The Computational Complexity Column

by

Jacobo Torán

Dept. Theoretische Informatik, Universität Ulm
Oberer Eselsberg, 89069 Ulm, Germany
jacobo.toran@uni-ulm.de
http://theorie.informatik.uni-ulm.de/Personen/jt.html

In the last years significant improvements have been obtained in the area of learning in the PAC model under the uniform distribution. The Fourier transform has been a common ingredient in many of these results. Johannes Köbbler and Wolfgang Lindner present here a very interesting survey on this important but maybe not so well known area of Complexity.

Learning Boolean Functions under the uniform distribution via the Fourier Transform

Johannes Köbler * Wolfgang Lindner †

Abstract

In this article we give a brief overview on some important learning results using the Fourier Transform and highlight some open questions.

*Institut für Informatik, Humboldt-Universität zu Berlin, D-10099 Berlin, Germany. Email: koebler@informatik.hu-berlin.de

†Abt. Theoretische Informatik, Universität Ulm, Oberer Eselsberg, D-89069 Ulm, Germany. Email: lindner@informatik.uni-ulm.de

1 Introduction

Learning via the Fourier transform is a basic tool when learning in the PAC model under the uniform distribution. It has been successfully applied to various natural concept classes ranging from decision trees to constant depth circuits. The most remarkable example is Jackson's Harmonic Sieve algorithm [19] for learning DNF formulas in polynomial time with membership queries. Learning via the Fourier transform has thus provided a successful attack on the notoriously open problem of learning DNF formulas in the distribution-free PAC model without membership queries. The fastest known algorithm for this problem runs in time $2^{\tilde{O}(n^{1/3})}$ [27].

The Fourier spectrum of Boolean functions has been first applied in theoretical computer science by Kahn, Kalai and Linial [22] to answer a question posed by Ben-David and Linial [3] concerning the sensitivity of Boolean functions. The first application in computational learning is due to Linial, Mansour and Nisan [33]. The Fourier transform of a Boolean function f can be regarded as a representation of f as a linear combination over the basis of all parity functions. Each coefficient is given by the correlation between f and the corresponding basis function. Learning can then be achieved through estimating the Fourier coefficients based on a sufficiently large sample of f. For simple concept classes it is often possible to establish a certain property of the concepts in terms of their Fourier transform, which implies that each concept in the class can be approximated by paying attention to only a small part of its Fourier spectrum. The learning problem is then reduced to estimating the Fourier coefficients in the important part of the spectrum.

In this column we concentrate on learning functions with properties which can be expressed in terms of their Fourier spectrum. In a first part we review some basic algorithms, starting with the ubiquitous low-degree algorithm of Linial et al. [33]. Then we present the KM-algorithm of Kushilevitz and Mansour [30] for finding all significant Fourier coefficients. Next we describe Jackson's Harmonic Sieve algorithm which combines the KM-algorithm with boosting, and finally we present the more recent algorithm of Jackson et al. [20] which can be regarded as a simplified Harmonic Sieve obtained by replacing the KM-algorithm by an exhaustive search.

In a second part we concentrate on learning classes of monotone functions based on sensitivity arguments, including Bshouty and Tamon's work on monotone functions [9], Servedio's learnability result for monotone DNF [40], and the very recent learning algorithm of O'Donnell and Servedio for monotone decision trees [39] based on a sensitivity result due to Friedgut [14].

We do not make any attempt to be comprehensive. However, we do hope to convince the reader that learning via the Fourier transform is a true success

story with no end in sight.

2 Notation and basic facts

In this section we fix the notation and give formal definitions for some of the concepts used in this paper. Further we state some basic facts for further reference.

Fourier Transform

We are interested in learning Boolean concept classes, i.e., each concept can be represented as a Boolean function $f\colon \{0,1\}^n \to \{true, false\}$. We denote the class of all Boolean functions of arity n by B_n. If we identify *true* with 1 and *false* with 0, then B_n forms a vector space of dimension 2^n over the field $\mathbb{F}_2$. The regular basis for B_n consists of the 2^n functions $term_a(x)$, $a \in \{0,1\}^n$, mapping x to 1, if $x = a$, and to 0 otherwise. Clearly, any Boolean function $f \in B_n$ has a unique representation $f = \sum_{a\in\{0,1\}^n} c_a term_a$ with coefficients $c_a = f(a)$. For many applications it is important to have a basis with the following useful properties.

(a) "Simple" functions should have small representations. E.g., if the value of f is already determined by a small subset of its variables, then many coefficients should vanish.

(b) Transforming f to a "similar" function f' (e.g., $f'(x) = f(x \oplus a)$) should allow for an easy conversion of the coefficient representations of the two functions.

By embedding B_n into a richer structure, namely the vector space $\mathbb{R}^{\{0,1\}^n} = \{f\colon \{0,1\}^n \to \mathbb{R}\}$ over $\mathbb{R}$, these properties can be easily achieved. In fact, if we require from our basis $\{\chi_a \mid a \in \{0,1\}^n\}$ that it contains the functions $\chi_{e_i}(x) = (-1)^{x_i}$ (in order to fulfill property (a)) and that all basis functions χ_a have the property

$$\chi_a(x \oplus b) = \chi_a(x)\chi_a(b), \tag{1}$$

then also property (b) is fulfilled. This is easy to see, since for any function f with the representation $f(x) = \sum_{a\in\{0,1\}^n} c_a\chi_a(x)$ it follows that the function $f_{\oplus b}(x) = f(x \oplus b)$ has the representation

$$f_{\oplus b}(x) = f(x \oplus b) = \sum_{a\in\{0,1\}^n} c_a\chi_a(x \oplus b) = \sum_{a\in\{0,1\}^n} c_a\chi_a(x)\chi_a(b), \tag{2}$$

implying that the coefficients c'_a of $f_{\oplus b}$ can be written as $c'_a = \chi_a(b)c_a$. Property (1) requires that the base functions are homomorphisms from the Abelian group $(\{0,1\}^n, \oplus)$ to the multiplicative group $(\mathbb{R}^*, \cdot)$ of non-zero real numbers. It is easy to show that exactly the *parity functions*

$$\chi_a(x) = (-1)^{\sum_{i=1}^n a_i x_i}, \ a \in \{0,1\}^n$$

have this property and that these functions indeed form a basis for the vector space $\mathbb{R}^{\{0,1\}^n}$. By identifying the vector $a \in \{0,1\}^n$ with the set $S = \{i \in [n] \mid a_i = 1\}$, where we use $[n]$ to denote the set $\{1, \ldots, n\}$, we get the more convenient representation

$$\chi_S(x) = (-1)^{\sum_{i \in S} x_i}, \ S \subseteq [n].$$

More generally it can be shown that for any finite Abelian group G the class of all homomorphisms from G to the multiplicative group $(\mathbb{C}^*, \cdot)$ of non-zero complex numbers form a group F (under multiplication, with the constant 1 function as neutral element) that is isomorphic to G. Moreover, the functions in F form an orthogonal basis for the vector space $\mathbb{C}^G$ over $\mathbb{C}$ of all functions $f \colon G \to \mathbb{C}$. The elements of F are called the *characters* of G and F is called the *Fourier basis* for $\mathbb{C}^G$.

In the case $G = (\{0,1\}^n, \oplus)$ all functions χ_S in the Fourier basis are real-valued and hence also form a basis for the subspace $\mathbb{R}^{\{0,1\}^n}$ of all functions $f \colon \{0,1\}^n \to \mathbb{R}$. In fact, using the natural notion of *inner-product*

$$\langle f, g \rangle = 2^{-n} \sum_{x \in \{0,1\}^n} f(x)g(x) = E[f(x)g(x)],$$

it is easy to verify that the Fourier basis $F = \{\chi_S \mid S \subseteq [n]\}$ forms an orthonormal system of functions, i.e.

$$\langle \chi_S, \chi_T \rangle = E[\chi_S(x)\chi_T(x)] = E[\chi_{S \Delta T}(x)] = \begin{cases} 1, & S = T, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

implying that all parity functions have unit norm $\|\chi_S\| = 1$, where the *norm* of a function f is induced by the inner-product using the rule

$$\|f\| = \sqrt{\langle f, f \rangle} = \sqrt{2^{-n} \sum_x f(x)^2}.$$

More generally, for $p > 0$ the *p-norm* is defined as

$$\|f\|_p = E[|f(x)|^p]^{1/p} = \left(2^{-n} \sum_x |f(x)|^p\right)^{1/p}$$

and the ∞-*norm* is

$$\|f\|_\infty = \max_x |f(x)|.$$

Notice that the norm induced by the inner-product coincides with the 2-norm. We remark for further use that for $0 < p \leq q \leq \infty$, $\|f\|_p \leq \|f\|_q$. Since the Fourier basis is orthonormal, the *Fourier coefficients* $\hat{f}(S)$ in the *Fourier expansion*

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

of f can be written as the inner-product $\hat{f}(S) = \langle f, \chi_S \rangle$ of f and the basis functions χ_S. The function $\hat{f}\colon 2^{[n]} \mapsto \mathbb{R}$ mapping each *frequency* $S \subseteq [n]$ to the corresponding Fourier coefficient $\hat{f}(S)$ is called the *Fourier transform* of f. A crucial property of $\hat{f}$ is that if f does not depend on a variable x_j, then all coefficients $\hat{f}(S)$ with $j \in S$ vanish (cf. property (a)). Using (3) it follows for any functions $f, g\colon \{0,1\}^n \to \mathbb{R}$ that

$$\begin{aligned} E[fg] &= E\big[\big(\textstyle\sum_S \hat{f}(S)\chi_S\big)\big(\sum_T \hat{g}(T)\chi_T\big)\big] = \textstyle\sum_{S,T} \hat{f}(S)\hat{g}(T)E[\chi_S\chi_T] \\ &= \textstyle\sum_S \hat{f}(S)\hat{g}(S). \end{aligned}$$

Hence, a further consequence of the orthonormality of the Fourier basis is *Parseval's identity* stating that

$$\|f\|^2 = \sum_S \hat{f}(S)^2.$$

If we identify *true* with -1 and *false* with 1, i.e., if we let $B_n = \{f\colon \{0,1\}^n \to \{-1,1\}\}$, then Parseval's identity implies that for a Boolean function $f \in B_n$, the squares $\hat{f}(S)^2$ of the Fourier coefficients of f sum up to 1 and thus induce a probability distribution on the frequencies. For a collection $G \subseteq 2^{[n]}$ of frequencies we refer to the "probability" $\sum_{S \in G} \hat{f}(S)^2$ of G as the *weight of the Fourier spectrum of f on G* or simply as *f's Fourier weight on G*.

