Learning juntas

Elchanan Mossel
Microsoft Research
mossel@microsoft.com

Ryan O'Donnell*
MIT Department of Mathematics
odonnell@theory.lcs.mit.edu

Rocco A. Servedio†
Department of Computer Science
Columbia University
rocco@deas.harvard.edu

Abstract

We consider a fundamental problem in computational learning theory: learning an arbitrary Boolean function which depends on an unknown set of k out of n Boolean variables. We give an algorithm for learning such functions from uniform random examples which runs in time roughly $(n^k)^{\frac{\omega}{\omega+1}}$, where $\omega < 2.376$ is the matrix multiplication exponent. We thus obtain the first polynomial factor improvement on the naive n^k time bound which can be achieved via exhaustive search. Our algorithm and analysis exploit new structural properties of Boolean functions.

*Supported by NSF grant 99-12342.

†Supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship and by NSF grant CCR-98-77049. This research was performed while the third author was at Harvard University.

1 Introduction

1.1 Background and motivation

One of the most important and challenging issues in machine learning is how to learn efficiently and effectively in the presence of irrelevant information. Many real-world learning problems can be modeled in the following way: we are given a set of labeled data points and we wish to find some hypothesis which accurately predicts the label of each data point. An oft-encountered situation in this framework is that each data point contains a large amount of information (i.e., each data point is a high dimensional vector of attribute values over a fixed large set of attributes), but only a small unknown portion of this information is relevant to the label of the data point (i.e., the label is determined by a function which only depends on a few of the attributes). For example, in a computational biology scenario each data point may correspond to a long DNA sequence, and the label may be some property which depends only on a small unknown active part of this sequence.

In this paper we consider the following learning problem which Blum [3] and Blum and Langley [4] proposed as a clean formulation of learning in the presence of irrelevant information: Let f be an unknown Boolean function over an n-bit domain which depends only on an unknown subset of $k \ll n$ variables. Such a function is called a *k-junta*. Given a data set of labeled examples $\langle x, f(x)\rangle$, where the points x are independently and uniformly chosen random n-bit strings, can the function f be learned by a computationally efficient algorithm? (We give a precise description of what it means to "learn f" in Section 2.) Note that a naive brute force search over all possible subsets of k relevant variables can be performed in time roughly n^k; we would like to have an algorithm which runs faster than this.

We believe that the problem of efficiently learning k-juntas is the single most important open question in uniform distribution learning. In addition to being natural and elegant, learning juntas is at the heart of the most notorious open problems in uniform distribution learning, namely learning DNF formulas and decision trees of superconstant size. Since every k-junta can be expressed as a decision tree or DNF formula of size 2^k, it is clear that efficient algorithms for learning 2^k-size decision trees or DNFs would also be efficient algorithms for learning k-juntas. But in fact more is true: obtaining efficient algorithms for decision trees or DNFs *requires* that we be able to efficiently learn juntas. Specifically, any size-k decision tree is also a k-junta, and any k-term DNF is ϵ-indistinguishable (under the uniform distribution) from a $k\log(k/\epsilon)$-junta. Thus, learning $\omega(1)$-size decision trees or $\omega(1)$-term DNFs in polynomial time is equivalent to learning $\omega(1)$-juntas in polynomial time.

We note that learning from uniform random examples seems to be the model in which this problem has the right amount of difficulty. As described in Section 5, allowing the learner to make membership queries makes the problem too easy, while restricting the learner to the statistical query model makes the problem provably hard.

1.2 Our results

We give the first learning algorithm for the problem of learning k-juntas which achieves a polynomial factor improvement over brute force exhaustive search. Under the uniform distribution, our algorithm exactly learns an unknown k-junta with confidence $1-\delta$ in time $n^{\frac{\omega}{\omega+1}k} \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$, where ω is the exponent in the time bound for matrix multiplication. Since Coppersmith and Winograd [7] have shown that $\omega < 2.376$, our algorithm runs in time roughly $N^{.704}$ where $N \approx n^k$ is the running time of a naive brute force approach. Our algorithm and analysis exploit new structural properties of Boolean functions which may be of independent interest.

We note that since this learning problem was first posed by Blum in 1994, little progress has been made. The first improvement over the trivial n^k time bound of which we are aware is a recent algorithm due to A. Kalai and Mansour [12] which runs in time roughly $n^{k-\Omega(k^{1/4})}$. Mansour [18] later improved this to $n^{k-\Omega(k^{1/2})}$. (In recent work Fischer *et al.* have studied the problem of *testing* k-juntas [8], but the learning and testing problems seem to require different techniques.)

1.3 Organization

In Section 2 we formally define the learning problem and give some background on polynomial representations of Boolean functions. In Section 3 we present the learning algorithms which allow us to reduce the learning problem to some questions about representing Boolean functions as polynomials. In Section 4 we prove the necessary new structural properties of Boolean functions and thus obtain our learning result. Finally, in Section 5 we use the developed machinery to analyze several variants of the juntas problem.

