# Learning, Cryptography, and the Average Case

**Andrew Wan**

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

**COLUMBIA UNIVERSITY**

2010

# ABSTRACT

## Learning, Cryptography, and the Average Case

## Andrew Wan

This thesis explores problems in computational learning theory from an *average-case* perspective. Through this perspective we obtain a variety of new results for learning theory and cryptography.

Several major open questions in computational learning theory revolve around the problem of efficiently learning polynomial-size DNF formulas, which dates back to Valiant's introduction of the PAC learning model [Valiant, 1984]. We apply an average-case analysis to make progress on this problem in two ways.

- *We prove that Mansour's conjecture is true for random DNF.* In 1994, Y. Mansour conjectured that for every DNF formula on $n$ variables with $t$ terms there exists a polynomial $p$ with $t^{O(\log(1/\epsilon))}$ non-zero coefficients such that $\mathbf{E}_{x\in\{0,1\}^n}[(p(x)-f(x))^2] \leq \epsilon$. We make the first progress on this conjecture and show that it is true for several natural subclasses of DNF formulas including randomly chosen DNF formulas and read-$k$ DNF formulas. Our result yields the first polynomial-time query algorithm for agnostically learning these subclasses of DNF formulas with respect to the uniform distribution on $\{0,1\}^n$ (for any constant error parameter and constant $k$). Applying recent work on sandwiching polynomials, our results imply that $t^{-O(\log 1/\epsilon)}$-biased distributions fool the above subclasses of DNF formulas. This gives pseudorandom generators for these subclasses with shorter seed length than all previous work.

- *We give an efficient algortihm that learns random monotone DNF.* The problem of efficiently learning the *monotone* subclass of polynomial-size DNF formulas from random examples was also posed in [Valiant, 1984]. This notoriously difficult question is still open, despite much study and the fact that known impediments to learning the

non-monotone class (cf. [Blum *et al.*, 1994; Blum, 2003a]) do not exist for monotone DNF formulas. We give the first algorithm that learns randomly chosen monotone DNF formulas of arbitrary polynomial size, improving results which efficiently learn $n^{2-\epsilon}$-size random monotone DNF formulas [Jackson and Servedio, 2005b]. Our main structural result is that most monotone DNF formulas reveal their term structure in their constant-degree Fourier coefficients.

In this thesis, we also see that connections between learning and cryptography are naturally made through average-case analysis. First, by applying techniques from average-case complexity, we demonstrate new ways of using cryptographic assumptions to prove *limitations* on learning. As counterpoint, we also exploit the average-case connection in the service of cryptography. Below is a more detailed description of these contributions.

- *We show that monotone polynomial-sized circuits are hard to learn if one-way functions exist.* We establish the first cryptographic hardness results for learning polynomial-size classes of monotone circuits, giving a computational analogue of the information-theoretic hardness results of [Blum *et al.*, 1998]. Some of our results show the cryptographic hardness of learning polynomial-size monotone circuits to accuracy only slightly greater than $1/2 + 1/\sqrt{n}$; this is close to the optimal accuracy bound, by positive results of Blum, *et al.* Our main tool is a complexity-theoretic approach to hardness amplification via noise sensitivity of monotone functions that was pioneered by O'Donnell [O'Donnell, 2004a].

- *Learning an overcomplete basis: analysis of lattice-based signatures with perturbations.* Lattice-based cryptographic constructions are desirable not only because they provide security based on worst-case hardness assumptions, but also because they can be extremely efficient and practical. We propose a general technique for recovering parts of the secret key in lattice-based signature schemes that follow the Goldreich-Goldwasser-Halevi (GGH) and NTRUSign design with *perturbations*. Our technique is based on solving a learning problem in the average-case. To solve the average-case problem, we propose a special-purpose optimization algorithm based on higher-order *cumulants* of the signature distribution, and give theoretical and experimental evidence of

its efficacy. Our results suggest (but do not conclusively prove) that NTRUSign is vulnerable to a polynomial-time attack.

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgments

Were it not for the generosity and patience of my advisors Tal Malkin and Rocco Servedio, I could not have written this thesis. Tal and Rocco created a familial, supportive, and fun atmosphere at school. They were always available to give guidance and advice, and I asked for it often. I hate to slander their advisorial capacities, but they've made me into the researcher that I am. This may seem like such a modest achievement that to credit them with it borders on insult, but considering the condition they found me in, I think it is pretty remarkable.

Thanks to Tal for her constant encouragement and positive attitude. I'm grateful that she was willing to point out areas that I needed to improve on. She found many opportunities for me and took initiative in matters that I was sometimes inclined (wrongly) to neglect.

Working with Rocco has been a truly humbling experience. It's taken some time for me to fully apprehend his technical facility, which he wields with nunlike modesty. I thank him for being so gentle and patient in our work together. I would also like to thank him for pointing me to *The Wind-up Bird Chronicle*. Rocco has referred me to many sources—all of them have turned out to be uncannily relevant.

Of my classmates and co-authors, I wish to give special thanks to Homin Lee. We've had the kind of collaboration that I've come to realize is extremely rare, one that can begin with barely formed ideas and intuitions. Research without him would not be nearly as fun or satisfying.

After most of my friends left Columbia last year, I thought that I would be less motivated to make the commute to campus. Li-Yang Tan provided a great reason to keep coming — I thank him for many entertaining moments. I wish him the best in his remaining years in graduate school.

Thanks to my co-authors — Dana Dachman-Soled Jeff Jackson, Adam Klivans, Tal

Malkin, Chris Peikert, Rocco Servedio, and Hoeteck Wee — for their contributions to this thesis. Thanks to the members of my thesis committee, Tal Malkin, Ryan O'donnell, Chris Peikert, Rocco Servedio, and Mihalis Yannakakis, for their insightful comments and questions.

Thanks to my colleagues and friends — Adi Akavia, Spyridon Antonakopoulos, Andrej Bogdanov, Ilias Diakonikolas, Seung Geol Choi, Dana Dachman-Soled, Ariels Elbaz and Gabizon, Moritz Hardt, Ragesh Jaiswal, Troy Lee, Kevin Matulef, Emanuele Viola, and Hoeteck Wee — for helpful, entertaining, and interesting conversations.

Thanks to my co-authors Andrej Bogdanov and Kunal Talwar on work that, while not appearing in this thesis, was one of my favorite projects.

I'd also like to thank my non-technical friends — Karen Emmerich, Emily Ford, Muzamil Huq, Carey Kasten, Gena Konstantinakos, Emily Kramer, Suzanne LiPuma, Jonathan Rick, Casey Shoop, Erica Siegel, Aron Wahl — for giving me the company and support that kept me sane these past six years.

Thanks to my Aunties Mimi, Sophie, Ratana, Jean, Judy, Felly, Letty, Jeannie, Tess, Francesca, Estela, Anna, Bonnie, Linda, Linda, Cecilia, Mazy and Lucy, my Uncles Raymond, Walter, John, Stephen, David, Vinzon, Shams, Joe, Larry, Dominic, Kin-Wah, K.K., Kenneth, Yau Shun-Chiu, Charlie, and Prudencio, my pediatrician Dr. Alice Goldin, Dr. Richard Goldin, Dr. James London, Dr. Jim Hedges, Trish London, Nancy Hedges, and Al and Joan Rosenstein.

Special thanks to my dog, George, for always listening, and to the mother of my dog, Jesse, for her continued love and support.

Finally, I thank my parents, Lawrence and Lourdes Wan, for everything they have taught me and for their constant, unquestioning love.

**Specific contributions**:

**Chapter 3**: Thanks to Sasha Sherstov for important contributions at an early stage of this work. Thanks to Omid Etesami for pointing out some errors in a previous version of this chapter, including a crucial flaw in the proof of the read-$k$ case. We also thank him for pointing out to us that Lemma 9, which was originally for monotone read-$k$ DNF formulas, holds for the non-monotone case (through Lemma 8). I would also like to thank him for

pointing out to us that our proof for the monotone read-$k$ case extends to the non-monotone case (through Lemma 8). Lemma 8 is due to Omid and James Cook. Thanks to Li-Yang Tan for the lovely graphs.

This thesis is dedicated to my parents, Lawrence and Lourdes Wan, and to the memory of my aunt, Doreen Chun-Yee Yau.

**Bibliographic note**

Some of the work in thesis has already appeared elsewhere in some form, and all of it involves the work of other researchers.

- Chapter 3 is based on the paper "Mansour's Conjecture is True for Random DNF" [Klivans *et al.*, 2010] and is joint work with Adam Klivans and Homin Lee. The result for read-$k$ DNF formulas is due in part to Omid Etesami, who pointed out a crucial error in a previous version and gave us a reduction from the non-monotone to the monotone case. The reduction is stated as Lemma 8 of Chapter 3 and was proven by Omid and James Cook.

- Chapter 4 is based on the paper "Learning Random Monotone DNF" [Jackson *et al.*, 2008a] and is joint work with Jeff Jackson, Homin Lee, and Rocco Servedio.

- Chapter 5 is based on the paper "Optimal Cryptographic Hardness of Learning Monotone Functions" [Dachman-Soled *et al.*, 2009] (initially appearing in [Dachman-Soled *et al.*, 2008]) and is joint work with Dana Dachman-Soled, Homin Lee, Tal Malkin, Rocco Servedio, and Hoeteck Wee.

- Chapter 6 is joint work with Chris Peikert, Tal Malkin, and Rocco Servedio.

# Chapter 1

# Introduction

## 1.1 Motivation

**Average-Case Complexity** Traditionally, a problem in computer science is considered tractable if there is an efficient algorithm which solves *any* instance of the problem. But, in practice, an algorithm may only see instances of a particular form or drawn according to a particular distribution. From the algorithm designer's perspective, this fact can make a difficult problem solvable. On the other hand, if a problem's hardness is the useful property, such as in cryptography, one requires the ability to find hard instances in order to utilize the problem's hardness. Thus, there are compelling reasons to study the performance of algorithms on instances drawn from fixed distributions, i.e. the performance of algorithms on the *average*.

The average-case has been well-studied in complexity theory, ranging from the foundational work on NP problems initiated by Levin [Levin, 1986; Impagliazzo and Levin, 1990; Ben-David and Chor, 1992], to the work on problems in EXP [Babai *et al.*, 1993; Impagliazzo, 1995; Impagliazzo and Wigderson, 1997] which show connections to worst-case hardness. Although amplification of average-case hardness in NP is a fruitful area of research [O'Donnell, 2004b; Trevisan, 2005; Healy *et al.*, 2006], we do not expect to connect this hardness to worst-case NP-hardness using current techniques [Akavia *et al.*, 2006; Bogdanov and Trevisan, 2003; Feigenbaum and Fortnow, 1993]. See the survey by Bogdanov and Trevisan [Bogdanov and Trevisan, 2006] for an excellent treatment of average-case com-

plexity.

Inspired by its usefuless in complexity theory, we apply the average-case perspective to problems in *computational learning theory* to obtain new positive and negative results.

**Average-Case Learning** As in the complexity-theoretic setting, we are partially motivated by the reality that algorithms may not encounter all instances of a learning problem. Our study is driven by more concrete considerations as well; despite the many steps that have been made in computational learning theory since Valiant introduced the PAC (probably approximately correct) learning model [Valiant, 1984] over 25 years ago, and despite the rich set of tools, the deep connections to other areas of theoretical computer science, and the clever approaches employed to make these steps, we are far from answering many basic questions. For some of these questions, research has stagnated, and we will make progress by studying them from the average-case perspective.

**Learning and Cryptography** Another motivation for looking at average-case learning comes from the connections between computational learning theory and cryptography. These two fields seem to have antithetical high-level aims: the former seeks ways of extracting information while the latter methods of hiding it. Many hardness of learning results are obtained by exploiting this relationship: if a class of Boolean functions is shown to have certain cryptographic properties, it will hide information essential to the learning process. Perhaps less frequently, the connection is used to construct cryptographic primitives from learning problems. In both cases, the learning problem is relaxed to the average case; that is, the success of the learning algorithm is measured according to specific distributions over the examples it sees and over the functions that label them (as opposed to arbitrary distributions for both). By exploiting this connection, we can obtain new results for hardness of learning and for cryptography.

## 1.2 Outline of Contributions

We organize the contributions of this thesis according to two categories: (1) efficient algrotihms for average-case learning and (2) connections between learning and cryptography.

**Efficient algorithms for learning most DNF formulas:** DNF formulas are an expressive (they can represent any Boolean function) and natural (bring an umbrella if: it is cloudy and raining, or it is hot and sunny) form of knowledge representation for humans. For this reason, the study of DNF formulas has become central to areas in artificial intelligence, from computational learning theory to automated theorem proving. The question of their efficient learnability dates back to Valiant's introduction of the PAC learning model [Valiant, 1984], and has remained open despite intensive study. Uniform distribution learning, where the accuracy of the learner's output is measured with respect to the uniform distribution over $\{0,1\}^n$, has received considerable attention, both because it is a natural formulation of the problem and because it facilitates the use of Fourier analysis. While important progress has been made [Linial *et al.*, 1993; Blum *et al.*, 1994; Mansour, 1995b; Jackson, 1997a; Bshouty *et al.*, 1999; Servedio, 2001; Bshouty *et al.*, 2003; Mossel *et al.*, 2004a], research in this area has slowed in recent years.

In this thesis we give efficient algorithms that (uniform distribution) learn the class of polynomial-size DNF formulas on the average, i.e., we describe a natural distribution over such formulas and show that our algorithms learn with high probability over the distribution. Our approach is based on proving new structural theorems, which are of independent interest, about most DNF formulas. Roughly speaking, we show that the polynomial representations of DNF formulas have certain properties (e.g. sparsity) that make learning feasible. This strategy has yielded landmark algorithms for learning in the worst-case setting (cf. the seminal results [Kushilevitz and Mansour, 1993a; Linial *et al.*, 1993; Mansour, 1995b]).

Our results address learning polynomial-size DNF formulas in two different models. The first result concerns the query model, in which the learning algorithm may query the target function on points of its choosing (its success is still measured with respect to the uniform distribution). A natural but extremely difficult variant of this model, which more closely resembles real life machine learning settings, is one where the learner may query a corrupted oracle, i.e., one which differs from the target function on a set of adversarially chosen points. While polynomial-size DNF formulas were shown to be efficiently learnable from queries in Jackson's seminal work [Jackson, 1997a], no matching result exists for the

noisy query model.

The second result concerns learning from labeled examples that are chosen uniformly from the Boolean hypercube. The fastest known algorithm for learning poly($n$)-size DNF formulas in this model takes time $n^{O(\log n)}$ [Verbeurgt, 1990]. We describe our contributions to learning DNF formulas in these two models below.

1. *Mansour's conjecture is true for random DNF.*

   Linial *et al.* were the first to reduce the problem of learning DNF formulas to certain properties of their representations as polynomials over the reals [Linial *et al.*, 1993]. In 1994, Mansour showed that every $t$-term DNF formula on $n$ variables could be approximated by a *sparse* polynomial, i.e., a polynomial $p : \{+1, -1\}^n \rightarrow \{+1, -1\}$, that satisfies $E[(p - f)^2] \leq \epsilon$ and has at most $t^{O(\log \log t \log(1/\epsilon))}$ non-zero coefficients. He conjectured that the sparsity could be improved to $t^{O(\log 1/\epsilon)}$. In Chapter 3, we make the first progress on this conjecture and show that it is true for several natural subclasses of DNF formulas, including randomly chosen (see Chapter 2) and read-$k$ DNF formulas. Recent work [Gopalan *et al.*, 2008b] shows that classes of Boolean functions with sparse polynomial approximations can be learned even when the queries are noisy. Thus, our result yields the first polynomial-time, noisy query algorithm for learning random and read-$k$ DNF formulas (for any constant error parameter and constant $k$). Applying recent work on sandwiching polynomials, our results also imply that $t^{-O(\log 1/\epsilon)}$-biased distributions fool the above subclasses of DNF formulas. This gives pseudorandom generators for these subclasses with shorter seed length than all previous work.

2. *Random monotone DNF are efficiently learnable from random examples.*

   The problem of efficiently learning the monotone subclass of polynomial-size DNF formulas from random examples was also posed in [Valiant, 1984]. This notoriously difficult question is still open, despite much study and the fact that known impediments to learning the non-monotone class (cf. [Blum *et al.*, 1994; Blum, 2003a]) do not exist for monotone DNF formulas. The best known algorithm [Servedio, 2004c] is efficient for $2^{O(\sqrt{\log n})}$-size monotone DNF only. In Chapter 4 we give the first algo-

rithm that learns randomly chosen monotone DNF formulas of arbitrary polynoimal size, improving results which efficiently learn $n^2$-size random monotone DNF formulas [Jackson and Servedio, 2005b]. Our main structural result is that most monotone DNF formulas reveal their term structure in their constant-degree Fourier coefficients.

**Connections between learning and cryptography:** In this thesis, we see that connections between learning and cryptography are naturally made through average-case analysis. First, by applying techniques from average-case complexity, we demonstrate new ways of using cryptographic assumptions to prove *limitations* on learning. We show the first computational hardness results for learning monotone Boolean functions. As counterpoint, we also exploit the average-case connection in the service of cryptography. We scrutinize the security of a cryptographic protocol, defining an average-case learning problem (that is hard in the worst-case) and proposing an efficient algorithm. A more detailed description of these contributions follows.

1. *Monotone polynomial-sized circuits are hard to learn if one-way functions exist.* Over the years a wide range of positive and negative results have been established for learning different classes of Boolean functions from uniformly distributed random examples. Despite intensive efforts, various simple classes are without efficient learning algorithms. Interestingly, for many of these classes, their monotone subclasses *do* have efficient algorithms. Prior to our work, however, the only negative result for learning monotone functions in this model is an information-theoretic lower bound showing that certain super-polynomial-size monotone circuits cannot be learned to accuracy $1/2 + \omega(\log n)/\sqrt{n}$ [Blum *et al.*, 1998]. This is in contrast with the situation for non-monotone functions, where a wide range of cryptographic hardness results establish that various "simple" classes of polynomial-size circuits are not learnable by polynomial-time algorithms.

   In Chapter 5 we establish cryptographic hardness results for learning various "simple" classes of monotone circuits, thus giving a computational analogue of the information-theoretic hardness results of Blum, *et al.* mentioned above. Some of our results show the cryptographic hardness of learning polynomial-size monotone circuits to accuracy

only slightly greater than $1/2 + 1/\sqrt{n}$; which is close to the optimal accuracy bound, by positive results of Blum, *et al.* Other results show that under a plausible cryptographic hardness assumption, a class of constant-depth, sub-polynomial-size circuits computing monotone functions is hard to learn. This result is close to optimal in terms of the circuit-size parameter by known positive results as well [Servedio, 2004a]. Our main tool is a complexity-theoretic approach to hardness amplification via noise sensitivity of monotone functions that was pioneered by O'Donnell [O'Donnell, 2004a].

2. *Learning an overcomplete basis: analysis of lattice-based signatures with perturbations.* Lattice-based cryptographic constructions are desirable not only because they provide security based on worst-case hardness assumptions, but also because they can be extremely efficient and practical. In 1997, Goldreich, Goldwasser and Halevi (GGH) [Goldreich *et al.*, 1997] proposed a lattice-based signature scheme and public-key encryption scheme that were inspired by the breakthrough work of Ajtai [Ajtai, 2004] and the apparent hardness of well-studied lattice problems. Since then, numerous variations of the GGH schemes have been proposed [Hoffstein *et al.*, 1998; Hoffstein *et al.*, 2001; Gentry *et al.*, 2008], including the commercial offering NTRUSign [Hoffstein *et al.*, 2003], which applies the ideas of the GGH signature scheme to the compact NTRU family of lattices. The NTRUSign scheme is reasonably efficient, but it has no known security proof.

   We propose in Chapter 6 a general technique for recovering parts of the secret key in lattice-based signature schemes that follow the Goldreich-Goldwasser-Halevi (GGH) and NTRUSign design with *perturbations*. Our technique is based on solving a learning problem in the average-case. Previously, Nguyen and Regev [Nguyen and Regev, 2009] cryptanalyzed GGH-style signature schemes (including NTRUSign) *without* perturbations; their attack was by reduction to a learning task they called the *hidden parallelepiped problem* (HPP). The main problem left open in their work was to handle schemes that use perturbation techniques.

   We observe that in such schemes, recovery of the secret key may be modeled as the problem of *learning an overcomplete basis*, a generalization of the HPP in which the

number of secret vectors exceeds the dimension. While HPP was solvable in the worst-case, it is easy to see that the problem of learning an overcomplete basis can not have a worst-case algorithm. To solve the average-case problem, we propose a special-purpose optimization algorithm based on higher-order *cumulants* of the signature distribution, and give theoretical and experimental evidence of its efficacy. Our results suggest (but do not conclusively prove) that NTRUSign is vulnerable to a polynomial-time attack.

## 1.3 Organizational Notes

In Chapter 2 we present the models, definitions, and standard tools used throughout this thesis. Some definitions and concepts are introduced in the specific chapters that they are used in. The results on efficiently learning DNF formulas follow in Chapters 3 and 4. We explore connections between learning and cryptography in the two following chapters. Chapter 5 contains the hardness result for polynomial-size monotone circuits, and Chapter 6 contains our analysis of lattice-based signature schemes. Finally, we conclude in Chapter 7 with suggestions for future work.

# Chapter 2

# Definitions and Preliminaries

## 2.1 Boolean Functions and Fourier Analysis

The concept classes we study will be subclasses of Boolean functions $f : \{0,1\}^n \rightarrow \{0,1\}$. We often refer to classes of Boolean functions by their particular representations. Any Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ can be expressed as a disjunction of conjunctions of Boolean literals, i.e. as an OR of ANDs. Such a logical formula is said to be a *disjunctive normal form* or DNF formula. Each AND in the formula is called a *term*, and the size of the DNF formula is measured by the number of terms it has. A *read-k* DNF formula is one in which the maximum number of variable occurences is bounded by $k$. Boolean functions may also be represented by decision trees, which are directed trees with variable nodes and arcs which represent assignments to a variable. See [Kushilevitz and Mansour, 1993a] for a formal definition. A Boolean function having at most $k$ relevant variables is called a $k$-junta. For any class of Boolean functions, we may consider the subclass of functions that are *monotone*, i.e. those functions satisfying $f(x) \geq f(y)$ whenever $x_i \geq y_i$ for all $i$. Recall that every monotone Boolean function $f$ has a unique representation as a reduced monotone DNF. We say that a term $T$ of such a monotone DNF is *uniquely satisfied* by input $x$ if $x$ satisfies $T$ and no other term of $f$.

In addition to the uniform distribution over the Boolean hypercube, we sometimes consider *product* distributions, i.e., a product of $n$ independent random variables in $\{0,1\}$, where the $i$'th variable takes the value 1 with probability $p_i$ for $i = 1, \cdots, n$. The following

consequence of the Four Functions Theorem [Kleitman, 1966] will be useful in our study of monotone functions over product distributions.

**Theorem 1.** *Let $e$, $f$, $\neg g$, and $\neg h$ be monotone Boolean functions over $\{0,1\}^n$. Then for any product distribution $\mathcal{D}$ over $\{0,1\}^n$, $\Pr_{\mathcal{D}}[e \wedge f] \geq \Pr_{\mathcal{D}}[e]\Pr_{\mathcal{D}}[f]$, $\Pr_{\mathcal{D}}[g \wedge h] \geq \Pr_{\mathcal{D}}[g]\Pr_{\mathcal{D}}[h]$, and $\Pr_{\mathcal{D}}[f \wedge g] \leq \Pr_{\mathcal{D}}[f]\Pr_{\mathcal{D}}[g]$.*

The application of discrete Fourier Analysis to Boolean functions is by now standard in theoretical computer science. We only mention a few concepts here to establish notation; see these surveys [De Wolf, 2008; O'Donnell, 2008] for excellent introductions to the topic.

It is convenient to view functions with Boolean outputs as having outputs in the range $\{+1,-1\}$, with $-1$ signifying TRUE and $+1$ signifying FALSE. (A function whose output is in the standard range $\{0,1\}$ can be considered via the conversion $1-2f(x)$.) Real-valued functions over the domain $\{+1,-1\}^n$ form a $2^n$-dimensional vector space with inner product

$$\langle f, g\rangle = E_x[f \cdot g] = 2^{-n} \sum_{x \in \{+1,-1\}^n} f(x)g(x).$$

For a set $S \subseteq [n]$, let $\chi_S = \prod_{i \in S} x_i$. It is easy to see that these $2^n$ different functions, also called *parity* functions, form an orthonormal basis for the space of functions $f : \{+1,-1\}^n \to \mathbb{R}$.

The Fourier inversion theorem says that every function $f : \{+1,-1\}^n \to \mathbb{R}$ can be uniquely expressed by its Fourier expansion, i.e., a polynomial over $\{+1,-1\}^n$:

$$f(x) = \sum_S \hat{f}(S)\chi_S(x),$$

where $\hat{f}(S) = \mathbf{E}[f \cdot \chi_S]$, the coefficient of $\chi_S$ is called a *Fourier coefficient*. Functions over $\{0,1\}^n$ may be thought of as ranging over $\{+1,-1\}^n$ via the conversion $x_i \to 1 - 2x_i$. In some settings we will consider instead the vector space over functions $f : \{0,1\}^n \to \mathbb{R}$ and redefine the basis functions to be $\chi_S(x) = \prod_{i \in S}(-1)^{x_i}$.

## 2.2 Learning

We consider problems where a learning algorithm has some type of access to the input-output behaviour of an unknown function $f$. The function will come from the class $\mathcal{C}$ of

concepts to be learned, which will be a set of functions over a domain $\mathcal{X}$ (typically $\{0,1\}^n$). The goal of the algorithm is to output a representation which approximates the behavior of the *target function* $f$. The learning model specifies the nature of the access to $f$ as well as the criterion a successful algorithm must satisfy, such as efficiency and quality of the output.

### 2.2.1 Uniform Distribution Learning

In this thesis we focus on the uniform distribution PAC (Probably Approximately Correct) model, a well-studied variant of the prevalent PAC model introduced by Valiant ([Valiant, 1984]). Access to the target function $f$ may be in the form of a *random example oracle* $EX(f, U_n)$, which, when queried generates a random labeled example $(x, f(x))$, where $x$ is drawn uniformly at random from $\mathcal{X}$. Alternatively, access to $f$ may be in the form of a *query oracle*, $Q(f)$, which receives as input any $x$ from the domain and outputs a label $f(x)$.

In the uniform distribution PAC learning model (with queries or random examples) the performance of the learning algorithm is measured according to the uniform distribution over the domain. After interacting with its oracle, the learning algorithm will output a hypothesis $h$, and the error of $h$ is defined to be $\Pr[h(x) \neq f(x)]$ where $x$ is drawn uniformly at random from $\mathcal{X}$. We say that $A$ learns $\mathcal{C}$ if for any $f \in \mathcal{C}$ and every $0 < \epsilon, \delta < 1$, with probability at least $1 - \delta$ (over the randomness of $A$ and any randomness used by the oracle), algorithm $A$ outputs a hypothesis $h$ which has error at most $\epsilon$. The efficiency of $A$ is measured with respect to the domain of the concepts $\mathcal{C}$ and the parameters $\epsilon$ and $\delta$, and $A$ is considered efficient if it learns $\mathcal{C}$ in time $p(n, 1/\epsilon, 1/\delta)$ for some polynomial $p$ where $n$ is the size of the input to $f$.

Algorithms and hardness results in the uniform distribution learning framework have interesting connections with topics such as discrete Fourier analysis [Mansour, 1994], circuit complexity [Linial *et al.*, 1993], noise sensitivity and influence of variables in Boolean functions [Kahn *et al.*, 1988; Benjamini *et al.*, 1999; Klivans *et al.*, 2004; O'Donnell and Servedio, 2007], coding theory [Feldman *et al.*, 2006], privacy [Blum *et al.*, 2008; Kasiviswanathan *et al.*, 2008], and cryptography [Blum *et al.*, 1993; Kharitonov, 1995].

### 2.2.2 Average-case Learning

The previous models admit a natural relaxation to the average-case in analogy with average-case complexity. For any concept class $\mathcal{C}$, we may define a distribution ensemble $\mathcal{D} = \{D_n\}$, where each $D_n$ is a distribution over those concepts $c \in \mathcal{C}$ with $c : \{0,1\}^n \rightarrow \{0,1\}$. We say that $A$ average-case learns $\mathcal{C}$ with respect to $\mathcal{D}$ if for every $n$ and any $0 < \epsilon < 1$, with probability at least $1 - n^{-\Omega(1)}$ (over a random draw of $f$ from $D_n$, the randomness of $A$, and any randomness used by its example oracle), the hypothesis output by $A$ satisfies:

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

We will consider $A$ to be efficient if it runs in time polynomial in $n$ and $1/\epsilon$.

For size-$t$ DNF formulas, we consider the distribution $D_n^t$ over $t$-term DNF formulas on $n$ variables induced by the following process: each term is independently and uniformly chosen at random from all $t \cdot \binom{n}{\lfloor \log t \rfloor}$ ANDs of size exactly $\lfloor \log t \rfloor$ over $x_1, \ldots, x_n$. A notion of random DNF formulas was first considered in [Aizenstein and Pitt, 1995] and later in [Jackson and Servedio, 2006]. We follow the definition from [Jackson and Servedio, 2006], which contains a discussion of the factors considered in the choice of the model. We briefly discuss here the choice of the term size. Consider drawing terms randomly from the space of terms of size exactly $k$ instead of $\log t$. If $k$ is too large relative to $t$, then a random $f \in D_n^t$ will likely have $\Pr_{x \in U_n}[f(x) = 1] \approx 0$, and if $k$ is too small relative to $t$ then a random $f \in D_n^t$ will likely have $\Pr_{x \in U_n}[f(x) = 1] \approx 1$; such functions are trivial to learn to high accuracy using either the constant-0 or constant-1 hypothesis. A straightforward analysis shows that for $k = \lfloor \log t \rfloor$ we have that $\mathbf{E}_{f \in D_{\mathcal{C},n}}[\Pr_{x \in U_n}[f(x) = 1]]$ is bounded away from both 0 and 1, and thus we feel that this is an appealing and natural choice.

### 2.2.3 Agnostic Learning

Agnostic models of learning [Kearns *et al.*, 1994b] attempt to weaken or eliminate assumptions on the target function. Removing assumptions about the target function reflects our belief that data encountered in the real world may not have a succinct explanation, but it also makes learning more difficult. We first describe the traditional framework for agnostically learning concept classes with respect to the uniform distribution and then give

a slightly modified definition for an "average-case" version of agnostic learning where the unknown concept (in this case a DNF formula) is randomly chosen.

**Definition 1** (Standard agnostic model)**.** *Let $\mathcal{U}$ be the uniform distribution on $\{+1,-1\}^n$, and let $f : \{+1,-1\}^n \to \{+1,-1\}$ be an arbitrary function. Define*

$$\mathsf{opt} = \min_{c \in \mathcal{C}} \Pr_{x \sim \mathcal{U}}[c(x) \neq f(x)].$$

*That is,* opt *is the error of the best fitting concept in $\mathcal{C}$ with respect to $\mathcal{U}$. We say that an algorithm $A$ agnostically learns $\mathcal{C}$ if the following holds for any $f$: if $A$ is given black-box access to $f$ then with high probability $A$ outputs a hypothesis $h$ such that $\Pr_{x \sim \mathcal{U}}[h(x) \neq f(x)] \leq \mathsf{opt} + \epsilon$.*

The intuition behind the above definition is that a learner—given access to a concept $c \in \mathcal{C}$ where an $\eta$ fraction of $c$'s inputs have been adversarially corrupted—should still be able to output a hypothesis with accuracy $\eta + \epsilon$ (achieving error better than $\eta$ may not be possible, as the adversary could embed a completely random function on an $\eta$ fraction of $c$'s inputs). Here $\eta$ plays the role of opt.

This motivates the following definition for agnostically learning a randomly chosen concept from some class $\mathcal{C}$:

**Definition 2** (Agnostically learning random concepts)**.** *Let $\mathcal{C}$ be a concept class and $\mathcal{D}$ be a distribution ensemble over $\mathcal{C}$. We say that an algorithm $A$ agnostically learns random concepts from $\mathcal{C}$ (with respect to $\mathcal{D}$) if with probability at least $1 - n^{-\Omega(1)}$ over the choice of $c$ according to $\mathcal{D}$ the following holds: if the learner is given black-box access to $c'$ and $\Pr_{x \in \{+1,-1\}^n}[c(x) \neq c'(x)] \leq \eta$, then $A$ outputs a hypothesis $h$ such that $\Pr_{x \in \{+1,-1\}^n}[h(x) \neq c'(x)] \leq \eta + \epsilon$.*

We are unaware of any prior work defining an agnostic framework for learning randomly chosen concepts.

The main result we use to connect the approximation of DNF formulas by sparse polynomials with agnostic learning is due to Gopalan *et al.* [Gopalan *et al.*, 2008b]:

**Theorem 2** ([Gopalan *et al.*, 2008b])**.** *Let $\mathcal{C}$ be a concept class such that for every $c \in \mathcal{C}$ there exists a polynomial $p$ such that $\|p\|_1 \leq s$ and $\mathbf{E}_{x \in \{+1,-1\}^n}[|p(x) - c(x)|^2] \leq \epsilon^2/2$. Then*

*there exists an algorithm $B$ such that the following holds: given black-box access to any Boolean function $f : \{+1,-1\}^n \to \{+1,-1\}$, $B$ runs in time* $\mathrm{poly}(n, s, 1/\epsilon)$ *and outputs a hypothesis $h : \{+1,-1\}^n \to \{+1,-1\}$ with*

$$\Pr_{x \in \{+1,-1\}^n}[h(x) \neq f(x)] \leq \mathsf{opt} + \epsilon.$$

## 2.3 Cryptography

We also work with one-way functions, pseudorandom generators and pseudorandom functions which are by now standard notions in theoretical computer science. The appropriate definition for these cryptographic primitives varies according to the setting. We give general definitions here and postpone formal definitions until the need arises.

**One-way Functions.** A function $f : \{0,1\}^* \to \{0,1\}^*$ is a *one-way function* if it is efficiently computable (by some family of circuits or a deterministic Turing Machine) and for any efficient adversary $A$ (which, depending on the context may be a circuit or randomized TM), it holds that $\Pr[A(f(x)) \in f^{-1}(x)]$ is small, where the probability is taken over a random input $x$ and the randomness of $A$.

**Pseudorandom Generators** A function $G : \{0,1\}^n \to \{0,1\}^m$ is a *pseudorandom generator* if it is computable in polynomial time and the distribution $G(x)$ (where $x$ is uniform random from $\{0,1\}^n$) is pseudorandom, i.e. no adversary (adversaries may be small circuits or PPTs) can distinguish $G(x)$ from the uniform distribution over $\{0,1\}^m$. The stretch of $G$ is $l(n) = m(n) - n > 0$.

**Pseudorandom functions.** A family of functions $\mathcal{F} = \{\mathcal{F}_n\}$ is a *pseudorandom function family* if for every $n$, $\mathcal{F}_n$ is comprised of functions haveing polynomial-size representations, and no PPT oracle algorithm $A$ can distinguish a random $f$ drawn from $\mathcal{F}_n$ and a truly random function over $\{0,1\}^n$.

It is well known (cf. [Håstad *et al.*, 1999; Goldreich *et al.*, 1986]) that the existence of pseudorandom generators and pseudorandom function families follows from the existence of one-way functions.

# Chapter 3

# Mansour's Conjecture is True for Random DNF

We now present our results on efficiently learning polynomial-size DNF formulas. In this chapter, we apply average-case analysis to the problem of learning DNF formulas from noisy queries. The main technical contribution of this chapter is to prove that the conjecture of Mansour [Mansour, 1995b] is true for *most* DNF formulas and for read-$k$ DNF formulas.

## 3.1 Introduction

Let $f : \{0,1\}^n \to \{0,1\}$ be a DNF formula, *i.e.,* a function of the form $T_1 \vee \cdots \vee T_t$ where each $T_i$ is a conjunction of at most $n$ literals. In this chapter, we are concerned with the following question: how well can a real-valued polynomial $p$ approximate the Boolean function $f$? This is an important problem in computational learning theory, as real-valued polynomials play a critical role in developing learning algorithms for DNF formulas.

Over the last twenty years, considerable work has gone into finding polynomials $p$ with certain properties (*e.g.,* low-degree, sparse) such that

$$\mathbf{E}_{x \in \{0,1\}^n}[(p(x) - f(x))^2] \leq \epsilon.$$

In 1989, Linial *et al.* [Linial *et al.*, 1993] were the first to prove that for any $t$-term DNF formula $f$, there exists a polynomial $p : \{0,1\}^n \to \mathbb{R}$ of degree $O(\log(t/\epsilon)^2)$ such

that $\mathbf{E}_{x\in\{0,1\}^n}[(p(x) - f(x))^2] \leq \epsilon$. They showed that this type of approximation implies a quasipolynomial-time algorithm for PAC learning DNF formulas with respect to the uniform distribution. Kalai *et al.* [Kalai *et al.*, 2008] observed that this fact actually implies something stronger, namely a quasipolynomial-time agnostic learning algorithm for learning DNF formulas (with respect to the uniform distribution). Additionally, the above approximation was used in recent work due to Bazzi [Bazzi, 2007] and Razborov [Razborov, 2008] to show that bounded independence fools DNF formulas.

Three years later, building on the work of Linial *et al.* Mansour [Mansour, 1995b] proved that for any DNF formula with $t$ terms, there exists a polynomial $p$ defined over $\{0,1\}^n$ with *sparsity* $t^{O(\log\log t \log(1/\epsilon))}$ such that $\mathbf{E}_{x\in\{0,1\}^n}[(p(x) - f(x))^2] \leq \epsilon$ (for $1/\epsilon = \text{poly}(n)$). By sparsity we mean the number of non-zero Fourier coefficients of $p$. This result implied a nearly polynomial-time *query* algorithm for PAC learning DNF formulas with respect to the uniform distribution.

Mansour conjectured [Mansour, 1994] that the bound above could improved to $t^{O(\log 1/\epsilon)}$. Such an improvement would imply a polynomial-time query algorithm for learning DNF formulas with respect to the uniform distribution (to within any constant accuracy), and learning DNF formulas in this model was a major open problem at that time.

In a celebrated work from 1994, Jeff Jackson proved that DNF formulas were learnable in polynomial time (with queries, with respect to the uniform distribution) *without* proving the Mansour conjecture. His "Harmonic Sieve" algorithm [Jackson, 1997b] used boosting in combination with some weak approximation properties of polynomials. As such, for several years, Mansour's conjecture remained open and attracted considerable interest, but its resolution did not imply any new results in learning theory.

In 2008, Gopalan *et al.* [Gopalan *et al.*, 2008b] proved that a positive resolution to the Mansour conjecture also implies an efficient query algorithm for *agnostically* learning (see Section 2.2.3) DNF formulas (to within any constant error parameter). The agnostic model of learning is a challenging learning scenario that requires the learner to succeed in the presence of adversarial noise. Roughly, Gopalan *et al.* showed that if a class of Boolean functions $\mathcal{C}$ can be $\epsilon$-approximated by polynomials of sparsity $s$, then there is a query algorithm for agnostically learning $\mathcal{C}$ in time $\text{poly}(s, 1/\epsilon)$ (since decision trees are

approximated by sparse polynomials, they obtained the first polynomial-time query algorithm for agnostically learning decision trees with respect to the uniform distribution on $\{0,1\}^n$). Whether DNF formulas can be agnostically learned (with queries, with respect to the uniform distribution) still remains a difficult open problem [Gopalan *et al.*, 2008a].

### 3.1.1 Our Results

We prove that the Mansour conjecture is true for several well-studied subclasses of DNF formulas. As far as we know, prior to this work, the Mansour conjecture was not known to be true for any interesting class of DNF formulas.

Our first result shows that the Mansour conjecture is true for the class of randomly chosen DNF formulas:

**Theorem 3.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a DNF formula with $t = n^{O(1)}$ terms where each term is chosen independently from the set of all terms of length $\lfloor \log t \rfloor$. Then with probability $1 - n^{-\Omega(1)}$ (over the choice of the DNF formula), there exists a polynomial $p$ with sparsity $t^{O(\log 1/\epsilon)}$ such that $\mathbf{E}[(p(x) - f(x))^2] \leq \epsilon$.*

For $t = n^{\Theta(1)}$, the conclusion of Theorem 3 holds with probability at least $1 - n^{-\Omega(\log t)}$. Our second result is that the Mansour conjecture is true for the class of read-$k$ DNF formulas (for constant $k$):

**Theorem 4.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a DNF formula with $t$ terms where each literal appears at most $k$ times. Then there exists a polynomial $p$ with sparsity $t^{O(2^{4k} \log 1/\epsilon)}$ such that $\mathbf{E}[(p(x) - f(x))^2] \leq \epsilon$.*

Even for the case $k = 1$, Mansour's conjecture was not known to be true. Mansour [Mansour, 1995b] proves that any polynomial that approximates read-once DNF formulas to $\epsilon$ accuracy must have *degree* at least $\Omega(\log t \log(1/\epsilon)/\log\log(1/\epsilon))$. He further shows that a "low-degree" strategy of selecting all of a DNF's Fourier coefficients of monomials up to degree $d$ results in a polynomial $p$ with sparsity $t^{O(\log\log t \log 1/\epsilon)}$. It is not clear, however, how to improve this to the desired $t^{O(\log 1/\epsilon)}$ bound.