Clearly, by the linearity of the vector space $\mathbb{R}^{\{0,1\}^n}$, the Fourier transform of $f + g$ is obtained as the sum $\widehat{f+g} = \hat{f} + \hat{g}$ of the Fourier transforms of f and g. Hence, if g is obtained from a function f by removing all coefficients outside G, i.e., $g = \sum_{S \in G} \hat{f}(S)\chi_S$, then Parseval's identity implies

$$\|f - g\|^2 = \sum_S \|\widehat{f-g}(S)\|^2 = \sum_S \|\hat{f}(S) - \hat{g}(S)\|^2 = \sum_{S \notin G} \hat{f}(S)^2.$$

Notice that g need not be Boolean. But it is easy to see that we can approximate any real-valued function g by the Boolean function

$$\operatorname{sgn}(g(x)) = \begin{cases} 1, & g(x) \geq 0 \\ -1, & \text{otherwise,} \end{cases}$$

where

$$\Pr[f(x) \neq \operatorname{sgn}(g(x))] \leq \|f - g\|^2.$$

We close this subsection with a useful inequality due to Beckner and Bonami [2, 6]. For any real $\delta \in [0,1]$ let T_δ be the linear operator mapping a function f to the function $T_\delta(f) = \sum_S \delta^{|S|} \hat{f}(S)\chi_S$, i.e., T_δ in some sense reduces the Fourier weight of the high frequencies (where high means that $|S|$ is large). The Beckner-Bonami inequality states that $\|T_\delta(f)\| \leq \|f\|_{1+\delta^2}$. Since for any function f taking only values in the range $\{-1, 0, 1\}$ we have $\|f\|_2^2 = \|f\|_p^p$ for all $p > 0$, the Beckner-Bonami inequality implies for such functions that

$$\|T_\delta(f)\|^2 \leq \|f\|_{1+\delta^2}^2 = \left(\|f\|_{1+\delta^2}^{1+\delta^2}\right)^{2/(1+\delta^2)} = (\|f\|^2)^{2/(1+\delta^2)}. \tag{4}$$

For further information and background on the Fourier transform of Boolean functions we refer to [12, 42, 38, 10, 32].

Learning

We consider the well-known distribution-specific variant of Valiant's Probably Approximate Correct (PAC) learning model [43]. Let f be a Boolean function and D be a distribution on the instance space $\{0,1\}^n$. For $\varepsilon > 0$ we say that a Boolean function h is an (ε, D)-*approximation* of f if $\Pr_D[h(x) \neq f(x)] \leq \varepsilon$, where x is chosen according to D. We use $EX(f, D)$ to denote an oracle which, when invoked, returns a random *labeled example* $(x, f(x))$ where x is chosen according to D. A *concept class* $C \subseteq B_n$ is *learnable with respect to* D if there is a randomized *learning algorithm* A which, for all *targets* $f \in C$ and parameters $\varepsilon, \delta > 0$, when given inputs ε, δ and oracle access to $EX(f, D)$, $A(\varepsilon, \delta, EX(f, D))$ outputs with probability $1 - \delta$ a Boolean function h which is an (ε, D)-approximation of f. Here, the probability is taken over the random choices of A and the random labeled examples returned by $EX(f, D)$, where we assume that the random examples are selected independently from each other and from the random choices of A.

We also consider *weak learnability*, where the hypothesis h produced by the learning algorithm A is only slightly more accurate than random guessing. For $\gamma > 0$ we say that a Boolean function h is a *weak* (γ, D)-*approximation* of f if $\Pr_D[h(x) \neq f(x)] \leq 1/2 - \gamma$. Now a concept class C is *weakly learnable with advantage* γ *and with respect to* D if there is a learning algorithm A such that for all $f \in C$ and parameters $\gamma, \delta > 0$, $A(\gamma, \delta, EX(f, D))$ outputs with probability $1 - \delta$ a weak (γ, D)-approximation of f.

If the learning algorithm A requires direct access to the oracle f rather than $EX(f, D)$, then we say that C is learnable *with membership queries*.

If D is the uniform distribution, we generally omit the reference to D. For more background on learning we refer to [24].

Prominent examples of natural concept classes are *decision trees* with a single variable at each inner node, *DNF formulas*, and *constant depth* AC^0 *circuits*. Note that every decision tree with m nodes can be transformed into a DNF formula with at most m terms, and that every m-term DNF formula can be regarded as a constant depth AC^0 circuit of size $m+1$ and depth 2. For background on circuit complexity we refer the reader to a standard textbook like [46]. For a discussion of the Fourier transform of these concept classes we refer to the excellent overview of Mansour [35]. A more recent overview is provided by Jackson and Tamon's tutorial [21].

Probability Theory

For later use we state the following result to which we refer to as the Chernoff-Hoeffding bound. Let $X_1, \ldots, X_m$ be independent random variables taking values in the real interval $[a, b]$ and having expectation $E[X_i] = \mu$. Then for any $\lambda > 0$, with probability at most $2e^{-2\lambda^2 m/(b-a)^2}$, the additive error of the estimate for μ obtained by taking the arithmetic mean of m observations of the random variables $X_1, \ldots X_m$ is greater than or equal to λ,

$$\Pr\left[\left|\tfrac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| \geq \lambda\right] \leq 2e^{-2\lambda^2 m/(b-a)^2}.$$

In other words, with confidence $1 - 2e^{-2\lambda^2 m/(b-a)^2}$, the arithmetic mean provides a λ-accurate estimate for μ.

Further Notation

We measure the closeness of two real-valued functions f and g in terms of the 2-norm squared of the difference of f and g. That is, we say that f and g are ε-*close* if $\|f - g\|^2 \leq \varepsilon$. Note that for Boolean functions f and g it holds that $\|f - g\|^2 = 4\Pr[f(x) \neq g(x)]$ and hence, f and g are ε-close if and only if f is an $(\varepsilon/4)$-approximation of g.

3 Basic algorithms

In this section we review some basic algorithms for learning via the Fourier transform. We start with the low-degree algorithm of Linial et al. [33] and its application to AC^0 circuits. Next we present the KM-algorithm of Kushilevitz and Mansour [30] which can be used to learn decision trees in polynomial time with membership queries and its application to DNF formulas due to

Mansour [36]. Then we describe Jackson's Harmonic Sieve [20] which learns DNF formulas in polynomial time with membership queries. Finally we brievly review the simple exhaustive search algorithm of Jackson, Klivans and Servedio [20] which can be applied to majorities over AC^0 circuits.

3.1 The low-degree algorithm

The first application of Fourier analysis in computational learning theory is due to Linial et al. [33]. The learning result is based on an upper bound for the 2-norm of $\hat{f}$ restricted to high frequencies, where the bound depends on the circuit complexity of f. So suppose that f is a Boolean function satisfying

$$\sum_{|S|>t} \hat{f}(S)^2 \leq \varepsilon.$$

For the real-valued function $g = \sum_{|S|\leq t} \hat{f}(S)\chi_S$ it follows by Parseval's identity that

$$\|f - g\|^2 = \sum_{|S|>t} \hat{f}(S)^2 \leq \varepsilon,$$

which means that f can be ε-approximated by fading out the high frequencies S with $|S| > t$. This observation reduces the learning problem for f to the problem of computing the Fourier coefficients $\hat{f}(S)$ for all low-order frequencies S with $|S| \leq t$.

This task can be accomplished by the *low-degree algorithm* which is the most fundamental algorithm in the context of learning via the Fourier transform. The algorithm is presented in Figure 1. The simple but crucial observation is that each coefficient $\hat{f}(S)$ can be expressed as the expectation $E[f(x)\chi_S(x)]$, where x is chosen uniformly at random. Thus each coefficient $\hat{f}(S)$ can be accurately estimated with high confidence by drawing a sufficiently large sample from $EX(f)$. More specifically, the low-degree algorithm draws a polynomial number in n^t, $1/\varepsilon$ and $\log 1/\delta$ of labeled examples from $EX(f)$ and computes for each S of size at most t an empirical estimate a_S for $\hat{f}(S)$. By applying the Chernoff-Hoeffding bound it follows that with probability $1 - \delta$, each estimate a_S is within additive error $\lambda = (\varepsilon/n^t)^{1/2}$ from its expected value $\hat{f}(S)$. In this case the approximation $g = \sum_{|S|\leq t} a_S\chi_S$ satisfies

$$\begin{aligned}\|f - g\|^2 = \sum_S (\hat{f}(S) - \hat{g}(S))^2 &= \sum_{|S|\leq t} (\hat{f}(S) - a_S)^2 + \sum_{|S|>t} \hat{f}(S)^2 \\ &\leq n^t\lambda^2 + \varepsilon = 2\varepsilon,\end{aligned}$$

input: frequency bound t, accuracy ε, confidence δ and access to $EX(f)$
output: a Boolean function h approximating f

1. request $m = \frac{2n^t}{\varepsilon} \ln(\frac{2n^t}{\delta})$ labeled examples $(x_i, f(x_i))$ from $EX(f)$
2. for each $S \subseteq [n]$ with $|S| \leq t$ compute $a_S = \frac{1}{m} \sum_{i=1}^{m} f(x_i)\chi_S(x_i)$
3. output $h = \mathrm{sgn}(\sum_{|S| \leq t} a_S \chi_S)$

Figure 1: The low-degree algorithm

and since $\Pr[\mathrm{sgn}(g(x)) \neq f(x)] \leq \|f - g\|^2$, it follows that the output hypothesis $h = \mathrm{sgn}(g)$ is an $(\varepsilon/2)$-approximation of f. The running-time is dominated by the sample size and thus polynomial in n^t, $1/\varepsilon$ and $\log(1/\delta)$.

