Deterministic Learning of DNF

by

Young Shin Oh

B.Sc., University of British Columbia, 2011

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master Of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2020

Declaration of Committee

Name: **Young Shin Oh**

Degree: **Master Of Science (Computing Science)**

Thesis title: **Deterministic Learning of DNF**

Committee: **Chair:** Oliver Schulte
Professor, Computing Science

Valentine Kabanets
Supervisor
Professor, Computing Science

Andrei Bulatov
Committee Member
Professor, Computing Science

Igor Shinkar
Examiner
Assistant Professor, Computing Science

Abstract

Our main result is a deterministic learning algorithm with membership queries that approximately learns a DNF on n variables with $\mathsf{poly}(n)$ number of terms to within an additive approximation error ϵ in time $n^{\tilde{O}\left(\log(n)\cdot\log^2(1/\epsilon)\right)}$. With random examples under the uniform distribution, the learning algorithm of [LMN93] for DNFs runs in time $n^{O(log^2(n/\epsilon))}$.

Our approach is to consider the Fourier expansion of the target DNF and approximate the heavy Fourier coefficients. Our hypothesis is the sign of the sparse polynomial that is defined with the approximated coefficients. We present two approaches for building our sparse polynomial.

First, we use Gopalan and Meka's [GMR13] PRG to deterministically approximate small degree coefficients of our target DNF. Second, we generalize the result of [DETT10] to show that a general DNF can be fooled by a small biased set to approximate coefficients of any degree.

We also present a derandomized Goldreich and Levin's algorithm for DNFs under the assumption that there exists an ϵ-PRG of seed length $\log(n/\epsilon)$ exists. This yields a deterministic learning algorithm for DNFs that runs in time $n^{O(\log\log n \log(1/\epsilon))}$ under the ideal PRG assumption.

Keywords: Always approximately correct learning ; Membership queries ; Disjunctive Normal Form ; Derandomization; Fourier Transform

Dedication

To my family for always being there.

Acknowledgements

I would like to express my deepest gratitude to my senior supervisor Dr. Valentine Kabanets for giving me the opportunity to research and guiding me with patience. This thesis is mainly based on joint work with Dr. Kabanets. His ability to explain hard concepts in a way that is intuitively clear and passion for the field of computational complexity has always been a source of inspiration.

I was very fortunate to be a part of many research meetings with Dr. Kabanets, Dr. Antonina Kolokolova, and Dr. Marco Carmosino. My sincere thanks to Dr. Kolokolova for her great advice which enriched this thesis. I want to thank Dr. Carmosino for his many presentations and intellectual openness which taught me how research should be done. I owe special thanks to Zhenjian Lu for providing helpful guidance at the beginning of this research.

I would like to extend my special thanks to Dr. Andrei Bulatov, Dr. Igor Shinkar, and Dr. Oliver Schulte for accepting to serve as the examining committee.

Table of Contents

Chapter 1

Introduction

1.1 Introduction

A logical formula is in DNF if it is a disjunction of one or more conjunctions of one or more literals. The size of a DNF formula is measured by either the number of conjunctions(terms) or the maximum width of a term. DNFs are widely used in machine learning because it is a natural way to express real concepts. Therefore much attention has been given to the class of functions with compact DNF representations and its learnability has been well-studied. This paper is about learning DNFs with polynomial number of terms.

The learning algorithm we will be discussing in this paper is an approximately correct learning. By approximately correct, we mean that the output hypothesis of the learner agrees with the target function on almost all inputs except some fraction under the uniform distribution. The error can be a constant or it can be a function of the size of the input. The quality of the hypothesis is then evaluated according to the error.

In the original PAC learning model [Val84], the adversary can choose any distribution on the inputs on which the target function evaluates to true and the learner is only allowed to get random examples from the distribution. Since we can exactly identify a Boolean function on $\{0,1\}^n$ by identifying the subset of $\{0,1\}^n$ on which the function is true, Valiant opts for one-sided error for the learning model. In other words, the hypothesis function will be false on those inputs that the target function is false but the hypothesis is allowed to make mistakes on those inputs that the target function is true. Assuming that the distribution is what naturally occur for the positive examples, the inputs that are sampled from the distribution will represent what inputs typically make the target function true. The idea behind PAC learning model is that after enough sampling from the distribution, with high probability, we will have seen all the examples that occur with high probability in which case the probability mass of the unseen examples is negligible. Then construction of a hypothesis simply becomes defining a function that is true on the examples we have seen so far. This hypothesis will then, with high probability, agree with the target function except on those inputs that occur very rarely. A class of function is PAC learnable if there is a polynomial time algorithm that can do the above procedure and output a hypothesis.

Despite intensive research to PAC learn DNF formulas, the fastest algorithm that learns polynomial size DNF under arbitrary distribution remains to be $2^{\tilde{O}\left(n^{1/3}\right)}$ [KS04]. Even if we restrict the distribution to the uniform, the fastest algorithm to this date is $n^{O(\log n)}$ [Ver90]. While the examples in the PAC learning model are independent from each other, Bshouty and others [BMOS05] presented a polynomial time algorithm under another natural assumption where the examples are not generated independently but are produced sequentially according to a random walk on the Boolean hypercube.

An important technique in analysis of Boolean function is Fourier analysis. Fourier expansion of a Boolean function is a multilinear polynomial with real valued coefficients where each monomial is a parity function. By studying the coefficients carefully, researchers have been able to derive many useful properties of a Boolean function. These properties can be used to construct a function that can approximate a given Boolean function. In their seminal work, Linial, Mansour and Nisan [LMN93] used Fourier analysis to learn Boolean functions that can be computed by depth d and polynomial size circuits in time $n^{O\left(\log^d(n/\epsilon)\right)}$ under the uniform distribution where ϵ is the error. They used the fact that a constant depth circuit has most of its Fourier coefficients on the small degree terms and estimated those coefficients using random sampling from the uniform distribution. Since then Fourier analysis played a central role in learning of DNFs.

Note that the examples presented to the learner in the original PAC learning model are random and the learner is passive in this sense. To overcome the limitation of random examples, *membership queries* were introduced to learning models. With membership queries, the learner can query the unknown target function on any input. Building on his previous work of [KM93], which uses membership queries in a recursive algorithm and outputs close approximations to all of the large Fourier coefficients of a Boolean function, Mansour [Man92] presented an $(n/\epsilon)^{O(\log\log(n/\epsilon)\log(1/\epsilon))}$-time algorithm to learn DNFs under the uniform distribution. In this paper, he shows that a DNF can be approximated by a polynomial with few non zero coefficients. Informally, his result follows from the fact that given a DNF with terms of size at most some bound w (w-DNF), there cannot be too many large coefficients of small degree. More specifically, he shows that given a w-DNF, the sum of absolute values of Fourier coefficients up to degree k is bounded by $L = w^{O(k)}$,therefore, the number of coefficient whose absolute value larger than $\epsilon/2L$ and degree less than k is at most $(2L/\epsilon)^2$. This combined with the fact that the total Fourier mass (i.e the sum of squares of all Fourier coefficients) is one, naturally yields a learning algorithm. That is, to find those large absolute value and small degree coefficients. Goldreich and Levin [GL89] showed that given a query access to an n-variate Boolean function and a threshold τ, there exists an algorithm running in $\mathsf{poly}(n, \tau)$ and outputs the list of subsets of $[n]$ that contains all subsets of $[n]$ that correspond to all of the coefficients whose absolute value is greater than τ. Later, [KM93] developed a learning algorithm based on [GL89] for a class of Boolean functions such that every function in the class has its all but ϵ Fourier spectrum on a small collection of subsets of $[n]$. Above mentioned work [Man92] is to show that w-DNF is a class of functions with this property.

Jackson later combined Freund's boosting algorithm [Fre95] and Fourier analysis to give a polynomial time algorithm [Jac97] to learn DNFs under the uniform distribution. Roughly speaking, boosting algorithm works as follows. It takes a weak learner that produces a hypothesis which merely beats coin toss and after training the weak learner number of times on different distribution of inputs, it outputs a strong hypothesis that is a combination of the weak hypothesis in some way. The distribution in each iteration is updated to give more weight on the inputs that the previous weak hypothesis didn't perform well. Jackson proves that for a DNF and any distribution $\mathcal{D}$, there exists an algorithm that produces in time that is polynomial in the number of terms of the DNF, and the maximum value of $2^n\mathcal{D}$, a parity function which weakly approximates the DNF. He then uses this weak learner in the boosting algorithm. Jackson's celebrated result is often called Harmonic Sieve.

Feldman [Fel12] later introduced a new way of learning a DNF expression under the product distribution, in particular the uniform distribution, from estimating heavy low degree Fourier coefficients. His algorithm uses membership queries and efficiently learns a DNF expression under the uniform distribution. He showed that a given DNF f of m terms, and any $[-1,1]$ bonded function on $\{-1,1\}^n$, the expectation of the difference between f and g is at most the multiple of $(2s+1)$ and the maximum difference in the Fourier coefficient of f and g under the uniform distribution. This means in order to ϵ learn f under the uniform distribution, finding a bounded function such that each of its Fourier coefficient is at most $\epsilon/(2s+1)$ different from the corresponding Fourier coefficient of f is enough. Feldman finds this hypothesis function g by using an iterative process such that in each iteration, the L_2 norm squared distance between the target and the updated hypothesis is strictly less than the L_2 norm squared distance between the target and the previous hypothesis.

In contrast to Valiant's probably approximately correct learning model using random sampling, Angluin [Ang87] considered the exact learning model using queries.

There have been some deterministic learning algorithms for constant depth circuits. Sitharam [Sit95] used Fourier mass concentration result from [LMN93] and showed that a distribution called polylog-wise decomposable distribution fools any constant depth circuits to present a deterministic learning algorithm for constant depth circuits running in $2^{O(\mathsf{poly}(\log n))}$ where the degree of $\mathsf{poly}(\log)$ depends on the depth and size of the circuit.

1.2 Our Results

The above mentioned randomized learning algorithms based on Fourier analysis of Boolean functions use random sampling to estimate the mean of relevant random variables. Our main result is to use a pseudorandom generator (PRG) for DNFs to derandomize this estimation process. A pseudorandom generator is an efficiently computable function that takes a short string called a seed and outputs a long string such that the distribution of the outputs is similar to the uniform distribution from the perspective of a class of functions. PRG can be used in derandomization by enumerating its all possible outputs and using them instead of random bits. Our algorithm is deterministic in the sense that the sample points from the input space is determined by our choice of PRG.

It is worthwhile to note that the deterministic training set produced by the choice of PRG for a class of function is independent of the particular function that is learnt. In other words, we have the same training set for every function in the class. This subject of uniform training set for deterministic learning is studied in depth by Sitharam and Straney [SS97]. Having fixed points to query the unknown target function poses a problem if we assume that the function is a particular type when, in fact, we only have blackbox access to the function that computes our target DNF. We deal with this issue by showing that an arbitrary DNF can be approximated by a particular type of DNF on which we can apply our choice of PRG and by using the fact that the PRG that fools the approximators can also fool the approximated function. We use Gopalan and Meka's [GMR13] PRG along with the fact that the expectation can be expressed as the sum of conditional expectations to approximate Fourier coefficients of a DNF. We then use this approximation method to present a deterministic algorithm to find small degree, large absolute value coefficients with which we build our polynomial that is close to the target function in L_2 norm. Tal's result [Tal17] which upper bounds the sum of squares of large degree coefficients as well as the sum of absolute values of small degree coefficients is then applied to bound the error of the polynomial.

The above procedure yields a deterministic $n^{\tilde{O}\left(\log(n)\log^2(1/\epsilon)\right)}$ time membership query learning algorithm for the class of DNFs with $\mathsf{poly}(n)$ terms.

Main Theorem. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function that can be computed by a DNF with $\mathsf{poly}(n)$ terms. Then for $\epsilon > 0$ there exists a deterministic learning algorithm that runs in time*

$$n^{\tilde{O}\left(\log(n)\log^2(1/\epsilon)\right)}$$

and uses membership queries to f, and outputs $g : \{0,1\}^n \to \mathbb{R}$ such that $\mathbb{E}_{x\sim\mathcal{U}}[(f(x) - g(x))^2] \leq \epsilon$.

In order to build such a polynomial g, we present two deterministic ways to approximate Fourier coefficients. First, we show that given a DNF of terms with width at most w (w-DNF) and a constant $\epsilon > 0$, there exists a deterministic algorithm to ϵ-approximate a Fourier coefficient of degree less than t in time $2^{\tilde{O}(w^2+w\cdot\log(2^t/\epsilon)+\log\log n)} \cdot 2^t$.

Main Lemma 1. *For a given width w-DNF, there is a deterministic procedure to ϵ-approximate its Fourier coefficient of degree less than t in time $2^{\tilde{O}\left(w^2+w\cdot\log\left(2^t/\epsilon\right)+\log\log n\right)} \cdot 2^t$.*

Although this method applies to w-DNFs due to the nature of the PRG used, we will show that by finding an approximating w-DNF to an arbitrary DNF with $\mathsf{poly}(n)$ terms, we can still apply the PRG for w-DNFs to approximate the small degree coefficients of the target DNF. This is possible because of the more general fact that a pseudo random distribution that fools an approximator of f also fools f.

Another approach we use to get an approximation of Fourier coefficients is to directly fool the product of our target function and the parity function. For this, we closely follow the proof of existence of small L_1 norm sandwich approximator for a width w DNF in [DETT10]. By using sparse polynomial approximation result [Tal17] that does not depend on the width of the DNF, we generalize this result and show that for an arbitrary DNF of $\mathsf{poly}(n)$ terms, there exists a small L_1 norm sandwiching approximators. Furthermore, using the fact that the product of a parity function χ_S for any $S \subseteq [n]$ and a function f has the same L_1 norm as f, we show that a small biased distribution that fools the sandwiching approximators of f also fools $f \cdot \chi_S$. A deterministic algorithm to approximate the mean of $f \cdot \chi_S$ then follows. We show that given an arbitrary DNF with $\mathsf{poly}(n)$ number of terms, there exists a deterministic algorithm to ϵ-approximate a Fourier coefficient of any degree in time $(\log n)^{O(\log n \log(1/\epsilon))}$.

Main Lemma 2. *Given a DNF with $m = \mathsf{poly}(n)$ terms $\phi : \{0,1\}^n \to \{0,1\}$ and $\epsilon > 0$, there exists a deterministic procedure that can approximate $\hat{\phi}(S)$ for $S \subseteq [n]$ within 6ϵ in time $(\log n)^{O(\log n \log(1/\epsilon))}$.*

This approximation is an improvement over our first method where we break a small degree Fourier coefficient into conditional expectations in the sense that it can approximate a Fourier coefficient of any degree for a general DNF.

Chapter 2

Main Result

2.1 Preliminaries

In this section we give all necessary definitions and some observations that will be useful for the discussion of our result.

2.1.1 Vector space

A vector space over a field $\mathbb{F}$ is a set $\mathbb{V}$ with two operations called vector addition and scalar multiplication. Vector addition is a mapping: $\mathbb{V} \times \mathbb{V} \to \mathbb{V}$. Scalar multiplication is a mapping: $\mathbb{F} \times \mathbb{V} \to \mathbb{V}$. Vector addition must meet associativity, commutativity,existence of the identity element, and existence of the inverse element axioms. Scalar multiplication must be compatible with field multiplication, have the identity scalar element, meet the axiom of distributivity with respect to vector addition and with scalar addition.

2.1.2 Fourier transform

With the definition of vector space in mind, we can see that the set of all real valued functions $f\colon \{0,1\}^n \to \mathbb{R}$ forms a vector space over the field of real numbers. Each element of the vector space can be thought of as a vector in $\mathbb{R}^{2^n}$ where the entries of the vector is the function values on all possible inputs in lexicographic order. Then it is easy to verify that element wise vector addition and multiplication by a real number meet all the axioms of vector space.

We can write any Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ as a DNF of at most 2^n terms such that for any string in $\{0,1\}^n$, at most one term of the DNF evaluates to 1. We can also convert this DNF into a multilinear polynomial $q\colon \{0,1\}^n \to \{0,1\}$ by replacing a conjunction with multiplication, disjunction with addition, $\bar{x_i}$ with $(1-x_i)$, and x_i with x_i. The coefficients of q will be in the range $[-2^n, 2^n]$. Now consider the same function f but instead of using 1 for True, we use -1 and instead of using 0 for False, we use 1. We can show that this function can also be expressed as a multilinear polynomial $p : \{1,-1\}^n \to \{1,-1\}$.

Since p and q both represent the same function f, we can write

$$q(x_1, ..., x_n) = \frac{1}{2} - \frac{1}{2}p(1 - 2x_1, ..., 1 - 2x_n)$$

From the above equation, we can see that the coefficients of p is in the range $[-1, 1]$. We denote the monomial in p that corresponds to a subset $S \subseteq [n]$ with character $\chi_s : \{1, -1\}^n \to \{1, -1\}$. In other words,

$$\chi_s\,(x) = \prod_{i \in S} x_i$$

Note that the characters correspond to the logical parity functions or exclusive OR when the input domain is $\mathbb{F}_2^n$. That is, we can define the character χ_S as follows.

Definition 2.1.1. *For $S \subseteq [n]$ define $\chi_S \colon \mathbb{F}_2^n \to \mathbb{R}$ by*

$$\chi_S(x) = (-1)^{\sum_{i \in S} x_i}$$

We also denote the coefficients of χ_S as $\hat{f}(S)$. So we can see that any real valued function $f \colon \{-1, 1\}^n \to \{1, -1\}$ can be represented as a multilinear polynomial as follows

$$f(x) = \sum_{s \subseteq [n]} \hat{f}(s)\chi_s(x)$$

This multilinear polynomial representation is called the Fourier expansion of f.