2 Preliminaries

The learning model we consider is a uniform distribution version of Valiant's Probably Approximately Correct (PAC) model [21] which has been studied by many researchers, e.g., [6, 10, 11, 14, 15, 17, 22, 23]. In this model a *concept class* C is a collection $\cup_{n\geq 1} C_n$ of Boolean functions, where each $c \in C_n$ is a function on n bits. Let $f \in \mathcal{C}_n$ be an unknown *target function.* A learning algorithm A for C takes as input an *accuracy parameter* $0 < \epsilon < 1$ and a *confidence parameter* $0 < \delta < 1$. During its execution A has access to an *example oracle* $\mathrm{EX}(f)$ which, when queried, generates a random labeled example $\langle x, f(x)\rangle$ where x is drawn uniformly from $\{0,1\}^n$. A outputs a hypothesis h which is a Boolean function over $\{0,1\}^n$; the error of this hypothesis is defined to be $\mathrm{error}(h,f) = \Pr[h(x) \neq f(x)]$. (Here and in the remainder of the paper, unless otherwise indicated all probabilities are taken over x chosen uniformly at random from $\{0,1\}^n$.) We say that A is a *uniform-distribution PAC learning algorithm for C* if the following condition holds: for every $f \in C$ and every ϵ, δ, with probability at least $1-\delta$ algorithm A outputs a hypothesis h which has $\mathrm{error}(h,f) \leq \epsilon$. For the purposes of this paper the accuracy parameter ϵ will always be 0, so our goal is to exactly identify the unknown function.

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to *depend on* the ith variable if there exist inputs $x, y \in \{0,1\}^n$ which differ only in the ith coordinate and which have $f(x) \neq f(y)$. Equivalently, we say that such a variable is *relevant* to f. If the function f has at most k relevant variables then we call f a *k-junta.* The concept class we consider in this paper is the set of k-juntas over n variables, i.e., $C_n = \{f : \{0,1\}^n \to \{0,1\} \text{ s.t. } f \text{ is a } k\text{-junta}\}$.

Equivalently, each function $f \in C_n$ is defined by a subset $R = \{i_1, \ldots, i_{k'}\} \subset \{1, \ldots, n\}$ of $k' \le k$ relevant variables and a truth table of $2^{k'}$ bits corresponding to all possible settings of these variables.

We are most interested in the case where k is $O(\log n)$ or even a large constant value. For such k the number of possible sets of relevant variables is $n^{(1-o(1))k}$. Hence the naive learning algorithm which performs an exhaustive search over all possible subsets of relevant variables will take time $n^{(1-o(1))k}$.

2.1 Representing Boolean functions as polynomials

A Boolean function g on n bits is a mapping $\{\mathrm{F}, \mathrm{T}\}^n \to \{\mathrm{F}, \mathrm{T}\}$. There are many possible ways to represent g as a multilinear polynomial. Since our analysis will use several different representations, we give a general definition which encompasses all of the cases we will need:

Definition 1 *Let $\mathbf{F}$ be a field and let $f, t \in \{-1, 0, 1\}$ be distinct elements of $\mathbf{F}$. We say that a multilinear polynomial p $\langle \mathbf{F}, f, t \rangle$-represents g if $p : \mathbf{F}^n \to \mathbf{F}$ has the following properties:*

- *for all inputs in $\{f, t\}^n$, p outputs a value in $\{f, t\}$; and,*
- *p and g induce the same mapping when* F *and* T *are identified with f and t in the input and output.*

Note that since $f^2, t^2 \in \{0, 1\}$, the assumption that p is multilinear is without loss of generality. It is well known that the $\langle \mathbf{F}, f, t \rangle$-representation of g always exists and is unique; for completeness we give a simple proof below.

Proposition 2 *Every Boolean function g has a unique multilinear $\langle \mathbf{F}, f, t \rangle$-representation.*

Proof: The condition that p is a multilinear polynomial which represents g is equivalent to a system of 2^n linear equations in 2^n unknowns, where the unknowns are the coefficients on the 2^n multilinear monomials. Let A_n denote the $2^n \times 2^n$ matrix arising from this linear system, so the columns of A_n correspond to monomials and the rows correspond to truth assignments. It suffices to prove that A_n has full rank; we now prove this by induction.

In the case $n = 1$ we have $A_1 = \begin{pmatrix} 1 & f \\ 1 & t \end{pmatrix}$ which has full rank over any field since $f \ne t$. In the general case, one can rearrange the rows and columns of A_n to get $A_n = \begin{pmatrix} A_{n-1} & fA_{n-1} \\ A_{n-1} & tA_{n-1} \end{pmatrix}$, where the columns on the left correspond to monomials not containing x_n and the others correspond to monomials containing x_n. By performing elementary row operations on this matrix, one can get $\begin{pmatrix} (f-t)A_{n-1} & 0 \\ 0 & A_{n-1} \end{pmatrix}$. Since $f \ne t$ and A_{n-1} has full rank by induction, this has full rank. $\square$

The fields we will consider in this paper are the two-element field $\mathbf{F}_2$ and the field $\mathbf{R}$ of real numbers. In $\mathbf{F}_2$ we will represent bits by $f = 0$, $t = 1$, and in $\mathbf{R}$ we will usually represent bits by $f = 1$, $t = -1$.

