Learning Boolean Functions *via* the Fourier Transform

Yishay Mansour*

February 23, 1994

Abstract

We survey learning algorithms that are based on the Fourier Transform representation. In many cases we simplify the original proofs and integrate the proofs of related results. We hope that this would give the reader a complete and comprehensive understanding of both the results and the techniques.

1 Introduction

The importance of using the "right" representation of a function in order to "approximate" it has been widely recognized. The Fourier Transform representation of a function is a classic representation which is widely used to approximate real functions (i.e. functions whose inputs are real numbers). However, the Fourier Transform representation for functions whose inputs are boolean has been far less studied. On the other hand it seems that the Fourier Transform representation can be used to learn many classes of boolean functions.

At this point it would be worthwhile to say a few words about the Fourier Transform of functions whose inputs are boolean. The basis functions are based on the parity of subsets of the input variables. Every function whose inputs are boolean can be written as a linear combination of this basis, and coefficients represent the correlation between the function and the basis function.

The work of [LMN89] was the first to point of the connection between the Fourier spectrum and learnability. They presented a quasi-polynomial-time (i.e. $O(n^{poly-log(n)})$) algorithm for learning the class AC^0 (polynomial size constant depth circuits); the approximation is with respect to the uniform distribution. Their main result is an interesting property of

*Computer Science Dept., Tel-Aviv University. This research was supported by THE ISRAEL SCIENCE FOUNDATION administered by THE ISRAEL ACADEMY OF SCIENCE AND HUMANITIES.

the representation of the Fourier Transform of AC^0 circuits; based on it they derived a learning algorithm for AC^0. For the specific case of DNF the result was improved in [Man92]. In [Kha93] it is shown, based on a cryptographic assumption, that the running time of $O(n^{poly-log(n)})$ for AC^0 circuits is the best possible.

In [AM91] polynomial time algorithms are given for learning both probabilistic decision lists and probabilistic read once decision trees with respect to the uniform distribution. In this paper we concentrate on deterministic functions, namely deterministic decision lists, hence, some of the techniques and the results of [AM91] do not appear here.

The work of [KM91] uses the Fourier representation to derive a polynomial time learning algorithm for decision trees, with respect to the uniform distribution. The algorithm is based on a procedure that finds the significant Fourier coefficients.

Most of the work on learning using the Fourier Transform assumes that the underline distribution is uniform. There has been a few successful attempts to extend some of the results to product distributions. In [FJS91] it is shown how to learn AC^0 circuit with respect to a product distribution. In [Bel92] the algorithm that searches for the significant coefficients is extended to work for product distributions. However, in this paper we concentrate on the uniform distribution.

There are additional works about the Fourier Transform representation of boolean function. The first work used Fourier Transform to the show results in theoretical computer science was the work of [KKL88], that proves properties about the sensitivity of boolean functions. The relation between DNFs and their Fourier Transform representation is also studied in [BHO90]. Other works that are investigating the Fourier Transform of Boolean functions are [Bru90, BS90, SB91].

In this work we focus on the main results about the connection between Fourier Transform and learnability. The survey is mainly based on the works that appeared in [LMN89, AM91, KM91, Man92]. Some of the proofs are a simplification of the original proofs and in many cases we try to give a common structure to different results.

Some of the learning results shown here are based on the lower bound techniques that were developed for proving lower bound for polynomial size constant depth circuit [Ajt83, FSS84, Yao85, Has86]. When we need to apply those results we only state the results that we use but do not prove them.

The paper is organized as following. Section 2 gives the definition of the learning model and some basic results that are used throughout the paper. Section 3 introduces the Fourier Transform and some of its properties. Section 4 establishes the connection between the Fourier Transform and learning. This section includes two important algorithms. The Low Degree algorithm, that approximates functions by considering their Fourier coefficients on small sets and the Sparse algorithm, that is based on the work of [GL89, KM91], which learns a function by approximating its significant coefficients.

In Section 5, we show various classes of functions that can be learned using the above algorithm. We start with the simple class of decision lists (from [AM91]). We continue with properties of decision trees (from [KM91]). The last class is boolean circuits there we show properties for both DNF and AC^0 circuits (from [LMN89, Man92]).

2 Preliminaries

2.1 Learning Model

The learning model has a class of functions $\mathcal{F}$ which we wish to learn. Out of this class there is a specific function $f \in \mathcal{F}$ which is chosen as a target function. A learning algorithm has access to examples. An example is a pair $< x, f(x) >$, where x is an input and $f(x)$ is the value of the target function on the input x. After requesting a finite number of examples, the learning algorithm outputs a hypothesis h. The error of a hypothesis h, with respect to the function f, is defined to be $error(f,h) \triangleq \Pr[f(x) \neq h(x)]$, where x is distributed uniformly over $\{0,1\}^n$.

We discuss two models for accessing the examples. In the *uniform distribution model* the algorithm has access to a random source of examples. Each time the algorithm requests an example, a random input $x \in \{0,1\}^n$ is chosen uniformly, and the example $< x, f(x) >$ is returned to the algorithm. In the *membership queries model*, the algorithm can query the unknown function f on any input $x \in \{0,1\}^n$ and receive the example $< x, f(x) >$.

A randomized algorithm A *learns* a class of functions $\mathcal{F}$ if for every $f \in \mathcal{F}$ and $\varepsilon, \delta > 0$ the algorithm outputs an hypothesis h such that with probability at least $1-\delta$,

$$error(f,h) \leq \varepsilon$$

The algorithm A learns in *polynomial time* if its running time is polynomial in n, $1/\varepsilon$, and $\log 1/\delta$.

2.2 Probability

In many places we use the Chernoff bound to bound the sum of random variables. (For a presentation of the bounds see [HR89].)

Lemma 2.1 (Chernoff) *Let $X_1, \ldots, X_m$ be independent identically distributed random variables such that, $X_i \in [-1,+1]$, $E[X_i] = p$ and $S_m = \sum_{i=1}^m X_i$. Then*

$$\Pr\left[|\frac{S_m}{m} - p| \geq \lambda \right] \leq 2e^{-\lambda^2 m/2}$$

3 The Fourier Basis

The functions we are interested in have boolean inputs and are of the form,

$$f : \{0,1\}^n \to \mathbb{R}.$$

We are mainly interested in boolean functions of the form,

$$f : \{0,1\}^n \to \{-1,+1\}.$$

We are interested in creating a basis for those functions. Recall that a basis, in this case, is a set of basis functions such that any function of the form $f : \{0,1\}^n \to \mathbb{R}$ can be represented as a linear combination of the basis functions. One basis is the functions $term_\alpha(x)$, for $\alpha \in \{0,1\}^n$, where $term_\alpha(\alpha) = 1$ and $term_\alpha(\beta) = 0$ for $\beta \neq \alpha$. Any function f can be written as $\sum_\alpha a_\alpha term_\alpha(x)$ where the constants are $a_\alpha = f(\alpha)$. In the following we describe a different basis which is called the Fourier basis.

The Fourier basis has 2^n functions; for each $\alpha \in \{0,1\}^n$ there is a function $\chi_\alpha : \{0,1\}^n \to \{+1,-1\}$. The value of a basis function χ_α is,

$$\chi_\alpha(x) = (-1)^{\sum_{i=1}^n x_i \alpha_i}.$$

An alternative way of defining the same functions, which we also use throughout the text, is to denote the basis functions using a subset $S \subseteq \{1,\dots,n\}$. The set S defines the set of inputs on which the function χ_S is defined. The value of χ_S depends on the parity of the inputs in S. Formally,

$$\chi_S(x) = \prod_{i \in S} (-1)^{x_i} = \begin{cases} +1 & \text{if } \sum_{i\in S} x_i \bmod 2 = 0 \\ -1 & \text{if } \sum_{i\in S} x_i \bmod 2 = 1 \end{cases}$$

Note that,

$$\chi_S(x) \equiv \chi_\alpha(x)$$

where $S \subseteq \{1,\dots,n\}$, and $i \in S \iff \alpha_i = 1$.

The *inner product* of two functions f and g is,

$$< f, g > = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)g(x) \; = \; E[f \cdot g],$$

where E is the expected value of $f \cdot g$ using the uniform distribution on $\{0,1\}^n$. The *norm* of a function is $\|f\| = \sqrt{< f, f >}$.

3.1 Basis Properties

We mention a few properties of the Fourier basis defined above.