Remark 2.1.2. *Although we are discussing Boolean valued functions in this paper, note that any real valued functions $f : \{-1, 1\}^n \to \mathbb{R}$ can be written as the Fourier expansion. We are considering Boolean valued functions as real valued functions $f \colon \{0, 1\}^n \to \mathbb{R}$.*

This representation shows that the set of all χ_S functions is a spanning set of the vector space. Since there are 2^n different such functions, the set of all χ_S is a basis for the vector space. Therefore, the representation is unique.

Definition 2.1.3. *We can also define an inner product on pairs of functions $f, g : \{-1, 1\}^n \to \mathbb{R}$ as follows*

$$\langle f, g \rangle = \mathop{\mathbb{E}}_{x \in \{-1,1\}^n} [f(x)g(x)]$$

It can be verified that the basis defined in Definition 2.1.1 above is an orthonormal basis, which means for any $S \subseteq [n]$, $\langle \chi_S, \chi_S \rangle = 1$ and for two distinct $S \subseteq [n]$ and $T \subseteq [n]$, $\langle \chi_S, \chi_T \rangle = 0$.

Remark 2.1.4. *It is worthwhile to note that even though the characters compute the logical parity where True, False is encoded as $\{1, 0\}$, the orthonormality follows when True, False is encoded as $(-1)^1 = -1, (-1)^0 = 1$.*

The orthonormality of the basis results in Parseval's theorem

$$\langle f, f\rangle = \mathop{\mathbb{E}}_{x\in\{-1,1\}^n}[f\left(x\right) f\left(x\right)] = \sum_{s\in[n]} \hat{f}^2(s)$$

We use the following notation for l_2 norm of f.

$$||f||_2 = \sqrt{\langle f, f\rangle} = \sqrt{\sum_{s\in[n]} \hat{f}^2(s)}$$

The orthonormality of the basis functions also let us compute the Fourier coefficients of $f : \{-1, 1\} \to \mathbb{R}$ easily

$$\langle \chi_s, f\rangle = \mathop{\mathbb{E}}_{x\in\{-1,1\}^n}[\chi_s(x)f(x)] = \hat{f}(s)$$

Remark 2.1.5. *[O'D14] When the two functions are $\{-1, 1\}$ valued in Definition 2.1.3, we can see that the inner product of them is measuring how similar they are. That is, for $f, g : \{-1, 1\}^n \to \{-1, 1\}$*

$$\langle f, g\rangle = Pr_{x\in\{-1,1\}}[f(x) = g(x)] - Pr_{x\in\{-1,1\}}[f(x) \neq g(x)]$$

We will refer to this measure as a correlation between f and g. In this sense, given $f : \{-1, 1\} \to \{-1, 1\}$ the Fourier coefficient $\hat{f}(S)$ is a correlation between f and χ_S.

Remark 2.1.6. *When $f : \{0, 1\} \to \mathbb{R}$ is $\{-1, 1\}$ valued, we can see that $\langle f, f\rangle = \sum_{S\subseteq[n]} \hat{f}(S) = 1$. It is worthwhile to note that there are some useful properties like this when we encode the value of the Boolean valued function with $\{-1, 1\}$*

We say that $f : \{-1, 1\}^n \to \mathbb{R}$ is ϵ concentrated up to degree k if

$$\sum_{s\subseteq[n] and |s|>k} \hat{f}(s)^2 \leq \epsilon$$

The degree of f denoted as $deg(f)$ is $max\{|s| : \hat{f}(s) \neq 0\}$
A t sparse function is a function that has at most t non-zero coefficients.
The following fact follows from Parseval's equality.

$$\mathop{\mathbb{E}}_{x\in\{-1,1\}^n}[(f(x) - g(x))^2] = \sum_{s\subseteq[n]} (\hat{f}(s) - \hat{g}(s))^2 = ||f - g||_2^2$$

We use the following notation for the Fourier l_1 norm of f and a l_1 norm excluding $\hat{f}(\emptyset) = \mathbb{E}_{x\in\{-1,1\}^n}[f(x)]$.

$$||f||_1 := \sum_S |\hat{f}(S)| \text{ and } ||f||_1^{\neq\emptyset} := \sum_{S\neq\emptyset} |\hat{f}(S)|$$

The l_1 norm is also called the spectral norm. This term is used more often when we refer to the sum of absolute values of Fourier coefficients of a certain degree. So we call the following the spectral norm of kth-level of f.

$$\sum_{S:|S|=k} |\hat{f}(S)|$$

Lemma 2.1.7. *For any $f, g : \{0,1\}^n \to \mathbb{R}$, $||fg||_1 \leq ||f||_1 ||g||_1$.*

The following fact and remarks will be useful in our discourse.

Lemma 2.1.8. *If $f : \{-1,1\}^n \to \mathbb{R}$ is a t sparse function, then $||f||_1 \leq t$.*

Proof. Let $\gamma = \{S : \hat{f}(S) \neq 0\}$

$$\begin{aligned}\left(\sum_{S\subseteq[n]} |\hat{f}(S)|\right)^2 &\leq |\gamma| \left(\sum_{S\subseteq[n]} |\hat{f}(S)|^2\right) && \text{(By Jensen's inequality)}\\ &\leq |\gamma| && \text{(Because } \textstyle\sum_{S\subseteq[n]} |\hat{f}(S)|^2 \leq 1)\end{aligned}$$

Now $||f||_1 = \sum_{S\subseteq[n]} |\hat{f}(S)| \leq \sqrt{|\gamma|} \leq t$. □

Remark 2.1.9. *If $f, g : \{0,1\} \to \mathbb{R}$, then $||f+g||_1 \leq ||f||_1 + ||g||_1$ and $||fg||_1 \leq ||f||_1 ||g||_1$.*

Remark 2.1.10. *If $\phi : \{0,1\}^n \to \{0,1\}$ is an AND of some subsets of literals, then $||\phi||_1 = 1$.*

2.1.3 Restriction

A restriction ρ on variables $\{x_1, ..., x_n\}$ is a mapping of the variables to $\{0, 1, *\}$. The function obtained from $f(x_1, ..., x_n)$ by applying a restriction ρ is denoted by f_ρ. The inputs of f_ρ are those variables x_i such that $\rho(x_i) = *$ while all other variables are set according to ρ.

2.1.4 Secure pseudorandom generators

In this section we will cover some theorems and definitions for secure pseudo random generators and introduce the Goldreich and Levin algorithm [GL89]. Although this algorithm was developed for cryptography, it has other applications such as learning Fourier coefficients.

We will use $\mathcal{U}$ with a subscript to denote the uniform distribution over the bit strings of length of the subscript. That is $\mathcal{U}_n$ is the uniform distribution over $\{0,1\}^n$. $\mathcal{U}$ without a subscript means U_n unless otherwise noted.

Definition 2.1.11 (Negligible functions)**.** *A function $\epsilon : \mathbb{N} \to [0,1]$ is called negligible if $\epsilon(n) < n^{-c}$ for every c and sufficiently large n.*

Definition 2.1.12 (Unpredictable functions)**.** *let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function with $|G(x)| = l(|x|)$ for all $x \in \{0,1\}^*$. G is unpredictable if for every probabilistic polynomial time B there exists a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that*

$$Pr_{x\in\{0,1\}^n, y=G(x), i\in[l(n)]}[B(1^n, y_1, ..., y_{i-1}) = y_i] \leq 1/2 + \epsilon(n)$$

Definition 2.1.13 (One-way functions)**.** *A polynomial time computable $f : \{0,1\}^* \to \{0,1\}^*$ is one-way function if for all polynomial time probabilistic algorithm A, there exists a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that for every $n \in \mathbb{N}$,*

$$Pr_{x\in\{0,1\}^n}\left[f\left(A\left(f(x)\right)\right) = f(x)\right] \leq \epsilon(n)$$

Definition 2.1.14 (Secure pseudorandom generators)**.** *Let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. Let $l : \mathbb{N} \to \mathbb{N}$ be a polynomial-time computable function such that $l(n) > n$ for every n. We say that G is a secure pseudorandom generator of stretch $l(n)$, if $|G(x)| = l(|x|)$ for all $x \in \{0,1\}^*$ and for every probabilistic polynomial-time A, there exists a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that*

$$Pr\left[A\left(G\left(\mathcal{U}_n\right)\right) = 1\right] - Pr[A\left(\mathcal{U}_{l(n)}\right) = 1] \leq \epsilon(n)$$

for every $n \in \mathbb{N}$.

In the cryptography setting, the algorithm A in above theorem can be thought of as an adversary. So G is secure in the sense that for any polynomial time adversary, it is not possible to distinguish between a random string of length $l(n)$ and a string generated by G.

Theorem 2.1.15. *[Yao82] Let $l : \mathbb{N} \to \mathbb{N}$ be some polynomial-time computable function and let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function such that $|G(x)| = l(|x|)$ for all $x \in \{0,1\}^*$. If G is unpredictable then it is a secure pseudorandom generator.*

The following theorem is a consequence of Goldreich and Levin algorithm. We will discuss its application in learning of Boolean functions in the next section.

Theorem 2.1.16. *[GL89] Suppose that $f : \{0,1\}^n \to \{0,1\}^n$ is a one-way function such that f is one-to-one. Then, for every probabilistic polynomial-time algorithm A, there is a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that*

$$Pr_{x,r\in\{0,1\}^n}[A(f(x), r) = \sum_{i=1}^{n} x_i \cdot r_i (mod 2)] \leq 1/2 + \epsilon(n)$$

Theorem 2.1.16 implies that the function $G(x,r) = f(x), r, \sum_{i=1}^{n} x_i \cdot r_i (mod2)$ is a secure pseudo random generator of stretch function $l(2n) = 2n + 1$ for $n \in \mathbb{N}$. To see this, assume otherwise. Then by Theorem 2.1.15, there exists a predictor for G which contradicts Theorem 2.1.16.

Theorem 2.1.16 is an immediate corollary of the following result.

Theorem 2.1.17. *[GL89] Let $f : \{0,1\}^n \to \{0,1\}$ be a function such that, for some unknown x,*

$$Pr_{r\in\{0,1\}^n}[f(r) = \sum_{i=1}^{n} x_i \cdot r_i(mod2)] \geq \frac{1}{2} + \epsilon$$

Then, there exists $O\left(n^2\epsilon^{-4}\log n\right)$ time algorithm that makes $O\left(n\epsilon^{-2}\log n\right)$ queries into f and outputs a list $L \subseteq [n]$ such that $|L| = O\left(\epsilon^{-2}\right)$ and $x \in L$ with probability at least $1/2$.

Remark 2.1.18. *Note that $\sum_{i=1}^{n} x_i \cdot r_i(mod2)$ is the logical parity of bits $r_i \in s$ where $s = \{i : x_i = 1\}$. In other words, $(-1)^{\sum_{i=1}^{n} x_i \cdot r_i(mod2)} = \chi_s(r)$. So we see that*

$$\begin{aligned}
&Pr_{r\in\{0,1\}^n}[f(r) = \sum_{i=1}^{n} x_i \cdot r_i(mod2)] \\
&= Pr_{r\in\{-1,1\}^n}[f(r) = \sum_{i=1}^{n} x_i \cdot r_i(mod2)]] && \text{(encoding } \mathbb{F}_2^n \text{ with } \{-1,1\}^n\text{)} \\
&= Pr_{r\in\{-1,1\}^n}[(-1)^{f(r)} = (-1)^{\sum_{i=1}^{n} x_i \cdot r_i(mod2)}] \\
&= Pr_{r\in\{-1,1\}^n}[g(r) = \chi_S(r)] && \text{(Letting } g(r) = (-1)^{f(r)} \text{ and } S = \{i : x_i = 1\}\text{)} \\
&= \frac{\hat{g}(S) + 1}{2}
\end{aligned}$$

From above remark, we see that Goldreich and Levin algorithm in Theorem 2.1.17 outputs a list of all parity functions χ_S that are at least slightly correlated with $g(x) = (-1)^{f(x)}$.

2.1.5 Derandomization using pseudorandom generators

We say that a probability distribution $\mathcal{D}$ over $\{0,1\}^n$ can be sampled efficiently with r random bits if $\mathcal{D}$ is a uniform distribution over a multiset $z^{(1)}, z^{(2)}, ..., z^{(s)}$ of strings from $\{0,1\}^n$ where $s \in [\frac{1}{\mathsf{poly}(n)}2^r, 2^r]$ and there is a deterministic algorithm G that, on uniformly random input from $[s]$, runs in $\mathsf{poly}(n,s)$ and outputs a string from $\mathcal{D}$.

Definition 2.1.19 (Pseudorandom generators). *For $\epsilon > 0$ and a class $\mathcal{F}$ of functions from $\{0,1\}^n$ to $\{0,1\}$, we say that a deterministic algorithm $G : \{0,1\}^r \to \{0,1\}^n$ that efficiently samples a distribution $\mathcal{D}$ as explained above is an ϵ-pseudorandom generator(PRG) for $\mathcal{F}$ if for all $f \in \mathcal{F}$, we have*

$$|\mathop{\mathbb{E}}_{y\sim\{0,1\}^r}[f(G(y))] - \mathop{\mathbb{E}}_{x\sim\{0,1\}^n}[f(x)]| \leq \epsilon$$

Remark 2.1.20. *It is natural to use the minimum size of the circuit that computes f to classify a given function. However, the pseudo random generators we will use in our derandomization is for a class of functions that are computable by DNFs. That is, depth 2 circuits of polynomial size.*

In the definition of PRG, the parameter r is called the seed length. The class of functions $\mathcal{F}$ is said to be fooled by G or equivalently, by the distribution $\mathcal{D}$. A pseudorandom generator for a class of function $\mathcal{F}$ can be though as a generator of a distribution over inputs that is statistically close to the uniform distribution when the test is in $\mathcal{F}$.

Below we explain formally how a pseudo random generator can be used for derandomization.

Lemma 2.1.21. *Let $\mathcal{F}$ be a class of Boolean circuits of polynomial size .*
Given $f : \{0,1\}^n \to \{0,1\}$, assume that there exists an algorithm A that can be computed in $\mathsf{poly}(n)$ time and $Pr_{r\in\{0,1\}^m}[A(x,r) = f(x)] > 1-\delta$ where m is the number of random bits required for A. If there exist an ϵ-pseudo random generator G for $\mathcal{F}$ with seed length l then there exists a deterministic algorithm B that runs in time $2^l\mathsf{poly}(n)$ such that $B(x) = f(x)$ for all $x \in \{0,1\}^n$.

Proof. We can construct a deterministic algorithm B as follows.
B : on input $x \in \{0,1\}^n$, will compute $A(x, G(z))$ for each $z \in \{0,1\}^l$. This will take $2^l\mathsf{poly}(n)$ time. Then B outputs the majority of $A(x, G(z))$ over all z.
We claim that the fraction of z such that $A(x, G(z)) = f(x)$ is at least $1-\delta-\epsilon$.
Assume to the contrary that $Pr_{z\in\{0,1\}^l}[A(x,G(z)) = f(x)] < 1-\delta-\epsilon$. Then we see that $Pr_{r\in\{0,1\}^m}[A(x,r) = f(x)] - Pr_{z\in\{0,1\}^l}[A(x,G(z)) = f(x)] > \epsilon$. It is a well known fact that a polynomial size circuit can be constructed for a polynomial time algorithm. Therefore we can construct a polynomial size circuit C_A that computes $r \to A(x,r)$ by hard wiring x. However, this means that G fails to fool the circuit $C_A \in \mathcal{F}$ which is a contradiction. □

Remark 2.1.22. *Note that the efficiency of the pseudo random generator in terms of the seed length was not considered in the derandomization shown in above proof. That is, the deterministic algorithm enumerates over all possible seeds which takes exponential time in seed length anyways.*

2.1.6 Biased distribution

Definition 2.1.23 (ϵ-biased distribution)**.** *A probability distribution $\mathcal{D}$ over $\{0,1\}^n$ is ϵ-biased if and only if it ϵ-fools the Fourier basis functions χ_S. That is,*

$$|\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\chi_S(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[\chi_S(x)]| \leq \epsilon \quad \forall S \subseteq [n]$$

Let us denote the uniform distribution over $\{0,1\}^n$ with the support $B \subseteq \{0,1\}^n$ with $\mathcal{U}_B$. A subset $B \subseteq \{0,1\}^n$ is called ϵ-biased if $\mathcal{U}_B$ is ϵ-biased.

Explicit construction of small biased space in polynomial time is used considerably in the field of algorithmic derandomization. The following result is due to Naor and Naor [NN93] and [AGHP92].

Theorem 2.1.24. *[NN93][AGHP92] There exists ϵ-biased sets of size $O\left(n^2/\epsilon^2\right)$ such that a random element from the set can be sampled using a seed length $2\log(n/\epsilon)$, in time $\mathsf{poly}(n, \log(1/\epsilon))$.*

When a real valued function $f : \{0,1\}^n \to \mathbb{R}$ has small L_1 norm, it can be fooled by a small ϵ-biased sets.