Theorem 1. *For any Boolean function f satisfying*

$$\sum_{|S|>t} \hat{f}(S)^2 \leq \varepsilon,$$

the low-degree algorithm on inputs t, ε, δ and access to $EX(f)$ outputs with probability $1 - \delta$ an $O(\varepsilon)$-approximation of f in time $\mathrm{poly}(n^t, 1/\varepsilon, \log(1/\delta))$.

Let us remark for further reference, that the low-degree algorithm can be easily generalized to Boolean functions f satisfying $\sum_{S \in G} \hat{f}(S)^2 \leq \varepsilon$ for an arbitrary collection $G \subseteq 2^{[n]}$ of frequencies, provided that G is explicitly given as part of the input. In this case, the running-time is $\mathrm{poly}(n, |G|, 1/\varepsilon, \log(1/\delta))$.

Based on Hastad's Switching Lemma [17], Linial et al. [33] showed that for any Boolean function f which is computable by an AC^0 circuit of depth d and size M it holds that

$$\sum_{|S|>t} \hat{f}(S)^2 \leq 2M2^{-t^{1/d}/20}.$$

Applying the low-degree algorithm with $t = O(\log(M/\varepsilon)^d)$ immediately yields the following learning result.

Corollary 2. [33] *The class of AC^0 circuits of depth d and size M over n variables is learnable in time*

$$\mathrm{poly}(n^{\log(M/\varepsilon)^d}, \log(1/\delta)).$$

Since an m-term DNF formula is computable by a circuit of size $m+1$ and depth 2 it further follows that m-term DNF formulas are learnable in time $\mathrm{poly}(n^{\log(m/\varepsilon)^2}, \log(1/\delta))$.

By a result due to Kharitonov [25], Corollary 2 cannot be significantly improved under a plausible cryptographic assumption. However, as we will see in Section 3.4, the exponent d can be reduced to $d-1$. This will imply that m-term DNF formulas are in fact learnable in time $\mathrm{poly}(n^{\log(m/\varepsilon)}, \log(1/\delta))$.

3.2 The KM-Algorithm

For sufficiently simple functions it is sometimes possible to bound the 1-norm of the Fourier transform. A nice example provide decision trees, for which the 1-norm of the Fourier transform can be bounded by the number of nodes in the tree by fairly elementary methods [30]. So suppose that f is a Boolean function satisfying

$$\|\hat{f}\|_1 \leq k.$$

Then clearly,

$$\sum_{|\hat{f}(S)| \leq \varepsilon/k} \hat{f}(S)^2 \;\leq\; (\varepsilon/k) \sum_{|\hat{f}(S)| \leq \varepsilon/k} \hat{f}(S) \;\leq\; \varepsilon,$$

which means that f can be ε-approximated by ignoring the small Fourier coefficients having absolute value at most ε/k. This observation reduces the learning problem for f to the problem of finding all frequencies S whose Fourier coefficients are larger in absolute value than some given threshold θ.

This problem can be solved by an algorithm of Goldreich and Levin [16] which is a key ingredient in their proof that parity functions are hard-core predicates for one-way functions. The algorithm has been first applied in the learning setting by Kushilevitz and Mansour [30]. The idea behind the algorithm is best described in terms of the recursive procedure Coef given in Figure 2, which can be regarded as a depth-first search on the binary tree of depth n. Here, each node is given by its level $k \in [n]$ and a set $S \subseteq [1,k]$, where we use the notation $[m,n]$ to describe the set $\{m,\ldots,n\}$. In each node (S,k) we consider the sum

$$C(S,k) = \sum_{T \subseteq [k+1,n]} \hat{f}(S \cup T)^2.$$

If $k = n$, then Coef has reached at a leaf of the tree, and since in this case $C(S,k) = \hat{f}(S)^2$, the procedure returns S if and only if $C(S,k) \geq \theta^2$. If (S,k) is an inner node with $C(S,k) < \theta^2$, then there is no set $T \subseteq [k+1,n]$

Coef(S,k):

1. **if** $k = n$ **and** $C(S,k) \geq \theta^2$ **then return**$(\{S\})$
2. **if** $k < n$ **and** $C(S,k) \geq \theta^2$ **then**
   return$(\text{Coef}(S, k+1) \cup \text{Coef}(S \cup \{k+1\}, k+1))$
3. **return**$(\emptyset)$

Figure 2: The recursive procedure Coef

with $|\hat{f}(S \cup T)| \geq \theta$. Thus there is no need to further explore the subtree of this node. Otherwise, Coef recursively continues the search with the two children $(S, k+1)$ and $(S \cup \{k+1\}, k+1)$ of (S,k). Invoking Coef with the root node $(\emptyset, 0)$, the procedure returns the collection of all frequencies corresponding to Fourier coefficients with an absolute value greater than θ.

Concerning the number of recursive calls performed by Coef, first note that by Parseval's identity,

$$\sum_{S \subseteq [1,k]} C(S,k) = \sum_{S \subseteq [n]} f(S)^2 = \|f\|^2 = 1,$$

implying that in each level k there are at most $1/\theta^2$ nodes (S,k) with $C(S,k) \geq \theta^2$. Thus, the total number of recursive calls can be bounded by n/θ^2. The main algorithmic difficulty, however, is the computation of the sums $C(S,k)$. This can be solved by expressing $C(S,k)$ as a nested expectation. More precisely, consider the function $g\colon \{0,1\}^{n-k} \to \mathbb{R}$ which maps a suffix $x \in \{0,1\}^{n-k}$ to the expectation $E_y[f(yx)\chi_S(y)]$ for a uniformly chosen prefix $y \in \{0,1\}^k$. For any prefix $y \in \{0,1\}^k$ and $T \subseteq [k+1, n]$ we have $\chi_{S \cup T}(z) = \chi_{S \Delta T}(z) = \chi_S(y)\chi_T(x)$, where $z = yx$. Hence,

$$\hat{f}(S \cup T) = E_z[f(z)\chi_{S \cup T}(z)] = E_x\big[E_y[f(yx)\chi_S(y)\chi_T(x)]\big] = E_x[g(x)\chi_T(x)],$$

implying that $\hat{f}(S \cup T) = \hat{g}(T)$. Now, by using Parseval's identity, we can express $C(S,k)$ as

$$C(S,k) = \sum_{T \subseteq [k+1,n]} \hat{g}(T)^2 = E_x[g(x)^2] = E_x\big[E_y[f(yx)\chi_S(y)]^2\big]$$

for uniformly chosen $x \in \{0,1\}^{n-k}$ and $y \in \{0,1\}^k$. By applying the Chernoff-Hoeffding bound, this means that we can estimate $C(S,k)$ by first estimating the inner expectation for a small number of random suffixes x and then estimate the outer expectation as the average sum of the squared

estimates for the inner expectation. For each given suffix x, the estimate for the inner expectation can be obtained from a small number of values $f(yx)$ for random prefixes y, which can be obtained by asking the oracle f. By using sufficiently accurate estimates of $C(S,k)$ it can be shown that with high probability, the modified search still runs in polynomial time and returns a set containing all frequencies S with $|\hat{f}(S)| \geq \theta$. Furthermore, each frequency S in the set satisfies $|\hat{f}(S)| = \Omega(\theta)$, which by Parseval's identity implies that the number of frequencies in the returned collection is at most $O(1/\theta^2)$.

Lemma 3. [30] *There is a randomized algorithm A such that for each Boolean function f and threshold $\theta > 0$, $A(\theta, \delta, f)$ outputs with probability $1-\delta$ a collection $G \subseteq 2^{[n]}$ of size $O(1/\theta^2)$ containing all frequencies S with $|\hat{f}(S)| \geq \theta$. A runs in time* $\mathrm{poly}(n, 1/\theta, \log(1/\delta))$.

As observed by Jackson [19] the algorithm of Lemma 3 can also be applied to real-valued functions $f\colon \{0,1\}^n \to \mathbb{R}$. In this case, the number of recursive calls of the procedure Coef is bounded by $\|f\|n/\theta^2$ rather than n/θ^2. Further notice that the confidence in the Chernoff-Hoeffding bound depends on the range of the random variables whose mean we want to estimate. In case f is real-valued, the range of these random variables is $[-\|f\|_\infty^2, \|f\|_\infty^2]$ rather than $[-1,1]$, implying that the number of labeled examples from f needed to estimate $C(S,k)$ additionally depends on the parameter $\|f\|_\infty$. Since $\|f\| \leq \|f\|_\infty$, the running-time becomes polynomial in n, $1/\theta$, $\log(1/\delta)$ and $\|f\|_\infty$. Let us state this extension to real-valued functions for later use.

Lemma 4. [19] *There is a randomized algorithm A such that for each function $f\colon \{0,1\}^n \to \mathbb{R}$ satisfying $\|f\|_\infty \leq k$ and threshold $\theta > 0$, $A(\theta, \delta, k, f)$ outputs with probability $1-\delta$ a collection $G \subseteq 2^{[n]}$ containing all frequencies S with $|\hat{f}(S)| \geq \theta$. A runs in time* $\mathrm{poly}(n, k, 1/\theta, \log(1/\delta))$.

Coming back to Boolean functions $f \in B_n$ satisfying $\|\hat{f}\|_1 \leq k$, we can use the algorithm of Lemma 3 to find a small collection G containing all frequencies S for which $|\hat{f}(S)| \geq \varepsilon/k$ in time $\mathrm{poly}(n, k, 1/\varepsilon, \log(1/\delta))$, and then use the generalized low-degree algorithm on G to output an $O(\varepsilon)$-approximation of f. This algorithm is known as the *Kushilevitz-Mansour algorithm*, or simply *KM-algorithm*, and we state its properties in the following theorem.

Theorem 5. [30] *For each Boolean function f satisfying $\|\hat{f}\|_1 \leq k$, the KM-algorithm, given inputs k, ε, δ and oracle access to f, outputs with probability $1-\delta$ an $O(\varepsilon)$-approximation of f in time* $\mathrm{poly}(n, k, 1/\varepsilon, \log(1/\delta))$.

As already mentioned at the beginning of this section, any Boolean function f computable by a decision tree with m nodes satisfies $\|\hat{f}\|_1 \leq m$. Thus,

the KM-algorithm learns decision trees in polynomial time, although, a disadvantage of this result is that membership queries are needed.

Corollary 6. [30] *The class of decision trees with m nodes is learnable with membership queries in time* $\mathrm{poly}(n, m, 1/\varepsilon, \log(1/\delta))$.