- The basis is *normal*, i.e. $\|\chi_S\| \equiv 1$, since $\forall x : \chi_S^2(x) = 1$.
- $\chi_\alpha \cdot \chi_\beta = \chi_{\alpha+\beta}$, where the addition is bit-wise exclusive or.
- If $\alpha \neq 0$ then $E[\chi_\alpha] = 0$.
- *Orthogonality*

$$< \chi_\alpha, \chi_\beta >= E[\chi_\alpha \cdot \chi_\beta] = E[\chi_{\alpha+\beta}] = \begin{cases} 1 & \text{if } \alpha = \beta \\ 0 & \text{if } \alpha \neq \beta \end{cases}.$$

- *Dimensionality.* The orthogonality implies that the dimension of the basis is 2^n.

From the dimensionality of the basis we can deduce that every function $f : \{0,1\}^n \to \mathbb{R}$ can be represented as a linear combination of basis functions.

Claim 3.1 *For any $f : \{0,1\}^n \to \mathbb{R}$ then,*

$$f(x) = \sum_{\alpha \in \{0,1\}^n} a_\alpha \chi_\alpha(x),$$

where $a_\alpha = \hat{f}(\alpha) =< f, \chi_\alpha >$.

The *Parseval's identity* relates the values of the coefficients to the values of the function.

Theorem 3.2 (Parseval's Identity) *For any $f : \{0,1\}^n \to \mathbb{R}$,*

$$\sum_{\alpha \in \{0,1\}^n} \hat{f}^2(\alpha) = E[f^2]$$

Proof: Consider the following simple algebraic manipulations.

$$\begin{aligned} E_x[f^2(x)] &= E_x\left[\left(\sum_\alpha \hat{f}(\alpha)\chi_\alpha(x)\right)\left(\sum_\beta \hat{f}(\beta)\chi_\beta(x)\right)\right] \\ &= \sum_\alpha \sum_\beta \hat{f}(\alpha)\hat{f}(\beta) E_x\left[\chi_{\alpha+\beta}(x)\right] \end{aligned}$$

If $\alpha \neq \beta$, then $E_x[\chi_{\alpha+\beta}(x)]$ is zero,, therefore, the expression reduces to,

$$E[f^2] = \sum_\alpha \hat{f}^2(\alpha),$$

which completes the proof. $\square$

Parseval's identity for *boolean* functions, i.e. $f : \{0,1\}^n \to \{-1,+1\}$, states that $\sum_\alpha \hat{f}^2(\alpha) \equiv 1$

4 Learning and Fourier Transform

We start by considering an example. Let f be a boolean function, such that for some (known) β it holds that $\hat{f}(\beta) = 0.9$, and no other information about f is known. A natural hypothesis would be, $h(x) \equiv 0.9\chi_\beta(x)$, and we would like to estimate the error squared for h. (Intuitively it is clear that if the expected error squared is small then we have a good estimation in some sense. Later we will show how this parameter relates to boolean prediction.)

Let the error function is $error_h(x) = |f(x) - h(x)|$. The expected error square is,

$$E[(f-h)^2] = \left(\sum_{\alpha \neq \beta} \hat{f}^2(\alpha)\right) + \left(\hat{f}^2(\beta) - \hat{f}^2(\beta)\right) = 1 - \hat{f}^2(\beta) = 1 - 0.81 = 0.19.$$

Introducing another piece of information, e.g. $\hat{f}(\gamma) = 0.3$, would reduce the error. Our new hypothesis would be $h(x) \equiv 0.9\chi_\beta(x) + 0.3\chi_\gamma(x)$, and the expected error square is,

$$E[(f-h)^2] = 1 - \hat{f}^2(\beta) - \hat{f}^2(\gamma) = 1 - 0.81 - 0.09 = 0.1.$$

Boolean Prediction

In the example above, our hypothesis $h(x)$ was not a boolean function. In order to get a boolean prediction we can output $+1$ if $h(x) \geq 0$ and -1 if $h(x) < 0$. More formally,

Definition The *Sign* function takes a real parameter and return its sign,

$$Sign(z) \stackrel{\text{def}}{=} \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}.$$

The following claim shows that the expected error squared bound the probability of an error in predicting according to the sign of h.

Claim 4.1 *If f is a boolean function then,*

$$\Pr[f(x) \neq Sign(h(x))] \leq E[(f-h)^2].$$

Proof: Let I be the indicator function, i.e.

$$I\left[f(x) \neq Sign(h(x))\right] \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } f(x) \neq Sign(h(x)) \\ 0 & \text{if } f(x) = Sign(h(x)) \end{cases}$$

The probability of error is,

$$\Pr[f(x) \neq Sign(h(x))] = \frac{1}{2^n} \sum_x I\left[f(x) \neq Sign(h(x))\right].$$

We show that for every $x \in \{0,1\}^n$, $I[f(x) \neq Sign(h(x))] \leq (f(x) - h(x))^2$, which implies the claim. We consider the following two cases.

- If $f(x) = Sign(h(x))$ then $I\left[f(x) \neq Sign(h(x))\right] = 0$, so clearly,

$$I\left[f(x) \neq Sign(h(x))\right] = 0 \leq (f(x) - h(x))^2.$$

- If $f(x) \neq Sign(h(x))$ then we have $|f(x) - h(x)| \geq 1$. Therefore,

$$I\left[f(x) \neq Sign(h(x))\right] = 1 \leq (f(x) - h(x))^2.$$

□

As a result from the claim we can use $E[(f-h)^2]$ as an upper bound for $\Pr[f(x) \neq Sign(h(x))]$. Notice that the above proof holds for any distribution although we apply it here only to the uniform distribution.

The following definition would be useful.

Definition 4.2 *A (real) function g ε-*approximates *f if $E[(f(x) - g(x))^2] \leq \varepsilon$.*

4.1 Approximating a single coefficient

Recall the example in which we "know" that the coefficient at β is "large". There we assumed that we are given the value of the coefficient of β (i.e. $\hat{f}(\beta)$ is given). In this section we show how to approximate it from random examples.

We are interested in approximating a coefficient $\hat{f}(\beta)$ for a given β. Recall that,

$$\hat{f}(\beta) =< f, \chi_\beta >= E[f \cdot \chi_\beta]$$

Since we are interested only in an estimate, we can sample randomly x_i's and take the average value. The sampling is done by choosing the x_is from the uniform distribution, and the estimate is,

$$a_\beta = \frac{1}{m} \sum_{i=1}^{m} f(x_i)\chi_\beta(x_i).$$

Using the Chernoff bounds, for $m \geq \frac{2}{\lambda^2} \ln\left(\frac{2}{\delta}\right)$, the probability that the error in the estimate is more than λ is,

$$\Pr[|\hat{f}(\beta) - a_\beta| \geq \lambda] \quad \leq \quad 2e^{-\lambda^2 m/2} \leq \delta.$$

Given that $|\hat{f}(\beta) - a_\beta| \leq \lambda$ then $\left(\hat{f}(\beta) - a_\beta\right)^2 \leq \lambda^2$, and

$$E[(f - a_\beta \chi_\beta)^2] = \sum_{\alpha \neq \beta} \hat{f}^2(\alpha) + \left(\hat{f}(\beta) - a_\beta\right)^2 \leq 1 - \hat{f}^2(\beta) + \lambda^2.$$

Recall that the original error was $1 - \hat{f}^2(\beta)$, given that we knew exactly the value of $\hat{f}(\beta)$. Hence, the "penalty" for estimating $\hat{f}(\beta)$ there is an additional error term of λ^2.

4.2 Low Degree Algorithm

In the previous section we showed how to approximate a single coefficient. For many classes of functions, each function can be approximated by considering only a small number of coefficients. Furthermore, those are the coefficients that correspond to small sets[1].

Assume f is defined *"mainly"* on the *"low"* coefficients. Formally, a function has an (α, d)-degree if $\sum_{S:|S|>d} \hat{f}^2(S) \leq \alpha$. The algorithm that approximates an (α, d)-degree function is the following.