Definition 3 *Given a Boolean function g on n bits:*

- *We write $g_{\mathbf{F}_2}$ for the multilinear polynomial which $\langle \mathbf{F}_2, 0, 1\rangle$-represents g, and we say that $g_{\mathbf{F}_2}$ $\mathbf{F}_2$-represents g. Note that $g_{\mathbf{F}_2}$ can be viewed as a parity of ANDs since $\mathbf{F}_2$ multiplication corresponds to AND and $\mathbf{F}_2$ addition corresponds to parity.*

- *We write $g_{\mathbf{R}}$ for the multilinear polynomial which $\langle \mathbf{R}, +1, -1\rangle$-represents g, and we say that $g_{\mathbf{R}}$ $\mathbf{R}$-represents g. Note that $g_{\mathbf{R}}$ is precisely the "Fourier representation" of g. As is standard, we write $\hat{g}(S)$ for the coefficient of x_S in $g_{\mathbf{R}}$, where x_S denotes the monomial $\prod_{i\in S} x_i$. We call $\hat{g}(S)$ the "S Fourier coefficient of g."*

As an example, if $g = \mathsf{PARITY}_n$ then we have $g_{\mathbf{F}_2} = x_1 + x_2 + \cdots + x_n$ and $g_{\mathbf{R}} = x_1 x_2 \cdots x_n$. Note that there is a huge difference in the degrees of these two polynomial representations; we will be very interested in the degree of Boolean functions under various representations. We observe that for a given field this degree is independent of the exact choice of f, t. This is because we can pass back and forth between any two such choices by nonconstant linear transformations on the inputs and outputs, and under such transformations the monomials of highest degree can never vanish. Thus we can make the following definition:

Definition 4 *$\deg_{\mathbf{F}}(g)$ is defined to be $\deg(p)$ where p is any $\langle \mathbf{F}, f, t\rangle$-representation of g.*

Hence we have $\deg_{\mathbf{F}_2}(\mathsf{PARITY}_n) = 1$ and $\deg_{\mathbf{R}}(\mathsf{PARITY}_n) = n$. In general $\deg_{\mathbf{F}_2}(g) \leq \deg_{\mathbf{R}}(g)$:

Fact 5 *For any Boolean function g, $\deg_{\mathbf{F}_2}(g) \leq \deg_{\mathbf{R}}(g)$.*

Proof: Let p be the $\langle \mathbf{R}, 0, 1\rangle$-representation of g and let $g_{\mathbf{F}_2}$ be the $\mathbf{F}_2$-representation of g. We have

$$p(x) = \sum_{z\in\{0,1\}^n} \left[p(z) \left(\prod_{i:z_i=1} x_i \right) \left(\prod_{i:z_i=0} (1-x_i) \right) \right].$$

This polynomial clearly has integer coefficients; $g_{\mathbf{F}_2}$ is obtained by reducing the coefficients of p mod 2, and this operation can only decrease degree. □

3 Learning tools

In this section we give the learning algorithms we will use for solving the junta problem. We first show that it suffices to give a learning algorithm which can identify a single relevant variable. We then give two learning algorithms that look for relevant variables. Our algorithm for learning k-juntas will end up trying both algorithms and we shall prove in Section 4 that at least one of them always works.

Throughout this section, f will denote a k-junta on n bits, R will denote the set of variables on which f depends, k' will denote $|R|$ (so $0 \leq k' \leq k$), and f' will denote the function $\{\mathrm{F}, \mathrm{T}\}^{k'} \to \{\mathrm{F}, \mathrm{T}\}$ given by restricting f to R.

3.1 Finding a single relevant variable is enough

Proposition 6 *Suppose that $\mathcal{A}$ is an algorithm running in time $n^\alpha \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$ which can identify at least one variable relevant to f with confidence $1-\delta$ (assuming f is nonconstant). Then there is an algorithm for exactly learning f which runs in time $n^\alpha \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$.*

Proof: First note that if f is nonconstant then for uniform random inputs each output value occurs with frequency at least $1/2^k$. Hence we can decide whether or not f is a constant function with confidence $1-\delta$ in time in time $\mathrm{poly}(2^k, n, \log(1/\delta))$.

Next, suppose ρ is any restriction fixing at most k bits. We claim that we can run any learning algorithm on $f|_\rho$ with a slowdown of at most $\mathrm{poly}(2^k)$. To do so, we only need to transform the example oracle for f into one for $f|_\rho$; this is easily done by rejecting all samples $\langle x, f(x)\rangle$ for which x does not agree with ρ. Since ρ fixes at most k bits, the probability that a random x agrees with ρ is at least 2^{-k}. Hence with probability $1-\delta$ we can get M samples for $f|_\rho$ by taking $M \cdot \mathrm{poly}(2^k)\log(M/\delta)$ samples from the oracle for f.

We now show how to identify all the variables R on which f depends in the requisite amount of time. By induction, suppose we have identified some relevant variables $R' \subseteq R$. For each of the $2^{|R'|}$ possible restrictions ρ which fix the bits in R', consider the function $f|_\rho$. Since $f|_\rho$ is also a k-junta, $\mathcal{A}$ can identify some variables relevant to $f|_\rho$ (or else we can check that $f|_\rho$ is constant). By running $\mathcal{A}$ (with the slowdown described above) for each possible ρ, we will identify new variables to add into R'. We repeatedly add new variables to R', testing all restrictions on these variables, until all of the restricted subfunctions are constant. It is clear that at this point we will have identified all variables relevant to f.