For some applications, it is more convenient to consider the *sparseness* of a Boolean function f. This property is closely related to the 1-norm of $\hat{f}$. We say that a function f is k-*sparse*, if the support $\{S \subseteq [n] \mid \hat{f}(S) \neq 0\}$ of $\hat{f}$ has size at most k. Clearly, if f is k-sparse then $\|\hat{f}\|_1 \leq k$. On the other hand, if $\|\hat{f}\|_1 \leq k$, then f is ε-close to the function $g = \sum_{|\hat{f}(S)|>\varepsilon/k} \hat{f}(S)\chi_S$. By Parseval's identity, the number of frequencies S with $|\hat{f}(S)| > \varepsilon/k$ is less than k^2/ε^2. Hence it follows that g is (k^2/ε^2)-sparse. We call a Boolean function (ε, k)-*sparse* if it is ε-close to a k-sparse function. It is not hard to show (see [30]) that for any (ε, k)-sparse function f,

$$\sum_{|\hat{f}(S)|\leq\varepsilon/k} \hat{f}(S)^2 \leq \varepsilon + \varepsilon^2/k.$$

Thus, the KM-algorithm is applicable to (ε, k)-sparse Boolean functions.

Corollary 7. [30] *For each (ε, k)-sparse function f, the KM-algorithm given inputs k, ε, δ and oracle access to f outputs an $O(\varepsilon)$-approximation of f in time* $\mathrm{poly}(n, k, 1/\varepsilon, \log(1/\delta))$.

Mansour [36] showed that each DNF with terms of size at most d is (ε, k)-sparse for $k = d^{O(d\log(1/\varepsilon))}$. By an argument attributed to Warmuth in [44], every m-term DNF is ε-close to a DNF with terms of size at most $\log(m/\varepsilon)$ (by simply ignoring all terms of size larger than $\log(m/\varepsilon)$). Hence, an m-term DNF f is ε-close to a (ε, k)-sparse function for $k = (\log(m/\varepsilon))^{O(\log(m/\varepsilon)\log(1/\varepsilon))} = (m/\varepsilon)^{O(\log\log(m/\varepsilon)\log(1/\varepsilon))}$, which by the triangle inequality implies that f itself is $(O(\varepsilon), k)$-sparse.

Corollary 8. [36] *The class of m-term DNF formulas is learnable with membership queries in time* $\mathrm{poly}(n, (m/\varepsilon)^{\log\log(m/\varepsilon)\log(1/\varepsilon)}, 1/\varepsilon, \log(1/\delta))$.

3.3 The Harmonic Sieve

In the last section we described a quasipolynomial-time learning algorithm for DNF formulas based on the sparseness of these functions. Another property of the Fourier transform of m-term DNF is that the ∞-norm can be lower bounded in terms of m [4]. This property provides the basis for Jackson's

celebrated polynomial-time learning algorithm for DNF formulas which we present in this section. So, for a Boolean function f satisfying

$$\|\hat{f}\|_\infty \geq \gamma,$$

assume that S is a frequency with $\hat{f}(S) \geq \gamma$. Expressing the coefficient $\hat{f}(S)$ in terms of the probability that $f(x) \neq \chi_S(x)$,

$$\hat{f}(S) = E[f(x)\chi_S(x)] = 1 - 2\Pr[f(x) \neq \chi_S(x)],$$

it follows that

$$\Pr[f(x) \neq \chi_S(x)] \leq \frac{1 - \hat{f}(S)}{2}.$$

This means that the function f can be weakly $(\gamma/2)$-approximated by a single parity function χ_S or its negation $-\chi_S$, where we use χ_S if $\hat{f}(S)$ is positive, and its negation otherwise. The corresponding frequency S can be found by the KM-algorithm with high probability in time $\mathrm{poly}(n, 1/\gamma)$, and the sign of $\hat{f}(S)$ can be easily determined by estimating $\hat{f}(S)$ within additive error less than γ. Thus, for any $\gamma > 0$ and every Boolean function f satisfying $\|\hat{f}\|_\infty \geq \gamma$ we can produce with high probability a weak $\Omega(\gamma)$-approximation of f in time $\mathrm{poly}(n, 1/\gamma)$, provided that we have access to the oracle f.

This reasoning can be generalized to arbitrary distributions D by using the crucial observation that the correlation $E_D[f\chi_S]$ between f and a parity χ_S with respect to D can be expressed as the correlation $E[f_D(x)\chi_S(x)]$ between the real-valued function $f_D(x) = 2^n D(x) f(x)$ and χ_S with respect to the uniform distribution. Thus, $E_D[f(x)\chi_S(x)] = E[f_D(x)\chi_S(x)]$ coincides with the Fourier coefficient $\hat{f}_D(S)$ of the function f_D, implying that

$$\Pr_D[f(x) \neq \chi_S(x)] \leq \frac{1 - \hat{f}_D(S)}{2}.$$

If we now assume that $\|\hat{f}_D\|_\infty \geq \gamma$ (rather than $\|\hat{f}\|_\infty \geq \gamma$), then f can be weakly $(\gamma/2, D)$-approximated by a single parity function χ_S or its negation. Further, on input γ, the corresponding frequency S with $|\hat{f}_D(S)| = \Omega(\gamma)$ can be found by the algorithm of Lemma 4 using oracle access to f_D.

Lemma 9. *There is a randomized algorithm A such that for each distribution D and Boolean function f satisfying $\|\hat{f}_D\|_\infty \geq \gamma$ and $\|f_D\|_\infty \leq k$, $A(k, \gamma, \delta, f_D)$ outputs with probability $1 - \delta$ a weak $(\Omega(\gamma), D)$-approximation of f in time* $\mathrm{poly}(n, k, 1/\gamma, \log(1/\delta))$.

Boosting [41] is a well-known technique to transform a weak learner into a strong learner. The technique can be most easily described in the

input: $\gamma, \varepsilon > 0$ and a weak learning algorithm A
output: a Boolean function h approximating f

1. $i \leftarrow 0$;
2. **while** $|M_i| > \varepsilon 2^n$ **do**
   run A to produce a weak (γ, D_i)-approximation h_i;
   $i \leftarrow i + 1$;
3. **return** $h = \mathrm{sgn}(\sum_{j=0}^{i} h_j(x))$

Figure 3: The IHA-boosting algorithm

distribution-free setting, where we assume that the weak learner produces a weak (γ, D)-approximation of the target f for any distribution D. With respect to some fixed but unknown target distribution D, the boosting algorithm runs the weak learner several times with respect to different distributions D_i, which forces the weak learner to perform well on different regions of the instance space. The resulting weak (γ, D_i)-approximations are then combined in some way to produce a strong (ε, D)-approximation of f. Obviously, the oracle access to f required by the boosting algorithm depends on how the weak learner accesses the oracle. If the weak learner asks membership queries, then also the boosting algorithm needs to ask membership queries. If the weak learner needs only access to $EX(f, D_i)$, then it is usually possible to apply a filtering technique in order to simulate the $EX(f, D_i)$ oracle by asking queries to $EX(f, D)$. This means that each example $(x, f(x))$ drawn from $EX(f, D)$ is discarded by the boosting algorithm with a certain probability depending on $(x, f(x))$ and D_i, and only the remaining examples are passed on to the weak learner.

There are several boosting strategies which mainly differ in how the distributions D_i are defined, and in how the final hypothesis is obtained from the weak hypotheses. The *IHA-boosting algorithm* (see Figure 3) uses a particularly well-suited boosting strategy which is based on a construction of a hard-core distribution due to Impagliazzo [18]. The boosting ability of this construction has been pointed out by Klivans and Servedio [26]. In order to achieve strong learning with respect to the uniform target distribution, the IHA-boosting algorithm uses the following distributions D_i. Suppose that the weak learning algorithm has already produced the hypotheses $h_0, \ldots, h_{i-1}$ for some $i \geq 0$, and let $h = \mathrm{sgn}(\sum_{j=0}^{i-1} h_j(x))$ denote the majority vote of these

hypotheses. First consider the *margin*

$$N_i(x) = f(x) \sum_{j=0}^{i-1} h_j(x),$$

by which h agrees with the target f on an instance x. Note that h disagrees with f on x only if $N_i(x)$ is negative. Next we define a *measure* M_i on the instance space $\{0,1\}^n$ which assigns weight 0 to the instances with large margin, weight 1 to the instances with negative margin, and intermediate weights to instances with non-negative but small margin. More precisely,

$$M_i(x) = \begin{cases} 0, & N_i(x) \geq 1/\gamma, \\ 1, & N_i(x) \leq 0, \\ 1 - \gamma N_i(x), & \text{otherwise.} \end{cases}$$

Notice that $M_0(x) = 1$ for all x. The distribution D_i is now obtained by standardizing the measure M_i by the *weight* $|M_i| = \sum_x M_i(x)$ of M_i,

$$D_i(x) = \frac{M_i(x)}{|M_i|}.$$

Whenever the IHA-boosting algorithm is going to run the weak learner A, it first checks whether the measure M_i satisfies $|M_i| \leq \varepsilon 2^n$. Observe that the current majority vote h disagrees with f on x only if $N_i(x) \geq 0$. In this case $M_i(x) = 1$ and hence the approximation error of h can be bounded by

$$\Pr[h(x) \neq f(x)] \leq 2^{-n} \sum_x M_i(x) = 2^{-n}|M_i|.$$

Thus, the condition $|M_i| \leq \varepsilon 2^n$ guarantees that the boosting algorithm has found the desired ε-approximation of f.

Impagliazzo [18] showed that the abort condition $|M_i| \leq \varepsilon 2^n$ is met after at most $O(1/\gamma^2\varepsilon^2)$ runs of A. Since for all x we have that $D_i(x) \leq 1/|M_i|$, the ∞-norm of the distributions D_i is bounded by $1/|M_i|$ and hence the weak learner A is run only on distributions D_i satisfying $\|2^n D_i\|_\infty \leq 1/\varepsilon$.