As mentioned earlier, by applying the result of Gopalan *et al.* [Gopalan *et al.*, 2008b], we obtain the first polynomial-time query algorithms for agnostically learning the above

classes of DNF formulas to within any constant accuracy parameter. We consider this an important step towards agnostically learning all DNF formulas.

**Corollary 1.** *Let $\mathcal{C}$ be the class of DNF formulas with $t = n^{O}(1)$ terms and define $\mathcal{D}^t$ as in Section 2.2.2. Then there is a query-algorithm for agnostically learning $\mathcal{C}$ with respect to $\mathcal{D}^t$ to accuracy $\epsilon$ in time $\mathrm{poly}(n) \cdot t^{O(\log 1/\epsilon)}$ with probability $1 - n^{-\Omega(1)}$ (over $\mathcal{D}^t$).*

We define the notion of agnostic learning with respect to randomly chosen concept classes in Chapter 2. For $t = n^{\Theta(1)}$, Corollary 1 holds for a $1-n^{-\Omega(\log t)}$ fraction of randomly chosen DNF formulas. We also obtain a corresponding agnostic learning algorithm for read-$k$ DNF formulas:

**Corollary 2.** *Let $\mathcal{C}$ be the class of read-$k$ DNF formulas with $t$ terms. Then there is a query-algorithm for agnostically learning $\mathcal{C}$ with respect to the uniform distribution on $\{0,1\}^n$ to accuracy $\epsilon$ in time $\mathrm{poly}(n) \cdot t^{O(2^{4k} \log 1/\epsilon)}$.*

Our sparse polynomial approximators can also be used in conjunction with recent work due to De *et al.* to show that for any randomly chosen or read-$k$ DNF $f$, $1/t^{O(\log 1/\epsilon)}$-biased distributions fool $f$ (for $k = O(1)$):

**Theorem 5.** *Let $f$ be a randomly chosen DNF formula or a read-$k$ DNF formula. Then (with probability $1 - n^{-\Omega(1)}$ for random DNF formulas) there exists a pseudorandom generator $G$ such that*

$$\left| \Pr_{x \in \{0,1\}^s}[f(G(x)) = 1] - \Pr_{z \in \{0,1\}^n}[f(z) = 1] \right| \leq \epsilon$$

*with $s = O(\log n + \log t \cdot \log(1/\epsilon))$.*

Previously it was only known that these types of biased distributions fool read-once DNF formulas [De *et al.*, 2009].

### 3.1.2 Related Work

As mentioned earlier, Mansour, using the random restriction machinery of Håstad and Linial *et al.* [Håstad, 1986; Linial *et al.*, 1993] had shown that for any DNF formula $f$, there exists a $p$ of sparsity $t^{O(\log \log t \log 1/\epsilon)}$ that approximates $f$. The approximating polynomial

is constructed by the "low-degree" strategy of selecting all of the DNF's Fourier coefficients up to degree $O(\log t/\epsilon \cdot \log 1/\epsilon)$. A much simpler analysis, essentially observed in [Boppana, 1997], shows that the low-degree strategy yields an $\epsilon$-approximation with degree $O(\log t/\epsilon \cdot 1/\epsilon)$. Later work [Håstad, 2001] shows that taking coefficients of degree up to $O(\log t/\epsilon) \cdot \min\{\log t, \log 1/\epsilon\}$ suffices, which improves on Mansour's bound for $1/\epsilon > t$.

The subclasses of DNF formulas that we show are agnostically learnable have been well-studied in the PAC model of learning. Monotone read-$k$ DNF formulas were shown to be PAC-learnable with respect to the uniform distribution by Hancock and Mansour [Hancock and Mansour, 1991b], and random DNF formulas were recently shown to be learnable on average with respect to the uniform distribution in the following sequence of works [Jackson and Servedio, 2005b; Jackson *et al.*, 2008b; Sellie, 2008b; Sellie, 2009].

Recently (and independently) De *et al.* proved that for any read-once DNF formula $f$, there exists an approximating polynomial $p$ of sparsity $t^{O(\log 1/\epsilon)}$. More specifically, De *et al.* showed that for any class of functions $\mathcal{C}$ fooled by $\delta$-biased sets, there exist sparse, sandwiching polynomials for $\mathcal{C}$ where the sparsity depends on $\delta$. Since they show that $1/t^{O(\log 1/\epsilon)}$-biased sets fool read-once DNF formulas, the existence of a sparse approximator for the read-once case is implicit in their work.

### 3.1.3 Our Approach

As stated above, our proof does not analyze the Fourier coefficients of DNF formulas, and our approach is considerably simpler than the random-restriction method taken by Mansour (we consider the lack of Fourier analysis a feature of the proof, given that all previous work on this problem has been Fourier-based). Instead, we use polynomial interpolation.

**A Basic Example.** Consider a DNF formula $f = T_1 \vee \cdots \vee T_t$ where each $T_i$ is on a disjoint set of $\log t$ variables (assume $t$ is a power of 2). The probability that each term is satisfied is exactly $1/t$, and the expected number of satisfied terms is one. Further, since the terms are disjoint, with high probability over the choice of random input, only a few—say $d$—terms will be satisfied. As such, we construct a univariate polynomial $p$ with $p(0) = 0$ and $p(i) = 1$ for $1 \leq i \leq d$. Then $p(T_1 + \cdots + T_t)$ will be exactly equal to $f$ as long as at

most $d$ terms are satisfied. A careful calculation shows that the inputs where $p$ is incorrect will not contribute too much to $\mathbf{E}[(f-p)^2]$, as there are few of them. Setting parameters appropriately yields a polynomial $p$ that is both sparse and an $\epsilon$-approximator of $f$.

**Random and read-once DNF formulas.** More generally, we adopt the following strategy: given a DNF formula $f$ (randomly chosen or read-once) either (1) with sufficiently high probability a random input does not satisfy too many terms of $f$ or (2) $f$ is highly biased. In the former case we can use polynomial interpolation to construct a sparse approximator and in the latter case we can simply use the constant 0 or 1 function.

The probability calculations are a bit delicate, as we must ensure that the probability of many terms being satisfied decays faster than the growth rate of our polynomial approximators. For the case of random DNF formulas, we make use of some recent work due to Jackson *et al.* on learning random monotone DNF formulas [Jackson *et al.*, 2008b].

**Read-$k$ DNF formulas.** Read-$k$ DNF formulas do not fit into the above dichotomy, so we do not use the sum $T_1 + \cdots + T_t$ inside the univariate polynomial. Instead, we use a sum of *formulas* (rather than terms) based on a construction from [Razborov, 2008]. We modify Razborov's construction to exploit the fact that terms in a read-$k$ DNF formula do not share variables with many other terms. Our analysis shows that we can then employ the previous strategy: either (1) with sufficiently high probability a random input does not satisfy too many formulas in the sum or (2) $f$ is highly biased.

## 3.2 Preliminaries

See Chapter 2 for a treatment of the learning models and various properties of Boolean functions used in this chapter.

### 3.2.1 Sparse Polynomials

Every function $f : \{0,1\}^n \to \mathbb{R}$ can be expressed by its Fourier expansion: $f(x) = \sum_S \hat{f}(S)\chi_S(x)$ where $\chi_S(x) = \prod_{i \in S}(-1)^{x_i}$ for $S \subseteq [n]$, and $\hat{f}(S) = \mathbf{E}[f \cdot \chi_S]$. The Fourier expansion of $f$ can be thought of as the unique polynomial representation of $f$ over $\{+1,-1\}^n$

under the map $x_i \mapsto 1 - 2x_i$.

Mansour conjectured that polynomial-size DNF formulas could be approximated by *sparse* polynomials over $\{+1,-1\}^n$. We say a polynomial $p : \{+1,-1\}^n \to \mathbb{R}$ has sparsity $s$ if it has at most $s$ non-zero coefficients. We state Mansour's conjecture as originally posed in [Mansour, 1994], which uses the convention of representing FALSE by $+1$ and TRUE by $-1$.

**Conjecture 1** ([Mansour, 1994])**.** *Let $f : \{+1,-1\}^n \to \{+1,-1\}$ be any function computable by a $t$-term DNF formula. Then there exists a polynomial $p : \{+1,-1\}^n \to \mathbb{R}$ with $t^{O(\log 1/\epsilon)}$ terms such that $\mathbf{E}[(f-p)^2] \leq \epsilon$.*

We will prove the conjecture to be true for various subclasses of polynomial-size DNF formulas. In our setting, Boolean functions will output 0 for FALSE and 1 for TRUE. However, we can easily change the range by setting $f^{\pm} := 1 - 2 \cdot f$. Changing the range to $\{+1,-1\}$ changes the accuracy of the approximation by at most a factor of 4: $\mathbf{E}[((1-2f)-(1-2p))^2] = 4\mathbf{E}[(f-p)^2]$, and it increases the sparsity by at most 1.

Given a Boolean function $f$, we construct a sparse approximating polynomial over $\{+1,-1\}^n$ by giving an approximating polynomial $p : \{0,1\}^n \to \mathbb{R}$ with real coefficients that has small spectral norm. The rest of the section gives us some tools to construct such polynomials and explains why doing so yields sparse approximators.

**Definition 3.** *The* Fourier $\ell_1$-norm *(also called the* spectral norm*) of a function $p : \{0,1\}^n \to \mathbb{R}$ is defined to be $\|p\|_1 := \sum_S |\hat{p}(S)|$. We will also use the following minor variant, $\|p\|_1^{\neq \emptyset} := \sum_{S \neq \emptyset} |\hat{p}(S)|$.*

The following two facts about the spectral norm of functions will allow us to construct polynomials over $\{0,1\}^n$ naturally from DNF formulas.

**Fact 1.** *Let $p : \{0,1\}^m \to \mathbb{R}$ be a polynomial with coefficients $p_S \in \mathbb{R}$ for $S \subseteq [m]$, and $q_1, \ldots, q_m : \{0,1\}^n \to \{0,1\}$ be arbitrary Boolean functions. Then $p(q_1, \ldots, q_m) = \sum_S p_S \prod_{i \in S} q_i$ is a polynomial over $\{0,1\}^n$ with spectral norm at most*

$$\sum_{S \subseteq [m]} |p_S| \prod_{i \in S} ||q_i||_1.$$

*Proof.* The fact follows by observing that for any $p, q : \{0,1\}^n \to \mathbb{R}$, we have $||p+q||_1 \leq ||p||_1 + ||q||_1$ and $||pq||_1 \leq ||p||_1||q||_1$. ■

**Fact 2.** *Let $T : \{0,1\}^n \to \{0,1\}$ be an AND of a subset of its literals. Then $||T||_1 = 1$.*

Finally, using the fact below, we show why approximating polynomials with small spectral norm give sparse approximating polynomials.

**Fact 3** ([Kushilevitz and Mansour, 1993b])**.** *Given any function $f : \{0,1\}^n \to \mathbb{R}$ and $\epsilon > 0$, let $\mathcal{S} = \left\{S \subseteq [n] : \left|\hat{f}(S)\right| \geq \epsilon/\|f\|_1\right\}$, and $g(x) = \sum_{S\in\mathcal{S}} \hat{f}(S)\chi_s(x)$. Then $\mathbf{E}[(f-g)^2] \leq \epsilon$, and $|\mathcal{S}| \leq \|f\|_1^2/\epsilon$.*

Now, given functions $f, p : \{0,1\}^n \to \mathbb{R}$ such that $E[(f-p)^2] \leq \epsilon$, we may construct a $4\epsilon$-approximator for $f$ with sparsity $||p||_1^2/\epsilon$ by defining $p'(x) = \sum_{S\in\mathcal{S}} \hat{p}(S)\chi_S(x)$ as in Fact 3. Clearly $p'$ has sparsity $||p||_1^2/\epsilon$, and

$$E[(f-p')^2] = E[(f-p+p-p')^2] \leq E[2((f-p)^2 + (p-p')^2)] \leq 4\epsilon,$$

where the first inequality follows from the inequality $(a+b)^2 \leq 2(a^2+b^2)$ for any reals $a$ and $b$.

## 3.3 Approximating DNFs using univariate polynomial interpolation

Let $f = T_1 \vee T_2 \vee \cdots \vee T_t$ be any DNF formula. We say $T_i(x) = 1$ if $x$ satisfies the term $T_i$, and 0 otherwise. Let $y_f : \{0,1\}^n \to \{0,\ldots,t\}$ be the function that outputs the number of terms of $f$ satisfied by $x$, *i.e.*, $y_f(x) = T_1(x) + T_2(x) + \cdots + T_t(x)$.

Our constructions will use the following univariate polynomial $P_d$ to interpolate the values of $f$ on inputs $\{x : y_f(x) \leq d\}$. For illustrations of $P_d$ for small values of $d$, see Figures 3.1 and 3.2 by Li-Yang Tan.

**Fact 4.** *Let*

$$P_d(y) := (-1)^{d+1}\frac{(y-1)(y-2)\cdots(y-d)}{d!} + 1. \tag{3.1}$$

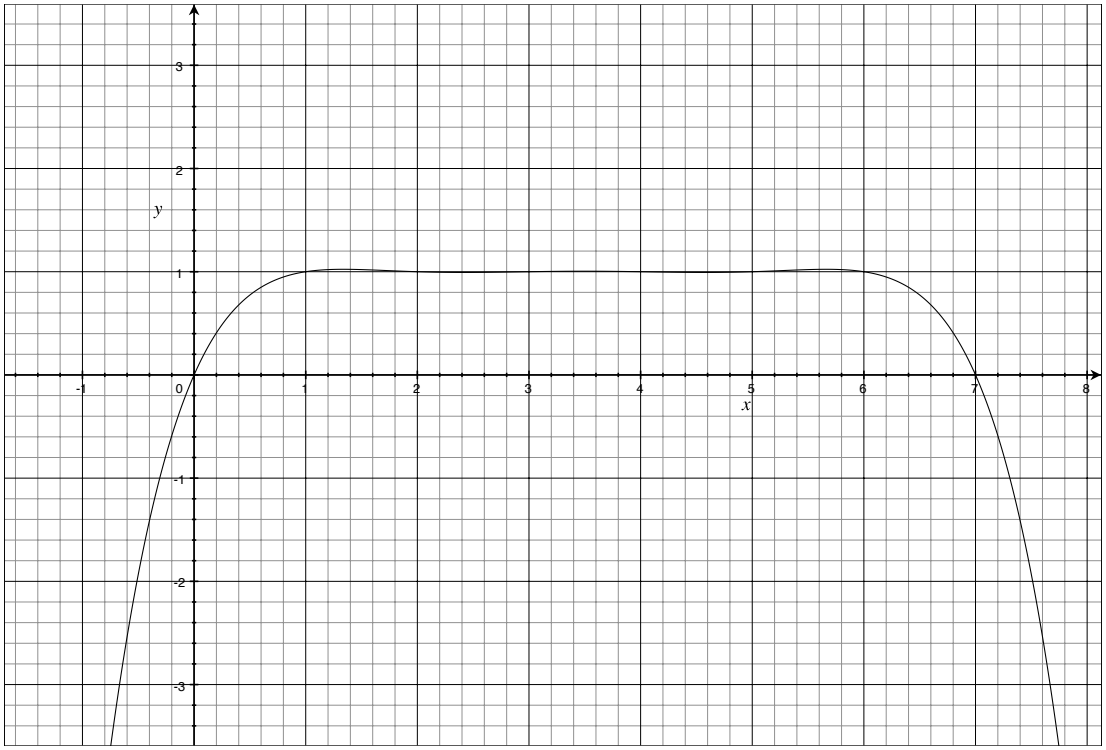


Figure 3.1: The polynomial $P_6$

*Then, (1) the polynomial $P_d$ is a degree-$d$ polynomial in $y$; (2) $P_d(0) = 0$, $P_d(y) = 1$ for $y \in [d]$, and for $y \in [t] \setminus [d]$, $P_d(y) = -\binom{y-1}{d} + 1 \leq 0$ if $d$ is even and $P_d(y) = \binom{y-1}{d} + 1 > 1$ if $d$ is odd; and (3) the sum of the magnitudes of $P_d$'s coefficients is $d$.*

*Proof.* Properties (1) and (2) can be easily verified by inspection. Expanding the falling factorial, we get that $(y-1)(y-2)\cdots(y-d) = \sum_{j=0}^{d}(-1)^{d-j}\left[{d+1 \atop j+1}\right] y^j$, where $\left[{a \atop b}\right]$ denotes a Stirling number of the first kind. The Stirling numbers of the first kind count the number of permutations of $a$ elements with $b$ disjoint cycles. Therefore, $\sum_{j=0}^{d}\left[{d+1 \atop j+1}\right] = (d+1)!$ [Graham *et al.*, 1994]. The constant coefficient of $P_d$ is 0 by Property (2), thus the sum of the absolute values of the other coefficients is $((d+1)! - d!)/d! = d$. ■

For any $t$-term DNF formula $f$, we can construct a polynomial $p_{f,d} : \{0,1\}^n \to \mathbb{R}$ defined as $p_{f,d} := P_d \circ y_f$. A simple calculation, given below, shows that the $\ell_1$-norm of $p_{f,d}$ is polynomial in $t$ and exponential in $d$.

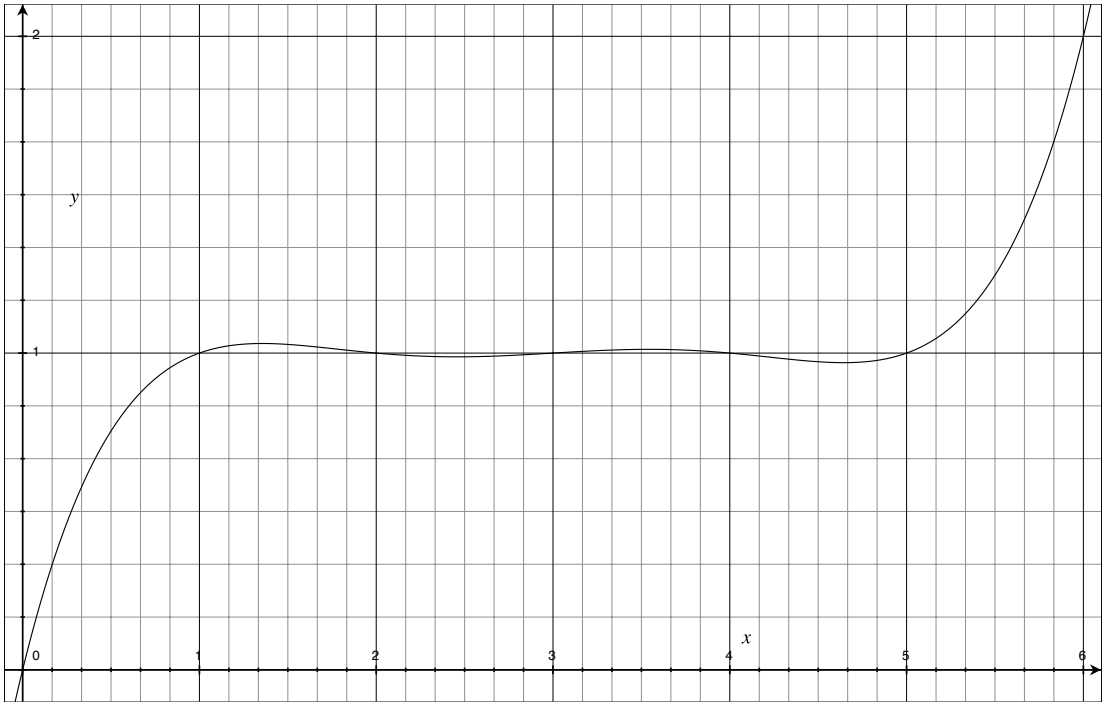


Figure 3.2: The polynomial $P_5$

**Lemma 1.** *Let $f$ be a $t$-term DNF formula, then $\|p_{f,d}\|_1 \leq t^{O(d)}$.*

*Proof.* By Fact 4, $P_d$ is a degree-$d$ univariate polynomial with $d$ non-zero coefficients of magnitude at most $d$. We can view the polynomial $p_{f,d}$ as the polynomial $P'_d(T_1, \ldots, T_t) := P_d(T_1 + \cdots + T_t)$ over variables $T_i \in \{0, 1\}$. Expanding out $P'_d$ gives us at most $dt^d$ monomials with coefficients of magnitude at most $d$. Now each monomial of $P'_d$ is a product of $T_i$'s, so applying Facts 2 and 1 we have that $\|p_{f,d}\|_1 \leq t^{O(d)}$. ■

The next section will show that the polynomial $p_{f,d}$ (for $d = \Theta(\log 1/\epsilon)$) is in fact a good approximation for random DNF formulas. As a warm-up, we will show the simple case of read-once DNF formulas.

### 3.3.1 A Simple Case: Read-Once DNF Formulas

For read-once DNF formulas, the probability that a term is satisfied is independent of whether or not any of the other terms are satisfied, and thus $f$ is unlikely to have many

terms satisfied simultaneously.

**Lemma 2.** *Let $f = T_1 \vee \cdots \vee T_t$ be a read-once DNF formula of size $t$ such that $\Pr[f] < 1-\epsilon$. Then the probability over the uniform distribution on $\{0,1\}^n$ that some set of $j > e \ln 1/\epsilon$ terms is satisfied is at most $\left(\frac{e \ln 1/\epsilon}{j}\right)^j$.*

*Proof.* For any assignment $x$ to the variables of $f$, let $y_f(x)$ be the number terms satisfied in $f$. By linearity of expectation, we have that $\mathbf{E}_x[y_f(x)] = \sum_{i=1}^t \Pr[T_i = 1]$. Note that $\Pr[\neg f] = \prod_{i=1}^t (1 - \Pr[T_i])$, which is maximized when each $\Pr[T_i] = \mathbf{E}[y_f]/t$, hence $\Pr[\neg f] \leq (1 - \mathbf{E}[y_f]/t)^t \leq e^{-\mathbf{E}[y_f]}$. Thus we may assume that $\mathbf{E}[y_f] \leq \ln 1/\epsilon$, otherwise $\Pr[f] \geq 1 - \epsilon$.

Assuming $\mathbf{E}[y_f] \leq \ln 1/\epsilon$, we now bound the probability that some set of $j > e \ln 1/\epsilon$ terms of $f$ is satisfied. Since all the terms are disjoint, this probability is $\sum_{S \subseteq [t], |S|=j} \prod_{i \in S} \Pr[T_i]$, and the arithmetic-geometric mean inequality gives that this is maximized when every $\Pr[T_i] = \mathbf{E}[y_f]/t$. Then the probability of satisfying some set of $j$ terms is at most:

$$\binom{t}{j} \left(\frac{\ln 1/\epsilon}{t}\right)^j \leq \left(\frac{et}{j}\right)^j \left(\frac{\ln 1/\epsilon}{t}\right)^j = \left(\frac{e \ln 1/\epsilon}{j}\right)^j,$$

which concludes the proof of the lemma. ■

The following lemma shows that we can set $d$ to be fairly small, $\Theta(\log 1/\epsilon)$, and the polynomial $p_{f,d}$ will be a good approximation for any DNF formula $f$, as long as $f$ is unlikely to have many terms satisfied simultaneously.

**Lemma 3.** *Let $f$ be any $t$-term DNF formula, and let $d = \lceil 4e^3 \ln 1/\epsilon \rceil$. If*

$$\Pr[y_f(x) = j] \leq \left(\frac{e \ln 1/\epsilon}{j}\right)^j$$

*for every $d \leq j \leq t$, then the polynomial $p_{f,d}$ satisfies $\mathbf{E}[(f - p_{f,d})^2] \leq \epsilon$.*

*Proof.* We condition on the values of $y_f(x)$, controlling the magnitude of $p_{f,d}$ by the unlikelihood of $y_f$ being large. By Fact 4, $p_{f,d}(x)$ will output 0 if $x$ does not satisfy $f$, $p_{f,d}(x)$

will output 1 if $y_f(x) \in [d]$, and $|p_{f,d}(x)| < \binom{y_f}{d}$ for $y_f(x) \in [t] \setminus [d]$. Hence:

$$\begin{aligned} \|f - p_{f,d}\|^2 &< \sum_{j=d+1}^{t} \binom{j}{d}^2 \left(\frac{e \ln 1/\epsilon}{j}\right)^j \\ &< \sum_{j=d+1}^{t} 2^{2j} \left(\frac{e \ln 1/\epsilon}{4e^3 \ln 1/\epsilon}\right)^j \\ &< \epsilon \sum_{j=d+1}^{t} \frac{1}{e^j} < \epsilon. \end{aligned}$$

■

Combining Lemmas 1, 2, and 3 gives us Mansour's conjecture for read-once DNF formulas.

**Theorem 6.** *Let $f$ be any read-once DNF formula with $t$ terms. Then there is a polynomial $p_{f,d}$ with $\|p_{f,d}\|_1 \leq t^{O(\log 1/\epsilon)}$ and $\mathbf{E}[(f - p_{f,d})^2] \leq \epsilon$ for all $\epsilon > 0$.*

## 3.4 Mansour's Conjecture for Random DNF Formulas

In this section, we establish various properties of random DNF formulas and use these properties to show that for almost all $f$, Mansour's conjecture holds. Roughly speaking, we will show that a random DNF formula behaves like a read-once DNF formula, in that any "large" set of terms is unlikely to be satisfied by a random assignment. This notion is formalized in Lemma 5. For such DNF formulas, we may use the construction from Section 3.3 to obtain a good approximating polynomial for $f$ with small spectral norm (Theorem 7).

We introduced the notion of a random DNF in Chapter 2; for convenience, we repeat the formal definition here. Let $\mathcal{D}_n^t$ be the probability distribution over $t$-term DNF formulas induced by the following process: each term is independently and uniformly chosen at random from all $t\binom{n}{\log t}$ possible terms of size exactly $\log t$ over $\{x_1, \ldots, x_n\}$. For convenience, we assume that $\log t$ is an integer throughout our discussion, although the general case is easily handled by taking terms of length $\lfloor \log t \rfloor$.

If $t$ grows very slowly relative to $n$, say $t(n) = n^{o(1)}$, then with high probability $(1 - n^{\Omega(1)})$ a random $f$ drawn from $\mathcal{D}_n^t$ will be a read-once DNF formula, in which case the results of

Section 3.3.1 hold. Therefore, throughout the rest of this section we will assume that $t = n^{\Theta(1)}$.

To prove Lemma 5, we require two Lemmas; these have a similar flavor to lemmas we will prove in Chapter 4. Lemma 4 shows that with high probability the terms of a random DNF formula are close to being disjoint, and thus cover close to $j \log t$ variables.

**Lemma 4.** *With probability at least* $1 - t^j e^{j \log t} (j \log t)^{\log t} / n^{\log t}$ *over the random draw of* $f$ *from* $\mathcal{D}_n^t$*, at least* $j \log t - (\log t)/4$ *variables occur in every set of* $j$ *distinct terms of* $f$*. The failure probability is at most* $1/n^{\Omega(\log t)}$ *for any* $j < c \log n$*, for some constant* $c$*.*

*Proof.* Let $k := \log t$. Fix a set of $j$ terms, and let $v \leq jk$ be the number of distinct variables (negated or not) that occur in these terms. We will bound the probability that $v > w := jk - k/4$. Consider any particular fixed set of $w$ variables. The probability that none of the $j$ terms include any variable outside of the $w$ variables is precisely $\left(\binom{w}{k}/\binom{n}{k}\right)^j$. Thus, the probability that $v \leq w$ is by the union bound:

$$\binom{n}{w}\left(\frac{\binom{w}{k}}{\binom{n}{k}}\right)^j < \left(\frac{en}{w}\right)^w \left(\frac{w}{n}\right)^{jk} = \frac{e^{jk-k/4}(jk-k/4)^{k/4}}{n^{k/4}} < \frac{e^{jk}(jk)^{k/4}}{n^{k/4}}.$$

Taking a union bound over all (at most $t^j$) sets, we have that with the correct probability every set of $j$ terms contains at least $w$ distinct variables. ■

We will use the method of bounded differences (a.k.a., McDiarmid's inequality) to prove Lemma 5.

**Proposition 1** (McDiarmid's inequality)**.** *Let* $X_1, \ldots, X_m$ *be independent random variables taking values in a set* $\mathcal{X}$*, and let* $f : \mathcal{X}^m \to \mathbb{R}$ *be such that for all* $i \in [m]$*,* $|f(a) - f(a')| \leq d_i$*, whenever* $a, a' \in \mathcal{X}^m$ *differ in just the* $i$*th coordinate. Then for all* $\tau > 0$*,*

$$\Pr[f > \mathbf{E}f + \tau] \leq \exp\left(-\frac{2\tau^2}{\sum_i d_i^2}\right) \text{ and } \Pr[f < \mathbf{E}f - \tau] \leq \exp\left(-\frac{2\tau^2}{\sum_i d_i^2}\right).$$

Lemma 5 below shows that with high probability over the choice of random DNF formula, the probability that exactly $j$ terms are satisfied is close to that for the basic example we considered in Section 3.1.3: $\binom{t}{j} t^{-j}(1-1/t)^{t-j}$.

**Lemma 5.** *There exists a constant $c$ such that for any $j < c \log n$, with probability at least $1 - 1/n^{\Omega(\log t)}$ over the random draw of $f$ from $\mathcal{D}_n^t$, the probability over the uniform distribution on $\{0,1\}^n$ that an input satisfies exactly $j$ distinct terms of $f$ is at most $2\binom{t}{j}t^{-j}(1-1/t)^{t-j}$.*

*Proof.* Let $f = T_1 \vee \cdots \vee T_t$, and let $\beta := t^{-j}(1-1/t)^{t-j}$. Fix any $J \subset [t]$ of size $j$, and let $U_J$ be the probability over $x \in \{0,1\}^n$ that the terms $T_i$ for $i \in J$ are satisfied and no other terms are satisfied. We will show that $U_J < 2\beta$ with high probability; a union bound over all possible sets $J$ of size $j$ in $[t]$ gives that $U_J \leq 2\beta$ for every $J$ with high probability. Finally, a union bound over all $\binom{t}{j}$ possible sets of $j$ terms (where the probability is taken over $x$) proves the lemma.

Without loss of generality, we may assume that $J = [j]$. For any fixed $x$, we have:

$$\Pr_{f \in \mathcal{D}_n^t}[x \text{ satisfies exactly the terms in } J] = \beta,$$

and thus by linearity of expectation, we have $\mathbf{E}_{f \in \mathcal{D}_n^t}[U_J] = \beta$. Now we show that with high probability that the deviation of $U_J$ from its expected value is low.

Applying Lemma 4, we may assume that the terms $T_1, \cdots, T_j$ contain at least $j \log t - (\log t)/4$ many variables, and that $J \cup T_i$ for all $i = j+1, \cdots, t$ includes at least $(j+1)\log t - (\log t)/4$ many unique variables, while increasing the failure probability by only $1/n^{\Omega(\log t)}$. Note that conditioning on this event can change the value of $U_J$ by at most $1/n^{\Omega(\log t)} < \frac{1}{2}\beta$, so under this conditioning we have $\mathbf{E}[P_j] \geq \frac{1}{2}\beta$. Conditioning on this event, fix the terms $T_1, \cdots, T_j$. Then the terms $T_{j+1}, \cdots, T_t$ are chosen uniformly and independently from the set of all terms $T$ of length $\log t$ such that the union of the variables in $J$ and $T$ includes at least $(j+1)\log t - (\log t)/4$ unique variables. Call this set $\mathcal{X}$.

We now use McDiarmid's inequality where the random variables are the terms $T_{j+1}, \ldots, T_t$ randomly selected from $\mathcal{X}$, letting $g(T_{j+1}, \cdots, T_t) = U_J$ and $g(T_{j+1}, \cdots, T_{i-1}, T_i', T_{i+1}, \cdots, T_t) = U_J'$ for all $i = j+1, \ldots, t$. We claim that:

$$\left|U_J - U_J'\right| \leq d_i := \frac{t^{1/4}}{t^{j+1}}.$$

This is because $U_J'$ can only be larger than $U_J$ by assignments which satisfy $T_1, \cdots, T_J$ and $T_i$. Similarly, $U_J'$ can only be smaller than $U_J$ by assignments which satisfy $T_1, \cdots, T_J$ and

$T_i'$. Because $T_i$ and $T_i'$ come from $\mathcal{X}$, we know that at least $(j+1)t - (\log t)/4$ variables must be satisfied.

Thus we may apply McDiarmid's inequality with $\tau = \frac{3}{2}\beta$, which gives that $\Pr_f[U_J > 2\beta]$ is at most

$$\exp\left(\frac{-2\frac{9}{4}\beta^2}{t^{3/2}/t^{2j+2}}\right) \leq \exp\left(\frac{-9\sqrt{t}(1-1/t)^{2(t-j)}}{2}\right).$$

Combining the failure probabilities over all the $\binom{t}{j}$ possible sets, we get that with probability at least

$$\binom{t}{j}\left(\frac{1}{n^{\Omega(\log t)}} + e^{-9\sqrt{t}(1-1/t)^{2(t-j)}/2}\right) = \frac{1}{n^{\Omega(\log t)}},$$

over the random draw of $f$ from $\mathcal{D}_n^t$, $U_J \leq 2\beta$ for all $J \subseteq [t]$ of size $j$. Thus, the probability that a random input satisfies exactly some $j$ distinct terms of $f$ is at most $2\binom{t}{j}\beta$. ■

Using these properties we can now show a lemma for random DNF formulas analgous to Lemma 3.

**Lemma 6.** *Let $f$ be any DNF formula with $t = n^{O(1)}$ terms, and let $\epsilon > 0$ which satisfies $1/\epsilon = o(\log\log n)$. Then set $d = \lceil 4e^3 \ln 1/\epsilon \rceil$ and $\ell = c\log n$, where $c$ is the constant in Lemma 5. If*

$$\Pr[y_f(x) = j] \leq \left(\frac{e\ln 1/\epsilon}{j}\right)^j$$

*for every $d \leq j \leq \ell$, then the polynomial $p_{f,d}$ satisfies $\mathbf{E}[(f - p_{f,d})^2] \leq \epsilon$.*

*Proof.* We condition on the values of $y_f(x)$, controlling the magnitude of $p_{f,d}$ by the unlikelihood of $y_f$ being large. By Fact 4, $p_{f,d}(x)$ will output 0 if $x$ does not satisfy $f$, $p_{f,d}(x)$ will output 1 if $y_f(x) \in [d]$, and $|p_{f,d}(x)| < \binom{y_f}{d}$ for $y_f(x) \in [t] \setminus [d]$. Hence:

$$\begin{aligned}
\|f - p_{f,d}\|^2 &< \sum_{j=d+1}^{\ell-1}\binom{j}{d}^2\left(\frac{e\ln 1/\epsilon}{j}\right)^j + \binom{t}{d}^2 \cdot \Pr[y_f \geq \ell] \\
&< \sum_{j=d+1}^{\ell-1} 2^{2j}\left(\frac{e\ln 1/\epsilon}{4e^3\ln 1/\epsilon}\right)^j + n^{-\Omega(\log\log n)} \\
&< \epsilon\sum_{j=d+1}^{\ell-1}\frac{1}{e^j} + n^{-\Omega(\log\log n)} < \epsilon.
\end{aligned}$$

■

We can now show that Mansour's conjecture [Mansour, 1994] is true with high probability over the choice of $f$ from $\mathcal{D}_n^t$.

**Theorem 7.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a $t = n^{\Theta(1)}$-term DNF formula where each term is chosen independently from the set of all terms of length $\log t$. Then with probability at least $1 - n^{-\Omega(\log t)}$ over the choice of $f$, there exists a polynomial $p$ with $\|p\|_1 \leq t^{O(\log 1/\epsilon)}$ such that $\mathbf{E}[(p(x) - f(x))^2] \leq \epsilon$.*

*Proof.* Let $\lceil d := 4e^3 \ln(1/\epsilon) \rceil$ and $p_{f,d}$ be as defined in Section 3.3. Lemma 1 tells us that $\|p_{f,d}\|_1 \leq t^{O(\log 1/\epsilon)}$. We show that with probability at least $1 - n^{-\Omega(\log t)}$ over the random draw of $f$ from $\mathcal{D}_n^t$, $p_{f,d}$ will be a good approximator for $f$. This follows by Lemma 5; with probability at least $1 - (c\log(n) - d - 1)/n^{\Omega(\log t)} = 1 - n^{-\Omega(\log t)}$, we have $\Pr[y = j]$ for all $d < j \leq c\log(n)$. Thus for such $f$ Lemma 3 tells us that $\mathbf{E}[(f - p_{f,d})^2] \leq \epsilon$. ■

## 3.5 Mansour's Conjecture for Read-$k$ DNF Formulas

In this section, we give an $\epsilon$-approximating polynomial for any read-$k$ DNF formula and show that its spectral norm is at most $t^{O(2^{4k} \log 1/\epsilon)}$. This implies that Mansour's conjecture holds for all read-$k$ DNF formulas where $k$ is any constant.

Read-$k$ DNF formulas may not satisfy the conditions of Lemma 3, so we must change our approach. Instead of using $\sum_{i=1}^t T_i$ inside our univariate polynomial, we use a different sum, which is based on a construction from [Razborov, 2008] for representing any DNF formula. We modify this representation to exploit the fact that for read-$k$ DNF formulas, the variables in a term can not share variables with too many other terms. Unlike for read-once DNF formulas, it is not clear that the number of terms satisfied in a read-$k$ DNF formula will be extremely concentrated on a small range. We show how to modify our construction so that a concentration result does hold.

Let $f = T_1 \vee \cdots \vee T_t$ be any $t$-term read-$k$ DNF formula, and let $|T_i|$ denote the number of variables in term $T_i$. We assume that the terms are ordered from longest to shortest, *i.e.,* $|T_j| \geq |T_i|$ for all $j \leq i$. For any term $T_i$ of $f$, let $\phi_i$ be the DNF formula consisting of

those terms (at least as large as $T_i$) in $T_1, \cdots, T_{i-1}$ that overlap with $T_i$, *i.e.,*

$$\phi_i := \bigvee_{j \in \mathcal{C}_i} T_j, \text{ for } \mathcal{C}_i = \{j < i \mid T_j \cap T_i \neq \emptyset\}.$$

We define $A_i := T_i \wedge \neg\phi_i$ and $z_f := \sum_{i=1}^{t} A_i$. The function $z_f : \{0,1\}^n \to \{0, \ldots, t\}$ outputs the number of disjoint terms of $f$ satisfied by $x$ (greedily starting from $T_1$). Note that if $f$ is a read-once DNF formula, then $z_f = y_f$.

Observe that each $A_i$ can be represented by the polynomial $T_i \cdot \prod_{j \in C_i}(1 - T_j)$ (and so $z_f$ can be represented by a polynomial), and that $||(1 - T_j)||_1 \leq 2$ for all $j$. As $f$ is a read-$k$ DNF formula, each $\phi_i$ has at most $k|T_i|$ terms, and $A_i$ has small spectral norm:

**Fact 5.** *Let $f = T_1 \vee \cdots \vee T_t$ be a $t$-term read-$k$ DNF formula. Then each $A_i$ has a polynomial representation, and $||A_i||_1 \leq 2^{k|T_i|}$.*

As we did in Section 3.3, we can construct a polynomial $q_{f,d} : \{0,1\}^n \to \mathbb{R}$ defined as $q_{f,d} := P_d \circ z_f$ for any $t$-term read-$k$ DNF formula $f$. The following lemma shows that $q_{f,d}$ has small spectral norm.

**Lemma 7.** *Let $f$ be a $t$-term read-$k$ DNF formula with terms of length at most $w$. Then $\|q_{f,d}\|_1 \leq 2^{O(d(\log t + kw))}$.*

*Proof.* By Fact 4, $P_d$ is a degree-$d$ univariate polynomial with $d$ terms and coefficients of magnitude at most $d$. We can view the polynomial $q_{f,d}$ as the polynomial $P'_d(A_1, \ldots, A_t) := P_d(A_1 + \cdots + A_t)$ over variables $A_i \in \{0,1\}$. Expanding out (but not recombining) $P'_d$ gives us at most $dt^d$ monomials of degree $d$ (over variables $A_i$) with coefficients of magnitude at most $d$.

We can now apply Facts 5 and 1 to bound the spectral norm of $q_{f,d}$. Since $P'_d$ has at most $dt^d$ monomials each of degree $d$ (over $A_i$), fnd each $A_i$ satisfies $||A_i||_1 \leq 2^{kw}$, we have that $||q_{f,d}||_1 \leq 2^{dkw} dt^d = 2^{O(d(\log t + kw))}$. ■

We will show that Mansour's conjecture holds for read-$k$ DNF formulas by showing that $z_f = \sum_{i=1}^{t} A_i$ behaves much like $y_f = \sum_{i=1}^{t} T_i$ would if $f$ were a read-once DNF formula, and thus we can use our polynomial $P_d$ (Equation 3.1) to approximate $f$.

