Learning Monotone Decision Trees in Polynomial Time

Ryan O'Donnell*
Theory Group
Microsoft Research
Redmond, WA
odonnell@microsoft.com

Rocco A. Servedio†
Department of Computer Science
Columbia University
New York, NY
rocco@cs.columbia.edu

December 3, 2005

Abstract

We give an algorithm that learns any monotone Boolean function $f: \{-1,1\}^n \to \{-1,1\}$ to any constant accuracy, under the uniform distribution, in time polynomial in n and in the decision tree size of f. This is the first algorithm that can learn arbitrary monotone Boolean functions to high accuracy, using random examples only, in time polynomial in a reasonable measure of the complexity of f. A key ingredient of the result is a new bound showing that the average sensitivity of any monotone function computed by a decision tree of size s must be at most $\sqrt{\log s}$. This bound has already proved to be of independent utility in the study of decision tree complexity [27].

We generalize the basic inequality and learning result described above in various ways; specifically, to partition size (a stronger complexity measure than decision tree size), p-biased measures over the Boolean cube (rather than just the uniform distribution), and real-valued (rather than just Boolean-valued) functions.

*Some of this work was done while at the Institute for Advanced Study, supported by National Science Foundation under agreement No. CCR-0324906. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

†Supported in part by NSF CAREER award CCF-0347282 and a Sloan Foundation Fellowship.

1 Introduction

1.1 Computationally efficient learning from random examples.

In the two decades since Valiant introduced the Probably Approximately Correct (PAC) learning model [32], a major goal in computational learning theory has been the design of computationally efficient algorithms for learning Boolean functions from random examples. The original distribution-free PAC learning model of Valiant required that for *any* distribution $\mathcal{D}$ over the domain of examples (which throughout this paper is $\{-1,1\}^n$), the learning algorithm must with high probability succeed in generating a hypothesis for the unknown target function which is highly accurate relative to $\mathcal{D}$. Despite much effort over a twenty year span, very few efficient learning algorithms have been obtained in this demanding model. Thus the focus of much work has shifted to the natural *uniform distribution* PAC learning model, in which the examples used for learning are uniformly distributed over $\{-1,1\}^n$ (we give a precise definition of this learning model in Section 2).

An easy information-theoretic argument shows that no poly(n)-time algorithm can learn arbitrary Boolean functions $f : \{-1,1\}^n \to \{-1,1\}$ to accuracy nonnegligibly better than 1/2. Consequently, the most ambitious conceivable goal in uniform distribution learning is to obtain an algorithm that can learn any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ in time polynomial in n and in a reasonable measure of the "size" or complexity of f. Different complexity measures for Boolean functions thus give rise to different notions of efficient learnability; for example, one might hope for an algorithm that can learn any Boolean function f in time polynomial in n and $DT(f)$, the number of leaves in the smallest Boolean decision tree that computes f (this is the well-studied — and notoriously difficult — problem of "learning decision trees under the uniform distribution"). A more ambitious goal would be to learn in time polynomial in $DNF(f)$, the number of terms in the smallest disjunctive normal form formula for f, or $AC^0_d(f)$, the size of the smallest depth-d AND/OR/NOT circuit for f.

Unfortunately, learning arbitrary Boolean functions in polynomial time in this sense has proved to be intractably difficult for all "reasonable" size measures. For the strongest reasonable size measure (Boolean circuit size), Valiant already observed in [32] that the existence of cryptographic pseudorandom functions [11] implies the nonexistence of uniform distribution algorithms that can learn any function f in time polynomial in the Boolean circuit size of f. This negative result was strengthened by Kharitonov [20], who showed that (under a strong but plausible assumption on the hardness of integer factorization) no uniform distribution algorithm can learn every f in time polynomial in $AC^0_d(f)$ for some fixed constant d. In fact, despite intensive research, no algorithm is currently known that learns arbitrary Boolean functions in time polynomial in *any* reasonable size measure; such an algorithm would constitute a tremendous breakthrough in computational learning theory, see e.g. [1]. (We stress that simple arguments such as those in [5] show that there is no *information-theoretic* impediment to learning from a polynomial number of examples; the apparent difficulty is in designing a *polynomial-time* algorithm.)

1.2 Background: learning monotone functions.

Confronted with the difficulties described above, researchers have tried to learn various restricted classes of Boolean functions. The most natural and intensively studied such class is the class of all *monotone* functions $f : \{-1,1\}^n \to \{-1,1\}$, i.e. functions that satisfy $f(x) \geq f(y)$ whenever $x \geq y$ in the partial order on $\{-1,1\}^n$.

Many partial results on learning restricted subclasses of monotone functions under the uniform distribution have been obtained. Sakai and Maruoka [28] gave a poly(n)-time algorithm that can learn any monotone size-$O(\log n)$ DNF under uniform; this result was subsequently generalized by

Bshouty [6] to a somewhat broader class than $O(\log n)$-term DNF. The main result of Bshouty and Tamon in [7] is a proof that any monotone function can be learned to accuracy ϵ in $2^{\tilde{O}(\sqrt{n}/\epsilon)}$ time; they used this result to obtain a poly(n)-time algorithm (for ϵ constant) that can learn a class of functions that includes monotone $O(\log^2 n/(\log\log n)^3)$-term DNF. More recently, Servedio [30] showed that monotone $2^{O(\sqrt{\log n})}$-term DNF can be learned to constant accuracy ϵ in poly(n) time. Other researchers have also studied the problem of learning monotone functions under uniform (see e.g. [18, 3, 34, 12, 21]), but prior to the current work no algorithms were known for learning arbitrary monotone functions in time polynomial in a reasonable size measure.

1.3 The main learning result.

We give the first algorithm that learns any monotone Boolean function f, under the uniform distribution, in time polynomial in a reasonable measure of the size of f. Given a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$, the *partition size* $P(f)$ of f is the minimum size partition of the Boolean cube $\{-1,1\}^n$ into disjoint subcubes such that f is constant on each subcube. Note that this is a strictly stronger measure of complexity than decision tree size; i.e., $P(f) \leq DT(f)$. Our main learning result is the following:

Theorem 1 *There is an algorithm that (with confidence $1-\delta$) can learn any monotone Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ to accuracy ϵ, given uniform random examples $(x, f(x))$, in time poly$(n, P(f)^{1/\epsilon^2}) \cdot \log(1/\delta)$.*

For any constant accuracy $\epsilon = \Theta(1)$, the algorithm runs in time polynomial in the partition size of f, and hence also in the decision tree size of f. We feel that this constitutes significant progress towards learning monotone functions in time polynomial in their DNF size, an outstanding open problem in computational learning theory (see e.g. the open questions posed in [15], [3] and [2]).

1.4 The approach: bounding average sensitivity of monotone functions.

The main ingredient of our learning algorithm is a new inequality bounding the average sensitivity (sum of influences of all coordinates) of monotone Boolean functions. We give here a simplified version of the theorem (the full result is given in Theorem 3):

Theorem 2 *Every monotone Boolean function f has average sensitivity at most $\sqrt{\log P(f)}$.* [1]

This edge-isoperimetric-type result is of independent interest; indeed, our most general version of it, Theorem 7, recently played a critical role in a new lower bound on the randomized decision tree complexity of monotone graph properties — see [27].

Combining this new inequality with a result of Friedgut [8] that says that Boolean functions with low average sensitivity essentially depend on only a small number of coordinates, we can show that: (i) there is a set of $P(f)^{O(1/\epsilon^2)}$ many Fourier coefficients of f which contain all but ϵ of the "Fourier weight" of f; and (ii) this set of Fourier coefficients can be efficiently identified from uniform random examples only. Applying standard machinery on approximating Boolean functions via their Fourier representations, we obtain Theorem 1.

Our approach seems quite robust. We generalize the basic scenario described above by: (i) considering *real-valued* monotone functions that map $\{-1,1\}^n$ into the continuous interval $[-1,1]$ rather than the discrete range $\{-1,1\}$; and (ii) considering general p-biased product measures over

[1] Here and throughout the paper log denotes logarithm to the base two.

$\{-1,1\}^n$ rather than the uniform distribution. We show that suitable variants of all of our intermediate results holds, and that our main learning result holds exactly as before (i.e., runs in time $P(f)^{O(1/\epsilon^2)}$) in these generalized scenarios.

2 Preliminaries

2.1 Boolean functions and complexity measures.

As is standard in complexity theory and learning theory, we will be interested in complexity measures for Boolean functions f given by the syntactic size of the smallest representation of f under various natural representation schemes. We will chiefly be concerned with partition size and decision tree size, two complexity measures that we now define.