Note that R' grows by at least one variable at each stage, and so we will never run $\mathcal{A}$ more than $k2^k$ times. Further, we can get confidence $1-\delta/k2^k$ for each run — even after the rejection-sampling slowdown — in time $n^\alpha \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$. Hence we can identify R in time $n^\alpha \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$ with confidence $1-\delta$.

Finally, once R is identified it is easy to learn f exactly. Simply draw $\mathrm{poly}(2^k, \log(1/\delta))$ samples; with probability $1-\delta$ we will see every possible bit setting for R so we can build f's truth table and output this as our hypothesis. $\square$

3.2 The Fourier-based learning algorithm

We describe a simple Fourier-based algorithm for trying to identify a variable relevant to f. The algorithm is based on the "Low Degree" learning algorithm of Linial, Mansour, and Nisan [15] (see also [16]). As with the Low Degree algorithm, our Fourier-based algorithm tries to learn the unknown function f by estimating all of f's Fourier coefficients $\hat{f}(S)$ with $1 \le |S| \le \alpha$. Unlike the Low Degree algorithm, our algorithm can stop as soon as it finds a nonzero coefficient, since all variables in the associated monomial must be relevant to f.

We first show how to compute the exact value of any desired Fourier coefficient:

Proposition 7 *We can exactly calculate any Fourier coefficient $\hat{f}(S)$ with confidence $1-\delta$ in time* $\mathrm{poly}(2^k, n, \log(1/\delta))$.

Proof: We view bits as being ± 1, as in the $\mathbf{R}$-representation. After multilinear reduction we see that the polynomial $x_S f_{\mathbf{R}}(x)$ has $\hat{f}(S)$ as its constant coefficient. Since $\mathrm{E}[x_T] = 0$ for all nonempty subsets T, linearity of expectation lets us conclude that $\mathrm{E}[x_S f_{\mathbf{R}}(x)] = \hat{f}(S)$. We can clearly compute the value of $x_S f_{\mathbf{R}}(x)$ in linear time given a labeled example $\langle x, f(x)\rangle$. By standard Chernoff bounds, $\mathrm{poly}(2^k, \log(1/\delta))$ independent samples of the ± 1 random variable $x_S f_{\mathbf{R}}(x)$ are sufficient for computing the expectation to within $\pm 1/2^{k+1}$ with confidence $1-\delta$. Since f and f' have the same Fourier expansion and f' is a function on at most k variables, $\hat{f}(S)$ must be of the form $a/2^k$ for some integer $a \in [-2^k, 2^k]$. Hence by rounding the empirical expectation to the nearest integer multiple of $1/2^k$ we will get the exact value of $\hat{f}(S)$ with confidence $1-\delta$. $\square$

The next proposition says that if f' has a nonzero Fourier coefficient of small (but nonzero) degree, then we can efficiently identify some variables relevant to f.

Proposition 8 *If $\hat{f}'(S) \neq 0$ for some S with $1 \leq |S| \leq \alpha$, then we can identify at least one relevant variable for f with confidence $1-\delta$ in time $n^\alpha \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$.*

Proof: We use Proposition 7 to compute each Fourier coefficient $\hat{f}(S)$, $1 \leq |S| \leq \alpha$, with confidence $1 - \delta/n^\alpha$. Since there are at most n^α possible sets S, with confidence $1-\delta$ we will obtain the exact values of all the desired Fourier coefficients in time at most $n^\alpha \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$. Since f and f' have the same Fourier coefficients, we will find an S with $\hat{f}(S) \neq 0$. It is easy to see that every variable in S must be relevant to f; for if f is does not depend on x_i then $\hat{f}(S) = \mathrm{E}[x_S f_{\mathbf{R}}(x)] = \mathrm{E}[x_i]\mathrm{E}[x_{S-i} f_{\mathbf{R}}(x)] = 0 \cdot \mathrm{E}[x_{S-i} f_{\mathbf{R}}(x)] = 0$. $\square$

3.3 The $\mathbf{F}_2$-based learning algorithm

In this subsection we show that if f' is a low-degree polynomial over $\mathbf{F}_2$, then in fact we can learn f' exactly. Here we view True and False as 1 and 0 respectively.

Recall the following well-known result from computational learning theory [9]:

Theorem 9 *Let $g : \{0,1\}^N \to \{0,1\}$ be a parity function on an unknown subset of the N Boolean variables $x_1, \ldots, x_N$. There is a learning algorithm $\mathcal{B}$ which, given access to labeled examples $\langle x, g(x)\rangle$ drawn from* any *probability distribution $\mathcal{D}$ on $\{0,1\}^N$, outputs a hypothesis h (which is a parity of some subset of $x_1, \ldots, x_N$) such that with probability $1-\delta$ we have $\Pr_{x\in\mathcal{D}}[h(x) \neq g(x)] \leq \epsilon$. Algorithm $\mathcal{B}$ runs in time $O((\frac{N}{\epsilon} + \frac{\log 1/\delta}{\epsilon})^\omega)$ where $\omega < 2.376$ is the exponent for matrix multiplication.*

The idea behind Theorem 9 is simple: since g is a parity function, each labeled example $\langle x, g(x)\rangle$ corresponds to a linear equation over $\mathbf{F}_2$ where the ith unknown corresponds to whether x_i is present in g. Algorithm $\mathcal{B}$ draws $O(\frac{N}{\epsilon} + \frac{\log 1/\delta}{\epsilon})$ examples and solves the resulting system of linear equations to find some parity over $x_1, \ldots, x_N$ which is consistent with all of the examples. Well-known results in PAC learning theory [5] imply that such a consistent parity will satisfy the ϵ, δ criterion.