- Sample m examples, $< x_i, f(x_i) >$. For each S, with $|S| \leq d$, compute $a_S = \frac{1}{m}\sum_{i=1}^{m} f(x_i)\chi_S(x_i)$, where $m \geq \frac{1}{2}\sqrt{\frac{n^d}{\varepsilon}} \cdot \ln\left(\frac{2n^d}{\delta}\right)$.
- Output the function $h(x)$,
$$h(x) \stackrel{\text{def}}{=} \sum_{|S|\leq d} a_S \chi_S(x).$$

Theorem 4.3 *Let f be an (α, d)-degree function. Then with probability $1-\delta$ the Low Degree Algorithm outputs a hypothesis h such that $E[(f-h)^2] \leq \alpha + \varepsilon$.*

Proof: First we claim that the algorithm approximates each coefficient within λ. More precisely,
$$\Pr[|a_S - \hat{f}(S)| \geq \lambda] \leq 2e^{-\lambda^2 m/2}$$
The error of $h(x)$ is bounded by,
$$E[(f-h)^2] = \alpha + \sum_{|S|\leq d} (\hat{f}(S) - a_S)^2 \leq \alpha + \sum_{|S|\leq d} \lambda^2 \leq \alpha + n^d \cdot \lambda^2.$$
We want to bound the error by $\alpha + \varepsilon$. Therefore,
$$n^d \cdot \lambda^2 \leq \varepsilon \Longrightarrow \lambda \leq \sqrt{\frac{\varepsilon}{n^d}}.$$
This needs to hold with probability $1-\delta$, therefore,
$$2e^{-\lambda^2 m/2} \cdot n^d \leq \delta,$$
which holds for
$$m \geq \frac{2n^d}{\varepsilon} \ln\left(\frac{2n^d}{\delta}\right).$$
□

[1]We call the coefficients of small sets the "low" coefficients, and the coefficients of large sets the "high" coefficients.

Note that we did not "really" use the low degree in any other way than to bound the size of the set of the "interesting" coefficients. We can use the same algorithm to learn any function that can be approximated by using a small set of coefficients which is *known in advance* to the algorithm. (In a similar way we would approximate each coefficient in the set separately, and the time complexity would be proportional to the number of coefficients.)

4.3 Learning Sparse Functions

In the previous section we showed how to learn a function by approximating its Fourier coefficients on sets that are smaller than a certain parameter d, with a running time of $O(n^d)$. However, in many cases most of the coefficients of sets that are smaller than d may still be negligible. Furthermore, it might be the case that a function can be approximated by a small number of coefficients, but the coefficients do not correspond to small sets.

In this section we describe a learning algorithm that learns the target function by finding a sparse function, i.e. a function with a small number of non-zero coefficients. The main advantage of the algorithm is that its running time is polynomial in the number of non-zero coefficients. However, the disadvantage is that the algorithm uses the query model. We start by defining a sparse function.

Definition 4.4 *A function f is* t-sparse *if it has at most t non zero Fourier coefficients.*

The main result in this section is that if f can be ε-approximated by some polynomially-sparse function g then there is a randomized polynomial time algorithm that finds a polynomially sparse function h that $O(\varepsilon)$-approximates f. This algorithm works in the query model and the approximation is with respect to the uniform distribution.

The first step is to show that if f can be approximated by a polynomially sparse function g, it can be approximated by a polynomially sparse function that has only "significant" coefficients. We remark that we do not make a "direct" use of g (e.g., by approximating g instead of approximating f) but only use its existence in the analysis.

Lemma 4.5 *If f can be ε-approximated by a t-sparse function g then there exists a t-sparse function h such that $E[(f-h)^2] \leq \varepsilon + \varepsilon^2/t$ and all the non-zero coefficients of h are at least ε/t.*

Proof: Let $\Lambda = \{S | \hat{g}(S) \neq 0\}$ and $\Lambda' = \Lambda \cap \{S| \ \ |\hat{f}(S)| \geq \frac{\varepsilon}{t}\}$. Since g is t-sparse, then $|\Lambda'| \leq |\Lambda| \leq t$. Let h be the function obtained from Λ' by taking the respective coefficients of f. Namely,

$$h(x) = \sum_{S \in \Lambda'} \hat{f}(S)\chi_S(x).$$

Clearly h is t-sparse. We now bound the value of $E[(f-h)^2]$ as follows,

$$E_x[(f(x)-h(x))^2] = \sum_{S\notin\Lambda} \hat{f}^2(S) + \sum_{S\in\Lambda-\Lambda'} \hat{f}^2(S).$$

The first term bounds $E[(f-g)^2]$ from below, since it includes all the coefficients that are not in the support of g. We bound the second term using the fact that $|\Lambda-\Lambda'| \leq t$, and for each $S \in \Lambda - \Lambda'$, it holds that $|\hat{f}(S)| \leq \frac{\varepsilon}{t}$. Hence

$$E_x[(f(x)-h(x))^2] \leq \varepsilon + (\frac{\varepsilon}{t})^2 t = \varepsilon + \varepsilon^2/t,$$

which completes the proof. □

The above lemma has reduced the problem of approximating f by a t-sparse function to the problem of finding all the coefficients of f that are greater than a threshold of ε/t. Note that the function h defined above does not necessarily contain all the coefficients of f which are greater than ε/t, but only those which appear also in g. However, adding these coefficients to h can only make h a better approximation for f. In any case, the number of these coefficients, as follows from Lemma 4.8 below, cannot be too high.

Our aim is to show a randomized polynomial time procedure that given a function f and a threshold θ outputs (with probability $1-\delta$) all the coefficients for which $|\hat{f}(z)| \geq \theta$. The procedure runs in polynomial time in n, $1/\theta$ and $\log 1/\delta$. (Having f means that the algorithm can use *queries*.)

The algorithm partitions the coefficients according to their prefix. Let

$$f(x) = \sum_{z\in\{0,1\}^n} \hat{f}(z)\chi_z(x).$$

For every $\alpha \in \{0,1\}^k$, we define the function $f_\alpha : \{0,1\}^{n-k} \to \mathbb{R}$ as follows:

$$f_\alpha(x) \triangleq \sum_{\beta\in\{0,1\}^{n-k}} \hat{f}(\alpha\beta)\chi_\beta(x)$$

In other words, the function $f_\alpha(x)$ includes all the coefficients $\hat{f}(z)$ of f such that z starts with α and no other coefficient (i.e. all other coefficients are 0). This immediately gives the key idea for how to find the significant coefficients of f: find (recursively) the significant coefficients of f_0 and f_1. During the learning process we can only query for the value of the target function f in certain points. Therefore, we first have to show that $f_\alpha(x)$ can be efficiently computed using such queries to f. Actually, we do not need to compute the exact value of $f_\alpha(x)$ but rather it is sufficient to approximate it. The following lemma gives an equivalent formulation of f_α, which is computationally much more appealing:

Lemma 4.6 *Let f be a function, $1 \leq k < n$, $\alpha \in \{0,1\}^k$ and $x \in \{0,1\}^{n-k}$, then*

$$f_\alpha(x) = E_{y\in\{0,1\}^k}[f(yx)\chi_\alpha(y)].$$

The above formulation implies that even though we cannot compute the value of $f_\alpha(x)$ exactly we can approximate it by approximating the above expectation.

Proof: Let $f(yx) = \sum_z \hat{f}(z)\chi_z(yx)$. Note that if $z = z_1 z_2$, where $z_1 \in \{0,1\}^k$, then $\chi_z(yx) = \chi_{z_1}(y)\chi_{z_2}(x)$. Therefore,

$$\begin{aligned} E_y[f(yx)\chi_\alpha(y)] &= E_y\left[\left(\sum_{z_1}\sum_{z_2}\hat{f}(z_1z_2)\chi_{z_1}(y)\chi_{z_2}(x)\right)\chi_\alpha(y)\right] \\ &= \sum_{z_1}\sum_{z_2}\hat{f}(z_1z_2)\chi_{z_2}(x)E_y[\chi_{z_1}(y)\chi_\alpha(y)] \end{aligned}$$

where y and z_1 are strings in $\{0,1\}^k$ and z_2 is in $\{0,1\}^{n-k}$. By the orthonormality of the basis, it follows that $E_y[\chi_{z_1}(y)\chi_\alpha(y)]$ equals 0 if $z_1 \neq \alpha$, and equals 1 if $z_1 = \alpha$. Therefore, only the terms with $z_1 = \alpha$ contribute in the sum. Thus, it equals,

$$E_y[f(yx)\chi_\alpha(y)] = \sum_{z_2 \in \{0,1\}^{n-k}} \hat{f}(\alpha z_2)\chi_{z_2}(x) = f_\alpha(x),$$

which completes the proof of the lemma. □

Since both $|f(x)| = 1$ and $|\chi_\alpha(y)| = 1$ we derive the following corollary on the value of $f_\alpha(x)$.

Corollary 4.7 *Let f be a boolean function, $1 \le k < n$, $\alpha \in \{0,1\}^k$ and $x \in \{0,1\}^{n-k}$, then*

$$|f_\alpha(x)| \le 1.$$

We showed how to decompose a function f into functions f_α, $\alpha \in \{0,1\}^k$, such that each coefficient of f appears in a unique f_α. Recall that our aim is to find the coefficients $\hat{f}(z)$ such that $|\hat{f}(z)| \ge \theta$. The next lemma claims that this cannot hold for "too many" values of z, and that the property $E[f_\alpha^2] \ge \theta^2$ cannot hold for "many" α (of length k) simultaneously.