An algorithmic difficulty is the computation of the exponentially large sum $|M_i|$ which is required to check the abort condition. This difficulty can be overcome by first expressing $2^{-n}|M_i|$ as the expected value $E[M_i(x)]$ for a uniformly chosen x. Then we only have to observe that a random example $(x, M_i(x))$ can be easily obtained from a random example $(x, f(x))$. Furthermore, $M_i(x)$ takes only values in the range $[0,1]$. Hence, with high probability we can get an accurate estimate for $2^{-n}|M_i|$ by drawing a small

sample from $EX(f)$. Now, by using an estimate rather than the exact value of $2^{-n}|M_i|$, it is easy to adjust the abort condition so that (1) the output h is still an ε-approximation of f and (2) the modified abort condition can still be met after at most $O(1/\gamma^2\varepsilon^2)$ runs of A with distributions D_i, where (3) each D_i still satisfies $\|2^n D_i\|_\infty = O(1/\varepsilon)$.

Now suppose that f is a Boolean function satisfying for *all* distributions D the bound

$$\|\hat{f}_D\|_\infty \geq \gamma.$$

The *Harmonic Sieve* for learning f, as suggested in [26], runs the IHA-boosting algorithm[1] based on the weak learner A provided by Lemma 9. Recall that A produces with sufficiently high probability a weak $(\Omega(\gamma), D_i)$-approximation of the target f in time $\mathrm{poly}(n, k, 1/\gamma, \log(1/\delta))$, provided that A gets an upper bound k on the ∞-norm of f_{D_i} and has access to the oracle f_{D_i}. Further, recall that f_{D_i} is defined as $f_{D_i}(x) = 2^n D_i(x) f(x)$ and each distribution D_i satisfies $\|2^n D_i\|_\infty = O(1/\varepsilon)$. Hence, $\|f_{D_i}\|_\infty = O(1/\varepsilon)$ and we can easily provide A with the required bound k. The resulting running time of A becomes $\mathrm{poly}(n, 1/\gamma, 1/\varepsilon, \log(1/\delta))$.

The remaining obstacle is the fact that the boosting algorithm cannot provide the weak learner with the exact values of $f_{D_i}(x)$. However, it can compute an accurate approximation of $f_{D_i}(x) = 2^n M_i(x)/|M_i|$ by using the already calculated estimate for $2^{-n}|M_i|$ together with the value $M_i(x)$. Note that the latter value can be exactly computed from $f(x)$ by using a single membership query to f. It can be shown that using a sufficiently accurate approximation of f_{D_i} does not have a significant impact on the learning ability of A (cf. [19]). Thus, with high probability, A indeed produces in each iteration a weak $(\Omega(\gamma), D_i)$-approximation of f which can be used by the boosting algorithm to produce an ε-approximation. The running time of the Harmonic Sieve is roughly $O(1/\gamma^2\varepsilon^2)$ times the time required for each simulation of A which is $\mathrm{poly}(n, 1/\gamma, 1/\varepsilon, \log(1/\delta))$. Thus, the Harmonic Sieve achieves the following performance.

Theorem 10. *(cf.* [19]*) For each Boolean function f satisfying for all distributions D the bound $\|\hat{f}_D\|_\infty \geq \gamma$, the Harmonic Sieve on inputs ε, γ, δ and oracle access to f outputs with probability $1-\delta$ an $O(\varepsilon)$-approximation of f in time* $\mathrm{poly}(n, 1/\varepsilon, 1/\gamma, \log(1/\delta))$.

Jackson [19] showed that for every m-term DNF f and for all distributions D on $\{0,1\}^n$ it holds that $\|\hat{f}_D\|_\infty = \max_S |E_D[f\chi_S]| \geq 1/(2m+1)$. Hence, the Harmonic Sieve can be used to efficiently learn DNF formulas with membership queries.

[1] In [19], Jackson used the F1 boosting algorithm of Freund [13].

Corollary 11. [19] *The class of m-term DNF formulas is learnable with membership queries in time* $\mathrm{poly}(n, m, 1/\varepsilon, \log(1/\delta))$.

We notice that in the *random walk model* [1], the need for membership queries can be avoided by using the *Bounded Sieve* of Bshouty and Feldman [7]. The random walk model is a variant of the PAC model where the examples are generated by performing a random walk on the cube (hence, they are not independent). The idea is to search for large coefficients by performing a breadth-first search on the Boolean hypercube rather than a depth-first search on the binary tree. Using the crucial property that the Fourier spectrum of a DNF provides a large coefficient within the low-order spectrum, Bshouty et al. [8] showed that the Bounded Sieve can be used to efficiently learn DNF formulas in the random walk model. This property of DNF formulas will also play an important role in the next subsection.

3.4 Exhaustive Search

The main drawback of the Harmonic Sieve is its need for membership queries which are used by the underlying KM-algorithm to guide the search for large coefficients. The expensive use of membership queries can be avoided, if the low-order spectrum of the target contains a large coefficient. More precisely, let f be a Boolean function satisfying for each distribution D the bound

$$\max_{|S| \leq t} |\hat{f}_D(S)| \geq \gamma.$$

Then f can be weakly $(\gamma/2, D)$-approximated by a single parity function χ_S or its negation where $|S| \leq t$. Hence, it suffices to perform an exhaustive search over all frequencies S with $|S| \leq t$; similar to the low-degree algorithm. This immediately yields the following weak learning result.

Lemma 12. [20] *There is a randomized algorithm A such that for each distribution D and Boolean function f satisfying* $\max_{|S| \leq t} |\hat{f}_D(S)| \geq \gamma$, *$A(t, \gamma, \delta, EX(f, D))$ outputs with probability $1 - \delta$ a weak $(\Omega(\gamma), D)$-approximation of f in time* $\mathrm{poly}(n^t, 1/\gamma, \log(1/\delta))$.

Applying the IHA boosting algorithm to the weak learning algorithm of Lemma 12, we can exploit the fact that the boosting algorithm only uses distributions D_i with $\|2^n D_i\|_\infty = O(1/\varepsilon)$. This yields the following strong learning result *without* membership queries.

Theorem 13. [20] *There is a randomized algorithm A such that for each Boolean function f satisfying for all distributions D with $\|2^n D\|_\infty = O(1/\varepsilon)$ the bound* $\max_{|S| \leq t} |\hat{f}_D(S)| \geq \gamma$, *$A(t, \gamma, \varepsilon, \delta, EX(f))$ outputs with probability $1 - \delta$ an ε-approximation of f in time* $\mathrm{poly}(n^t, 1/\varepsilon, 1/\gamma, \log(1/\delta))$.

An MAC^0 circuit consists of a majority-gate over a polynomial number of AC^0 circuits. Jackson et al. [20] showed that for every distribution D and for every Boolean function f computable by an MAC^0 circuit of size M and depth d it holds that

$$\max_{|S| \leq t} |\hat{f}_D(S)| = \Omega(1/Mn^t),$$

where $t = O(\log(M\|2^n D\|_\infty))^{d-1}$. In particular, $t = O(\log(M/\varepsilon))^{d-1}$ for every distribution D satisfying $\|2^n D\|_\infty \leq 1/\varepsilon$. Thus we get the following improvement of Corollary 2.

Corollary 14. [20] *The class of* MAC^0 *circuits of size* M *and depth* d *is learnable in time*

$$\mathrm{poly}(n^{O(\log(M/\varepsilon))^{d-1}}, \log(1/\delta)).$$

By Corollary 14 it immediately follows that m-term DNF formulas are learnable in time $\mathrm{poly}(n^{O(\log(m/\varepsilon))}, \log(1/\delta))$ without membership queries. Let us remark that a similar result already has been obtained by Verbeurgt [44], though by using a different approach.

An algorithm similar to the one in Theorem 13 can also be applied in the model of *statistical queries* [23]. In this model it is possible to obtain $\hat{f}(S)$ within additive error τ by asking a single statistical query. The parameter τ is called the *tolerance* of the query. It can be shown that m-term DNF formulas are learnable with $n^{O(\log(m/\varepsilon))}$ statistical queries, provided that $\tau^{-1} = \mathrm{poly}(m/\varepsilon)$ [28]. This has been improved to $\tau^{-1} = O(m/\varepsilon)$ in [31]. Interestingly, learning m-term DNF formulas *requires* $n^{\Omega(\log(m))}$ statistical queries as long as the tolerance is sufficiently large [4].

3.5 Problems

Let us close this section by highlighting some important problems and suggestions for further research concerning the learnability of DNF formulas without membership queries. For the sake of clarity of exposition we omit the reference to the parameters ε and δ.

Clearly, the ultimate goal is to achieve the analogue of Jackson's learnability result for DNF formulas without using membership queries.

Problem 15. *Are* m*-term DNF formulas learnable in time* $\mathrm{poly}(n, m)$*?*

Less ambiguous, but still a major break-through would be a polynomial-time learning algorithm for DNF formulas with a non-constant number $m(n)$ of terms. In Section 4.3 we will present an algorithm which achieves this goal for the subclass of monotone DNF formulas with running-time $\mathrm{poly}(n, (m \log n)^{\varphi(m)})$ for $\varphi(m) = \sqrt{m \log(m)}$.

Problem 16. *Is the class of m-term DNF formulas learnable in time* $\mathrm{poly}(n, (m \log n)^{\varphi(m)})$, *where φ does not depend on n?*

In Section 3.4 we saw that m-term DNF formulas are learnable in time $\mathrm{poly}(n^{\log m})$. This is the best known learning result for general m-term DNF without membership queries. So even an improvement to $\mathrm{poly}(n^{(\log m)^{\alpha}})$ for some $\alpha < 1$ would be very interesting.

As a first step towards solving Problem 16, one might attack the easier problem of learning decision trees instead of DNF formulas.

Problem 17. *Is the class of decision trees with m nodes learnable in time* $\mathrm{poly}(n, (m \log n)^{\varphi(m)})$, *where φ does not depend on n?*

As we will see in section 4.4, the subclass of monotone decision trees with m nodes is learnable in time $\mathrm{poly}(n, m)$.

4 Monotone functions and influence

The sensitivity of a Boolean function f is a measure of how strongly $f(x)$ reacts to a change of its variables. It is closely related to the notion of influence of single variables on the value of f. Interestingly, the sensitivity (as well as the influence) can be expressed in terms of the Fourier coefficients of f yielding good approximations for functions having low sensitivity. This approach works especially well for monotone functions, since in this case, the influence values of the individual variables $x_1, \ldots, x_n$ constitute the Fourier spectrum on the singleton frequencies $\{x_1\}, \ldots, \{x_n\}$.

