarity function $PLANAR_n$ which is one if and only if the graph is planar, and so on. We investigate the problem of approximating these monotone graph-theoretic functions.

We adopt the *probabilistic* model of the random graph on n vertices (see Spencer [1994] for other models). The random graph $G = G(n, p)$ is a probability distribution on the edges of the complete graph K_n on n vertices, where each edge exists independently with probability $p \in [0, 1]$. A Boolean function f on the edge set $E(G)$ is called a *monotone graph property* if f is a monotone (or antimonotone) Boolean function over $E(G)$. Any monotone graph property exhibits a *threshold phenomena* (see Bollobás [1995] and Spencer [1994]). Let f be a monotone graph property on $G(n, p)$. A function $r(n)$ is called a *threshold function* for f if it satisfies

(1) if $\lim_{n\to\infty} p(n)/r(n) = 0$, then $\lim_{n\to\infty} \Pr[f(G(n, p)) = 1] = 0$.
(2) if $\lim_{n\to\infty} r(n)/p(n) = 0$, then $\lim_{n\to\infty} \Pr[f(G(n, p)) = 1] = 1$.

Definition 6.1.1. (*ϵ-$G(n, p)$ circuit*). For a graph property A, for fixed $\epsilon > 0$ and $p \in [0, 1]$, and random inputs drawn from $G(n, p)$, we call a Boolean circuit an ϵ-$G(n, p)$ circuit if it outputs a correct answer to the property A with probability at least $1 - \epsilon$. Here the probability is with respect to the distribution $G(n, p)$ on the input graphs.

THEOREM 6.1.2. *Let f be a monotone graph property with a threshold function* $r(n)$ (*the number of inputs to f is* $m = \binom{n}{2}$). *Then, for any fixed* $\epsilon > 0$, $p \in [0, 1]$, *there is an* ϵ-$G(n, p)$ *circuit for f of size*

$$2^{O((\sqrt{nr(n)}/\epsilon)log(\epsilon\sqrt{n}/\sqrt{r(n)}))}.$$

PROOF. Fix ϵ, $p > 0$. By (1) and (2), there exists a constant $c > 1$ such that for $p \geq cr(n)$ we have $\Pr[f(G(n, p)) = 1] \geq 1 - \epsilon$ and for $p \leq r(n)/c$ we have $\Pr[f(G(n, p)) = 0] \geq 1 - \epsilon$. Therefore, for $p \geq cr(n)$, we build the constant circuit 1 and for $p \leq r(n)/c$ we build the constant circuit 0. For p in the range $(r(n)/c, cr(n))$, we build the circuit in Theorem 6.2 with $\mu = p$ and error ϵ/c. This circuit has the required size and depth. □

Next we consider two monotone graph properties that are NP-complete: $CLIQUE_k^n$ and HAM_n. The clique function has a threshold of $r(n) = n^{-2/(k-1)}$ and the Hamiltonicity function has a threshold of $r(n) = \ln n/n$. Our approach failed to give an interesting bound for the clique function. For the Hamiltonicity function, we have the following:

COROLLARY 6.1.3. *For any ϵ and a fixed p, there is a Boolean circuit that approximates HAM_n to within error ϵ and has size*

$$2^{O(1/\epsilon \sqrt{n} \log^{1.5} n)}$$

and depth

$$O\left(\frac{1}{\epsilon} \sqrt{n} \log^{1.5} n\right).$$

PROOF. Apply Theorem 6.1.2. with $r(n) = \ln n/n$. □

Remark 6.1.4. The above results can be contrasted with several works on solving the search versions of NP-hard problems on random graphs. For example, Bollobás et al. [1987] described an expected polynomial time algorithm for finding Hamiltonian cycles in the random graph $G(n, m)$ with n vertices and

$$m = n \log n/2 + n \log\log n/2 + c_n n$$

edges, where c_n is some sequence of integers. Their model of the random graph $G(n, m)$ is a uniform distribution on undirected graphs with n vertices and m edges. They proved that the success probability of their algorithm equals to that of the existence of such Hamiltonian cycles. Their result differs from ours in several ways. First, our result only concerns the approximation of the decision problem, and second, we don't need to impose any restriction on the number of edges in our random graph (also, our random graph model is slightly different from theirs, that is, $G(n, p)$ vs. $G(n, m)$), but third, their algorithm runs in expected polynomial time whereas our algorithm runs in subexponential time.

7. *Approximating over a Convex Mixture of Distributions*

The result in this section states that approximability over a collection of distributions $\{D_i\}$ implies the approximability over any distribution in the convex space of $\{D_i\}$. First, we need to introduce a notion of *convex dimension* of a probability distribution.