Lemma 4.8 *Let f be a* boolean *function, and $\theta > 0$. Then,*

1. *At most $1/\theta^2$ values of z satisfy $|\hat{f}(z)| \ge \theta$.*
2. *For any $1 \le k < n$, at most $1/\theta^2$ functions f_α with $\alpha \in \{0,1\}^k$ satisfy $E[f_\alpha^2] \ge \theta^2$.*

Proof: By the assumption that f is a boolean function combined with Parseval's equality, we get

$$\sum_{z \in \{0,1\}^n} \hat{f}^2(z) = E[f^2] = 1.$$

Therefore, (1) immediately follows. Similarly, using the definition of f_α,

$$E[f_\alpha^2] = \sum_{\beta \in \{0,1\}^{n-k}} \hat{f}^2(\alpha\beta).$$

```
SUBROUTINE SA($\alpha$)
        IF $E[f_\alpha^2] \geq \theta^2$ THEN
                IF $|\alpha| = n$ THEN OUTPUT $\alpha$
                                   ELSE SA($\alpha 0$); SA($\alpha 1$);
```

Figure 1: Subroutine SA

Thus, if $|\hat{f}(\alpha\beta)| \geq \theta$, for some $\beta \in \{0,1\}^{n-k}$, then $E[f_\alpha^2] \geq \theta^2$. By the above two equalities the following holds,

$$\sum_{\alpha \in \{0,1\}^k} E[f_\alpha^2] = E[f^2] = 1.$$

Therefore, at most $1/\theta^2$ functions f_α have $E[f_\alpha^2] \geq \theta^2$, which completes the proof of (2). $\Box$

The Sparse Algorithm

By now the algorithm for finding the significant coefficients of a function f should be rather obvious. It is described by the recursive subroutine SA, appearing in Figure 1. We start the algorithm by calling SA(λ), where λ is the empty string.

As mentioned earlier in this section, we know that each coefficient of f_α appears in exactly one of $f_{\alpha 0}$ and $f_{\alpha 1}$. Also, if $|\hat{f}(\alpha\beta)| \geq \theta$, for some $\beta \in \{0,1\}^{n-k}$, then $E[f_\alpha^2] \geq \theta^2$ (note that when $|\alpha| = n$, then $E[f_\alpha^2] = \hat{f}^2(\alpha)$). Therefore, the algorithm outputs all the coefficients that are larger than θ.

By lemma 4.8, we also know that the number of α's for which $E[f_\alpha^2] \geq \theta^2$ is bounded by $1/\theta^2$, for each length of α. Thus, the total number of recursive calls is bounded by $O(n/\theta^2)$.

This is a sketch of the algorithm. Formally, we are still not done, since this algorithm assumes that we can compute $E[f_\alpha^2]$ exactly, something that is not achievable in polynomial time. On the other hand, we can approximate $E[f_\alpha^2]$ very accurately in polynomial time[2]. To conclude, the Sparse Algorithm achieves the following goal.

Theorem 4.9 *Let f be a boolean function such that there exists a t-sparse function g that ε-approximates f. Then there exists a randomized algorithm, that on input f and $\delta > 0$ outputs a function h, such that with probability $1-\delta$ the function h $O(\varepsilon)$-approximates, the input function f. The algorithm runs in time polynomial in n, t, $1/\varepsilon$ and $\log 1/\delta$.*

[2]We do not show the details of the algorithm and the interested reader is referred to [KM91].

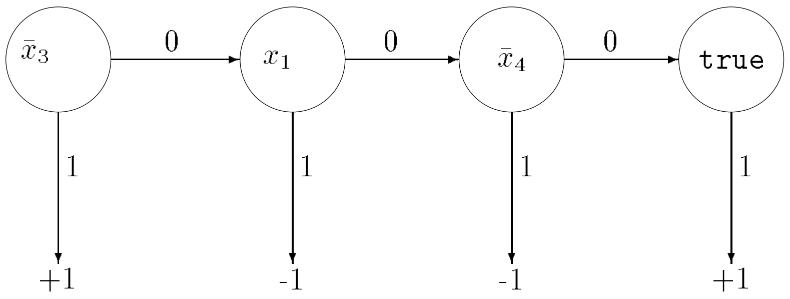

Figure 2: An example of a decision list

5 Properties of complexity classes

In this section we show various complexity class which can be learned through their Fourier spectrum. We show two kind of properties. The first is that the class can be approximated by considering the "low" coefficients, such functions can be learn by the Low Degree Algorithm. The other property is that the class can be approximated by a sparse function, such functions can be learn by the Sparse Algorithm.

5.1 Decision Lists

A *Decision List* L is a list of pairs $(b_1, \sigma_1), ..., (b_r, \sigma_r)$ where b_i is either a variable or its negation and $\sigma_i \in \{-1, +1\}$. (See Figure 2.) The last pair is the terminal pair, and b_r is viewed as the constant predicate **true**. A Decision List L defines a boolean function as follows. For an input x, $L(x)$ equals to σ_j where j is the *least* index such that $b_j(x) = 1$. (Such an index always exists, since the last predicate is always **true**.) It is clear that the length of L is at most $n + 1$, since if a variable appears twice we can either eliminate its second appearance or replace it by a terminal pair. (We should remark that such deterministic decision list are learnable by other methods, see [Riv87]. Our main aim of introducing this class here is to demonstrate how to use the techniques that we developed.)

Claim 5.1 *Given a decision list $L(x)$, there is a function $h(x)$ of at most t variables, such that $E[(L - h)^2] \le \varepsilon$, where $t \ge \log(\frac{4}{\varepsilon})$.*

Proof: Let $h(x)$ be the decision list that is a prefix of $L(x)$ up to the t-th node, and the terminal pair has the value of the next leaf in L. The decision list $h(x)$ may have redundant nodes at the end. In order to change h to a reduced form we can replace the maximal suffix of h, for which all the pairs that have the same value as the terminal pair, by the terminal pair.

Clearly h is correct on any input that terminates in one of the first t pairs of L. In the worst case $h(x)$ makes an error on any other input. Since only 2^{-t} of the inputs continue past the tth node in L, the probability that h and L disagrees is at most $\varepsilon/4$. Each disagreement contributes 4 to the error squared, which completes the proof. $\Box$

The above claim shows that we can concentrate only on a small set of variables, approximate the Fourier coefficients for every subset of this set and use the approximation to define the approximating function h. More precisely, let S be the set of t variables. For any $T \subset S$ we approximate the value of $\hat{f}(T)$, let a_T be the approximated value. The hypothesis we output is $\sum_{T\subset S} a_T \chi_T(x)$.

There is still a question of how to find this set of variables. One approach is to try any set of t variables by running the Low Degree algorithm. This results in a running time of $O(n^{\lceil \log 4/\varepsilon \rceil})$. Below we show a more efficient approach.

Claim 5.2 *Let $L(x)$ be a decision list. Let x_{j_i} be the variable in the ith node of $L(x)$, then*

$$|\hat{L}(\{j_i\})| \ \leq 2 \cdot 2^{-i}$$

Proof: By definition $\hat{L}(\{j_i\}) = E[L \cdot \chi_{\{j_i\}}]$. Let C_k be the inputs that reach the kth leaf. Clearly the C_ks are a partition of the inputs and the number of inputs in C_k is 2^{n-k}. For an input $x \in C_k$, for $1 \leq k < i$, let x' be the input x with the j_i bit flipped. The two inputs x and x' cancel each other influence on $\hat{L}(\{j_i\})$. Therefore we can ignore all the inputs that reach leaves before the ith leaf. The number of remaining inputs is $2 \cdot 2^{n-i}$, from which the claim follows. $\Box$

The algorithm for finding the *interesting set* approximates the coefficients of the single variables. If the coefficient is larger than $\varepsilon/2 = 22^{-t}$ the variable is in the set, otherwise it is out of the set. Clearly, each variable that we added is one of the t first variables in the list. We are guaranteed to add the variables of $h(x)$ (whose definition appear Claim 5.1), since their coefficients are at least $\varepsilon/2 = 22^{-t}$. After finding this set S the algorithm approximates all the coefficients of sets $T \subset S$.