Given a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$, the *decision tree size* of f, denoted $DT(f)$, is the number of leaves in the smallest Boolean decision tree (with variables $x_1,\dots,x_n$ at the internal nodes and bits -1, 1 at the leaves) that computes f. The *partition size* $P(f)$ of f is the minimum number of disjoint subcubes that the Boolean cube $\{-1,1\}^n$ can be partitioned into such that f is constant on each subcube. Since any s-leaf decision tree induces a partition of $\{-1,1\}^n$ into s disjoint subcubes (corresponding to the root-to-leaf paths in the tree), we have that $P(f) \le DT(f)$ for all f. In fact, $P(\cdot)$ is known to be a superpolynomially stronger measure than $DT(\cdot)$ even for monotone functions; Savický [29] has given a monotone Boolean function $g : \{-1,1\}^n \to \{-1,1\}$ which has $P(g) = \mathrm{poly}(n)$ and $DT(g) = 2^{\Omega(\log^{1.26}(n))}$.

2.2 Background: uniform distribution learning.

A *concept class* $\mathcal{F}$ is a collection $\cup_{n\ge1}\mathcal{F}_n$ of Boolean functions where each $f \in \mathcal{F}_n$ is a function from $\{-1,1\}^n$ to $\{-1,1\}$. Throughout this paper we consider the concept class consisting of all monotone Boolean functions.

The uniform distribution Probably Approximately Correct (PAC) learning model has been studied by many authors; see e.g. [4, 7, 13, 14, 20, 22, 24, 28, 30, 33]. In this framework a learning algorithm has access to an *example oracle* $EX(f)$, where $f \in \mathcal{F}_n$ is the unknown *target function* the algorithm is trying to learn. The oracle $EX(f)$ takes no inputs and, when queried, outputs a labelled example $(x, f(x))$, where x is drawn from the uniform distribution $\mathcal{U}$ over $\{-1,1\}^n$.

We say that a Boolean function $h \colon \{-1,1\}^n \to \{-1,1\}$ is an *ϵ-approximator for f* if it satisfies $\Pr_{x\in\mathcal{U}}[h(x) = f(x)] \ge 1-\epsilon$. The goal of a uniform distribution PAC learning algorithm is to generate an ϵ-approximator for the unknown target function f. More precisely, an algorithm A is a *learning algorithm for concept class* $\mathcal{F}$ if the following condition holds: for all $n \ge 1$, all $f \in \mathcal{F}_n$, and all $0 < \epsilon, \delta < 1$, if A is given ϵ and δ as input and has access to $EX(f)$, then with probability at least $1-\delta$ algorithm A outputs an ϵ-approximator for f. We further say that *A PAC learns $\mathcal{F}$ in time t* if A runs for at most t time steps and outputs a hypothesis h which can be evaluated on any point $x \in \{-1,1\}^n$ in time t. Here t will depend on the dimension n and the size s of f under some complexity measure, as well as on ϵ and δ.

2.3 Fourier representation.

Fourier techniques have proven to be a powerful tool for obtaining uniform distribution learning algorithms; see the survey of Mansour [23] for an overview.

Except in Section 5, we will always view $\{-1,1\}^n$ as a probability space under the uniform distribution which we denote by $\mathcal{U}$. Let $f : \{-1,1\}^n \to \mathbf{R}$ be a real-valued function. Recall that the *Fourier expansion* of f is

$$f(x) = \sum_{S\subseteq[n]} \hat{f}(S)\chi_S(x),$$

where $\chi_S(x)$ denotes $\prod_{i\in S} x_i$ and $\hat{f}(S)$ denotes $\mathbf{E}_{x\in\mathcal{U}}[f(x)\chi_S(x)]$. It is well known that every f has a unique Fourier expansion. Pareseval's theorem states that for any $f : \{-1,1\}^n \to \mathbf{R}$ we have $\sum_{S\subseteq[n]} \hat{f}(S)^2 = \mathbf{E}_{x\in\mathcal{U}}[f(x)^2]$, which is clearly 1 if f's range is $\{-1,1\}$.

For Boolean-valued functions $f\colon \{-1,1\}^n \to \{-1,1\}$, the *influence of coordinate i on f* is defined as $\mathrm{Inf}_i(f) = \Pr_{x\in\mathcal{U}}[f(x) \neq f(x^{(\oplus i)})]$, where $x^{(\oplus i)}$ denotes x with the ith bit flipped. In general we have $\mathrm{Inf}_i(f) = \sum_{S\ni i} \hat{f}(S)^2$; it is also well known (see e.g. [17]) that if f is monotone then $\mathrm{Inf}_i(f) = \hat{f}(\{i\})$. For notational ease we will henceforth write $\hat{f}(i)$ in place of $\hat{f}(\{i\})$. The *average sensitivity* of a Boolean function f is $\mathrm{I}(f) = \sum_{i=1}^n \mathrm{Inf}_i(f)$; this is the expected number of sensitive coordinates for a random input $x \in \{-1,1\}^n$. Note that $\mathrm{I}(f) = \sum_{i=1}^n \hat{f}(i)$ for monotone f.

3 The average sensitivity of monotone functions

A well known, folkloric edge-isoperimetric inequality for the Boolean cube states that for any monotone function $f\colon \{-1,1\}^n \to \{-1,1\}$, we have $\mathrm{I}(f) \leq I(\mathsf{Maj}_n) = \Theta(\sqrt{n})$. (This follows from, e.g., the Kruskal-Katona theorem; see [9] for an explicit proof.) This bound $\mathrm{I}(f) \leq O(\sqrt{n})$ is the key to the main result of [7] that any monotone Boolean function can be learned to accuracy ϵ in time $2^{\tilde{O}(\sqrt{n}/\epsilon)}$.

In this section we give a more refined bound on $\mathrm{I}(f)$ that depends on $P(f)$, the partition size of f. Our new bound states that $\mathrm{I}(f) \leq \sqrt{\log P(f)}$ for any monotone f. This yields the usual isoperimetric inequality mentioned as a special case but is much stronger for functions f which have partition size $P(f) = 2^{o(n)}$.

3.1 Subcube partitions.

Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a Boolean function and let $\mathcal{C} = \{C^1,\dots,C^s\}$ be a subcube partition for f, so $C^1,\dots,C^s$ partition $\{-1,1\}^n$ into s subcubes on each of which f is constant. By abuse of notation we will also identify a cube C^t with a length-n vector over $\{-1,0,1\}$ in the obvious way; i.e. the ith coordinate of the string C^t is

$$(C^t)_i = \begin{cases} 1 & \text{if } x_i = 1 \text{ for all } x \in C^t, \\ -1 & \text{if } x_i = -1 \text{ for all } x \in C^t, \\ 0 & \text{otherwise.} \end{cases}$$

Let us also introduce notation for the sets of coordinates which cubes fix:

$$\mathrm{pluses}(C^t) = \{i : (C^t)_i = 1\}, \qquad \mathrm{minuses}(C^t) = \{i : (C^t)_i = -1\},$$

$$\mathrm{fixed}(C^t) = \mathrm{pluses}(C^t) \cup \mathrm{minuses}(C^t).$$

Given an input $x \in \{-1,1\}^n$, we write $C(x)$ to denote the subcube C^t in $\mathcal{C}$ to which x belongs. We also write δ_i to denote $\Pr_{x\in\mathcal{U}}[i \in \mathrm{fixed}(C(x))]$, the probability that the subcube partition "queries" x_i. Note that $\sum_{i=1}^n \delta_i$ equals $\mathbf{E}_{x\in\mathcal{U}}[|\mathrm{fixed}(C(x))|]$, the average number of coordinates $\mathcal{C}$ "queries".

When we draw $x \in \mathcal{U}$, this determines $C(x)$. However, we can equally well view the random determination of $(x, C(x))$ the other way around. Indeed, we will almost always consider choosing a uniformly random string x as follows:

1. Pick a random subcube R from $\mathcal{C}$ by choosing each C^t with probability $2^{-|\text{fixed}(C^t)|}$. In general we will write $R \in \mathcal{C}$ to indicate that R is a random variable given by choosing a subcube from among $C^1, \ldots, C^s$ according to this natural probability distribution on subcubes.

2. Now choose x uniformly at random from the strings in R. We will write $x \in R$ to indicate that x is chosen randomly in this way.

After this procedure, x indeed has the uniform distribution. Furthermore, note that the value $f(x)$ is determined as soon as R is chosen; thus we may abuse notation and write $f(R)$ for this quantity.