Definition 7.1. (*Convex Dimension of a Distribution*). Let I be some index set. Let $\mathcal{D} = \{D_i\}_{i \in I}$ be the set of all product distributions over $\{0, 1\}^n$. Consider a distribution D of the form

$$D = \sum_{i \in I} \lambda_i D_i,$$

where $\lambda_i \in [0, 1]$, for each $i \in I$, and $_{i \in I} \lambda_i = 1$. We call D the convex linear combination of distributions $\{ D_i : i \in I\}$. The convex dimension cdim(D) of D is the least m such that D can be represented as a convex linear combination of m product distributions.

LEMMA 7.2. *Let $\tau_1, \dots, \tau_m \in \{-1, 1\}$ and let $d_1, \dots, d_m$ be positive real numbers. If* ${}^{m}_{i=1} d_i \tau_i < 0$, *then*

$$\sum_{\tau_i = -1} d_i \geq \frac{1}{2} \sum_{i=1}^{m} d_i.$$

PROOF. If ${}^{m}_{i=1} d_i \tau_i < 0$, then $_{i:\tau_i = 1} d_i <$ $_{i:\tau_i = -1} d_i$. Thus, ${}^{m}_{i=1} d_i < 2$ $_{i:\tau_i = -1} d_i$. □

THEOREM 7.3. *Let f be a Boolean function that can be approximated over each D_i from the set $\{D_i\}_{i=1}^m$ of distributions. Then f can be approximated over any distribution D that is a convex linear combination of the D_is, say $D =$ ${}^{m}_{i=1} \lambda_i D_i$, provided that the λ_is are known and each distribution D_i is known and is polynomial time computable.*

PROOF. Suppose ${}^{m}_{i=1} \lambda_i = 1$ and $D =$ ${}^{m}_{i=1} \lambda_i D_i$. Let h_i be a hypothesis over distribution D_i satisfying $E_{D_i}[h_i \neq f] \leq \epsilon/2$. Define the hypothesis H over D to be

$$H(x) = sgn\left(\sum_{i=1}^{m} \lambda_i D_i(x) h_i(x) \right).$$

Then,

$$E_D[H \neq f] = E_D[\lambda_1 D_1 h_1 f + \cdots + \lambda_m D_m h_m f < 0].$$

Using Lemma 7.2, if $\lambda_1 D_1 h_1 f + \cdots + \lambda_m D_m h_m f < 0$ for some x, then

$$\sum_{i: h_i(x) f(x) = -1} \lambda_i D_i \geq \frac{\lambda_1 D_1 + \cdots + \lambda_m D_m}{2}.$$

Therefore, we have the following.

$$\begin{aligned} E_D[H \neq f] &= \sum_{x_0 : H(x_0) \neq f(x_0)} \sum_{i} \lambda_i D_i(x_0) \\ &\leq 2 \sum_{x_0 : H(x_0) \neq f(x_0)} \sum_{i : h_i(x_0) \neq f(x_0)} \lambda_i D_i(x_0), \text{ Lemma 7.2} \\ &\leq 2 \sum_{x_0} \sum_{i : h_i(x_0) \neq f(x_0)} \lambda_i D_i(x_0) \\ &= 2 \sum_{i} \lambda_i E_{D_i}[h_i \neq f] \leq \epsilon. \end{aligned}$$

□

8. *Learning in Polynomial Time*

In this section, we describe some polynomial time learning results on some subclasses of monotone Boolean functions.

8.1. WEAK LEARNING. Kearns and Valiant [1989] proved that monotone Boolean functions are weakly learnable under the uniform distribution with error $1/2 - 1/(2n)$, In the following, we improve their result slightly. In particular, there is a weak learner with error $1/2 - \Omega(\log^2 n/n)$ under the uniform distribution and there are weak learners with error $1/2 - c/n$, for any constant c, under any product distribution.

For learning under the uniform distribution, we will use the following result of Kahn et al. [1988] on the lower bound of the sum of the squares of the influences of variables.

LEMMA 8.1.1. [KAHN ET AL. 1988]. *Let $f \in \{0, 1\}$ be a Boolean function with $p = Pr[f(x) = 1] \leq 1/2$. Then*

$$\sum_{i=1}^{n} I_i(f)^2 \geq \frac{p^2}{5} \frac{(\log n)^2}{n}.$$

THEOREM 8.1.2. *There is a polynomial time weak PAC learning algorithm with error $\epsilon = 1/2 - \Omega(log^2 n/n)$ for any monotone Boolean function under the uniform distribution.*

PROOF. We will assume that without loss of generality that $p = \Pr[f(x) = 1] \leq 1/2$, as we can take $\neg f(\neg x_1, \ldots, \neg x_n)$. This transformation does not affect the influences.