One crucial property of our construction is that only disjoint sets of terms can contribute to $z_f$.

**Claim 1.** *Let $T_1 \vee \cdots \vee T_t$ be a $t$-term DNF formula. Then for any $S \subseteq [t]$, $\Pr[\wedge_{i\in S} A_i] \leq \prod_{i\in S} \Pr[T_i]$.*

*Proof.* If there is a pair $j, k \in S$ such that $T_j \cap T_k \neq \emptyset$ for some $j < k$, then $\phi_k$ contains $T_j$ and both $T_j \wedge \neg\phi_j$ and $T_k \wedge \neg\phi_k$ cannot be satisfied simultaneously, so $\Pr[\wedge_{i\in S} A_i] = 0$. If no such pair exists, then all the terms indexed by $S$ are disjoint. Now

$$\Pr[\wedge_{i\in S} A_i] \leq \Pr[\wedge_{i\in S} T_i] = \prod_{i\in S} \Pr[T_i].$$

■

The following claim tells us that, for a read-$k$ *monotone* DNF formula, the probability of satisfying $A_i$ compared to that of satisfying $T_i$ is only smaller by a constant (for constant $k$). While the claim only holds for monotone read-$k$ DNF formulas, it will be useful for analyzing the behavior of $\sum_{i=1}^{t} A_i$ for non-monotone formulas as well.

**Claim 2.** *Let $T_1 \vee \cdots \vee T_t$ be a $t$-term monotone read-$k$ DNF formula. Then $\Pr[A_i] \geq 2^{-4k} \Pr[T_i]$.*

*Proof.* Let $I$ be the set of indices of the terms in $\phi_i$. For each $T_j \in \phi_i$, let $T'_j$ be $T_j$ with all the variables of $T_i$ set to 1, and let $\phi'_i = \vee_{\{j:T_j\in\phi_i\}} T'_j$. (For example, if $T_i = x_1x_2x_3$ and $T_j = x_2x_4x_5$ is a term of $\phi_i$, then $\phi'_i$ contains the term $T'_j = x_4x_5$.) Observe that $\Pr[A_i] = \Pr[T_i \wedge \neg\phi_i] = \Pr[T_i \wedge \neg\phi'_i] = \Pr[T_i]\Pr[\neg\phi'_i]$. Thus it suffices to show that $\Pr[\neg\phi'_i] \geq 2^{-4k}$.

Let $a_j$ be the number of variables in $T_j \cap T_i$. By the definition of $\phi_i$, $1 \leq a_j \leq |T_i| - 1$, and note that $\Pr[T'_j] = 2^{a_j - |T_j|}$. Applying the Four Functions Theorem (Theorem 1 from Chapter 2), we obtain:

$$\Pr[\neg\phi'_i] \geq \prod_{j\in I} \Pr[\neg T'_j] = \prod_{j\in I} (1 - 2^{a_j - |T_j|}) \geq \prod_{j\in I} (1 - 2^{a_j - |T_i|}).$$

We partition $I$ into two sets: $J = \{j : a_j \leq |T_i|/2\}$ and $J' = \{j : a_j > |T_i|/2\}$. (Assume that $|T_i| \geq 4$ or else we are done, because there can be at most $4k$ terms.) As $\phi_i$ is a read-$k$ DNF formula, we have that $\sum_{j\in I} a_j \leq k|T_i|$, and thus $|J'| \leq 2k$, and $|J| \leq k|T_i|$.

We will lower bound the products over each set of indices separately. For those terms in $J$, we have that $\Pr[T'_j] \leq 2^{-|T_i|/2}$, hence

$$\prod_{j \in J}(1 - \Pr[T'_j]) \geq \prod_{j \in J}(1 - 2^{-|T_i|/2}) \geq (1 - 2^{-|T_i|/2})^{k|T_i|} \geq 2^{-2k}.$$

For those terms $T_j$, $j \in J'$ (which share many variables with $T_i$), we use the facts that each $\Pr[T'_j] \leq 1/2$ and that there are at most $2k$ such terms, so that

$$\prod_{j \in J'}(1 - \Pr[T'_j]) \geq 2^{-2k}.$$

Taking the product over the set $J \cup J'$ completes the proof of the claim. ■

We remark that we used the fact that $\phi_i$ only contains terms that are at least as long as $T_i$ to get a much stronger lower bound than that implied by Lemma 3 of [Hancock and Mansour, 1991b].

As in the read once-case, we will prove (in Lemma 9) that for any read-$k$ DNF formula $f$, if $\sum_{i=1}^{t} \Pr[T_i]$ is large then $f$ is biased towards one. To do so we will prove this for monotone read-$k$ DNF formulas and then use the following lemma to obtain the general case. Lemma 8 was communicated to us by Omid Etesami and James Cook [Etesami and Cook, 2010].

**Lemma 8.** *Let $f = T_1 \vee \ldots \vee T_t$ be a $t$-term read-$k$ DNF formula, and let $f' = T'_1 \vee \ldots \vee T'_t$ be the monotone formula obtained from $f$ by replacing all the negative literals by their positive counterparts. Then* $\Pr[f'] \leq \Pr[f]$.

*Proof.* For each $0 \leq i \leq n$, define $f^{(i)}$ as the DNF formula obtained from $f$ when replacing each occurrence of $\neg x_j$ by $x_j$ for all $1 \leq j \leq i$. In particular, $f^{(0)} = f$ and $f^{(n)} = f'$. Let $f^{(i-1)} = (g_{x_i} \wedge x_i) \vee (g_{\neg x_i} \wedge \neg x_i) \vee g_\emptyset$ where $g_{x_i} \wedge x_i$ is the OR of all terms from $f^{(i-1)}$ that have the literal $x_i$, $g_{\neg x_i} \wedge \neg x_i$ is the OR of all terms that have the literal $\neg x_i$, and $g_\emptyset$ is the OR of all terms that neither contain $x_i$ nor contain $\neg x_i$. Note that $f^{(i)} = ((g_{x_i} \vee g_{\neg x_i}) \wedge x_i) \vee g_\emptyset$. Thus

$$\Pr\left[f^{(i-1)}\right] = \frac{1}{2}\Pr[g_{x_i} \wedge \neg g_\emptyset] + \frac{1}{2}\Pr[g_{\neg x_i} \wedge \neg g_\emptyset] + \Pr[g_\emptyset],$$

and

$$\Pr\left[f^{(i)}\right] = \frac{1}{2}\Pr[(g_{x_i} \vee g_{\neg x_i}) \wedge \neg g_\emptyset] + \Pr[g_\emptyset].$$

A union bound on the events $(g_{x_i} \wedge \neg g_\emptyset)$ and $(g_{\neg x_i} \wedge \neg g_\emptyset)$ tells us that $\Pr[f^{(i-1)}] \geq \Pr[f^{(i)}]$, and thus $\Pr[f^{(0)}] \geq \Pr[f^{(n)}]$. ■

Finally, we will prove Lemma 9. Using Lemma 9 with Claim 1, we can prove a lemma analogous to Lemma 2 by a case analysis of $\sum_{i=1}^{t} \Pr[T_i]$; either it is large and $f$ must be biased toward one, or it is small so $z_f$ is usually small.

**Lemma 9.** *Let $f$ be a $t$-term read-$k$ DNF formula. Then,*

$$\sum_{i=1}^{t} \Pr[T_i] \leq 2^{4k} \ln\left(\frac{1}{\Pr[\neg f]}\right).$$

*Proof.* First, let us consider the case when $f$ is monotone. Let $\rho_i$ be those terms among $T_1, \ldots, T_{i-1}$ that are not present in $\phi_i$. We can upper-bound $\Pr[\neg f]$ by:

$$\begin{aligned}
\Pr[\neg f] &= \prod_{i=1}^{t} (1 - \Pr[T_i \mid \neg\phi_i \wedge \neg\rho_i]) \\
&\leq \prod_{i=1}^{t} (1 - \Pr[T_i \wedge \neg\phi_i \mid \neg\rho_i]) = \prod_{i=1}^{t} (1 - \Pr[T_i \mid \neg\rho_i] \Pr[\neg\phi_i \mid T_i \wedge \neg\rho_i]) \\
&\leq \prod_{i=1}^{t} (1 - \Pr[T_i] \Pr[\neg\phi_i \mid T_i]) = \prod_{i=1}^{t} (1 - \Pr[A_i]).
\end{aligned}$$

The first inequality comes from $\Pr[A \mid B \wedge C] \geq \Pr[A \wedge B \mid C]$ for any $A$, $B$, and $C$. The last inequality holds because $\Pr[T_i \mid \neg\rho_i] = \Pr[T_i]$ (by the mutual independence of $T_i$ and $\rho_i$) and $\Pr[\neg\phi_i \mid T_i] \leq \Pr[\neg\phi_i \mid T_i \wedge \neg\rho_i]$. The last fact may be obtained by applying the Four Functions Theorem to $\neg\phi_i$ and $\neg\rho_i$ under the product distribution induced by setting all the variables of $T_i$ to be true.

We apply Claim 2 to obtain $\Pr[\neg f] \leq \prod_{i=1}^{t}(1 - \Pr[T_i] 2^{-4k})$, and the arithmetic-geometric mean inequality shows that our upper-bound on $\Pr[\neg f]$ is maximized when all the $\Pr[T_i]$ are equal, hence:

$$\Pr[\neg f] \leq \left(1 - 2^{-4k} \frac{\sum_{i=1}^{t} \Pr[T_i]}{t}\right)^t \leq \exp\left(-2^{-4k} \sum_{i=1}^{t} \Pr[T_i]\right).$$

Solving for $\sum_{i=1}^{t} \Pr[T_i]$ yields the lemma for monotone $f$.

Now let $f$ be a non-monotone DNF formula, and let $f'$ be the monotonized version of $f$. Then by Lemma 8 we have:

$$\sum_{i=1}^{t} \Pr[T_i] = \sum_{i=1}^{t} \Pr[T'_i] \leq 2^{4k} \ln\left(\frac{1}{\Pr[\neg f']}\right) \leq 2^{4k} \ln\left(\frac{1}{\Pr[\neg f]}\right),$$

as was to be shown. ■

**Lemma 10.** *Let $f = T_1 \vee \cdots \vee T_t$ be a read-$k$ DNF formula of size $t$ such that $\Pr[f] < 1 - \epsilon$. Then the probability over the uniform distribution on $\{0,1\}^n$ that $z_f \geq j$ (for any $j > 2^{4k} e \ln(1/\epsilon)$) is at most $\left(\frac{2^{4k} e \ln(1/\epsilon)}{j}\right)^j$.*

*Proof.* By Lemma 9, $T_A := \sum_{i=1}^{t} \Pr[T_i] < 2^{4k} \ln(1/\epsilon)$. The probability that some set of $j$ $A_i$'s is satisfied is at most $\sum_{S \subseteq [t], |S|=j} \Pr[\wedge_{i \in S} A_i]$. Applying Claim 1, we have:

$$\sum_{S \subseteq [t], |S|=j} \Pr[\wedge_{i \in S} A_i] \leq \sum_{S \subseteq [t], |S|=j} \prod_{i \in S} \Pr[T_i].$$

The arithmetic-geometric mean inequality shows that this quantity is maximized when all $\Pr[T_i]$ are equal, hence:

$$\sum_{S \subseteq [t], |S|=j} \prod_{i \in S} \Pr[T_i] \leq \binom{t}{j} \left(\frac{T_A}{t}\right)^j \leq \left(\frac{e T_A}{j}\right)^j \leq \left(\frac{2^{4k} e \ln 1/\epsilon}{j}\right)^j$$

■

We can now show that Mansour's conjecture holds for read-$k$ DNF formulas with any constant $k$.

**Theorem 8.** *Let $f : \{0,1\}^n \to \{0,1\}$ be any read-$k$ DNF formula with $t$ terms. Then there is a polynomial $q_{f,d}$ with $\|q_{f,d}\|_1 = t^{O(2^{4k} \log 1/\epsilon)}$ and $\mathbf{E}[(f - q_{f,d})^2] \leq \epsilon$ for all $\epsilon > 0$.*

*Proof.* If $\Pr[f = 1] > 1 - \epsilon$, the constant 1 is a suitable polynomial. Let $g$ be the DNF formula $f$ after dropping terms of length greater than $w := \log(2t/\epsilon)$. (This only changes the probability by $\epsilon/2$.) Let $d := \lceil 4e^3 2^{4k} \ln(2/\epsilon) \rceil$ and $q_{g,d}$ be as defined at the beginning of Section 3.5. Lemma 7 tells us that $\|q_{g,d}\|_1 \leq t^{O(2^{4k} \log 1/\epsilon)}$, and Lemma 10 combined with Lemma 3 tells us that $\mathbf{E}[(g - q_{g,d})^2] \leq \epsilon/2$. ■

## 3.6 Pseudorandomness

De *et al.* [De *et al.*, 2009] recently improved long-standing pseudorandom generators against DNF formulas.

**Definition 4.** *A probability distribution $X$ over $\{0,1\}^n$ $\epsilon$-fools a real function $f : \{0,1\}^n \to \mathbb{R}$ if*

$$|\mathbf{E}[f(X)] - \mathbf{E}[f(U_n)]| \leq \epsilon.$$

*Here $U_n$ denotes the uniform distribution over the Boolean hypercube. If $\mathcal{C}$ is a class of functions, then we say that $X$ $\epsilon$-fools $\mathcal{C}$ if $X$ $\epsilon$-fools every function $f \in \mathcal{C}$.*

*We say a probability distribution $X$ over $\{0,1\}^n$ is* $\epsilon$-biased *if it $\epsilon$-fools the character function $\chi_S$ for every $S \subseteq [n]$.*

De *et al.* observed that a result of Bazzi [Bazzi, 2007] implied a pseudorandom generator that $\epsilon$-fools $t$-term DNF formulas over $n$ variables with seed length $O(\log n \cdot \log^2(t/\beta))$, which already improves the long-standing upper bound of $O(\log^4(tn/\epsilon))$ of Luby *et al.* [Luby *et al.*, 1993]. They go on to show a pseudorandom generator with seed length $O(\log n + \log^2(t/\epsilon) \log\log(t/\epsilon))$.

They prove that a sufficient condition for a function $f$ to be $\epsilon$-fooled by an $\epsilon$-biased distribution is that the function be "sandwiched" between two bounded real-valued functions whose Fourier transform has small $\ell_1$ norm:

**Lemma 11** (Sandwich Bound [De *et al.*, 2009])**.** *Suppose $f, f_\ell, f_u : \{0,1\}^n \to \mathbb{R}$ are three functions such that for every $x \in \{0,1\}^n$, $f_\ell(x) \leq f(x) \leq f_u(x)$, $\mathbf{E}[f_u(U_n)] - \mathbf{E}[f(U_n)] \leq \epsilon$, and $\mathbf{E}[f(U_n)] - \mathbf{E}[f_\ell(U_n)] \leq \epsilon$. Let $L = \max(\|f_\ell\|_1^{\neq\emptyset}, \|f_u\|_1^{\neq\emptyset})$. Then any $\beta$-biased probability distribution $(\epsilon + \beta L)$-fools $f$.*

Naor and Naor [Naor and Naor, 1993] prove that an $\epsilon$-biased distribution over $n$ bits can be sampled using a seed of $O(\log(n/\epsilon))$ bits. Using our construction from Section 3.4, we show that random DNF formulas are $\epsilon$-fooled by a pseudorandom generator with seed length $O(\log n + \log(t)\log(1/\epsilon))$:

**Theorem 9.** *Let $f = T_1 \vee \cdots \vee T_t$ be a random DNF formula chosen from $\mathcal{D}_n^t$ for $t = n^{\Theta(1)}$. For $1 \leq d \leq t$, with probability $1 - 1/n^{\Omega(\log t)}$ over the choice of $f$, $\beta$-biased distributions $O(2^{-\Omega(d)} + \beta t^d)$-fool $f$. In particular, we can $\epsilon$-fool most $f \in \mathcal{D}_n^t$ by a $t^{-O(\log(1/\epsilon)}$-biased distribution.*

*Proof.* Let $d^+$ be the first odd integer greater than $d$, and let $d^-$ be the first even integer greater than $d$. Let $f_u = p_{f,d^+}$ and $f_\ell = p_{f,d^-}$ (where $p_{f,d}$ is defined as in Section 3.3).

By Lemma 1, the $\ell_1$-norms of $f_u$ and $f_\ell$ are $t^{O(d)}$. By Fact 4, we know that $P_{d^+}(y) = \binom{y-1}{d} + 1 > 1$ and $P_{d^-}(y) = -\binom{y-1}{d} + 1 \leq 0$ for $y \in [t] \setminus [d]$, hence:

$$\mathbf{E}[f_u(U_n)] - \mathbf{E}[f(U_n)] = \sum_{j=d+1}^{t} \left( \binom{j-1}{d} + 1 - 1 \right) \Pr[y_f = j],$$

which with probability $1 - 1/n^{\Omega(\log t)}$ over the choice of $f$ is at most $2^{-\Omega(d)}$ by the analysis in Lemma 3. The same analysis applies for $f_\ell$, thus applying Lemma 11 gives us the theorem. ■

De *et al.* match our bound for random DNF formulas for the special case of read-once DNF formulas. Using our construction from Section 3.5 and a similar proof as the one above, we can show that read-$k$ formulas are $\epsilon$-fooled by a pseudorandom generator with seed length $O(\log n + \log(t)\log(1/\epsilon))$.

**Theorem 10.** *Let $f = T_1 \vee \cdots \vee T_t$ be a read-$k$ DNF formula for constant $k$. For $1 \leq d \leq t$, $\beta$-biased distributions $O(2^{-\Omega(d)} + \beta t^d)$-fool $f$. In particular, we can $\epsilon$-fool read-$k$ DNF formulas by a $t^{-O(\log(1/\epsilon))}$-biased distribution.*

## 3.7 Discussion

**On the relationship between Mansour's Conjecture and the Entropy-Influence Conjecture.** As a final note, we would like to make a remark on the relationship between Mansour's conjecture and the entropy-influence conjecture. The *spectral entropy* of a function is defined to be $E(f) := \sum_S -\hat{f}(S)^2 \log(\hat{f}(S)^2)$ and the *total influence* to be $I(f) := \sum_S |S| \hat{f}(S)^2$. The *entropy-influence* conjecture is that $E(f) = O(I(f))$ [Friedgut and Kalai, 1996].[1] Boppana showed that the total influence of $t$-term DNF formulas is $O(\log t)$ [Boppana, 1997]. From this it follows that Mansour's conjecture (actually, $t^{O(1/\epsilon)}$ concentration) is implied by the entropy-influence conjecture. Thus, proving that DNF can be $\epsilon$-approximated (for constant $\epsilon$) by *poly*$(n)$-sparse polynomials is necessary for proving the entropy-influence conjecture. It can also be shown that for $n^{O(1)}$-size DNF formulas, Mansour's conjecture implies that their spectral entropy is at most $O(\log n)$.

[1] `http://terrytao.wordpress.com/2007/08/16/gil-kalai-the-entropyinfluence-conjecture/`

See Appendix A for a rigorous treatment of the previous statements.

# Chapter 4

# Learning Random Monotone DNF

We next apply average-case analysis to the problem of learning DNF formulas from random examples.

## 4.1 Introduction

### 4.1.1 Motivation and Background

In this chapter, we address the tantalizing question of learning *monotone* DNF from random examples, which has been posed as a goal by many authors (see e.g. [Jackson, 1997a; Jackson and Tamon, 1997; Blum *et al.*, 1998; Blum, 2003b; Servedio, 2004b]). Besides being a natural restriction of the uniform distribution DNF learning problem, this problem is interesting because several impediments to learning general DNF under uniform – known lower bounds for Statistical Query based algorithms [Blum *et al.*, 1994], the apparent hardness of learning the subclass of $\log(n)$-juntas [Blum, 2003a] – do not apply in the monotone case. This chapter solves a natural average-case version of this problem using previously unknown Fourier properties of monotone functions.

### 4.1.2 Previous Work

Many partial results have been obtained on learning monotone DNF under the uniform distribution. Verbeurgt [Verbeurgt, 1990] gave an $n^{O(\log n)}$-time uniform distribution algorithm for learning any poly$(n)$-term DNF, monotone or not. Several authors [Kučera *et al.*, 1994;

Sakai and Maruoka, 2000; Bshouty and Tamon, 1996] have given results on learning monotone $t$-term DNF for larger and larger values of $t$; most recently, [Servedio, 2004b] gave a uniform distribution algorithm that learns any $2^{O(\sqrt{\log n})}$-term monotone DNF to any constant accuracy $\epsilon = \Theta(1)$ in poly$(n)$ time. O'Donnell and Servedio [O'Donnell and Servedio, 2006] have recently shown that poly$(n)$-leaf *decision trees* that compute monotone functions (a subclass of poly$(n)$-term monotone DNF) can be learned to any constant accuracy under uniform in poly$(n)$ time. Various other problems related to learning different types of monotone functions under uniform have also been studied, see e.g. [Kearns *et al.*, 1994a; Blum *et al.*, 1998; Verbeurgt, 1998; Hancock and Mansour, 1991a; Amano and Maruoka, 2002].

Aizenstein and Pitt [Aizenstein and Pitt, 1995] first proposed a model of random DNF formulas and gave an exact learning algorithm that learns random DNFs generated in this way. As noted in [Aizenstein and Pitt, 1995] and [Jackson and Servedio, 2006], this model admits a trivial learning algorithm in the uniform PAC setting. Jackson and Servedio [Jackson and Servedio, 2005a] gave a uniform distribution algorithm that learns log-depth decision trees on average in a natural random model. Previous work on average-case uniform PAC DNF learning, also by Jackson and Servedio, is described below.

### 4.1.3 Our Results

The main result of this chapter is a polynomial-time algorithm that can learn random poly$(n)$-term monotone DNF drawn from a particular distribution. (We give a full description of the exact probability distribution defining our random DNFs in Section 4.4; briefly, the reader should think of our random $t$-term monotone DNFs as being obtained by independently drawing $t$ monotone conjunctions uniformly from the set of all conjunctions of length $\log_2 t$ over variables $x_1, \ldots, x_n$. Although many other distributions could be considered, this seems a natural starting point. Some justification for the choice of term length is given in Sections 4.4 and 4.6.)

**Theorem 11. [Informally]** *Let $t(n)$ be any function such that $t(n) \leq$ poly$(n)$, and let $c > 0$ be any fixed constant. Then random monotone $t(n)$-term DNFs are PAC learnable (with failure probability $\delta = n^{-c}$) to accuracy $\epsilon$ in poly$(n, 1/\epsilon)$ time under the uniform*

*distribution. The algorithm outputs a monotone DNF as its hypothesis.*

In independent and concurrent work, Sellie [Sellie, 2008a] has given an alternate proof of this theorem using different techniques. Later, she generalized the result to non-monotone DNF formulas as well [Sellie, 2009].

### 4.1.4 Our Technique

Jackson and Servedio [Jackson and Servedio, 2006] showed that for any $\gamma > 0$, a result similar to Theorem 11 holds for random $t$-term monotone DNF with $t \leq n^{2-\gamma}$. The main open problem stated in [Jackson and Servedio, 2006] was to prove Theorem 11. Our work solves this problem by using the previous algorithm to handle $t \leq n^{3/2}$, developing new Fourier lemmas for monotone DNF, and using these lemmas together with more general versions of techniques from [Jackson and Servedio, 2006] to handle $t \geq n^{3/2}$.

The crux of our strategy is to establish a connection between the term structure of certain monotone DNFs and their low-order Fourier coefficients. There is an extensive body of research on Fourier properties of monotone Boolean functions [Bshouty and Tamon, 1996; Mossel and O'Donnell, 2003a; Blum *et al.*, 1998], polynomial-size DNF [Jackson, 1997a; Mansour, 1994], and related classes such as constant-depth circuits and decision trees [Linial *et al.*, 1993; Kushilevitz and Mansour, 1993a; Jackson *et al.*, 2002; O'Donnell and Servedio, 2006]. These results typically establish that *every* function in the class has a Fourier spectrum with certain properties; unfortunately, the Fourier properties that have been obtainable to date for general statements of this sort have not been sufficient to yield polynomial-time learning algorithms.

We take a different approach by defining a natural distribution over monotone DNF and proving Fourier properties for the DNF drawn from this distribution. To this end, we carefully define a set of conditions and show in Section 4.4 that the random DNFs satisfy these conditions with high probability. We prove in Sections 4.2 and 4.3 that *if a monotone DNF $f$ satisfies these conditions then the structure of the terms of $f$ will be reflected in the low-order Fourier coefficients of $f$.* In [Jackson and Servedio, 2006], the degree two Fourier coefficients were shown to reveal the structure of the terms for certain (including random) monotone DNFs having at most $n^{2-\gamma}$ terms. In this work we develop new lemmas about the

Fourier coefficients of more general monotone DNF, and use these new lemmas to establish a connection between term structure and constant degree Fourier coefficients for monotone DNFs with any polynomial number of terms. Roughly speaking, this connection holds for monotone DNF that satisfy the following conditions:

- each term has a reasonably large fraction of assignments which satisfy it and no other term;
- for each small tuple of distinct terms, only a small fraction of assignments simultaneously satisfy all terms in the tuple; and
- for each small tuple of variables, only a small number of terms contains the entire tuple.

The "small" tuples referred to above should be thought of as tuples of constant size. The constant degree coefficients capture the structure of the terms in the following sense: tuples of variables that all co-occur in some term will have a large magnitude Fourier coefficient, and tuples of variables that do not all co-occur in some term will have a small magnitude Fourier coefficient (even if subsets of the tuple do co-occur in some terms). We show this in Section 4.2.

Next we show a reconstruction procedure for obtaining the monotone DNF from tuple-wise co-occurrence information. Given a hypergraph with a vertex for each variable, the procedure turns each co-occurrence into a hyperedge, and then searches for all hypercliques of size corresponding to the term length. The hypercliques that are found correspond to the terms of the monotone DNF hypothesis that the algorithm constructs. This procedure is described in Section 4.3; we show that it succeeds in constructing a high-accuracy hypothesis if the monotone DNF $f$ satisfies a few additional conditions. This is a significant generalization of a reconstruction procedure from [Jackson and Servedio, 2006] that was based on finding cliques in a graph (in the $n^{2-\gamma}$-term DNF setting, the algorithm deals only with co-occurrences of pairs of variables so it is sufficient to consider only ordinary graphs rather than hypergraphs).

The ingredients described so far thus give us an efficient algorithm to learn any monotone DNF that satisfies all of the required conditions. Finally, we show that random monotone

DNF satisfy all the required conditions with high probability. We do this in Section 4.4 via a fairly delicate probabilistic argument. Section 4.5 combines the above ingredients to prove Theorem 11. We close the chapter by showing that our technique lets us easily recapture the result of [Hancock and Mansour, 1991a] that read-$k$ monotone DNF are learnable in polynomial time under the uniform distribution.

## 4.2 Fourier Coefficients and the Term Structure of Monotone DNF

Throughout Section 4.2 let $f(x_1, \ldots, x_n)$ be a monotone DNF and let $S \subseteq \{1, \ldots, n\}$ be a fixed subset of variables. We write $s$ to denote $|S|$ throughout this section. The Fourier coefficient, written $\hat{f}(S)$, measures the correlation between $f$ and the parity of the variables in $S$.

The main result of this section is Lemma 3, which shows that under suitable conditions on $f$, the value $|\hat{f}(S)|$ is "large" if and only if $f$ has a term containing all the variables of $S$. To prove this, we observe that the inputs which uniquely satisfy such a term will make a certain contribution to $\hat{f}(S)$. (In Section 4.2.1 we explain this in more detail and show how to view $\hat{f}(S)$ as a sum of contributions from inputs to $f$.) It remains then to show that the contribution from other inputs is small. The main technical novelty comes in Sections 4.2.2 and 4.2.3, where we show that all other inputs which make a contribution to $\hat{f}(S)$ must satisfy the terms of $f$ in a special way, and use this property to show that under suitable conditions on $f$, the fraction of such inputs must be small.

### 4.2.1 Rewriting $\hat{f}(S)$.

We observe that $\hat{f}(S)$ can be expressed in terms of $2^s$ conditional probabilities, each of which is the probability that $f$ is satisfied conditioned on a particular setting of the variables in $S$. That is:

$$\begin{aligned}\hat{f}(S) &\stackrel{\text{def}}{=} \mathbf{E}_{x \in U^n}\left[(-1)^{\sum_{i \in S} x_i} \cdot f(x)\right] = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{\sum_{i \in S} x_i} \cdot f(x) \\ &= \frac{1}{2^n} \sum_{U \subseteq S} (-1)^{|U|} \sum_{x \in Z_S(U)} f(x) = \frac{1}{2^s} \sum_{U \subseteq S} (-1)^{|U|} \Pr_x[f(x) = 1 \mid x \in Z_S(U)],\end{aligned}$$

where $Z_S(U)$ denotes the set of those $x \in \{0,1\}^n$ such that $x_i = 1$ for all $i \in U$ and $x_i = 0$ for all $i \in S \setminus U$. If $f$ has some term $T$ containing all the variables in $S$, then $\Pr_x[f(x) = 1 \mid x \in Z_S(S)]$ is at least as large as $\Pr_x[T$ is uniquely satisfied in $f \mid x \in Z_S(S)]$. On the other hand, if $f$ has no such term, then $\Pr_x[f(x) = 1 \mid x \in Z_S(S)]$ does not receive this contribution. We will show that this contribution is the chief determinant of the magnitude of $\hat{f}(S)$.

It is helpful to rewrite $\hat{f}(S)$ as a sum of contributions from each input $x \in \{0,1\}^n$. To this end, we decompose $f$ according to the variables of $S$. Given a subset $U \subseteq S$, we will write $g_U$ to denote the disjunction of terms in $f$ that contain every variable indexed by $U \subseteq S$ and no variable indexed by $S \setminus U$, but with the variables indexed by $U$ removed from each term. (So for example if $f = x_1x_2x_4x_6 \vee x_1x_2x_5 \vee x_1x_2x_3 \vee x_3x_5 \vee x_1x_5x_6$ and $S = \{1,2,3\}$ and $U = \{1,2\}$, then $g_U = x_4x_6 \vee x_5$.) Thus we can split $f$ into disjoint sets of terms: $f = \bigvee_{U \subseteq S}(t_U \wedge g_U)$, where $t_U$ is the term consisting of exactly the variables indexed by $U$.

Suppose we are given $U \subseteq S$ and an $x$ that belongs to $Z_S(U)$. We have that $f(x) = 1$ if and only if $g_{U'}(x)$ is true for some $U' \subseteq U$. (Note that $t_{U'}(x)$ is true for every $U' \subseteq U$ since $x$ belongs to $Z_S(U)$.) Thus we can rewrite the Fourier coefficients $\hat{f}(S)$ as follows: (Below we write $I(P)$ to denote the indicator function that takes value 1 if predicate $P$ is true and value 0 if $P$ is false.)

$$\begin{aligned}\hat{f}(S) &= \frac{1}{2^n}\sum_{U\subseteq S}(-1)^{|U|}\sum_{x\in Z_S(U)} f(x) = \sum_{U\subseteq S}(-1)^{|U|}\frac{1}{2^n}\sum_{x\in Z_S(U)} I\left(\bigvee_{U'\subseteq U} g_{U'}(x)\right) \\ &= \sum_{U\subseteq S}(-1)^{|U|}\frac{1}{2^s}\frac{1}{2^n}\sum_{x\in\{0,1\}^n} I\left(\bigvee_{U'\subseteq U} g_{U'}(x)\right) = \sum_{x\in\{0,1\}^n}\frac{1}{2^s}\frac{1}{2^n}\sum_{U\subseteq S}(-1)^{|U|} I\left(\bigvee_{U'\subseteq U} g_{U'}(x)\right).\end{aligned}$$

We can rewrite this as

$$\hat{f}(S) = \sum_{x\in\{0,1\}^n} \mathrm{Con}_S(x), \quad \text{where} \quad \mathrm{Con}_S(x) \stackrel{\text{def}}{=} \frac{1}{2^s}\frac{1}{2^n}\sum_{U\subseteq S}(-1)^{|U|} I\left(\bigvee_{U'\subseteq U} g_{U'}(x)\right). \quad (4.1)$$

The value $\mathrm{Con}_S(x)$ may be viewed as the "contribution" that $x$ makes to $\hat{f}(S)$. Recall that when $f$ has a term $T$ which contains all the variables in $S$, those $x \in Z_S(S)$ which uniquely satisfy $T$ will contribute to $\hat{f}(S)$. We will show that under suitable conditions on $f$, the other $x$'s make little or no contribution.

### 4.2.2 Bounding the contribution to $\hat{f}(S)$ from various inputs.

The variable $\mathcal{C}$ will denote a subset of $\mathcal{P}(S)$, the power set of $S$; i.e. $\mathcal{C}$ denotes a collection of subsets of $S$. We may view $\mathcal{C}$ as defining a set of $g_U$'s (those $g_U$'s for which $U$ belongs to $\mathcal{C}$).

We may partition the set of inputs $\{0,1\}^n$ into $2^{|\mathcal{P}(S)|} = 2^{2^s}$ parts according to what subset of the $2^s$ functions $\{g_U\}_{U \subseteq S}$ each $x \in \{0,1\}^n$ satisfies. For $\mathcal{C}$ a subset of $\mathcal{P}(S)$ we denote the corresponding piece of the partition by $P_{\mathcal{C}}$; so $P_{\mathcal{C}}$ consists of precisely those $x \in \{0,1\}^n$ that satisfy $\left(\bigwedge_{U \in \mathcal{C}} g_U\right) \wedge \left(\bigwedge_{U \notin \mathcal{C}} \overline{g}_U\right)$. Note that for any given fixed $\mathcal{C}$, each $x$ in $P_{\mathcal{C}}$ has exactly the same contribution $\mathrm{Con}_S(x)$ to the Fourier coefficient $\hat{f}(S)$ as every other $x'$ in $P_{\mathcal{C}}$; this is simply because $x$ and $x'$ will satisfy exactly the same set of $g_{U'}$'s in (4.1). More generally, we have the following:

**Lemma 12.** *Let $\mathcal{C}$ be any subset of $\mathcal{P}(S)$. Suppose that there exist $U_1, U_2 \in \mathcal{C}$ such that $U_1 \subsetneq U_2$. Then for any $y, z$ where $y \in P_{\mathcal{C}}$ and $z \in P_{\mathcal{C} \setminus U_2}$, we have that:* $\mathrm{Con}_S(y) = \mathrm{Con}_S(z)$.

*Proof.* Consider (4.1). Fix any subset $U$ of $S$. We shall show that the indicator variable $I\left(\bigvee_{U' \subseteq U} g_{U'}(x)\right)$ takes the same value on $y$ and on $z$.

Recall that $y$ satisfies precisely those $g_r$'s such that $r \in \mathcal{C}$, and $z$ satisfies precisely those $g_r$'s such that $r \in (\mathcal{C} \setminus U_2)$. We have that:

1. $\bigvee_{U' \subseteq U} g_{U'}(y)$ is true if and only if there exists some $U' \subseteq U$, $U' \in \mathcal{C}$; and
2. $\bigvee_{U' \subseteq U} g_{U'}(z)$ is true if and only if there exists some $U'' \subseteq U$, $U'' \in (\mathcal{C} \setminus U_2)$.

Since $U_1 \subsetneq U_2$ and $U_1, U_2 \in \mathcal{C}$, there exists a $U'$ as described above if and only if there exists a $U''$ as described above. ■

Given a collection $\mathcal{C}$ of subsets of $S$, we write $\mathrm{Con}_S(\mathcal{C})$ to denote $\sum_{x \in P_{\mathcal{C}}} \mathrm{Con}_S(x)$, and we refer to this quantity as the contribution that $\mathcal{C}$ makes to the Fourier coefficient $\hat{f}(S)$. It is clear that we have $\hat{f}(S) = \sum_{\mathcal{C} \subseteq \mathcal{P}(S)} \mathrm{Con}_S(\mathcal{C})$.

The following lemma establishes a broad class of $\mathcal{C}$'s for which $\mathrm{Con}_S(\mathcal{C})$ is zero:

**Lemma 13.** *Let $\mathcal{C}$ be any collection of subsets of $S$. If $\bigcup_{U \in \mathcal{C}} U \neq S$ then* $\mathrm{Con}_S(x) = 0$ *for each $x \in P_{\mathcal{C}}$ and hence* $\mathrm{Con}_S(\mathcal{C}) = 0$.

Before proving Lemma 13 we first introduce some notation and make a few easy observations. Let $\text{odd}(U) \subset \mathcal{P}(S)$ be the set of all the odd-sized subsets of $S$ that are supersets of $U$, and let $\text{even}(U)$ be defined similarly. For any $U \subsetneq S$ we have $|\text{odd}(U)| = |\text{even}(U)|$ since there are exactly $2^{|S|-|U|}$ subsets of $S$ containing $U$, half of which are even and half of which are odd. Note that if $U$ is the entire set $S$, then $S$ is the only superset of $U$, and of course $|S|$ cannot be both even and odd. Finally, note that given subsets $U_1, \ldots, U_k$ of $S$, we have:

$$\cap_{i=1}^{k} \text{odd}(U_i) = \text{odd}(\cup_{i=1}^{k} U_i). \tag{4.2}$$

(This just says that the intersection of the $\text{odd}(U_i)$'s is equal to the set of all odd subsets of $S$ that contain the union of all the $U_i$'s.) A similar equality $\cap_{i=1}^{k}\text{even}(U_i) = \text{even}(\cup_{i=1}^{k} U_i)$ also holds.

Now we can give the proof:

**Proof of Lemma 13.** We know that each $x$ in $P_{\mathcal{C}}$ makes the same contribution to $\hat{f}(S)$. So fix any $x \in P_{\mathcal{C}}$; it suffices to show that the quantity $\sum_{U \subseteq S}(-1)^{|U|} I\left(\bigvee_{U' \subseteq U} g_{U'}(x)\right)$ is zero. This quantity will be zero if $x$ satisfies an equal number of $\bigvee_{U' \subseteq U} g_{U'}(x)$ for which $|U|$ is even, and for which $|U|$ is odd. The $U$ for which $x$ satisfies $\bigvee_{U' \subseteq U} g_{U'}(x)$ are the $U$ for which there exist some $U' \in \mathcal{C}$ such that $U' \subseteq U$. Thus, we need to count the number of even and odd-sized $U \subseteq S$ containing some $U' \in \mathcal{C}$, and show that $|\cup_{U' \in \mathcal{C}} \text{odd}(U')| = |\cup_{U' \in \mathcal{C}} \text{even}(U')|$. Let $\mathcal{C} = \{U_1, \ldots, U_k\} \subseteq \mathcal{P}(\mathcal{S})$. By inclusion-exclusion,

$$\left|\cup_{U' \in \mathcal{C}} \text{odd}(U')\right| = \sum_{i=1}^{k} |\text{odd}(U_i)| - \sum_{i_1, i_2} |\text{odd}(U_{i_1}) \cap \text{odd}(U_{i_2})| \ldots + (-1)^{k-1} \left|\cap_{i=1}^{k} \text{odd}(U_i)\right|, \tag{4.3}$$

and we have a similar expression for $|\cup_{U' \in \mathcal{C}} \text{even}(U')|$ (identical to the RHS of (4.3) except with "even" everywhere in place of "odd").

By (4.2) we can rewrite each intersections of some $\text{odd}(U_i)$'s as $\text{odd}(\cup U_i)$, and similarly we can rewrite each intersection of some $\text{even}(U_i)$'s as $\text{even}(\cup U_i)$'s. Thus the RHS of (4.3) can be rewritten as a sum of $|\text{odd}(\cup U_i)|$'s, and similarly $|\cup_{U' \in \mathcal{C}} \text{even}(U')|$ can be rewritten as an identical sum of $|\text{even}(\cup U_i)|$'s. Since by assumption each of these $\cup U_i$'s cannot be the whole set $S$, for each $\cup U_i$ we have $|\text{odd}(\cup U_i)| = |\text{even}(\cup U_i)|$. Therefore all the terms

of $|\cup_{U' \in \mathcal{C}} \text{odd}(U')|$ in (4.3) will match up with all the terms of $|\cup_{U' \in \mathcal{C}} \text{even}(U')|$. It follows that $|\cup_{U' \in \mathcal{C}} \text{odd}(U')|$ is indeed equal to $|\cup_{U' \in \mathcal{C}} \text{even}(U')|$, and the lemma is proved. ■

It remains to analyze those $\mathcal{C}$'s for which $\bigcup_{U \in \mathcal{C}} U = S$; for such a $\mathcal{C}$ we say that $\mathcal{C}$ *covers* $S$.