We will require the following very easy lemmas:

Lemma 1 *Let R be any subcube and let $i \neq j$ be in $[n]$. Then $\mathbf{E}_{x \in R}[x_i] = R_i$ and $\mathbf{E}_{x \in R}[x_i x_j] = R_i R_j$.*

Proof: Immediate from the definitions. ∎

Lemma 2 *Let $i \neq j$ be in $[n]$. Then $\mathbf{E}_{R \in \mathcal{C}}[R_i] = 0$ and $\mathbf{E}_{R \in \mathcal{C}}[R_i R_j] = 0$.*

Proof: We prove the second statement, with the first being even easier:

$$0 = \mathop{\mathbf{E}}_{x \in \mathcal{U}}[x_i x_j] = \mathop{\mathbf{E}}_{R \in \mathcal{C}} \mathop{\mathbf{E}}_{x \in R}[x_i x_j] = \mathop{\mathbf{E}}_{R \in \mathcal{C}}[R_i R_j],$$

where in the last step we used Lemma 1. ∎

3.2 Proof of the main inequality.

The proof requires one basic lemma:

Lemma 3 *Let $f \colon \{-1,1\}^n \to \{-1,1\}$ be a Boolean function with a subcube partition $\mathcal{C} = \{C^1, \ldots, C^s\}$. Then $\sum_{i=1}^n \hat{f}(i) = \mathbf{E}_{R \in \mathcal{C}}\Big[f(R) \cdot \sum_{i=1}^n R_i\Big]$.*

Proof: We have

$$\sum_{i=1}^n \hat{f}(i) = \sum_{i=1}^n \mathop{\mathbf{E}}_{x \in \mathcal{U}}[f(x) x_i] = \mathop{\mathbf{E}}_{R \in \mathcal{C}} \mathop{\mathbf{E}}_{x \in R}\Big[f(x) \sum_{i=1}^n x_i\Big] = \mathop{\mathbf{E}}_{R \in \mathcal{C}}\left[f(R) \sum_{i=1}^n \mathop{\mathbf{E}}_{x \in R}[x_i]\right] = \mathop{\mathbf{E}}_{R \in \mathcal{C}}\Big[f(R) \sum_{i \in n} R_i\Big],$$

where in the last step we used Lemma 1. ∎

With this lemma in hand we can give the proof that $\mathrm{I}(f) \leq \sqrt{\log P(f)}$ for monotone f:

Theorem 3 *Let $f \colon \{-1,1\}^n \to \{-1,1\}$ be a Boolean function with a subcube partition $\mathcal{C} = \{C^1, \ldots, C^s\}$. Then we have*

$$\sum_{i=1}^n \hat{f}(i) \leq \sqrt{\sum_{i=1}^n \delta_i} \leq \sqrt{\log s},$$

and if f is monotone we may thus write $\mathrm{I}(f) \leq \sqrt{\log s}$.

Proof: Since f is ± 1-valued, from Lemma 3 we have

$$\sum_{i=1}^{n} \hat{f}(i) \leq \mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\Big|\sum_{i=1}^{n} R_i\Big|\Big] \tag{1}$$

with equality iff $f(x) = \mathrm{sgn}(\sum_{i=1}^{n} C(x)_i)$ for all x, i.e. $f(x)$ is the majority of the bits that are set in $C(x)$. Applying Cauchy-Schwarz, we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\Big|\sum_{i=1}^{n} R_j\Big|\Big] \leq \sqrt{\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\Big(\sum_{i=1}^{n} R_i\Big)^2\Big]} &= \sqrt{\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\sum_{i=1}^{n} R_i^2 + 2\sum_{i<j} R_i R_j\Big]} \\
&= \sqrt{\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[|\mathrm{fixed}(R)|\Big]} \qquad (2) \\
&= \sqrt{\mathop{\mathbf{E}}_{x\in\mathcal{U}}\Big[|\mathrm{fixed}(C(x))|\Big]} = \sqrt{\sum_{i=1}^{n} \delta_i},
\end{aligned}$$

where (2) uses Lemma 2.

This proves the first inequality; to finish the proof we must show that $\sum_{i=1}^{n} \delta_i \leq \log s$. We have

$$\sum_{i=1}^{n} \delta_i = \mathop{\mathbf{E}}_{R\in\mathcal{C}}[|\mathrm{fixed}(R)|] = \sum_{t=1}^{s} 2^{-|\mathrm{fixed}(C^t)|} \cdot |\mathrm{fixed}(C^t)| = H(R),$$

where $H(R)$ denotes the binary entropy of the random variable $R \in \mathcal{C}$. Since $\mathcal{C}$, the support of R, is of cardinality s, this entropy is at most $\log s$. ■

Remarks:

1. We note that our proof can easily be used to recover the standard bound $\mathrm{I}(f) \leq \mathrm{I}(\mathsf{Maj}_n) \sim \sqrt{\frac{2}{\pi}}\sqrt{n}$ for arbitrary monotone Boolean functions on n variables. This is because in upper-bounding $\mathbf{E}_{R\in\mathcal{C}}[|\sum_{i=1}^{n} R_i|]$, we may assume without loss of generality that each subcube $C^t \in \mathcal{C}$ fixes exactly n bits. (To see this, suppose that C^t fixes $n' < n$ bits and we subdivide C^t into two subcubes each fixing one more bit. If $\sum_{i=1}^{n}(C^t)_i \neq 0$ then the contribution of C^t to $\mathbf{E}_{R\in\mathcal{C}}[|\sum_{i=1}^{n} R_i|]$ is unchanged by this subdivision, and if $\sum_{i=1}^{n}(C^t)_i = 0$ then the contribution increases.) But now observe that equality occurs in inequality (1), as noted above, if $f(x)$ always equals the majority of the bits set in $C(x)$, i.e. if $f(x) = \mathsf{Maj}_n(x)$ for all x.

2. The bound $\mathrm{I}(f) \leq \sqrt{\log P(f)}$ need not hold for nonmonotone f; an easy example is the parity function on n variables for which $I(f) = \log P(f) = n$.

4 Learning monotone Boolean functions

4.1 Spectral concentration.

In this subsection we show that any monotone Boolean function has all but ϵ of its Fourier spectrum concentrated on a set of $P(f)^{O(1/\epsilon^2)}$ many Fourier coefficients.

In [8] Friedgut showed that any Boolean function with "low" average sensitivity is well approximated by a function that depends only on a "small" number of coordinates. In particular, the proof of Corollary 3.2 in [8] yields the following:

Theorem 4 *There is a universal constant $C < \infty$ such that for all $f\colon \{-1,1\}^n \to \{-1,1\}$ and $\epsilon > 0$, if*

$$t = 2\mathrm{I}(f)/\epsilon, \quad J = \{i : \mathrm{Inf}_i(f) \geq C^{-t}\}, \quad \mathcal{S} = \{S : \ S \subseteq J, \ |S| \leq t\},$$

then $\sum_{S \notin \mathcal{S}} \hat{f}(S)^2 \leq \epsilon$.

Combining this result with Theorem 3, we obtain:

Theorem 5 *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a monotone function, $\epsilon > 0$, and $t = 2\sqrt{\log P(f)}/\epsilon$. Let J and $\mathcal{S}$ be as in Theorem 4. Then $|\mathcal{S}| = P(f)^{O(1/\epsilon^2)}$ and $\sum_{S \notin \mathcal{S}} \hat{f}(S)^2 \leq \epsilon$.*

Proof: The second part of the conclusion follows immediately from combining Theorems 3 and 4. As for bounding $|\mathcal{S}|$, we have $|\mathcal{S}| = \sum_{i=0}^{t} \binom{|J|}{i} \leq O(|J|^t)$. But we also have $|J| \leq \mathrm{I}(f)C^t \leq tC^t$ using Theorem 3, and so $|J|^t \leq 2^{O(t^2)} = P(f)^{O(1/\epsilon^2)}$, as claimed. ∎

4.2 Approximating Boolean functions with spectral concentration.

The following proposition is a straightforward generalization of the "low-degree" algorithm of Linial, Mansour, and Nisan [22].