Now suppose $\deg_{\mathbf{F}_2}(f') = \alpha \leq k$. Then f' is a $\mathbf{F}_2$-linear combination (i.e., a parity) over the set of monomials (conjunctions) in $x_1, \ldots, x_n$ of degree up to α. This lets us learn f' in time roughly $n^{\omega\alpha}$:

Proposition 10 *If* $\deg_{\mathbf{F}_2}(f') = \alpha$*, then we can learn* f *exactly in time* $n^{\omega\alpha} \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$ *with confidence* $1 - \delta$*. (Hence we can certainly identify a variable on which* f *depends.)*

Proof: Consider the expanded variable space consisting of all monomials over $x_1, \ldots, x_n$ of degree at most α. There are at most $N = n^\alpha$ variables in this space. Run algorithm $\mathcal{B}$ from Theorem 9 on this variable space, with ϵ set to $2^{-(k+1)}$. That is, given an example $\langle x, f(x)\rangle$, translate it to the example $\langle (x_S)_{|S| \le \alpha}, f(x)\rangle$, and run $\mathcal{B}$ using this new example oracle. Simulating a draw from this new oracle takes time $N \cdot \mathrm{poly}(n)$, so constructing all the necessary examples for $\mathcal{B}$ takes time $N^2 \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$. Solving the resulting system of equations takes time $N^\omega \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$. Hence the total time for the algorithm is $n^{\omega\alpha} \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$ as claimed.

We now argue that $\mathcal{B}$'s output hypothesis is precisely the $\mathbf{F}_2$-representation of f. Let $\mathcal{D}$ be the distribution over the expanded variable space induced by the uniform distribution on $x_1, \ldots, x_n$. Since f' (equivalently f) is a parity over the expanded variable space, the output of $\mathcal{B}$ will be a parity hypothesis h over the expanded variable space which satisfies $\Pr_{x \in \mathcal{D}}[h(x) \neq f(x)] \le 2^{-(k+1)}$. View both f and h as $\mathbf{F}_2$-polynomials of degree α over the original variables $x_1, \ldots, x_n$.

If f and h are not identical, then $f + h \not\equiv 0$ and we have $\Pr[f(x) \neq h(x)] = \Pr[f(x) + h(x) \neq 0]$. Now since $\deg_{\mathbf{F}_2}(f + h) \le \alpha$ and $f + h$ is not identically 0, the polynomial $f + h$ must be nonzero on at least a $2^{-\alpha} \ge 2^{-k}$ fraction of the points in $(\mathbf{F}_2)^n$. (This is a slightly nonstandard form of the Schwartz-Zippel Lemma; see [20] for an explicit proof.) But this contradicts the fact that $\Pr_{x \in \mathcal{D}}[h(x) \neq f(x)] \le 2^{-(k+1)}$. □

4 Learning juntas via new structural properties of Boolean functions

With our learning tools in hand we are ready to give the algorithm for learning k-juntas. The basic idea is to show that every Boolean function f' must either have a nonzero Fourier coefficient of "not too large" positive degree, or must be a polynomial over $\mathbf{F}_2$ of "not too large" degree. Then by Propositions 8 and 10, in either case we can find a relevant variable for f' without performing a full-fledged exhaustive search.

The Fourier learning algorithm described earlier fails only on functions whose low-degree Fourier coefficients are all zero (except for possibly the constant coefficient; if this is nonzero the Fourier algorithm can still fail). Let us make a definition for such functions:

Definition 11 *Suppose that* g *satisfies* $\hat{g}(S) = 0$ *for all* $1 \le |S| < t$*. If* $\hat{g}(\emptyset)$ *is also* 0 *then we say that* g *is* strongly balanced up to size t*. If* $\hat{g}(\emptyset)$ *is nonzero we say that* g *is* strongly biased up to size t*.*

These definitions were essentially first made by Bernasconi in [2]. The justification of the terminology is this: if g is strongly balanced up to size t, then it is easy to show that every subfunction of g obtained by fixing $0 \le \ell \le t - 1$ bits is balanced (i.e. is true with probability exactly $1/2$). Similarly, if g is strongly biased up to size t then it is easy to show that every such subfunction has the same bias as g itself.

We now show that strongly balanced functions have low $\mathbf{F}_2$-degree:

Theorem 12 *Let $g \notin \{\mathsf{PARITY}_n, \neg\mathsf{PARITY}_n\}$ be a Boolean function on n bits which is strongly balanced up to size t. Then $\deg_{\mathbf{F}_2}(g) \le n - t$.*