5.2 Decision Trees

A *Decision Tree* is a binary tree where each internal node is labeled with a variable, and each leaf is labeled with either $+1$ or -1. Each decision tree defines a boolean function as follows. An assignment to the variables determines a unique path from the root to a leaf: at each internal node the left (respectively right) edge to a child is taken if the variable named at that internal node is 0 (respectively 1) in the assignment. The value of the function at the assignment is the value at the leaf reached. (See Figure 3.) The *depth* of a decision tree is the length of the longest path from the root to a leaf and denoted by `DT-depth`(T). For a function f we denote by `DT-depth`(f) the minimum depth of a decision tree that computes f. Note that every boolean function on n variables, can be represented by a decision tree with at most 2^n nodes and depth at most n.

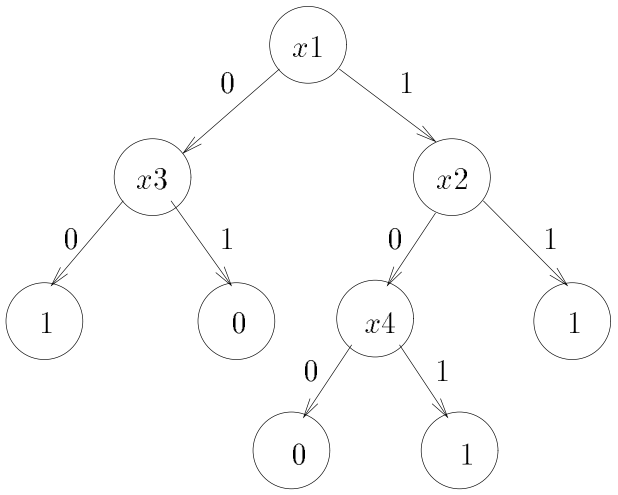

Figure 3: A decision tree of depth 3, the concept so defined is equivalent to the Boolean formula: $(\bar{x_1}\ \bar{x_3})\ \vee\ (x_1\ \bar{x_2}\ x_4)\ \vee\ (x_1\ x_2)$.

We show, similar to the case in decision lists, that decision trees also can be approximated by considering only the "low" coefficients. In this case the low coefficients are coefficients of set of at most logarithmic size in the number of nodes in the tree.

Claim 5.3 *Let T be a decision tree with m nodes. There exists a function h such that all the Fourier coefficients of h on sets larger than t are 0, where $t = \log m/\varepsilon$, and $E[(T-h)^2] \leq \varepsilon$.*

Proof: Let h be the decision tree that results from truncating T at depth t. At the truncated places we add a leaf with value $+1$. The probability of reaching one of the leaves we added is ε, therefore $\Pr[T \neq h] \leq \varepsilon$.

We need to show that $\hat{h}(S) = 0$, for any S such that $|S| > t$. Since h is a decision tree, we can write a term for each leaf[3], and we are guaranteed that exactly one term is `true` for each input. Let $term_i$ be the terms, then we can write $h(x) = \sum_{i=1}^{m} \sigma_i term_i(x)$. Each term has at most t variables, hence, its largest non-zero coefficient is of a set of size at most t. In other words, for each $term_i(x)$, all the coefficients of sets larger than t are zero. Since h is a linear combination of such terms all its coefficients of sets larger than t are zero. $\square$

From the above claim we see that it is sufficient to approximate the coefficients of size at most t to approximate the decision tree. This can be done by running the Low Degree algorithm, which results in a running time of $O(n^{\log m/\varepsilon})$.

In what follows we show that in fact a decision tree can be approximated by a sparse function, where the number of coefficients is polynomial in the size of the decision tree.

[3]The term will be the conjunction of the variables on the path from the root to the leaf (with the appropriate variables negated), such that the term is true iff the input reaches this leaf.

First, we show that function with small L_1 (to be define latter) can be approximated by sparse functions. Later we show that decision trees are included in this class.

5.2.1 Learning Functions with the norm L_1

We start by defining the L_1 norm of a function.

Definition 5.4 *Let $L_1(f)$ be,*

$$L_1(f) = \sum_{S \subseteq \{1,\ldots,n\}} |\hat{f}(S)|.$$

The next Lemma shows that if $L_1(f)$ is "small" then f can be approximated by a "few" coefficients.

Theorem 5.5 *For every boolean function f there exists a function h, such that h is $\frac{(L_1(f))^2}{\varepsilon}$-sparse and,*

$$E[(f-h)^2] \le \varepsilon.$$

Proof: Let Λ be,

$$\Lambda = \{S \ \mid\ |\hat{f}(S)| \ge \frac{\varepsilon}{L_1(f)}\}$$

Intuitively, Λ includes all the "significant" coefficients. From the bound of the sum of the coeffiecnts in absolute value (i.e. $L_1(f)$) it follows Λ size is bounded as follows,

$$|\Lambda| \le \frac{L_1(f)}{\varepsilon/L_1(f)} = \frac{(L_1(f))^2}{\varepsilon}$$

We define the function h to be,

$$h(\vec{x}) = \sum_{S \in \Lambda} \hat{f}(S)\chi_S(\vec{x}).$$

Using the Parseval identity we bound the error as follows,

$$E[(f-h)^2] = \sum_{S \notin \Lambda} \left(\hat{f}(S) - \hat{h}(S)\right)^2 = \sum_{S \notin \Lambda} \left(\hat{f}(S)\right)^2 \le \overbrace{\max_{S \notin \Lambda} |\hat{f}(S)|}^{\le \frac{\varepsilon}{L_1(f)}} \cdot \overbrace{\sum_{S \notin \Lambda} |\hat{f}(S)|}^{\le L_1(f)} \le \varepsilon.$$

The function h has $|\Lambda|$ non-zero coefficients, and the theorem follows from the bound on the size of Λ. □

The above theorem defines a wide class of functions that can be learned efficiently using the Sparse Algorithm. Namely, if $L_1(f)$ is bounded by a polynomial then f can be learned in polynomial time by the Sparse Algorithm. In the next section we show that decision trees belong to this class of functions.

The following two simple claims would be helpful in bounding the L_1 norm of functions.

Claim 5.6 $L_1(f) + L_1(g) \geq L_1(f+g)$

Claim 5.7 $L_1(f) \cdot L_1(g) \geq L_1(f \cdot g)$

5.2.2 Sparse approximation of Decision Trees

Our aim is to show that decision trees have a small $L_1(f)$. This implies that decision trees can be approximated by a sparse function, and hence are learnable by the Sparse Algorithm. The following function would be helpful in describing decision trees.

Definition 5.8 *For $d \geq 1$ and a boolean vector $\vec{b} \in \{0,1\}^d$ we define the function $AND_{(b_1,\ldots,b_d)} : \{0,1\}^d \rightarrow \{0,1\}$ to be,*

$$AND_{(b_1,\ldots,b_d)}(\vec{x}) = \begin{cases} 1 & \forall i,\ 1 \leq i \leq d \quad b_i = x_i \\ 0 & Otherwise \end{cases}$$

Lemma 5.9 *For $d \geq 1$ and $(b_1,\ldots,b_d) \in \{0,1\}^d$,*

$$L_1(AND_{(b_1,\ldots,b_d)}) = 1$$

Proof: Rewrite the function $AND_{(b_1,\ldots,b_d)}$ as,

$$AND_{(b_1,\ldots,b_d)}(x_1,\ldots,x_d) = \prod_{i=1}^{d} \left(\frac{1 + (-1)^{b_i}\chi_i(x)}{2} \right).$$

Using Claim 5.7,

$$L_1(AND_{(b_1,\ldots,b_d)}) \leq \prod_{i=1}^{d} L_1 \left(\frac{1 + (-1)^{b_i}\chi_i(x)}{2} \right) \leq 1.$$

Since $AND_{(b_1,\ldots,b_d)}(b_1,\ldots,b_d) = 1$, then the sum of the coefficients is at least 1, hence $L_1(AND) = 1$. $\square$

The following theorem bounds the L_1 norm of decision trees as a function of the number of nodes in the tree.