Proposition 4 *There is an algorithm A with the following property: Let $f : \{-1,1\}^n \to [-1,1]$ and let $\mathcal{S} \subseteq 2^{[n]}$ be a collection of subsets of $[n]$ with the property that $\sum_{S \in \mathcal{S}} \hat{f}(S)^2 \geq 1 - \epsilon$. Then if A is given $\mathcal{S}$, access to $EX(f)$, and parameters $\delta, \theta > 0$, it runs in poly$(n, |\mathcal{S}|, 1/\theta) \cdot \log(1/\delta)$ time and with probability $1 - \delta$ outputs a real-valued function $g : \{-1,1\}^n \to \mathbf{R}$ of the form $g(x) = \sum_{S \in \mathcal{S}} c_S \chi_S(x)$ such that $\mathbf{E}_{x \in \mathcal{U}}[(f(x) - g(x))^2] \leq \epsilon + \theta$.*

Proof sketch: Algorithm A draws a sample of m labelled examples from $EX(f)$ and uses them to empirically estimate each of the Fourier coefficients $\hat{f}(S)$ for $S \in \mathcal{S}$, using the fact that $\hat{f}(S) = E[f(x)\chi_S(x)]$; the coefficients c_S are the empirical estimates thus obtained. A standard analysis (see e.g. Theorem 4.3 of [23]) shows that $m = \mathrm{poly}(|\mathcal{S}|, 1/\theta) \cdot \log(1/\delta)$ suffices to give the proposition. ∎

We remark that if $f : \{-1,1\}^n \to \{-1,1\}$ is Boolean-valued, and $g : \{-1,1\}^n \to \mathbf{R}$ satisfies $\mathbf{E}_{x \in \mathcal{U}}[(f(x) - g(x))^2] \leq \epsilon'$, then defining $h : \{-1,1\}^n \to \{-1,1\}$ by $h(x) = \mathrm{sgn}(g(x))$, it is easily seen that $\Pr_{x \in \mathcal{U}}[h(x) \neq f(x)] \leq \epsilon'$ (see e.g. [22, 23]).

4.3 Learning monotone Boolean functions in polynomial time.

We now give the proof of Theorem 1. Given Theorem 5 and Proposition 4, the idea behind our main learning algorithm is obvious: Given uniform examples from a target function f, identify all coordinates with influence at least $2^{-O(\sqrt{\log P(f)}/\epsilon)}$, and then run the algorithm from Proposition 4 using the set $\mathcal{S}$ from Theorem 5. (We note that a similar algorithm is used by Servedio in [30], though the analysis is completely different.)

By a standard doubling argument, we may assume the partition size $P(f)$ is known to the learner (see Exercise 1.5 of [19]). We now show that the learner can actually identify the sufficiently influential coordinates. This is because f is monotone, and consequently $\mathrm{Inf}_i(f) = \hat{f}(i) = \mathbf{E}_{x \in U}[f(x)x_i]$. Since the learner can empirically estimate this latter quantity to within $\pm\theta$ in time poly$(n, 1/\theta) \cdot \log(1/\delta)$ (with confidence $1-\delta$) by sampling, the learner can determine each influence $\mathrm{Inf}_i(f)$ of f to within an additive $2^{-O(\sqrt{\log P(f)}/\epsilon)}$ in poly$(n, 2^{O(\sqrt{\log P(f)}/\epsilon)})$ time steps, and it's easy to see this is sufficient to maintain correctness and the same time bounds. Complete details can be found in a more general setting in Appendix B.

5 Generalizations: real-valued functions and p-biased measures

In this section we extend our learning result to real-valued functions $f : \{-1,1\}^n \to [-1,1]$ on the p-biased discrete cube. As in the Boolean case, we say a real-valued function f is monotone if $f(x) \geq f(y)$ whenever $x \geq y$. The partition size $P(f)$ of $f : \{-1,1\}^n \to [-1,1]$ is still defined as the minimum number of disjoint subcubes that $\{-1,1\}^n$ can be partitioned into such that f is constant on each subcube.

The p-biased measure on $\{-1,1\}^n$ is the probability distribution assigning probability $p^{|\text{pluses}(x)|}q^{|\text{minuses}(x)|}$ to the input $x \in \{-1,1\}^n$. (Here and throughout q denotes $1-p$). We will write $\{-1,1\}^n_{(p)}$ to indicate that $\{-1,1\}^n$ is endowed with the p-biased measure and write $\Pr_p[\cdot]$ and $\mathbf{E}_p[\cdot]$ to denote probabilities and expectations over $x \in \{-1,1\}^n_{(p)}$.

We use standard notions of PAC learning for functions $f : \{-1,1\}^n_{(p)} \to [-1,1]$. This only involves slightly altering the definitions from Section 2.2. Specifically, examples are now from the p-biased distribution $\{-1,1\}^n_{(p)}$ instead of the uniform distribution[2]; and, the definition of an ϵ-approximator is a function $h : \{-1,1\}^n_{(p)} \to \mathbf{R}$ satisfying $\mathbf{E}_p[(h-f)^2] \leq \epsilon$ (note that we use the "square loss" as is common in learning or approximating real-valued functions). For other work studying PAC learning under the p-biased distribution see e.g. [10, 12, 25, 30].

Our main learning theorem completely extends to the p-biased, real-valued case, as follows:

Theorem 6 *There is an algorithm that (with confidence $1-\delta$) can learn any monotone Boolean function $f : \{-1,1\}^n_{(p)} \to [-1,1]$ to accuracy ϵ, given p-biased random examples $(x, f(x))$, in time poly$(n, P(f)^{1/\epsilon^2}) \cdot \log(1/\delta)$.*

Again, note that for any constant accuracy $\epsilon = \Theta(1)$, the algorithm runs in polynomial time in the partition size of f. Further note that unlike some p-biased PAC learning algorithms such as [10, 30], our algorithm's running time has no dependence on p and thus we have the claimed runtime bound even if p depends on n or $P(f)$, such as $p = 1/\sqrt{n}$.

5.1 Background: Fourier analysis under p-biased measures.

Given two functions $f, g : \{-1,1\}^n_{(p)} \to \mathbf{R}$, the p-biased inner product is defined as $\langle f, g\rangle_p = \mathbf{E}_p[f(x)g(x)]$. For $S \subseteq [n]$ the function $\phi_S(x) : \{-1,1\}^n_{(p)} \to \mathbf{R}$ is defined by

$$\phi_S(x) = \prod_{i \in S} \phi(x_i) \qquad \text{where } \phi(x_i) = \begin{cases} \sqrt{q/p} & \text{if } x_i = 1, \\ -\sqrt{p/q} & \text{if } x_i = -1. \end{cases}$$

The functions $\{\phi_S\}_{S\subseteq[n]}$ form an orthonormal basis with respect to $\langle \cdot, \cdot \rangle_p$. The *$p$-biased Fourier expansion* of $f : \{-1,1\}^n_{(p)} \to \mathbf{R}$ is $f(x) = \sum_{S\subseteq[n]} \tilde{f}(S)\phi_S(x)$ where $\tilde{f}(S) = \mathbf{E}_p[f(x)\phi_S(x)]$; note that we write $\tilde{f}$ rather than $\hat{f}$ to denote p-biased Fourier coefficients. Parseval's identity continues to hold, $\mathbf{E}_p[f^2] = \sum_S \tilde{f}(S)^2$.

We define the operator D_i on functions $f : \{-1,1\}^n_{(p)} \to \mathbf{R}$ by $(D_i f)(x) = \sqrt{pq}(f(x^{(i=1)}) - f(x^{(i=-1)})$, where $x^{(i=b)}$ denotes x with the ith bit set to b. It is not difficult to verify that $(D_i f)(x) = \sum_{S\ni i} \tilde{f}(S)\phi_{S\setminus i}(x)$. We now give the definition of p-biased influence:

[2]There is a question as to whether or not the learning algorithm "knows" the value of p in advance. We show in Appendix B that we may assume without loss of generality that the learning algorithm knows p.

Definition 1 *The* p-biased influence of the ith coordinate on $f : \{-1,1\}^n_{(p)} \to \mathbf{R}$ *is*

$$\mathrm{Inf}_i^{(p)}(f) = \mathbf{E}_p[(D_i f)^2] = \sum_{S \ni i} \tilde{f}(S)^2.$$

Note that if $f : \{-1,1\}^n_{(p)} \to \{-1,1\}$ *then* $\mathrm{Inf}_i^{(p)}(f) = 4pq \Pr_p[f(x) \neq f(x^{\oplus i})]$.

(We remark that this definition differs from the ones in [8, 9] by a multiplicative factor of $4pq$.) We define the *p-biased average sensitivity* to be $\mathrm{I}^{(p)}(f) = \sum_{i=1}^n \mathrm{Inf}_i^{(p)}(f) = \sum_{S \subseteq [n]} |S| \tilde{f}(S)^2$. Note that in the case when $p = 1/2$ and f's range is $\{-1,1\}$, these definitions agree with the standard uniform-distribution definitions from Section 2.3.

We conclude this section with a useful relationship in the p-biased case between influences of monotone real-valued functions and singleton Fourier coefficients:

Fact 5 *For any monotone* $f : \{-1,1\}^n \to [-1,1]$ *we have* $\mathrm{Inf}_i^{(p)}(f) \leq 2\sqrt{pq} \cdot \tilde{f}(i)$, *with equality iff the range of* f *is* $\{-1,1\}$.