Recall from the previous discussion that $\text{Con}_S(\mathcal{C}) = |P_{\mathcal{C}}| \cdot \text{Con}_S(x)$ where $x$ is any element of $P_{\mathcal{C}}$. Since $|\text{Con}_S(x)| \leq \frac{1}{2^n}$ for all $x \in \{0,1\}^n$, for any collection $\mathcal{C}$, we have that

$$|\text{Con}_S(\mathcal{C})| \;\leq\; \Pr_{x \in U_n}[x \in P_{\mathcal{C}}] \;=\; \Pr_{x \in U_n}[(\bigwedge_{U \in \mathcal{C}} g_U) \wedge (\bigwedge_{U \notin \mathcal{C}} \overline{g}_s)] \;\leq\; \Pr_{x \in U_n}[(\bigwedge_{U \in \mathcal{C}} g_U)].$$

We are interested in bounding this probability for $\mathcal{C} \neq \{S\}$ (we will deal with the special case $\mathcal{C} = \{S\}$ separately later). Recall that each $g_U$ is a disjunction of terms; the expression $\bigwedge_{U \in \mathcal{C}} g_U$ is satisfied by precisely those $x$ that satisfy at least one term from each $g_U$ as $U$ ranges over all elements of $\mathcal{C}$. For $j \geq 1$ let us define a quantity $B_j$ as follows

$$B_j \stackrel{\text{def}}{=} \max_{i_1,\ldots,i_j} \Pr_{x \in U_n}[x \text{ simultaneously satisfies terms } T_{i_1}, \ldots, T_{i_j} \text{ in } \vee_{U \subseteq S}(g_U)]$$

where the max is taken over all $j$-tuples of distinct terms in $\vee_{U \subseteq S}(g_U)$. Then it is not hard to see that by a union bound, we have

$$|\text{Con}_S(\mathcal{C})| \leq B_{|\mathcal{C}|} \prod_{U \in \mathcal{C}} (\#g_U), \tag{4.4}$$

where $\#g_U$ denotes the number of terms in the monotone DNF $g_U$.

The idea of why (4.4) is a useful bound is as follows. Intuitively, one would expect that the value of $B_j$ decreases as $j$ (the number of terms that must be satisfied) increases. One would also expect the value of $\#g_U$ to decrease as the size of $U$ increases (if $U$ contains more variables then fewer terms in $f$ will contain all of those variables). This means that there is a trade-off which helps us bound (4.4): if $|\mathcal{C}|$ is large then $B_{|\mathcal{C}|}$ is small, but if $|\mathcal{C}|$ is small then (since we know that $\bigcup_{U \in \mathcal{C}} U = S$) some $U$ is large and so $\prod_{U \in \mathcal{C}} \#g_U$ will be smaller.

### 4.2.3 Bounding $\hat{f}(S)$ based on whether $S$ co-occurs in some term of $f$.

We are now ready to state formally the conditions on $\hat{f}$ that allow us to detect a co-occurrence of variables in the value of the corresponding Fourier coefficient.

**Lemma 14.** *Let $f : \{0,1\}^n \to \{-1,1\}$ be a monotone DNF. Fix a set $S \subset [n]$ of size $|S| = s$ and let*

$$\mathscr{Y} = \{\mathcal{C} \subseteq \mathcal{P}(S) : \mathcal{C} \text{ covers } S \text{ and } S \notin \mathcal{C}\}.$$

*Suppose that we define $\alpha, \beta_1, \ldots, \beta_{2^s}$ and $\Phi : \mathscr{Y} \to \mathbb{R}$ so that:*

**(C1)** *Each term in $f$ is uniquely satisfied with probability at least $\alpha$;*

**(C2)** *For $1 \leq j \leq 2^s$, each $j$-tuple of terms in $f$ is simultaneously satisfied with probability at most $\beta_j$; and*

**(C3)** *For every $\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}$ we have $\prod_{U \in \mathcal{C}_{\mathscr{Y}}} (\# g_U) \leq \Phi(\mathcal{C}_{\mathscr{Y}})$.*

*Then*

1. *If the variables in $S$ do not simultaneously co-occur in any term of $f$, then*

$$|\hat{f}(S)| \leq \Upsilon \quad \text{where} \quad \Upsilon := \sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} \left(2^s \beta_{|\mathcal{C}_{\mathscr{Y}}|} \Phi(\mathcal{C}_{\mathscr{Y}})\right);$$

2. *If the variables in $S$ do simultaneously co-occur in some term of $f$, then $|\hat{f}(S)| \geq \frac{\alpha}{2^s} - 2 \cdot \Upsilon$.*

Using Lemma 14, if $f$ satisfies conditions **(C1)** through **(C3)** with values of $\beta_j$ and $\Phi(\cdot)$ so that there is a "gap" between $\alpha/2^s$ and $3\Upsilon$, then we can determine whether all the variables in $S$ simultaneously co-occur in a term by estimating the magnitude of $\hat{f}(S)$.

*Proof.* Let $\mathcal{C}^\star$ denote the 'special' element of $P(S)$ that consists solely of the subset $S$, i.e. $\mathcal{C}^\star = \{S\}$, and let $\mathscr{X} = \{\mathcal{C} \subseteq \mathcal{P}(S) : \mathcal{C} \text{ covers } S \text{ and } S \in \mathcal{C} \text{ and } \mathcal{C} \neq \mathcal{C}^\star\}$. Using Lemma 13, we have

$$\hat{f}(S) = \mathrm{Con}_S(\mathcal{C}^\star) + \sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} \mathrm{Con}_S(\mathcal{C}_{\mathscr{Y}}) + \sum_{\mathcal{C}_{\mathscr{X}} \in \mathscr{X}} \mathrm{Con}_S(\mathcal{C}_{\mathscr{X}}). \tag{4.5}$$

We first prove point 1. Suppose that the variables of $S$ do not simultaneously co-occur in any term of $f$. Then $g_S$ is the empty disjunction and $\# g_S = 0$, so $\mathrm{Con}_S(\mathcal{C}) = 0$ for any $\mathcal{C}$ containing $S$. Thus in this case we have $\hat{f}(S) = \sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} \mathrm{Con}_S(\mathcal{C}_{\mathscr{Y}})$; using (4.4) and condition **(C3)**, it follows that $|\hat{f}(S)|$ is at most $\sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} B_{|\mathcal{C}_{\mathscr{Y}}|} \Phi(\mathcal{C}_{\mathscr{Y}})$.

We now claim that $B_{|\mathcal{C}_{\mathscr{Y}}|} \leq 2^s \beta_{|\mathcal{C}_{\mathscr{Y}}|}$; we establish this by showing that $B_j \leq 2^s \beta_j$ for all $j$. In other words, we shall bound the probability of simultaneously satisfying any fixed collection of $j$ terms in the DNF $f' = \vee_{U \subseteq S}(g_U)$. We have that for $1 \leq j \leq 2^s$, each $j$-tuple of terms in $f$ is simultaneously satisfied with probability at most $\beta_j$. Consider any fixed sequence $T'_{i_1}, \ldots, T'_{i_j}$ of terms from $f'$. Let $T_{i_1}, \ldots, T_{i_j}$ denote the sequence of terms in $f$ from which the terms $T'_{i_1}, \ldots, T'_{i_j}$ were derived, i.e. each term $T$ consists of $T' \wedge (\wedge_{i \in U} x_i)$ for some $U \subseteq S$. Since $T_{i_1} \wedge \cdots \wedge T_{i_j}$ is simply a monotone conjunction and $T'_{i_1} \wedge \cdots \wedge T'_{i_j}$ is simply the corresponding conjunction obtained by removing at most $|S| = s$ variables from $T_{i_1} \wedge \cdots \wedge T_{i_j}$, we have that $\Pr_x[T'_{i_1} \wedge \cdots \wedge T'_{i_j}] \leq 2^s \beta_j$.

So in this case we have

$$|\hat{f}(S)| \leq \sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} |\mathrm{Con}_S(\mathcal{C}_{\mathscr{Y}})| \leq \sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} B_{|\mathcal{C}_{\mathscr{Y}}|} \Phi(\mathcal{C}_{\mathscr{Y}}) \leq \sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} \left(2^s \beta_{|\mathcal{C}_{\mathscr{Y}}|} \Phi(\mathcal{C}_{\mathscr{Y}})\right) = \Upsilon.$$

Now we turn to point 2. Suppose that the variables of $S$ do co-occur in some term of $f$. Let $x$ be any element of $P_{\mathcal{C}^\star}$, so $x$ satisfies $g_U$ if and only if $U = S$. It is easy to see from (4.1) that for such an $x$ we have $\mathrm{Con}_S(x) = (-1)^{|S|}/(2^n 2^s)$. We thus have that

$$\mathrm{Con}_S(\mathcal{C}^\star) = \frac{(-1)^{|S|}}{2^s} \cdot \Pr[x \in P_{\mathcal{C}^\star}] = \frac{(-1)^{|S|}}{2^s} \Pr[g_S \wedge (\bigwedge_{U \subsetneq S} \overline{g}_U)]. \tag{4.6}$$

Since $S$ co-occurs in some term of $f$, we have that $g_S$ contains at least one term $T$. By condition **(C1)**, the corresponding term $(T \wedge (\wedge_{i \in S} x_i))$ of $f$ is uniquely satisfied with probability at least $\alpha$. Since each assignment that uniquely satisfies $(T \wedge (\wedge_{i \in S} x_i))$ (among all the terms of $f$) must satisfy $g_S \wedge (\bigwedge_{U \subsetneq S} \overline{g}_U)$, we have that the magnitude of (4.6) is at least $\alpha/2^s$.

We now show that $|\sum_{\mathcal{C}_{\mathscr{X}} \in \mathscr{X}} \mathrm{Con}_S(\mathcal{C}_{\mathscr{X}})| \leq \Upsilon$, which completes the proof, since we already have that $|\sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} \mathrm{Con}_S(\mathcal{C}_{\mathscr{Y}})| \leq \sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} |\mathrm{Con}_S(\mathcal{C}_{\mathscr{Y}})| \leq \Upsilon$. First note that if the set $\mathcal{C}_{\mathscr{X}} \setminus \{S\}$ does not cover $S$, then by Lemmas 12 and 13 we have that $\mathrm{Con}_S(x) = 0$ for each $x \in P_{\mathcal{C}_{\mathscr{X}}}$ and thus $\mathrm{Con}_S(\mathcal{C}_{\mathscr{X}}) = 0$. So we may restrict our attention to those $\mathcal{C}_{\mathscr{X}}$ such that $\mathcal{C}_{\mathscr{X}} \setminus \{S\}$ covers $S$. Now since such a $\mathcal{C}_{\mathscr{X}} \setminus \{S\}$ is simply some $\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}$, and each $\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}$ is obtained as $\mathcal{C}_{\mathscr{X}} \setminus \{S\}$ for at most one $\mathcal{C}_{\mathscr{X}} \in \mathscr{X}$, we have

$$\left| \sum_{\mathcal{C}_{\mathscr{X}} \in \mathscr{X}} \mathrm{Con}_S(\mathcal{C}_{\mathscr{X}}) \right| \leq \sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} |\mathrm{Con}_S(\mathcal{C}_{\mathscr{Y}})| \leq \Upsilon.$$

■

## 4.3 Hypothesis Formation

In this section, we show that if a target monotone DNF $f$ satisfies the conditions of Lemma 14 and two other simple conditions stated below (see Theorem 12), then it is possible to learn $f$ from uniform random examples.

**Theorem 12.** *Let $f$ be a $t$-term monotone DNF. Fix $s \in [n]$. Suppose that*

- *For all sets $S \subset [n], |S| = s$, conditions* **(C1)** *through* **(C3)** *of Lemma 14 hold for certain values $\alpha$, $\beta_j$, and $\Phi(\cdot)$ satisfying $\Delta > 0$, where $\Delta := \alpha/2^s - 3 \cdot \Upsilon$. (Recall that $\Upsilon := \sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} \left(2^s \beta_{|\mathcal{C}_{\mathscr{Y}}|} \Phi(\mathcal{C}_{\mathscr{Y}})\right)$, where $\mathscr{Y} = \{\mathcal{C} \subseteq \mathcal{P}(S) : \mathcal{C}$ covers $S$ and $S \notin \mathcal{C}\}$.)*

**(C4)** *Every set $S$ of $s$ co-occurring variables in $f$ appears in at most $\gamma$ terms (here $\gamma \geq 2$); and*

**(C5)** *Every term of $f$ contains at most $\kappa$ variables (note that $s \leq \kappa \leq n$).*

*Then algorithm $\mathcal{A}$ PAC learns $f$ to accuracy $\epsilon$ with confidence $1-\delta$ given access to $EX(f, U_n)$, and runs in time poly$(n^{s+\gamma}, t, 1/\Delta, \gamma^{\kappa}, 1/\epsilon, \log(1/\delta))$.*

Algorithm $\mathcal{A}$ is described on the following page.

**Algorithm** $\mathcal{A}$ (inputs are $\epsilon, \delta, s, \alpha, \Upsilon, \gamma, \kappa$, and access to $EX(f, U_n)$)

1. Define $\Delta := \alpha/2^s - 3 \cdot \Upsilon$.
2. For each $S \subset [n], |S| = s$, empirically estimate $\hat{f}(S)$ to within $\pm\Delta/3$ (with confidence $1 - \delta/3$ that all estimates have the required accuracy); let $\tilde{f}(S)$ be the empirical estimate thus obtained. Mark as "good" each $S$ for which $|\tilde{f}(S)| \geq \Upsilon + \frac{\Delta}{2}$.
3. Let $G_f$ denote the following $n$-vertex hypergraph: the vertices of $G_f$ correspond to variables $x_1, \ldots, x_n$, and $G_f$ contains each $s$-vertex hyperedge $S$ if and only if $S$ was marked as "good" in the previous step.
4. Run algorithm $\mathcal{A}'$ to identify the set $HC_f$ of all of the $k$-hypercliques in $G_f$, as $k$ ranges over $\{s, \ldots, \kappa\}$.
5. Run the standard elimination algorithm for disjunctions—with $\epsilon$ as the accuracy input parameter and $\delta/3$ as the confidence—over the "features" that are the monotone conjunctions corresponding to the hypercliques identified in the previous step. Output the resulting hypothesis $h$ (which is a monotone DNF).

**Algorithm** $\mathcal{A}'$ (input is the list of "good" sets $S$ identified in Step 1 of Algorithm $\mathcal{A}$)

1. For each good set $S$, run Algorithm $\mathcal{A}''$ to identify the set $N_S$ of all variables in $[n] \setminus S$ that occur in some term that also contains all variables in $S$.
2. For all $s \leq k \leq \kappa$, using brute-force search over all subsets $N'$ of at most $(k - s)$ many elements from $N_S$, check whether $N' \cup S$ is a $k$-hyperclique in $G_f$.

**Algorithm** $\mathcal{A}''$ (input is a good set $S$)

1. For each subset $N$ of at most $\gamma$ variables from $[n] \setminus S$, perform the following:
   (a) Empirically estimate $\widehat{f_{N \leftarrow 0}}(S)$ to additive accuracy $\pm\Delta/3$; let $\widetilde{f_{N \leftarrow 0}}(S)$ be the empirical estimate thus obtained. Mark each $N$ for which $\widetilde{f_{N \leftarrow 0}}(S) \geq \Upsilon + \frac{\Delta}{2}$.
2. Let $N_S$ be the union of all the $N$'s that were marked in the previous step. Return $N_S$.

*Proof.* Lemma 14 implies that for each set $S \subset [n], |S| = s$,

- if the variables in $S$ all co-occur in some term of $f$, then $|\hat{f}(S)|$ is at least $\Delta/2$ larger than $\Upsilon + \Delta/2$;
- if the variables in $S$ do not all co-occur in some term of $f$, then $|\hat{f}(S)|$ is at least $\Delta/2$ smaller than $\Upsilon + \Delta/2$.

A straightforward application of Hoeffding bounds (to estimate the Fourier coefficients using a random sample of uniformly distributed examples) shows that Step 1 of Algorithm $\mathcal{A}$ can be executed in $\text{poly}(n^s, 1/\Delta, \log(1/\delta))$ time, and that with probability $1 - \delta/3$ the $S$'s that are marked as "good" will be precisely the $s$-tuples of variables that co-occur in some term of $f$.

Conceptually, the algorithm next constructs the hypergraph $G_f$ that has one vertex per variable in $f$ and that includes an $s$-vertex hyperedge if and only if the corresponding $s$ variables co-occur in some term of $f$. Clearly there is a $k$-hyperclique in $G_f$ for each term of $k$ variables in $f$. So if we could find all of the $k$-hypercliques in $G_f$ (where again $k$ ranges between $s$ and $\kappa$), then we could create a set $HC_f$ of monotone conjunctions of variables such that $f$ could be represented as an OR of $t$ of these conjunctions. Treating each of the conjunctions in $HC_f$ as a variable in the standard elimination algorithm for learning disjunctions (see e.g. Chapter 1 of [Kearns and Vazirani, 1994]) would then enable us to properly PAC learn $f$ to accuracy $\epsilon$ with probability at least $1 - \delta/3$ in time polynomial in $n$, $t$, $|HC_f|$, $1/\epsilon$, and $\log(1/\delta)$. Thus, $\mathcal{A}$ will use a subalgorithm $\mathcal{A}'$ to find all of the $k$-hypercliques in $G_f$ and will then apply the elimination algorithm over the corresponding conjunctions to learn the final approximator $h$.

We now explain the subalgorithm $\mathcal{A}'$ for locating the set $HC_f$ of $k$-hypercliques. For each set $S$ of $s$ co-occurring variables, let $N_S \subseteq ([n] \setminus S)$ be defined as follows: a variable $x_i$ is in $N_S$ if and only if $x_i$ is present in some term that contains all of the variables in $S$. Since by assumption there are at most $\gamma$ terms containing such variables and each term contains at most $\kappa$ variables, this means that $|N_S| < \kappa\gamma$. The subalgorithm will use this bound as follows. For each set $S$ of $s$ co-occurring variables, $\mathcal{A}'$ will determine the set $N_S$ using a procedure $\mathcal{A}''$ described shortly. Then, for each $s \leq k \leq \kappa$ and each $(k-s)$-element

subset $N'$ of $N_S$, $\mathcal{A}'$ will test whether or not $N' \cup S$ is a $k$-hyperclique in $G_f$. The set of all $k$-hypercliques found in this way is $HC_f$. For each $S$, the number of sets tested in this process is at most

$$\sum_{i=0}^{\kappa} \binom{|N_S|}{i} \leq \sum_{i=0}^{\kappa} \binom{\kappa\gamma}{i} \leq \left(\frac{e\kappa\gamma}{\kappa}\right)^{\kappa} = (e\gamma)^{\kappa}.$$

Thus, $|HC_f| = O(n^s(e\gamma)^{\kappa})$, and this is an upper bound on the time required to execute Step 2 of subalgorithm $\mathcal{A}'$.

Finally, we need to define the procedure $\mathcal{A}''$ for finding $N_S$ for a given set $S$ of $s$ co-occurring variables. Fix such an $S$ and let $N_\gamma$ be a set of at most $\gamma$ variables in $([n] \setminus S)$ having the following properties:

**(P1)** In the projection $f_{N_\gamma \leftarrow 0}$ of $f$ in which all of the variables of $N_\gamma$ are fixed to 0, the variables in $S$ do not co-occur in any term; and

**(P2)** For every set $N'_\gamma \subset N_\gamma$ such that $|N'_\gamma| = |N_\gamma| - 1$, the variables in $S$ do co-occur in at least one term of $f_{N'_\gamma \leftarrow 0}$.

We will use the following claim:

**Claim 3.** *$N_S$ is the union of all sets $N_\gamma$ of cardinality at most $\gamma$ that satisfy* **(P1)** *and* **(P2)**.

*Proof.* We first show that the union of all sets satisfying **(P1)** and **(P2)** is a subset of $N_S$. To see this, note that if variable $x_i$ is not in $N_S$ (i.e. $x_i$ does not co-occur with $S$ in any term), then any set $N_\gamma$ that includes $x_i$ cannot satisfy both properties. This is because if $N_\gamma$ satisfies **(P1)** (i.e. $S$ does not co-occur in any term of $f_{N_\gamma \leftarrow 0}$), then the set $N'_\gamma = N_\gamma \setminus x_i$ will also be such that $S$ do not co-occur in any term of $f_{N'_\gamma \leftarrow 0}$, since $x$ does not co-occur with $S$ in any term.

Next, consider the minimal monotone DNF representation $D_f$ of the target $f$. Let $D_{f_S}$ be the monotone DNF expression obtained from $D_f$ by removing from $D_f$ all terms in which the variables of $S$ do not co-occur and then fixing all of the variables in $S$ to 1. Since $D_{f_S}$ has at most $\gamma$ terms, there is an equivalent minimal CNF $C_{f_S}$ in which each clause contains at most $\gamma$ variables. For each clause $C_i$ in $C_{f_S}$, the set of variables in $C_i$ satisfies both **(P1)** and **(P2)**: setting all of the variables in $C_i$ to 0 falsifies both $C_{f_S}$ and $D_{f_S}$ and therefore

removes from $f$ all terms in which the variables of $S$ co-occur; but setting any proper subset of the variables in $C_i$ to 0 does not falsify $D_{f_S}$ and therefore leaves at least one term in $f$ in which the variables of $S$ co-occur. Furthermore, all of the variables in $D_{f_S}$ are also relevant in $C_{f_S}$, so every variable in $D_{f_S}$ appears in at least one clause of $C_{f_S}$. It follows that the union of the variables in the sets $N_\gamma$ satisfying **(P1)** and **(P2)** is a superset of the set of variables in $D_{f_S}$, that is, the set $N_S$. ■

There are only $O(n^\gamma)$ possible candidate sets $N_\gamma$ to consider, so our problem now reduces to the following: given a set $N$ of at most $\gamma$ variables, determine whether the variables in $S$ co-occur in $f_{N\leftarrow 0}$.

Recall that since $f$ satisfies the three conditions **(C1)**, **(C2)** and **(C3)**, Lemma 14 implies that $|\hat{f}(S)|$ is either at most $\Upsilon$ (if the variables in $S$ do not co-occur in any term of $f$) or at least $\frac{\alpha}{2^s} - 2 \cdot \Upsilon$ (if the variables in $S$ do co-occur in some term). We now claim that the function $f_{N\leftarrow 0}$ has this property as well: i.e., $|\widehat{f_{N\leftarrow 0}}(S)|$ is either at most the same value $\Upsilon$ (if the variables in $S$ do not co-occur in any term of $f_{N\leftarrow 0}$) or at least the same value $\frac{\alpha}{2^s} - 2 \cdot \Upsilon$ (if the variables in $S$ do co-occur in some term of $f_{N\leftarrow 0}$). To see this, observe that the function $f_{N\leftarrow 0}$ is just $f$ with some terms removed. Since each term in $f$ is uniquely satisfied with probability at least $\alpha$ (this is condition **(C1)**), the same must be true of $f_{N\leftarrow 0}$ since removing terms from $f$ can only increase the probability of being uniquely satisfied for the remaining terms. Since each $j$-tuple of terms in $f$ is simultaneously satisfied with probability at most $\beta_j$ (this is condition **(C2)**), the same must be true for $j$-tuples of terms in $f_{N\leftarrow 0}$. Finally, for condition **(C3)**, the value of $\#g_U$ can only decrease in passing from $f$ to $f_{N\leftarrow 0}$. Thus, the upper bound of $\Upsilon$ that follows from applying Lemma 14 to $f$ is also a legitimate upper bound when the lemma is applied to $|\widehat{f_{N\leftarrow 0}}(S)|$, and similarly the lower bound of $\frac{\alpha}{2^s} - 2 \cdot \Upsilon$ is also a legitimate lower bound when the lemma is applied to $f_{N\leftarrow 0}$. Therefore, for every $|N| \leq \gamma$, a sufficiently accurate (within $\Delta/2$) estimate of $\widehat{f_{N\leftarrow 0}}(S)$ (as obtained in Step 1 of subalgorithm $\mathcal{A}''$) can be used to determine whether or not the variables in $S$ co-occur in any term of $f_{N\leftarrow 0}$.

To obtain the required estimate for $\widehat{f_{N\leftarrow 0}}$, observe that for a given set $N$, we can simulate a uniform example oracle for $f_{N\leftarrow 0}$ by filtering the examples from the uniform oracle for $f$ so that only examples setting the variables in $N$ to 0 are accepted. Since $|N| \leq \gamma$, the

filter accepts with probability at least $1/2^\gamma$. A Hoeffding bound argument then shows that the Fourier coefficients $\widehat{f_{N\leftarrow 0}}(S)$ can be estimated (with probability of failure no more than a small fraction of $\delta$) from an example oracle for $f$ in time polynomial in $n$, $2^\gamma$, $1/\Delta$, and $\log(1/\delta)$.

Algorithm $\mathcal{A}''$, then, estimates Fourier coefficients of restricted versions of $f$, using a sample size sufficient to ensure that all of these coefficients are sufficiently accurate over all calls to $\mathcal{A}''$ with probability at least $1 - \delta/3$. These estimated coefficients are then used by $\mathcal{A}''$ to locate the set $N_S$ as just described. The overall algorithm $\mathcal{A}$ therefore succeeds with probability at least $1 - \delta$, and it is not hard to see that it runs in the time bound claimed. ■

**Required parameters.** In the above description of Algorithm $\mathcal{A}$, we assumed that it is given the values of $s, \alpha, \Upsilon, \gamma$, and $\kappa$. In fact it is not necessary to assume this; a standard argument gives a variant of the algorithm which succeeds without being given the values of these parameters.

The idea is simply to have the algorithm "guess" the values of each of these parameters, either exactly or to an adequate accuracy. The parameters $s, \gamma$ and $\kappa$ take positive integer values bounded by poly($n$). The other parameters $\alpha, \Upsilon$ take values between 0 and 1; a standard argument shows that if approximate values $\alpha'$ and $\Upsilon'$ (that differ from the true values by at most $1/\text{poly}(n)$) are used instead of the true values, the algorithm will still succeed. Thus there are at most poly($n$) total possible settings for $(s, \gamma, \kappa, \alpha, \Upsilon)$ that need to be tried. We can run Algorithm $\mathcal{A}$ for each of these candidate parameter settings, and test the resulting hypothesis; when we find the "right" parameter setting, we will obtain a high-accuracy hypothesis (and when this occurs, it is easy to recognize that it has occurred, simply by testing each hypothesis on a new sample of random labeled examples). This parameter guessing incurs an additional polynomial factor overhead. Thus Theorem 12 holds true for the extended version of Algorithm $\mathcal{A}$ that takes only $\epsilon, \delta$ as input parameters.

## 4.4 Random Monotone DNF

### 4.4.1 The Random Monotone DNF Model

We define a random monotone DNF according to the non-monotone definition from 2. Let $\mathcal{M}_n^{t,k}$ be the probability distribution over monotone $t$-term DNF induced by the following process: each term is independently and uniformly chosen at random from all $\binom{n}{k}$ monotone ANDs of size exactly $k$ over $x_1, \ldots, x_n$.

Given a value of $t$, throughout this section we consider the $\mathcal{M}_n^{t,k}$ distribution where $k = \lfloor \log t \rfloor$ (we will relax this and consider a broader range of values for $k$ in Section 4.6). To motivate this choice, consider a random draw of $f$ from $\mathcal{M}_n^{t,k}$. If $k$ is too large relative to $t$ then a random $f \in \mathcal{M}_n^{t,k}$ will likely have $\Pr_{x \in U_n}[f(x) = 1] \approx 0$, and if $k$ is too small relative to $t$ then a random $f \in \mathcal{M}_n^{t,k}$ will likely have $\Pr_{x \in U_n}[f(x) = 1] \approx 1$; such functions are trivial to learn to high accuracy using either the constant-0 or constant-1 hypothesis. A straightforward analysis (see e.g. [Jackson and Servedio, 2006]) shows that for $k = \lfloor \log t \rfloor$ we have that $\mathbf{E}_{f \in \mathcal{M}_n^{t,k}}[\Pr_{x \in U_n}[f(x) = 1]]$ is bounded away from both 0 and 1, and thus we feel that this is an appealing and natural choice.

### 4.4.2 Probabilistic Analysis

In this section we will establish various useful probabilistic lemmas regarding random monotone DNF of polynomially bounded size. We have grouped the statements of our probabilistic lemmas close together in this section, and provide all proofs in Appendix B.

**Assumptions:** Throughout the rest of Section 4.4 we assume that $t(n)$ is any function such that $n^{3/2} \leq t(n) \leq \text{poly}(n)$. To handle the case when $t(n) \leq n^{3/2}$, we will use the results from [Jackson and Servedio, 2006]. Let $a(n)$ be such that $t(n) = n^{a(n)}$. For brevity we write $t$ for $t(n)$ and $a$ for $a(n)$ below, but the reader should keep in mind that $a$ actually denotes a function $\frac{3}{2} \leq a = a(n) \leq O(1)$. The first lemma provides a bound of the sort needed by condition **(C3)** of Lemma 14:

**Lemma 15.** *Let $|S| = s = \lfloor a \rfloor + 2$. Fix any $\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}$. Let $\delta_{\text{terms}} = n^{-\Omega(\log n)}$. With probability at least $1 - \delta_{\text{terms}}$ over the random draw of $f$ from $\mathcal{M}_n^{t,k}$, we have that for some*

*absolute constant $c$ and all sufficiently large $n$,*

$$\prod_{U \in \mathcal{C}_{\mathscr{D}}} (\#g_U) \leq c \cdot \frac{t^{|\mathcal{C}_{\mathscr{D}}|-1} k^{2^s}}{\sqrt{n}}. \tag{4.7}$$

The following lemma shows that for $f$ drawn from $\mathcal{M}_n^{t,k}$, with high probability each term is "uniquely satisfied" by a noticeable fraction of assignments as required by condition **(C1)**. (Note that since $k = O(\log n)$ and $t > n^{3/2}$, we have $\delta_{\text{usat}} = n^{-\Omega(\log\log n)}$ in the following.)

**Lemma 16.** *Let $\delta_{\text{usat}} := \exp(\frac{-tk}{3n}) + t^2(\frac{k}{n})^{\log\log t}$. For $n$ sufficiently large, with probability at least $1 - \delta_{\text{usat}}$ over the random draw of $f = T_1 \vee \cdots \vee T_t$ from $\mathcal{M}_n^{t,k}$, $f$ is such that for all $i = 1, \ldots, t$ we have $\Pr_x[T_i$ is satisfied by $x$ but no other $T_j$ is satisfied by $x\ ] \geq \frac{\Theta(1)}{2^k}$.*

We now upper bound the probability that any $j$ distinct terms of a random DNF $f \in \mathcal{M}_n^{t,k}$ will be satisfied simultaneously (condition **(C2)**). (In the following lemma, note that for $j = \Theta(1)$, since $t = n^{\Theta(1)}$ and $k = \Theta(\log n)$ we have that the quantity $\delta_{\text{simult}}$ is $n^{-\Theta(\log\log n)}$.)

**Lemma 17.** *Let $1 \leq j \leq 2^s$, and let $\delta_{\text{simult}} := \frac{t^j e^{jk - \log k}(jk - \log k)^{\log k}}{n^{\log k}}$. With probability at least $1 - \delta_{\text{simult}}$ over the random draw of $f = T_1 \vee \cdots \vee T_t$ from $\mathcal{M}_n^{t,k}$, for all $1 \leq \iota_1 < \cdots < \iota_j \leq t$ we have $\Pr[T_{\iota_1} \wedge \ldots \wedge T_{\iota_j}] \leq \beta_j$, where $\beta_j := \frac{k}{2^{jk}}$.*

Finally, the following lemma shows that for all sufficiently large $n$, with high probability over the choice of $f$, every set $S$ of $s$ variables appears in at most $\gamma$ terms, where $\gamma$ is independent of $n$ (see condition **(C4)**).

**Lemma 18.** *Fix any constant $c > 0$. Let $s = \lfloor a \rfloor + 2$ and let $\gamma = a + c + 1$. Let $\delta_\gamma = n^{-c}$. Then for $n$ sufficiently large, with probability at least $1 - \delta_\gamma$ over the random draw of $f$ from $\mathcal{M}_n^{t,k}$, we have that every $s$-tuple of variables appears in at most $\gamma$ terms of $f$.*

## 4.5 Proof of Theorem 11

**Theorem 1. [Formally]** *Let $t(n)$ be any function such that $t(n) \leq poly(n)$, let $a(n) = O(1)$ be such that $t(n) = n^{a(n)}$, and let $c > 0$ be any fixed constant. Then for any $n^{-c} < \delta < 1$*

*and* $0 < \epsilon < 1$, $\mathcal{M}_n^{t(n),\lfloor \log t(n) \rfloor}$ *is PAC learnable under* $U_n$ *in poly*($n^{2a(n)+c+3}$,$(a(n)+c+1)^{\log t(n)}$,$t(n)$,$1/\epsilon$, $\log 1/\delta$) *time.*

*Proof.* The result is proved for $t(n) \leq n^{3/2}$ already in [Jackson and Servedio, 2006], so we henceforth assume that $t(n) \geq n^{3/2}$. We use Theorem 12 and show that for $s = \lfloor a(n) \rfloor + 2$, random monotone $t(n)$-term DNFs, with probability at least $1-\delta$, satisfy conditions **(C1)**–**(C5)**with values $\alpha, \beta_j, \Phi(\cdot), \Delta, \gamma$, and $\kappa$ such that $\Delta > 0$ and the quantities $n^{s+\gamma}, 1/\Delta$, and $\gamma^\kappa$ are polynomial in $n$. This will show that the extended version of Algorithm $\mathcal{A}$ defined in Section 4.3 PAC learns random monotone $t(n)$-term DNFs in time poly$(n, 1/\epsilon)$. Let $t = t(n)$ and $k = \lfloor \log t \rfloor$, and let $f$ be drawn randomly from $\mathcal{M}_n^{t,k}$. By Lemmas 15–18, with probability at least $1 - \delta_{\text{usat}} - \delta_\gamma - 2^{2^s}\delta_{\text{terms}} - \delta_{\text{simult}}$, $f$ will satisfy **(C1)**–**(C5)** with the following values:

$$\textbf{(C1)}\ \alpha > \tfrac{\Theta(1)}{2^k};\ \ \textbf{(C2)}\ \beta_j \leq \tfrac{k}{2^{jk}} \text{ for } 1 \leq j \leq 2^s;$$
$$\textbf{(C3)}\ \Phi(\mathcal{C}_{\mathscr{Y}}) \leq O(1)\tfrac{t^{|\mathcal{C}_{\mathscr{Y}}|-1}k^{2^s}}{\sqrt{n}} \text{ for all } \mathcal{C}_{\mathscr{Y}} \in \mathscr{Y};\ \textbf{(C4)}\ \gamma \leq a(n)+c+1;$$
$$\textbf{(C5)}\ \kappa = k = \lfloor \log t \rfloor,$$

which gives us that $n^{s+\gamma} = n^{2a+c+3}$ and $\gamma^\kappa = (a+c+1)^{\lfloor \log t \rfloor}$. Finally, we show that $\Delta = \Omega(1/t)$ so $1/\Delta$ is polynomial in $n$:

$$\begin{aligned}\Delta &= \alpha/2^s - 3\cdot\Upsilon = \frac{\Theta(1)}{t2^s} - 3\sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} 2^s \beta_{|\mathcal{C}_{\mathscr{Y}}|}\Phi(\mathcal{C}_{\mathscr{Y}}) \geq \frac{\Theta(1)}{t2^s} - \Theta(1)\sum_{\mathcal{C}_{\mathscr{Y}} \in \mathscr{Y}} 2^s \frac{k}{t^{|\mathcal{C}_{\mathscr{Y}}|}} \cdot \frac{t^{|\mathcal{C}_{\mathscr{Y}}|-1}k^{2^s}}{\sqrt{n}} \\ &= \frac{\Theta(1)}{t2^s} - \frac{\Theta(1)k^{2^s+1}}{t\sqrt{n}} = \Omega(1/t).\end{aligned}$$

■

## 4.6 Discussion

**Robustness of parameter settings.** Throughout Sections 4.4 and 4.5 we have assumed for simplicity that the term length $k$ in our random $t$-term monotone DNF is exactly $\lfloor \log t \rfloor$. In fact, the results extend to a broader range of $k$'s; one can straightforwardly verify that by very minor modifications of the given proofs, Theorem 11 holds for $\mathcal{M}_n^{t,k}$ for any $(\log t) - O(1) \leq k \leq O(\log t)$.

**Relation to previous results.** Our results are powerful enough to subsume some known "worst-case" results on learning restricted classes of monotone DNF formulas. Hancock

and Mansour [Hancock and Mansour, 1991a] have shown that read-$k$ monotone DNF (in which each Boolean variable $x_i$ occurs in at most $k$ terms) are learnable under the uniform distribution in poly$(n)$ time for constant $k$. Their result extends an earlier result of Kearns *et al.* [Kearns *et al.*, 1994a] showing that read-once DNF (which can be assumed monotone without loss of generality) are polynomial-time learnable under the uniform distribution.

It is not hard to see that (a very restricted special case of) our algorithm can be used to learn read-$k$ monotone DNF in polynomial time. Note first that we may assume the unknown target read-$k$ DNF $f$ has $\frac{\epsilon}{2} \leq \Pr[f(x) = 1] \leq 1 - \frac{\epsilon}{2}$, since otherwise it is trivial to learn to accuracy $\epsilon$.

We show that we can apply Theorem 12 to learn $f$. Any read-$k$ DNF has at most $kn$ total occurrences of variables, so we certainly have that $f$ is a $t(n)$-term DNF with $t(n) = O(n)$. We will take $s = 1$. Since $f$ is a read-$k$ DNF, we may take $\gamma = 2$ in condition **(C4)**. By the usual reasoning, we may suppose without loss of generality that each term of $f$ contains at most $O(\log \frac{n}{\epsilon})$ many variables (this is because the probability that any longer term is ever satisfied by any example in a poly$(n/\epsilon)$-size set of random examples is negligibly small). Thus we may take $\kappa = O(\log \frac{n}{\epsilon})$ in condition **(C5)**.

Turning to Lemma 14, since $s = 1$ we have that the collection $\mathscr{Y}$ is in fact empty – for $S = \{x_i\}$, the only $\mathcal{C} \subseteq \mathcal{P}(S)$ that cover $S$ are $\mathcal{C} = \{\emptyset, \{x_i\}\}$ and $\mathcal{C} = \{\{x_i\}\}$, both of which clearly contain $S$. We thus have $\Upsilon = 0$, so $\Delta = \frac{\alpha}{2}$ and it remains only to prove that $\alpha$ is "not too small," i.e. that each term in $f$ is uniquely satisfied with probability at least $\Omega(1/\text{poly}(n/\epsilon))$. An easy argument in [Hancock and Mansour, 1991a] gives precisely the desired result; they show that for any monotone read-$k$ DNF $f$ that has $\Pr[f(x) = 0] = p$, every term $T$ that contains $\mathcal{C}$ variables satisfies $\Pr[T$ is true and all other terms are false$] \geq p2^{-k|\mathcal{C}|}$. Since we have $p \geq \frac{\epsilon}{2}$ and $\mathcal{C} \leq \kappa = O(\log \frac{n}{\epsilon})$, we obtain $\alpha \geq \Omega(1/\text{poly}(n/\epsilon))$ as required. So we may apply Theorem 12 to conclude that our algorithm learns $f$ in time poly$(n, 1/\epsilon, \log(1/\delta))$.

# Chapter 5

# Optimal Cryptographic Hardness of Learning Monotone Functions

In the previous chapter, we saw that polynomial-size monotone DNF formulas could be significantly easier to learn than the entire class of polynomial-size DNF formulas. This trend appears in uniform distribution learning for several other classes, such as polynomial-size decision trees.

In this chapter, we will apply average-case analysis to prove a cryptographic hardness result for learning monotone functions, the first hardness result of this kind for monotone functions.

## 5.1 Introduction

### 5.1.1 Background and Motivation

**Monotonicity makes learning easier** For many classes of functions, uniform distribution learning algorithms have been devised that substantially improve on a naive exponential-time approach to learning via brute-force search. However, despite intensive efforts, researchers have not yet obtained poly($n$)-time learning algorithms in this model for various simple classes of functions. Interestingly, in many of these cases restricting the class of functions to the corresponding class of *monotone* functions has led to more

efficient—sometimes poly($n$)-time—algorithms. We list some examples:

1. A simple algorithm learns monotone $O(\log n)$-juntas to perfect accuracy in poly($n$) time, and a more complex algorithm [Bshouty and Tamon, 1996] learns monotone $\tilde{O}(\log^2(n))$-juntas to any constant accuracy in poly($n$) time. In contrast, the fastest known algorithm for learning arbitrary $k$-juntas runs in time $n^{.704k}$ [Mossel *et al.*, 2004b].