Proof: Given such a g, let $h = g \oplus \mathsf{PARITY}_n$. Then $h_{\mathbf{R}} = g_{\mathbf{R}} \cdot x_1 x_2 \cdots x_n$. By assumption, $g_{\mathbf{R}}$ has zero coefficient on all monomials x_S with $|S| < t$. By multilinear reduction ($x_i^2 = 1$) we see that $h_{\mathbf{R}}$ has zero coefficient on all monomials x_S with $|S| > n-t$. Hence $\deg_{\mathbf{R}}(h) \le n-t$, so by Fact 5, $\deg_{\mathbf{F}_2}(h) \le n - t$. But since $g = h \oplus \mathsf{PARITY}_n$, the $\mathbf{F}_2$-representation of g is simply $g_{\mathbf{F}_2}(x) = h_{\mathbf{F}_2}(x) + x_1 + \cdots + x_n$. Adding a degree 1 polynomial to $h_{\mathbf{F}_2}$ does not increase degree (since g is neither PARITY_n nor its negation, h is not a constant function and hence $\deg_{\mathbf{F}_2}(h) \ge 1$), and consequently $\deg_{\mathbf{F}_2}(g) \le n - t$. □

The bound $n - t$ in Theorem 12 is best possible. To see this, consider the function

$$g(x) = (x_1 \wedge \cdots \wedge x_{n-t}) \oplus x_{n-t+1} \oplus \cdots \oplus x_n.$$

This function has $\mathbf{F}_2$-representation $g_{\mathbf{F}_2}(x) = x_1 \cdots x_{n-t} + x_{n-t+1} + \cdots + x_n$ so $\deg_{\mathbf{F}_2}(g) = n-t$. Moreover, g is balanced and every subfunction of g fixing fewer than t bits is also balanced, since to make g unbalanced one must restrict all of $x_{n-k+1}, \ldots, x_n$.

It remains to deal with strongly biased functions. Our next theorem shows that no Boolean function can be strongly biased up to too large a size:

Theorem 13 *If g is a Boolean function on n bits which is strongly biased up to size t, then $t \le \frac{2}{3}n$.*

Proof: Let $g_{\mathbf{R}}(x) = \sum_S c_S x_S$ be the $\mathbf{R}$-representation of g. Since g is strongly biased up to size t we have $0 < |c_\emptyset| < 1$ and $c_S = 0$ for all $0 < |S| < t$. As in Theorem 12, we let $h = g \oplus \mathsf{PARITY}_n$ so $h_{\mathbf{R}}(x) = c_\emptyset x_1 x_2 \cdots x_n + \sum_{|S| \le n-t} c'_S x_S$, where $c'_S = c_{[n] \setminus S}$.

Let $h' : \{+1,-1\}^n \to \{1+c_\emptyset, 1-c_\emptyset, -1+c_\emptyset, -1-c_\emptyset\}$ be the real-valued function given by $h'(x) = h_{\mathbf{R}}(x) - c_\emptyset x_1 x_2 \cdots x_n$; note that $\deg(h') \le n-t$. Furthermore, for $x \in \{+1,-1\}^n$ we have $h'(x) \in \{1+c_\emptyset, 1-c_\emptyset\}$ iff $h_{\mathbf{R}}(x) = +1$, and $h'(x) \in \{-1+c_\emptyset, -1-c_\emptyset\}$ iff $h_{\mathbf{R}}(x) = -1$. Since $0 < |c_\emptyset| < 1$ we have that $\{1+c_\emptyset, 1-c_\emptyset\}$ and $\{-1+c_\emptyset, -1-c_\emptyset\}$ are disjoint two-element sets.

Let $p : \mathbf{R} \to \mathbf{R}$ be the degree 3 polynomial which maps $1 + c_\emptyset$ and $1 - c_\emptyset$ to $+1$ and $-1 - c_\emptyset$ and $-1 + c_\emptyset$ to -1. Now consider the polynomial $p \circ h'$. By construction $p \circ h'$ maps $\{+1,-1\}^n \to \{+1,-1\}$, and $p \circ h'$ $\mathbf{R}$-represents h. But the $\mathbf{R}$-representation of h is unique, so after multilinear reduction $p \circ h'$ must be identical to $h_{\mathbf{R}}$. Since $c_\emptyset \ne 0$, we know that $\deg_{\mathbf{R}}(h)$ is exactly n. Since p has degree exactly 3 and $\deg(h') \le n - t$, we conclude that $3(n-t) \ge n$, whence $t \le \frac{2}{3}n$. □

O. Regev [19] communicated to us an alternate proof of Theorem 13 which we include here with his permission:

Proof: (Regev) Let $g_{\mathbf{R}}(x)$ be as in the previous proof. Let U be any set of maximal size such that $c_U \ne 0$; since g is nonconstant and strongly biased up to size t we have $|U| \ge t$. Consider expanding $g_{\mathbf{R}}(x)^2 = (\sum_S c_S x_S)(\sum_T c_T x_T)$; there will be a nonzero coefficient on the cross-term $x_\emptyset x_U$. But $g_{\mathbf{R}}(x)^2$ must be identically 1 after multilinear reduction, since $g_{\mathbf{R}}$

takes on only the values ± 1. Thus the nonzero coefficient on x_U must be cancelled in the expansion. But if $t > \frac{2}{3}n$ then $c_T = 0$ for all $1 \leq |T| \leq \frac{2}{3}n$, and so all nonzero cross-terms not involving the constant term $c_\emptyset$ will be on terms x_V with $V < \frac{2}{3}n$. This contradicts the fact that the x_U term must be cancelled. □

The bound $\frac{2}{3}n$ in Theorem 13 is best possible. To see this, let $n = 3m$ and consider the function

$$f(x_1, \ldots, x_n) = \left(\bigoplus_{i=1}^{2m} x_i \right) \bigwedge \left(\bigoplus_{i=m+1}^{n} x_i \right).$$

It is easy to see that this function is unbalanced, and also that its bias cannot change under any restriction of fewer than $2m$ bits (to change the bias, one must set bits $1 \ldots 2m$ or $m+1 \ldots 3m$ or $1 \ldots m, 2m+1 \ldots 3m$).