Proof: We have $\mathrm{Inf}_i^{(p)}(f) = \mathbf{E}_p[(D_i f)^2]$. Since f is monotone and has range $[-1,1]$ it is easy to see that $0 \leq (D_i f)(x) \leq 2\sqrt{pq}$ for all x. Thus $(D_i f)^2 \leq 2\sqrt{pq} \cdot (D_i f)$ with equality iff f's range is $\{-1,1\}$, and hence $\mathrm{Inf}_i^{(p)}(f) \leq 2\sqrt{pq} \cdot \mathbf{E}_p[D_i f] = 2\sqrt{pq} \cdot \tilde{f}(i)$. ∎

5.2 Bounding influence in monotone real-valued functions under p-biased measures.

In this section we describe our analogue of Theorem 3 for real functions under p-biased measures. We first set up some p-biased preliminaries before proving the theorem. Let $\mathcal{C} = \{C^1, \ldots, C^s\}$ be a subcube partition of $\{-1,1\}^n$. We now identify the C^t's with length-n vectors in a way compatible with the ϕ-basis, i.e.

$$(C^t)_i = \begin{cases} \phi(1) = \sqrt{q/p} & \text{if } x_i = 1 \text{ for all } x \in C^t, \\ \phi(-1) = -\sqrt{p/q} & \text{if } x_i = -1 \text{ for all } x \in C^t, \\ 0 & \text{otherwise.} \end{cases}$$

The definitions of $\mathrm{pluses}(C^t)$, $\mathrm{minuses}(C^t)$ and $\mathrm{fixed}(C^t)$ are as before. We now define $\delta_i^{(p)}$ to be the p-biased version of δ_i:

$$\delta_i^{(p)} = \Pr_{x \in \{-1,1\}^n_{(p)}}[i \in \mathrm{fixed}(C(x))].$$

The observation that choosing $x \in \{-1,1\}^n_{(p)}$ and considering $(x, C(x))$ can be viewed as choosing $R \in \mathcal{C}$ and then $x \in R$ still holds with the obvious p-biased interpretation. Specifically, the random choice $R \in \mathcal{C}$ means selecting the cube C^t with probability $p^{|\mathrm{pluses}(C^t)|} q^{|\mathrm{minuses}(C^t)|}$; then the choice $x \in R$ means picking the unfixed coordinates according to the p-biased distribution.

The analogue of Lemma 2 and the analogue of Lemma 3 (for functions $f : \{-1,1\}^n_{(p)} \to \mathbf{R}$) now hold with no changes in the statements. To prove them we simply repeat their proofs and also the statement and proof of Lemma 1, everywhere replacing x_i and x_j with $\phi(x_i)$ and $\phi(x_j)$. As a consequence, we have the following additional lemma:

Lemma 6 *Given* $\alpha, \beta \in \mathbf{R}$, *the quantity* $\mathbf{E}_{R \in \mathcal{C}}[\alpha \cdot \mathrm{pluses}(R) + \beta \cdot \mathrm{minuses}(R)]$ *depends only on* $p\alpha + q\beta$.

Proof: Sum the first statement of the p-biased analogue of Lemma 2 over all i, and then expand the definition of R_i; one gets:

$$\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\sqrt{q/p}\cdot \mathrm{pluses}(R) - \sqrt{p/q}\cdot \mathrm{minuses}(R)\Big] = 0.$$

$$\Rightarrow \mathop{\mathbf{E}}_{R\in\mathcal{C}}[\mathrm{minuses}(R)] = (q/p)\cdot \mathop{\mathbf{E}}_{R\in\mathcal{C}}[\mathrm{pluses}(R)].$$

So substituting this in we get

$$\begin{aligned}\mathop{\mathbf{E}}_{R\in\mathcal{C}}[\alpha\cdot \mathrm{pluses}(R) + \beta\cdot \mathrm{minuses}(R)] &= \mathop{\mathbf{E}}_{R\in\mathcal{C}}[\alpha\cdot \mathrm{pluses}(R) + (q/p)\beta\cdot \mathrm{pluses}(R)] \\ &= (1/p)\mathop{\mathbf{E}}_{R\in\mathcal{C}}[(p\alpha+q\beta)\cdot \mathrm{pluses}(R)],\end{aligned}$$

completing the proof. ■

With this preparation in hand, we now give our p-biased, real-valued generalization of Theorem 3:

Theorem 7 *Let $f:\{-1,1\}^n_{(p)}\to\mathbf{R}$ be a function with subcube partition $\mathcal{C}=\{C^1,\ldots,C^s\}$. Then we have*

$$\sum_{i=1}^n \tilde{f}(i) \quad\leq\quad \|f\|_2\cdot\sqrt{\sum_{i=1}^n \delta_i^{(p)}} \quad\leq\quad \|f\|_2\cdot\sqrt{\log s}/\sqrt{H(p)},$$

where $H(p)=p\log(1/p)+q\log(1/q)$. If $f:\{-1,1\}^n_{(p)}\to[-1,1]$ is monotone then by Fact 5 we may write $\mathrm{I}^{(p)}(f)\leq\sqrt{4pq/H(p)}\sqrt{\log s}$.

Proof: Applying Cauchy-Schwarz directly to the analogue of Lemma 3, we have

$$\sum_{i=1}^n\tilde{f}(i)\leq\sqrt{\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[f(R)^2\Big]}\cdot\sqrt{\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\Big(\sum_{i=1}^n R_i\Big)^2\Big]}=\|f\|_2\cdot\sqrt{\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\sum_{i=1}^n R_i^2\Big]},$$

where we used the p-biased analogue of the second statement of Lemma 2 in the equality, just as in the proof of Theorem 3. Let us now consider the quantity inside the square root. By definition,

$$\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\sum_{i=1}^n R_i^2\Big] = \mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[(q/p)\cdot\mathrm{pluses}(R)+(p/q)\cdot\mathrm{minuses}(R)\Big]. \tag{3}$$

Now $p(q/p)+q(p/q)=q+p=p\cdot 1+q\cdot 1$, so by Lemma 6,

$$\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[(q/p)\cdot\mathrm{pluses}(R)+(p/q)\cdot\mathrm{minuses}(R)\Big]=\mathop{\mathbf{E}}_{R\in\mathcal{C}}[\mathrm{pluses}(R)+\mathrm{minuses}(R)]=\mathop{\mathbf{E}}_{R\in\mathcal{C}}[|\mathrm{fixed}(R)|]=\sum_{i=1}^n\delta_i^{(p)},$$

completing the proof of the first inequality. As for the second inequality, note that the binary entropy $H(R)$ of the random variable $R\in\mathcal{C}$ is

$$\begin{aligned}H(R)=\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\log(1/\Pr[R])\Big] &= \mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\log(1/p)\cdot\mathrm{pluses}(R)+\log(1/q)\cdot\mathrm{minuses}(R)\Big] \\ &= H(p)\cdot\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\frac{\log(1/p)}{H(p)}\cdot\mathrm{pluses}(R)+\frac{\log(1/q)}{H(p)}\cdot\mathrm{minuses}(R)\Big].\end{aligned}$$

But since $p\frac{\log(1/p)}{H(p)}+q\frac{\log(1/q)}{H(p)}=1$ as well, applying Lemma 6 again yields

$$(3) = H(R)/H(p).$$

But $H(R)\leq\log s$ as observed in the proof of Theorem 3, and the proof is complete. ■

Using the bound $pq\log(1/pq)\leq H(p)$, we have the following corollary:

Corollary 7 *If $f : \{-1,1\}^n_{(p)} \to [-1,1]$ is monotone then* $\mathrm{I}^{(p)}(f) \le 2\sqrt{\log P(f)}/\sqrt{\log(1/pq)}$.

5.3 Spectral concentration under p-biased measures.

We now need to extend Friedgut's result to the p-biased, real-valued case. There are some difficulties involved. In [8], Friedgut gave a p-biased version of Theorem 4; however, he left the quantitative details of the dependence on p unspecified. More seriously, Friedgut's theorem is simply not true for $[-1,1]$-valued functions, even in the $p = 1/2$ case. (See Appendix A for an example demonstrating this.)

However, we are able to circumvent this problem. The necessary insight is the following: A real-valued function with small average sensitivity depends on only a small number of coordinates *if its range is sufficiently "discrete"*. And for the purposes of learning an unknown function to some prescribed accuracy, we don't lose much by "rounding" the function's values to a discrete range.