Theorem 5.10 *Let $f(\vec{x})$ be a function represented by a decision tree T. If T has m leaves, then,*

$$L_1(f) \leq m.$$

Proof: Given a leaf v of T, we look at path from the root of the tree to v. There are d_v nodes in this path. Let $x_{i_1}, \ldots, x_{i_{d_v}}$ be the variables on this path. There is a unique assignment of values to $x_{i_1}, \ldots, x_{i_{d_v}}$ such that f reaches the leaf v. Let $b_1^v, \ldots, b_{d_v}^v$ be this assignment. Then,

$$\forall \vec{x} \in \{0,1\}^n \quad AND_{(b_1^v,\ldots,b_{d_v}^v)}(x_{i_1}, \ldots, x_{i_{d_v}}) = 1 \quad \Leftrightarrow \quad f(\vec{x}) \ \mathit{arrive\ to\ the\ leaf}\ v.$$

The $AND_{(b_1^v,\ldots,b_{d_v}^v)}$ function identifies all the inputs arriving to the leaf v. For every $v \in T$ let σ_v be the value returned by the leaf v. Since on any input f reaches exactly one leaf, it follows that,

$$\forall \vec{x} \in \{0,1\}^n \quad f(\vec{x}) = \sum_{v \in Leaves} \sigma_v \cdot AND_{(b_1^v,\ldots,b_{d_v}^v)}(x_{i_1^v}, \ldots, x_{i_{d_v}^v}).$$

Finally, by Claim 5.6 and Lemma 5.9,

$$L_1(f) \leq \sum_{v \in T} L_1(AND) \leq m \cdot L_1(AND) = m,$$

which completes the proof of the theorem. □

Remark: If in the internal nodes in the decision tree we allow to query the value of a parity of a subset of the input, i.e. the value of $\chi_S(\vec{x})$, then still $L_1(f)$ is bounded by the number of leaves. The proof is along the same lines and is derived by defining,

$$AND_{(b_1,\ldots,b_d,\chi_{\alpha_1},\ldots\chi_{\alpha_d})}(\vec{x}) = 1 \qquad \Longleftrightarrow \forall i \ \ \chi_{\alpha_i}(\vec{x}) = b_i,$$

and noting that $L_1(AND_{(b_1,\ldots,b_d,\chi_{\alpha_1},\ldots\chi_{\alpha_d})})$ =1.

A more general case is that the nodes in the decision tree includes arbitrary predicates. [Bel92] derives a general method for bounding the L_1 of the decision tree as a function of L_1 of the predicates in the nodes.

Open Problem: Is there a polynomial time algorithm that, given the Fourier representation of a decision tree, outputs a decision tree for that function.

5.3 Boolean Circuits

A formula is said to be in *conjunctive normal form* (CNF) if it is the conjunction of clauses, and to be in *disjunctive normal form* (DNF) if it is the disjunction of terms. The size of a term is the number of variables in the term. For example, $x_1 x_2 \vee x_3 \overline{x}_4 x_5$ is in DNF.

A depth of a circuit is the maximum distance from an input to the output. Both DNF and CNF can be view as a two level circuits. The class of AC^0 includes circuits of polynomial size with AND, NOT an OR of unbounded fan-in and constant depth.

First we consider a property of the "large" Fourier coefficients of a DNF with terms of size d. We show that the sum of squares of the coefficients, that correspond to sets larger than a certain threshold $\tau(d)$ is bounded by ε. This implies that when approximating such a DNF, we can ignore the coefficients of sets larger than $\tau(d)$, and use the Low Degree Algorithm. Later we show that there is a better sparse approximation for a DNF.

We introduce the notion of a restriction and a random restriction. A restriction ρ is a mapping of the input variables to 0, 1 and $*$. The function obtained from $f(x_1, \cdots, x_n)$ by applying a restriction ρ is denoted by f_ρ. The inputs of f_ρ are those x_i for which $\rho(x_i) = *$, while all other variables are set according to ρ.

The set of *live variables* with respect to a restriction ρ is the set of variables that is assigned the value $*$, this set is denoted by $live(\rho) = \{x_i | \rho(x_i) = *\}$. A *random restriction ρ with a parameter p* is obtained by setting each x_i, independently, to a value from $\{*, 0, 1\}$, such that $\Pr[\rho(x_i) = *] = p$, and $\Pr[\rho(x_i) = 1] = \Pr[\rho(x_i) = 0] = \frac{1-p}{2}$.

We base our proof on the following two lemmas. The proof of the first lemma appears in appendix A.

Lemma 5.11 *Let f be a boolean function and ρ a random restriction with parameter p. Then,*

$$\sum_{|S|>t} \hat{f}^2(S) \leq 2 \Pr_\rho[\texttt{DT-depth}(f_\rho) \geq tp/2].$$

Note that the above lemma is non-trivial only if $\Pr_\rho[\texttt{DT-depth}(f_\rho) \geq tp/2] \leq \frac{1}{2}$.

The property of DNF that we would use is based on random restriction. The following lemma, from [Has86], states that a DNF after a random restriction can be described by a small decision tree.

Lemma 5.12 (Hastad) *Let f be given by a DNF formula where each term has size at most d, and a random restriction ρ with parameter p (i.e. $\Pr[\rho(x_i) = *] = p$). Then,*

$$\Pr_\rho[\texttt{DT-depth}(f_\rho) \geq s] \leq (5pd)^s.$$

Based on the above two lemmas we show the following lemma.

Lemma 5.13 *Let f be a function that can be written by a DNF with terms of size d. Then,*

$$\sum_{|S|>20d\log\frac{2}{\varepsilon}} \hat{f}^2(S) \leq \varepsilon$$

Proof: Combining Lemma 5.11 with Lemma 5.12 and setting $p = 1/10d$, $t = 20d\log\frac{2}{\varepsilon}$ and $s = tp/2 = \log\frac{2}{\varepsilon}$ gives,

$$\sum_{|S|>t} \hat{f}^2(S) \leq 2(5pd)^s = 2(\frac{1}{2})^s = \varepsilon.$$

□

The above Lemma demonstrates that in order to approximate a DNF with terms of size d, it is sufficient to consider the coefficients of sets of size at most $\tau = O(d \log \frac{1}{\varepsilon})$. Using the Low Degree Algorithm we can learn this class in $O(n^\tau)$ time. Later we show how this class can be better approximated using a sparse function, which would result in a significantly improved running time.

5.3.1 Approximation the AC^0 Class

The class AC^0 can be viewed as a generalization of DNF. It consists of circuits composed from AND, OR and NOT gates with unbounded fan-in, where the number of gates is polynomial in the number of inputs and the depth of the circuit is constant.

The following lemma is from [Has86], and can be derived by repeated applications of Lemma 5.12.

Lemma 5.14 (Hastad) *Let f be an AC^0 circuit with M gates and depth d. Then,*

$$\Pr[\texttt{DT-depth}(f_\rho) \geq s] \leq M 2^{-s},$$

where ρ is a random restriction with parameter $p \leq \frac{1}{10^d s^{d-1}}$.

Choosing the parameters $p = 1/(10 t^{d-1/d})$ and $s = \frac{pt}{2} = t^{1/d}/20$, and applying Lemma 5.11, we have the following theorem.

Theorem 5.15 *Let f be an AC^0 circuit with M gates and depth d. Then,*

$$\sum_{|A| \geq t} \hat{f}^2 \leq M 2^{-\frac{t^{\frac{1}{d}}}{20}}.$$

For $t = (20 \log \frac{M}{\varepsilon})^d$ the sum is bounded by ε. Thus, running the Low Degree Algorithm results in time complexity of $O(n^{poly-log(n)})$.
Remark: From the result of [Kha93] we cannot hope to get a better running time than $O(n^{poly-log(n)})$, unless some cryptographic assumption about factoring is false.

5.3.2 Sparse approximation of DNF

We show that DNF with "small" terms can be approximated by a sparse function.

Theorem 5.16 *For any function f that can be described by a DNF with terms of size d there exists an M-sparse function g that ε-approximates f and $M \leq d^{O(d \log \frac{1}{\varepsilon})}$.*

The proof of the above theorem is based on combining two lemmas. The first is Lemma 5.13, that shows that the coefficients of "large" sets are negligible. The second is Lemma 5.17 (proved in Appendix B), in which we restrict our attention to coefficients of sets of size at most τ. We show that the sum of the absolute values of the coefficients of all the sets of size at most τ is bounded by $d^{O(\tau)}$.

Lemma 5.17 *If a function f can be described by a DNF with terms of size d then*

$$\sum_{S:|S|\leq\tau} |\hat{f}(S)| \leq 4(20d)^{\tau} = d^{O(\tau)}$$

Based on the above lemma we prove Theorem 5.16.

Proof of Theorem 5.16: Given a function f, that is described by a DNF with terms of size d, we need to exhibit a function g that ε-approximates f. Let $\tau = 20d\log\frac{4}{\varepsilon}$.

Define g' to be the function whose Fourier coefficients of sets less than τ are identical to those of f and the Fourier coefficients of sets larger than τ are zero. By Lemma 5.13 $E[(f-g')^2] \leq \varepsilon/2$.