We can now prove our main theorem:

Theorem 14 *The class of k-juntas over n bits can be exactly learned under the uniform distribution with confidence $1-\delta$ in time $n^{\frac{\omega}{\omega+1}k} \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$.*

Proof: Let f be a k-junta on n bits and f' be the function on at most k bits given by restricting f to its relevant variables. Let $t = \frac{\omega}{\omega+1}k > \frac{2}{3}k$. If f' is strongly balanced up to size t then by Theorem 12 f' is an $\mathbf{F}_2$-polynomial of degree at most $k - t = k/(\omega+1)$. By Proposition 10 f' can be learned in time $(n^{k/(\omega+1)})^\omega \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$. On the other hand, suppose f' is not strongly balanced up to size t. By Theorem 13, f' cannot be strongly biased up to size t, since $t > \frac{2}{3}k$. Hence f' has a nonzero Fourier coefficient of degree less than t and greater than 0. So by Proposition 8, some relevant variable for f can be identified in time $n^t \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$.

In either case, we can identify some relevant variable for f in time $n^{\frac{\omega}{\omega+1}k} \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$. Proposition 6 completes the proof. □

5 Variants of the junta learning problem

We can use the ideas developed thus far to analyze some variants and special cases of the juntas learning problem.

5.1 Some easier special cases

For various subclasses of k-juntas, the learning problem is more easily solved.

Monotone juntas: It is easy to verify that if f' is a monotone function, then $\hat{f}'(\{i\}) > 0$ for every relevant variable x_i. (Use the fact that $\hat{f}'(\{i\}) = \mathrm{E}[x_i f'(x)] = \Pr[f(x) = x_i] - \Pr[f(x) \neq x_i]$.) Hence monotone juntas can be learned in time $\mathrm{poly}(2^k, n, \log(1/\delta))$ using the Fourier learning algorithm of Proposition 8.

Random juntas: As observed in [4], almost every k-junta on n variables can be learned in time $\mathrm{poly}(2^k, n, \log(1/\delta))$. To see this, observe that if a function f' on k bits is chosen uniformly at random, then for every S we have $\hat{f}'(S) = 0$ only if exactly half of all inputs have $f'(x) = x_S$. This occurs with probability $\binom{2^k}{2^{k-1}}/2^{2^k} = O(1)/2^{k/2}$. Consequently, with overwhelming probability in terms of k — at least $1 - O(k)/2^{k/2}$ — a random function on k variables will have every Fourier coefficient of degree 1 nonzero, and hence we can learn using Proposition 8.

Symmetric juntas: A *symmetric* k-junta is a junta whose value depends only on how many of its k relevant variables are set to 1. We can learn any symmetric k-junta in time $n^{\frac{2}{3}k} \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$, which somewhat improves on our bound for arbitrary k-juntas. To prove this, we show that every symmetric function f' on k variables, other than parity and its negation, has a nonzero Fourier coefficient $\hat{f}'(S)$ for $1 \le |S| < \frac{2}{3}k$. Hence we can identify at least one relevant variable in time $n^{\frac{2}{3}k} \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$ using Proposition 8, and we can use the algorithm of Proposition 6 since the class of symmetric functions is closed under subfunctions.

To prove this claim about the Fourier coefficients of symmetric functions, first note that if f' is not balanced then by Theorem 13 it must have a nonzero Fourier coefficient of positive degree less than $\frac{2}{3}k$. Otherwise, if f' is balanced and is neither parity nor its negation, then $g := f' \oplus \mathsf{PARITY}_k$ is a symmetric nonconstant function and $\deg_{\mathbf{R}}(g) < k$; this last fact follows because the $x_1 x_2 \cdots x_k$ coefficient of g is the constant coefficient of f', and f' is balanced. By a result of von zur Gathen and Roche [24], every nonconstant symmetric function g on k variables has $\deg_{\mathbf{R}}(g) \ge k - O(k^{.548})$. Hence $\hat{g}(S) \ne 0$ for some $k - O(k^{.548}) \le |S| < k$, so $\hat{f}'([n] \setminus S) \ne 0$ and $1 \le |[k] \setminus S| \le O(k^{.548}) \le \frac{2}{3}k$.

In [24] von zur Gathen and Roche conjecture that every nonconstant symmetric Boolean function f' on k variables has $\deg_{\mathbf{R}}(f) \ge k - O(1)$. We note that if a somewhat stronger conjecture were true — that every nonconstant symmetric function has a nonzero Fourier coefficient of degree d for some $k - O(1) \le d \le k - 1$ — then using the above approach we could learn symmetric juntas in $\mathrm{poly}(2^k, n, \log(1/\delta))$ time. (The von zur Gathen/Roche conjecture does not appear to suffice since f' could conceivably have a nonzero Fourier coefficient of degree k and yet have no nonzero Fourier coefficients of degree $k - O(1)$.)