If $p < 1/4$ we already have a weak learning using the all-zero hypothesis, otherwise, as $I_i(f)^2 = \hat{f}^2(e_i)$ and using Lemma 8.1.1, ${}_{i=1}^{n} \hat{f}^2(e_i) \geq p^2 \log^2 n/(5n) \geq \log^2 n/80n$. Combining this with Fact 1 with $A = \{e_i : i \in [n]\}$, we get a weak learner with the claimed accuracy. □

THEOREM 8.1.3. *For any constant k there is a polynomial time weak* PAC *learning algorithm with error $\epsilon = 1/2 - k/n$ for monotone Boolean functions under any product distribution.*

PROOF. By Fact 1, if ${}_{\|a\| \geq k} \tilde{f}^2(a) \leq 1/2$, then we get a 1/4-approximator by the standard low-degree algorithm (using again the fact that $\|a\| \geq |a|$). Otherwise, by Theorem 4.3, we have $1.062 \sqrt{n}/k \; I_D(f) \geq 1/2$ and hence ${}_i$ $\tilde{f}^2(e_i) \geq k^2/4(1.062)^2 n$. □

8.2. STRONG LEARNING. Sakai and Maruoka [1994] proved that monotone $O(\log n)$-term DNF is PAC learnable under the uniform distribution. We improve their result in two ways. First, we extend the class to a larger subclass of the monotone Boolean functions and second we extend the distribution to constant-bounded product distributions.

A variable x_i is *relevant* for f if there are $a, b \in \{0, 1\}^n$ with $a = b \oplus e_i$ and $f(a) \neq f(b)$. Note that x_i is relevant if and only if $I_i(f) > 0$.

Definition 8.2.1. (*The concept class $A(k)$*). Let $A(k)$ be the class of Boolean functions of the form $f(T_1, \ldots, T_k)$, where f is an arbitrary monotone Boolean function on k inputs and each T_i is a monotone conjunction or a disjunction over n variables.

THEOREM 8.2.2. *The class $A(\log n)$ is PAC learnable under constant bounded product distributions*.

PROOF. From Lemma 4.1, the influence of a variable is

$$I_{D,i}(f) = \frac{1}{4\sigma_i^2} \sum_{a:a_i=1} \tilde{f}(a)^2.$$

As the product distribution is constant-bounded, σ_i is a constant and therefore if the influence is small, then we may assume that x_i is not relevant, as this incurrs only a small additional error to the hypothesis. Now the divide-conquer learning algorithm presented in Bshouty [1995] can be used in the same way. □

THEOREM 8.2.3. *For any constant ϵ, the class of monotone functions that depend on $O(\log^2 n/\log^2 \log n)$ variables is PAC learnable with error ϵ under constant bounded product distributions.*

PROOF. By the assumption, there are at most $m(n) = \log^2 n/\log^2 \log n$ variables having nonzero influence. Note that we may ignore all the variables with very small influence as we are dealing with constant bounded product distributions. So we can apply the low-degree algorithm which will run in time $2^{\sqrt{m(n)}\log m(n)} = n^{O(1)}$. □

THEOREM 8.2.4. *For any constant ϵ, the class $A(\log^2 n/\log^3 \log n)$ is PAC learnable with error ϵ under constant bounded product distributions.*

PROOF. First, we claim that any term with size $\Omega(\log \log n)$ may be ignored without incurring an error of more than $O(1)$. Now observe that with this simplification, there are at most $\log^2 n/(\log \log n)^2$ variables. This problem reduces to Theorem 8.2.3. □

Remark 8.2.5. The learning algorithms discussed so far fit into the *statistical query* learning model introduced by Kearns [1993]. Hence by Kearns' results, these algorithms are robust against *classification noise* in the example oracle.

ACKNOWLEDGMENTS. This paper would not have existed without Jeff Jackson's telling us about all the *neat* things related to Fourier transform. His enthusiastic lectures during his visit to the Department of Computer Science, University of Calgary had more impact on this work than anything else. We wish to thank him for his generosity and helpful contributions during our work on this article. We also thank Dan Boneh, for suggesting the notion of influence norm, Yishay Mansour, for suggesting the average sensitivity viewpoint, and Uriel Feige, for pointing out to us some references on Hamiltonian cycles in random graphs.

We also would like to thank the referee for very helpful and insightful comments that greatly improve the presentation of our work and for catching some omissions and errors in an earlier draft of this article.