2. The fastest known uniform distribution learning algorithm for the general class of $s$-term DNF, due to Verbeurgt [Verbeurgt, 1990], runs in time $n^{O(\log s)}$ to learn to any constant accuracy. In contrast, [Servedio, 2004a] gives an algorithm that runs in time $s^{O(\log s)}$ for $s$-term monotone DNF. Thus, the class of $2^{O(\sqrt{\log n})}$-term monotone DNF can be learned to any constant accuracy in poly($n$) time, but no such result is known for $2^{O(\sqrt{\log n})}$-term general DNF.

3. The fastest known algorithm for learning poly($n$)-size general decision trees to constant accuracy takes $n^{O(\log n)}$ time (following from [Verbeurgt, 1990]), but poly($n$)-size decision trees that compute monotone functions can be learned to any constant accuracy in poly($n$) time [O'Donnell and Servedio, 2007].

4. No poly($n$)-time algorithm can learn the general class of all Boolean functions on $\{0,1\}^n$ to accuracy better than $1/2 + \text{poly}(n)/2^n$, but a simple polynomial-time algorithm can learn the class of all monotone Boolean functions to accuracy $1/2 + \Omega(1/\sqrt{n})$ [Blum *et al.*, 1998]. Recent work [O'Donnell and Wimmer, 2009] gives a simple polynomial-time algorithm that learns to accuracy $1/2 + \Omega(\log n/\sqrt{n})$. We note also that the result of [Bshouty and Tamon, 1996] mentioned above follows from a $2^{\tilde{O}(\sqrt{n})}$-time algorithm for learning arbitrary monotone functions on $n$ variables to constant accuracy. (It is easy to see that no comparable algorithm can exist for learning arbitrary Boolean functions to constant accuracy.)

**Cryptography and hardness of learning** Essentially all known representation-independent hardness of learning results (*i.e.,* , results that apply to learning algorithms that do not have any restrictions on the syntactic form of the hypotheses they output) rely on some

cryptographic assumption, or an assumption that easily implies a cryptographic primitive. For example, under the assumption that certain subset sum problems are hard on average, Kharitonov [Kharitonov, 1995] showed that the class $\mathsf{AC}^1$ of logarithmic-depth, polynomial-size Boolean circuits (circuits with AND, OR, and NOT gates) is hard to learn under the uniform distribution. Subsequently Kharitonov [Kharitonov, 1993] showed that if factoring Blum integers is $2^{n^\epsilon}$-hard for some fixed $\epsilon > 0$, then even the class $\mathsf{AC}^0$ of constant-depth, polynomial-size Boolean circuits similarly cannot be learned in polynomial time under the uniform distribution. In later work, Naor and Reingold [Naor and Reingold, 2004] gave constructions of pseudorandom functions with very low circuit complexity. Their results imply that if factoring Blum integers is super-polynomially hard, then even depth-5 $\mathsf{TC}^0$ circuits cannot be learned in polynomial time under the uniform distribution. ($\mathsf{TC}^0$ circuits are Boolean circuits that can also use MAJ gates. The value of a MAJ gate is one if at least half of its inputs are one, and zero otherwise.) We note that all of these hardness results apply even to algorithms that may make black-box "membership queries" to obtain the value $f(x)$ for inputs $x$ of their choosing.

**Monotonicity versus cryptography?** Given that cryptography precludes efficient learning while monotonicity seems to make efficient learning easier, it is natural to investigate how these phenomena interact. One could argue that prior to the current work there was something of a mismatch between known positive and negative results for uniform-distribution learning: as described above, a fairly broad range of polynomial-time learning results have been obtained for various classes of monotone functions, but there were no corresponding computational hardness results for monotone functions. Can all monotone Boolean functions computed by polynomial-size circuits be learned to 99% accuracy in polynomial time from uniform random examples? As far as we are aware, prior to our work answers were not known even to such seemingly basic questions about learning monotone functions as this one. This gap in understanding motivated the research presented in this chapter (which, as we describe below, lets us answer "no" to the above question in a strong sense).

### 5.1.2 Our results and techniques: cryptography trumps monotonicity

We present several different constructions of "simple" (polynomial-time computable) monotone Boolean functions and prove that these functions are hard to learn under the uniform distribution, even if membership queries are allowed. We now describe our main results, followed by a high-level description of how we obtain them.

Blum, Burch, and Langford [Blum *et al.*, 1998] showed that arbitrary monotone functions cannot be learned to accuracy better than $1/2 + O(\log n/\sqrt{n})$ by any algorithm that makes $\text{poly}(n)$ many membership queries. This is an information-theoretic bound that is proved using randomly generated monotone DNF formulas of size (roughly) $n^{\log n}$ that are not polynomial-time computable. A natural goal is to obtain *computational* lower bounds for learning polynomial-time computable monotone functions that match, or nearly match, this level of hardness (which is close to optimal by the $(1/2 + \Omega(1/\sqrt{n}))$-accuracy algorithm of Blum *et al.* described above). We prove near-optimal hardness for learning polynomial-size monotone Boolean circuits (circuits with AND and OR gates):

**Theorem 13** (informal statement)**.** *If one-way functions exist, then there is a class of* $\text{poly}(n)$*-size monotone Boolean circuits that cannot be learned to accuracy* $1/2 + 1/n^{1/2-\epsilon}$ *for any fixed* $\epsilon > 0$*.*

Our approach yields even stronger lower bounds if we make stronger assumptions:

- Assuming the existence of sub-exponential one-way functions, we improve the bound on the accuracy to $1/2 + \text{polylog}(n)/\sqrt{n}$.
- Assuming the hardness of factoring Blum integers, our hard-to-learn functions may be computed in monotone $\mathsf{NC}^1$.
- Assuming that Blum integers are $2^{n^\epsilon}$-hard to factor on average (which is the same hardness assumption used in Kharitonov's work [Kharitonov, 1993]), we obtain a lower bound for learning constant-depth circuits of sub-polynomial size that almost matches the positive result from [Servedio, 2004a]. More precisely, we show that for any (sufficiently large) constant $d$, the class of monotone functions computed by depth-$d$ Boolean circuits of size $2^{(\log n)^{O(1)/(d+1)}}$ cannot be learned to accuracy 51% under the

| **Hardness assumption** | **Complexity of $f$** | **Accuracy bound** | **Ref.** |
|---|---|---|---|
| none | random $n^{\log n}$-term mono. DNF | $\frac{1}{2} + \frac{\omega(\log n)}{n^{1/2}}$ | [Blum *et al.*, 1998] |
| OWF (poly) | poly-size monotone circuits | $\frac{1}{2} + \frac{1}{n^{1/2-\epsilon}}$ | Thm. 13 |
| OWF ($2^{n^\alpha}$) | poly-size monotone circuits | $\frac{1}{2} + \frac{\mathrm{poly}(\log n)}{n^{1/2}}$ | Thm. 15 |
| factoring BI (poly) | monotone $\mathsf{NC}^1$-circuits | $\frac{1}{2} + \frac{1}{n^{1/2-\epsilon}}$ | Thm. 16 |
| factoring BI ($2^{n^\alpha}$) | depth-$d$, size $2^{(\log n)^{O(1)/(d+1)}}$ AND/OR/NOT circuits | $\frac{1}{2} + o(1)$ | Thm. 17 |

Figure 5.1: Summary of known hardness results for learning monotone Boolean functions. The meaning of each row is as follows: under the stated hardness assumption, there is a class of monotone functions computed by circuits of the stated complexity which no poly($n$)-time membership query algorithm can learn to the stated accuracy. In the first column, OWF and BI denote one-way functions and hardness of factoring Blum Integers respectively, and "poly" and "$2^{n^\alpha}$" means that the problems are intractable for poly($n$)- and $2^{n^\alpha}$-time algorithms respectively (for some fixed $\alpha > 0$). Recall that the poly($n$)-time algorithm of [Blum et al., 1998] for learning monotone functions implies that the best possible accuracy bound for monotone functions is $1/2 + \Omega(1)/n^{1/2}$.

uniform distribution in poly($n$) time. In contrast, the positive result of [Servedio, 2004a] shows that monotone functions computed by depth-$d$ Boolean circuits of size $2^{O((\log n)^{1/(d+1)})}$ can be learned to any constant accuracy in poly($n$) time.

These results are summarized in Figure 5.1.

**Proof techniques** A natural first approach is to try to replace the random $n^{\log n}$-term monotone DNFs constructed in [Blum *et al.*, 1998] by pseudorandom DNFs of polynomial size. We were not able to do this directly; indeed, as we discuss in Section 7, construct-

ing such DNFs seems closely related to an open problem of Goldreich, Goldwasser, and Nussboim [Goldreich *et al.*, 2003]. However, it turns out that a closely related approach does yield some results along the desired lines; in Section 5.4 we present and analyze a simple variant of the information-theoretic construction from [Blum *et al.*, 1998] and then show how to replace random choice by pseudorandom in this the variant. Because our variant gives a weaker quantitative bound on the information-theoretic hardness of learning than [Blum *et al.*, 1998], this gives a construction of polynomial-time-computable monotone functions that, assuming the existence of one-way functions, cannot be learned to accuracy $1/2 + 1/\text{polylog}(n)$ under the uniform distribution. While this answers the question posed above (even with "51%" in place of "99%"), the $1/\text{polylog}(n)$ factor is rather far from the $O(\log n/\sqrt{n})$ factor that one might hope for as described above.

In Section 5.2 we use a different construction to obtain much stronger quantitative results; another advantage of this second construction is that it enables us to show hardness of learning *monotone circuits* rather than just circuits computing monotone functions. We start with the simple observation that using standard tools it is easy to construct polynomial-size monotone circuits computing "slice" functions that are pseudorandom on the middle layer of the Boolean cube $\{0,1\}^n$. Such functions are easily seen to be mildly hard to learn, *i.e.,* , hard to learn to accuracy $1 - \Omega(1/\sqrt{n})$. We then use the elegant machinery of hardness amplification of monotone functions pioneered by O'Donnell [O'Donnell, 2004a] to amplify the hardness of this construction to near-optimal levels (as summarized in rows 2–4 of Figure 5.1). We obtain our result for constant-depth, sub-polynomial-size circuits (row 5 of Figure 5.1) by augmenting this approach with an argument that, at a high level, is similar to one used in [Allender *et al.*, 2006], by "scaling down" and modifying our hard-to-learn functions using a variant of Nepomnjaščiĭ's theorem [Nepomnjascii, 1970].

Recent work [Feldman *et al.*, 2010] generalizes this "amplifying hardness of learning" approach to obtain lower bounds for learning small-depth monotone formulas with Statistical Query algorithms.

### 5.1.3 Preliminaries

We consider Boolean functions of the form $f : \{0,1\}^n \to \{0,1\}$. We view $\{0,1\}^n$ as endowed with the natural partial order: $x \leq y$ if and only if $x_i \leq y_i$ for all $i = 1, \ldots, n$. A Boolean function $f$ is *monotone* if $x \leq y$ implies $f(x) \leq f(y)$.

Our work uses various standard definitions from the fields of circuit complexity, learning, and cryptographic pseudorandomness; and for completeness we recall this material below.

**Learning** As described earlier, all of our hardness results apply even to learning algorithms that may make *membership queries* (see Chapter 2 for a description of query learning in the uniform distribution setting). The goal of the learning algorithm is to construct a hypothesis $h$ so that $\Pr_x[h(x) \neq f(x)]$ is small, where the probability is taken over the uniform distribution. We shall only consider learning algorithms that are allowed to run in poly($n$) time, so the learning algorithm $L$ may be viewed as a probabilistic polynomial-time oracle machine that is given black-box access to the function $f$ and attempts to output a hypothesis $h$ with small error relative to $f$.

We establish that a class $\mathcal{C}$ of functions is hard to learn by showing that for a uniform random $f \in \mathcal{C}$, the expected error of any poly($n$)-time learning algorithm $L$ is close to $1/2$ when run with $f$ as the target function. Thus we bound the quantity

$$\Pr_{f \in \mathcal{C}, x \in \{0,1\}^n} \left[ L^f(1^n) \to h; h(x) = f(x) \right] \tag{5.1}$$

where the probability is also taken over any internal randomization of the learning algorithm $L$. We say that a *class $\mathcal{C}$ is hard to learn to accuracy* $1/2 + \epsilon(n)$ if, for every poly($n$)-time membership query learning algorithm $L$ (*i.e.,* , probabilistic polynomial-time oracle algorithm), we have that the above quantity Equation 5.1 is smaller than $1/2 + \epsilon(n)$ for all sufficiently large $n$. As noted in [Blum *et al.*, 1998], it is straightforward to transform a lower bound of this sort into a lower bound for the usual $\epsilon, \delta$ formulation of PAC learning.

**Circuit complexity** We shall consider various classes of circuits computing Boolean functions, including the classes $\mathsf{NC}^1$ (polynomial-size, logarithmic-depth, bounded fan-in Boolean circuits, $\mathsf{AC}^0$ (polynomial-size, constant-depth, unbounded fan-in Boolean circuits),

and $\mathsf{TC}^0$ (polynomial-size, constant-depth unbounded fan-in Boolean circuits with MAJ gates).

A circuit is said to be monotone if it is composed entirely of AND/OR gates with no negations. Every monotone circuit computes a monotone Boolean function, but of course non-monotone circuits may compute monotone functions as well. The famous result of [Razborov, 1985] shows that there are natural monotone Boolean functions (such as the perfect matching function) that can be computed by polynomial-size circuits but cannot be computed by quasi-polynomial-size monotone circuits, and Tardos [Tardos, 1988] observed that the separation can be increased to an exponential gap.

Thus, in general, it is a stronger result to show that a function can be computed by a small monotone circuit than to show that it is monotone and can be computed by a small circuit.

**Pseudorandom functions** Pseudorandom functions [Goldreich *et al.*, 1986] are the main cryptographic primitive that underlie our constructions. Fix $k(n) \leq n$, and let $\mathcal{G}$ be a family of functions $\left\{g : \{0,1\}^{k(n)} \to \{0,1\}\right\}$ each of which is computable by a circuit of size $\mathrm{poly}(k(n))$. We say that $\mathcal{G}$ is a $t(n)$-*secure pseudorandom function family* if the following condition holds: for any probabilistic $t(n)$-time oracle algorithm $A$, we have

$$\left| \Pr_{g \in \mathcal{G}}[A^g(1^n) \text{ outputs } 1] - \Pr_{g' \in \mathcal{G}'}[A^{g'}(1^n) \text{ outputs } 1] \right| \leq 1/t(n)$$

where $\mathcal{G}'$ is the class of all $2^{2^{k(n)}}$ functions from $\{0,1\}^{k(n)}$ to $\{0,1\}$ (so the second probability above is taken over the choice of a truly random function $g'$). Note that the purported distinguisher $A$ has oracle access to a function on $k(n)$ bits but is allowed to run in time $t(n)$.

It is well known that a pseudorandom function family that is $t(n)$-secure for all polynomials $t(n)$ can be constructed from any one-way function [Goldreich *et al.*, 1986; Håstad *et al.*, 1999]. We shall use the following folklore quantitative variant that relates the hardness of the one-way function to the security of the resulting pseudorandom function:

**Proposition 2.** *Fix $t(n) \geq \mathrm{poly}(n)$ and suppose there exist one-way functions that are hard to invert on average for $t(n)$-time adversaries. Then there exists a constant, $0 < c < 1$, such*

*that for any $k(n) \leq n$, there is a pseudorandom family $\mathcal{G}$ of functions $\{g : \{0,1\}^{k(n)} \to \{0,1\}\}$ that is $(t(k(n)))^c$-secure.*

## 5.2 Lower bounds via hardness amplification of monotone functions

In this section we prove our main hardness results, summarized in Figure 5.1, for learning various classes of monotone functions under the uniform distribution with membership queries.

Let us start with a high-level explanation of the overall idea. Inspired by the work on hardness amplification within NP initiated by O'Donnell [O'Donnell, 2004a; Trevisan, 2003; Healy *et al.*, 2006], we study constructions of the form

$$f(x_1, \ldots, x_m) = C\left(f'(x_1), \ldots, f'(x_m)\right)$$

where $C$ is a Boolean "combining function" with low noise stability (we give precise definitions later) that is both *efficiently computable* and *monotone*. Recall that O'Donnell showed that if $f'$ is weakly hard to compute and the combining function $C$ has low noise stability, then $f$ is very hard to compute. This result holds for general (not necessarily monotone) functions $C$, and thus generalizes Yao's XOR lemma, which addresses the case where $C$ is the XOR of $m$ bits (and hence has the lowest noise stability of all Boolean functions [O'Donnell, 2004a]).

Roughly speaking, we establish an analogue of O'Donnell's result for learning. Our analogue, given in Section 5.2.2, essentially states that for certain well-structured[1] functions $f'$ that are hard to learn to high accuracy, if $C$ has low noise stability then $f$ is hard to learn to accuracy even slightly better than $1/2$. As our ultimate goal is to establish that "simple" classes of monotone functions are hard to learn, we shall use this result with combining functions $C$ that are computed by "simple" monotone Boolean circuits. In order for the overall function $f$ to be monotone and efficiently computable, we need the initial $f'$ to be well-structured, monotone, efficiently computable, and hard to learn to high accuracy.

[1] As will be clear from the proof, we require that $f'$ be balanced and have a "hard-core set."

Such functions are easily obtained by a slight extension of an observation of Kearns, Li, and Valiant [Kearns *et al.*, 1994c]. They noted that the middle slice $f'$ of a random Boolean function on $\{0,1\}^k$ is hard to learn to accuracy greater than $1-\Theta\big(1/\sqrt{k}\big)$ [Blum *et al.*, 1998; Kearns *et al.*, 1994c]; by taking the middle slice of a *pseudorandom* function instead, we obtain an $f'$ with the desired properties. In fact, by a result of Berkowitz [Berkowitz, 1982] (see also [Valiant, 1986; Beals *et al.*, 1998]), this slice function is computable by a polynomial-size monotone circuit, so the overall hard-to-learn functions we construct are computed by polynomial-size monotone circuits.

**Organization**

In Section 5.2.2 we adapt the analysis from [O'Donnell, 2004a; Trevisan, 2003; Healy *et al.*, 2006] to reduce the problem of constructing hard-to-learn monotone Boolean functions to constructing monotone combining functions $C$ with low noise stability. In 5.2.3 we show how constructions and analyses in [O'Donnell, 2004a; Mossel and O'Donnell, 2003b] can be used to obtain a "simple" monotone combining function with low noise stability. In 5.2.4 we establish Theorems 14 and 15 (lines 2 and 3 of 5.1) by making different assumptions about the hardness of the initial pseudorandom functions. Finally, in 5.3 we establish Theorems 16 and 17 by making specific number theoretic assumptions (namely, the hardness of factoring Blum integers) to obtain hard-to-learn monotone Boolean functions that can be computed by very simple circuits.

### 5.2.1 Preliminaries

**Functions** Let $C : \{0,1\}^m \to \{0,1\}$ and $f' : \{0,1\}^k \to \{0,1\}$ be Boolean functions. We write $C \circ f'^{\otimes m}$ to denote the Boolean function over $(\{0,1\}^k)^m$ given by

$$C \circ f'^{\otimes m}(x) = C\left(f'(x_1), \ldots, f'(x_m)\right), \qquad \text{where } x = (x_1, \ldots, x_m).$$

For $g : \{0,1\}^k \to \{0,1\}$, we write $\mathsf{slice}(g)$ to denote the "middle slice" function:

$$\mathsf{slice}(g)(x) = \begin{cases} 1 & \text{if } |x| > \lfloor k/2 \rfloor, \\ g(x) & \text{if } |x| = \lfloor k/2 \rfloor, \\ 0 & \text{if } |x| < \lfloor k/2 \rfloor, \end{cases}$$

where $|x|$ denotes the number of ones in the string $x$. It is immediate that $\mathsf{slice}(g)$ is a monotone Boolean function for any $g$.

**Bias and noise stability** Following the analysis in [O'Donnell, 2004a; Trevisan, 2003; Healy *et al.*, 2006], we shall study the bias and noise stability of various Boolean functions. Specifically, we adopt the following notations and definitions from [Healy *et al.*, 2006]. The *bias* of a 0-1 random variable $X$ is defined to be

$$\mathrm{Bias}[X] \stackrel{\mathrm{def}}{=} \left|\Pr[X=0] - \Pr[X=1]\right| .$$

Recall that a probabilistic Boolean function $h$ on $\{0,1\}^k$ is a probability distribution over Boolean functions on $\{0,1\}^k$ (so for each input $x$, the output $h(x)$ is a 0-1 random variable). The *expected bias* of a probabilistic Boolean function $h$ is

$$\mathrm{ExpBias}[h] \stackrel{\mathrm{def}}{=} \mathrm{E}_x[\mathrm{Bias}[h(x)]] .$$

Let $C : \{0,1\}^m \to \{0,1\}$ be a Boolean function and $0 \le \delta \le 1/2$. The *noise stability of $C$ at noise rate $\delta$*, denoted $\mathrm{NoiseStab}_\delta[C]$, is defined to be

$$\mathrm{NoiseStab}_\delta[C] \stackrel{\mathrm{def}}{=} \mathrm{E}_{x,\eta}[C(x) \oplus C(x \oplus \eta)] = 2 \cdot \Pr_{x,\eta}[C(x) = C(x \oplus \eta)] - 1$$

where $x \in \{0,1\}^m$ is uniform random, $\eta \in \{0,1\}^m$ is a vector whose bits are each independently 1 with probability $\delta$, and $\oplus$ denotes bitwise XOR.

### 5.2.2 Hardness amplification for learning

Throughout this subsection we write $m$ for $m(n)$ and $k$ for $k(n)$. We shall establish the following:

**Lemma 19.** *Let $C : \{0,1\}^m \to \{0,1\}$ be a polynomial-time computable function. Let $\mathcal{G}'$ be the family of all $2^{2^k}$ functions from $\{0,1\}^k$ to $\{0,1\}$, where $n = mk$ and $k = \omega(\log n)$. Then the class*

$$\mathcal{C} = \{f = C \circ \mathsf{slice}(g)^{\otimes m} \mid g \in \mathcal{G}'\}$$

*of Boolean functions over $\{0,1\}^n$ is hard to learn to accuracy*

$$\frac{1}{2} + \frac{1}{2}\sqrt{\mathrm{NoiseStab}_{\Theta(1/\sqrt{k})}[C]} + \frac{1}{n^{\omega(1)}} .$$

This easily yields Corollary 3, which is an analogue of Lemma 19 with pseudorandom rather than truly random functions, and which we use to obtain our main hardness of learning results.

*Proof of Lemma 19.* Let $k, m$ be such that $mk = n$, and let $C : \{0,1\}^m \to \{0,1\}$ be a Boolean combining function. We prove the lemma by establishing an upper bound on the probability

$$\Pr_{g \in \mathcal{G}', x \in \{0,1\}^n} \left[ L^f(1^n) \to h;\ h(x) = f(x) \right] \tag{5.2}$$

where $L$ is an arbitrary probabilistic polynomial-time oracle machine (running in time $\mathrm{poly}(n)$ on input $1^n$) that is given oracle access to $f \stackrel{\text{def}}{=} C \circ \mathsf{slice}(g)^{\otimes m}$ and outputs some hypothesis $h : \{0,1\}^n \to \{0,1\}$.

We first observe that because $C$ is computed by a uniform family of circuits of size $\mathrm{poly}(m) \leq \mathrm{poly}(n)$, it is easy for a $\mathrm{poly}(n)$-time machine to simulate oracle access to $f$ if it is given oracle access to $g$. So, the probability in 5.2 is at most

$$\Pr_{g \in \mathcal{G}',\ x \in \{0,1\}^n} \left[ L^g(1^n) \to h;\ h(x) = (C \circ \mathsf{slice}(g)^{\otimes m})(x) \right]. \tag{5.3}$$

To analyze the above probability, suppose that in the course of its execution $L$ never queries $g$ on any of the inputs $x_1, \ldots, x_m \in \{0,1\}^k$, where $x = (x_1, \ldots, x_m)$. Then the *a posteriori* distribution of $g(x_1), \ldots, g(x_m)$ (for uniform random $g \in \mathcal{G}'$), given the responses to the queries of $L$ that it received from $g$, is identical to the distribution of $g'(x_1), \ldots, g'(x_m)$, where $g'$ is an independent uniform draw from $\mathcal{G}'$: both distributions are uniform random over $\{0,1\}^m$. (Intuitively, this just means that if $L$ never queries the random function $g$ on any of $x_1, \ldots, x_m$, then giving $L$ oracle access to $g$ does not help it predict the value of $f$ on $x = (x_1, \ldots, x_m)$.) As $L$ runs in $\mathrm{poly}(n)$ time, for any fixed $x_1, \ldots, x_m$ the probability that $L$ queried $g$ on any of $x_1, \ldots, x_m$ is at most $m \cdot \mathrm{poly}(n)/2^k$. Hence 5.3 is bounded by

$$\Pr_{g, g' \in \mathcal{G}',\ x \in \{0,1\}^n} \left[ L^g(1^n) \to h;\ h(x) = (C \circ \mathsf{slice}(g')^{\otimes m})(x) \right] + \frac{m \cdot \mathrm{poly}(n)}{2^k}. \tag{5.4}$$

The first summand in 5.4 is the probability that $L$ correctly predicts the value $C \circ \mathsf{slice}(g')^{\otimes m}(x)$, given oracle access to $g$, where $g$ and $g'$ are independently random functions and $x$ is uniform over $\{0,1\}^n$. It is clear that the best possible strategy for $L$ is to use a maximum likelihood algorithm, *i.e.,* , predict according to the function $h$ that, for any fixed input $x$,

outputs 1 if and only if the random variable $(C \circ \mathsf{slice}(g')^{\otimes m})(x)$ (which we emphasize has randomness over the choice of $g'$) is biased towards 1. The expected accuracy of this $h$ is precisely

$$\frac{1}{2} + \frac{1}{2}\,\mathrm{ExpBias}[C \circ \mathsf{slice}(g')^{\otimes m}]\,. \tag{5.5}$$

Now fix

$$\delta \stackrel{\text{def}}{=} \frac{1}{2^k}\binom{k}{\lfloor k/2 \rfloor} = \Theta\left(\frac{1}{\sqrt{k}}\right)$$

to be the fraction of inputs in the "middle slice" of $\{0,1\}^k$. We observe that the probabilistic function $\mathsf{slice}(g')$ (where $g'$ is truly random) is "$\delta$-random" in the sense of Definition 3.1 of [Healy *et al.*, 2006], meaning that it is balanced, truly random on inputs in the middle slice, and deterministic on all other inputs. This means that we may apply the following technical lemma (Lemma 3.7 from [Healy *et al.*, 2006], see also [O'Donnell, 2004a]):

**Lemma 20.** *Let $h : \{0,1\}^n \to \{0,1\}$ be a function that is $\delta$-random. Then*

$$\mathrm{ExpBias}[C \circ h^{\otimes m}] \leq \sqrt{\mathrm{NoiseStab}_\delta[C]}\,.$$

Applying this lemma to the function $\mathsf{slice}(g')$ we obtain

$$\mathrm{ExpBias}[C \circ \mathsf{slice}(g')^{\otimes m}] \leq \sqrt{\mathrm{NoiseStab}_\delta[C]}\,. \tag{5.6}$$

Combining 5.4, 5.5 and 5.6 and recalling that $k = \omega(\log n)$, we obtain 19. ■

**Corollary 3.** *Let $C : \{0,1\}^m \to \{0,1\}$ be a polynomial-time computable function. Let $\mathcal{G}$ be a pseudorandom family of functions from $\{0,1\}^k$ to $\{0,1\}$ that are secure against $\mathrm{poly}(n)$-time adversaries, where $n = mk$ and $k = \omega(\log n)$. Then the class*

$$\mathcal{C} = \left\{ f = C \circ \mathsf{slice}(g)^{\otimes m} \mid g \in \mathcal{G} \right\}$$

*of Boolean functions over $\{0,1\}^n$ is hard to learn to accuracy*

$$\frac{1}{2} + \frac{1}{2}\sqrt{\mathrm{NoiseStab}_{\Theta(1/\sqrt{k})}[C]} + \frac{1}{n^{\omega(1)}}\,.$$

*Proof.* The corollary follows from the fact that 5.3 must differ from its pseudorandom counterpart,

$$\Pr_{g \in \mathcal{G},\, x \in \{0,1\}^n}\left[L^g(1^n) \to h;\ h(x) = (C \circ \mathsf{slice}(g)^{\otimes m})(x)\right], \tag{5.7}$$

by less than any fixed $1/\operatorname{poly}(n)$. Otherwise, we would easily obtain a $\operatorname{poly}(n)$-time distinguisher that, given oracle access to $g$, runs $L$ to obtain a hypothesis $h$ and checks whether

$$h(x) = (C \circ \mathsf{slice}(g)^{\otimes m})(x)$$

for a random $x$ to determine whether $g$ is drawn from $\mathcal{G}$ or $\mathcal{G}'$. ■

By instantiating Corollary 3 with a "simple" monotone function $C$ having low noise stability, we obtain strong hardness results for learning simple monotone functions. We exhibit such a function $C$ in the next section.

### 5.2.3 A simple monotone combining function with low noise stability

In this section we combine known results of [O'Donnell, 2004a; Mossel and O'Donnell, 2003b] to obtain:

**Proposition 3.** *Given a value $k$, let $m = 3^\ell d 2^d$ for $\ell, d$ satisfying $3^\ell \leq k^6 < 3^{\ell+1}$ and $d \leq O(k^{.35})$. Then there exists a monotone function $C : \{0,1\}^m \to \{0,1\}$ computed by a uniform family of $\operatorname{poly}(m)$-size, $\log(m)$-depth monotone circuits such that*

$$\operatorname{NoiseStab}_{\Theta(1/\sqrt{k})}[C] = O\left(\frac{k^6 \log m}{m}\right). \tag{5.8}$$

Note that in this proposition we may have $m$ as large as $2^{\Theta(k^{.35})}$ but not larger. O'Donnell [O'Donnell, 2004a] established the lower bound

$$\operatorname{NoiseStab}_{\Theta\left(1/\sqrt{k}\right)}[C] = \Omega\left(\frac{\log^2 m}{m}\right)$$

for every monotone $m$-variable function $C$, so the above upper bound is fairly close to the best possible (within a $\operatorname{polylog}(m)$ factor if $m = 2^{k^{\Theta(1)}}$).

Following [O'Donnell, 2004a; Healy *et al.*, 2006], we use the "recursive majority of 3" function and the "tribes" function in our construction. We require the following results on noise stability:

**Lemma 21** ([O'Donnell, 2004a])**.** *Let* Rec-Maj-$3_\ell : \{0,1\}^{3^\ell} \to \{0,1\}$ *be defined as follows: for $x = (x^1, x^2, x^3)$ where each $x^i \in \{0,1\}^{3^{\ell-1}}$,*

$$\text{Rec-Maj-}3_\ell(x) \stackrel{def}{=} \operatorname{Maj}\left(\text{Rec-Maj-}3_{\ell-1}(x^1), \text{Rec-Maj-}3_{\ell-1}(x^2), \text{Rec-Maj-}3_{\ell-1}(x^3)\right).$$

*Then for $\ell \geq \log_{1.1}(1/\delta)$, we have*

$$\mathrm{NoiseStab}_\delta[\text{Rec-Maj-3}_\ell] \leq \delta^{-1.1}(3^\ell)^{-.15}.$$

**Lemma 22** ([Mossel and O'Donnell, 2003b])**.** *Let* $\mathrm{Tribes}_d : \{0,1\}^{d2^d} \to \{0,1\}$ *denote the "tribes" function on* $d2^d$ *variables, i.e., , the read-once* $2^d$*-term monotone d-DNF*

$$\mathrm{Tribes}_d(x_1, \ldots, x_{d2^d}) \stackrel{def}{=} (x_1 \wedge \cdots \wedge x_d) \vee (x_{d+1} \wedge \cdots \wedge x_{2d}) \vee \cdots.$$

*Then if* $\eta = O(1/d)$*, we have*

$$\mathrm{NoiseStab}_{\frac{1-\eta}{2}}[\mathrm{Tribes}_d] = O\Big(\frac{\eta d^2}{d2^d}\Big) = O\Big(\frac{1}{2^d}\Big).$$

**Lemma 23** ([O'Donnell, 2004a])**.** *If* $h$ *is a balanced Boolean function and* $\psi : \{0,1\}^r \to \{0,1\}$ *is arbitrary, then for any* $\delta$ *we have*

$$\mathrm{NoiseStab}_\delta[\psi \circ h^{\otimes r}] = \mathrm{NoiseStab}_{\frac{1}{2} - \frac{\mathrm{NoiseStab}_\delta[h]}{2}}[\psi].$$

*Proof of Proposition 3.* We take $C$ to be $\mathrm{Tribes}_d \circ \text{Rec-Maj-3}_\ell^{\otimes d2^d}$. Given that Rec-Maj-$3_\ell$ is balanced, by Lemma 23 we have

$$\mathrm{NoiseStab}_\delta[C] = \mathrm{NoiseStab}_{\frac{1}{2} - \frac{\mathrm{NoiseStab}_\delta[\text{Rec-Maj-3}_\ell]}{2}}[\mathrm{Tribes}_d].$$

Setting $\delta = \Theta\big(1/\sqrt{k}\big)$ and recalling that $3^\ell \leq k^6$, we have $\ell \geq \log_{1.1}(1/\delta)$ so we may apply Lemma 21 to obtain

$$\mathrm{NoiseStab}_{\Theta(1/\sqrt{k})}[\text{Rec-Maj-3}_\ell] \leq \Theta\left(\left(\sqrt{k}\right)^{1.1}\right)\left(k^6\right)^{-.15} = O\left(k^{-.35}\right).$$

As $O(k^{-.35}) \leq O(1/d)$, we may apply Lemma 22 with the previous inequalities to obtain

$$\mathrm{NoiseStab}_{\Theta(1/\sqrt{k})}[C] = O\left(\frac{1}{2^d}\right).$$

The bound 5.8 follows from a rearrangement of the bounds on $k, m, d$ and $\ell$. It is easy to see that $C$ can be computed by monotone circuits of depth $O(\ell) = O(\log m)$ and size $\mathrm{poly}(m)$. This completes the proof. ■

### 5.2.4 Nearly optimal hardness of learning polynomial-size monotone circuits

Given a value of $k$, let $m = 3^\ell d 2^d$ for $\ell, d$ as in 3. Let $\mathcal{G}$ be a pseudorandom family of functions $\left\{g : \{0,1\}^k \to \{0,1\}\right\}$ secure against poly$(n)$-time adversaries, where $n = mk$. Given that we have $k = \omega(\log n)$, we may apply Corollary 3 with the combining function from Proposition 3 and conclude that the class $\mathcal{C} = \{C \circ \mathsf{slice}(g)^{\otimes m} \mid g \in \mathcal{G}\}$ is hard to learn to accuracy

$$\frac{1}{2} + O\left(\frac{k^3\sqrt{\log m}}{\sqrt{m}}\right) + o(1/n) \leq \frac{1}{2} + O\left(\frac{k^{3.5}\sqrt{\log n}}{\sqrt{n}}\right). \tag{5.9}$$

We claim that the functions in $\mathcal{C}$ can, in fact, be computed by poly$(n)$-size monotone circuits. This follows from a result of Berkowitz [Berkowitz, 1982] that states that if a $k$-variable slice function is computed by a Boolean circuit of size $s$ and depth $d$, then it is also computed by a monotone Boolean circuit with MAJ gates of size $O(s + k)$ and depth $d + 1$. Combining these monotone circuits for $\mathsf{slice}(g)$ with the monotone circuit for $C$, we obtain a poly$(n)$-size monotone circuit for each function in $\mathcal{C}$.

By making various different assumptions on the hardness of one-way functions, Proposition 2 lets us obtain different quantitative relationships between $k$ (the input length for the pseudorandom functions) and $n$ (the running time of the adversaries against which they are secure), and thus different quantitative hardness results from Equation 5.9 above:

**Theorem 14.** *Suppose that standard one-way functions exist. Then for any constant $\epsilon > 0$ there is a class $\mathcal{C}$ of poly$(n)$-size monotone circuits that is hard to learn to accuracy $1/2 + 1/n^{1/2-\epsilon}$.*

*Proof.* If poly$(n)$-hard one-way functions exist then we may take $k = n^c$ in Proposition 2 for an arbitrarily small constant $c$; this corresponds to taking $d = \gamma \log k$ for $\gamma$ a large constant in 3. The claimed bound on Equation 5.9 easily follows. (We note that while not every $n$ is of the required form $mk = 3^\ell d 2^d k$, it is not difficult to see that this and our subsequent theorems hold for all (sufficiently large) input lengths $n$ by padding the hard-to-learn functions.) ■

**Theorem 15.** *Suppose that sub-exponentially hard ($2^{n^\alpha}$ for some fixed $\alpha > 0$) one-way*

*functions exist. Then there is a class $\mathcal{C}$ of poly$(n)$-size monotone circuits that is hard to learn to accuracy $1/2 + \text{polylog}(n)/\sqrt{n}$.*

*Proof.* As above, but now we take $k = \log^{\gamma} n$ for some sufficiently large constant $\gamma$ (*i.e.,* , $d = c \log k$ for a small constant $c$). ■

## 5.3 Hardness of Learning Simple Circuits

In this section we obtain hardness results for learning very simple classes of circuits computing monotone functions under a concrete hardness assumption for a specific computational problem, namely factoring Blum integers. Naor and Reingold [Naor and Reingold, 2004] showed that if factoring Blum integers is computationally hard then there is a pseudorandom function family, which we denote $\mathcal{G}^\star$, that is computable in $\mathsf{TC}^0$. From this it easily follows that the functions $\{\mathsf{slice}(g) \mid g \in \mathcal{G}^\star\}$ are also computable in $\mathsf{TC}^0$.

We now observe that the result of Berkowitz [Berkowitz, 1982] mentioned earlier for converting slice circuits into monotone circuits applies not only to Boolean circuits, but also to $\mathsf{TC}^0$ circuits.

This means that the functions in $\{\mathsf{slice}(g) \mid g \in \mathcal{G}^\star\}$ are in fact computable in monotone $\mathsf{TC}^0$, *i.e.,* , by polynomial-size, constant-depth circuits composed only of AND/OR/MAJ gates. As the majority function can be computed by polynomial-size, $O(\log n)$-depth monotone Boolean circuits, (see, *e.g.,* , [Ajtai *et al.*, 1983]), the functions in $\{\mathsf{slice}(g) \mid g \in \mathcal{G}^\star\}$ are computable by $O(\log n)$-depth monotone Boolean circuits. Finally, using the parameters in 14 we have a combining function $C$ that is a $O(\log n)$-depth poly-size monotone Boolean circuit, which implies the following lemma:

**Lemma 24.** *Let $C$ be the monotone combining function from 3 and $\mathcal{G}^\star$ be a family of pseudorandom functions computable in $\mathsf{TC}^0$. Then every function in $\{C \circ \mathsf{slice}(g)^{\otimes m} \mid g \in \mathcal{G}^\star\}$ is computable in monotone $\mathsf{NC}^1$.*

This directly yields a hardness result for learning monotone $\mathsf{NC}^1$ circuits (the fourth line of Figure 5.1):

**Theorem 16.** *If factoring Blum integers is hard on average for any poly($n$)-time algorithm, then for any constant $\epsilon > 0$ there is a class $\mathcal{C}$ of poly($n$)-size monotone $\mathsf{NC}^1$ circuits that is hard to learn to accuracy $1/2 + 1/n^{1/2-\epsilon}$.*

Now we show that under a stronger but still plausible assumption on the hardness of factoring Blum integers, we get a hardness result for learning a class of *constant-depth* monotone circuits that is very close to a class known to be learnable to any constant accuracy in poly($n$) time. Suppose that $n$-bit Blum integers are $2^{n^\alpha}$-hard to factor on average for some fixed $\alpha > 0$ (which is the same hardness assumption that was earlier used by Kharitonov [Kharitonov, 1993]). This means there exist $2^{n^{\alpha/2}}$-secure pseudorandom functions that are computable in $\mathsf{TC}^0$. Using such a family of functions in place of $\mathcal{G}^\star$ in the construction for the preceding theorem and fixing $\epsilon = 1/3$, we obtain:

**Lemma 25.** *Assume that Blum integers are $2^{n^\alpha}$-hard to factor on average. Then there is a class $\mathcal{C}$ of poly($n$)-size monotone $\mathsf{NC}^1$ circuits that is hard for any $2^{n^{\alpha/20}}$-time algorithm to learn to accuracy $1/2 + 1/n^{1/6}$.*

Now we "scale down" this class $\mathcal{C}$ as follows. Let $n'$ be such that $n' = (\log n)^\kappa$ for a suitable constant $\kappa > 20/\alpha$, and let us substitute $n'$ for $n$ in the construction of the previous lemma; we call the resulting class of functions $\mathcal{C}'$. In terms of $n$, the functions in $\mathcal{C}'$ (which are functions over $\{0,1\}^n$ that only depend on the first $n'$ variables) are computed by $(\log n)^{O(\kappa)}$-size, $O(\log\log n)$-depth monotone circuits whose inputs are the first $(\log n)^\kappa$ variables in $x_1, \ldots, x_n$. We moreover have that $\mathcal{C}'$ is hard for any $2^{(n')^{\alpha/20}} = 2^{(\log n)^{\kappa\alpha/20}} = \omega(\text{poly}(n))$-time algorithm to learn to some accuracy

$$\frac{1}{2} + \frac{1}{(n')^{1/6}} = \frac{1}{2} + o(1)\,.$$

We now recall the following variant of Nepomnjaščiĭ's theorem that is implicit in [Allender *et al.*, 2006].