For $\gamma > 0$, let $\gamma\mathbf{Z}$ denote the set of real numbers of the form γm, where m is an integer. By making some small changes to Friedgut's proof we can derive the following result (the proof is in Appendix A):

Theorem 8 *There is a universal constant $C < \infty$ such that for all $0 < \epsilon, \gamma < 1/2$ and all $f : \{-1,1\}^n_{(p)} \to [-1,1] \cap (\gamma\mathbf{Z})$, if*

$$t = 2\mathrm{I}^{(p)}(f)/\epsilon, \quad \tau = \gamma^C (pq)^{Ct}, \quad J = \{i : \mathrm{Inf}_i^{(p)}(f) \ge \tau\}, \quad \mathcal{S} = \{S :\ S \subseteq J,\ |S| \le t\},$$

then $\sum_{S \notin \mathcal{S}} \tilde{f}(S)^2 \le \epsilon$.

We now combine Theorem 8 with Corollary 7, exactly in the manner of Theorem 5. The $\sqrt{\log(1/pq)}$ saved in Corollary 7 cancels with the pq paid in the τ from Theorem 8, and the factor of $\gamma^{O(1)}$ becomes negligible if we take $\gamma = \epsilon$ (indeed, even $\gamma = 2^{-O(1/\epsilon)}$ would be negligible). We get:

Theorem 9 *Let $\epsilon > 0$, $f : \{-1,1\}^n_{(p)} \to [-1,1] \cap (\epsilon\mathbf{Z})$ be a monotone function, and let $t = 4\sqrt{\log P(f)}/(\epsilon\sqrt{\log(1/pq)})$. Let $J = \{i : \mathrm{Inf}_i(f) \ge (C')^{-t\log(1/pq)}\}$, where $C' < \infty$ is a universal constant, and let $\mathcal{S} = \{S :\ S \subseteq J,\ |S| \le t\}$. Then $|\mathcal{S}| = P(f)^{O(1/\epsilon^2)}$ and $\sum_{S \notin \mathcal{S}} \tilde{f}(S)^2 \le \epsilon$.*

5.4 Learning monotone real-valued functions under p-biased measures.

With Theorem 9 in hand, the proof of our main learning result Theorem 6 is now not very difficult. Given an unknown target function $f : \{-1,1\}^n_{(p)} \to [-1,1]$ and $\epsilon > 0$, let f_ϵ denote f with its values "rounded" to the nearest integer multiples of ϵ. Clearly given examples from $EX(f,p)$ we can simulate examples from $EX(f_\epsilon, p)$. We now simply try to learn f_ϵ. It is easy to check that an ϵ-approximator hypothesis for f_ϵ is also an $O(\epsilon)$-approximator for f. Further, we have $P(f_\epsilon) \le P(f)$ so a $P(f_\epsilon)^{O(1/\epsilon^2)}$ runtime is also $P(f)^{O(1/\epsilon^2)}$ as desired. The p-biased analogue of Proposition 4 holds with essentially the same proof. The only new difficulty is that we cannot exactly estimate the quantities $\mathrm{Inf}_i(f_\epsilon)$. However from Fact 5, the quantities $\tilde{f}(i)$ — which we can estimate empirically — are upper bounds on the influences; so by taking all the coordinates i with $\tilde{f}(i) \ge \tau$, we get all the sufficiently influential coordinates. There cannot be too many coordinates with large $\tilde{f}(i)$, since $\sum_{i=1}^n \tilde{f}(i)^2 \le 1$.

For completeness, we give all the details of the proof of Theorem 6 in Appendix B.

6 Extension to stronger complexity measures?

It is natural to wonder whether our results can be extended to stronger complexity measures than decision tree size and partition size. An obvious next complexity measure to consider is the minimum number of (not necessarily disjoint) subcubes that cover $\{-1,1\}^n$ and are such that f is constant on each subcube. We refer to this as the *subcube covering complexity* of f and denote it by $CDNF(f)$, since it is equal to the minimum number of terms in any DNF formula for f plus the minimum number of clauses in any CNF formula for f.

The following theorem shows that Theorem 3 does not hold for subcube covering complexity:

Theorem 10 *There is a monotone Boolean function $g : \{-1,1\}^n \to \{-1,1\}$ for which $\mathrm{I}(g) = \Omega(n^{\log_4(6-2\sqrt{5})}) = \Omega(n^{0.305})$ but $\sqrt{\log CDNF(g)} = O(n^{1/4})$.*

The proof is by the probabilistic method. We define a distribution $\mathcal{D}$ over monotone Boolean functions and show that some function g that is assigned nonzero weight under $\mathcal{D}$ must satisfy the bounds of the theorem. See Appendix C.

7 Conclusion

In this paper we established a new bound on average sensitivity of monotone functions, and used this bound to give the first algorithm that uses random examples to learn any monotone function to high accuracy in time polynomial in the function's decision tree or partition size.

A natural goal for future work is to obtain even stronger learning results for monotone functions. Can the boosting methods used by Jackson in his Harmonic Sieve algorithm [13] be applied here? We note that while the Harmonic Sieve algorithm makes essential use of membership queries, related algorithms that combine boosting with Fourier techniques have been successfully developed for the framework of learning from random examples only [14].

8 Acknowledgments

We are grateful for several suggestions for simplifications of the proofs of Theorem 3.

References

[1] A. Blum. Learning a function of r relevant variables (open problem). In *Proceedings of the 16th Annual Conference on Learning Theory and 7th Kernel Workshop*, pages 731–733, 2003.

[2] A. BLum. Machine learning: a tour through some favorite results, directions, and open problems. FOCS 2003 tutorial slides, available at http://www-2.cs.cmu.edu/~avrim/Talks/FOCS03/tutorial.ppt, 2003.

[3] A. Blum, C. Burch, and J. Langford. On learning monotone boolean functions. In *Proceedings of the Thirty-Ninth Annual Symposium on Foundations of Computer Science*, pages 408–415, 1998.

[4] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the Twenty-Sixth Annual Symposium on Theory of Computing*, pages 253–262, 1994.

[5] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[6] N. Bshouty. Exact learning via the monotone theory. *Information and Computation*, 123(1):146–153, 1995.

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

[8] E. Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18(1):474–483, 1998.

[9] E. Friedgut and G. Kalai. Every monotone graph property has a sharp threshold. *Proceedings of the AMS*, 124:2993–3002, 1996.

[10] M. Furst, J. Jackson, and S. Smith. Improved learning of AC^0 functions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 317–325, 1991.

[11] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the Association for Computing Machinery*, 33(4):792–807, 1986.

[12] T. Hancock and Y. Mansour. Learning monotone k-μ DNF formulas on product distributions. In *Proceedings of the Fourth Annual Conference on Computational Learning Theory*, pages 179–193, 1991.

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

[14] J. Jackson, A. Klivans, and R. Servedio. Learnability beyond AC^0. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, 2002.

[15] J. Jackson and C. Tamon. Fourier analysis in machine learning. ICML/COLT 1997 tutorial slides, available at http://learningtheory.org/resources.html, 1997.

[16] S. Jukna, A. Razborov, P. Savický, and I. Wegener. On P versus NP∩co-NP for decision trees and read-once branching programs. *Computational Complexity*, 8(4):357–370, 1999.

[17] J. Kahn, G. Kalai, and N. Linial. The influence of variables on boolean functions. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 68–80, 1988.

[18] M. Kearns, M. Li, and L. Valiant. Learning Boolean formulas. *Journal of the ACM*, 41(6):1298–1328, 1994.

[19] M. Kearns and U. Vazirani. *An introduction to computational learning theory.* MIT Press, Cambridge, MA, 1994.

[20] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of the Twenty-Fifth Annual Symposium on Theory of Computing*, pages 372–381, 1993.

[21] L. Kucera, A. Marchetti-Spaccamela, and M. Protassi. On learning monotone DNF formulae under uniform distributions. *Information and Computation*, 110:84–95, 1994.

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

[23] Y. Mansour. *Learning Boolean functions via the Fourier transform*, pages 391–424. Kluwer Academic Publishers, 1994.

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

[25] Y. Mansour and M. Parnas. Learning conjunctions with noise under product distributions. *Information Processing Letters*, 68(4):189–196, 1998.

[26] Krzysztof Oleszkiewicz. On a nonsymmetric version of the khinchine-kahane inequality. *Progress In Probability*, 56:156–168, 2003.

[27] O. Schramm R. O'Donnell, M. Saks and R. Servedio. Every decision tree has an influential variable. In *Proceedings of the Forty-sixth Annual Symposium on Foundations of Computer Science*, pages 31–39, 2005.

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

[29] P. Savický. On determinism versus unambiguous nondeterminism for decision trees. ECCC report TR02-009, available at http://eccc.uni-trier.de/eccc-reports/2002/TR02-009/, 2002.

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

[31] L. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5:363–366, 1984.

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

[33] K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 314–326, 1990.

[34] K. Verbeurgt. Learning sub-classes of monotone DNF on the uniform distribution. In *Proceedings of the Ninth Conference on Algorithmic Learning Theory*, pages 385–399, 1998.