Lemma 2.1.25. *All functions $f : \{0,1\}^n \to \mathbb{R}$ is $\epsilon||f||_1$-fooled by any ϵ-biased probability distribution*

Proof. Let $\mathcal{D}$ be an ϵ-biased distribution. Then

$$\begin{aligned} |\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[f(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f(x)]| &= |\mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[\sum_S \hat{f}(S)\chi_S(x)\right] - \hat{f}(\emptyset)| \\ &= |\sum_{S\neq\emptyset} \hat{f}(S) \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\chi_S(x)]\,| \\ &\leq \epsilon \sum_{S\neq\emptyset} |\hat{f}(S)| \leq \epsilon||f||_1^{\neq\emptyset} \leq \epsilon||f||_1 \end{aligned}$$

□

2.1.7 Learning model

Our learning model uses deterministic membership queries. We assume that the learner is given some unknown target function which it can only access by black box. More formally, a membership oracle for f is an oracle that given any instance x, returns the value $f(x)$. The number of queries the learner makes is bounded by the number of all possible outputs of a PRG of our choice. Therefore we need a PRG with small seed length. The learner than outputs a hypothesis h.

A deterministic algorithm A learns a class of function $\mathcal{F}$ if for every $f \in \mathcal{F}$ and $\epsilon > 0$, the algorithm outputs $h = A(f, \epsilon)$ such that $Pr_{x\sim\mathcal{U}}[f(x) \neq h(x)] \leq \epsilon$.

A real valued function g ϵ approximates f in norm L_2 if $\mathbb{E}_{x\sim\mathcal{U}}[(f(x) - g(x))^2] \leq \epsilon$.

Remark 2.1.26. *We will refer to the quantity $\mathbb{E}_{x\sim\mathcal{U}}[(f(x)-g(x))^2]$ as a squared L_2 distance from f to g.*

Given a real valued function that ϵ approximates $f : \{0,1\}^n \to \{-1,1\}$, we can get a hypothesis h by setting $h = sign(g)$ where the value of $sign(g)$ is 1 when g is positive, -1 if g is negative and 0 when $g = 0$.

This hypothesis then satisfies the following inequality.

$$Pr_{x\sim\mathcal{U}}\left[f(x) \neq h(x)\right] \leq Pr_{x\sim\mathcal{U}}\left[|f(x) - g(x)| \geq 1\right] \leq \mathop{\mathbb{E}}_{x\sim\mathcal{U}}\left[(f(x) - g(x))^2\right] \leq \epsilon$$

Another useful measure of the distance between the target function $f:\{0,1\}^n \to \{-1,1\}$ and a real valued function $g:\{0,1\}^n \to \mathbb{R}$ when we want to upper bound the probability $Pr_{x\sim\mathcal{U}}[f(x) - sign(g(x))]$ is the expectation of the difference between f and g since we have the following

$$Pr_{x\sim\mathcal{U}}[f(x) - sign(g(x))] \leq \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[|f(x) - g(x)|]$$

2.1.8 Sandwiching Approximators

Definition 2.1.27. *Let $f\colon \{0,1\}^n \to \{0,1\}$. We say that functions $f_l, f_u\colon \{0,1\}^n \to \mathbb{R}$ are ϵ- sandwiching approximators for f if $\forall x \in \{0,1\}^n$, $f_l(x) \leq f(x) \leq f_u(x)$ and*

$$\mathop{\mathbb{E}}_{x\sim\mathcal{U}}[|f(x) - f_l(x)|] \leq \epsilon$$

$$\mathop{\mathbb{E}}_{x\sim\mathcal{U}}[|f(x) - f_u(x)|] \leq \epsilon$$

We will show that there exists ϵ-sandwiching approximator for an arbitrary DNF with polynomial number of terms. Finding those approximator is useful because of the following fact.

Lemma 2.1.28. *If G is a PRG that fools the ϵ- sandwich approximators of f, f_l and f_u then G also fools f.*

Proof. Let G be a pseudorandom generator for f_l and f_u. Let $\mathbb{E}_{x\sim\mathcal{G}}[f(x)]$ denote the expectation of f when the inputs are distributed according to the distribution generated by G.

Our goal is to show that for an arbitrary small $\epsilon > 0$,

$$|\mathop{\mathbb{E}}_{x\sim\mathcal{G}}[f(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f(x)]| \leq 2\epsilon$$

Observe that

$$\begin{aligned}
\mathop{\mathbb{E}}_{x\sim\mathcal{G}}[f(x)] \leq \mathop{\mathbb{E}}_{x\sim\mathcal{G}}[f_u(x)] \leq \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f_u(x)] + \epsilon &= \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f_u(x) + f(x) - f(x)] + \epsilon \\
&= \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f_u(x) - f(x)] + \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f(x)] + \epsilon \\
&\leq \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[|f_u(x) - f(x)|] + \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f(x)] + \epsilon \\
&\leq \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f(x)] + 2\epsilon
\end{aligned}$$

where the first inequality follows from the fact that $\forall x \in \{0,1\}^n$, $f(x) \leq f_u(x)$, second inequality follows from the fact that G is a pseudorandom generator for f_u and the last inequality is due to the definition of ϵ-sandwiching approximator. In a symmetric manner, observe that

$$\begin{aligned}\mathop{\mathbb{E}}_{x\sim\mathcal{G}}[f(x)] \geq \mathop{\mathbb{E}}_{x\sim\mathcal{G}}[f_l(x)] \geq \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f_l(x)] - \epsilon &= \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f_l(x) + f(x) - f(x)] - \epsilon \\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f_l(x) - f(x)] + \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f(x)] - \epsilon \\ &\geq \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f(x)] - \epsilon - \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[|f_l(x) - f(x)|] \\ &\geq \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f(x)] - 2\epsilon\end{aligned}$$

□

Our main technique in approximating Fourier coefficients is to fool an arbitrary DNF of polynomial number of terms using a PRG. However, the seed length of the PRG of our choice depends on the maximum width of the DNF and, in our learning model, we only have black box access to our DNF. To resolve this problem, we show that DNFs with "short" term length ϵ-sandwich approximate an arbitrary DNF. Then by Lemma 2.1.28 we will be able to use our choice of PRG to fool an arbitrary DNF.

Lemma 2.1.29. *Given $f\colon \{0,1\}^n \to \{0,1\}$ that can be computed by a DNF with $\mathsf{poly}(n)$ number of terms, there exist $f_l, f_u\colon \{0,1\}^n \to \{0,1\}$ that can be computed by DNFs with terms of length $O(\log(n/\epsilon))$ and they ϵ-sandwich approximate f.*

Proof. Let m be the number of terms in the DNF that computes F.
We let f_l to be the function computed by the DNF obtained by removing all terms of width greater than $\log\left(\frac{m}{\epsilon}\right)$ from the DNF that computes f. Let us denote this DNF with DNF_l. For $x \in \{0,1\}^n$, if $f(x) = 0$ then all terms of the DNF that computes f is 0. This means $f_l(x) = 0$. If $f(x) = 1$ then there must be a term in the DNF that computes f that is 1. In this case, the term may or may not be in DNF_l which implies $f_l(x)$ is either 0 or 1. This shows that

$$\forall x \in \{0,1\}^n, f_l(x) \leq f(x)$$

Since each term that is removed contribute to at most $\frac{\epsilon}{m}$ fraction of the inputs in $\{0,1\}^n$, the probability that f and f_l disagree is at most ϵ. So we have

$$Pr_{x\sim\mathcal{U}}[f(x) \neq f_l(x)] = \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[|f(x) - f_l(x)|] \leq \epsilon$$

Now we let f_u to be the function computed by DNF obtained by shortening all the terms of length greater than $\log\left(\frac{m}{\epsilon}\right)$ to length $\log\left(\frac{m}{\epsilon}\right)$from the DNF that computes f. Let us denote this DNF with DNF_u. For $x \in \{0,1\}^n$, if $f(x) = 1$ then there exists a term in the DNF computing f that is 1. This term, whether or not it is shortened, will make the DNF_u to be 1. On the other hand, if $f(x) = 0$, although all terms of DNF computing f is 0, some of the shortened term in the DNF$_u$ may evaluate to 1. This establishes that

$$\forall x \in \{0,1\}^n, f(x) \leq f_u(x)$$

Furthermore, observe the following fact.

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x \sim \mathcal{U}}[||f(x) - f_u(x)||] \\
&= Pr_{x \sim \mathcal{U}}[f(x) \neq f_u(x)] \\
&= Pr_{x \sim \mathcal{U}}[f(x) = 0 \,\text{and} f_u(x) = 1] \\
&= Pr_{x \sim \mathcal{U}}[f(x) = 0 \,\text{and at least one of the shortened terms in DNF}_u\text{ is } 1] \\
&\leq m \times Pr_{x \sim \mathcal{U}}[f(x) = 0 \,\text{and a term of width} \log\left(\frac{m}{\epsilon}\right) \text{ in DNF}_u \text{ is } 1] \\
&\leq m \times Pr_{x \sim \mathcal{U}}[\text{a term of width} \log\left(\frac{m}{\epsilon}\right) \text{ in DNF}_u \text{ is } 1] \\
&= m \times \frac{\epsilon}{m} = \epsilon
\end{aligned}$$

□

When the sandwiching approximators have small L_1 norms, we have the following useful fact which states that a small ϵ-biased set can also fool the sandwiched function.

Lemma 2.1.30. *Suppose $f, f_l, f_u : \{0,1\}^n \to \mathbb{R}$ are such that for every $x \in \{0,1\}^n$ we have $f_l(x) \leq f(x) \leq f_u(x)$ and $\mathbb{E}_{x \sim \mathcal{U}}[f(x) - f_l(x)] \leq \delta$ and $\mathbb{E}_{x \sim \mathcal{U}}[f_u(x) - f(x)] \leq \delta$. Let $l = max(||f_l||_1^{\neq \emptyset}, ||f_u||_1^{\neq \emptyset})$. Then any ϵ-biased distribution $(\delta + \epsilon l)$ fools f*

Proof. Let $\mathcal{D}$ be an ϵ-biased distribution. Then

$$\begin{aligned}
\mathop{\mathbb{E}}_{x \sim \mathcal{D}}[f(x)] &\leq \mathop{\mathbb{E}}_{x \sim \mathcal{D}}[f_u(x)] \\
&\leq \mathop{\mathbb{E}}_{x \sim \mathcal{U}}[f_u(x)] + \epsilon ||f_u||_1^{\neq \emptyset} \\
&\leq \mathop{\mathbb{E}}_{x \sim \mathcal{U}}[f(x)] + \delta + \epsilon ||f_u||_1^{\neq \emptyset}
\end{aligned}$$

Similarly we have $\mathbb{E}_{x \sim \mathcal{D}}[f(x)] \geq \mathbb{E}_{x \sim \mathcal{U}}[f(x)] - \delta - \epsilon ||f_u||_1^{\neq \emptyset}$ □

2.2 Main Results

In this section, we will briefly cover some previous works on sparse polynomial approximation which involves randomness and compare our result with these works.

2.2.1 Estimation of Fourier coefficients from random examples

Hoeffding bound is an important tool in a randomized learning algorithm because it gives the upper bound on the number of examples needed to well estimate the mean of a given random variable.

Theorem 2.2.1 (Hoeffding Bound)**.** *Let $X_1, X_2, ..., X_m$ be random variables that are independent, all with mean $\mathbb{E}[X_i] = \mu$ such that $a \leq X_i \leq b$ for all i. Then for any $\lambda > 0$,*

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

Theorem 2.2.2. *Given access to random examples for $f : \{-1,1\}^n \to \{-1,1\}$, there exists an algorithm that on input $0 < \epsilon, \delta < 1/2$ runs in $\mathsf{poly}(n, 1/\epsilon)\ln(1/\delta)$ and outputs an approximation $\tilde{f}(S)$ for $\hat{f}(S)$ such that*

$$Pr\left[|\tilde{f}(S) - \hat{f}(S)| \geq \epsilon\right] \leq \delta$$

Proof. After getting examples $(x, f(x))$,the algorithm outputs $\tilde{f}(S) = \frac{\sum_{i=1}^{m} f(x_i)\chi_S(x_i)}{m}$. Note that random variables $-1 \leq f(x_i)\chi_S(x_i) \leq 1$ are independent and all have mean $\hat{f}(S)$. We can apply Hoeffding bound and get

$$Pr\left[\left|\tilde{f}(S) - \hat{f}(S)\right| \geq \epsilon\right] \leq 2e^{\frac{-2\epsilon^2 m}{4}}$$

Setting the right hand side of the inequality to δ, we get that $m = \frac{2\ln(2/\delta)}{\epsilon^2}$ □

Our result is a deterministic estimation of the mean of $f(x)\chi_S(x)$ using membership queries. As opposed to the sample bound obtained from Hoeffding bound in a randomized estimation, our sample bound will be determined by our choice PRG.

2.2.2 Goldreich and Levin Algorithm: Application in Learning of Boolean functions

In this section, we will discuss the Kushilevitz and Mansour version [KM93] of Goldreich and Levin algorithm. As alluded in Remark 2.1.18, the algorithm outputs all parity functions that are at least slightly correlated with a given function. The following is the formal statement of Kushilevitz and Mansour's version.

Theorem 2.2.3. *There is a randomized algorithm, that given a query access to any Boolean function $f : \{-1,1\} \to \{-1,1\}$, any $\delta > 0$, any $0 < \theta \leq 1$ runs in time $\mathsf{poly}(n, 1/\theta, \log(1/\delta))$ and outputs a list L of vectors $\alpha \in \{0,1\}^n$ such that with probability at least $1 - \delta$,*

- $\left|\hat{f}(\alpha)\right| \geq \theta \implies \alpha \in L$

- $\alpha \in L \implies \left|\hat{f}(\alpha)\right| \geq \theta/2$

Remark 2.2.4. *Note that, by Parseval's Theorem, the second property of L implies that $|L| \leq 4/\theta^2$. Upon the output L of above algorithm, if we want to build a hypothesis, we need to estimate $\hat{f}(\alpha)$ for each of the $\alpha \in L$ by using a randomized process such as Theorem 2.2.2 or using a deterministic process.*

As stated in Theorem 2.2.2, given $S \subseteq [n]$, we can efficiently check if $\hat{f}(S)$ is big within small error with high probability. However, to find all large coefficients using random examples, we will need to check all 2^n many coefficients for each $S \subseteq [n]$. Kushilevitz and Mansour solve this problem by using a recursive algorithm where in each level of recursion, buckets of Fourier coefficients are considered. That is, on level k of recursion, there can be at most 2^k buckets for each fixed $\alpha \in \{0,1\}^k$ and a bucket contains Fourier coefficients $\hat{f}(\alpha\beta)$ for each $\beta \in \{0,1\}^{n-k}$.

Remark 2.2.5. *Note that our notation for Fourier coefficients changed slightly. $\hat{f}(x) = \hat{f}(S)$ where $x \in \{0,1\}^n$ and $S = \{i : x_i = 1\}$*

The idea is that for each bucket, if the sum of squares of the coefficients is smaller than θ^2 for some threshold θ, then for no $\beta \in \{0,1\}^{n-k}$, $\left|\hat{f}(\alpha\beta)\right| \geq \theta$. Therefore we don't need to consider further about coefficients with the prefix $\alpha \in \{0,1\}^k$ and this prunes the recursion tree which greatly reduce the number of recursive calls. Furthermore, on each level of recursion, there can be at most $\frac{1}{\theta^2}$ many buckets such that the sum of squares of the coefficients in the bucket is greater than θ^2. This is because, for $f : \{0,1\}^n \to \{-1,1\}$, we have $\sum_{\alpha\in\{0,1\}^k} \sum_{\beta\in\{0,1\}^{n-k}} \hat{f}^2(\alpha\beta) = 1$. If the sum of squares of the coefficients in a bucket is greater than θ^2 then we recurs with $\alpha 0$ and $\alpha 1$. Since there are n levels of recursion, the total number of recursive call will be at most $\frac{n}{\theta^2}$.

The only problem that remains is whether or not we can efficiently compute exactly or approximate the sum of squares of the coefficients in a bucket. Computing the sum exactly is not achievable in polynomial time, so it needs to be approximated within small error.

To make the above intuition more precise, let us introduce a few notations. For a given function $f : \{0,1\}^n \to \{-1,1\}$ with its Fourier expansion $f(x) = \sum_{z\in\{0,1\}^n} \hat{f}(z)\chi_z(x)$, we define $f_\alpha(x) : \{0,1\}^{n-k} \to \mathbb{R}$ for $\alpha \in \{0,1\}^k$ as follows

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

Remark 2.2.6. *Note that f_α defined above can be obtained by first collecting all terms $\hat{f}(z)\chi_z(x)$ in the Fourier expansion of f such that the prefix of z is the same as α and then restricting each of the terms by $x \in \{0,1\}^{n-k}$ and summing up the resulting coefficients of the restricted monomials. In other words, f_α is the Fourier coefficient for χ_α of the* *restriction of f. Therefore, denoting the restriction of f by $x \in \{0,1\}^{n-k}$ with f_x, we get*

$$f_\alpha(x) = \sum_{\beta \in \{0,1\}^{n-k}} \hat{f}(\alpha\beta)\chi_\beta(x) = \hat{f}_x(\alpha) \tag{2.2.1}$$

Also, note that $||f_\alpha||_1 \leq ||f||_1$ by definition.