In this section we review some important learning results for monotone Boolean functions that are based on sensitivity arguments, including Bshouty and Tamon's [9] work on monotone functions, Servedio's learnability result for monotone DNF [40], and the very recent learning algorithm of O'Donnell and Servedio for monotone decision trees [39]. We start with a discussion of the influence and sensitivity of a Boolean function f.

Influence and sensitivity

The concept of influence of a variable on a Boolean function was introduced by Ben-Or and Linial [3]. Let f be a Boolean function on n variables $x_1, \ldots, x_n$. The *influence* $I_j(f)$ of x_j on f is defined as the probability that flipping the j-th bit in a uniformly at random chosen assignment $x = (x_1, \ldots, x_n)$ changes the value of f. More formally,

$$I_j(f) = \Pr[f(x) \neq f(x \oplus e_j)],$$

where x is uniformly at random chosen from $\{0,1\}^n$ and e_j denotes the assignment $0^{j-1}10^{n-j-1}$. The *total influence* of f is defined as $I(f) = \sum_j I_j(f)$. It is easy to see that $I(f)$ equals the average sensitivity of f on all assignments $x \in \{0,1\}^n$, where the *sensitivity* of f on $x = (x_1, \ldots, x_n)$ is defined as the number of bits in x whose flipping causes f to change its value. Note that $I(f)$ further coincides with the fraction of edges in the Boolean hypercube that connect assignments x and x' having different values under f. Let us consider the influence of some basic functions.

- The dictatorship function χ_i maps $(x_1, \ldots, x_n) \mapsto (-1)^{x_i}$. Clearly, $I_j(\chi_i) = 1$ if $i = j$, and $I_j(\chi_i) = 0$ otherwise. Hence, the dictatorship function has total influence $I(\chi_i) = 1$.

- The influence of a single variable x_j on the parity function $\chi_{[n]}$ is $I_j(\chi_{[n]}) = 1$ and hence the total influence $I(\chi_{[n]})$ sums up to n.

- The influence of x_j on the majority function MAJ_n is

$$I_j(\mathrm{MAJ}_n) = \tbinom{n-1}{\lfloor n/2 \rfloor} \Big/ 2^{n-1} \,,$$

  implying that $I(\mathrm{MAJ}_n) < \sqrt{2n/\pi}$ for $n \geq 2$. In fact, it is not hard to show (e.g., see [15]) that for any monotone n-ary Boolean function f, $I(f) \leq I(\mathrm{MAJ}_n)$ implying $I(f) < \sqrt{2n/\pi}$. We will prove a slightly weaker bound in Proposition 20.

- The influence of a *k-junta*, i.e., of a function f depending only on a fixed set R of at most k variables, is clearly bounded by $I(f) \leq k$, since all variables x_j with $j \notin R$ have influence $I_j(f) = 0$.

As has been observed in [22], the influence of x_j on a Boolean function f coincides with the weight of the Fourier spectrum on all frequencies containing j. In fact, since $\hat{f}_{\oplus y}(S) = \chi_S(y)\hat{f}(S)$ (cf. Equation (2) in Section 2), the function $f_j = (f - f_{\oplus e_j})/2$ has the coefficients

$$\hat{f}_j(S) = \frac{\hat{f}(S) - \hat{f}_{\oplus e_j}(S)}{2} = \frac{\hat{f}(S) - \chi_S(e_j)\hat{f}(S)}{2} = \begin{cases} \hat{f}(S), & j \in S \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

From Parseval's identity it follows that

$$I_j(f) = E[|f_j|] = E[f_j^2] = 2^{-n} \sum_x f_j(x)^2 = \sum_S \hat{f}_j(S)^2 = \sum_{S\colon j \in S} \hat{f}(S)^2.$$

Proposition 18. *For any Boolean function f,*

$$I_j(f) = \|f_j\|^2 = \sum_{S:\, j \in S} \hat{f}(S)^2.$$

Hence, the total influence is $I(f) = \sum_S |S| \hat{f}(S)^2$.

For monotone f, the influence $I_j(f)$ coincides with the Fourier coefficient $\hat{f}(j)$. Here we adopt the convention that f is called *monotone*, if flipping any 0-bit in x to 1 does not change the value of $f(x)$ from *true* to *false* (recall that *true* is represented by the number -1 and *false* by 1). For a bit $b \in \{0,1\}$ we use $f_{j,b}$ to denote the Boolean function $f_{j,b}(x) = f(x_1, \ldots, x_{j-1}, b, x_{j+1}, \ldots, n)$. Now it is easy to see that

$$|f - f_{\oplus e_j}| = (f_{j,0} - f_{j,1}) = (f - f_{\oplus e_j})\chi_j,$$

and hence, using (5), the influence of x_j evaluates to

$$I_j(f) = E\left[\frac{|f - f_{\oplus e_j}|}{2}\right] = \frac{E[f\chi_j - f_{\oplus e_j}\chi_j]}{2} = \frac{\hat{f}(j) - \hat{f}_{\oplus e_j}(j)}{2} = \hat{f}_j(j) = \hat{f}(j).$$

Proposition 19. *For any monotone Boolean function f,*

$$I_j(f) = \hat{f}(j).$$

Hence, the total influence is $I(f) = \sum_j \hat{f}(j)$.

Letting $R = \{j \in [n] \mid I_j(f) > 0\}$ be the set of relevant variables of f and denoting the number of relevant variables by k, it follows from Cauchy-Schwarz's inequality and Parseval's identity that

$$I(f)^2 = \big(\sum_{j \in R} I_j(f)\big)^2 \leq k \sum_{j \in R} I_j(f)^2 = k \sum_{j \in R} \hat{f}(j)^2 \leq k.$$

Hence the total influence of a monotone function can be bounded as follows.

Proposition 20. *For any monotone k-junta f,*

$$I(f) \leq \sqrt{k}.$$

4.1 Monotone Boolean functions

As we have seen, the low-degree algorithm succeeds on all targets having small weight on the high frequencies of their Fourier spectrum. As we will see in the proof of following proposition, the high frequency weight can be bounded in terms of the influence. Thus, a small bound on the influence guarantees a good performance of the low-degree algorithm.

Proposition 21. *Let f be a Boolean function satisfying $I(f) \leq l$. Then*

$$\sum_{|S| \geq l/\varepsilon} \hat{f}(S)^2 \leq \varepsilon.$$

Proof. We can bound the high frequency weight for any bound t as follows.

$$\sum_{|S| \geq t} \hat{f}(S)^2 \leq \sum_{|S| \geq t} |S| \hat{f}(S)^2 / t \leq \sum_{S} |S| \hat{f}(S)^2 / t = I(f)/t.$$

Hence, the claim follows by choosing $t = l/\varepsilon$. ∎

By applying the low-degree algorithm, Proposition 21 immediately yields the following theorem.

Theorem 22. *There is a randomized algorithm A such that for each Boolean function f satisfying $I(f) \leq l$, $A(l, \varepsilon, \delta, EX(f))$ outputs with probability $1-\delta$ an $O(\varepsilon)$-approximation for f in time* $\mathrm{poly}(n^{l/\varepsilon}, \log(1/\delta))$.

By Proposition 20, for a monotone function f we have the bound $I(f) \leq \sqrt{n}$, which immediately yields the following learning result of Bshouty and Tamon.

Corollary 23. [9] *The class of monotone Boolean functions is learnable in time* $\mathrm{poly}(n^{\sqrt{n}/\varepsilon}, \log(1/\delta))$.

In Section 4.4 we will present an algorithm for monotone Boolean functions running in time $\mathrm{poly}(n, 2^{(l/\varepsilon)^2})$ rather than $\mathrm{poly}(n^{l/\varepsilon})$ as in Theorem 22.

4.2 Monotone juntas

By Proposition 20 we know that the influence of a monotone k-junta is bounded by $\sqrt{k}$. Hence, the algorithm of Theorem 22 finds a low degree $O(\varepsilon)$-approximation g for f in time $\mathrm{poly}(n^{\sqrt{k}/\varepsilon}, \log(1/\delta))$. Bshouty and Tamon [9] observed that by ignoring variables of sufficiently small influence, the running-time can be improved to $\mathrm{poly}(k^{\sqrt{k}/\varepsilon}, \log(1/\delta))$.

For a bound $\theta \geq 0$ let

$$R(\theta) = \{j \in [n] \mid I_j(f) > \theta\}$$

denote the set of variables having influence greater than θ. Then it is easy to bound the Fourier weight of a k-junta on the frequencies containing at least one variable of small influence.

Proposition 24. *For any k-junta f,*

$$\sum_{S \not\subseteq R(\varepsilon/k)} \hat{f}(S)^2 \leq \varepsilon.$$

Proof. Since for a k-junta the number $|R|$ of relevant variables is bounded by k and since each variable x_j with $j \notin R(\theta)$ has influence $I_j(f) \leq \theta$, it follows that

$$\sum_{S \not\subseteq R(\theta)} \hat{f}(S)^2 \leq \sum_{j \notin R(\theta)} \sum_{S\colon j \in S} \hat{f}(S)^2 = \sum_{j \in R - R(\theta)} I_j(f) \leq \theta k.$$

Hence, the claim follows by choosing $\theta = \varepsilon/k$. ∎

In the following we will frequently consider the collection of all frequencies S of order at most t containing only variables having influence greater than θ, which we denote by

$$G(\theta, t) = \{S \subseteq R(\theta) \mid |S| \leq t\}.$$

The following proposition shows that a k-junta f with influence $I(f) \leq l$ can be $O(\varepsilon)$-approximated by taking only the coefficients corresponding to frequencies inside $G(\varepsilon/k, l/\varepsilon)$.