A Proof of Theorem 8 and a counterexample

Proof of Theorem 8: Recall that we have

$$D_i f(x) = \sqrt{pq}(f(x^{(i=1)}) - f(x^{(i=-1)})) = \sum_{S:\ i\in S} \tilde{f}(S)\phi_{S\setminus i}(x)$$

and that $\mathrm{Inf}_i^{(p)}(f) = \mathbf{E}_p[(D_i f)^2] = \|D_i f\|_2^2$, where throughout this section $\|\cdot\|$ denotes the norm induced by the p-biased measure.

Since $\mathrm{I}^{(p)}(f) = \sum_S |S|\tilde{f}(S)^2$, Markov's inequality immediately gives that $\sum_{S:\ |S|>t} \tilde{f}(S)^2 < \epsilon/2$. Let $J' = [n] \setminus J$. It now suffices to show that

$$\sum_{S:\ S\cap J'\neq\emptyset, |S|\leq t} \tilde{f}(S)^2 \leq \epsilon/2. \tag{4}$$

Certainly the left side of (4) is at most

$$\sum_{i\in J'} \sum_{S:\ i\in S, |S|\leq t} \tilde{f}(S)^2 = \sum_{i\in J'} \|D_i f^{\leq t}\|_2^2 = \sum_{i\in J'} \langle D_i f^{\leq t}, D_i f\rangle, \tag{5}$$

where we use the notation $f^{\leq t}$ to denote the function $f^{\leq t}(x) = \sum_{|S|\leq t} \tilde{f}(S)\phi_S$. Now we have:

$$\begin{aligned}
\langle D_i f^{\leq t}, D_i\rangle &\leq \|D_i f^{\leq t}\|_4 \, \|D_i f\|_{4/3} & (6)\\
&\leq (1 + 1/\sqrt{pq})^{t/2} \, \|D_i f^{\leq t}\|_2 \, \|D_i f\|_{4/3} & (7)\\
&\leq (1/pq)^t \, \mathrm{Inf}_i^{(p)}(f)^{1/2} \, \mathbf{E}[|D_i f|^{4/3}]^{3/4}. & (8)
\end{aligned}$$

Here (6) is Hölder's inequality, inequality (7) follows from a p-biased version of Bonami-Beckner (here with the best bounds provided by [26]), and inequality (8) uses the generous bound $(1 + 1/\sqrt{pq})^{1/2} < (1/pq)$ and also $\|D_i f^{\leq t}\|_2 \leq \|D_i f\|_2 = \mathrm{Inf}_i^{(p)}(f)^{1/2}$.

We now observe that by virtue of the assumption that f's range is contained in $\gamma\mathbf{Z}$, we have that $|D_i f(x)|$ is always either 0 or at least $\gamma\sqrt{pq}$. This implies that (8) is at most

$$\begin{aligned}
(1/pq)^t \, \mathrm{Inf}_i^{(p)}(f)^{1/2} \, \mathbf{E}_p[(\gamma\sqrt{pq})^{-2/3} \cdot |D_i f|^2]^{3/4} &= (1/pq)^{t+1/4} \, \gamma^{-1/2} \, \mathrm{Inf}_i^{(p)}(f)^{5/4}\\
&\leq (1/pq)^{t+1/4} \, \gamma^{-1/2} \, \mathrm{Inf}_i^{(p)}(f) \, \tau^{1/4}, & (9)
\end{aligned}$$

where we have used the definitions of $\mathrm{Inf}_i^{(p)}(f)$ and τ. Using the fact that $\sum_{i\in J'} \mathrm{Inf}_i^{(p)}(f) \leq \mathrm{I}^{(p)}(f)$, we can sum (9) and conclude that (5) is at most

$$(1/pq)^{t+1/4} \, \gamma^{-1/2} \, \mathrm{I}^{(p)}(f) \, \tau^{1/4}.$$

Thus to ensure (4) holds we only need

$$t(1/pq)^{t+1/4} \, \gamma^{-1/2} \, \tau^{1/4} \leq 1;$$

upper-bounding $t(1/pq)^{t+1/4}$ by $(1/pq)^{O(t)}$ (acceptable for all $t \geq 0$), we see that $\tau = \gamma^{O(1)}(pq)^{O(t)}$ suffices. Thus the choice of τ given in the definition of Theorem 8 suffices and the proof of this theorem is complete. ■

We now justify the remark from Section 5.3 indicating that Friedgut's theorem does not in general hold for real-valued functions; in other words, the condition that f's range is contained in $\gamma\mathbf{Z}$ cannot be removed.

To see this, consider (in the uniform measure case) the function $f : \{-1,1\}^n \to [-1,1]$ defined by

$$f(x) = \begin{cases} \mathrm{sgn}(\sum_{i=1}^n x_i) & \text{if } |\sum_{i=1}^n x_i| > \sqrt{n} \\ \frac{1}{\sqrt{n}} \sum_{i=1}^n x_i & \text{if } |\sum_{i=1}^n x_i| \le \sqrt{n}. \end{cases}$$

It is easy to see that for each $i = 1, \ldots, n$, $D_i f(x)$ is always either 0 or $1/\sqrt{n}$, and is $1/\sqrt{n}$ for a $\Theta(1)$ fraction of all x's. Consequently we have $\mathrm{Inf}_i(f) = \Theta(1/n)$ and thus $\mathrm{I}(f) = \Theta(1)$. In addition, it's clear that both $\mathbf{E}[f(x)] = 0$ and that $|f(x)| \ge 1/2$ for a $\Theta(1)$ fraction of all x's; hence we have $\sum_{|S|>0} \hat{f}(S)^2 \ge \Omega(1)$. But now if we take ϵ to be any constant smaller than this $\Omega(1)$ then we get a contradiction, since the choice of τ in Theorem 8 will be a constant and so J and hence $\mathcal{S}$ will be empty (for all n sufficiently large).

B Technical details for learning

We begin with some basic learning details for the p-biased measure. First, as mentioned earlier, we may assume without loss of generality that the learning algorithm "knows" p. The proof is quite similar to the proof that a noise-tolerant learning algorithm can be assumed to know the exact noise rate (see [19]). The basic idea is that we can run the learning algorithm repeatedly using successively finer estimates (easily obtained from sampling) for the value of p. If the original algorithm runs for T time steps, then if the guessed value for p is within Δ/T of the true value, the statistical distance between the algorithm's output when run with the guessed value versus the true value will be at most Δ. It can be shown that at most a polynomial factor runtime overhead is incurred in coming up with a sufficiently accurate guess; we give the details in the full version.

Next, we remark that low-degree algorithm of Linial, Mansour, and Nisan, Proposition 4, easily carries over to the real-valued p-biased case with essentially the same proof:

Proposition 8 *There is an algorithm A with the following property: Let $f : \{-1,1\}^n_{(p)} \to [-1,1]$ and let $\mathcal{S} \subseteq 2^{[n]}$ be a collection of subsets of $[n]$ with the property that $\sum_{S:S\notin\mathcal{S}} \tilde{f}(S)^2 \le \epsilon$. Then if A is given p, $\mathcal{S}$, access to a source $EX(f,p)$ of p-biased random examples, and parameters $\delta, \theta > 0$, it runs in poly$(n, |\mathcal{S}|, 1/\theta) \cdot \log(1/\delta)$ time and with probability $1-\delta$ outputs a real-valued function $g : \{-1,1\}^n \to \mathbf{R}$ of the form $g(x) = \sum_{S\in\mathcal{S}} c_S \phi_S(x)$ such that $\mathbf{E}_p[(f-g)^2] \le \epsilon + \tau$.*

We now proceed to discuss the proof of Theorem 6. Let $f : \{-1,1\}^n_{(p)} \to [-1,1]$ be the target function. Given $\epsilon > 0$, let f_ϵ denote the "rounded" version of f in which each of its values is rounded to the nearest integer multiple of ϵ. It is clear that given access to $EX(p,f)$ we can simulate access to $EX(p,f_\epsilon)$. Our algorithm will use $EX(p,f_\epsilon)$ to learn f_ϵ in time $\mathrm{poly}(n, P(f_\epsilon)^{O(1/\epsilon^2)}) \cdot \log(1/\delta)$. This is sufficient for learning f in the same time bound, because $P(f_\epsilon) \le P(f)$ and because if $\mathbf{E}_p[(h-f_\epsilon)^2] \le \epsilon$ then

$$\mathbf{E}_p[(h-f)^2] = \mathbf{E}_p[((h-f_\epsilon)+(f_\epsilon-f))^2] \le 2\mathbf{E}_p[(h-f_\epsilon)^2] + 2\mathbf{E}_p[(f_\epsilon-f)^2] \le 2\epsilon + \epsilon^2/2 = O(\epsilon).$$

Our goal is now essentially to use Proposition 8 given Theorem 9. As mentioned in Section 5.4, unlike in the algorithm for Boolean-valued functions we cannot estimate the influences of f_ϵ directly since the relationship $\mathrm{Inf}_i^{(p)}(f_\epsilon) = \tilde{f}_\epsilon(i)$ does not hold in the real-valued case. We may, however, use Fact 5 which says that $\tilde{f}_\epsilon(i)$ — a quantity we can empirically estimate — is an upper bound on $\mathrm{Inf}_i^{(p)}(f_\epsilon)$.