Recall that we need to estimate in each bucket the sum of squares of the Fourier coefficients

$$\sum_{\beta \in \{0,1\}^{n-k}} \hat{f}^2(\alpha\beta) = \mathop{\mathbb{E}}_{x \in \{0,1\}^{n-k}} \left[f_\alpha^2(x)\right] \tag{2.2.2}$$

Also recall that the learning model only has black box access to query f. The most crucial fact that enables us to estimate this quantity is that, as stated in Remark 2.2.6, f_α is the Fourier coefficient of f_x. Having a query access to f allows a query access to any restriction of f. Also, note that by the definition of Fourier coefficient,

$$\hat{f}_x(\alpha) = \mathop{\mathbb{E}}_{y \in \{0,1\}^k} [f_x(y)\chi_\alpha(y)] \tag{2.2.3}$$

We know from Theorem 2.2.2 that this Fourier coefficient can be approximated efficiently by getting random samples $(y, f_x(y))$. Combining Equation (2.2.1), Equation (2.2.2), and Equation (2.2.3) we get

$$\sum_{\beta \in \{0,1\}^{n-k}} \hat{f}^2(\alpha\beta) = \mathop{\mathbb{E}}_{x \in \{0,1\}^{n-k}} \left[\left(\mathop{\mathbb{E}}_{y \in \{0,1\}^k} [f_x(y)\chi_\alpha(y)]\right)^2\right] \tag{2.2.4}$$

Kushilevitz and Mansour then show that getting good approximations by random samples $(y_1, f_{x_i}(y_1)), ..., (y_{m_2}, f_{x_i}(y_{m_2}))$for $\hat{f_{x_i}}(\alpha)$ for each of $x_1, ..., x_{m_1}$ and averaging over the squares of the approximations gives a good approximation for the true value of $\mathbb{E}_{x \in \{0,1\}^{n-k}}[f_\alpha^2(x)]$. Let us denote the approximation for $\mathbb{E}_{x \in \{0,1\}^{n-k}}[f_\alpha^2(x)]$ with B_α. They show the following which proves the correctness of the algorithm in Theorem 2.2.3.

- $\mathbb{E}_{x \in \{0,1\}^{n-k}}[f_\alpha^2(x)] \geq \theta^2 \implies B_\alpha \geq \theta^2/2$
- $\mathbb{E}_{x \in \{0,1\}^{n-k}}[f_\alpha^2(x)] \leq \theta^2/4 \implies B_\alpha \leq \theta^2/2$

2.2.3 Derandomization of Goldreich Levin algorithm for small L_1 norm functions

Lemma 2.2.7. *Given $f : \{0,1\}^n \to \mathbb{R}$, if $||f||_1 \leq l$ then $||f_x||_1 \leq l$*

Proof. It follows immediately from the definition of Fourier expansion and L_1 norm. That is

$$||f_x||_1 = \sum_{s \subseteq [n]} |\hat{f}_x(S)| \leq \sum_{s \subseteq [n]} |\hat{f}(S)| = ||f||_1 \leq l$$

□

Lemma 2.2.8. *Given $f : \{0,1\}^n \to \mathbb{R}$, if $||f||_1 \leq l$ then $||f \cdot \chi_S||_1 \leq l$ for any $S \subseteq [n]$.*

Proof. Since $\chi_S \cdot \chi_T = \chi_{S \triangle T}$ where $S \triangle T$ denotes the symmetric difference of S and T, we have $f(x) \cdot \chi_S = \sum_{T \subseteq [n]} \hat{f}(T)\chi_T \cdot \chi_S = \sum_{T \subseteq [n]} \hat{f}(T)\chi_{T \triangle S}$. From the uniqueness of Fourier expansion, we get $||f||_1 = ||f \cdot \chi_S||_1$. □

Using Lemma 2.2.8,Lemma 2.1.25, and Theorem 2.1.24 we get the following theorem.

Theorem 2.2.9. *For $f : \{0,1\}^n \to \mathbb{R}$ such that $||f||_1 \leq l$, there is a deterministic algorithm that given query access to f, $S \subseteq [n]$, and $0 < \epsilon \leq 1/2$, and $1 \leq l$ runs in time $\mathsf{poly}(l, n/\epsilon)$ and outputs an approximation $\tilde{f}(S)$ for $\hat{f}(S)$ such that*

$$|\tilde{f}(S) - \hat{f}(S)| \leq \epsilon$$

Proof. Let $\mathcal{D}$ be an ϵ/l-biased distribution. By Theorem 2.1.24, time to construct the support of this distribution is $\mathsf{poly}(n, l, 1/\epsilon)$. By Lemma 2.2.8 and Lemma 2.1.25 we see that $|\mathbb{E}_{x \sim \mathcal{D}}[f(x) \cdot \chi_S(x)] - \mathbb{E}_{x \sim \mathcal{U}}[f(x) \cdot \chi_S(x)]| \leq \epsilon$. □

In order to derandomize Goldreich and Levin algorithm, we need a deterministic way of approximating the Fourier coefficient of a restricted function. We can use a small biased set that fools f to estimate $\mathbb{E}_{y \in \{0,1\}^k}[f_x(y)\chi_\alpha(y)]$ because of the following fact.

Lemma 2.2.10. *Let $f : \{0,1\}^n \to \mathbb{R}$ such that $||f||_1 \leq l$ and $J \subseteq [n]$ with $|J| = k$. Let us denote by $\{0,1\}^J$ length k bit strings that corresponds to the coordinates of $J \subseteq [n]$ where J does not need to be consecutive. Let $\mathcal{D}$ be an ϵ/l biased distribution over $\{0,1\}^n$ defined by a probability mass function $\phi : \{0,1\}^n \to \mathbb{R}^{\geq 0}$. Then the marginal distribution $\mathcal{D}^J$ on $\{0,1\}^J$ defined by $\phi' : \{0,1\}^J \to \mathbb{R}^{\geq 0}$ where $\phi'(y) = \sum_{x \in \{0,1\}^{\bar{J}}} \phi(yx)$ for each $y \in \{0,1\}^J$ is still ϵ/l biased on $\{0,1\}^J$. That is*

$$\left| \mathop{\mathbb{E}}_{y \sim \mathcal{D}^J}[\chi_\alpha(y)] - \mathop{\mathbb{E}}_{y \in \{0,1\}^J}[\chi_\alpha(y)] \right| \leq \epsilon/l$$

for any $\alpha \in \{0,1\}^J$

Proof.

$$\begin{aligned}
&\left|\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\chi_\alpha(x)] - \mathop{\mathbb{E}}_{x\in\{0,1\}^n}[\chi_\alpha(x)]\right| \\
&= \left|\sum_{y\in\{0,1\}^k}\sum_{x\in\{0,1\}^{n-k}}\phi(yx)\chi_\alpha(yx) - \sum_{y\in\{0,1\}^k}\sum_{x\in\{0,1\}^{n-k}}\frac{1}{2^n}\chi_\alpha(yx)\right| \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{(since } \alpha\in\{0,1\}^k,\ \chi_\alpha(x)=1) \\
&= \left|\sum_{y\in\{0,1\}^k}\sum_{x\in\{0,1\}^{n-k}}\phi(yx)\chi_\alpha(y) - \sum_{y\in\{0,1\}^k}\sum_{x\in\{0,1\}^{n-k}}\frac{1}{2^n}\chi_\alpha(y)\right| \\
&= \left|\sum_{y\in\{0,1\}^k}\phi'(y)\chi_\alpha(y) - \sum_{y\in\{0,1\}^k}\frac{1}{2^k}\chi_\alpha(y)\right| \\
&= \left|\mathop{\mathbb{E}}_{y\sim\mathcal{D}^J}[\chi_\alpha(y)] - \mathop{\mathbb{E}}_{y\in\{0,1\}^k}[\chi_\alpha(y)]\right| \leq \epsilon/l
\end{aligned}$$

□

We need one more fact to derandomize Goldreich and Levin algorithm for small L_1 norm functions.

Lemma 2.2.11. *For $f:\{0,1\}^n\to\mathbb{R}$ with $||f||_1\leq l$, and an ϵ-biased distribution $\mathcal{D}$ on $\{0,1\}^n$,*

$$\left|\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[f^2(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f^2(x)]\right| \leq l^2\epsilon$$

Proof.

$$\begin{aligned}
\left|\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[f^2(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f^2(x)]\right| &\leq ||f^2||_1\epsilon && \text{(by Lemma 2.1.25)} \\
&\leq ||f||_1^2\epsilon \leq l^2\epsilon && \text{(by Lemma 2.1.7)}
\end{aligned}$$

□

Recall that the quantity that we need to approximate is the expectation over $\{0,1\}^{\bar{J}}$ of the square of a function with domain $\{0,1\}^J$ for some $J\subseteq[n]$ as we see in Equation (2.2.2).Let us assume $||f||_1\leq l$. Let us construct an ϵ/l^2-biased set B according to Theorem 2.1.24, for $\{0,1\}^n$. Let $\mathcal{D}_B$ be the ϵ/l^2-biased distribution over B. Then by Lemma 2.2.10 and by Lemma 2.2.11, we can use the marginal distribution of $\mathcal{D}_B$ to estimate this quantity. In other words, the marginal distribution $\mathcal{D}_B^{\bar{J}}$ on $\{0,1\}^{\bar{J}}$ will fool $f_\alpha^2(x)$.

$$\begin{aligned}
\left|\mathop{\mathbb{E}}_{x\sim\mathcal{D}_B^{\bar{J}}}[f_\alpha^2(x)] - \mathop{\mathbb{E}}_{x\sim\{0,1\}^{\bar{J}}}[f_\alpha^2(x)]\right| &\leq ||f_\alpha(x)||_1^2\frac{\epsilon}{l^2} && \text{(by Lemma 2.2.11)} \\
&\leq ||f(x)||_1^2\frac{\epsilon}{l^2}\leq\epsilon && \text{(by Remark 2.2.6)}
\end{aligned}$$

One more problem remains. That is, we don't have a query access to f_α. However by Remark 2.2.6, we can estimate its value by querying f_x. As mentioned earlier, since the algorithm can query into f for any input of the domain, it also has a query access to f_x for any $x \in \{0,1\}^{n-k}$. Furthermore, by Lemma 2.2.7 and Lemma 2.2.8, the L_1 norm of $f_x(y)\chi_S(y)$ is at most the L_1 norm of f. The same ϵ/l^2-biased set B can be used to estimate the expectation of $f_x(y)\chi_S(y)$ over $\{0,1\}^J$. In other words, the marginal distribution $\mathcal{D}_B^J$ on $\{0,1\}^J$ will fool $f_x(y)\chi_S(y)$.

$$\left|\mathop{\mathbb{E}}_{y\sim\mathcal{D}_B^J}[f_x(y)\chi_S(y)] - \mathop{\mathbb{E}}_{y\sim\{0,1\}^J}[f_x(y)\chi_S(y)]\right| \leq ||f(x)||_1\frac{\epsilon}{l^2} \leq \frac{\epsilon}{l} \tag{2.2.5}$$

Note that if we get an approximation β of $\mathbb{E}_{x\sim\mathcal{D}_B^{\bar{J}}}[f_\alpha^2(x)]$ such that $\left|\beta - \mathbb{E}_{x\sim\mathcal{U}_B'}[f_\alpha^2(x)]\right| \leq \epsilon$ then we have

$$\begin{aligned}&\left|\beta - \mathop{\mathbb{E}}_{x\sim\{0,1\}^{\bar{J}}}[f_\alpha^2(x)]\right| \\ &\leq \left|\beta - \mathop{\mathbb{E}}_{x\sim\mathcal{D}_B^{\bar{J}}}[f_\alpha^2(x)]\right| + \left|\mathop{\mathbb{E}}_{x\sim\mathcal{D}_B^{\bar{J}}}[f_\alpha^2(x)] - \mathop{\mathbb{E}}_{x\sim\{0,1\}^{\bar{J}}}[f_\alpha^2(x)]\right| \leq 2\epsilon\end{aligned}$$

Since f_α is a Fourier coefficient of a Boolean function, we have that $|f_\alpha| \leq 1$. By Equation (2.2.5),we can approximate f_α within ϵ/l. Let us denote this approximation of f_α with α. Then we have the following

$$\alpha - f_\alpha \leq \epsilon/l \implies \alpha^2 - f_\alpha^2 \leq f_\alpha^2 + 2\epsilon/lf_\alpha + \epsilon^2/l^2 - f_\alpha^2 \leq 3\epsilon/l$$

$$f_\alpha - \alpha \leq \epsilon/l \implies f_\alpha^2 - \alpha^2 \leq f_\alpha^2 - f_\alpha^2 + 2\epsilon/lf_\alpha - \epsilon^2/l^2 \leq 2\epsilon/l$$

So α^2 is $3\epsilon/l$ approximation of f_α^2. Now we have the approximation β of $\mathbb{E}_{x\sim\mathcal{U}_B^{\bar{J}}}[f_\alpha^2(x)]$ such that $\left|\beta - \mathbb{E}_{x\sim\mathcal{U}_B^{\bar{J}}}[f_\alpha^2(x)]\right| \leq 3\epsilon/l$. That is we enumerate each $x \sim \mathcal{U}_B^{\bar{J}}$ and get $3\epsilon/l$-estimate of f_α^2 as described by above. Then the equally weighted average of the estimates of f_α^2 will be $3\epsilon/l$-estimate of $\mathbb{E}_{x\sim\mathcal{U}_B^{\bar{J}}}[f_\alpha^2(x)]$. In conclusion, we have the following.

Theorem 2.2.12. *There exists a deterministic algorithm that given a query access to any $f : \{0,1\}^n \to \{0,1\}$ such that $||f||_1 \leq l$, any $\theta > 0$ runs in time $\mathsf{poly}(n, 1/\theta)$ and outputs a list L of vectors $\alpha \in \{0,1\}^n$ such that*

- $\left|\hat{f}(\alpha)\right| \geq \theta \implies \alpha \in L$
- $\alpha \in L \implies \left|\hat{f}(\alpha)\right| \geq \theta/2$

Our result is a deterministic approximation of $\mathbb{E}_{x\in\{0,1\}^{\bar{J}}}[f_\alpha^2(x)]$ where f is computed by a DNF of polynomial number of terms for small S_α where $S_\alpha = \{i : \alpha_i = 1\}$ provided that there exists a good pseudo random generator for DNF. It is a derandomization of Goldreich and Levin algorithm that does not depend on f having the small L_1 norm. Instead our result is for the special case when we only need to find small degree coefficients and conditioned with the existence of an ideal PRG for DNFs.

2.2.4 Sparse polynomial approximation

The following lemma says that if f can be approximated by a sparse polynomial g, then f can be approximated by h that has only large coefficients of g.

Theorem 2.2.13. *[KM93] If f can be approximated by a t-sparse function g such that $\mathbb{E}[(f-g)^2] \leq \epsilon$, then there exists a t-sparse function h such that $\mathbb{E}[(f-h)^2] \leq \epsilon + O\left(\epsilon^2/t\right)$ and all the nonzero coefficients of h are at least ϵ/t*

Remark 2.2.14. *Without loss of generality, Kushilevitz and Mansour assumes that the nonzero coefficients of the sparse functions g are the same as the coefficients of f of the corresponding vectors. h is then defined with the coefficients of g that are at least ϵ/t. In other words*

$$g(x) = \sum_{i=1}^{t} \hat{f}(z_i)\chi_{z_i}(x), \;\; h(x) = \sum_{\hat{f}(z_i)\geq \epsilon/t} \hat{f}(z_i)\chi_{z_i}(x)$$

Since h is the same as g except at most t many small coefficients where each of them contributes at most ϵ/t, we can see that the increase in the L_2 norm squared distance from h to f compared to the L_2 norm squared distance from g to f is at most $\left(\frac{\epsilon}{t}\right)^2 t$.

Remark 2.2.15. *We can see that f is ϵ approximated by a t-sparse function g, if and only if there exists a list L of vectors $\alpha \in \{0,1\}^n$ such that $\sum_{z\notin L} \hat{f}^2(z) \leq \epsilon$ and $|L| \leq t$. Therefore, we can equivalently define the function h in Theorem 2.2.13 as*

$$h = \sum_{\alpha\in L and \hat{f}(\alpha)\geq\epsilon/t} \hat{f}(\alpha)$$

The L_2 norm squared distance between h and f is then,

$$\sum_{\alpha\in\{0,1\}^n} \left(\hat{f}(\alpha) - \hat{h}(\alpha)\right)^2 = \sum_{\alpha\notin L} \hat{f}^2(\alpha) + \sum_{\alpha\in L and \hat{f}(\alpha)\leq\epsilon/t} \hat{f}^2(\alpha) \leq \epsilon + t\left(\frac{\epsilon}{t}\right)^2$$

Mansour's [Man92] main contribution in the paper was to show that for a w-DNF, there exist a sparse function that ϵ-approximates the DNF, which is a sufficient condition in Theorem 2.2.13. Furthermore, he showed that When we are limited to a w-DNF, finding large coefficients of small degree and approximating them gives the desired bound on the sum of squared difference. That is, for a w-DNF, the list L as described as above only consists of small degree vectors.

Remark 2.2.16. *Note that in Theorem 2.2.13, only the existence of a sparse polynomial is mentioned without an explicit construction for it. In fact, in [Man92], the sparse polynomial* *consists of a subset of Fourier coefficients of the target function which we do not have an access to in our learning model. However, it narrows down the number of Fourier coefficients that we need to estimate.*

The sparsity comes from the fact that a w-DNF has small Fouier concentration for large degree and small sum of absolute values of Fourier coefficients of small degree. This is formally stated in the following two lemmas.

Lemma 2.2.17. *[Man92] Let f be a function that can be written as a DNF with terms of size w. Then,*

$$\sum_{|S|>20w\log 2/\epsilon} \hat{f}^2(S) \leq \epsilon$$

Lemma 2.2.18. *[Man92] Let f be a function that can be written as a DNF with terms of size w. Then,*

$$\sum_{S:|S|\leq\gamma} |\hat{f}(S)| \leq w^{O(\gamma)}$$

Since the total sum of absolute values of small degree Fourier coefficients for a w-DNF f is upper bounded, the number of small degree coefficients that can have its absolute value larger than a certain threshold is upper bounded. More specifically, denoting $L = \sum_{S:|S|\leq\gamma} |\hat{f}(S)|$ where $\gamma = 20w\log\frac{1}{\epsilon}$, Mansour sets the threshold to $\frac{\epsilon}{L}$. Let us denote the set of Fourier coefficients of f with absolute value larger than $\frac{\epsilon}{L}$ and degree smaller than γ as G. So

$$G = \left\{ s : |s| \leq \gamma \text{ and } |\hat{f}(s)| \geq \frac{\epsilon}{L} \right\}$$

Then there can be at most $\left(\frac{L}{\epsilon}\right)^2$ elements in G. The sparse function g that approximates f is exactly the function with those coefficients in G. Mansour then shows, by simple analysis, that the Fourier mass of f outside G is bounded by ϵ which proves that the sparse function g is indeed ϵ close to f in L_2 norm squared.