Lemma 5.17 gives a property of sets less than τ. This property shows that the sum, in absolute value, of the coefficients of the "small" sets, is small, specifically, $L_1(g') \leq d^{O(\tau)}$. By Theorem 5.5, there exists a function g with at most $\frac{(2d^{O(\tau)})^2}{\varepsilon}$ non zero coefficients, such that $E[(g'-g)^2] \leq \varepsilon/2$, which concludes the proof of the theorem. □

A major open problem is Computational Learning Theory is the complexity of learning a DNF with a polynomial number of terms. We offer here a conjecture, that if resolved in the affirmative, implies that the Sparse Algorithm learns polynomial size DNF efficiently.

Conjecture 5.18 *Any DNF with at most m terms can be ε-approximated by a t-sparse function, where $t = m^{O(\log\frac{1}{\varepsilon})}$.*

Acknowledgement

I would like to thank Nader Bshouty, Eyal Kushilevitz, Alon Orlitrsky and Dana Ron for their helpful comments on this manuscript.

References

[Ajt83] M. Ajtai. $\sum_1^1$-formulae on finite structure. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[AM91] William Aiello and Milena Mihail. Learning the fourier spectrum of probabilistic lists and trees. In *Proceedings SODA 91*, pages 291–299. ACM, Jan 1991.

[Bel92] Mihir Bellare. A technique for upper bounding the spectral norm with applications to learning. In 5^{th} *Annual Workshop on Computational Learning Theory*, pages 62–70, July 1992.

[BHO90] Y. Brandman, J. Hennessy, and A. Orlitsky. A spectral lower bound technique for the size of decision trees and two level circuits. *IEEE Trans. on Computers.*, 39(2):282–287, 1990.

[Bru90] J. Bruck. Harmonic analysis of polynomial threshold functions. *Siam J. on Disc. Math.*, 3(2):168–177, May 1990.

[BS90] J. Bruck and R. Smolensky. Polynomial threshold functions, AC^0 functions and spectral norms. In 31^{th} *Annual Symposium on Foundations of Computer Science, St. Louis, Missouri*, pages 632–641, October 1990.

[FJS91] Merrick L. Furst, Jeffrey C. Jackson, and Sean W. Smith. Improved learning of AC^0 functions. In 4^{th} *Annual Workshop on Computational Learning Theory*, pages 317–325, August 1991.

[FSS84] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st ACM Symposium on Theory of Computing*, pages 25–32. ACM, 1989.

[Has86] J. Hastad. *Computational limitations for small depth circuits.* MIT Press, 1986. Ph.D. thesis.

[HR89] Torben Hagerup and Christine Rub. A guided tour to chernoff bounds. *Info. Proc. Lett.*, 33:305–308, 1989.

[Kha93] Michael Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of STOC '93*, pages 372–381. ACM, 1993.

[KKL88] J. Kahn, G. Kalai, and N. Linial. The influence of variables on boolean functions. In 29^{th} *Annual Symposium on Foundations of Computer Science, White Plains, New York*, pages 68–80, October 1988.

[KM91] E. Kushilevitz and Y. Mansour. Learning decision trees using the fourier spectrum. In *Proceedings of the* 23^{rd} *Annual ACM Symposium on Theory of Computing*, pages 455–464, May 1991. (To appear in Siam J. on Computing.)

[LMN89] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier Transform and learnability. In 30^{th} *Annual Symposium on Foundations of Computer Science, Reseach Triangle Park, NC*, pages 574–579, October 1989.

[Man92] Yishay Mansour. An $O(n^{\log\log n})$ learning algorihm for DNF under the uniform distribution. In *Workshop on Computational Learning Theory*, pages 53–61, July 1992.

[Riv87] Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

[SB91] Kai-Yeung Siu and Jehoshua Bruck. On the power of threshold circuits with small weights. *Siam J. on Disc. Math.*, 4(3):423–435, Aug 1991.

[Yao85] A. C. Yao. Separating the polynomial-time hierarchy by oracles. In 26^{th} *Annual Symposium on Foundations of Computer Science, Portland, Oregon*, pages 1–10, October 1985.

A Proof of Lemma 5.11

We want to prove that for any boolean function f:

$$\sum_{|S|>t} \hat{f}^2(S) \leq 2 \Pr_{\rho}[\texttt{DT-depth}(f_\rho) \geq tp/2]$$

Define $live(A) = |\{x_i : x_i \in A, \rho(x_i) = *\}|$. Recall from section 5.2 that $\texttt{DT-depth}(f) \leq k$ implies for every S, such that $|S| > k$, then $\hat{f}(S) = 0$. Therefore a sufficient statement is,

$$\sum_{|S|>t} \hat{f}^2(S) \leq 2 \Pr_{\rho}[\exists A : live(A) \geq \frac{tp}{2} \text{ and } \hat{f}_\rho(A) \neq 0]$$

Lemma A.1 *Let f be a boolean function and ρ a random restriction with parameter p. Then,*

$$E_\rho[\sum_{live(A)\geq \frac{tp}{2}} \hat{f}^2_\rho(A)] \leq \Pr_{\rho}[\exists A : live(A) \geq \frac{tp}{2} \textit{ and } \hat{f}_\rho(A) \neq 0].$$

Proof: Let $Ind(\rho)$ be 1 if there exists a set A, such that $live_\rho(A) \geq \frac{tp}{2}$ and $\hat{f}_\rho(A) \neq 0$. We can rewrite the probability as,

$$\Pr_{\rho}[\exists A : live(A) \geq \frac{tp}{2} \text{ and } \hat{f}_\rho(A) \neq 0] = \sum_{\rho} \Pr[\rho] Ind(\rho).$$

Consider a restriction ρ for which $Ind(\rho) = 1$. Since,

$$0 \leq \sum_{live(A)\geq \frac{tp}{2}} \hat{f_\rho}^2(A) \leq 1,$$

then,

$$Ind(\rho) = 1 = \sum_{A} \hat{f}^2(A) \geq \sum_{live(A)\geq \frac{tp}{2}} \hat{f}^2_\rho(A).$$

For a random restriction for which $Ind(\rho) = 0$, there is no A such that $live(A) \geq \frac{tp}{2}$ and $\hat{f}_\rho(A) \neq 0$. Hence, the sum $\sum_{live(A)\geq \frac{tp}{2}} \hat{f}^2_\rho(A) = 0$, therefore,

$$Ind(\rho) = 0 = \sum_{live(A)\geq \frac{tp}{2}} \hat{f}^2_\rho(A).$$

We showed that for any restriction ρ,

$$Ind(\rho) \geq \sum_{live(A)\geq \frac{tp}{2}} \hat{f}^2_\rho(A).$$

Since this holds for any restriction ρ, the lemma follows. □

Claim A.2 *Let f be a boolean function, then*

$$\sum_{|A|>t} \hat{f}^2(A) \leq 2E_S[\sum_{|A\cap S|\geq \frac{tp}{2}} \hat{f}^2(A)],$$

where S is a random set such that the probability that $i \in S$ is p.

Proof: Using Chernoff bounds, for $tp > 8$ it holds that $\Pr[|A \cap S| > \frac{tp}{2}] > \frac{1}{2}$ and the claim follows. $\square$

Combining Lemma A.1 and Claim A.2, it is sufficient to prove that,

$$E_S[\sum_{|A\cap S|\geq \frac{tp}{2}} \hat{f}^2(A)] = E_\rho[\sum_{live(A)\geq \frac{tp}{2}} \hat{f}_\rho^2(A)].$$

Definition A.3 *Let $f_{S^c \leftarrow x}$ be the function f_ρ, where the restriction ρ has as the live variables the set S and the other variables (i.e. those in S^c) are assigned a value according to x.*

A restriction ρ maps many coefficients of f to the same coefficient α in f_ρ, thus the mapping is not unique. The following lemma states that the sum of squares of coefficients which are being mapped to α, is the same as the expected value of the coefficient square at α. Since the order of variables has no meaning we can permute the variables such that all variables of S appear first.

Lemma A.4 *For $S = \{1, \ldots, k\}$ and $\alpha \in \{0,1\}^k$,*

$$\sum_{\beta\in\{0,1\}^{n-k}} \hat{f}^2(\alpha\beta) = E_{x\in\{0,1\}^{n-k}}[\hat{f}^2_{S^c\leftarrow x}(\alpha)]$$

Proof: We use the tools developed in section 4.3. Recall that $f_\alpha(x) = \sum_\beta \hat{f}(\alpha\beta)\chi_S(x)$ when $||\alpha|| = k$ and $||x|| = ||\beta|| = n - k$. Lemma 4.6 states that,

$$f_\alpha(x) = E_y[f(xy)\chi_\alpha(y)].$$

Using this we get that,

$$\sum_\beta \hat{f}^2(\alpha\beta) = E_x[f_\alpha^2(x)] = E_x[{E_y}^2[f(xy)\chi_S(y)]] = E_x[\hat{f}^2_{S^c\leftarrow x}(\alpha)].$$

$\square$

Lemma A.5 *Let f be a boolean function and ρ a random restriction with parameter p. Then,*

$$E_S[\sum_{|A\cap S|\geq \frac{tp}{2}} \hat{f}^2(A)] = E_\rho[\sum_{live(A)\geq \frac{tp}{2}} \hat{f}_\rho^2(A)],$$

where S is a random set such that the probability that $i \in S$ is p.