Proposition 25. *Let f be a k-junta satisfying $I(f) \leq l$. Then*

$$\sum_{S \notin G(\varepsilon/k, l/\varepsilon)} \hat{f}(S)^2 \leq 2\varepsilon.$$

Proof. Using Propositions 21 and 24 it immediately follows that

$$\sum_{S \notin G(\varepsilon/k, l/\varepsilon)} \hat{f}(S)^2 \leq \sum_{S \not\subseteq R(\varepsilon/k)} \hat{f}(S)^2 + \sum_{|S| \geq l/\varepsilon} \hat{f}(S)^2 \leq 2\varepsilon.$$

∎

If f is monotone, we can collect all variables having large influence by estimating the Fourier coefficients of all singleton frequencies. More precisely, in order to find all variables x_j having influence $I_j(f) \geq \theta$, we compute estimates a_j for the Fourier coefficients $\hat{f}(j)$ by drawing sufficiently many examples from the oracle $EX(f)$ and collect all variables x_j with $a_j \geq 3\theta/4$. Then, with high probability, we get all variables x_j with $I_j(f) \geq \theta$ and no variables x_j with $I_j(f) \leq \theta/2$. This algorithm has been called `Find-Variables` by Servedio [40].

Proposition 26. *For each monotone Boolean function f, `Find-Variables` on inputs θ, δ and access to $EX(f)$, outputs with probability $1-\delta$ in time* $\mathrm{poly}(n, 1/\theta, \log(1/\delta))$ *a set R^* with $R(\theta) \subseteq R^* \subseteq R(\theta/2)$.*

In order to compute an ε-approximation for a monotone k-junta f, we first use `Find-Variables` with $\theta = \varepsilon/k$ to obtain with high probability a variable set $R^* \subseteq R(\varepsilon/2k)$ containing all variables x_j with $I_j(f) \geq \varepsilon/k$. Since $R^* \subseteq R(\varepsilon/2k)$ implies that R^* contains only relevant variables, the size of $G^* = \{S \subseteq R^* \mid |S| \leq l/\varepsilon\}$ is polynomial in $k^{l/\varepsilon}$. Hence, we can apply the generalized low-degree algorithm to get the following learning result for monotone k-juntas having small influence.

Theorem 27. *There is a randomized algorithm A such that for each monotone k-junta f with $I(f) \leq l$, $A(k, l, \varepsilon, \delta, EX(f))$ outputs with probability $1-\delta$ an $O(\varepsilon)$-approximation for f in time* $\mathrm{poly}(n, k^{l/\varepsilon}, \log(1/\delta))$.

Corollary 28. [9] *The class of monotone k-juntas is learnable in time*

$$\mathrm{poly}(n, k^{\sqrt{k}/\varepsilon}, \log(1/\delta)).$$

4.3 Monotone functions that are close to juntas

By refining the arguments used in the preceding subsection we will now see that the generalized low-degree algorithm also succeeds on monotone Boolean functions that are sufficiently close to a monotone k-junta [9]. As a consequence, the generalized low-degree algorithm (in conjunction with `Find-Variables`) becomes applicable to monotone m-term DNF formulas.

We call a function f an (ε, k)-*junta*, if f is ε-close to a k-junta. We first show that for an (ε, k)-junta, the number of variables x_j having influence $I_j(f) \geq \varepsilon$ is bounded by k.

Proposition 29. *For any (ε, k)-junta, $|R(\varepsilon)| \leq k$.*

Proof. Let h be a k-junta that is ε-close to f. Observe that for any variable x_j with $I_j(h) = 0$ it holds that $\hat{h}(S) = 0$ if $j \in S$. Hence,

$$I_j(f) = \sum_{S\colon j \in S} \hat{f}(S)^2 = \sum_{S\colon j \in S} (\hat{f}(S) - \hat{h}(S))^2 \leq \|f - h\|^2 \leq \varepsilon.$$

This shows that any variable in $R(\varepsilon)$ must be relevant for h. ■

In the next proposition we bound the weight of the high order Fourier spectrum of f under the assumption that f is close to some function g with a small weight on this region.

Proposition 30. *For any function f that is ε-close to some Boolean function g with $\sum_{|S|\geq t} \hat{g}(S)^2 \leq \varepsilon$,*

$$\sum_{|S|\geq t} \hat{f}(S)^2 \leq 4\varepsilon.$$

Proof. Using the inequality $\hat{f}^2 = (\hat{f} - \hat{g} + \hat{g})^2 \leq 2(\hat{f} - \hat{g})^2 + 2\hat{g}^2$ (Cauchy-Schwarz) it follows that

$$\sum_{|S|\geq t} \hat{f}(S)^2 \leq 2 \sum_{|S|\geq t} (\hat{f}(S) - \hat{g}(S))^2 + 2 \sum_{|S|\geq t} \hat{g}(S)^2 \leq 4\varepsilon.$$

■

Now assume that f is an $(\varepsilon/n, k)$-junta with the additional property that it is ε-close to some Boolean function g whose Fourier weight on the frequencies S with $|S| \geq t$ is bounded by ε. Then by using Proposition 24 with $k = n$ as well as Proposition 30, it follows that the frequency collection $G(\varepsilon/n, t)$ has the property

$$\sum_{S \notin G(\varepsilon/n,t)} \hat{f}(S)^2 \leq \sum_{S \not\subseteq R(\varepsilon/n)} \hat{f}(S)^2 + \sum_{|S|\geq t} \hat{f}(S)^2 = O(\varepsilon).$$

This means that f can be $O(\varepsilon)$-approximated by using only Fourier coefficients corresponding to frequencies in $G(\varepsilon/n, t)$. Since by Proposition 29 the size of $R(\varepsilon/n)$ is bounded by k, it further follows that the size of $G(\varepsilon/n, t)$ is bounded by k^t. Thus we can use the algorithm `Find-Variables` to compute a superset R^* of $R(\varepsilon/2n)$ containing only variables in $R(\varepsilon/n)$ in order to get the following learning result.

Theorem 31. *There is a randomized algorithm A such that for each monotone $(\varepsilon/n, k)$-junta f that is ε-close to some Boolean function g with $\sum_{|S|\geq t} \hat{g}(S)^2 \leq \varepsilon$, $A(k, t, \varepsilon, \delta, EX(f))$ outputs with probability $1 - \delta$ an $O(\varepsilon)$-approximation for f in time* $\mathrm{poly}(n, k^t, \log(1/\delta))$.

Since by Proposition 21, $I(g) \leq l$ implies that the Fourier weight of g on the frequencies S with $|S| > l/\varepsilon$ is bounded by ε, we also obtain the following corollary.

Corollary 32. *There is a randomized algorithm A such that for each monotone $(\varepsilon/n, k)$-junta f that is ε-close to some Boolean function g with influence $I(g) \leq l$, $A(k, l, \varepsilon, \delta, EX(f))$ outputs with probability $1 - \delta$ an $O(\varepsilon)$-approximation for f in time* $\mathrm{poly}(n, k^{l/\varepsilon}, \log(1/\delta))$.

Since for any $\varepsilon > 0$, an m-term DNF is an $(\varepsilon, m\log(m/\varepsilon))$-junta, Corollary 32 implies the following learning result for monotone m-term DNF, which is in fact a consequence of a stronger result from [9].

Corollary 33. [9] *The class of monotone m-term DNF formulas is learnable in time*

$$\mathrm{poly}(n, (m\log(n/\varepsilon))^{\sqrt{m\log(m/\varepsilon)}/\varepsilon}, \log(1/\delta)).$$

Hence, for $m = O((\log n)^2/(\log\log n)^3)$ and constant ε, monotone m-term DNF formulas are learnable in polynomial time.

Servedio [40] used a bound of Mansour [36] to show that any m-term DNF f is ε-close to a Boolean function g satisfying $\sum_{|S|>t} \hat{g}(S)^2 \leq \varepsilon$ for $t = O(\log(m/\varepsilon)\log(1/\varepsilon))$. By Theorem 31, this implies the following exponential improvement of Corollary 33.

Corollary 34. [40] *The class of monotone m-term DNF formulas is learnable in time*

$$\mathrm{poly}(n, (m\log(n/\varepsilon))^{\log(m/\varepsilon)\log(1/\varepsilon)}, \log(1/\delta)).$$

Hence, monotone $O(2^{\sqrt{\log n}})$-term DNF formulas are learnable in polynomial time.

4.4 Monotone functions with bounded influence

Since k-juntas have low influence, it is clear that all functions that are sufficiently close to some k-junta also must have low influence. As shown by Friedgut [14], also the converse is true: any Boolean function f having low influence must be close to a k-junta where k only depends on the influence and not on the arity of f. More precisely, if $I(f) \leq l$ then f is a $(\varepsilon, 2^{O(\varepsilon/l)})$-junta. Very recently, this relationship has been used by O'Donnell and Servedio [39] to demonstrate the applicability of the generalized low-degree algorithm to the class of monotone decision trees.

The proof of Friedgut's result yields the following approximability result for Boolean functions with bounded influence.

Proposition 35. [14] *For any Boolean function f with $I(f) \leq l$,*

$$\sum_{S \notin G(\theta,t)} \hat{f}(S)^2 = O(\varepsilon),$$

where $\theta = (\varepsilon 2^{-l/\varepsilon}/l)^3$ and $t = l/\varepsilon$.

Proof. For a given ε we want to derive a bound on θ such that at most an ε fraction of the weight of the Fourier spectrum of f is outside of the collection $G(\theta, t)$. Let $H = \{S \subseteq [n] \mid S \not\subseteq R(\theta) \wedge |S| < t\}$ and note that $S \notin G(\theta, t)$ implies that either $|S| \geq t$ or $S \in H$. Now it follows by Proposition 21 that

$$\sum_{S \notin G(\theta,t)} \hat{f}(S)^2 \leq \sum_{|S| \geq t} \hat{f}(S)^2 + \sum_{S \in H} \hat{f}(S)^2 \leq \varepsilon + \sum_{S \in H} \hat{f}(S)^2,$$

Hence, it suffices to choose θ small enough such that the Fourier weight of f on H is bounded by $O(\varepsilon)$. Using Beckner-Bonami (cf. inequality (4) in Section 2) and letting $H_j = \{S \subseteq [n] \mid j \in S \wedge |S| < t\}$ it follows for each position j that

$$\sum_{S \in H_j} 2^{-t} \hat{f}(S)^2 \leq \sum_{S:\, j \in S} 2^{-|S|} \hat{f}(S)^2 = \|T_{1/\sqrt{2}} f_j\|^2 \leq (\|f_j\|^2)^{4/3} = I_j(f)^{4/3},$$

implying that $\sum_{S \in H_j} \hat{f}(S)^2 \leq 2^t I_j(f)^{4/3}$. Note that for each $S \in H$ there is some $j \notin R(\theta)$ such that $S \in H_j$. Hence, summing up and using the bounds $I(f) \leq l$ and $I_j(f) \leq \theta$ for all $j \notin R(\theta)$ we get

$$\sum_{S \in H} \hat{f}(S)^2 \leq \sum_{j \notin R(\theta)} \sum_{S \in H_j} \hat{f}(S)^2 \leq \sum_{j \notin R(\theta)} 2^t I_j(f)^{4/3} \leq 2^t \theta^{1/3} \sum_j I_j(f) \leq 2^t \theta^{1/3} l.$$

Thus, choosing $\theta = (\varepsilon 2^{-t}/l)^3$ yields the desired bound. Since $t = l/\varepsilon$, the size of $R(\theta)$ is bounded by $l/\theta = l^4 2^{3l/\varepsilon}/\varepsilon^3 = 2^{O(l/\varepsilon)}$ and it follows that the size of G is bounded by $2^{O(l/\varepsilon)^2}$. ∎

As an immediate consequence we can state the following learning result.