Theorem 2.2.19. *[Man92] For any function f that can be described by a DNF with terms of size d there exists an t-sparse function g that ϵ-approximates f and $t \leq d^{O(d\log 1/\epsilon)}$*

So if there exists an algorithm that finds coefficients larger than some threshold, then the sparse function h mentioned in Theorem 2.2.13 can be found explicitly. As we discussed in the previous section, based on the idea of Goldreich and Levin [GL89], Kushilevitz and Mansour [KM93] presents a randomized algorithm that given a membership query access to a function f and a threshold θ, finds with probability at least $1-\delta$ in $poly(n, 1/\theta, log(1/\delta))$ time, every $S \subseteq [n]$ for which $|\hat{f}(S)| \geq \theta$.

Since Mansour showed that a t-sparse function exists in Theorem 2.2.19, we can use Kushilevitz and Mansour algorithm with threshold ϵ/t to find the list of $S \subset [n]$ such that $|\hat{f}(S)| \geq \epsilon/t$. Then according to Theorem 2.2.13, the function defined by the positive

Fourier coefficients of the vectors in the output list of the KM algorithm will $\epsilon + O\left(\epsilon^2/t\right)$ approximate the target DNF f. Since we do not have access to the coefficients of f, we estimate them upon the output of the Kushilevitz and Mansour's algorithm where each coefficient can be approximated efficiently with the standard sampling as in Theorem 2.2.2. This approximated coefficients are then used to build a sparse polynomial that is close to f in L_2 norm.

As explained in Remark 2.2.4, the output list from KM algorithm contains at most t^2/ϵ vectors for which we need to approximate the coefficients. Therefore we need to approximate each of them so that the squared difference of the target function's coefficient and the estimated coefficient is at most our desired total squared difference divided by the size of the output list. Since our desired total squared difference is ϵ, the approximation error will be ϵ^2/t^2. The run time to do this is still in $\mathsf{poly}(t, 1/\epsilon)$ as shown in Theorem 2.2.2.

Theorem 2.2.20. *[KM93] Let f be a Boolean function such that there exists a t-sparse function that ϵ-approximates f. There exists a randomized algorithm whose running time is polynomial in $t, n, 1/\epsilon$ and $\log \frac{1}{\delta}$ that given f(as a black box) and $\delta > 0$ outputs a function h, such that the probability that h $O(\epsilon)$ approximates f is at least $1 - \delta$.*

Mansour makes an assumption that the largest term of the target DNF is $O(\log(n/\epsilon))$ which is valid because the sample points in the learning model is uniformly randomly selected. The sparsity of the function approximating the target DNF is, by Theorem 2.2.19, $(n/\epsilon)^{O(\log\log(n/\epsilon)\log(1/\epsilon))}$ which dominates the run time of Mansour's algorithm [Man92] according to Theorem 2.2.20. [Tal17] later has improved this sparsity result as follows.

Theorem 2.2.21. *[Tal17] Given a DNF $f : \{0,1\}^n \to \{0,1\}$of m terms and $\epsilon > 0$, there exists a t-sparse function $g : \{0,1\}^n \to \mathbb{R}$ such that $||f - g||_2 \leq \epsilon$ and $t \leq (\log(m))^{O(\log(m)\log(1/\epsilon))}$*

Chapter 3

Deterministic approximation of the Fourier coefficients

Learning of a Boolean function can be done if we can estimate its Fourier coefficients such that the sum of squared difference of the true and the estimated coefficients is small. In the case of [LMN93], it estimates each coefficients $\hat{f}(S)$ for each S of size less than some bound by randomly sampling x from the input space to get an average over $f(x)\chi_S$ and set all large degree coefficients to zero. This is taking advantage of the fact that AC^0 circuits have Fourier representations such that most of its coefficients is concentrated at small degree. Tal [Tal17] has later improved this concentration bound as following

Theorem 3.0.1 ([Tal17])**.** *Let f be a Boolean circuit with depth d and size m. Then*

$$\textstyle\sum_{s:|s|\geq k} \hat{f}(s)^2 \leq 2 \cdot 2^{-k/O(\log m)^{d-1}}$$

Remark 3.0.2. *From the above theorem, a Boolean circuit with depth d ad size m has at most ϵ of its Fourier mass on degree greater than $O\left(\log^{d-1} m \log(1/\epsilon)\right)$*

3.1 Estimation based on conditional expectation and GMR pseudorandom generator

Now we will use Gopalan and others' PRG [GMR13] for DNFs to deterministically approximate the Fourier coefficients. Before we do so, we need to express a Fourier coefficient in terms of expectations of restricted DNFs as stated in the following lemma.

Lemma 3.1.1. *Given $f\colon \{0,1\}^n \to \mathbb{R}$ and a subset $S \subseteq [n]$ of size t, let f_x denote f restricted on the coordinates corresponding to S with $x \sim \{0,1\}^t$, then we have*

$$\mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f(x)\chi_S(x)] = \frac{1}{2^t} \sum_{x\sim\{0,1\}^t} \chi_S(x) \mathop{\mathbb{E}}_{y\sim\{0,1\}^{n-t}}[f_x(y)]$$

Note that if f is in a circuit class that is closed under restriction, then we can use the PRG for the circuit class to fool f_x. The class of functions computable by DNFs is closed under restriction. Furthermore, note that if f can be computed by a w-DNF than f_x can also be computed by a w-DNF.

Our learning model has black box query access to an unknown target function and this in turn gives a black box query access to a restriction of the target function.

The runtime of derandomized algorithm using a PRG depends on the seed length and the seed length in turn depends on various parameters. In the case of Gopalan and others' PRG, the the dependency is on the maximum width of the terms.

Theorem 3.1.2 ([GMR13])**.** *For all δ, there exists an explicit generator $G\colon \{0,1\}^r \to \{0,1\}^n$ that δ-fools width w-DNF and has seed length*

$$r = \tilde{O}\left(w^2 + w\log\left(\frac{1}{\delta}\right) + \log\log(n)\right)$$

It is therefore desirable that our DNF has small width. In the case of randomized learning where the examples are randomly selected, we can ignore the large terms as we explain in the following. When we want to learn a DNF formula of size $m = \text{poly}(n)$ up to ϵ error probability, we can assume without loss of generality that our DNF formula has width at most $O(\log(m/\epsilon))$. This is because the probability that a term larger than $\log(\frac{m}{\epsilon})$ is satisfied is at most $\frac{\epsilon}{m}$ and so the total error probability will increase by at most $O(\epsilon)$. Since we are discussing DNFs with polynomial number of terms, we can then assume that the width is $O(\log(n/\epsilon))$.

This simplified DNF with only short terms is then learned by the randomized learning algorithm. Note that even if the learning model only has back box access to the original function , in randomized case, this simplification does not pose a problem because, for each random example, the probability that the original function and the simplified one disagree is negligible. Therefore, assuming that we need at most $\mathsf{poly}(n)$ number of examples to learn the DNF with only short terms, the original DNF and the simplified one will agree on all of the examples with high probability except at most $\mathsf{poly}(n)\epsilon$.

With deterministic learning, in contrast, this assumption cannot be justified. To see this, we can imagine the worst case scenario when the set of example points that we make queries to the black box of original function includes all points that the simplified function disagrees with the original one.

Combining Lemma 3.1.1 and Theorem 3.1.2 we get the following theorem

Theorem 3.1.3. *For a given width w-DNF, there is a deterministic procedure to ϵ-approximate its Fourier coefficient of degree less than t in time $2^{\tilde{O}\left(w^2+w\cdot\log\left(2^t/\epsilon\right)+\log\log n\right)} \cdot 2^t$*

Proof. In order to approximate each coefficient within error ϵ, one term in the summation should be approximated within $\frac{\epsilon}{2^t}$. Then the seed length follows from Theorem 3. □

Derandomized LMN learning for DNF To derandomize the LMN learning for the case of width w-DNF, we use Theorem 3.0.1 for a better concentration bound. So we need to estimate the coefficients of degree at most $O(\log n \log(1/\epsilon))$. As it is in LMN, we will set the coefficients of larger degree to zero. Therefore,the number of coefficients to be estimated is at most $n^{O(\log n \log(1/\epsilon))}$. This means if we want the final hypothesis to ϵ-approximate f, then we need to have that the approximation of each coefficients is within at most $\frac{\epsilon}{n^{O(\log n \log(1/\epsilon))}}$ of the true value of the coefficient.

Using Theorem 3.1.3, this approximation of one coefficient takes

$$2^{\tilde{O}\left(w^2+w\log n\log(1/\epsilon)+\log\log n\right)} \cdot n^{\log(1/\epsilon)}$$

time. Therefore for a small w such that $w \leq log\,(n/\epsilon)$, we can deterministically learn a w-DNF formula in time $(n/\epsilon)^{\tilde{O}(\log(n/\epsilon)\log(1/\epsilon))}$.

Theorem 3.1.4. *Given a w-DNF and $\epsilon > 0$, where $w \leq \log(n/\epsilon)$, there exists a deterministic learning algorithm that outputs an ϵ approximating hypothesis in time $\left(\frac{n}{\epsilon}\right)^{\tilde{O}\left(\log\left(\frac{n}{\epsilon}\right)\log\frac{1}{\epsilon}\right)}$.*

Note that if we were to estimate the coefficient by directly fooling $f(x)\chi_s(x)$ using a pseudo random generator for the class of DNFs, then we would have to construct a large circuit for $f(x)\chi_s(x)$ in a DNF formula. This in turn will result in a long seed length. In contrast, our method does not suffer this overhead because, in each term of the summation, the parity is constant.

Read once DNF In the special case of read once DNFs, Gopalan and others [GMR$^+$12] showed that there exist a near optimal PRG.

Theorem 3.1.5. *[GMR$^+$12] For every $\epsilon > 0$ there exists an explicit PRG $G\colon \{0,1\}^r \to \{1,-1\}^n$ that fools all read once CNFs on n-variables with error at most ϵ and seed-length $r = O\left(\log(n/\epsilon)\left(\log\log(n/\epsilon)\right)^3\right)$*

It can be verified that a PRG that ϵ-fools the class of all read once CNFs also ϵ-fools the class of all read once DNFs. Since the class of read once DNFs is closed under restriction, we can use our Lemma 3.1.1 to approximate its coefficients.

Theorem 3.1.6. *Given a read once DNF, there is a deterministic procedure to ϵ-approximate a Fourier coefficient of degree less than t in time $2^{\tilde{O}(\log(n/\epsilon)+t)}$*

Proof. To estimate one coefficient of degree t within ϵ error, one term of the summation in lemma 3.1.1 needs to be approximated within $\frac{\epsilon}{2^t}$ which require $\tilde{O}\left(\log(n/\epsilon)+\log(2^t)\right)$ seed length according to theorem 3.1.5. □

Theorem 3.1.6 immediately gives a deterministic learning algorithm for read once DNFs that runs in time $n^{\tilde{O}(\log n \log(1/\epsilon))}$.

3.2 Estimation based on biased distribution that fools small L_1 norm functions

De and others [DETT10] showed that a w-DNF $\phi_w : \{0,1\}^n \to \{0,1\}$ can be fooled by a small ϵ-biased set by showing that ϕ_w has sandwich approximators that have small l_1 norms. Their argument starts with Mansour's [Man92] result that says a sparse polynomial defined with a subset of Fourier coefficients of ϕ_w can approximate ϕ_w. Then by Lemma 2.1.8, we have a small l_1 norm function approximating ϕ_w. They then follow the proof by Razborov [Raz09] to show that there exists a function g approximating ϕ_w that has slightly larger l_1 norm than that of Mansour's sparse polynomial but has an advantage that $g(x) = 0$ whenever $\phi_w(x) = 0$.
Finally, using the construction of Bazzi [Baz09], they construct sandwich approximators for ϕ_w with slightly larger l_1 norm then $||g||_1$. Then by using Lemma 2.1.30, the small ϵ-biased set that fools the approximators also fools ϕ_w. In fact, in the construction of the sandwiching approximators in [DETT10], the l_1 norm of both lower and upper sandwiching approximators are bounded by the same quantity which is the sparsity of the approximating polynomial that is shown to exists for a given w-DNF by [Man92]. Then they reduce the case of arbitrary DNFs with m terms to that of bounded width, by deleting the terms of width greater than $\log(m/\delta)$ and show that the distribution that $\delta/4$ fools the bounded width DNF δ-fools the original DNF.

We will use the improved sparsity result by Tal [Tal17] stated in Theorem 2.2.21 to show that there exists small l_1 norm sandwich approximators for an arbitrary DNF. Our proof closely follow the proof for w-DNF by [DETT10] and that is because the case of w-DNF can be generalized to a general DNF without breaking the proof of the lemmas that lead to the existence of the small L1 norm approximators.
The following lemma is proven in [DETT10].

Lemma 3.2.1 ([DETT10]). *Let $\phi : \{0,1\}^n \to \{0,1\}$ be a DNF with m terms and $g : \{0,1\}^n \to \mathbb{R}$ be such that: $||g||_1 \leq l, ||\phi - g||_2 \leq \epsilon_1$ and $g(x) = 0$ whenever $\phi(x) = 0$. Then we can construct $f_l, f_u : \{0,1\}^n \to \mathbb{R}$ such that*

- $\forall x, f_l(x) \leq \phi_(x) \leq f_u(x)$
- $\mathbb{E}_{x \sim \mathcal{U}}[f_u(x) - \phi_(x)] \leq m\epsilon_1^2$ *and* $\mathbb{E}_{x \sim \mathcal{U}}[\phi_(x) - f_l(x)] \leq m\epsilon_1^2$
- $||f_l||_1, ||f_u||_1 \leq (m+1)(l+1)^2 + 1$

The following lemma is a slight variant of Lemma4.4 of [DETT10]. The only change is that the DNF is not limited to a width w-DNF but it is a general DNF with m terms. However the proof of the lemma does not depend on the DNF being bounded width but what is required is that the DNF is estimated by a sparse polynomial.

Lemma 3.2.2. *Let $\phi : \{0,1\}^n \to \{0,1\}$ be a DNF with m terms. Suppose for every DNF ϕ_1 with at most m terms, there exists $g_1 : \{0,1\}^n \to \mathbb{R}$ such that: $|||g_1||_1 \leq l_1$ and $||\phi_1 - g_1||_2 \leq \epsilon_2$. Then we can get $g : \{0,1\}^n \to \mathbb{R}$ such that $||g||_1 \leq m(l_1+1), ||\phi - g||_2 \leq m\epsilon_2$ and $g(x) = 0$ whenever $\phi(x) = 0$.*

The proof closely follow the proof of [DETT10] but we will state it here for completeness

Proof. Given a DNF $\phi = \vee_{i=1}^m A_i$ where $A_i \in \{0,1\}$ are individual terms, we can write $\phi = \sum_{i=1}^m A_i \left(1 - \vee_{j=1}^{i-1} A_j\right)$ [Raz09]. Letting $\vee_{j=1}^{i-1} A_j = \phi_i$ where $\phi_i = 0$ if $i = 1$, we get $\phi = \sum_{i=1}^m A_i (1 - \phi_i)$. Since each ϕ_i is a DNF of at most m terms, by our hypothesis, for each ϕ_i, there exist $g_i : \{0,1\}^n \to \mathbb{R}$ such that: $|||g_i||_1 \leq l_1$ and $||\phi_i - g_i||_2 \leq \epsilon_2$. Now we define a function

$$g = \sum_{i=1}^m A_i (1 - g_i)$$

If $\phi(x) = 0$ then $A_i(x) = 0$ for all i therefore $g(x) = 0$. Using the fact that A_i is a conjunction and Remark 2.1.10 and Remark 2.1.9, we get $||g||_1 \leq m(l_1 + 1)$. Now

$$\begin{aligned}
||g - \phi||_2^2 &= \mathop{\mathbb{E}}_{x \in \{0,1\}^n} \left[\left(\sum_{i=1}^m A_i(\phi_i - g_i)(x) \right)^2 \right] \\
&\leq m \mathop{\mathbb{E}}_{x \in \{0,1\}^n} \left[\sum_{i=1}^m (A_i(\phi_i - g_i)(x))^2 \right] && \text{(By Jensen's inequality)} \\
&= m \sum_{i=1}^m \mathop{\mathbb{E}}_{x \in \{0,1\}^n} \left[(A_i(\phi_i - g_i)(x))^2 \right] \\
&\leq m \sum_{i=1}^m \mathop{\mathbb{E}}_{x \in \{0,1\}^n} \left[(\phi_i - g_i)(x)^2 \right] && (A_i \text{ is at most } 1) \\
&= m \sum_{i=1}^m ||\phi_i - g_i||_2^2 \leq m^2 \epsilon_2^2
\end{aligned}$$

□

With Tal's improved sparsity result in theorem 2.2.21, we have the following which is just a restatement of theorem 2.2.21.