Proof: From Lemma A.4 we have that

$$\sum_{\beta} \hat{f}^2(\alpha\beta) = E_x[\hat{f}^2_{S^c\leftarrow x}(\alpha)].$$

Let α be the charateristic vector of the set $A \subset S$ and β the charateristic vector of the set $B \subset S^c$, then,

$$\sum_{B\subset S^c} \hat{f}^2(A\cup B) = E_x[\hat{f}^2_{S^c\leftarrow x}(A)].$$

Summing over all A of size greater than k gives that,

$$\sum_{\substack{A\subset S\\ |A|>k}} \sum_{B\subset S^c} \hat{f}^2(A\cup B) = \sum_{\substack{A\subset S\\ |A|>k}} E_x[\hat{f}^2_{S^c\leftarrow x}(A)].$$

Averaging over S maintain the identity,

$$E_S[\sum_{|A\cap S|\geq k} \hat{f}^2(A)] = E_S E_x[\sum_{\substack{A\subset S\\ |A|>k}} \hat{f}^2_{S^c\leftarrow x}(\alpha)].$$

Since $E_S E_x$ is simply E_ρ and setting $k = \frac{tp}{2}$, then,

$$E_S[\sum_{|A\cap S|\geq \frac{tp}{2}} \hat{f}^2(A)] = E_\rho[\sum_{live(A)\geq \frac{tp}{2}} \hat{f}^2_\rho(A)].$$

From the last equality, the correctness of lemma follows. $\square$

Proof of Lemma 5.11: From Lemma A.1 we have that,

$$E_\rho[\sum_{live(A)\geq \frac{tp}{2}} \hat{f}^2_\rho(A)] \leq \Pr_\rho[\exists A : live(A) \geq \frac{tp}{2} \text{ and } \hat{f}_\rho(A)\neq 0] = \Pr_\rho[\texttt{DT-depth}(f_\rho) \geq tp/2].$$

By Lemma A.5 we have that

$$E_S[\sum_{|A\cap S|\geq \frac{tp}{2}} \hat{f}^2(A)] = E_\rho[\sum_{live(A)\geq \frac{tp}{2}} \hat{f}^2_\rho(A)].$$

By Claim A.2 we have,

$$\sum_{|A|>t} \hat{f}^2(A) \leq 2E_S[\sum_{|A\cap S|\geq \frac{tp}{2}} \hat{f}^2(A)].$$

The lemma follows from combining the three above expressions. $\square$

B Proof of Lemma 5.17

In this proof we focus on coefficients of small sets. While any specific coefficient of a set of size less than $\tau = 20d\log 4/\varepsilon$ can potentially be "significant", in order to achieve a good approximation, we show that only a relatively small number of such coefficients can be simultaneously "significant". This is done by bounding the sum in absolute value of those coefficients. In the derivation of the bounds we use the following definitions.

Definition B.1 *Let*

$$L_{1,k}(f) = \sum_{|S|=k} |\hat{f}(S)|,$$

and

$$L_1(f) = \sum_{i=0}^{n} L_{1,k}(f) = \sum_{S} |\hat{f}(S)|$$

Our main aim is to bound $L_{1,k}(f)$ by $d^{O(k)}$, where d is the size of the largest term in a DNF representation of f.

The proof uses the fact that after a random restriction, the restricted DNF can be written as a decision tree with a small depth. Using Theorem 5.10 which states that $L_1(f) \leq m$, where m is the number of nodes in the decision tree that computes f, we bound the $L_1(f)$ as a function of its depth.

Claim B.2 *For a function f, if* $\texttt{DT-depth}(f) \leq s$ *then* $L_1(f) \leq 2^s$*.*

Proof: Since $\texttt{DT-depth}(f) \leq s$, the number of leaves is at most 2^s and the Claim follows from Theorem 5.10 that the bounds the $L_1(f)$ of decision trees by the number of leaves in the tree. □

We start by showing that after a random restriction the L_1 norm of the restricted function is very small.

Lemma B.3 *Let f be given by a DNF formula where each clause has size at most d. Let ρ be random restriction with parameter $p \leq \frac{1}{20d}$, then*

$$E_\rho[L_1(f_\rho)] \leq 2.$$

Proof: We can express the expectation as,

$$E_\rho[L_1(f_\rho)] = \sum_{s=0}^{n} \Pr[\texttt{DT-depth}(f_\rho) = s] \cdot E_\rho[L_1(f_\rho) \mid \texttt{DT-depth}(f_\rho) = s].$$

By Lemma B.2, for any ρ, such that $\texttt{DT-depth}(f_\rho) = s$, then $L_1(f_\rho) \leq 2^s$. By Lemma 5.12

$$\Pr[\texttt{DT-depth}(f_\rho) \geq s] \leq (5dp)^s$$

Therefore,

$$E_\rho[L_1(f)] \leq \sum_{s=0}^{n} (5pd)^s 2^s = \sum_{s=0}^{n} (10pd)^s.$$

For $p \leq \frac{1}{20d}$, the sum is less than 2, and the lemma follows. □

The next lemma establishes the connection between $L_{1,k}(f)$ and the value of $E_\rho[L_{1,k}(f_\rho)]$.

Lemma B.4 *Let f be a boolean function and ρ a random restriction with parameter p (i.e. $\Pr[x_i = *] = p$), then*

$$L_{1,k}(f) \leq (\frac{1}{p})^k E_\rho[L_{1,k}(f_\rho)]$$

Proof: Consider a random variable $\mathcal{L} \subset \{x_1 \cdots x_n\}$, such that for each x_i, independently, $\Pr[x_i \in \mathcal{L}] = p$. The random variable $\mathcal{L}$ is the set of live variables in a random restriction with parameter p. We can rewrite $L_{1,k}$ in the following way.

$$L_{1,k}(f) = \sum_{|S|=k} |\hat{f}(S)| = (\frac{1}{p})^k E_{\mathcal{L}}[\sum_{S \subset \mathcal{L} \& |S|=k} |\hat{f}(S)|].$$

Note that in both summations we are summing the original coefficients of the function.

Consider an an arbitrary choice for $\mathcal{L}$ and a subset $S \subset \mathcal{L}$.

$$\begin{aligned} |\hat{f}(S)| &= |E_{x_1,\ldots,x_n}[f(x_1,\ldots,x_n)\chi_S(x_1,\ldots,x_n)]| \\ &\leq E_{x_i \notin \mathcal{L}} |E_{x_j \in \mathcal{L}}[f(x_1,\ldots,x_n)\chi_S(x_1,\ldots,x_n)]| \\ &= E_\rho[|\hat{f}_\rho(S)| \mid live(\rho) = \mathcal{L}]. \end{aligned}$$

The last equality follows from the observation that averaging over $x_i \notin \mathcal{L}$ is the same as taking the expectation of a random restriction whose set of live variables is restricted to be $\mathcal{L}$. Since the absolute value of every coefficient S is expected to increase, this implies that,

$$\sum_{S \subset \mathcal{L} \& |S|=k} |\hat{f}(S)| \leq E_\rho[\sum_{S \subset \mathcal{L} \& |S|=k} |\hat{f}_\rho(S)| \mid live(\rho) = \mathcal{L}] = E_\rho[L_{1,k}(f_\rho) \mid live(\rho) = \mathcal{L}].$$

Now we can go back and use the first equality we derived. In that equality we are averaging over $\mathcal{L}$. Therefore,

$$L_{1,k}(f) = (\frac{1}{p})^k E_{\mathcal{L}}[\sum_{S \subset \mathcal{L} \& |S|=k} |\hat{f}(S)|] \leq (\frac{1}{p})^k E_\rho[L_{1,k}(f_\rho)],$$

which completes the proof of the lemma. □

We can now prove Lemma 5.17.

Proof of Lemma 5.17: Note that $\sum_{S:|S| \leq \tau} |\hat{f}(S)| = \sum_{k=0}^{\tau} L_{1,k}(f)$. By setting $p = \frac{1}{20d}$, and combining Claim B.3 and Lemma B.4, we have that

$$L_{1,k}(f) \leq 2(20d)^k$$

and the lemma follows. □