REFERENCES

AIELLO, W., AND MIHAIL, M. 1991. Learning the Fourier spectrum of probabilistic lists and trees. In *Proceedings of the 2nd Annual ACM–SIAM Symposium on Discrete Algorithms* (San Francisco, Calif., Jan. 28–30). ACM, New York, pp. 291–299.

ALON, N., AND SPENCER, J. 1992. *The Probabilistic Method*. Wiley-Interscience, New York.

BOLLOBÁS, B. 1985. *Random Graphs*. Academic Press, Inc., Orlando, Fla.

BELLARE, M. 1992. A technique for upper bounding the spectral norm with applications to learning. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. ACM, New York, pp. 62–70.

BSHOUTY, N. 1995. Simple learning algorithms using divide and conquer. In *Proceedings of the 8th Annual ACM Conference on Computational Learning Theory* (Santa Cruz, Calif., July 5–8). ACM, New York, pp. 447–453.

BOLLOBÁS, B., FENNER, T., AND FRIEZE, A. 1987. An algorithm for finding Hamiltonian paths and cycles in random graphs. *Combinatorica 7*.

BLUM, A., FURST, M., JACKSON, J., KEARNS, M., MANSOUR, Y., AND RUDICH, S. 1994. Weakly learning DNF and characterizing statistical query using Fourier analysis. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing* (Montréal, Que., Canada, May 23–25). ACM, New York, pp. 253–262.

BRUCK, J., AND SMOLENSKY, R. 1992. Polynomial threshold functions, AC^0 functions, and spectral norms. *SIAM J. Comput. 21*, 1, 33–42.

COPPERSMITH, D., KRAWCZYK, H., AND MANSOUR, Y. 1993. The shrinking generator. In *Advances in Cryptology CRYPTO.*

EHRENFEUCHT, A., HAUSSLER, D., KEARNS, M., AND VALIANT, L. 1988. A general lower bound on the number of examples needed for learning. In *Proceedings of the 1988 Workshop on Computational Learning Theory*. pp. 139–154.

FURST, M., JACKSON, J., AND SMITH, S. 1991. Improved learning of AC^0 functions. In *Proceedings of the 4th Annual Workshop on Computational Learning Theory*. pp. 317–325.

HANCOCK, T., AND MANSOUR, Y. 1991. Learning monotone k-μ DNF formulas on product distributions. In *Proceedings of the 4th Annual Workshop on Computational Learning Theory*. pp. 179–183.

HAGERUP, T., AND RUB, C. 1989. A guided tour to Chernoff bounds. *Inf. Proc. Lett. 33*, 305–308.

JACKSON, J. 1994. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. IEEE, New York, pp. 42–53.

KEARNS, M. 1990. *The Computational Complexity of Machine Learning*. MIT Press, Cambridge, Mass.

KEARNS, M. 1993. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing* (San Diego, Calif., May 16–18). ACM, New York, pp. 392–401.

KAHN, J., KALAI, G., AND LINIAL, N. 1988. The influence of variables on Boolean functions. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*. IEEE, New York, pp. 68–80.

KUSHILEVITZ, E., AND MANSOUR, Y. 1993. Learning decision trees using the fourier spectrum. *SIAM J. Comput. 22*, 6, 1331–1348.

KEARNS, M., AND VALIANT, L. 1989. Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing* (Seattle, Wash., May 15–17, 1989) ACM, New York, pp. 433–444.

LINIAL, N., MANSOUR, Y., AND NISAN, N. 1993. Constant depth circuits, Fourier transform and learnability. *J. ACM 40*, 3, (July) 607–620. IEEE, New York.

MANSOUR, Y. 1992. An $O(n^{\log \log n})$ learning algorithm for DNF. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. ACM, New York, pp. 53–61.

MANSOUR, Y. 1994. Learning Boolean functions via the Fourier transform. Tutorial notes for the *Workshop on Computational Learning Theory.*

MANSOUR, Y. 1995. Randomized interpolation and approximation of sparse polynomials. *SIAM J. Comput. 24*, 2, 357–368.

SPENCER, J. 1994. *Ten Lectures on the Probabilistic Method. SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*, 2nd ed.

SAKAI, Y., AND MARUOKA, A. 1994. Learning monotone log-term DNF formulas. In *Proceedings of the 7th Annual ACM Conference on Computational Learning Theory* (New Brunswick, N.J., July 12–15). ACM, New York, pp. 165–172.

VALIANT, L. G. 1984. A theory of the learnable. *Commun. ACM*, *27*, 11 (Nov.), 1134–1142.

WEGENER, I. 1987. *The Complexity of Boolean Functions*. Wiley-Teubner.

RECEIVED MAY 1995; REVISED FEBRUARY 1996; ACCEPTED MARCH 1996