5.2 Other learning models

Blum and Langley observed [4] that if the learning algorithm is allowed to make *membership queries* for the value of the target junta at points of its choice, then any k-junta can be learned in time $\mathrm{poly}(2^k, n, \log(1/\delta)$. By drawing random examples, the learner will either determine that the function is constant or it will obtain two inputs x and y with $f(x) \ne f(y)$. In the latter case the learner then selects a path in the Hamming cube between x and y and queries f on all points in the path. The learner will thus find two neighboring points z and z' on which f has different values, so the coordinate in which z and z' differ is relevant. The learner then recurses as in Proposition 6.

While membership queries make the problem of learning juntas easy, casting the problem in the more restrictive *statistical query* learning model of Kearns (see [13] for background on this model) makes the problem provably hard. The class of k-juntas over n variables contains at least $\binom{n}{k}$ distinct parity functions, and for any two distinct parity functions $x_S \ne x_T$ we have that $\mathrm{E}[x_S x_T] = 0$. Consequently, an information-theoretic lower bound of Bshouty and Feldman [1] implies that *any* statistical query algorithm for learning k-juntas under the uniform distribution must have $q/\tau^2 \geq \binom{n}{k}$, where q is the number of statistical queries which the algorithm makes and $\tau \in (0, 11$ is the additive error tolerance required for each query. Thus, as noted earlier, the PAC model of learning from random examples seems to be the right framework for the juntas problem.

We close by observing that if the uniform distribution is replaced by a product measure in which $\Pr[x_i = \mathrm{T}] = p_i$, then for almost every choice of $(p_1, \ldots, p_n) \in [0,1]^n$, k-juntas are learnable in time $\mathrm{poly}(2^k, n, \log(1/\delta))$. In particular, we claim that for every product distribution except for a set of measure zero in $[0,1]^n$, every k-junta f has nonzero correlation with every variable on which it depends, and consequently a straightforward variant of the Fourier-based learning algorithm will identify all relevant variables in the claimed time bound. This is a consequence of the following easily verified fact:

Fact 15 *If f' is neither a single variable nor its negation and f' depends on x_i, then $\mathbf{E}_{p_1,\ldots,p_n}[f'(x)x_i]$, when viewed formally as a multivariable polynomial in $p_1, \ldots, p_n$, is a non-constant polynomial.*

Consequently, the set of points $(p_1, \ldots, p_n) \in [0,1]^n$ on which this polynomial takes value 0 has measure 0. The union of all such sets for all (finitely many) choices of i and f' still has measure 0, and the claim is proved.

6 Conclusion

A major goal for future research is to give an algorithm which runs in polynomial time for $k = \log n$ or even $k = \omega(1)$. We hope that further study of the structural properties of Boolean functions will lead to such an algorithm. Right now, the bottleneck preventing an improved runtime for our algorithm is the case of strongly balanced juntas. A. Kalai has asked the following question:

Question: *Is it true that for any Boolean function f on k bits which is strongly balanced up to size $\frac{2}{3}k$, there is a restriction fixing at most $\frac{2}{3}k$ bits under which f becomes a parity function?*

If the answer were yes, then it would be straightforward to give a learning algorithm for k-juntas running in time $n^{\frac{2}{3}k}$. (Of course, another way to get such an algorithm would be to give a quadratic algorithm for matrix multiplication!)

Finally, we close by observing that there are still several important generalizations of the k-junta problem for which no algorithm with running time better than $n^{(1-o(1))k}$ is known. Can we learn juntas under any fixed nonuniform product distribution? Can we learn ternary juntas (i.e. functions on $\{0,1,2\}^n$ with k relevant variables) under uniform? There are many directions for future work.

7 Acknowledgements

We would like to thank Adam Kalai and Yishay Mansour for helpful discussions and for telling us about [12, 18]. We also thank Oded Regev for allowing us to include his proof of Theorem 13.

References

[1] N. Bshouty andV. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research*, 2:359–395, 2002.

[2] A. Bernasconi. On a hierarchy of boolean functions hard to compute in constant depth. *Discrete Mathematics & Theoretical Computer Science*, 4:2:79–90, 2001.

[3] A. Blum. Relevant examples and relevant features: Thoughts from computational learning theory. In AAAI Fall Symposium on 'Relevance', 1994.

[4] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

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

[6] N. Bshouty, J. Jackson, and C. Tamon. More efficient PAC learning of DNF with membership queries under the uniform distribution. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 286–295, 1999.

[7] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the Nineteenth Symposium on Theory of Computing*, pages 1–6, 1987.

[8] E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky. Testing juntas. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, 2002.

[9] D. Helmbold, R. Sloan, and M. Warmuth. Learning integer lattices. *SIAM Journal on Computing*, 21(2):240–266., 1992.

[10] 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.

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

[12] A. Kalai and Y. Mansour. Personal communication, 2001.

[13] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.

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

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

[16] Y. Mansour. *Learning Boolean functions via the Fourier transform*, pages 391–424. 1994.

[17] 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.

[18] Y. Mansour. Personal communication, 2001.

[19] O. Regev. Personal communication, 2003.

[20] S. Rudich. Problem set 6 model solutions. Available at `http://www-2.cs.cmu.edu/~rudich/complexity/handouts/Probsets/Probset6/`.

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

[22] 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.

[23] 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.

[24] J. von zur Gathen and J. Roche. Polynomials with two values. *Combinatorica*, 17(3):345–362, 1997.