We now describe the algorithm to learn f_ϵ using $EX(p, f_\epsilon)$. As in Section 4.3 we may assume that the partition size $P(f_\epsilon)$ is known. The algorithm is as follows:

1. For $i = 1, \dots, n$ empirically estimate $\tilde{f}_\epsilon(i) = \mathbf{E}_p[f_\epsilon(x)\phi_i(x)]$ to within an additive $\pm\tau/4$ (with confidence $1-\delta$), where $\tau = (C')^{-t\log(1/pq)}$ and t is defined in Theorem 9. Let $J \subseteq [n]$ be the set of those i for which the obtained estimate is greater than $\tau/2$.

2. Now run algorithm A from Proposition 8 with $\mathcal{S} = \{S : S \subseteq J, |S| \le t\}$ and $\theta = \epsilon$, outputting its hypothesis g.

Let us first confirm the running time of this algorithm. In step (1), standard sampling bounds ensure that $\mathrm{poly}(n, 1/\tau) \cdot \log(1/\delta)$ samples suffice. We may then conclude that $|J| \le O(1/\tau^2)$, since $\sum_{i=1}^n \tilde{f}_\epsilon(S)^2 \le 1$. It follows that $|\mathcal{S}| \le \mathrm{poly}(1/\tau^t) = P(f_\epsilon)^{O(1/\epsilon^2)}$, as necessary to bound the running time. Finally, we still have $\sum_{S \notin \mathcal{S}} \tilde{f}_\epsilon(S)^2 \le \epsilon$ because (with confidence $1-\delta$) the J the algorithm finds is a superset of the J from Theorem 9. Hence the algorithm correctly gives a $O(\epsilon)$-approximator hypothesis g with confidence $1 - O(\delta)$, and the proof of Theorem 6 is complete.

C Proof of Theorem 10

Let $n = 4^k$. Let $f_1(a, b, c, d)$ be the "AND-OR" function on four Boolean variables $f_1(a, b, c, d) = (a \wedge b) \vee (c \wedge d)$. An important property of f_1 is that if each of its four arguments is independently set to be 1 (true) with probability p, then $\Pr[f_1 = 1]$ equals $2p^2 - p^4$. For $i = 2, 3, \dots,$ we define the function f_i on 4^i variables to be $f_i = f_1(f_{i-1}^1, f_{i-1}^2, f_{i-1}^3, f_{i-1}^4)$ where the superscripts indicate distinct copies of f_{i-1} on disjoint sets of variables. Thus f_k is a function on n variables computed by a read-once Boolean formula that is a tree of ANDs and ORs at alternating levels.

We now define distributions $\mathcal{D}_1, \dots, \mathcal{D}_k$ over monotone Boolean functions, where $\mathcal{D}_i$ is a distribution over functions from $\{-1,1\}^{4^i}$ to $\{-1,1\}$. The distribution $\mathcal{D}_i$ is defined in the following way: a random draw from $\mathcal{D}_i$ is obtained by independently substituting 1 for each of the 4^i Boolean arguments to f_i with probability α, where $\alpha = \sqrt{5} - 2 \approx 0.236$. (This construction and some of the subsequent analysis is reminiscent of [31].) Note that for a random g drawn from $\mathcal{D}_1$ and a random x drawn uniformly from $\{-1,1\}^4$, we have that each of the four arguments to f_1 is independently 1 with probability $\frac{1}{2} + \frac{\alpha}{2} = \frac{\sqrt{5}-1}{2}$; we denote this value by ρ. Consequently we have $\Pr_{g \in \mathcal{D}_1, x \in \{-1,1\}^4}[g(x) = 1] = 2\rho^2 - \rho^4$, but this is easily seen to equal ρ. It follows from the recursive definition of f_i that for all $i = 1, 2, \dots$ we have $\Pr_{g \in \mathcal{D}_i, x \in \{-1,1\}^{4^i}}[g(x) = 1] = \rho$.

It is not difficult to show (see Theorem 2.4 of [16]) that $CDNF(f_k) \le 2^{2^k+1}$; as an immediate consequence we have that $CDNF(g) \le 2^{2^k+1}$ (and thus $\sqrt{\log CDNF(g)} = O(2^{k/2}) = O(n^{1/4})$) for every g that is in the support of $\mathcal{D}_k$. But by Lemma 9 below we have that $\mathbf{E}_{g \in \mathcal{D}_k}[\mathrm{I}(g)] = \Theta((6 - 2\sqrt{5})^k)$; clearly this implies that there is some g in the support of $\mathcal{D}_k$ for which $\mathrm{I}(g)$ is $\Omega((6-2\sqrt{5})^k) = \Omega(n^{\log_4(6-2\sqrt{5})})$. This proves Theorem 10. ■

Lemma 9 *For $i = 1, 2, \dots$ we have $\mathbf{E}_{g \in \mathcal{D}_i}[\mathrm{I}(g)] = (3 - \sqrt{5})(6 - 2\sqrt{5})^i$.*

Proof: It is clear from symmetry that $\mathbf{E}_{g \in \mathcal{D}_i}[\mathrm{I}(g)] = 4^i \cdot \mathbf{E}_{g \in \mathcal{D}_i}[\mathrm{Inf}_1(g)]$. We have

$$\begin{aligned}
\mathbf{E}_{g \in \mathcal{D}_i}[\mathrm{Inf}_1(g)] &= \mathbf{E}_{g \in \mathcal{D}_i, x \in \{-1,1\}^{4^i}}[\Pr[g(1, x_2, \dots, x_{4^i}) \ne g(-1, x_2, \dots, x_{4^i})] \\
&= \Pr_{g \in \mathcal{D}_i, x \in \{-1,1\}^{4^i}}[g(1, x_2, \dots, x_{4^i}) \ne g(-1, x_2, \dots, x_{4^i})].
\end{aligned}$$

¿From the definition of $\mathcal{D}_i$, we have that with probability $\alpha = \sqrt{5} - 2$ the constant 1 is substituted for the first argument of f_i in g; if this occurs then clearly $g(1, x_2, \ldots, x_{4^i}) = g(-1, x_2, \ldots, x_{4^i})$ for all x since g does not depend on its first argument. If this does not occur, then we have (for a random $g \in \mathcal{D}_i$ and a uniform $x \in \{-1, 1\}^{4^i}$) that each of the other 4^{i-1} arguments to f_i independently takes value 1 with probability $\rho = \frac{\sqrt{5}-1}{2}$.

Under the distribution on inputs to f_i described in the previous paragraph, if $i = 1$ it is easy to see that flipping the first argument of f_1 flips the value of f_1 if and only if the second argument is 1 (probability ρ) and the AND of the third and fourth arguments is 0 (probability $1 - \rho^2$). Thus flipping the first argument of f_1 flips the value of f_1 with probability precisely $\rho(1 - \rho^2)$ which is easily seen to equal $1 - \rho$, using the fact that $2\rho^2 - \rho^4 = \rho$. Similarly, if $i = 2$, then flipping the first of the 16 arguments to $f_2 = f_1(f_1^1, f_1^2, f_1^3, f_1^4)$ (again under the distribution of inputs to f_i described above) will flip the value of f_2 if and only if the value of f_1^1 flips (probability $1 - \rho$ as shown above), f_1^2 equals 1 (probability ρ), and $f_1^3 \wedge f_1^4$ equals 0 (probability $1 - \rho^2$). We thus have that flipping the first argument of f_2 flips the value of f_2 with probability $(1-\rho)\rho(1-\rho^2) = (1-\rho)^2$. An easy induction in this fashion shows that for all i, under the distribution of inputs described above flipping the first argument of f_i causes f_i to flip with probability $(1 - \rho)^i$.

We thus have that

$$\Pr_{g \in \mathcal{D}_i, x \in \{-1,1\}^{4^i}} [g(1, x_2, \ldots, x_{4^i}) \neq g(-1, x_2, \ldots, x_{4^i})] = (1 - \alpha)(1 - \rho)^i = (3 - \sqrt{5}) \left(\frac{3 - \sqrt{5}}{2} \right)^i,$$

which proves the lemma. ■