Lemma 3.2.3. *Let $\phi : \{0,1\}^n \to \{0,1\}$ be a DNF with m terms and $\epsilon_2 > 0$. Then there exists $g_1 : \{0,1\}^n \to \mathbb{R}$ such that: $|||g_1||_1 \leq (\log(m))^{O(\log(m)\log(1/\epsilon_2))}$ and $||\phi - g_1||_2 \leq \epsilon_2$.*

Proof. By theorem 2.2.21, we get a $t \leq (\log m))^{O(\log m \log(1/\epsilon_2))}$-sparse function $g_1 : \{0,1\}^n \to \mathbb{R}$ such that $||\phi - g_1||_2 \leq \epsilon_2$. By Lemma 2.1.8, $||g_1|| \leq (\log m))^{O(\log m \log(1/\epsilon_2))}$. □

Combining Lemma 3.2.1, Lemma 3.2.2, and Lemma 3.2.3 we get the following result.

Lemma 3.2.4. *Let $\phi : \{0,1\}^n \to \{0,1\}$ be a DNF with m terms. Then we can construct $f_l, f_u : \{0,1\}^n \to \mathbb{R}$ such that*

- $\forall x, f_l(x) \leq \phi_(x) \leq f_u(x)$
- $\mathbb{E}_{x\sim\mathcal{U}}[f_u(x) - \phi_(x)] \leq \delta/2$ *and* $\mathbb{E}_{x\sim\mathcal{U}}[\phi_(x) - f_l(x)] \leq \delta/2$
- $||f_l||_1, ||f_u||_1 \leq (\log(m))^{O(\log(m)\log(m/\delta))}$

Proof. This proof follows the proof of [DETT10]. Set $\epsilon_2 = \sqrt{\delta/2m^3}$ and $\epsilon_1 = \sqrt{\delta/2m}$. By applying Lemma 3.2.3, for every DNF with at most m terms ϕ_1, we can get a function $g_1 : \{0,1\}^n \to \mathbb{R}$ such that

- $||\phi_1 - g_1|| \leq \epsilon_2 = \sqrt{\delta/2m^3}$
- $||g_1||_1 \leq (\log(m))^{O(\log(m)\log(1/\epsilon_2))} = (\log(m))^{O(\log(m)\log(m/\delta))}$

Now apply Lemma 3.2.2 with $l_1 = (\log(m))^{O(\log(m)\log(m/\delta))}$ and $\epsilon_2 = \sqrt{\delta/2m^3}$. Then for the given DNF ϕ, we get a function g such that $||g||_1 \leq (\log(m))^{O(\log(m)\log(m/\delta))}$ and $||g - \phi||_2 \leq m\epsilon_2 = \epsilon_1 = \sqrt{\delta/2m}$. Then, apply Lemma 3.2.1 with g and ϵ_1 as defined and $l = (\log(m))^{O(\log(m)\log(m/\delta))}$ to get f_l and f_u such that ϕ is sandwiched by f_l and f_u and $||f_l||_1, ||f_u||_1 \leq (\log(m))^{O(\log(m)\log(m/\delta))}$ and

$$\mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f_u(x) - \phi_(x)] \leq \delta/2 \text{ and } \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[\phi_(x) - f_l(x)] \leq \delta/2$$

□

In this section, we will use the following lemma that says that any function $f : \{0,1\}^n \to \mathbb{R}$ that is sandwiched by small l_1 norm functions $f_l, f_u : \{0,1\}^n \to \mathbb{R}$, a small biased set that fools the approximator also fools $f \cdot \chi_S$.

Lemma 3.2.5. *Suppose $f, f_l, f_u : \{0,1\}^n \to \mathbb{R}$ are such that for every $x \in \{0,1\}^n$ we have $f_l(x) \leq f(x) \leq f_u(x)$ and $\mathbb{E}_{x\sim\mathcal{U}}[f(x) - f_l(x)] \leq \delta$ and $\mathbb{E}_{x\sim\mathcal{U}}[f_u(x) - f(x)] \leq \delta$.Let $l = max(||f_l||_1^{\neq\emptyset}, ||f_u||_1^{\neq\emptyset})$. Let χ_S be an arbitrary Fourier basis for $S \subseteq [n]$. Let $\mathcal{D}$ be a ϵ-biased distribution. Then,*

$$|\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[f \cdot \chi_S] - \mathop{\mathbb{E}}_{x\sim\mathcal{U}}[f \cdot \chi_S]| \leq 4l \cdot \epsilon + 2\delta$$

The proof is shown in the appendix for a more general case when $\mathcal{D}$ is a pseudorandom distribution that ϵ fools a class of functions that is closed under multiplication with Fourier basis but here is the brief idea. The class of functions that have small l_1 norm is closed under multiplication with any Fourier basis function χ_S for $S \subseteq [n]$ as shown in Lemma 2.2.8. Therefore, $f_l \cdot \chi_S$ and $f_u \cdot \chi_S$ are also fooled by a small ϵ-biased set. Furthermore, since f_l and f_u are approximators for f, $f_l \cdot \chi_S$ and $f_u \cdot \chi_S$ are approximators for $f \cdot \chi_S$ for $S \subseteq [n]$. Combining Lemma 3.2.4 and Lemma 3.2.5 we get the following theorem.

Theorem 3.2.6. *Given a DNF with* $m = \mathsf{poly}(n)$ *terms* $\phi : \{0,1\}^n \to \{0,1\}$ *and* $\epsilon > 0$, *there exists a deterministic procedure that can approximate* $\hat{\phi}(S)$ *for* $S \subseteq [n]$ *within* 6ϵ *in time* $(\log n)^{O(\log n \log(1/\epsilon))}$

Proof. We get from Lemma 3.2.4, that there exists f_l and f_u such that

- $\mathbb{E}_{x\sim\mathcal{U}}[f_u(x) - \phi_(x)] \leq \epsilon$ and $\mathbb{E}_{x\sim\mathcal{U}}[\phi_(x) - f_l(x)] \leq \epsilon$
- $||f_l||_1, ||f_u||_1 \leq (\log n)^{O(\log n \log(1/\epsilon))}$

Then we can apply Lemma 3.2.5 and see that $\frac{\epsilon}{(\log n)^{O(\log n \log(1/\epsilon))}}$-biased distribution will $4\epsilon + 2\epsilon$ fool $\phi \cdot \chi_S$. The size of the support of the biased distribution is $(\log n)^{O(\log n \log(1/\epsilon))}$ and it can be constructed in time $\mathsf{poly}\left((\log n)^{O(\log n \log(1/\epsilon))}\right)$ by Theorem 2.1.24. □

Chapter 4

Deterministic learning of DNF

In this chapter, we will discuss our main result which uses our deterministic procedure to estimate the Fourier coefficients of an arbitrary DNF with $\mathsf{poly}(n)$ terms and define a sparse polynomial with the estimates of low degree, large coefficients.

4.1 Learning from GMR PRG

Our first result is an $n^{\tilde{O}\left(\log(n)\log^2(1/\epsilon)\right)}$ time deterministic algorithm that finds a function that ϵ-approximates a DNF with polynomial number of terms. We will be using the method we developed in Section 3.1.

Theorem 4.1.1. *(Main Theorem restated) Let f be a boolean function that can be computed by a DNF with polynomial number of terms. Then for $\epsilon > 0$, there exists a deterministic algorithm that runs in time $n^{\tilde{O}\left(\log(n)\log^2(1/\epsilon)\right)}$ and outputs $g : \{0,1\}^n \to \mathbb{R}$ such that g ϵ approximates f.*

Proof. We aim to discard all coefficients of small degree and use the estimates for all of the small degree, large absolute value coefficients. Specifically, let us denote the threshold for the absolute value of a coefficient as θ. We will show a deterministic procedure that outputs a list of $S \subseteq [n]$ that contains every S for which $|\hat{f}(S)| \geq \theta$ and does not contain any S for which $|\hat{f}(S)| \leq \frac{8\theta}{10}$.

Our approximating polynomial g will have zero coefficients for degree larger than $O\left(\log(n)\log(1/\epsilon)\right)$ and this bound comes from Theorem 3.0.1. Let us denote this bound with k.

For each $S \in [n]$ such that $|S| \leq k = O\left(\log(n)log(1/\epsilon)\right)$, we approximate $\mathop{\mathbb{E}}_{x \sim \mathcal{U}}[f(x)\chi_S(x)]$ with error $\pm\frac{\theta}{10}$ where

$$\theta = \frac{\epsilon}{(n)^{O(\log\log(n)\log(1/\epsilon))}}$$

Notice that the denominator of the threshold is the sparsity in Theorem 2.2.21. In order to use the approximation discussed in Section 3.1, we need a PRG that can fool an arbitrary DNF. However, PRG by [GMR13] can be used when we have the parameter for the

maximum width of the DNF being fooled. Fortunately, as we discussed in the preliminary chapter Lemma 2.1.28, if we can find small width DNFs that ϵ-sandwich approximate an arbitrary DNF, we can use the PRG that fools the approximators to also fool the arbitrary DNF. The existence of those approximators is what we showed in Lemma 2.1.29. All we need to do is to tune the error parameter according to the number of coefficients to be estimated and the number of terms in the summation as shown in Lemma 3.1.1.
For each coefficient, there are $2^{O(\log(n)\log(1/\epsilon))} = (n)^{O(\log(1/\epsilon))}$ terms. So the mean of a restricted DNF in each term in the summation should be approximated within $\frac{\epsilon}{(n)^{O(\log\log(n)\log(1/\epsilon))}}$. Let us denote this error as δ'.

$$\delta' = \frac{\epsilon}{(n)^{O(\log\log(n)\log(1/\epsilon))}}$$

Then, by Lemma 2.1.29 there exists $\delta'/2$-sandwiching approximator for a restricted DNF in each term that can be computed by a DNF of width $\log(\frac{2n}{\delta'})$. The PRG that $\delta'/2$ fools any of those sandwich approximators will δ' fool the arbitrary restricted DNF for which we want to approximate the mean.
This PRG, by Theorem 3.1.2, will require seed length

$$\begin{aligned} r &= \tilde{O}\left(\log^2(\frac{2n}{\delta'}) + \log(n/\delta')\log\left(\frac{1}{\delta'}\right) + \log\log(n)\right) \\ &= \tilde{O}\left(\log^2(n)\log\log(n)\log^2(1/\epsilon)\right) \end{aligned}$$

So the time to estimate one coefficient is $2^r = (n)^{\tilde{O}\left(\log(n)\log^2(1/\epsilon)\right)}$. Since there are at most $n^{O(\log(n)\log(1/\epsilon))}$ coefficients to estimate, the total time is $(n)^{\tilde{O}\left(\log(n)\log^2(1/\epsilon)\right)}$.
Let $G = \{S_1, ..., S_l\}$ be the collection of the subsets $S \in [n]$ with $|S| \leq k$ such that the absolute value of the estimate for $\hat{f}(S_i)$ is at least $\frac{9\theta}{10}$. We have the bounds on the sum of absolute values of small degree coefficients from [Tal17] as follows

Lemma 4.1.2. *[Tal17] Let f be a Boolean circuit of depth d and size $m > 1$. Then*

$$\sum_{s:|s|<O\left(\log^{d-1} m\log(1/\epsilon)\right)} |\hat{f}(s)| \leq O\left(\log^{d-1} m\right)^{O\left(\log^{d-1} m\log(1/\epsilon)\right)}$$

In particular, for our case where $m = \mathsf{poly}(n)$ and $d = 2$ above lemma gives us the upper bound of $(n)^{O(\log\log(n)\log(1/\epsilon))}$. Notice that if the absolute value of an approximation to a coefficient is at least $\frac{9\theta}{10}$, the absolute value of the true coefficient must be at least $\frac{8\theta}{10}$. Therefore, G contains every S for which $|\hat{f}(S)| \geq \theta$ and does not contain any S for which $|\hat{f}(S)| \leq \frac{8\theta}{10}$.
Let γ_i denote the estimate within $\pm\frac{\theta}{10}$ of $\hat{f}(S_i)$ such that $S_i \in G$. We then set our approximating polynomial $g : \{0,1\}^n \to \mathbb{R}$ as $g(x) = \sum \gamma_i \chi_{S_i}(x)$.
Then we have the following

$$\begin{aligned}\mathbb{E}[(f(x)-g(x))^2] &= \sum_{s\in[n]} \left(\hat{f}(s)-\hat{g}(s)\right)^2 \\ &= \sum_{|s|>k} \hat{f}^2(s) + \sum_{|s|\leq k \mathrm{and} s\notin G} \hat{f}^2(s) + \sum_{s\in G}\left(\hat{f}(s)-\hat{g}(s)\right)^2\end{aligned}$$

By Lemma 4.1.2, there are at most $\frac{(n)^{O(\log\log(n)\log(1/\epsilon))}}{\frac{8\theta}{10}} = n^{O(\log\log n \log(1/\epsilon))}$ many S such that $|\hat{f}(S)| \geq \frac{8\theta}{10}$. So we can see that the number of non zero coefficients in our approximating polynomial g is small (i.e $|G| \leq n^{O(\log\log n\log(1/\epsilon))}$).
Now we bound each sum in the above equation.
The first summation is bounded above by ϵ due to Theorem 3.0.1.
Since for any $S \notin G$ we have $|\hat{f}(S)| < \theta$, we can bound the second summation as follows

$$\sum_{|s|\leq k \mathrm{and} s\notin G} \hat{f}^2(s) \leq \left(\max_{s\notin G and |s|\leq k} |\hat{f}(S)|\right)\left(\sum_{|S|\leq k} |\hat{f}(S)|\right) \leq \theta\,(n)^{O(\log\log(n)\log(1/\epsilon))} \leq \epsilon.$$

The third summation can be bounded as follows

$$\sum_{s\in G}\left(\hat{f}(s)-\hat{g}(s)\right)^2 \leq (n)^{O(\log\log(n)\log(1/\epsilon))}\,(\theta/10)^2 \leq \epsilon$$

Therefore, g ϵ-approximates f. □

4.2 Learning from biased distribution

Our second result is an $n^{\tilde{O}(\log(n)\log(1/\epsilon))}$ time deterministic algorithm that finds a function that ϵ-approximates a DNF with $\mathsf{poly}(n)$ number of terms. We will use Theorem 3.2.6 in Section 3.2.

Theorem 4.2.1. *Let f be a Boolean function that can be computed by a DNF with polynomial number of terms. Then for $\epsilon > 0$, there exists a deterministic algorithm that runs in time $n^{\tilde{O}(\log(n)\log(1/\epsilon))}$ and outputs $g : \{0,1\}^n \to \mathbb{R}$ such that g ϵ approximates f*

Proof. The proof follows the same procedure as in the proof of Theorem 4.1.1 except that the estimation of the Fourier coefficients of degree less than $k = O(\log n\log(1/\epsilon))$ is performed by enumerating the support of a biased distribution that fools the sandwiching approximators of f.
By Theorem 3.2.6, there exists a deterministic procedure that can approximate a Fourier coefficients of f within $\frac{\theta}{10} = \frac{\epsilon}{n^{O(\log\log n\log(1/\epsilon))}}$ in time

$$n^{O\left(\log\log n\log(n^{\log\log n\log(1/\epsilon)})\right)} = n^{\tilde{O}(\log n\log(1/\epsilon))}$$

Since there are at most $n^{O(\log(n)\log(1/\epsilon))}$ coefficients to estimate, the total time is $n^{\tilde{O}(\log(n)\log(1/\epsilon))}$. We define $g : \{0,1\}^n \to \mathbb{R}$ with the estimates of the Fourier coefficient that are larger than $\frac{9\theta}{10}$ as described in the proof of Theorem 4.1.1. The rest of the proof that shows that g is ϵ close to f in l_2 norm is the same. □

Chapter 5

Deterministic learning of AC^0

In this section, we will compare two approaches to learn AC^0 circuits. An AC^0 circuit consists of AND and OR gates. It has a depth bounded by a constant d and the number of gates is bounded by a polynomial in the input size n. Each gate has unbounded fanin. We also assume without loss of generality that the gates are leveled. That is, all gates on level i have their inputs coming from level $i-1$, all gates at the same level are of the same kind where AND and OR alternate on each level.

5.1 Fourier spectrum bound approach

We will use the fact that for a given AC^0 circuit, all but ϵ of Fourier mass is concentrated on small degree. If we approximate all of the small degree coefficient of the target AC^0 function within a small fraction of ϵ, then we will have an approximating polynomial. This was the approach used by LMN [LMN93]. [LMN93] showed that all but ϵ Fourier mass of an AC^0 circuit of depth d and size m is concentrated on degree smaller then $\log^d m/\epsilon$. They then approximated each of the small coefficient with random sampling. Our approach is the same except that we are using the best Fourier concentration bound due to Tal [Tal17] and we derandomize the estimation of small coefficients. In the case of AC^0 circuits of polynomial size and for $0 < \epsilon < 1$, its Fourier representation is ϵ-concentrated up to degree $k = O(\log^{d-1} n \log(1/\epsilon))$ by Theorem 3.0.1.

For deterministic approximation of Fourier coefficients, we will use the best PRG known for AC^0 due to Servedio and Tan [ST19]. This AC^0 PRG is optimal given the current circuit lower bounds for the class of AC^0 circuits.

Theorem 5.1.1. *[ST19] For every $d \geq 2$, $M \geq n$ and $\epsilon > 0$ there is an ϵ-PRG for the class $\mathcal{C}$ of n-variable size M depth d circuits with seed length $\log^{d+O(1)}(M)\log(1/\epsilon)$.*

Since we want the squared L_2 distance of our polynomial g and the target function f to be less then ϵ, we need to approximate $\mathbb{E}_{x\sim\mathcal{U}}[f(x)\chi_s(x)]$ within $\pm\frac{\epsilon}{n^k}$ for each s such that $|s|$ is at most k. The coefficients of g for degree higher then k will be zero. To do this with

a PRG for AC^0, we need to build an AC^0 circuit that computes $f(x) \cdot \chi_s(x)$.
We need the following fact to build such a circuit.