Theorem 36. *There is a randomized algorithm A such that for each monotone Boolean function f with $I(f) \leq l$, $A(l, \varepsilon, \delta, EX(f))$ outputs with probability $1 - \delta$ an $O(\varepsilon)$-approximation for f in time* $\mathrm{poly}(n, 2^{(l/\varepsilon)^2}, \log(1/\delta))$.

As shown by O'Donnell and Servedio [39], any Boolean function f computable by a decision tree of size m has the property that $\sum_i \hat{f}(i) \leq \sqrt{\log m}$. If f is monotone, this implies that $I(f) = \sum_i \hat{f}(i) \leq \sqrt{\log m}$.

Corollary 37. [39] *The class of monotone decision trees of size m is learnable in time*

$$\mathrm{poly}(n, m^{(1/\varepsilon)^2}, \log(1/\delta)).$$

Hence, for constant ε, monotone decision trees are learnable in polynomial time.

4.5 Problems

Let us conclude with some interesting problems concerning the learnability of monotone DNF formulas and juntas. As in Section 3.5 we omit the parameters ε and δ for clarity reasons.

Monotone DNF

In the light of O'Donnell and Servedio's efficient learning algorithm for monotone decision trees [39] an interesting question to ask is whether this result can be extended to monotone DNF formulas.

Problem 38. *Are monotone m-term DNF formulas learnable in time* $\mathrm{poly}(n, m)$*?*

It is further shown in [39] that the influence bound $I(f) \leq \sqrt{\log m}$ cannot be extended to monotone m-term DNF formulas or even to functions f that are computable by *both* an m-term DNF as well as by an m-clause CNF. Thus, Problem 38 cannot be solved by solely relying on Theorem 36.

The currently best learning algorithm for monotone m-term DNF is the one of Corollary 34 due to Servedio [40] which runs in time $\mathrm{poly}(n, (m \log n)^{\varphi(m)})$ where $\varphi(m) = \log m$. It would be very interesting to improve this result to an algorithm having *fixed parameterized complexity* in the sense of Downey and Fellows [11].

Problem 39. *Are monotone m-term DNF formulas learnable in time* $\mathrm{poly}(n, \varphi(m))$*, where φ does not depend on n?*

Juntas

As we saw in Section 4.2, the analogue of Problem 39 for monotone k-juntas has been solved by Bshouty and Tamon [9] by providing an algorithm running in time $\mathrm{poly}(n, \varphi(k))$ for $\varphi(k) = k^{\sqrt{k}}$ (cf. Corollary 28). An immediate question is whether this can be extended to general juntas.

Problem 40. *Are k-juntas learnable in time* $\mathrm{poly}(n, \varphi(k))$*, where φ does not depend on n?*

We want to emphasize that Problem 40 is a very important question whose answer would have immediate applications to the learnability of DNF formulas (as every DNF formula is close to a junta) as well as to the important area of *feature selection* (cf. [5]). In fact, Mossel et al. [37] consider the problem of learning juntas as the single most important problem in uniform distribution learning. A slightly less ambiguous goal is to learn k-juntas in time $\text{poly}(n, (k \log n)^{\varphi(k)})$. This would imply that juntas with a non-constant number $k(n)$ of relevant variables are learnable in polynomial time.

The currently best learning algorithm for k-juntas is due to Mossel et al. [37] and runs in time $O(n^{\alpha k})$ for some $\alpha < 0.7$. So even an improvement to an $\alpha < 0.5$ would be a significant progress.

Problem 41. *Are k-juntas learnable in time $O(n^{\alpha k})$ for some $\alpha < 0.5$?*

For the special case of symmetric juntas, Mossel et al. [37] used a result of von zur Gathen and Roche [45] to bound the running-time by $n^{\alpha k}$ for some $\alpha < 2/3$. The analogue of Problem 41 for symmetric juntas has been solved by Lipton et al. [34] by providing an algorithm with running-time $O(n^{\alpha k})$ for $\alpha = 3/31$. This bound has been subsequently improved to $O(n^{k/\log k})$ in [29].

References

[1] P. Bartlett, P. Fischer, and K.-U. Hoffgen. Exploiting random walks for learning. In *Proc. 7th Annual ACM Conference on Computational Learning Theory*, pages 318–327, 1994.

[2] W. Beckner. Inequalities in Fourier analysis. *Annals of Mathematics*, 102: 159–182, 1975.

[3] M. Ben-Or and N. Linial. Collective coin flipping. In S. Micali, editor, *Randomness and Computation*, pages 91–115. Academic Press, 1989.

[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 *Proc. 26th ACM Symposium on Theory of Computing*, pages 253–262, 1994.

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

[6] A. Bonami. Etude des coefficients fourier des fonctiones de $l^p(g)$. *Ann. Inst. Fourier*, 20(2):335–402, 1970.

[7] J. H. Bshouty and V. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research*, 2:359–395, 2002.

[8] N. Bshouty, E. Mossel, R. O'Donnell, and R. Servedio. Learning DNF from random walks. *Journal of Computer and System Sciences*, 71(3):250–265, 2005.

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

[10] I. Dinur and E. Friedgut. Analytical methods in combinatorics and computer science. Lecure Notes, 2005. URL `http://www.cs.huji.ac.il/~analyt/content.html`.

[11] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[12] H. Dym and H.P. McKean. *Fourier Series and Integrals*. Academic Press, 1972.

[13] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[14] E. Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18(1):27–36, 1998.

[15] E. Friedgut and G. Kalai. Every monotone graph property has a sharp threshold. In *Proc. Amer. Math. Soc.*, volume 124, pages 2993–3002, 1996.

[16] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st ACM Symposium on Theory of Computing*, 1989.

[17] J. Hastad. *Computational limitations of small-depth circuits*. MIT Press, 1987.

[18] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proc. 36th IEEE Symposium on the Foundations of Computer Science*, pages 538–545, 1995.

[19] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997.

[20] J. Jackson, A. Klivans, and R. Servedio. Learnability beyond AC^0. In *Proc. 34th ACM Symposium on Theory of Computing*, pages 776–784, 2002.

[21] J. Jackson and C. Tamon. Fourier analysis in machine learning. COLT 97 Tutorial, 1997. URL `http://learningtheory.org/tutorial/COLT97Fourier.ps`.

[22] J. Kahn, G. Kalai, and N. Linial. The influence of variables on Boolean functions. In *Proc. 29th IEEE Symposium on the Foundations of Computer Science*, pages 68–80, 1988.

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

[24] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[25] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proc. 25th ACM Symposium on Theory of Computing*, pages 372–381, 1993.

[26] A. Klivans and R. Servedio. Boosting and hard-core sets. In *Proc. 40th IEEE Symposium on the Foundations of Computer Science*, pages 624–633, 1999.

[27] A. Klivans and R. Servedio. Learning DNF in time $2^{\tilde{o}(n^{1/3})}$. *Journal of Computer and System Sciences*, 68(2):303–318, 2004.

[28] J. Köbler and W. Lindner. A general dimension for approximately learning Boolean functions. In *Proc. 13th International Workshop on Algorithmic Learning Theory*, pages 139–148, 2002.

[29] M. Kolountzakis, E. Markakis, and A. Mehta. Learning symmetric juntas in time $n^{o(k)}$. Interface between Harmonic Analysis and Number Theory, 2005.

[30] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SICOMP*, 22(6):1331–1348, 1993.

[31] W. Lindner. Learning DNF by statistical and proper distance queries under the uniform distribution. In *Proc. 16th International Workshop on Algorithmic Learning Theory*, pages 198–210, 2005.

[32] N. Linial. Harmonic analysis and combinatorial applications. Lecure Notes, 2005. URL `http://www.cs.huji.ac.il/~nati/PAPERS/uw/`.

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

[34] R. Lipton, E. Markakis, A. Mehta, and N.K. Vishnoi. On the Fourier spectrum of symmetric Boolean functions with applications to learning symmetric juntas. In *Proc. 20th Annual IEEE Conference on Computational Complexity*, pages 112 – 119, 2005.

[35] Y. Mansour. Learning Boolean functions via the Fourier transform. In V.P. Roychodhury, K-Y. Siu, and A. Orlitsky, editors, *Theoretical Advances in Neural Computation and Learning*, pages 391–424. Kluwer Academic Publishers, 1994.

[36] 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.

[37] E. Mossel, R. O'Donnell, and R. Servedio. Learning juntas. In *Proc. 35th Ann. ACM Symp. on the Theory of Computing, 2003.*, 2003.

[38] R. O'Donnell. *Computational Applications of Noise Sensitivity*. PhD thesis, Massachutes Institute of Technology, 2003.

[39] R. O'Donnell and R. Servedio. Learning monotone decision trees in polynomial time. In *Proc. 21st Annual IEEE Conference on Computational Complexity*, 2006.

[40] R. Servedio. On learning monotone DNF under product distributions. *Inf. Comput.*, 193(1):57–74, 2004.

[41] R. Shapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[42] D. Stefankovic. Fourier transforms in computer science. Master's thesis, University of Chicago, 2002.

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

[44] K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proc. 3rd Annual ACM Conference on Computational Learning Theory*, pages 314–326, 1990.

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

[46] I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.