**Lemma 26.** *For every language $\mathcal{L} \in \mathsf{NL}$, for all sufficiently large constant $d$ there are $\mathsf{AC}^0_d$ circuits of size $2^{n^{O(1)/(d+1)}}$ that recognize $\mathcal{L}$.*

As every function in $\mathcal{C}'$ can be computed in $\mathsf{NC}^1$, which is contained in $\mathsf{NL}$, combining Lemma 26 with the paragraph that proceeds it, we obtain the following theorem (final line of Figure 5.1):

**Theorem 17.** *Suppose that Blum integers are sub-exponentially hard to factor on average. Then there is a class $\mathcal{C}$ of monotone functions that is hard for any poly$(n)$-time algorithm to learn to accuracy $1/2 + o(1)$ and that, for all sufficiently large constant $d$, are computed by $\mathsf{AC}^0_d$ circuits of size $2^{(\log n)^{O(1)/(d+1)}}$.*

This final hardness result is of interest because it is known that constant-depth circuits of only slightly smaller size *can* be learned to any constant accuracy in poly$(n)$ time under the uniform distribution (without needing membership queries):

**Theorem 18** ([Servedio, 2004a] Corollary 2)**.** *For all $d \geq 2$, the class of $\mathsf{AC}^0_d$ circuits of size $2^{O((\log n)^{1/(d+1)})}$ that compute monotone functions can be learned to any constant accuracy $1 - \epsilon$ in* poly$(n)$*-time.*

Theorem 17 is thus nearly optimal in terms of the size of the constant-depth circuits for which it establishes hardness of learning.

## 5.4 A computational analogue of the Blum-Burch-Langford lower bound

In this section we first present a simple variant of the lower bound construction in [Blum *et al.*, 1998], obtaining an information-theoretic lower bound on the learnability of the general class of all monotone Boolean functions. The quantitative bound our variant achieves is weaker than that of [Blum *et al.*, 1998], but has the advantage that it can be easily derandomized. Indeed, as mentioned in Section 7 (and further discussed below), our construction uses a certain probability distribution over monotone DNFs, such that a typical random input $x$ satisfies only poly$(n)$ many "candidate terms" (which are terms that may be present in a random DNF drawn from our distribution). By selecting terms for inclusion in the DNF in a pseudorandom rather than truly random way, we obtain a class of poly$(n)$-size monotone circuits that is hard to learn to accuracy $1/2 + 1/\text{polylog}(n)$ (assuming one-way functions exist).

Below we start with an overview of why it is difficult to obtain a computational analogue of the exact construction of [Blum *et al.*, 1998] using the pseudorandom approach sketched

above, and the idea behind our variant, which overcomes this difficulty. We then provide our information theoretic construction and analysis, followed by its computational analogue.

### 5.4.1 Idea

Recall the information-theoretic lower bound from [Blum *et al.*, 1998]. It works by defining a distribution $P_s$ over monotone functions of the form $\{0,1\}^n \rightarrow \{0,1\}$ as follows. (Here $s$ is a numerical parameter which should be thought of as the number of membership queries that a learning algorithm is allowed to make.) Take $t = \log(3sn)$. A draw from $P_s$ is obtained by randomly including each length-$t$ monotone term in the DNF independently with probability $p'$, where $p'$ is chosen so that the function is expected to be balanced on "typical inputs" (more precisely, on inputs with exactly $n/2$ ones). The naive idea for derandomizing this construction is to simply use a pseudorandom function with bias $p'$ to determine whether each possible term of size $t$ should be included or excluded in the DNF. However, there is a problem with this approach: we do not know an efficient way to determine whether a typical example $x$ (with, say, $n/2$ ones) has any of its $\binom{n/2}{t}$ candidate terms (each of which is pseudorandomly present/not present in $f$) actually present in $f$, so we do not know how to evaluate $f$ on a typical input $x$ in less than $\binom{n/2}{t}$ time.

We get around this difficulty by instead considering a new distribution of random monotone DNFs. In our construction we will again use a random function with bias $p$ to determine whether each possible term of length $t$ is present in the DNF. However, in our construction, a typical example $x$ will have only a polynomial number of candidate terms that could be satisfied, and thus it is possible to check all of them and evaluate the function in poly($n$) time.

The main difficulty of this approach is to ensure that although a typical example has only a polynomial number of candidate terms, the function is still hard to learn in polynomial time. We achieve this by partitioning the variables into blocks of size $k$ and viewing each block as a "super-variable" (corresponding to the AND of all $k$ variables in the block). We then construct the DNF by randomly choosing length-$t$ terms over these super-variables. It is not difficult to see that with this approach, we can equivalently view our problem as learning a $t$-DNF $f$ with terms chosen as above, where each of the $n/k$ variables is drawn

from a product distribution with bias $1/2^k$. By fine-tuning the parameters that determine $t$ (the size of each term of the DNF) and $k$ (the size of the partitions), we are able to achieve an information-theoretic lower bound showing that this distribution over monotone functions is hard to learn to accuracy $1/2 + o(1)$.

### 5.4.2 Construction

Let us partition the variables $x_1, \ldots, x_n$ into $m = n/k$ blocks $B_1, \ldots, B_m$ of $k$ variables each. Let $X_i$ denote the conjunction of all $k$ variables in $B_i$ ($X_1, \ldots, X_m$ are the super-variables). The following is a description of our distribution $P$ over monotone functions. A function $f$ is drawn from $P$ as follows (we fix the values of $k, t$ later):

- Construct a monotone DNF $f_1$ as follows: each possible conjunction of $t$ super-variables chosen from $\{X_1, \ldots, X_m\}$ is placed in the target function $f_1$ independently with probability $p$, where $p$ is defined as the solution to:

$$(1-p)^{\binom{m/2^k}{t}} = \frac{1}{2}\,. \tag{5.10}$$

  Note that for a uniform choice of $x \in \{0,1\}^n$, we expect $m/2^k$ ones in the corresponding "super-assignment" $X = (X_1, \ldots, X_m)$, and any superassignment with this many ones will be satisfied by $\binom{m/2^k}{t}$ many terms. Thus $p$ is chosen such that a "typical" example $X$, with $m/2^k$ ones, has probability $1/2$ of being labeled positive under $f_1$.

- Let

$$f(x) = \begin{cases} f_1(x) & \text{if the number of supervariables satisfied in } x \text{ is at most } m/2^k + (m/2^k)^{2/3}, \\ 1 & \text{otherwise.} \end{cases}$$

Note that because of the final step of the construction, the function $f$ is not actually a DNF (though it is a monotone function). Intuitively, the final step is there because if too many supervariables were satisfied in $x$, there could be too many (more than $\mathrm{poly}(n)$) candidate terms to check, and we would not be able to evaluate $f_1$ efficiently. We will show later that the probability that the number of supervariables satisfied in $x$ is greater than $m/2^k + (m/2^k)^{2/3}$ is at most $2e^{-(m/2^k)^{1/3}/3} = 1/n^{\omega(1)}$, and thus the function $f$ is $1/n^{\omega(1)}$-close to $f_1$; so hardness of learning results established for the random DNFs $f_1$ carry over

to the actual functions $f$. For most of our discussion we shall refer to $P$ as a distribution over DNFs, meaning the functions $f_1$.

### 5.4.3 Information-theoretic lower bound

As discussed previously, we view the distribution $P$ defined above as a distribution over DNFs of terms of size $t$ over the supervariables. Each possible combination of $t$ supervariables appears in $f_1$ independently with probability $p$ and the supervariables are drawn from a product distribution that is 1 with probability $1/2^k$ and 0 with probability $1 - 1/2^k$. We first observe that learning $f$ over the supervariables drawn from the product distribution is equivalent to learning the original function over the original variables. This is because if we are given the original membership query oracle for $n$-bit examples we can simulate answers to membership queries on $m$-bit "supervariable" examples and vice versa. Thus we henceforth analyze the product distribution.

We follow the proof technique of [Blum *et al.*, 1998]. To simplify our analysis, we consider an "augmented" oracle, as in [Blum *et al.*, 1998]. Given a query $X$, with ones in positions indexed by the set $S_X$, the oracle will return the first conjunct in lexicographic order that appears in the target function and is satisfied by $X$. Additionally, the oracle returns 1 if $X$ is positive and 0 if $X$ is negative. (So instead of just giving a single bit as its response, if the example is a positive one the oracle tells the learner the lexicographically first term in the target DNF that is satisfied.) Clearly, lower bounds for this augmented oracle imply the same bounds for the standard oracle.

We are interested in analyzing $P_s$, the conditional distribution over functions drawn from the initial distribution $P$ that are consistent with the information learned by $A$ in the first $s$ queries. We can think of $P_s$ as a vector $V_s$ of $\binom{m}{t}$ elements, one for each possible conjunct of size $t$. Initially, each element of the vector contains $p$, the probability that the conjunct is in the target function. When a query is made, the oracle examines one by one the entries that satisfy $X$. For each entry having value $p$, we can think of the oracle as flipping a coin and replacing the entry by 0 with probability $1-p$ and by 1 with probability $p$. After $s$ queries, $V_s$ will contain some entries set to 0, some set to 1 and the rest set to $p$. Because $V_s$ describes the conditional distribution $P_s$ given the queries made so far, the

Bayes-optimal prediction for an example $X$ is simply to answer 1 if $V_s(X) \geq 1/2$ and 0 otherwise.

We now analyze $V_s(X)$, the conditional probability over functions drawn from $P$ that are consistent with the first $s$ queries that a random example, $X$, drawn from the distribution, evaluates to 1, given the answers to the first $s$ queries. We will show that for $s = \text{poly}(n)$, for $X$ drawn from the product distribution on $\{0,1\}^m$, with probability at least $1 - 1/n^{\omega(1)}$ the value $V_s(X)$ lies in $1/2 \pm 1/\log n$. This is easily seen to give a lower bound of the type we require.

Following [Blum *et al.*, 1998], we first observe that after $s$ queries there can be at most $s$ entries set to one in the vector $V_s$. We shall also use the following lemma from [Blum *et al.*, 1998]:

**Lemma 27** ([Blum *et al.*, 1998])**.** *After $s$ queries, with probability $1 - e^{-s/4}$, there are at most $2s/p$ zeros in $V_s$.*

We thus may henceforth assume that there are at most $2s/p$ zeros in $V_s$.

We now establish the following, which is an analogue (tailored to our setting) of Claim 3 of [Blum *et al.*, 1998]:

**Lemma 28.** *For any vector $V_s$ of size $\binom{m}{t}$ with at most $s$ entries set to 1, at most $2s/p$ entries set to 0, and the remaining entries set to $p$, for a random example $X$ (drawn from $\{0,1\}^m$ according to the $1/2^k$-biased product distribution), we have that with probability at least $1 - \epsilon_1$, the quantity $V_s(X)$ lies in the range*

$$1 - (1-p)^{\left[\binom{m/2^k - (m/2^k)^{1/3}}{t} - \frac{2s\sqrt{n}}{p2^{kt}}\right]} \leq V_s(X) \leq 1 - (1-p)^{\binom{m/2^k + (m/2^k)^{1/3}}{t}}. \tag{5.11}$$

*Here*

$$\epsilon_1 = s \cdot \left(\frac{2\sqrt{n}}{p} + 1\right) 2^{-kt} + 2e^{-(m/2^k)^{1/3}/3}. \tag{5.12}$$

*Proof.* Let $X$ be a random example drawn from the $1/2^k$-biased product distribution over $\{0,1\}^m$, and consider the following 3 events:

- **None of the 1-entries in $V_s$ are satisfied by $X$.**

  There are at most $s$ 1-entries in $V_s$ and the probability that any one is satisfied by $X$ is $2^{-kt}$. Therefore the probability that some 1-entry is satisfied by $X$ is at most

$s2^{-kt}$ and the probability that none of the 1-entries in $V_s$ are satisfied by $X$ is at least $1 - s2^{-kt}$.

- **At most $(2s\sqrt{n}/p)2^{-kt}$ of the 0-entries in $V_s$ are satisfied by $X$.**

  Because there are at most $2s/p$ entries set to 0 in $V_s$, the expected number of 0-entries in $V_s$ satisfied by $X$ is at most $(2s/p)2^{-kt}$. By Markov's inequality, the probability that the actual number exceeds this by a $\sqrt{n}$ factor is at most $1/\sqrt{n}$.

- **The number of ones in $X$ lies in the range $m/2^k \pm (m/2^k)^{2/3}$.**

  Using a multiplicative Chernoff bound, we have that this occurs with probability at least $1 - 2e^{-(m/2^k)^{1/3}/3}$. Note that for any $X$ in this range, $f(X) = f_1(X)$. So, conditioning on this event occurring, we can assume that $f(X) = f_1(X)$.

Therefore, the probability that all 3 of the above events occurs is at least $1 - \epsilon_1$ where

$$\epsilon_1 = s \cdot \left( \frac{2\sqrt{n}}{p} + 1 \right) 2^{-kt} + 2e^{-(m/2^k)^{1/3}/3}.$$

Given that these events all occur, we show that $V_s(X)$ lies in the desired range. We follow the approach of [Blum *et al.*, 1998].

For the lower bound, $V_s(X)$ is minimized when $X$ has as few ones as possible and when as many of the 0-entries in $V_s$ are satisfied by $X$ as possible. So $V_s(X)$ is at least

$$V_s(X) \geq 1 - (1-p)^{\left[ \binom{m/2^k - (m/2^k)^{2/3}}{t} - \frac{2s\sqrt{n}}{p2^{kt}} \right]}.$$

For the upper bound, $V_s(X)$ is maximized when $X$ has as many ones as possible and as few zeros as possible. So, $V_s(X)$ is at most

$$V_s(X) \leq 1 - (1-p)^{\binom{m/2^k + (m/2^k)^{2/3}}{t}},$$

which completes the proof. ■

Now let us choose values for $k$ and $t$. What are our goals in setting these parameters? First off, we want $\binom{m/2^k}{t}$ to be at most poly$(n)$ (so that there are at most poly$(n)$ candidate terms to be checked for a "typical" input). Moreover, for any $s = \text{poly}(n)$ we want both sides of Equation 5.11 to be close to 1/2 (so the accuracy of any $s$-query learning algorithm

is indeed close to 1/2 on typical inputs), and we want $\epsilon_1$ to be small (so almost all inputs are "typical"). With this motivation, we set $k = \Theta(\log n)$ to be such that $m/2^k$ (recall, $m = n/k$) equals $\log^6 n$, and we set $t = \sqrt{\log n}$. This means

$$\binom{m/2^k}{t} = \binom{\log^6 n}{\sqrt{\log n}} \leq 2^{6\log(\log n)\sqrt{\log n}} \ll n\,.$$

Now Equation 5.10 gives $p \gg 1/n$; together with $k = \Theta(\log n)$, for any $s = \mathrm{poly}(n)$ we have $\epsilon_1 = 1/n^{\omega(1)}$.

Now we analyze Equation 5.11. First the lower bound:

$$\begin{aligned} V_s(X) &\geq 1 - (1-p)^{\left[\binom{m/2^k-(m/2^k)^{2/3}}{t} - \frac{2s\sqrt{n}}{p2^{kt}}\right]} \\ &\geq 1 - (1-p)^{\binom{m/2^k-(m/2^k)^{2/3}}{t}} \left(e^{\frac{3s\sqrt{n}}{p2^{kt}}}\right) \\ &= 1 - (1-p)^{\binom{m/2^k-(m/2^k)^{2/3}}{t}} \left(1 + 1/n^{\omega(1)}\right) \\ &= 1 - \left[2^{-\binom{m/2^k-(m/2^k)^{2/3}}{t}/\binom{m/2^k}{t}}\right] \cdot \left(1 + 1/n^{\omega(1)}\right)\,. \end{aligned}$$

(In the last step we are using the definition of $p$ from Equation 5.10.) Let us bound the exponent:

$$\begin{aligned} \frac{\binom{m/2^k-(m/2^k)^{2/3}}{t}}{\binom{m/2^k}{t}} &\geq \left(\frac{m/2^k - (m/2^k)^{2/3} - t}{m/2^k}\right)^t \\ &= \left(\frac{\log^6 n - \log^4 n - \sqrt{\log n}}{\log^6 n}\right)^{\sqrt{\log n}} \\ &\geq \left(\frac{\log^6 n - 2\log^4 n}{\log^6 n}\right)^{\sqrt{\log n}} \\ &= \left(1 - \frac{2}{\log^2 n}\right)^{\sqrt{\log n}} \\ &\geq 1 - \frac{2}{\log^{1.5} n}\,. \end{aligned}$$

So

$$V_s(X) \geq 1 - \left[2^{-(1-2/\log^{1.5} n)}\right] \cdot (1 + 1/n^{\omega(1)}) \geq \frac{1}{2} - \frac{1}{\log n}.$$

Now for the upper bound:

$$V_s(x) \leq 1 - (1-p)^{\binom{m/2^k+(m/2^k)^{2/3}}{t}} = 1 - 2^{-\binom{m/2^k+(m/2^k)^{2/3}}{t}/\binom{m/2^k}{t}}\,.$$

Again bounding the exponent:

$$\begin{aligned}
\frac{\binom{m/2^k+(m/2^k)^{2/3}}{t}}{\binom{m/2^k}{t}} &= \frac{\binom{\log^6 n+\log^4 n}{\sqrt{\log n}}}{\binom{\log^6 n}{\sqrt{\log n}}} \\
&\leq \left(\frac{\log^6 n+\log^4 n}{\log^6 n-\sqrt{\log n}}\right)^{\sqrt{\log n}} \\
&\leq \left(1+\frac{2\log^4 n}{\log^6 n-\sqrt{\log n}}\right)^{\sqrt{\log n}} \\
&\leq 1+\frac{4}{\log^{1.5} n}\,.
\end{aligned}$$

So

$$V_s(X) \leq 1-2^{-\left(1+\frac{4}{\log^{1.5} n}\right)} \leq \frac{1}{2}+\frac{1}{\log n}\,.$$

The above analysis has thus established the following.

**Lemma 29.** *Let $L$ be any poly$(n)$-time learning algorithm. If $L$ is run with a target function that is a random draw $f$ from the distribution $P$ described above, then for all but a $1/n^{\omega(1)}$ fraction of inputs $x \in \{0,1\}^n$, the probability that $h(x) = f(x)$ (where $h$ is the hypothesis output by $L$) is at most $1/2 + 1/\log n$.*

It is easy to see that by slightly modifying the values of $t$ and $k$ in the above construction, it is actually possible to replace $1/\log n$ with any $1/\text{polylog}\, n$ in the above lemma.

### 5.4.4 Computational lower bound

To obtain a computational analogue of Lemma 29, we make a pseudorandom choice of terms in a draw of $f_1$ from $P$.

Recall that the construction of $P$ placed each possible term (conjunction of $t$ super-variables) in the target function with probability $p$, as defined in Equation 5.10. We first consider a distribution that uses uniform bits to approximate the probability $p$. This can be done by approximating $\log(p^{-1})$ with $\text{poly}(n)$ bits, associating each term with independent uniform $\text{poly}(n)$ bits chosen this way, and including that term in the target function if all bits are set to 0. It is easy to see that the resulting construction yields a probability distribution that is statistically close to $P$, and we denote it by $P^{\text{stat}}$.

Now, using a pseudorandom function rather than a truly random (uniform) one for the source of uniform bits will yield a distribution, which we denote by $P^{\mathrm{PSR}}$. Similar arguments to those we give elsewhere in the paper show that a poly($n$) time adversary cannot distinguish the resulting construction from the original one (or else a distinguisher could be constructed for the pseudorandom function).

To complete the argument, we observe that every function $f$ in the support of $P^{\mathrm{PSR}}$ can be evaluated with a poly($n$)-size circuit. It is obviously easy to count the number of supervariables that are satisfied in an input $x$, so we need only argue that the function $f_1$ can be computed efficiently on a "typical" input $x$ that has "few" supervariables satisfied. But by construction, such an input will satisfy only poly($n$) candidate terms of the monotone DNF $f_1$ and thus a poly($n$)-size circuit can check each of these candidate terms separately (by making a call to the pseudorandom function for each candidate term to determine whether it is present or absent). Thus, as a corollary of Lemma 29, we can establish the main result of this section:

**Theorem 19.** *Suppose that standard one-way functions exist. Then there is a class $\mathcal{C}$ of poly($n$)-size monotone circuits that is hard to learn to accuracy* $1/2 + 1/\mathrm{polylog}(n)$.

# Chapter 6

# Learning an Overcomplete Basis: Analaysis of Lattice-based Signature Schemes

In this chapter we see that it is useful to study the security of cryptosystems as average-case learning problems. Here we propose an attack on a family of lattice-based signature schemes by showing that recovery of secret keys from signed messages reduces to a learning problem. We then show that "random" instances of this learning problem have a special structure that makes learning feasible.

## 6.1 Introduction

### 6.1.1 Background and Motivation

In 1997, Goldreich, Goldwasser and Halevi (GGH) [Goldreich *et al.*, 1997] proposed a lattice-based signature scheme and public-key encryption scheme that were inspired by the breakthrough work of hardness of well-studied lattice problems. Since then, numerous variations of the GGH schemes have been proposed including the commercial offering NTRUSign [Hoffstein *et al.*, 2003], which applies the ideas of the GGH signature scheme to the compact NTRU family of lattices. The NTRUSign scheme is reasonably efficient, but it has no known

security proof.

In a beautiful paper from 2006, Nguyen and Regev [Nguyen and Regev, 2009] described a cryptanalytic key-recovery attack against GGH-style signature schemes, including the basic version of NTRUSign. Their work describes a polynomial-time algorithm (which is also quite efficient in practice) for recovering the secret key from a transcript of signatures on (arbitrary) messages. The first main observation behind their attack is that the signature transcript can serve as the data set in a "hidden parallelepiped" learning problem. In this problem, the learner is given a collection of (unlabeled) example vectors in $\mathbb{R}^n$, each of which is drawn uniformly at random from an unknown $n$-dimensional parallelepiped in $\mathbb{R}^n$ (centered at the origin). The job of the learner is to reconstruct the basis vectors of the unknown parallelepiped (or a close approximation thereof).

Even before the full attack of [Nguyen and Regev, 2009] was developed, it was known that the basic approach behind GGH and NTRU signatures leaked some information about the secret key that was potentially useful in an attack [Gentry and Szydlo, 2002; Szydlo, 2003]. As a result, the design of NTRUSign also includes a method for applying one or more "*perturbations*" in the signing algorithm as a countermeasure against cryptanalysis; this technique is part of the proposed IEEE P1363.1 standard that is currently under consideration [IEEE, 2008]. When perturbations are used, signatures are drawn from a non-uniform probability distribution over a convex set in $\mathbb{R}^n$ that is more complicated than a parallelepiped (we elaborate more on this below). The drawback is that each perturbation increases the running time of the signing algorithm, and necessitates a larger dimension $n$ (and correspondingly larger key sizes) to avoid certain other attacks.

While the cryptanalytic attack of Nguyen and Regev [Nguyen and Regev, 2009] completely breaks the *basic* NTRUSign scheme, it does not apply when one or more perturbations are used. Cryptanalysis of the perturbation technique was left as the main open problem, and to our knowledge no such analysis has yet appeared. Given this state of affairs, a very natural and important question is whether perturbations actually are an effective security measure. Further motivation is provided by the recent work of Gentry, Peikert, and Vaikuntanathan [Gentry *et al.*, 2008] (followed by [Peikert, 2009]), who gave theoretically justified variants of GGH-style signature schemes that are *provably unforgeable*

under worst-case lattice assumptions. Given the existence of provably secure alternatives, the question of whether already-implemented schemes like NTRUSign with perturbations are comparably secure takes on added interest.

### 6.1.2 Our Contributions

In this chapter we address the main problem left open by the work of Nguyen and Regev [Nguyen and Regev, 2009] — namely, cryptanalysis of GGH/NTRU-style signature schemes *with perturbations*.

Following [Nguyen and Regev, 2009], we first present an abstract model of the signature distribution, and phrase the key-recovery task as a learning problem. We generalize the "hidden parallelepiped" problem (see above in Section 6.1.1) as formulated by [Nguyen and Regev, 2009]; in the new problem, each sample is obtained as a random convex combination of an *overcomplete* basis of $m > n$ vectors in $\mathbb{R}^n$, and our goal is to find or approximate as many of the $m$ vectors as possible (see Section 6.3 for a formal definition of the problem).

Based on other security considerations for lattice-based signature schemes and some standard facts about high-dimensional geometry, we then argue that it is reasonable to view the $m$ unknown vectors as *quasi-orthogonal* elements of $\mathbb{R}^n$. For quasi-orthogonal vectors, we show (in Section 6.4.2) that it suffices to identify local maxima of the $\ell_\infty$-norm of a certain function over $\mathbb{R}^n$. While we are not able to find the local maxima of the $\ell_\infty$-norm efficiently, we observe (in Section 6.4.3) that for a sufficiently large constant $k$, the $\ell_k$-norm is a very good proxy for the $\ell_\infty$-norm. This motivates an approach in which we search for local maxima of the *kth cumulant* of a certain random variable, for an appropriate choice of $k$.

In Section 6.4.4, we give an optimization algorithm that is specifically tailored to finding these local maxima using the $k$th cumulant of the random variable. Then, in Section 6.4.5, we show how to compute estimates of the $k$th cumulant from signature samples. To demonstrate the efficacy of our attack, we need to argue that the algorithm is efficient and correct when given the necessary statsitics and then show that we can efficiently compute reliable estimates of the statistics.

We provide very strong empirical results indicating that the optimization algorithm,

using real or noisy values for the cumulant, converges quickly to the hidden columns of the basis. In Section 6.5.1 we present experimental results showing that our algorithm performs extremely well on real-world parameters (and far beyond) with noisy or exact values of the $k$th cumulant for $k = 6$. In contrast, for $k = 4$ the algorithm performs poorly even when given exact values of the cumulants, which indicates that our approach of using the higher-order cumulants is an essential novelty of this work. Though we do not yet have a full theoretical proof of its correctness or convergence, we also present a variety of theoretical results that indicate various ways in which our algorithm is effective and well-motivated, see Section 6.4. For example, we prove that the $\ell_k$ norm typically has local maxima very close to those of the $\ell_\infty$ norm, which give the vectors we are trying to find. We have not found a proof that no other local maxima exist, but our experiments (Section 6.5) have yet to encounter one.

We provide a full proof in Section 6.4.5 that the necessary (estimated) statistic, the $k$th cumulant of the random variable, can be estimated efficiently from polynomially many random samples. We further conduct experiments, presented in Section 6.5.2, which indicate that we can use significantly fewer signatures (roughly $n^7$) than the (polynomial) number required by our theoretical bound (Lemma 37).

We stress that our optimization algorithm is quite generally applicable; it makes no use of the special structure of NTRU lattices, or even of the public signature verification key. (However, by an observation attributed to Whyte in [Nguyen and Regev, 2009], a single NTRU signature in dimension $n$ may heuristically be expanded into $n$ samples for the learning algorithm, which decreases the sample complexity of the algorithm somewhat.) The only barrier preventing a *practical* attack is the number of signatures needed to estimate the $k$th cumulant reliably, even for $k = 6$. Our experiments indicate that the number of signatures required to mount an effective attack using our approach, while significantly smaller than the number required by our theoretical bound, is still not yet practical. We note that the designers of NTRUSign recommend releasing a limited numbers of signatures for each key [Hoffstein *et al.*, 2003; Hoffstein *et al.*, 2005], yet they also claim that the number of signatures allowed can be exponential in the number of perturbations. In contrast, our theoretical analysis shows that only polyomially many signatures are needed to compute

the statistics needed by our algorithm. The effectiveness of our attack underscores the importance of using a limited number of signatures.

### 6.1.3 Overview of Our Attack

As in [Nguyen and Regev, 2009], we model the cryptanalytic task as an abstract learning problem (see Section 6.3 for full details and justification of the model). For positive integers $m \geq n$ (e.g., $m = 2n$), let $\mathbf{C} \in \mathbb{R}^{n \times m}$ be an unknown rank-$n$ matrix, which corresponds to the secret key of the signature scheme for dimension $n$. (Note that the case $m = n$ corresponds to the basic GGH-style signature scheme *without* perturbations, for which prior attacks apply; the case $m > n$ corresponds to signatures with perturbations.) The signing algorithm implicitly produces independent random samples of the form

$$\mathbf{x} = \mathbf{Cs} \in \mathbb{R}^n$$

for uniformly random $\mathbf{s} \in [-1, 1)^n$. The goal is to recover many (or all) of the columns $\mathbf{c}_i$ of the matrix $\mathbf{C}$, or at least close approximations thereof, using oracle access to the signer. (An approximation suffices because the secret key consists of *integer* lattice points, so the approximate solution may be rounded to the true value.)

As in prior works [Hyvärinen *et al.*, 2001; Gentry and Szydlo, 2002; Frieze *et al.*, 1996; Nguyen and Regev, 2009], the first basic step of our attack is to "sphere" the distribution of $\mathbf{x} = \mathbf{Cs}$ by estimating and correcting its covariance; we can then assume that the rows of $\mathbf{C}$ are *orthonormal*. Moreover, because the secret key is random, we heuristically model the rows of $\mathbf{C}$ as spanning a random $n$-dimensional subspace of $\mathbb{R}^m$. The exact distribution of the subspace is not too important; the only fact we need is that with noticeable probability, the $m$ columns $\mathbf{c}_i \in \mathbb{R}^n$ of $\mathbf{C}$ are *quasi-orthogonal*, i.e., $\langle \mathbf{c}_i, \mathbf{c}_j \rangle \approx 0$ when $i \neq j$ and $\approx n/m$ when $i = j$. This condition can be guaranteed to an appropriate degree of tolerance as a consequence of, e.g., the Johnson-Lindenstrauss lemma.

We then search for the columns $\mathbf{c}_i$ of $\mathbf{C}$ by measuring the statistics of the random variable $\mathbf{x} = \mathbf{Cs}$. The main task is to determine the *direction* of some $\mathbf{c}_i$. (Its length will be apparent once the direction is known, as explained below). By considering the inner product of $\mathbf{x}$ with some unit vector $\mathbf{w} \in \mathbb{R}^n$ of our choice, we get a random variable of the

form

$$\langle \mathbf{w}, \mathbf{x} \rangle = \langle \mathbf{w}, \mathbf{C}\mathbf{s} \rangle = \langle \mathbf{C}^t\mathbf{w}, \mathbf{s} \rangle.$$

(Note that $\mathbf{C}^t\mathbf{w} \in \mathbb{R}^m$ is always a unit vector, because $\mathbf{w}$ is unit and the columns of $\mathbf{C}^t$ are orthonormal.) Our first key observation, which follows from quasi-orthogonality, is that the $\ell_\infty$ norm $\|\mathbf{C}^t\mathbf{w}\|_\infty$ is *locally maximized* exactly when $\mathbf{w}$ is aligned with any column of $\mathbf{C}$.[1] Ideally, then, the statistics of $\langle \mathbf{C}^t\mathbf{w}, \mathbf{s} \rangle$ would let us estimate the value of $\|\mathbf{C}^t\mathbf{w}\|_\infty$ and find its local maxima efficiently, which would reveal $\mathbf{C}$ exactly. However, we do not know if this is possible.

Instead, we use the $\ell_k$ *norm* as a proxy for the the $\ell_\infty$ norm, for some large enough even integer $k$ (e.g., $k = 6$).[2] One can see that $\|\mathbf{C}^t\mathbf{w}\|_k$ closely approximates $\|\mathbf{C}^t\mathbf{w}\|_\infty$, especially in the regions of our interest, where $\mathbf{C}^t\mathbf{w}$ has one dominant entry. We observe that the value of $\|\mathbf{C}^t\mathbf{w}\|_k$ corresponds exactly to a statistical quantity called the *kth cumulant* of the random variable $\langle \mathbf{C}^t\mathbf{w}, \mathbf{s} \rangle$. (Cumulants are similar to *moments*, but with the extra crucial property that they are *additive* for independent random variables.) Furthermore, it is possible to show that the $k$th cumulant of $\langle \mathbf{C}^t\mathbf{w}, \mathbf{s} \rangle$, along with its *gradient* as a function of $\mathbf{w}$, may be efficiently estimated using about $m^{O(k)}$ samples, which is polynomial for constant $k$.

Given this ability to estimate the $\ell_k$ norm and its gradient, standard optimization algorithms such as gradient ascent or Newton's method can then be used to find local maxima. However, we find that the particular geometric form of our objective function admits a *special-purpose* iterative optimization algorithm in which the current direction $\mathbf{w}$ is simply *replaced by* the gradient, which seems to perform more robustly in our experiments.

Finally, once the direction $\mathbf{w} = \mathbf{c}_i/\|\mathbf{c}_i\|$ of some $\mathbf{c}_i$ is discovered, we see that the length $\|\mathbf{c}_i\|$ is

$$\|\mathbf{c}_i\| = \frac{\langle \mathbf{c}_i, \mathbf{c}_i \rangle}{\|\mathbf{c}_i\|} = \frac{\|\mathbf{C}^t\mathbf{c}_i\|_\infty}{\|\mathbf{c}_i\|} = \|\mathbf{C}^t\mathbf{w}\|_\infty.$$

This quantity is approximately $\|\mathbf{C}^t\mathbf{w}\|_k$, which can be measured efficiently, as discussed above.

---

[1] The $\ell_\infty$ norm is defined as $\|\mathbf{z}\|_\infty = \max_i |z_i|$.

[2] The $\ell_k$ norm is defined as $\|\mathbf{z}\|_k = (\sum_i |z_i|^k)^{1/k}$.

### 6.1.4 Relation to Prior Work

As briefly noted above, the learning problem we consider is a generalization of the abstract learning problem considered in [Frieze *et al.*, 1996; Nguyen and Regev, 2009]. The problem studied in those works corresponds to the special case $m = n$ in our problem, in which the $m = n$ unknown vectors in $\mathbb{R}^n$ are the basis vectors defining a paralellepiped in $\mathbb{R}^n$. Heuristics for attacking this kind of learning problem had previously been given in the Independent Component Analysis (ICA) literature (see, e.g., [Hyvärinen *et al.*, 2001]); a polynomial-time algorithm was described by Frieze *et al* [Frieze *et al.*, 1996], and a similar algorithm was analyzed rigorously by Nguyen and Regev [Nguyen and Regev, 2009].

The more general problem that we address, where $m$ may be substantially larger than $n$, seems considerably more difficult. It has received far less attention in the ICA literature, and we do not know of any work on theoretically efficient algorithms with provable performance guarantees. In a setting of quasi-orthogonal basis vectors similar to ours, Hyvärinen and Inki [Hyvärinen and Inki, 2002] propose an algorithm based on a Bayesian maximum likelihood approach. In contrast, our approach is based on estimating the cumulants of the data and a special-purpose optimization algorithm for finding local maxima of the $\ell_k$ norm, as described above.

We point out that our cryptanalytic attack, as well as the one of [Nguyen and Regev, 2009], fails when applied to the (provably secure) signature schemes of [Gentry *et al.*, 2008; Peikert, 2009], because the signatures are drawn from (essentially) an $n$-dimensional Gaussian that is independent of the secret key basis, i.e., the signatures are "zero-knowledge."

## 6.2 Background

For the purposes of this work, a lattice is a full-rank discrete subgroup of $\mathbb{R}^n$. Any lattice $\mathcal{L}$ can be represented by a *basis*, that is, a set of linearly independent $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathcal{L}$ such that $\mathcal{L} = \{\sum_{i=1}^n c_i \mathbf{b}_i | c_i \in \mathbb{Z}\}$. Conversely, any set of linearly independent vectors is a basis for some lattice. We denote the gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ by $\nabla f = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$, and the unit sphere in $\mathbb{R}^n$ by $S_n = \{\mathbf{w} \in \mathbb{R}^n \ : \ \|\mathbf{w}\| = 1\}$.

### 6.2.1 GGH-Style Signatures and Perturbations

The signature scheme originally proposed in [Goldreich *et al.*, 1997], referred to as the GGH signature scheme, works with lattices in $\mathbb{Z}^n$. The private key of the signer is a "short" basis for a lattice $\mathcal{L}$ (where each basis vector has small Euclidean norm), while the public key is a much longer basis for the same lattice. To create a signature, the message is first mapped to $\mathbb{Z}^n$ using a hash function, which can be modeled heuristically as a "random oracle." Using the private key and Babai's round off algorithm [Babai, 1986], the signer finds a vector in $\mathcal{L}$ near the hashed message, outputting it as the signature. Using the public basis, the verifier may check that the vector belongs to the lattice and is close enough to the hashed message. As mentioned previously, different incarnations of the original scheme exist, including the NTRUSign scheme [Hoffstein *et al.*, 2003], which uses a compact family of lattices from the NTRU encryption scheme [Hoffstein *et al.*, 1998].

The signatures of GGH-style schemes are known to leak information about the secret key, as first shown in [Gentry and Szydlo, 2002]. The perturbation technique was proposed as part of NTRUSign in the hopes of increasing its security [Hoffstein *et al.*, 2003; Hoffstein *et al.*, 2005]. Abstractly, the technique involves generating and storing some number of additional short vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots \in \mathbb{Z}^n$ as part of the secret key, and adding a random linear combination of them to the hashed message before applying the usual signing algorithm to the perturbed hash value. The signature still verifies because it is still relatively close to the original hashed message (though it is somewhat farther away than if the perturbation had been left out). In the particular case of NTRUSign, the extra secret vectors are simply the secret short bases from one or more independently generated keypairs; the corresponding long bases are discarded and never made public.

### 6.2.2 Statistical Quantities

Let $X$ be a random variable over the reals $\mathbb{R}$. For nonnegative integer $k$, the $k$th *moment* $\mu_k$ (about the origin) of $X$ is defined as

$$\mu_k(X) = \mathbf{E}[X^k].$$

The *cumulants* $\kappa_k$ of $X$ are closely related to the moments. Formally, they are defined as the coefficients of the cumulant-generating function

$$g(t) = \ln \mathbf{E}[e^{t \cdot X}] = \sum_{k=1}^{\infty} \kappa_k(X) \cdot \frac{t^k}{k!}.$$

The first cumulant $\kappa_1$ equals the mean $\mu_1$, and the second and third cumulants $\kappa_2, \kappa_3$ are respectively the second and third *central* moments about the mean $\mu_1$. The general relationship between cumulants and moments is more complex, and is given by the following recursive formula:

$$\kappa_k = \mu_k - \sum_{j=1}^{k-1} \binom{k-1}{j-1} \cdot \kappa_j \cdot \mu_{k-j}.$$

Like the $k$th moment, the $k$th cumulant is homogeneous of degree $k$, that is, $\kappa_k(c \cdot X) = c^k \cdot \kappa_k(X)$ for any constant $c \in \mathbb{R}$. Unlike moments, however — and crucially for our purposes — cumulants are also *additive*. That is, for independent random variables $X, Y$,

$$\kappa_k(X + Y) = \kappa_k(X) + \kappa_k(Y).$$

## 6.3 The Learning Problem

We model the key recovery attack on lattice-based signatures with perturbations as follows.

**Definition 5** (**Learning an Overcomplete Basis Problem**). *For $m \geq n$, let $\mathbf{A} \in \mathbb{R}^{n \times m}$ be a matrix of rank $n$. The input consists of polynomially many samples $\mathbf{As} \in \mathbb{R}^n$ for independent, uniformly distributed $\mathbf{s} \in [-1, 1]^m$. The goal is to find a good approximation to all or most of the columns $\mathbf{a}_i$ of $\mathbf{A}$.*

The special case of $m = (k+1)n$ correspond to NTRU-style signature with $k$ perturbations, where a transcript of signatures provides samples that are sums of random points from $k+1$ different parallelepipeds. The case $k = 0$ (no perturbations) corresponds to the problem addressed by [Nguyen and Regev, 2009]. We note that in the case of NTRU signatures, it suffices to find just one column of $\mathbf{A}$ belonging to the "true" secret key, then the remainder of the key may be found using the symmetries of NTRU lattices.

The learning problem may be considered in a "worst-case" form, where $\mathbf{A}$ is arbitrary, or as an "average-case" problem, where $\mathbf{A}$ is chosen from some distribution. For a cryptanalytic

attack, it suffices to solve the average-case problem where the distribution on $\mathbf{A}$ is induced by the key generation procedure, with some significant probability over all the randomness of the experiment. We note that while [Nguyen and Regev, 2009] address the worst-case version of their learning problem, our approach exploits the random distribution of $\mathbf{A}$ in an important way.