Lemma 5.1.2. *For some $S \subseteq [n]$, let $\chi_S(x) : \{0,1\}^n \to \{-1,1\}$ be a function defined by*

$$\chi_S(x) = \prod_{i \in S} (-1)^{x_i} = (-1)^{\sum_{i \in S} x_i mod\ 2}$$

Then χ_S can be computed by a depth d circuit of size $\mathsf{poly}\left(|S|\right) 2^{|S|^{1/(d-1)}}$ for every $d \geq 2$

Proof. We need to prove that a parity of n bits can be computed by a circuit of size $\mathsf{poly}\left(n\right) 2^{n^{1/(d-1)}}$. We can show it inductively. For $d = 2$, we already know that any function of n variable can be computed by a DNF or CNF of 2^n terms. So we have the base case. Now assume that the statement is true for d for any $d \geq 2$. We can partition the n bits into smaller bit strings of length $n^{\frac{d-1}{d}}$. So there will be $n^{\frac{1}{d}}$ many partitions. By the inductive assumption, the parity for each partition can be computed by a depth d circuit of size, $\mathsf{poly}(n) 2^{n^{1/d}}$. Those circuits will have either AND or OR on the top of the circuit. Let us assume without loss of generality that it is AND. Then finally, we can compute the parity of $n^{1/d}$ output bits by a DNF of $2^{n^{1/d}}$ terms and we can collapse the two levels of AND into one. Therefore, the overall size of the depth $d+1$ circuit is $\mathsf{poly}(n) 2^{n^{1/d}}$. □

By above lemma, each $\chi_s(x)$ can be computed by a depth d circuit of size at most

$$\mathsf{poly}\left(\log(1/\epsilon) \cdot \log^{d-1} n\right) \cdot 2^{\left(\log(1/\epsilon) \cdot \log^{d-1} n\right)^{1/(d-1)}} = O\left(n^{\log^{1/(d-1)}(1/\epsilon)}\right)$$

Lemma 5.1.3. *If $f, g : \{0,1\}^n \to \{-1,1\}$ can be computed by AC^0 circuits of depth d and size $O\left(t(n)\right)$ for some function t of n, $f \cdot g$ can be computed by an AC^0 circuit of size $O\left(t(n)\right)$ and depth $d+1$.*

Proof. Since f and g are $\{-1,1\}$ valued, $f \cdot g$ corresponds to logical XOR of the two bits. We can assume without loss of generality that the AC^0 circuits computing f and g both have OR as the output gate. Since for two bits X and Y, we have $X \oplus Y = \left(\overline{X} \vee \overline{Y}\right) \wedge (X \vee Y)$, we can merge two levels of OR into one and make depth $d+1$ circuit of size $O\left(h(n)\right)$ computing $f \cdot g$. □

Since f can be computed by a polynomial size depth d circuit and χ_s can be computed by a depth d circuit of size $O\left(n^{\log^{1/(d-1)}(1/\epsilon)}\right)$, $f \cdot \chi_s$ can be computed by a depth $d+1$ circuit of size $O\left(n^{\log^{1/(d-1)}(1/\epsilon)}\right)$. Then using Theorem 5.1.1,we can approximate $f(x) \cdot \chi_s(x)$ within $\pm \frac{\epsilon}{n^{\log^{d-1} n \log(1/\epsilon)}}$ with seed length

$$l = \log^{d+O(1)} \left(n^{\log^{1/(d-1)}(1/\epsilon)}\right) \log\left(\frac{n^{log^{d-1} n \log(1/\epsilon)}}{\epsilon}\right) = \log^{\frac{d}{d-1}} 1/\epsilon \log^{2d+O(1) n}$$

Therefore, the approximation can be done in time

$$2^l = n^{\log^{2d+O(1)} n \log(1/\epsilon)}$$

Since there are at most $n^{\log^{d-1} n \log(1/\epsilon)}$ many coefficients to estimate, the total time to build our approximating polynomial $g : \{0,1\}^n \to \mathbb{R}$ is $n^{\log^{2d+O(1)} n \log(1/\epsilon)}$. Since

$$\begin{aligned}
&\sum_{S \subseteq [n]} \left(\hat{f}(S) - \hat{g}(S)\right)^2 = \sum_{S \subseteq [n]} \left|\hat{f}(S) - \hat{g}(S)\right|^2 \\
&\leq \sum_{S \subseteq [n], |S| \leq O(\log^{d-1} n \log(1/\epsilon))} \left(\frac{\epsilon}{n^{\log^{d-1} n \log(1/\epsilon)}}\right)^2 + \sum_{S \subseteq [n], |S| \geq O(\log^{d-1} n \log(1/\epsilon))} \hat{f}^2(S) \\
&\leq 2\epsilon
\end{aligned}$$

We can see that g 2ϵ approximate f.We then set our hypothesis h as $h = sign(g)$.

Theorem 5.1.4. *For a given function $f : \{0,1\}^n \to \{-1,1\}$ that is computed by a polynomial size AC^0 circuit of depth d and for $0 < \epsilon < 1$, there exists a deterministic learning algorithm that runs in time $n^{\log^{2d+O(1)} n \log(1/\epsilon)}$ and outputs a hypothesis $h : \{0,1\}^n \to \{-1,1\}$ such that $Pr_{x \in \mathcal{U}}[f(x) \neq h(x)] \leq 2\epsilon$*

5.2 Sparse polynomial approach

In this section, we will apply our deterministic approximation of Fourier coefficients developed in Section 3.1 to learn AC^0 circuits of polynomial size. We also use the improved sparsity result from Tal [Tal17]. In LMN, the sparsity of the polynomial that ϵ-approximates an AC^0 circuit is $2^{O(\log n \cdot \log^d(m/\epsilon))}$. The following theorem is a better sparsity result by Tal

Theorem 5.2.1. *[Tal17] Let f be a Boolean circuit of depth d and size $m > 1$. Then f is ϵ-concentrated on at most*

$$O\left(\log^{d-1} m\right)^{O\left(\log^{d-1}(m) \log(1/\epsilon)\right)} = 2^{O\left(\log\log(m) \log^{d-1}(m) \log(1/\epsilon)\right)}$$

Fourier coefficients

Remark 5.2.2. *Note that from Tal's Fourier mass bound in Theorem 3.0.1 we can get the trivial sparsity $n^{O\left(\log^{d-1} m \log(1/\epsilon)\right)}$. For $m = \mathsf{poly}(n)$, Tal's sparsity in above theorem is $n^{O\left((d-1)\log^{d-2} n \log\log n \log(1/\epsilon)\right)}$. Therefore, Tal's sparsity is better than the trivial sparsity by the factor of $\frac{(d-2)\log\log n}{\log n}$ in the exponent.*

Remark 5.2.3. *Using Goldreich and Levin's algorithm to find all coefficients greater than $\frac{\epsilon}{t}$ where t is Tal's sparsity above gives a randomized learning algorithm for AC^0 circuits that runs in time that is polynomial in t. In particular for DNFs, the run time is $n^{O(\log\log n \log(1/\epsilon))}$ which matches [Man92].*

Tal's sparsity result follows from his improved small degree spectral norm bound

Theorem 5.2.4. *[Tal17] Let f be a Boolean circuit of depth d and size $m > 1$. Then*

$$\sum_{s:|s|<O\left(\log^{d-1} m \log(1/\epsilon)\right)} |\hat{f}(s)| \leq O\left(\log^{d-1} m\right)^{O\left(\log^{d-1} m \log(1/\epsilon)\right)}$$

He then shows that the set of subsets of $[n]$ which captures $1-\epsilon$ Fourier mass is

$$\mathcal{F} = \left\{ s : |s| \leq \log^{d-1} m \cdot \log(1/\epsilon) \text{ and } |\hat{f}(s)| \geq \frac{\epsilon}{\log^{d-1} m^{\log^{d-1} m \log(1/\epsilon)}} \right\}$$

By using the same analysis of Mansour's sparsity result for DNFs [Man92] as described in Section 2.2.4, we see that $|\mathcal{F}| \leq t$ and

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

In order to get a deterministic algorithm, we can use the approximation discussed in Section 3.1 except that, for approximation of each term in the summation, we will need a PRG for AC^0. Since the ϵ- concentration degree bound for AC^0 circuit of size $m = \mathsf{poly}(n)$ and depth d is $O\left(\log^{d-1} n \log(1/\epsilon)\right)$ by Theorem 3.0.1, we need to approximate $n^{O\left(log^{d-1} n \log(1/\epsilon)\right)}$ many coefficients. As described in Section 4.1, we can approximate each coefficient within $\pm\frac{\epsilon}{10t}$. For AC^0 in each of $2^{\log^{d-1} m \log(1/\epsilon)}$ many terms in the summation, we can $\frac{\epsilon}{2^{O\left(\log\log(m) \log^{d-1}(m) \log(1/\epsilon)\right)}}$ fool it. With Servedio and Tan's [ST19] PRG, the seed length required for this is

$$\log^{d+O(1)}(m) \log\log m \log^{d-1} m \log(1/\epsilon) = log^{2d+O(1)}(m) \log(1/\epsilon)$$

which will take $2^{log^{2d+O(1)} m \log(1/\epsilon)}$ time. Therefore the total time to estimate all $n^{O\left(log^{d-1} n \log(1/\epsilon)\right)}$ many coefficients is $2^{log^{2d+O(1)} m \log(1/\epsilon)}$. We then define our approximating polynomial g with the approximation of coefficients such that the estimated value is greater than ϵ/t.

Theorem 5.2.5. *There exists a $2^{log^{2d+O(1)} m}$ time deterministic algorithm that learns a function computable by a depth d, polynomial size circuit*

Note that we do not gain in terms of run time by using a sparse polynomial approximating AC^0 in comparison to DNF learning. This is perhaps due to the fact that the PRG for AC^0 is not as good as PRG for DNFs.

Chapter 6

Deterministic Goldreich and Levin algorithm for DNFs

Recall that the crucial step in Goldreich and Levin algorithm is to approximate for any $1 \leq k < n$ the sum over all $\beta \in \{0,1\}^{n-k}$ of squares of Fourier coefficients of the form $\hat{f}(\alpha\beta)$ for a fixed $\alpha \in \{0,1\}^k$. Also recall that DNFs have most of its Fourier mass on degree less than $\log(n)\log(1/\epsilon)$ by Theorem 3.0.1. Because of this concentration of Fourier mass, we only need to find large Fourier coefficients of low degree to learn a DNF.

Given a fixed $\alpha \in \{0,1\}^k$, let us denote the set of i such that $\alpha_i = 1$ with α_1. In other words, $\alpha_1 = \{i : \alpha_i = 1\}$ and $\alpha_0 = \{0,1\}^k/\alpha_1$. Then on the top of the original recurs condition, that is $\sum_{\beta\in\{0,1\}^{n-k}} \hat{f}(\alpha\beta) \geq \theta^2$, we have another condition which is $\alpha_1 \leq \log(n)\log(1/\epsilon)$. In any case, the number of recursive calls is at most $\frac{n}{\theta^2}$. Therefore, in order to efficiently derandomize the Goldreich and Levin algorithm, we only need to derandomize the estimation of $\mathbb{E}_{x\in\{0,1\}^{n-k}}[f_\alpha^2(x)] = \mathbb{E}_{x\in\{0,1\}^{n-k}}\left[\left(\mathbb{E}_{y\in\{0,1\}^k}[f_x(y)\chi_\alpha(y)]\right)^2\right]$ efficiently.

Before we show that this can be done, let us first prove a lemma that will be used in our proof of derandomization of Goldreich and Levin algorithm for the special case of small degree coefficients and a good pseudo random generator.

Lemma 6.0.1. *Assume that we have an ϵ-pseudorandom generator G of seed length $O(\log(n/\epsilon))$ for DNFs with $\mathsf{poly}(n)$ number of terms. Let us denote the distribution that is generated by G with $\mathcal{D}$. Then for given DNFs $\phi : \{0,1\} \to \{0,1\}$ of size m_1 and $\psi : \{0,1\} \to \{0,1\}$ of size m_2, we can approximate $\Pr_{x\in\{0,1\}^n}[\phi(x) \oplus \psi(x) = 0]$ within 2ϵ in time $\mathsf{poly}(n/\epsilon)$.*

Proof. We have that

$$\phi(x) \oplus \psi(x) = 0 \iff (\phi(x) \wedge \psi(x)) \vee \left(\overline{\phi(x)} \wedge \overline{\psi(x)}\right)$$

Let us denote $S = \{x : \phi(x) \oplus \psi(x) = 0\}$. Note that $(\phi(x) \wedge \psi(x))$ is a DNF of size $m_1 m_2$. Let us denote this DNF with ξ. We can also observe the following

$$\left(\overline{\phi(x)} \wedge \overline{\psi(x)}\right) = \overline{\phi(x) \vee \psi(x)}$$

Let us denote the DNF $\phi(x) \vee \psi(x)$ with η. Let us denote the set of x that satisfy ξ with A and the set of x that satisfy η with B. That is,

$$A = \{x : \xi(x) = 1\}, B = \{x : \eta(x) = 1\}$$

Then we see that A and $\overline{B}$ are disjoint. Since $S = A \cup \overline{B}$ we have that

$$|S| = |A| + |\overline{B}| = |A| + |U| - |B|$$

where U denotes the set of all 2^n assignments. Therefore we have the following

$$\begin{aligned}
\Pr_{x \in \{0,1\}^n}[\phi(x) \oplus \psi(x) = 0] = \frac{|S|}{|U|} &= \frac{|A| + |U| - |B|}{|U|} \\
&= \frac{A}{U} + 1 - \frac{B}{U} \\
&= \Pr_{x \in \{0,1\}^n}[\xi(x) = 1] + 1 - \Pr_{x \in \{0,1\}^n}[\eta(x) = 1] \\
&= \mathbb{E}_{x \in \{0,1\}^n}[\xi(x)] + 1 - \mathbb{E}_{x \in \{0,1\}^n}[\eta(x)] \\
&= \mathbb{E}_{x \sim \mathcal{D}}[\xi(x)] + 1 - \mathbb{E}_{x \sim \mathcal{D}}[\eta(x)] \pm 2\epsilon
\end{aligned}$$

Time to construct the distribution $\mathcal{D}$ is $\mathsf{poly}(n/\epsilon)$ by our assumption. □

Theorem 6.0.2. *Given a DNF with* $\mathsf{poly}(n)$ *terms* $f : \{0,1\}^n \to \{-1,1\}$, *a fixed* $\alpha \in \{0,1\}^k$ *such that* $|\alpha_1| \leq O(\log(n)\log(1/\delta))$ *for* $\delta > 0$, *there exists a deterministic algorithm that runs in time* $\mathsf{poly}(n/\epsilon) + (n)^{O(\log(1/\delta))}$ *for* $\epsilon > 0$ *to approximate*

$$\mathbb{E}_{x \in \{0,1\}^{n-k}}\left[\left(\mathbb{E}_{y \in \{0,1\}^k}[f_x(y)\chi_\alpha(y)]\right)^2\right]$$

within 4ϵ *under the assumption that an* ϵ*-pseudorandom generator for DNFs of seed length* $\log(n/\epsilon)$ *exists.*

Proof. For a given $y \in \{0,1\}^k$, let us denote with y_1 for the bit string that corresponds to the coordinates of the set α_1. Likewise, let y_0 be the bit string corresponding to α_0. Let G be a pseudo random generator with seed length $\log(n/\epsilon)$ and let $\mathcal{D}$ be the distribution over $\{0,1\}^n$ that is generated by G.