The distribution on $\mathbf{A}$ that we focus on is the idealized one in which each entry is an independent, normally distributed real (with the same variance). Under this distribution, each row in $\mathbf{A}$ is an independent normally distributed vector in $\mathbb{R}^m$, and thus (with probability 1) the rows span a random $n$-dimensional subspace of $\mathbb{R}^m$. We argue that this is the most natural way to model the key-generation algorithm in GGH-based signatures with perturbations. We note that the specific key generation algorithm in NTRUSign applies a more biased process, resulting in some longer and some shorter vectors in $\mathbf{A}$. The method appears to be motivated by efficiency concerns, and it seems that in terms of security, a matrix spanning a random subspace would be preferable. In particular, in our attack the longer vectors of the secret key will be found more easily (at the expense of the shorter vectors).

As explained in Section 6.1.2, we need to assume a certain distribution on $\mathbf{A}$ because our learning algorithm relies on the *quasi-orthogonality* of the columns of $\mathbf{A}$. (This is in contrast with the simpler case of $m = n$, where the columns of $\mathbf{A}$ are *guaranteed* to be orthogonal following a preprocessing step). Our learning algorithm is actually quite robust and works for a variety of distributions, as long as they provide a reasonable probability of quasi-orthogonality. We prove in Theorem 20 that this condition is likely to occur when the rows of $\mathbf{A}$ span a random $n$-dimensional subspace of $\mathbb{R}^m$.

## 6.4 Approach and Learning Algorithm

Here we expand in detail upon the approach outlined in Section 6.1.3. We first show in Section 6.4.1 how to transform instances of our learning problem into instances where the rows of the unknown matrix $\mathbf{C}$ are orthonormal. In Section 6.4.2 we establish a key property of $\mathbf{C}$, namely quasi-orthogonality, and demonstrate an objective function over

the unit sphere $S_n$ that is locally maximized exactly in the direction of each column of $\mathbf{C}$. While we do not know how to maximize this exact objective function efficiently, we show in Section 6.4.3 that there is a suitable proxy that can be optimized using standard algorithms, given access to certain statistics. In Section 6.4.4 we additionally describe a particular optimization algorithm that is tailored specifically to our objective function. In Section 6.4.5 we give theoretical bounds on the number of samples needed to obtain reliable estimates of the statistics required by the optimization algorithms.

### 6.4.1 Sphering the Distribution

As in prior theoretical work [Frieze *et al.*, 1996; Nguyen and Regev, 2009] and the ICA literature [Hyvärinen *et al.*, 2001], we first reduce our learning problem to one in which the data are "whitened" or "sphered," i.e., have a trivial covariance matrix $\mathbf{I}_n$. The following two lemmas are slight generalizations of those appearing in [Nguyen and Regev, 2009].

**Lemma 30.** *For any $\mathbf{A} \in \mathbb{R}^{n \times m}$ and uniformly random $\mathbf{s} \in [-1, 1)^m$, we have*

$$\mathbf{E}_{\mathbf{x}=\mathbf{As}}[\mathbf{x}\mathbf{x}^t] = \mathbf{A} \cdot \mathbf{E}_{\mathbf{s}}[\mathbf{s}\mathbf{s}^t] \cdot \mathbf{A}^t = \frac{1}{3}\mathbf{A}\mathbf{A}^t.$$

**Lemma 31.** *Let $\mathbf{A} \in \mathbb{R}^{n \times m}$ have full row rank, and let $\mathbf{G} \in \mathbb{R}^{n \times n}$ be the symmetric positive definite Gram matrix $\mathbf{G} = \mathbf{A}\mathbf{A}^t$. Let $\mathbf{L} \in \mathbb{R}^{n \times n}$ be the Cholesky factor of $\mathbf{G}^{-1}$, that is, is the unique lower-triangular matrix such that $\mathbf{G}^{-1} = \mathbf{L}\mathbf{L}^t$. Then the rows of the matrix $\mathbf{C} = \mathbf{L}^t\mathbf{A}$ are orthonormal, i.e., $\mathbf{C}\mathbf{C}^t = \mathbf{I}_n$.*

*Proof.* The matrix $\mathbf{G}^{-1}$ is symmetric positive definite, because the Gram matrix $\mathbf{G} = \mathbf{A}\mathbf{A}^t$ is. Thus $\mathbf{G}^{-1}$ has a Cholesky factorization $\mathbf{G}^{-1} = \mathbf{L}\mathbf{L}^t$, where $\mathbf{L}$ is lower triangular and nonsingular. Then $\mathbf{A}\mathbf{A}^t = (\mathbf{L}\mathbf{L}^t)^{-1}$, so

$$\mathbf{C}\mathbf{C}^t = \mathbf{L}^t\mathbf{A}\mathbf{A}^t\mathbf{L} = \mathbf{L}^t(\mathbf{L}\mathbf{L}^t)^{-1}\mathbf{L} = \mathbf{I}_n. \blacksquare$$

By Lemma 30 and standard Chernoff bounds, we can approximate the Gram matrix $\mathbf{G} = \mathbf{A}\mathbf{A}^t$ to a high degree of accuracy by averaging $\mathbf{x}\mathbf{x}^t$ over many samples $\mathbf{x} = \mathbf{As}$. We can then compute the Cholesky factorization of $\mathbf{G}^{-1} = \mathbf{L}\mathbf{L}^t$ efficiently using, e.g., Gram-Schmidt orthogonalization. By Lemma 31, we may apply the transformation $\mathbf{L}^t$ to our data

samples to obtain "sphered" samples of the form $\mathbf{Cs} = \mathbf{L}^t\mathbf{As}$, where $\mathbf{CC}^t$ is very close to the identity $\mathbf{I}_n$. In other words, we can reduce the learning problem to one in which the rows of the unknown matrix $\mathbf{C}$ are essentially orthonormal. Moreover, note that the row spaces of $\mathbf{A}$ and $\mathbf{C} = \mathbf{L}^t\mathbf{A}$ are the same, so the probability distribution over the row space from the learning problem remains unchanged. Finally, given a solution that approximates $\mathbf{C}$ (or any of its columns), we can recover an approximation of $\mathbf{A}$ (or the corresponding columns) simply by applying $\mathbf{L}^{-t}$.

For the remainder of this section, we therefore assume that the unknown matrix in our learning problem is some $\mathbf{C} \in \mathbb{R}^{n\times m}$ such that $\mathbf{CC}^t = \mathbf{I}_n$.

### 6.4.2 Quasi-Orthogonality and the $\ell_\infty$ Norm

When $\mathbf{C}$ is *square* (i.e., $m = n$) and its rows are orthonormal, then its columns are orthonormal as well. The algorithms from [Frieze *et al.*, 1996; Nguyen and Regev, 2009] crucially exploit this fact, defining a multivariate objective function (related to the fourth moment of projected samples) that is optimized precisely at each of the columns of $\mathbf{C}$. When $m > n$, however, the columns of $\mathbf{C}$ cannot possibly be mutually orthogonal, and the objective function used in [Frieze *et al.*, 1996; Nguyen and Regev, 2009] does not have the required properties.

Still, it is possible for $m \gg n$ directions in $\mathbb{R}^n$, represented by unit vectors $\mathbf{u}_1, \ldots, \mathbf{u}_m \in \mathbb{R}^n$, to satisfy the more relaxed notion of *quasi-orthogonality*, in which all the inner products $\langle \mathbf{u}_i, \mathbf{u}_j \rangle$ for $i \neq j$ are relatively small (near zero). In fact, for $m = \text{poly}(n)$, "most" choices of $m$ directions in $\mathbb{R}^n$ are quasi-orthogonal. Therefore, the notion is reasonably robust to different distributions of the chosen directions.

Here we make this intuition more formal, and demonstrate a particular objective function that is optimized precisely in the direction of the unknown columns of $\mathbf{C}$ (when they are indeed quasi-orthogonal).

**Lemma 32** (Johnson-Lindenstrauss)**.** *Let $\epsilon \in (0, 1)$, let $S$ be any finite set of points in $\mathbb{R}^m$, and let $n \geq n_0 = O(\epsilon^{-2} \ln |S|)$. Let $f$ be the orthogonal projection from $\mathbb{R}^m$ onto a random $n$-dimensional subspace of $\mathbb{R}^m$ (i.e., the subspace spanned by $n$ independent normally distributed vectors in $\mathbb{R}^m$). Then except with probability $O(1/|S|)$ over the choice of the*

*subspace, for every* $\mathbf{v} \in S$ *we have*

$$\|f(\mathbf{v})\|^2 \in (1 \pm \epsilon) \cdot \frac{n}{m} \cdot \|\mathbf{v}\|^2.$$

We remark that the probability $O(1/|S|)$ in the statement of Lemma 32 is relatively arbitrary, and may be reduced to $1/|S|^c$ for any constant $c > 0$ at the expense of a larger constant factor in the threshold $n_0 = O(\epsilon^{-2} \ln |S|)$.

**Definition 6.** *For* $\gamma \geq 1$*, we say that the columns* $\mathbf{x}_i$ *of a matrix* $\mathbf{X}$ *are* $\gamma$-quasi-orthogonal *if each diagonal entry of the symmetric Gram matrix* $\mathbf{X}^t\mathbf{X}$ *has magnitude at least a* $\gamma$ *factor larger than that of every other entry in its row/column, i.e., if* $|\langle \mathbf{x}_i, \mathbf{x}_i \rangle| \geq \gamma \cdot |\langle \mathbf{x}_i, \mathbf{x}_j \rangle|$ *for every distinct* $i, j$*.*

In order to apply the Johnson-Lindenstrauss lemma in our context, we first observe that the Gram matrix $\mathbf{G} = \mathbf{C}^t\mathbf{C} \in \mathbb{R}^{m \times m}$ is the unique matrix representing the orthogonal projection from $\mathbb{R}^m$ onto the linear subspace $V = \text{span}(\mathbf{C}^t)$ spanned by the columns of $\mathbf{C}^t$. To see this, note that $\mathbf{C}^t\mathbf{C}\mathbf{x} = \mathbf{C}^t\mathbf{0} = \mathbf{0}$ for any $\mathbf{x}$ in the orthogonal complement of $V$, and that any $\mathbf{x} \in V$ may be written as $\mathbf{x} = \mathbf{C}^t\mathbf{z}$ for some $\mathbf{z} \in \mathbb{R}^n$, so $\mathbf{C}^t\mathbf{C}\mathbf{x} = \mathbf{C}^t(\mathbf{C}\mathbf{C}^t)\mathbf{z} = \mathbf{C}^t\mathbf{I}_n\mathbf{z} = \mathbf{x}$.

**Theorem 20.** *Let* $1/2 \geq \epsilon \geq \epsilon_0 = O(\sqrt{\ln(m)/n})$*. Then except with probability at most* $O(1/m^2)$ *over the random* $n$*-dimensional subspace of* $\mathbb{R}^m$ *spanned by* $\mathbf{C}^t$*, the following holds for all* $i \neq j$*:*

$$\langle \mathbf{c}_i, \mathbf{c}_i \rangle \in (1 \pm \epsilon) \cdot \frac{n}{m} \quad \textit{and} \quad \langle \mathbf{c}_i, \mathbf{c}_j \rangle \in \pm\epsilon \cdot \frac{n}{m}.$$

*In particular, the columns of* $\mathbf{C}$ *are* $\Omega(\sqrt{n/\ln m})$*-quasi-orthogonal.*

*Proof.* For $\mathbf{v} \in \mathbb{R}^m$, define $f(\mathbf{v}) = \mathbf{C}^t\mathbf{C}\mathbf{v}$ as the orthogonal projection from $\mathbb{R}^m$ onto $\text{span}(\mathbf{C}^t)$, which is a random $n$-dimensional subspace of $\mathbb{R}^m$. Also note that $\|f(\mathbf{v})\| = \|\mathbf{C}\mathbf{v}\|$, because the columns of $\mathbf{C}^t$ are orthonormal.

Define the set $S = \{\mathbf{e}_1, \ldots, \mathbf{e}_m\} \cup \{\mathbf{e}_i - \mathbf{e}_j : i > j\}$, where the $\mathbf{e}_i \in \mathbb{R}^m$ are the standard basis vectors. Note that $|S| = \Theta(m^2)$. Then Lemma 32 says that except with probability $O(1/m^2)$, we have

$$\langle \mathbf{c}_i, \mathbf{c}_i \rangle = \|\mathbf{C}\mathbf{e}_i\|^2 = \|f(\mathbf{e}_i)\|^2 \in (1 \pm \epsilon) \cdot \frac{n}{m}.$$

Moreover, for $i \neq j$ we have (again by Lemma 32)

$$2\langle \mathbf{c}_i, \mathbf{c}_j \rangle = \langle \mathbf{c}_i, \mathbf{c}_i \rangle + \langle \mathbf{c}_j, \mathbf{c}_j \rangle - \langle \mathbf{c}_i - \mathbf{c}_j, \mathbf{c}_i - \mathbf{c}_j \rangle = \|f(\mathbf{e}_i)\|^2 + \|f(\mathbf{e}_j)\|^2 - \|f(\mathbf{e}_i - \mathbf{e}_j)\|^2 \in \pm 2\epsilon \cdot \frac{n}{m}. \blacksquare$$

We now define an objective function $f_\infty : S_n \to \mathbb{R}$ over the unit sphere as

$$f_\infty(\mathbf{w}) = \|\mathbf{C}^t \mathbf{w}\|_\infty.$$

**Theorem 21.** *If the columns of* $\mathbf{C}$ *are* $\gamma$*-quasi-orthogonal for some* $\gamma > 1$*, then the local maxima of* $f_\infty$ *are exactly those unit vectors parallel to the columns* $\mathbf{c}_i$ *of* $\mathbf{C}$*.*

*Proof.* That each unit vector $\pm\mathbf{u}_i = \pm\mathbf{c}_i / \|\mathbf{c}_i\|$ is a local maximum of $f_\infty$ follows by hypothesis: the magnitude of the $i$th coordinate of $\mathbf{C}^t\mathbf{u}_i$ is $|\langle \mathbf{c}_i, \mathbf{c}_i \rangle| / \|\mathbf{c}_i\| = \|\mathbf{c}_i\|$, while the magnitude of every other coordinate $|\langle \mathbf{c}_j, \mathbf{u}_i \rangle| < \|\mathbf{c}_i\|$ for $j \neq i$. Thus $f_\infty(\pm\mathbf{u}_i) = \|\mathbf{c}_i\|$. Now for any unit vector $\mathbf{w} \neq \pm\mathbf{u}_i$ in a sufficiently small neighborhood of $\pm\mathbf{u}_i$, we have $|\langle \mathbf{c}_i, \mathbf{w} \rangle| < \|\mathbf{c}_i\|$ and $|\langle \mathbf{c}_j, \mathbf{w} \rangle| < \|\mathbf{c}_i\|$, thus $f_\infty(\mathbf{w}) < f_\infty(\pm\mathbf{u}_i) = \|\mathbf{c}_i\|$.

Now we show that there are no other local maxima. Given any unit vector $\mathbf{w} \notin \{\pm\mathbf{u}_1, \ldots, \pm\mathbf{u}_m\}$, let $i$ be such that $|\langle \mathbf{c}_i, \mathbf{w} \rangle| = f_\infty(\mathbf{w}) = \max_j |\langle \mathbf{c}_j, \mathbf{w} \rangle|$. We may rotate $\mathbf{w}$ slightly toward $\pm\mathbf{u}_i$ (where the sign matches the sign of $\langle \mathbf{c}_i, \mathbf{w} \rangle$) to obtain a new unit vector $\mathbf{w}'$ such that $|\langle \mathbf{c}_i, \mathbf{w}' \rangle| > |\langle \mathbf{c}_i, \mathbf{w} \rangle|$. Then we have $f_\infty(\mathbf{w}') \geq |\langle \mathbf{c}_i, \mathbf{w}' \rangle| > |\langle \mathbf{c}_i, \mathbf{w} \rangle| = f_\infty(\mathbf{w})$, so $\mathbf{w}$ is not a local maximum. $\blacksquare$

Ideally, we would hope to use the samples $\langle \mathbf{C}^t\mathbf{w}, \mathbf{s} \rangle$ to efficiently find the local maxima of $f_\infty(\mathbf{w})$, i.e., the columns of $\mathbf{C}$, but unfortunately we do not know how to do this. In the remainder of this section, we show that for our purposes the $\ell_k$ *norm* (for a large enough even integer $k$) is a good substitute for the $\ell_\infty$ norm, and that its maxima can be found relatively efficiently.

### 6.4.3 Using the $\ell_k$ Norm

Our first motivation for using the $\ell_k$ norm (for some even integer $k > 2$) is that for reasonably large $k$, it is a close approximation to the $\ell_\infty$ norm. Namely, for $\mathbf{v} \in \mathbb{R}^m$, it is a standard fact that $\|\mathbf{v}\|_\infty \leq \|\mathbf{v}\|_k \leq m^{1/k} \cdot \|\mathbf{v}\|_\infty$. Moreover, the first inequality is nearly tight when $\mathbf{v}$ has one dominant entry, which is indeed the case for $\mathbf{v} = \mathbf{C}^t\mathbf{w}$ in the regions of our interest

(where $\mathbf{w}$ is closely aligned with some column of $\mathbf{C}$). Naturally, a larger value of $k$ yields a better approximation to the $\ell_\infty$ norm, but as we see in Section 6.4.5, it also increases the sample complexity and running time of our attack.

Because the $\ell_k$ norm is only an approximation of the $\ell_\infty$ norm, the local maxima and other stationary points may differ. Fortunately, we prove in Theorem 22 that the $\ell_k$ norm typically has local maxima very close to those of of the $\ell_\infty$ norm. We have not found a proof that no other local maxima exist, but our experiments have yet to encounter one. (See Section 6.5 for details.)

The second reason for using the $\ell_k$ norm is its close connection to the *kth cumulant* of the random variables that arise in our learning problem, as shown in the following lemma.

**Lemma 33.** *Let $k$ be an even positive integer, let $\mathbf{v} \in \mathbb{R}^m$ be fixed, and let $\mathbf{s} \in \mathbb{R}^m$ be a random variable where each coordinate $s_i$ is independent and identically distributed to some random variable $X$. Then*

$$\kappa_k(\langle \mathbf{v}, \mathbf{s} \rangle) = \|\mathbf{v}\|_k^k \cdot \kappa_k(X).$$

*Proof.* We have

$$\kappa_k(\langle \mathbf{v}, \mathbf{s} \rangle) = \sum_{i=1}^{m} \kappa_k(v_i s_i) = \sum_{i=1}^{m} v_i^k \cdot \kappa_k(s_i) = \|\mathbf{v}\|_k^k \cdot \kappa_k(X),$$

by additivity of $\kappa_k$ for independent random variables, homogeneity of degree $k$, and the definition of the $\ell_k$ norm (for even $k$), respectively. ■

Setting $\mathbf{v} = \mathbf{C}^t\mathbf{w}$, Lemma 33 says that $\|\mathbf{C}^t\mathbf{w}\|_k^k$ is proportional to $\kappa_k(\langle \mathbf{w}, \mathbf{Cs} \rangle)$. As we show below in Section 6.4.5, the latter quantity can be estimated to a high degree of accuracy for any $\mathbf{w}$ using access to independent samples of the form $\mathbf{Cs}$.

For our optimization algorithm, we also need to analyze the *gradient* of the $\ell_k$ norm (actually, its $k$th power). For a vector $\mathbf{x}$, denote by $[\mathbf{x}]^j$ the operation that raises each entry of $\mathbf{x}$ to the $j$th power. Define the function $f_k : \mathbb{R}^n \to \mathbb{R}$ as

$$f_k(\mathbf{w}) = \|\mathbf{C}^t\mathbf{w}\|_k^k = \sum_{i=1}^{m} \langle \mathbf{c}_i, \mathbf{w} \rangle^k.$$

By linearity, the chain rule, and the fact that the transpose Jacobian $(D\langle \mathbf{c}, \mathbf{w} \rangle)^t = \mathbf{c}$ for

any $\mathbf{c} \in \mathbb{R}^n$, the gradient of $f_k$ is

$$\nabla f_k(\mathbf{w}) = \sum_{i=1}^{m} \nabla(\langle \mathbf{c}_i, \mathbf{w}\rangle^k) = k \cdot \sum_{i=1}^{m} \mathbf{c}_i \cdot \langle \mathbf{c}_i, \mathbf{w}\rangle^{k-1} = k \cdot \mathbf{C} \cdot [\mathbf{C}^t\mathbf{w}]^{k-1}. \tag{6.1}$$

We also show in Section 6.4.5 that this gradient can be expressed and measured in terms of the statistics of $\langle \mathbf{w}, \mathbf{Cs}\rangle$.

We now show that the $\ell_k$ norm has local maxima that are very close to those of the $\ell_\infty$ norm, i.e., the columns $\mathbf{c}_i$ of $\mathbf{C}$.

**Theorem 22.** *Let $m = \gamma^c$ for some constant $c > 2$. Then $f_k$ has a local maximum within distance $\delta = O(\gamma^{c-k+1})$ of every local maxima of $f_\infty$ as long as $k - 2 > 1.1(c+1)$.*

*Proof.* Define $g : S_n \to \mathbb{R}^n$ as $g(\mathbf{w}) = \nabla f_k(\mathbf{w})/\|\nabla f_k(\mathbf{w})\|$. We argue that $g$ must have a fixed point within distance $\delta$ of each normalized column $\mathbf{u}_i = \mathbf{c}_i/\|\mathbf{c}_i\|$ of $\mathbf{C}$. Say that a vector $\mathbf{w}$ has gap $\alpha$ if there is some $i$ such that $|\langle \mathbf{c}_i, \mathbf{w}\rangle| \geq \alpha \cdot |\langle \mathbf{c}_j, \mathbf{w}\rangle|$ for all $i \neq j$. Our argument will proceed in two steps: first, we show in Lemma 34 that there is some $\alpha > 0$ so that every $\mathbf{w}$ with gap $\alpha$ satisfies $g(\mathbf{w}) > \alpha$. This implies that there is a fixed point, i.e., for which $g(\mathbf{w}) = \mathbf{w}$, among the vectors having gap $\alpha$. Then, we show in Lemma 35 that if $\mathbf{w}$ has gap larger than $\alpha$, then $g(\mathbf{w})$ must be within distance $\delta$ of $\mathbf{u}_i$.

Note that for any vector $\mathbf{v} \in \mathbb{R}^m$, the $i$th coordinate of $\mathbf{C}^t\mathbf{Cv}$ is given by

$$\langle \mathbf{C}^t\mathbf{Cv}, \mathbf{e}_i\rangle = \langle \sum_\ell \mathbf{v}_\ell \mathbf{c}_\ell, \mathbf{Ce}_i\rangle = \sum_\ell \mathbf{v}_\ell \langle \mathbf{c}_\ell, \mathbf{c}_i\rangle.$$

Thus, if we set $\mathbf{v} = k \cdot [\mathbf{C}^t\mathbf{w}]^{k-1}$, then $\nabla f_k(\mathbf{w}) = \mathbf{Cv}$. We observe that when $\mathbf{w}$ has a gap $\alpha$, $g(\mathbf{w})$ is dominated by the contribution from $\mathbf{v}_i$:

$$\|\sum_{\ell \neq i} \mathbf{v}_\ell \mathbf{c}_\ell\| \leq \sum_{\ell \neq i} \mathbf{v}_\ell \|\mathbf{c}_\ell\| \leq |\mathbf{v}_i| \frac{m}{\alpha^{k-1}} \tag{6.2}$$

by the triangle inequality and the fact that each $\|\mathbf{c}_i\| = \|\mathbf{C}^t\mathbf{Ce}_i\| \leq 1$. We proceed to prove the two main lemmas.

**Lemma 34.** *If $\mathbf{w} \in \mathbb{R}^n$ has gap $\alpha > 0$, then $g(\mathbf{w})$ has gap $\alpha' \geq \gamma(1 - O(\frac{m\gamma}{\alpha^{k-1}}))^2$.*

*Proof.* Using equation 6.2, we have that

$$|\langle \mathbf{Cv}, \mathbf{c}_i\rangle| \geq |\mathbf{v}_i|\langle \mathbf{c}_i, \mathbf{c}_i\rangle - |\mathbf{v}_i|\frac{m}{\alpha^{k-1}} \tag{6.3}$$

$$|\langle \mathbf{Cv}, \mathbf{c}_j\rangle| \leq |\mathbf{v}_i|\langle \mathbf{c}_i, \mathbf{c}_j\rangle + |\mathbf{v}_i|\frac{m}{\alpha^{k-1}}. \tag{6.4}$$

Since $\mathbf{C}$ is $\gamma$-quasi-orthogonal, we have (6.4) is at most $\frac{1}{\gamma}|\mathbf{v}_i|(\langle \mathbf{c}_i, \mathbf{c}_i\rangle + \frac{\gamma m}{\alpha^{k-1}})$. Thus the gap of $g(\mathbf{w})$ is at least

$$\gamma \frac{\langle \mathbf{c}_i, \mathbf{c}_i\rangle - m/\alpha^{k-1}}{\langle \mathbf{c}_i, \mathbf{c}_i\rangle + \gamma m/\alpha^{k-1}} \geq \gamma(1 - O(\frac{m\gamma}{\alpha^{k-1}}))^2.$$

■

**Lemma 35.** *If $\mathbf{w} \in R^n$ has gap $\alpha'$, then*

$$\|g(\mathbf{w}) - \mathbf{u}_i\| \leq O\left(\frac{m}{\alpha'^{k-1}}\right).$$

*Proof.* Note that the magnitude of the vector in Equation 6.2 bounds the distance of $\nabla f_k(\mathbf{w})$ from the line spanned by $\mathbf{c}_i$. Normalizing gives us that

$$\|g(\mathbf{w}) - \mathbf{u}_i\| \leq \frac{|\mathbf{v}_i|\frac{m}{\alpha'^{k-1}}}{\|Cv\|} \leq \frac{|\mathbf{v}_i|\frac{m}{\alpha'^{k-1}}}{|\mathbf{v}_i|\|\mathbf{c}_i\|} = O\left(\frac{m}{\alpha'^{k-1}}\right).$$

■

To complete the proof of Theorem 22, suppose $m = O(\gamma^c)$ and consider any $w$ with gap $\alpha = \gamma^h$ for some $h < 1$. Applying Lemma 34 we have $\alpha' \geq \gamma(1 - \frac{2}{\gamma}) > \alpha$ as long as $k - 2 > \frac{c+1}{h}$. Applying Lemma 35 with $\alpha' = O(\gamma)$, we have $\delta \leq O(\gamma^{c-k+1})$.

■

### 6.4.4 Special-Purpose Optimization Algorithm

A standard algorithmic approach for multivariate optimization is *gradient ascent/descent*, in which one starts at an arbitrary point and iteratively steps in the direction of the gradient until an extremum is found. More advanced techniques such as Newton's method use additional information about the function (such as the Hessian) to converge more quickly. Prior works [Frieze *et al.*, 1996; Nguyen and Regev, 2009] use these approaches with an objective function that measures the fourth moment of the projected data.

For our objective function $f_k$, which measures the $k$th cumulant, we observe that the gradient has a nice geometric interpretation that inspires a somewhat different style of optimization algorithm. First recall that by the method of Lagrange multipliers, $f_k$ is maximized over the unit sphere $S_n$ only where $\nabla f_k(\mathbf{w})$ is parallel to $\mathbf{w}$ (this is a necessary but not sufficient condition).

Now consider the expression of the gradient from Equation (6.1). The vector $\mathbf{C}^t\mathbf{w} \in \mathbb{R}^m$ indicates the degree of alignment between $\mathbf{w}$ and each unknown column $\mathbf{c}_i$. The "powering" operation $[\mathbf{C}^t\mathbf{w}]^{k-1}$ *amplifies* the largest of these inner products relative to the others, in effect "pulling" the vector toward alignment with some standard basis vector $\mathbf{e}_i \in \mathbb{R}^m$. (A larger value of $k$ yields greater amplification; in the limit as $k$ approaches $\infty$, the result is perfectly aligned with $\mathbf{e}_i$.) Finally, multiplying on the left by $\mathbf{C}$ produces a gradient vector $\nabla f_k(\mathbf{w})$ that is closely aligned with $\mathbf{c}_i$, with relatively small contributions from the other $\mathbf{c}_j$. We may then iterate this process, with each iteration having better alignment with $\mathbf{e}_i$ by the powering operation, followed by better alignment with $\mathbf{c}_i$, until some fixed point is reached where $\mathbf{w}$ and $\nabla f_k(\mathbf{w})$ are parallel. Due to the small contributions of the other $\mathbf{c}_j$, we do not expect the process to converge exactly in the direction of $\mathbf{c}_i$, but to some direction very close to it.

Finally, once we have converged to the unit vector $\mathbf{w} = \mathbf{c}_i/\|\mathbf{c}_i\|$ in the direction of some $\mathbf{c}_i$, we need to scale $\mathbf{w}$ appropriately. By quasi-orthogonality, we have

$$\|\mathbf{c}_i\| = \frac{\langle \mathbf{c}_i, \mathbf{c}_i \rangle}{\|\mathbf{c}_i\|} = \frac{\|\mathbf{C}^t\mathbf{c}_i\|_\infty}{\|\mathbf{c}_i\|} = \|\mathbf{C}^t\mathbf{w}\|_\infty \approx \|\mathbf{C}^t\mathbf{w}\|_k = (f_k(\mathbf{w}))^{1/k}.$$

The above discussion naturally translates into the following simple "replace with gradient" optimization algorithm for finding some column $\mathbf{c}_i$. The algorithm requires oracle access to $f_k(\cdot)$ and $\nabla f_k(\cdot)$; below we show that these oracles may be implemented efficiently (to a high degree of accuracy).

1. Choose $\mathbf{w}$ uniformly at random from $S_n$.
2. Repeat until $\mathbf{w}$ and $\nabla f_k(\mathbf{w})$ are (nearly) parallel: Let $\mathbf{w} \leftarrow \nabla f_k(\mathbf{w})/\|\nabla f_k(\mathbf{w})\| \in S_n$.
3. Output $\mathbf{w} \cdot (f_k(\mathbf{w}))^{1/k}$.

### 6.4.5 Measuring the Statistics

In this section we show that the statistics needed by our optimization algorithm can be estimated efficiently. We rely on the standard Hoeffding bound to quantify the quality of our estimates.

**Lemma 36** (Hoeffding Bound)**.** *Let $X_1, \ldots, X_N$ be $N$ independent, identically distributed random variables in the interval $[a, b]$ with mean $\mu$, and let $X = \frac{1}{N}\sum_i X_i$. Then for any $\epsilon > 0$,*

$$\Pr[X \notin (1 \pm \epsilon) \cdot \mu] \leq 2 \exp\left(-2N \cdot \left(\frac{\epsilon \cdot \mu}{b-a}\right)^2\right).$$

#### 6.4.5.1 Moments and Cumulants

Define the statistics

$$\mu_{\mathbf{C},k}(\mathbf{w}) = \mu_k(\langle \mathbf{w}, \mathbf{Cs} \rangle) \quad \text{and} \quad \kappa_{\mathbf{C},k}(\mathbf{w}) = \kappa_k(\langle \mathbf{w}, \mathbf{Cs} \rangle),$$

where $\mathbf{s} \in [-1, 1)^m$ is uniformly random. Notice that $\mu_{\mathbf{C},k}(\mathbf{w}) = \kappa_{\mathbf{C},k}(\mathbf{w}) = 0$ for any odd integer $k$, because the distribution of $\langle \mathbf{w}, \mathbf{Cs} \rangle = \langle \mathbf{C}^t\mathbf{w}, \mathbf{s} \rangle$ is symmetric about the origin. Therefore, we need only estimate the statistics for even $k$.

The following lemma bounds the number of samples that suffice for estimating the cumulants $\kappa_{\mathbf{C},k}(\mathbf{w})$ to within relative (multiplicative) error $\epsilon > 0$. When $k$ is held constant, the number of samples grows with $m^{3k}/\epsilon^2$. (We have not attempted to optimize the bound, and improvements may be possible. Our experimental results in Section 6.5.2 indicate that significantly fewer samples would suffice.)

**Lemma 37.** *Let $k = 2k' \geq 2$ be an even integer, let $\epsilon, \delta \in (0, \frac{1}{2})$, and let $\alpha = |\kappa_k(X)|$ for a uniformly random variable $X$ over $[-1, 1)$. Define*

$$\tilde{\kappa}_k = \tilde{\mu}_k - \sum_{j=1}^{k-1} \binom{k-1}{j-1} \cdot \tilde{\kappa}_j \cdot \tilde{\mu}_{k-j}, \tag{6.5}$$

*where each estimate $\tilde{\kappa}_j$ of $\kappa_{\mathbf{C},j}(\mathbf{w})$ for even $j$ is computed recursively using the estimates $\tilde{\mu}_j = \frac{1}{N}\sum_i x_i^j$ of $\mu_{\mathbf{C},j}(\mathbf{w})$ for independent draws $x_1, \ldots, x_N$ of $\langle \mathbf{w}, \mathbf{Cs} \rangle$, for*

$$N \geq (6m^3)^k \cdot (k!/\alpha)^2 \cdot \epsilon^{-2} \log(k/\delta).$$

*Then except with probability at most $\delta$, we have*

$$\tilde{\kappa}_k \in (1 \pm \epsilon) \cdot \kappa_{\mathbf{C},k}(\mathbf{w}).$$

*Proof.* For even $j \geq 2$ and any $\mathbf{w} \in S^n$, any draw of $\langle \mathbf{w}, \mathbf{Cs} \rangle^j = \langle \mathbf{C}^t\mathbf{w}, \mathbf{s} \rangle^j$ is in the interval $[0, m^{j/2}]$, because $\|\mathbf{C}^t\mathbf{w}\| = 1$ and $\|\mathbf{s}\| \leq \sqrt{m}$. The draws have mean

$$\mu_{\mathbf{C},j}(\mathbf{w}) = \mathbf{E}_{\mathbf{s}}[\langle \mathbf{C}^t\mathbf{w}, \mathbf{s} \rangle^j] \geq \mathbf{E}_{\mathbf{s}}[\langle \mathbf{C}^t\mathbf{w}, \mathbf{s} \rangle^2]^{j/2} = (1/3)^{j/2},$$

by Jensen's inequality and the fact that the variance of $\langle \mathbf{C}^t\mathbf{w}, \mathbf{s} \rangle$ is $1/3$, since $\|\mathbf{C}^t\mathbf{w}\|^2 = 1$. Applying the Hoeffding bound with $N$ samples and the union bound (over all $k'$ estimates), all the $\tilde{\mu}_j$ are accurate estimates of $\mu_{\mathbf{C},j}(\mathbf{w})$ to within a multiplicative factor of $1 \pm \epsilon'$ except with probability at most $\delta$, where

$$\epsilon' = \frac{\epsilon \cdot \alpha}{(2m^2)^{k'} \cdot k!}.$$

We now analyze the accuracy of the estimate $\tilde{\kappa}_k$. Ignoring the vanishing terms that involve an odd-numbered moment, the recursive expansion of $\kappa_k = \kappa_{\mathbf{C},k}(\mathbf{w})$ in terms of the moments $\mu_j = \mu_{\mathbf{C},j}(\mathbf{w})$ has exactly $2^{k'-1}$ terms $T_i$. Each $T_i$ is a product of binomial coefficients that does not exceed $(k-1)!$ and up to $k'$ moments $\mu_{j_1}, \mu_{j_2}, \ldots, \mu_{j_s}$, where $\sum j_i = k$. As noted above, each moment $\mu_j \leq m^{j/2}$, so the product of moments appearing in each term is at most $m^{k'}$. Therefore, the sum of the terms' absolute values is

$$S = \sum |T_i| \leq (2m)^{k'} \cdot (k-1)!.$$

Now because each estimate $\tilde{\mu}_j$ is accurate to within a factor $1 \pm \epsilon'$, each product of up to $k'$ estimates $\prod \tilde{\mu}_{j_i}$ in the terms of $\tilde{\kappa}_k$ is accurate to within a $(1 \pm \epsilon')^{k'} \subseteq (1 \pm \epsilon' k)$ factor. Then by the triangle inequality, the total *additive* error of $\tilde{\kappa}_k$ is

$$|\tilde{\kappa}_k - \kappa_k| \leq \epsilon' k \cdot S \leq \epsilon \cdot \frac{\alpha}{m^{k'}}.$$

Now because $\mathbf{C}^t\mathbf{w} \in \mathbb{R}^m$ is a unit vector, we have $\|\mathbf{C}^t\mathbf{w}\|_k^k \geq m^{1-k'}$. By Lemma 33, we have

$$|\kappa_k| = |\kappa_k(X)| \cdot \|\mathbf{C}^t\mathbf{w}\|_k^k \geq \frac{\alpha}{m^{k'-1}}.$$

Therefore, the relative error of the estimate $\tilde{\kappa}_k$ is $|\tilde{\kappa}_k - \kappa_k| \,/\, |\kappa_k| \leq \epsilon$, as desired. ■

#### 6.4.5.2 Gradients

In order to express the gradient $\nabla f_k(\mathbf{w})$ in terms of statistics, we first invoke Lemma 33 to write

$$f_k(\mathbf{w}) = \|\mathbf{C}^t\mathbf{w}\|_k^k \propto \kappa_{\mathbf{C},k}(\mathbf{w})$$

(for simplicity, we elide the hidden constant). We then invoke the recursive definition of the cumulant and the product rule for the gradient to obtain

$$\nabla\kappa_{\mathbf{C},k}(\mathbf{w}) \propto \nabla\mu_{\mathbf{C},k}(\mathbf{w}) - \sum_{j=1}^{k-1}\binom{k-1}{j-1}\cdot\left(\nabla\kappa_{\mathbf{C},j}(\mathbf{w})\cdot\mu_{\mathbf{C},k-j}(\mathbf{w}) + \kappa_{\mathbf{C},j}(\mathbf{w})\cdot\nabla\mu_{\mathbf{C},k-j}(\mathbf{w})\right).$$

The gradient $\nabla f_k(\mathbf{w})$ can therefore be expressed recursively in terms of the moments $\mu_{\mathbf{C},j}(\mathbf{w})$ and their gradients. The gradient of the $j$th moment is

$$\nabla\mu_{\mathbf{C},j}(\mathbf{w}) = \mathbf{E}_{\mathbf{x}=\mathbf{Cs}}\left[\nabla(\langle\mathbf{w},\mathbf{x}\rangle^j)\right] = j\cdot\mathbf{E}_{\mathbf{x}=\mathbf{Cs}}\left[\langle\mathbf{w},\mathbf{x}\rangle^{j-1}\cdot\mathbf{x}\right].$$

These quantities may be estimated to a high degree of accuracy from many independent samples $\mathbf{x} = \mathbf{Cs}$, using analysis similar to that in Lemma 37.

## 6.5 Experiments

The chief difficulty with testing our proposed attack is the large sample complexity needed (at least in theory) to obtain accurate estimates of the objective function $f_k(\cdot)$ and its gradient. (E.g., see Lemma 37). We therefore conduct two types of experiments to test the efficacy of our approach.

First, we test the *idealized* version of our "replace with gradient" optimization algorithm (defined in Section 6.4.4), in which the oracles for $f_k(\cdot)$ and $\nabla f_k(\cdot)$ are implemented at essentially no cost using knowledge of the secret matrix $\mathbf{C}$ and the definition of $f_k(\mathbf{w}) = \|\mathbf{C}^t\mathbf{w}\|_k^k$. In order to simulate the effect of empirical estimation, we also test the performance of the algorithm when the quality of the oracles is degraded by various amounts of noise. We stress that *no other information about* $\mathbf{C}$ is available to the optimization algorithm, except what can be obtained via the oracles. The main purpose of this experiment is to test on real-world parameters (and beyond) the soundness of our approach involving the $\ell_k$ norm and our special-purpose optimization algorithm.

Next, we try to determine the number of signatures needed in practice to implement the oracles $f_k(\cdot)$ and $\nabla f_k(\cdot)$ with sufficient accuracy. In these experiments, we choose a random $\mathbf{C}$ and $\mathbf{w}$ and generate instances of the random variable $\langle\mathbf{w},\mathbf{Cs}\rangle$. Using Equation 6.5, we compute cumulant estimates for various sample sizes and compare these to the

actual cumulant (which is easily computed given $\mathbf{C}$). Note that by Lemma 33, $f_k$ can be computed from the cumulant with the same multiplicative error.

### 6.5.1 Performance Using Statistical Oracles

Our algorithm performed exceptionally well using the statistical oracles, even on outlandish parameter settings that would never be used in practice. We generated instances by first choosing a random $n \times m$ matrix $\mathbf{A}$ having normally distributed entries (with a standard deviation ranging from 3 to 100), then rounding its entries to integers. We then applied the sphering operation to obtain $\mathbf{C}$ and ran the optimization algorithm, implementing its oracles using knowledge of $\mathbf{C}$. Each iteration of the algorithm applied the inverse sphering transformation to the current value of $\mathbf{w}$, rounded it to the nearest integer vector, and checked for equality with any of the columns of $\mathbf{A}$. We also simulated the effect of empirical estimation by introducing artificial noise into the oracle's answers.