Note that for a DNF $f : \{0,1\}^n \to \{0,1\}$, the marginal distribution denoted by $\mathcal{D}^J$ of the distribution $\mathcal{D}$ on the bit strings corresponding to any subset $J \subseteq [n]$ still fools f that is restricted on the coordinates $\bar{J} = [n]/J$ That is, for a fixed $z \in \{0,1\}^{\bar{J}}$

$$\begin{aligned}
\left| \mathop{\mathbb{E}}_{x \sim \{0,1\}^n} [f_z(x)] - \mathop{\mathbb{E}}_{x \sim \mathcal{D}} [f_z(x)] \right| &= \left| \mathop{\mathbb{E}}_{w \in \{0,1\}^J} \left[\mathop{\mathbb{E}}_{u \in \{0,1\}^{\bar{J}}} [f_z(wu)] \right] - \mathop{\mathbb{E}}_{w \sim \mathcal{D}^J} \left[\mathop{\mathbb{E}}_{u \sim \mathcal{D}^{\bar{J}}} [f_z(wu)] \right] \right| \\
&= \left| \mathop{\mathbb{E}}_{w \in \{0,1\}^J} [f_z(wu)] - \mathop{\mathbb{E}}_{w \sim \mathcal{D}^J} [f_z(wu)] \right| \\
&= \left| \mathop{\mathbb{E}}_{w \in \{0,1\}^J} [f_z(w)] - \mathop{\mathbb{E}}_{w \sim \mathcal{D}^J} [f_z(w)] \right| \\
&\leq \epsilon \qquad \text{(since } f_z \text{ is a DNF thus } \mathcal{D} \text{ } \epsilon\text{-fools } f_z)
\end{aligned}$$

Let us denote the marginal distribution of $\mathcal{D}$ on the bit strings corresponding to the coordinates of α_0 with $\mathcal{D}^{\alpha_0}$. Then we have

$$\begin{aligned}
f_\alpha(x) &= \mathop{\mathbb{E}}_{y \in \{0,1\}^k} [f_x(y) \chi_\alpha(y)] \\
&= \mathop{\mathbb{E}}_{y_0} \left[\mathop{\mathbb{E}}_{y_1} [f_x(y) \chi_\alpha(y)] \right] \\
&= \mathop{\mathbb{E}}_{y_1} \left[\chi_{\alpha_1}(y_1) \cdot \mathop{\mathbb{E}}_{y_0} [f_{xy_1}(y_0)] \right] \\
&= \mathop{\mathbb{E}}_{y_1} \left[\chi_{\alpha_1}(y_1) \cdot \left(\mathop{\mathbb{E}}_{y_0 \sim \mathcal{D}^{\alpha_0}} [f_{xy_1}(y_0)] \pm \epsilon \right) \right] \qquad \text{(because } \mathcal{D}^{\alpha_0} \text{ } \epsilon\text{-fools } f_{xy_1}) \\
&= \mathop{\mathbb{E}}_{y_1} \left[\chi_{\alpha_1}(y_1) \cdot \left(\mathop{\mathbb{E}}_{y_0 \sim \mathcal{D}^{\alpha_0}} [f_{xy_1}(y_0)] \right) \right] \qquad \text{(because } \mathbb{E}_{y_1} [\chi_{\alpha_1}(y_1) \cdot \epsilon] = 0)
\end{aligned}$$

Now,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x \in \{0,1\}^k} \left[f_\alpha^2(x) \right] \\
&= \mathop{\mathbb{E}}_{x \in \{0,1\}^k} \left[\left(\mathop{\mathbb{E}}_{y_1} \left[\chi_{\alpha_1}(y_1) \cdot \mathop{\mathbb{E}}_{y_0 \sim \mathcal{D}^{\alpha_0}} [f_{xy_1}(y_0)] \right] \right)^2 \right] \\
&= \mathop{\mathbb{E}}_{x \in \{0,1\}^k} \left[\mathop{\mathbb{E}}_{y_1 \in \{0,1\}^{|\alpha_1|}} \left[\mathop{\mathbb{E}}_{z_1 \in \{0,1\}^{|\alpha_1|}} [\chi_{\alpha_1}(y_1) \chi_{\alpha_1}(z_1)] \right] \mathop{\mathbb{E}}_{y_0 \sim \mathcal{D}^{\alpha_0}} \left[\mathop{\mathbb{E}}_{z_0 \sim \mathcal{D}^{\alpha_0}} [f_{xy_1}(y_0) f_{xz_1}(z_0)] \right] \right] \\
&= \mathop{\mathbb{E}}_{y_1 \in \{0,1\}^{|\alpha_1|}, z_1 \in \{0,1\}^{|\alpha_1|}} [\chi_{\alpha_1}(y_1) \chi_{\alpha_1}(z_1)] \mathop{\mathbb{E}}_{y_0 \sim \mathcal{D}^{\alpha_0}, z_0 \sim \mathcal{D}^{\alpha_0}} \left[\mathop{\mathbb{E}}_{x \in \{0,1\}^k} [f_{y_0 y_1}(x) f_{z_0 z_1}(x)] \right]
\end{aligned}$$

Since $|\alpha_1| \leq O(\log(n) \log(1/\delta))$, calculating $\mathbb{E}_{y_1 \in \{0,1\}^{|\alpha_1|}, z_1 \in \{0,1\}^{|\alpha_1|}} [\chi_{\alpha_1}(y_1) \chi_{\alpha_1}(z_1)]$ takes

$$2^{2O(\log(n) \log(1/\delta))} = n^{O(2 \log(1/\delta))}$$

time.

Furthermore, the size of the support of the distribution $\mathcal{D}$ is at most $\mathsf{poly}(n/\epsilon)$ therefore if we can estimate $\mathbb{E}_{x \in \{0,1\}^k} [f_{y_0 y_1}(x) f_{z_0 z_1}(x)]$ for each fixed y_0, y_1, z_0, z_1 efficiently, then we will approximate

$$\underset{y_0 \sim \mathcal{D}^{\alpha_0}, z_0 \sim \mathcal{D}^{\alpha_0}}{\mathbb{E}} \left[\underset{x \in \{0,1\}^k}{\mathbb{E}} \left[f_{y_0 y_1}(x) f_{z_0 z_1}(x) \right] \right]$$

efficiently by enumerating each of $y_0 \sim \mathcal{D}^{\alpha_0}, z_0 \sim \mathcal{D}^{\alpha_0}$.

Let us denote with $\phi : \{0,1\}^k \to \{0,1\}$ for $f_{y_0 y_1}$ and $\psi : \{0,1\}^k \to \{0,1\}$ for $f_{z_0 z_1}$. Note that we only changed the encoding of True and False for ϕ and ψ. So we see that ϕ and ψ are both computed by DNFs. Since for $f_{y_0 y_1}$ and $f_{z_0 z_1}$, True and False are encoded with $(-1)^1 = -1$ and $(-1)^0 = 1$, we see that the product $f_{y_0 y_1} \cdot f_{z_0 z_1}$ corresponds to XOR of two bits ϕ and ψ. Therefore, we get

$$\begin{aligned} \underset{x \in \{0,1\}^k}{\mathbb{E}} \left[f_{y_0 y_1}(x) f_{z_0 z_1}(x) \right] &= Pr_{x \in \{0,1\}^k} \left[\phi(x) = \psi(x) \right] - Pr_{x \in \{0,1\}^k} \left[\phi(x) \neq \psi(x) \right] \\ &= Pr_{x \in \{0,1\}^k} \left[\phi(x) \oplus \psi(x) = 0 \right] - Pr_{x \in \{0,1\}^k} \left[\phi(x) \oplus \psi(x) = 1 \right] \\ &= 2 Pr_{x \in \{0,1\}^k} \left[\phi(x) \oplus \psi(x) = 0 \right] - 1 \end{aligned}$$

By Lemma 6.0.1, we see that

$$\begin{aligned} \underset{x \in \{0,1\}^k}{\mathbb{E}} \left[f_{y_0 y_1}(x) f_{z_0 z_1}(x) \right] &= 2 \left(\underset{x \sim \mathcal{D}}{\mathbb{E}} [\xi(x)] + 1 - \underset{x \sim \mathcal{D}}{\mathbb{E}} [\eta(x)] \pm 2\epsilon \right) - 1 \\ &= 2 \left(\underset{x \sim \mathcal{D}}{\mathbb{E}} [\xi(x)] + \underset{x \sim \mathcal{D}}{\mathbb{E}} [\eta(x)] \right) \pm 4\epsilon + 1 \end{aligned}$$

where η and ξ are defined as in the proof of Lemma 6.0.1. This estimation is done in $\mathsf{poly}(n/\epsilon)$. We need to do this estimation for each of $y_0 \sim \mathcal{D}^{\alpha_0}, z_0 \sim \mathcal{D}^{\alpha_0}$ which is at most $\mathsf{poly}(n/\epsilon)$ times. Therefore, we can 4ϵ-estimate $\mathbb{E}_{x \in \{0,1\}^k} \left[f_\alpha^2(x) \right]$ in time

$$n^{O(\log(1/\delta))} + \mathsf{poly}(n/\epsilon)$$

□

Now we will use the above deterministic version of Goldreich and Levin algorithm to learn a DNF with $\mathsf{poly}(n)$ number of terms.

Theorem 6.0.3. *Assume that an ϵ-pseudorandom generator for DNFs of seed length* $\log(n/\epsilon)$ *exists. Then given a DNF with* $\mathsf{poly}(n)$ *terms $f : \{0,1\}^n \to \{-1,1\}$, and for $\delta > 0$, there exists a deterministic algorithm that outputs a sparse polynomial that δ-approximates f in L_2 norm in time $\frac{n^{O(\log \log n \log(1/\delta))}}{\mathsf{poly}(\delta)}$.*

Proof. Recall that the threshold we want for the small degree Fourier coefficients is the total error of the sparse polynomial δ-approximating the target DNF divided by the spectral norm of the DNF up to δ-concentration bound degree for some $\delta > 0$. That is we set

$$\theta = \frac{\delta}{n^{O(\log \log n \log(1/\delta))}}$$

So we need to approximate $\mathbb{E}_{x\in\{0,1\}^k}\left[f_\alpha^2(x)\right]$ within, say, $\theta/4$. By substituting $\epsilon = \theta/4$ in the above theorem, this approximation takes

$$n^{O(\log(1/\delta))} + \mathsf{poly}\left(\frac{n^{O(\log\log n \log(1/\delta))}}{\delta}\right) = \frac{n^{O(\log\log n \log(1/\delta))}}{\mathsf{poly}(\delta)}$$

□

For constant δ, we get $n^{O(\log\log n)}$ time which matches Mansour's time [Man92].

Chapter 7

Conclusions

We have shown that there exists a deterministic algorithm to learn a DNFs that produces a sparse polynomial approximating the target DNF. In the case of DNFs,we were able to show that there is an advantage in using the current best pseudo random generator [GMR13] to estimate the small degree coefficients. That is, we were able to get a better time than the randomized algorithm of [LMN93]. However, we did not get an improvement over [LMN93] in the run time when we used the same approach with constant depth circuits (AC0) with the best PRG for AC0 circuits [ST19]. It will be interesting to research why the current state of AC0 PRG is not as good as PRG for DNFs. It can be in two directions. One is to see if there is a lower bound in the number of random bits that is required to generate a pseudorandom distribution for AC0. The other will be to improve the bound and get a better PRG for AC0.

We have also covered in great details how for small L_1 norm functions, Goldreich and Levin algorithm can be derandomized. We then derandomized the algorithm in a different condition where we are only looking for small degree Fourier coefficients of a DNF assuming that we have an ideal PRG for DNFs. Our original goal was to generalize the derandomization to the case when we don't know if a function has a small l_1 norm but know that the function is sandwiched or just approximated by another function that has a small l_1 norm. If we know that there exists a collection U of sets $S \subseteq [n]$ such that U contains all S such that $|\hat{f}(S)| \geq \theta$ and $\sum_{T \notin U} \hat{f}^2(T) \leq \epsilon$ then we can use the Goldreich and Levin algorithm to find such collection U. So the existence of the collection U is a sufficient condition for the hypothesis formed from the output of Goldreich and Levin algorithm is a good hypothesis. Conversely, if the hypothesis formed from the Goldreich and Levin algorithm is a good one, then there exists the collection U which is just the output of the algorithm.

The problem is that we need to let the algorithm know the threshold. In the case of small l_1 norm Boolean function f, [KM93] showed that the threshold for the collection U is $\epsilon/||f||_1$. It will be interesting to see if such a collection with a threshold exists for a function that is approximated by a small l_1 norm function.

We do not have L1 bound for general DNF and we are limited to knowing the bound on the sum of absolute values of small degree Fourier coefficients. The bound given by Mansour [Man92] is nearly tight for bounded width DNFs.

Since the fastest randomized algorithm under the uniform distribution with membership query is in polynomial time [Jac97], it would be interesting to consider the problem whether this randomized method can be derandomized. Toward this direction, it would be interesting to look into the connection between the Boosting and the hardcore set construction [Imp95] such as the result of [KS03]. The randomized process in Jackson's algorithm is Boosting where the examples are randomly output according to the updated distribution in each round. It would be interesting to see if we can derandomize this sampling process, just as we used a pseudo random distribution to mimic sampling from the uniform distribution.

Bibliography

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992.

[Ang87] Dana Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, 1987.

[Baz09] Louay M. J. Bazzi. Polylogarithmic independence can fool DNF formulas. *SIAM J. Comput.*, 38(6):2220–2272, 2009.

[BMOS05] Nader H. Bshouty, Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning DNF from random walks. *J. Comput. Syst. Sci.*, 71(3):250–265, 2005.

[DETT10] Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In Maria J. Serna, Ronen Shaltiel, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, volume 6302 of *Lecture Notes in Computer Science*, pages 504–517. Springer, 2010.

[Fel12] Vitaly Feldman. Learning DNF expressions from fourier spectrum. In Shie Mannor, Nathan Srebro, and Robert C. Williamson, editors, *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, volume 23 of *JMLR Proceedings*, pages 17.1–17.19. JMLR.org, 2012.

[Fre95] Yoav Freund. Boosting a weak learning algorithm by majority. *Inf. Comput.*, 121(2):256–285, 1995.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32. ACM, 1989.

[GMR$^+$12] Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 120–129. IEEE Computer Society, 2012.

[GMR13] Parikshit Gopalan, Raghu Meka, and Omer Reingold. DNF sparsification and a faster deterministic counting algorithm. *Computational Complexity*, 22(2):275–310, 2013.

[Imp95] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 538–545. IEEE Computer Society, 1995.

[Jac97] Jeffrey C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *J. Comput. Syst. Sci.*, 55(3):414–440, 1997.

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM J. Comput.*, 22(6):1331–1348, 1993.

[KS03] Adam R. Klivans and Rocco A. Servedio. Boosting and hard-core set construction. *Mach. Learn.*, 51(3):217–238, 2003.

[KS04] Adam R. Klivans and Rocco A. Servedio. Learning DNF in time $2^{\tilde{\mathrm{o}}(\mathrm{n}^{1/3})}$. *J. Comput. Syst. Sci.*, 68(2):303–318, 2004.

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.

[Man92] Yishay Mansour. An $O(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution. In David Haussler, editor, *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992*, pages 53–61. ACM, 1992.

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions.* Cambridge University Press, 2014.

[Raz09] Alexander A. Razborov. A simple proof of bazzi's theorem. *ACM Trans. Comput. Theory*, 1(1):3:1–3:5, 2009.

[Sit95] Meera Sitharam. Pseudorandom generators and learning algorithms for acˆ0. *Comput. Complex.*, 5(3/4):248–266, 1995.

[SS97] Meera Sitharam and Timothy Straney. Deranomized learning of boolean functions. In Ming Li and Akira Maruoka, editors, *Algorithmic Learning Theory, 8th International Conference, ALT '97, Sendai, Japan, October 6-8, 1997, Proceedings*, volume 1316 of *Lecture Notes in Computer Science*, pages 100–115. Springer, 1997.

[ST19] Rocco A. Servedio and Li-Yang Tan. Improved pseudorandom generators from pseudorandom multi-switching lemmas. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, volume 145 of *LIPIcs*, pages 45:1–45:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Tal17] Avishay Tal. Tight bounds on the Fourier spectrum of AC^0. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 15:1–15:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[Val84] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[Ver90] Karsten A. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In Mark A. Fulk and John Case, editors, *Proceedings of the Third Annual Workshop on Computational Learning Theory, COLT 1990, University of Rochester, Rochester, NY, USA, August 6-8, 1990*, pages 314–326. Morgan Kaufmann, 1990.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982.

Appendix A

Estimating Fourier coefficients

First, observe that the sandwiching approximation works also for $\{-1,1\}$-valued functions, with the same proof. Let D be a pseudorandom distribution that ϵ-fools the class of functions C. Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be any function that is ϵ-sandwiched by f_l (from below) and f_u (from above), for some functions $f_l, f_u \in C$. Then we get that D (2ϵ)-fools f.

Let $\chi = \chi_S = \prod_{i\in S} x_i$ be an arbitrary n-variate Fourier basis function, for some $S \subseteq [n]$. Suppose that $f_u \cdot \chi_S \in C$ for every set $S \subseteq [n]$ (i.e., C is closed under multiplication with any Fourier basis function) and so D ϵ-fools also $f_u \cdot \chi_S$, for every $S \subseteq [n]$. We show that then D also (6ϵ)-fools $f \cdot \chi_S$, for every S, i.e., D can be used to approximately compute each Fourier coefficient $\hat{f}(S)$.

Lemma A.0.1. *Under the assumptions on D, f, f_u, f_l, and χ above, we get that*

$$\left| \mathop{\mathbb{E}}_{D}[\hat{f}(S)] - \mathop{\mathbb{E}}_{U}[\hat{f}(S)] \right| \leq 6\epsilon.$$

Proof. We have that $\hat{f}(S) = \mathbb{E}_U[f \cdot \chi]$, where $\chi = \chi_S$ is the Fourier basis function for the set S. We have

$$\begin{aligned}
\mathop{\mathbb{E}}_{D}[f\cdot\chi] &= \mathop{\mathbb{E}}_{D}[(f - f_u + f_u)\cdot\chi] \\
&= \mathop{\mathbb{E}}_{D}[(f - f_u)\cdot\chi] + \mathop{\mathbb{E}}_{D}[f_u\cdot\chi] \\
&= \mathop{\mathbb{E}}_{U}[f_u\cdot\chi] \pm (\epsilon + \mathop{\mathbb{E}}_{D}[(f_u - f)]) && \text{(because } D\ \epsilon\text{-fools } f_u\cdot\chi) \\
&= \mathop{\mathbb{E}}_{U}[f_u\cdot\chi] \pm (4\epsilon + \mathop{\mathbb{E}}_{U}[(f_u - f)]) && \text{(because } D\ (3\epsilon)\text{-fools } (f_u - f)) \\
&= \mathop{\mathbb{E}}_{U}[f_u\cdot\chi] \pm 5\epsilon && \text{(because } f_u\ \epsilon\text{-approximates } f) \\
&= \mathop{\mathbb{E}}_{U}[(f_u - f + f)\cdot\chi] \pm 5\epsilon \\
&= \mathop{\mathbb{E}}_{U}[(f_u - f)\cdot\chi] + \mathop{\mathbb{E}}_{U}[f\cdot\chi] \pm 5\epsilon \\
&= \mathop{\mathbb{E}}_{U}[f\cdot\chi] \pm 6\epsilon && \text{(because } f_u\ \epsilon\text{-approximates } f)
\end{aligned}$$

□