We ran the algorithm with various sizes for $\mathbf{A}$, with $n$ as small as 200 to $n$ as large as 500. We set $m$ to be anywhere from to $2n$ to as large as $5n$ (though we did not try this for the larger values of $n$). For each parameter setting, we performed the experiments without noise, and then we introduced noise by perturbing each gradient vector $\nabla f(w)$ with a random vector from a sphere of radius $\epsilon \cdot ||\nabla f(w)||$ for some $\epsilon > 0$.

With $k = 6$, the idealized algorithm discovered a (different) column of $\mathbf{A}$ within 7 iterations (usually fewer) in 99.7% of the runs, with every run successfully finding a column of $\mathbf{A}$ within 11 iterations. This performance held across all reasonable values of parameters, for dimensions ranging from $n = 200$ to $n = 500$, and for values of $m$ ranging from $m = 2n$ all the way up to $m = 200n$.

Our trials with noisy estimates at $k = 6$ were successful as well; at $n = 200$ and $m = 2n$ and noise rate $\epsilon = .031$, over half the trials returned a vector within 10 iterations and over 77% of the trials returned a vector within 19 iterations. Since our results at $m = 2n$ had no discernible difference from those at $m = 4n$, Figure 6.1 only presents the results for $m = 2n$. We also include a full histogram for $n = 200, m = 400$ and $k = 6$, without error and with error $\epsilon = .031$ (see Figure 6.1). These results can be contrasted with the algorithm's comparatively poor performance for $k = 4$, which essentially corresponds to the objective

| $k = 6$ | $\epsilon = 0$ | | $\epsilon = .021$ | | $\epsilon = .031$ | | |
|---|---|---|---|---|---|---|---|
| $n$ | 20 iterations | 11 iter | 20 iter | 11 iter | 40 iter | 20 iter | 11 iter |
| 200 | 100% | 99.7% | 100% | 100% | 93.2% | 77.15% | 55.93% |
| 300 | 100% | 99.99% | 100% | 99.9% | 79.0% | 59.24% | 37.65% |
| 400 | 100% | 99.99% | 100% | 100% | 61.6% | 38.49% | 21.84% |
| 500 | 100% | 100% | 100% | 100% | 40.4% | 23.69% | 12.28% |

Table 6.1: Each table entry gives the percentage of 10,000 sample runs for which a column of $\mathbf{A}$ was found within the stated number of iterations. The percentages at 40 iterations and $\epsilon = .031$ are out of 1000 runs.

| $k = 4$ | $\epsilon = 0$ | | $\epsilon = .021$ | | $\epsilon = .031$ | | |
|---|---|---|---|---|---|---|---|
| $n$ | 20 iterations | 11 iter | 20 iter | 11 iter | 40 iter | 20 iter | 11 iter |
| 200 | 9.6% | 8.8% | 3.59 % | 1.79 % | .1% | 0% | 0% |

Table 6.2: This table shows the analogous values for $k = 4$ when $n = 200$. Percentages are out of 10,000 samples, except for the one at 40 iterations, which is out of 1000 runs.

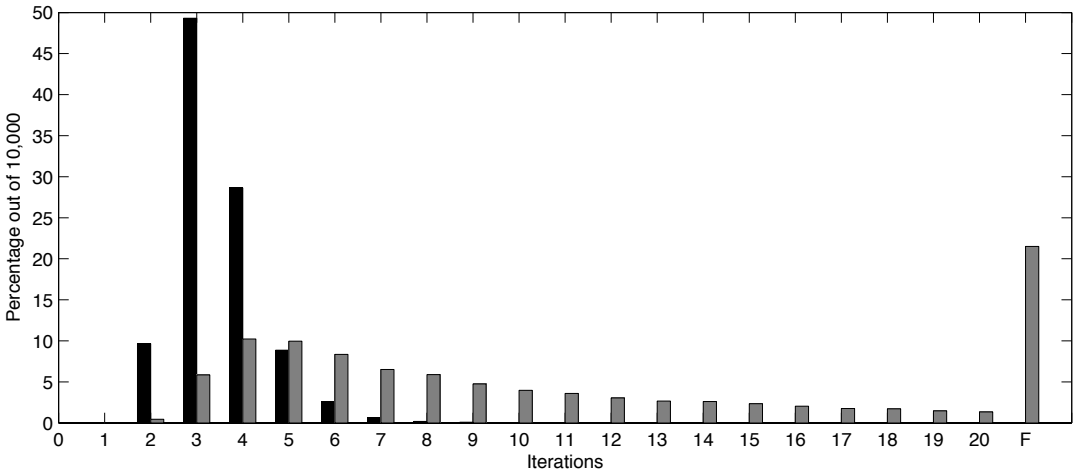


Figure 6.1: $n = 200, m = 2n$. Each bar at position $j$ on the $x$-axis represents the percentage of trials (out of 10,000) that found a column of $\mathbf{A}$ after $j$ iterations. Trials that did not find a column within 20 iterations are in the bin 'F'. The black bars represent trials without noise and gray bars trials with noise $\epsilon = .031$.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 0.0508 | 0.0280 | 0.0242 | 0.0135 | 0.0110 | 0.0079 | 0.0066 | 0.0043 | 0.0037 |
| 25 | 0.1193 | 0.0739 | 0.0562 | 0.0367 | 0.0260 | 0.0198 | 0.0144 | 0.0112 | 0.0062 |
| 30 | 0.2808 | 0.1691 | 0.1284 | 0.0681 | 0.0607 | 0.0433 | 0.0234 | 0.0169 | 0.0136 |
| 35 | 0.3630 | 0.2556 | 0.2551 | 0.1315 | 0.0807 | 0.0675 | 0.0472 | 0.0315 | 0.0236 |
| 39 | 0.5157 | 0.3384 | 0.2067 | 0.1379 | 0.1070 | 0.0716 | 0.0703 | 0.0489 | 0.0257 |

Table 6.3: The entry in the $i$th column of the row labeled by $m$ gives the averaged magnitudes of the relative error for $m$ columns using $N = 2^i \cdot 10^6$ samples.

function optimized in [Nguyen, 1999]. For the overcomplete basis problem, our (idealized) algorithm performs substantially better than the one from [Nguyen, 1999], demonstrating that the use of higher-order cumulants is an essential novelty in our approach. Figure 6.2 shows the results with different error rates for $n = 200$ and $m = 2n$.

### 6.5.2 Estimating the Cumulant

Our experimental evidence indicates that estimating the cumulants to tolerable accuracy (one which was tolerated in our noisy experiments) requires significantly fewer signatures than stated in Lemma 37. Specifically, we extrapolate from our results that $m^7$ samples will achieve good accuracy for $k = 6$.

To obtain this estimate, we first fix $m$ and $N$ (the number of samples) and choose a random $\mathbf{C}$ and $\mathbf{w}$ (or equivalently, a random unit vector $\mathbf{v} = \mathbf{C}^t\mathbf{w}$). Then we compute a set of *independent* estimates $\{\tilde{\kappa}_{C,6}(\mathbf{w})_j\}_{j=1\ldots20}$ using $N$ random draws of $\langle \mathbf{w}, \mathbf{C}s \rangle$ for each estimate. We compute the multiplicative error at sample size $N$ by averaging the magnitude of the multiplicative errors of each estimate $\tilde{\kappa}_{C,6}(\mathbf{w})_j$. We recorded these results for $m = 20, 25, 30, 35, 39$ and $N = [1 \cdot 10^6 \ldots 5.12 \cdot 10^8]$ in Figure 6.5.2. For all values of $m$, we plotted the logarithm of the error against the exponent $j$ in $N = 2^j \cdot 10^6$ and found that these values decreased linearly with $j$, at the same rate regardless of $m$. Next, we calculated for each $m$, the exponent $j$ required to obtain error below $e^{-3.5} < .0301$. This was done

by linear interpolation of the $\ln \epsilon$ versus $j$ values. From this data we computed the sample size, as a function of $m$, that suffices to obtain error at most .0301. A log/log graph reveals that a linear function should yield accurate prediction for larger values of $m$. Using linear regression, we obtain the relation $\ln N = 7.09 \ln m - 5.8047$.

# Chapter 7

# Conclusions and Future Work

- **Mansour's conjecture**

  Mansour's conjecture remains open for the class of DNF formulas and for a variety of natural subclasses, such as non-monotone read-k DNF formulas (in which each variable may appear at most $k$ times in the formula), CDNF formulas (Boolean functions which can be represented by polynomial-size DNF formulas *and* CNF formulas), and Talagrand's function [Talagrand, 1996; Mossel and O'Donnell, 2003b].

  Our approach in Chapter 3 may be contrasted with the Fourier analytic approach (via the switching lemma of Hastad's) taken by [Mansour, 1995a], and seems well-suited to DNF formulas which are "hard" for [Mansour, 1995a]. That is, the DNF formulas considered in Chapter 3 seem less amenable to simplification by random restrictions than other DNFs, such as CDNFs where different terms share many variables. Thus, it is natural to ask if the two approaches can be combined to strengthen [Mansour, 1995b] or obtain Mansour's conjecture for a broader class of DNF formulas.

- **Learning monotone DNF from random examples**

  The problem of efficiently learning all polynomial-size monotone DNF formulas is still open. A natural question is if our approach in Chapter 4 can be extended to handle a larger class of monotone DNF formulas. Since our algorithm is *statistical query* based [Blum *et al.*, 1994], a related question is whether any SQ based algorithm could learn monotone DNF formulas.

- **Hardness of monotone functions**

  An obvious goal for future work is to establish even sharper quantitative bounds on the cryptographic hardness of learning monotone functions. Blum, Burch, and Langford [Blum *et al.*, 1998] obtained their

  $$\frac{1}{2} + \frac{\omega(\log n)}{\sqrt{n}}$$

  information-theoretic lower bound by considering random monotone DNF that are constructed by independently including each of the $\binom{n}{\log n}$ possible terms of length $\log n$ in the target function. Can we match this hardness with a class of polynomial-size circuits?

  As mentioned in Section 5.1, it is natural to consider a pseudorandom variant of the construction in [Blum *et al.*, 1998] in which a pseudorandom rather than truly random function is used to decide whether or not to include each of the $\binom{n}{\log n}$ candidate terms. However, we have not been able to show that a function $f$ constructed in this way can be computed by a poly$(n)$-size circuit. Intuitively, the problem is that for an input $x$ with (typically) $n/2$ bits set to 1, to evaluate $f$ we must check the pseudorandom function's value on all of the $\binom{n/2}{\log n}$ potential "candidate terms" of length $\log n$ that $x$ satisfies. Indeed, the question of obtaining an efficient implementation of these "huge pseudorandom monotone DNF" has a similar flavor to Open Problem 5.4 of [Goldreich *et al.*, 2003]. In that question the goal is to construct pseudorandom functions that support "subcube queries" that give the parity of the function's values over all inputs in a specified subcube of $\{0,1\}^n$. In our scenario we are interested in functions $f$ that are pseudorandom only over the $\binom{n}{\log n}$ inputs with precisely $\log n$ ones (these inputs correspond to the "candidate terms" of the monotone DNF) and are zero everywhere else, and we only need to support "monotone subcube queries" (*i.e.,* , given an input $x$, we want to know whether $f(y) = 1$ for any $y \leq x$).

- **Learning an overcomplete basis**

  The actual distribution output by the key generation algorithm of NTRUSign is difficult to analyze; running experiments to generate NTRUSign keys and verifying that

they satisfy quasi-orthogonality in practice would be one way to confirm that the distribution we defined over the learning an overcomplete basis problem effectively models the task of recovering secret keys from NTRU signatures.

An obvious goal would be to give a proof of convergence and correctness of our special purpose optimization algorithm. While we were able to prove (see Theorem 22) that the optimization problem over the $k$-norm will have local maxima very close to the local maxima of the optimization problem over the $\infty$-norm (recall from Theorem 21 that these points reveal the secret keys), we do not rule out the existence of other local maxima that our algorithm might converge to (though we gave strong experimental evidence that these maxima would never be encountered). Our results in Sections 6.4.2 and 6.4.3 suggest that such a proof for the $k$-norm with $k = \log n$ is within reach. This would yield a quasi-polynomial time algorithm for the problem of learning an overcomplete basis.

Finally, it would be interesting to see if a similar attack is possible for GGH-style public-key encryption schemes (such as NTRU). GGH-style signature schemes and encryption schemes are both based on a "trapdoor" type property of the closest vector problem. A natural direction to pursue is to see if a chosen ciphertext attack is possible based on the HPP algorithm.

# Bibliography

[Aizenstein and Pitt, 1995] Howard Aizenstein and Leonard Pitt. On the learnability of disjunctive normal form formulas. *Machine Learning*, 19:183, 1995.

[Ajtai *et al.*, 1983] M. Ajtai, J. Komlos, and E. Szemeredi. An $O(n \log n)$ sorting network. *Combinatorica*, 3(1):1–19, 1983.

[Ajtai, 2004] Miklós Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[Akavia *et al.*, 2006] A. Akavia, O. Goldreich, S. Goldwasser, and D. Moshkovitz. On basing one-way functions on np-hardness. In *Proceedings of the 38th Symposium on Theory of Computing*, 2006.

[Allender *et al.*, 2006] E. Allender, L. Hellerstein, P. McCabe, T. Pitassi, and M. Saks. Minimizing DNF Formulas and $AC^0_d$ Circuits Given a Truth Table. In *Proc. 21st IEEE Conference on Computational Complexity (CCC)*, pages 237–251. IEEE Comp. Soc. Press, 2006.

[Amano and Maruoka, 2002] K. Amano and A. Maruoka. On learning monotone boolean functions under the uniform distribution. In *Proceedings of the 13th International Conference on Algorithmic Learning Theory (ALT)*, pages 57–68, 2002.

[Babai *et al.*, 1993] L. Babai, L. Fortnow, N. Nisan, and A.Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[Babai, 1986] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[Bazzi, 2007] Louay Bazzi. Polylogarithmic independence can fool DNF formulas. In *Proc. 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 63–73, 2007.

[Beals *et al.*, 1998] R. Beals, T. Nishino, and K. Tanaka. On the Complexity of Negation-Limited Boolean Networks. *SIAM Journal on Computing*, 27(5):1334–1347, 1998.

[Ben-David and Chor, 1992] S. Ben-David and B. Chor. 0. Goldreich, and M. Luby. On the theory of average case complexity. *Journal of Computer and System Sciences*, 44(2):193–219, 1992.

[Benjamini *et al.*, 1999] I. Benjamini, G. Kalai, and O. Schramm. Noise sensitivity of Boolean functions and applications to percolation. *Inst. Hautes Études Sci. Publ. Math.*, 90:5–43, 1999.

[Berkowitz, 1982] S. J. Berkowitz. On some relationships between monotone and non-monotone circuit complexity. Technical report, Technical Report, University of Toronto, 1982.

[Blum *et al.*, 1993] A. Blum, M. Furst, M. Kearns, and R. Lipton. Cryptographic Primitives Based on Hard Learning Problems. In *Advances in Cryptology – CRYPTO '93*, pages 278–291, 1993.

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

[Blum *et al.*, 1998] A. Blum, C. Burch, and J. Langford. On learning monotone boolean functions. In *Proc. 39th FOCS*, pages 408–415. IEEE Computer Society Press, 1998.

[Blum *et al.*, 2008] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory perspective on data privacy: New hope for releasing useful databases privately. In *Proc. 40th STOC*, pages pp. 609–618. ACM Press, 2008. Manuscript.

[Blum, 2003a] A. Blum. Learning a function of $r$ relevant variables (open problem). In *Proc. 16th Annual COLT*, pages 731–733, 2003.

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

[Bogdanov and Trevisan, 2003] A. Bogdanov and L. Trevisan. On worst-case to average-case reductions for np problems. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 308–317, 2003.

[Bogdanov and Trevisan, 2006] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *CoRR*, abs/cs/0606037, 2006.

[Boppana, 1997] Ravi B. Boppana. The average sensitivity of bounded-depth circuits. *Inform. Process. Lett.*, 63(5):257–261, 1997.

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

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

[Bshouty *et al.*, 2003] N. Bshouty, E. Mossel, R. O'Donnell, and R. Servedio. Learning DNF from Random Walks. In *Proceedings of the 44th IEEE Symposium on Foundations on Computer Science*, pages 189–198, 2003.

[Dachman-Soled *et al.*, 2008] D. Dachman-Soled, H.K. Lee, T. Malkin, R.A. Servedio, A. Wan, and H. Wee. Optimal Cryptographic Hardness of Learning Monotone Functions. *Lecture Notes in Computer Science*, 5125:36–47, 2008.

[Dachman-Soled *et al.*, 2009] Dana Dachman-Soled, Homin K. Lee, Tal Malkin, Rocco A. Servedio, Andrew Wan, and Hoeteck Wee. Optimal cryptographic hardness of learning monotone functions. *Theory of Computing*, 5(1):257–282, 2009.

[De *et al.*, 2009] Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. Technical Report 141, Electronic Colloquium in Computational Complexity, 2009.

[De Wolf, 2008] R. De Wolf. A brief introduction to Fourier analysis on the Boolean cube. *Theory of Computing Library–Graduate Surveys*, 1:1–20, 2008.

[Etesami and Cook, 2010] Omid Etesami and James Cook. personal communication, 2010.

[Feigenbaum and Fortnow, 1993] J. Feigenbaum and L. Fortnow. Random-self-reducibility of complete sets. *SIAM J. on Computing*, 22(5):994–1005, 1993.

[Feldman *et al.*, 2006] V. Feldman, P. Gopalan, S. Khot, and A. Ponnuswami. New results for learning noisy parities and halfspaces. In *Proc. 47th FOCS*, pages 563–576. IEEE Comp. Soc. Press, 2006.

[Feldman *et al.*, 2010] V. Feldman, H.K. Lee, and R.A. Servedio. Lower Bounds and Hardness Amplification for Learning Shallow Monotone Formulas. *Electronic Colloquium on Computational Complexity (ECCC)*, 2010.

[Friedgut and Kalai, 1996] Ehud Friedgut and Gil Kalai. Every monotone graph property has a sharp threshold. *Proceedings of the American Mathematical Society*, 124(10), 1996.

[Frieze *et al.*, 1996] Alan M. Frieze, Mark Jerrum, and Ravi Kannan. Learning linear transformations. In *FOCS*, pages 359–368, 1996.

[Gentry and Szydlo, 2002] Craig Gentry and Michael Szydlo. Cryptanalysis of the revised NTRU signature scheme. In *EUROCRYPT*, pages 299–320, 2002.

[Gentry *et al.*, 2008] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[Goldreich *et al.*, 1986] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):210–217, 1986.

[Goldreich *et al.*, 1997] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *CRYPTO*, pages 112–131, 1997.

[Goldreich *et al.*, 2003] O. Goldreich, S. Goldwasser, and A. Nussboim. On the Implementation of Huge Random Objects. In *Proc. 44th FOCS*, pages 68–79. IEEE Comp. Soc. Press, 2003.

[Gopalan *et al.*, 2008a] Parikshit Gopalan, Adam Kalai, and Adam R. Klivans. A query algorithm for agnostically learning DNF? In *21st Annual Conference on Learning Theory - COLT 2008, Helsinki, Finland, July 9-12, 2008*, pages 515–516. Omnipress, 2008.

[Gopalan *et al.*, 2008b] Parikshit Gopalan, Adam Tauman Kalai, and Adam R. Klivans. Agnostically learning decision trees. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 527–536. ACM, 2008.

[Graham *et al.*, 1994] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 1994.

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

[Hancock and Mansour, 1991b] Thomas Hancock and Yishay Mansour. Learning monotone $k$-$\mu$ DNF formulas on product distributions. pages 179–183, 1991.

[Håstad *et al.*, 1999] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[Håstad, 1986] Johan Håstad. *Computational Limitations for Small Depth Circuits*. MIT Press, 1986.

[Håstad, 2001] J. Håstad. A slight sharpening of LMN. *Journal of Computer and System Sciences*, 63(3):498–508, 2001.

[Healy *et al.*, 2006] A. Healy, S. Vadhan, and E. Viola. Using Nondeterminism to Amplify Hardness. *SIAM Journal on Computing*, 35(4):903–931, 2006.

[Hoffstein *et al.*, 1998] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

[Hoffstein *et al.*, 2001] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NSS: An NTRU lattice-based signature scheme. In *EUROCRYPT*, pages 211–228, 2001.

[Hoffstein *et al.*, 2003] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In *CT-RSA*, pages 122–140, 2003.

[Hoffstein *et al.*, 2005] Jeff Hoffstein, Nicholas Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. Performance improvements and a baseline parameter generation algorithm for NTRUSign. In *Proc. of Workshop on Mathematical Problems and Techniques in Cryptology*, pages 99–126, 2005.

[Hyvärinen and Inki, 2002] A. Hyvärinen and M. Inki. Estimating overcomplete independent component bases for image windows. *Journal of Mathematical Imaging and Vision*, 17:139–152, 2002.

[Hyvärinen *et al.*, 2001] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[IEEE, 2008] IEEE P1363.1: Public-key cryptographic techniques based on hard problems over lattices., October 2008. `http://grouper.ieee.org/groups/1363/lattPK/`.

[Impagliazzo and Levin, 1990] Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Proc. 31st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 812–821. IEEE, 1990.

[Impagliazzo and Wigderson, 1997] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 220–229. ACM, 1997.

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

[Jackson and Servedio, 2005a] Jeffrey Jackson and Rocco Servedio. Learning random log-depth decision trees under the uniform distribution. *SIAM J. on Computing*, 34(5), 2005.

[Jackson and Servedio, 2005b] Jeffrey C. Jackson and Rocco A. Servedio. On learning random DNF formulas under the uniform distribution. In *8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems and 9th International Workshop on Randomization and Computation — (RANDOM-APPROX 2005)*, volume 3624 of *Lect. Notes in Comp. Sci.*, pages 342–353. Springer-Verlag, 2005.

[Jackson and Servedio, 2006] J. Jackson and R. Servedio. On learning random DNF formulas under the uniform distribution. *Theory of Computing*, 2(8):147–172, 2006. (Preliminary version in RANDOM 2005).

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

[Jackson *et al.*, 2002] Jeffrey C. Jackson, Adam R. Klivans, and Rocco A. Servedio. Learnability beyond $AC^0$. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 776–784, New York, NY, USA, 2002. ACM Press.

[Jackson *et al.*, 2008a] J. Jackson, H. Lee, R. Servedio, and A. Wan. Learning random monotone dnf. *12th International Workshop on Randomness and Computation (RANDOM)*, pages 483–497, 2008.

[Jackson *et al.*, 2008b] Jeffrey C. Jackson, Homin K. Lee, Rocco A. Servedio, and Andrew Wan. Learning random monotone DNF. pages 483–497. Springer-Verlag, 2008.

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

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

[Kahn *et al.*, 1988] J. Kahn, G. Kalai, and N. Linial. The influence of variables on boolean functions. In *Proc. 29th FOCS*, pages 68–80. IEEE Comp. Soc. Press, 1988.

[Kalai *et al.*, 2008] A. Kalai, A. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008.

[Kasiviswanathan *et al.*, 2008] Shiva Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *Proc. 49th FOCS*, pages 531–540. IEEE Comp. Soc. Press, 2008. Manuscript.

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

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

[Kearns *et al.*, 1994b] M. Kearns, R. Schapire, and L. Sellie. Toward Efficient Agnostic Learning. *Machine Learning*, 17(2/3):115–141, 1994.

[Kearns *et al.*, 1994c] Michael J. Kearns, Ming Li, and Leslie G. Valiant. Learning boolean formulas. *J. ACM*, 41(6):1298–1328, 1994.

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

[Kharitonov, 1995] M. Kharitonov. Cryptographic lower bounds for learnability of Boolean functions on the uniform distribution. *Journal of Computer and System Sciences*, 50:600–610, 1995.

[Kleitman, 1966] D. J. Kleitman. Families of non-disjoint subsets. *Journal of Combinatorial Theory*, 1:153–155, 1966.

[Klivans *et al.*, 2004] A. Klivans, R. O'Donnell, and R. Servedio. Learning intersections and thresholds of halfspaces. *Journal of Computer & System Sciences*, 68(4):808–840, 2004.

[Klivans *et al.*, 2010] A. Klivans, H. Lee, and A. Wan. Mansour's Conjecture is True for Random DNF Formulas. In *Proceedings of the 23rd Conference on Learning Theory (COLT)*, 2010. To appear.

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

[Kushilevitz and Mansour, 1993a] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM J. on Computing*, 22(6):1331–1348, 1993.

[Kushilevitz and Mansour, 1993b] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM J. on Comput.*, 22(6):1331–1348, December 1993. Prelim. ver. in *Proc. of STOC'91*.

[Levin, 1986] L. Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, 1986.

[Linial *et al.*, 1993] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *J. ACM*, 40(3):607–620, 1993.

[Luby *et al.*, 1993] Michael Luby, Boban Velickovic, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *ISTCS 1993*, pages 18–24, 1993.

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

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

[Mansour, 1995b] Yishay Mansour. An $O(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution. *J. Comp. Sys. Sci.*, 50:543–550, 1995. Prelim. ver. in *Proc. of COLT'92*.

[Mossel and O'Donnell, 2003a] E. Mossel and R. O'Donnell. On the noise sensitivity of monotone functions. *Random Structures and Algorithms*, 23(3):333–350, 2003.

[Mossel and O'Donnell, 2003b] Elchanan Mossel and Ryan O'Donnell. On the noise sensitivity of monotone functions. *Random Struct. Algorithms*, 23(3):333–350, 2003.

[Mossel *et al.*, 2004a] E. Mossel, R. O'Donnell, and R. Servedio. Learning functions of $k$ relevant variables. *Journal of Computer & System Sciences*, 69(3):421–434, 2004. Preliminary version in *Proc. STOC'03*.

[Mossel *et al.*, 2004b] E. Mossel, R. O'Donnell, and R. Servedio. Learning functions of $k$ relevant variables. *J. Comput. & Syst. Sci.*, 69(3):421–434, 2004.

[Motwani and Raghavan, 1995] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

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

[Naor and Reingold, 2004] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

[Nepomnjascii, 1970] V.A. Nepomnjascii. Rudimentary predicates and Turing calculations. *Math Dokl.*, 11:1462–1465, 1970.

[Nguyen and Regev, 2009] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009. Preliminary version in Eurocrypt 2006.

[Nguyen, 1999] Phong Q. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto '97. In *CRYPTO*, pages 288–304, 1999.

[O'Donnell and Servedio, 2006] R. O'Donnell and R. Servedio. Learning monotone decision trees in polynomial time. In *Proceedings of the 21st Conference on Computational Complexity (CCC)*, pages 213–225, 2006.

[O'Donnell and Servedio, 2007] R. O'Donnell and R. Servedio. Learning monotone decision trees in polynomial time. *SIAM Journal on Computing*, 37(3):827–844, 2007.

[O'Donnell and Wimmer, 2009] R. O'Donnell and K. Wimmer. KKL, Kruskal-Katona, and Monotone Nets. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 725–734. IEEE, 2009.

[O'Donnell, 2004a] R. O'Donnell. Hardness amplification within NP. *Journal of Computer and System Sciences*, 69(1):68–94, 2004.

[O'Donnell, 2004b] R. O'Donnell. Hardness amplification within NP. *Journal of Computer and System Sciences*, 69(1):68–94, 2004.

[O'Donnell, 2008] R. O'Donnell. Some topics in analysis of Boolean functions. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 569–578. ACM, 2008.

[Peikert, 2009] Chris Peikert. Bonsai trees (or, arboriculture in lattice-based cryptography). Cryptology ePrint Archive, Report 2009/359, July 2009. `http://eprint.iacr.org/`.

[Razborov, 1985] A. Razborov. Lower bounds on the monotone network complexity of the logical permanent. *Mat. Zametki*, 37:887–900, 1985. English translation in: Math. Notes of the Academy of Sciences of USSR, Vol 37, pp. 485-493, 1985.

[Razborov, 2008] A. Razborov. A simple proof of Bazzi's theorem. Technical Report 81, Electronic Colloquium in Computational Complexity, 2008.

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

[Sellie, 2008a] L. Sellie. Learning Random Monoton DNF Under the Uniform Distribution. In *Proc. 21st Colt*, 2008.

[Sellie, 2008b] Linda Sellie. Learning random monotone DNF under the uniform distribution. pages 181–192, 2008.

[Sellie, 2009] Linda Sellie. Exact learning of random DNF over the uniform distribution. pages 45–54, 2009.

[Servedio, 2001] R. Servedio. On learning monotone DNF under product distributions. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, pages 558–573, 2001.

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

[Servedio, 2004b] R. Servedio. On learning monotone DNF under product distributions. *Information and Computation*, 193(1):57–74, 2004. Preliminary version in *Proc. COLT'01*.

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

[Szydlo, 2003] Michael Szydlo. Hypercubic lattice reduction and analysis of GGH and NTRU signatures. In *EUROCRYPT*, pages 433–448, 2003.

[Talagrand, 1996] M. Talagrand. How much are increasing sets positively correlated? *Combinatorica*, 16(2):243–258, 1996.

[Tardos, 1988] E. Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988.

[Trevisan, 2003] L. Trevisan. List-decoding using the xor lemma. *44th FOCS*, pages 126–135, 2003.

[Trevisan, 2005] Luca Trevisan. On uniform amplification of hardness in NP. In *Proc. 37th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2005.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. In *Proc. 16th Annual ACM Symposium on Theory of Computing (STOC)*, pages 436–445. ACM, 1984.

[Valiant, 1986] L. G. Valiant. Negation is Powerless for Boolean Slice Functions. *SIAM Journal on Computing*, 15:531, 1986.

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

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

# Appendix A

# The Entropy-Influence Conjecture and DNF formulas

The spectral entropy of a Boolean function $f$ is defined as $E(f) = \sum_{S\subseteq[n]} \hat{f}(S)^2 \log \frac{1}{\hat{f}(S)^2}$ and its *total influence* (also called its *average sensitivity*) as $I(f) := \sum_S |S|\hat{f}(S)^2$. The *entropy-influence* conjecture is that $E(f) = O(I(f))$.

The spectral entropy and the sparsity of a Boolean function are closely related. The following two facts relate these two quantities. The first was observed by Li-Yang Tan, and the proof of the second is due to Rocco Servedio.

**Fact 6.** *Let $f$ be any Boolean function. Then $f$ is $\epsilon$-concentrated on a set of at most $2^{E(f)/\epsilon}$ many of its Fourier coefficients.*

*Proof.* Let $\mathcal{C} = \{S \subseteq [n] : \hat{f}(S)^2 \leq 2^{-E(f)/\epsilon}\}$. Then $\sum_{S\in\mathcal{C}} \hat{f}(S)^2 \leq \epsilon$, otherwise:

$$E(f) = \sum_{S\subseteq[n]} \hat{f}(S)^2 \log \frac{1}{\hat{f}(S)^2} \geq \sum_{S\in\mathcal{C}} \hat{f}(S)^2 \log \frac{1}{\hat{f}(S)^2} > \epsilon \log 2^{-E(f)/\epsilon} = E(f).$$

By Parseval's inequality, the size of $\mathcal{C}$ can be at most $2^{E(f)/\epsilon}$. ■

**Lemma 38.** *Let $f$ be a Boolean function, and $s : \mathbb{Z}\to\mathbb{R}$ with $s(n) \geq 4$ for all $n$. Suppose that $f$ is $\epsilon$-concentrated on $s(n)^{\log 1/\epsilon}$ many coefficients, i.e., for every $\epsilon > 0$, there exists a set $\mathcal{C}(\epsilon)$ of size at most $s(n)^{\log 1/\epsilon}$ such that $\sum_{S\notin\mathcal{C}(\epsilon)} \hat{f}(S)^2 \leq \epsilon$. Then $E(f) \leq O(\log s(n))$.*

*Proof.* Set $\epsilon_i = 2^{-i}$ for $\epsilon_1, \cdots, \epsilon_{\log n}$, and set $J_i = \mathcal{C}(\epsilon_i) \setminus \mathcal{C}(\epsilon_{i-1})$, taking $\mathcal{C}(\epsilon_0) = \emptyset$. Then

$$E(f) = \sum_{S \subseteq [n]} \hat{f}(S)^2 \log \hat{f}(S)^{-2} \leq \sum_{i=1}^{\log n} \sum_{S \in J_i} \hat{f}(S)^2 \log \hat{f}(S)^{-2} + O(1),$$

where the constant term comes from coefficients which contribute at most $1/n$ to the total Fourier weight.

We have for any $i > 1$ that $W(J_i) := \sum_{S \in J_i} \hat{f}(S)^2 \leq \epsilon_{i-1}$, because $J_i \cap \mathcal{C}(\epsilon_{i-1}) = \emptyset$ and the Fourier weight outside of $\mathcal{C}(\epsilon_{i-1})$ is at most $\epsilon_{i-1}$. Furthermore, we have that $|J_i| \leq s(n)^i$, so

$$\sum_{S \in J_i} \hat{f}(S)^2 \log \hat{f}(S)^{-2} \leq W(J_i) \log \frac{s(n)^i}{W(J_i)} \leq \epsilon_{i-1} i (\log s(n) + 1).$$

The last inequality holds because $x \log 1/x$ is increasing for $x \in (0, 1/e)$.

Summing over the $J_i$, we obtain

$$\sum_{i=1}^{\log n} \epsilon_{i-1} i (\log s(n) + 1) \leq (\log s(n) + 1) \sum_{i=1}^{\log n} i \epsilon_{i-1} \leq 4(\log s(n) + 1),$$

which completes the proof. ■

Let $f$ be any $t(n)$-term DNF formula. If the entropy-influence conjecture holds, then there is some $c > 0$ such that $E(f) < cI(f)$. The total influence of any size-$t$ DNF formula is at most $O(\log t)$ [Boppana, 1997]. Thus, as a consequence of the entropy-influence conjecture, (this observation was noted in `http://terrytao.wordpress.com/2007/08/16/gil-kalai-the-entropyinfluence-conjecture/`) we obtain a sparse approximating polynomial (which is close to the sparsity conjectured by Mansour) by Applying Fact 6 to get concentration on at most $t^{O(1/\epsilon)}$ many coefficients.

Our results in Chapter 3 show that polynomial size random and read-$k$ (for constant $k$) DNF formulas are $\epsilon$-concentrated on $n^{O(\log 1/\epsilon)}$ many coefficients (see Fact 3 in Section 3.2). Lemma 38 tells us that the entropy of any such DNF formula is $O(\log n)$.

# Appendix B

# Proofs of Probabilistic Results for Random Monotone DNF

**Proof of Lemma 15.** We prove the lemma assuming that that $\emptyset \notin \mathcal{C}_{\mathscr{Y}}$. This is sufficient because if $\emptyset \in \mathcal{C}_{\mathscr{Y}}$, then $\mathcal{C}'_{\mathscr{Y}} = \mathcal{C}_{\mathscr{Y}} \setminus \emptyset$ is still contained in $\mathscr{Y}$, and applying the result to $\mathcal{C}'_{\mathscr{Y}}$ gives that $\prod_{U \in \mathcal{C}'_{\mathscr{Y}}} (\# g_U) \leq c \cdot \frac{t^{|\mathcal{C}'_{\mathscr{Y}}|-1} k^{2^s}}{\sqrt{n}}$ with probability at least $1 - \delta_{\text{terms}}$. Since $\# g_\emptyset \leq t$ and $|\mathcal{C}'_{\mathscr{Y}}| = |\mathcal{C}_{\mathscr{Y}}| - 1$, the conclusion of the lemma holds for $\mathcal{C}_{\mathscr{Y}}$ as well.

Fix any $\emptyset \neq U \in \mathcal{C}_{\mathscr{Y}}$ (note that since $U \in \mathcal{C}_{\mathscr{Y}}$ we also have $U \neq S$, and hence $|U| \leq s - 1 = \lfloor a \rfloor + 1$.). Recall that $f$ is chosen by picking each term $T_i$ to be a uniformly chosen set of $k$ distinct variables. The probability (over a random choice of $f$) that $T_1$ contains all the elements of $U$ and none of the elements of $S \setminus U$ is $\binom{n-s}{k-|U|} / \binom{n}{k}$; let us write $p_U$ to denote this quantity. Using the facts that $k = \Theta(\log n)$ and $1 \leq |U| < s = O(1)$, one can verify that

$$\frac{1}{2} (k/n)^{|U|} \leq p_U \leq (k/n)^{|U|}. \tag{B.1}$$

Since each of the $t$ terms of $f$ is chosen independently, we have that $\# g_U$ is binomially distributed according to $B(t, p_U)$, so $t \cdot \frac{1}{2}(\frac{k}{n})^{|U|} \leq \mathbf{E}[\# g_U] = t p_U \leq t(\frac{k}{n})^{|U|}$. Now recall that the Chernoff bound gives that $\Pr[X \geq (1+\zeta)\mathbf{E}[X]] \leq e^{-\zeta^2 t p/3}$ where $X$ is an independent and identically distributed random variable. For $X$ drawn from $B(t,p)$ and taking $\zeta = 1$, we get

$$\Pr\left[\# g_U > 2t\,(k/n)^{|U|}\right] \leq \Pr[\# g_U > 2 t p_U] \leq \exp(-t p_U/3) \leq \exp(-t(k/n)^{|U|}/6). \tag{B.2}$$

Suppose first that $|U| \leq s-2 = \lfloor a \rfloor$. If $|U| = 1$, then since $t \geq n^{3/2}$ we have that (B.2) is at most $\exp(-\sqrt{n}\log n)$. On the other hand, if $|U| > 1$ then $\lfloor a \rfloor \geq 2$, and since $t/n^{|U|} \geq t/n^{\lfloor a \rfloor} \geq 1$ we have that $(B.2) \leq \exp(-k^{|U|}/6) \leq n^{-\Omega(\log n)}$.

Now suppose that $|U| = s-1$. In this case we use the following form of the Hoeffding bound (see e.g. Exercise 4.1 in [Motwani and Raghavan, 1995]): if $\zeta > 2e-1$, then $\Pr[X > (1+\zeta)\mathbf{E}[X]] \leq 2^{-(1+\zeta)\mathbf{E}[X]}$ for $X$ drawn from $B(t,p)$. Let $\zeta$ be such that $(1+\zeta)t(k/n)^{|U|} = t^{1/2a}$; note that this gives

$$1+\zeta = t^{(1/(2a))-1}(n/k)^{|U|} = (\sqrt{n}/t)(n/k)^{|U|} \geq (\sqrt{n}/t)(n/k)^{a} = \sqrt{n}/\mathrm{polylog}(n) \gg 2e,$$

so we may indeed apply the Hoeffding bound for this choice of $\zeta$. Using (B.1), we obtain

$$\Pr[\#g_U > t^{1/2a}] \leq \Pr[\#g_U > (1+\zeta)tp_U] \leq 2^{\frac{-t^{1/2a}}{2}} \leq 2^{-\sqrt{n}/2}.$$

Taking a union bound over all possible sets $U \neq \emptyset$ (at most $2^s = O(1)$ many possibilities), we have that with probability at least $1 - \delta_{\text{terms}}$ over the draw of $f$, every such set $U \in \mathcal{C}_{\mathscr{Y}}$ satisfies

- if $|U| \leq s-2$ then $\#g_U \leq 2t(k/n)^{|U|}$; and
- if $|U| = s-1$ then $\#g_U \leq t^{1/2a}$.

We henceforth assume the above conditions are satisfied, and now show that this gives the bound (4.7).

We partition $\mathcal{C}_{\mathscr{Y}}$ according to the size of $U$: let $\mathcal{C}^A_{\mathscr{Y}} = \{U \in \mathcal{C}_{\mathscr{Y}} : |U| = s-1\}$ and $\mathcal{C}^B_{\mathscr{Y}} = \mathcal{C}_{\mathscr{Y}} \setminus \mathcal{C}^A_{\mathscr{Y}} = \{U \in \mathcal{C}_{\mathscr{Y}} : |U| \leq s-2\}$. Then

$$\prod_{U \in \mathcal{C}_{\mathscr{Y}}} (\#g_U) = \prod_{U \in \mathcal{C}^A_{\mathscr{Y}}} (\#g_U) \prod_{U \in \mathcal{C}^B_{\mathscr{Y}}} (\#g_U) \leq t^{|\mathcal{C}^A_{\mathscr{Y}}|/2a} \cdot (2t)^{|\mathcal{C}^B_{\mathscr{Y}}|} (k/n)^{\sum_{U \in \mathcal{C}^B_{\mathscr{Y}}} |U|}.$$

By definition of $\mathcal{C}_{\mathscr{Y}}$ we have that $\sum_{U \in \mathcal{C}_{\mathscr{Y}}} |U| \geq s$. Now if $|\mathcal{C}^A_{\mathscr{Y}}| = 0$ , then we have

$$\prod_{U \in \mathcal{C}_{\mathscr{Y}}} (\#g_U) \leq (2t)^{|\mathcal{C}_{\mathscr{Y}}|} (k/n)^{\sum_{U \in \mathcal{C}_{\mathscr{Y}}} |U|} \leq (2t)^{|\mathcal{C}_{\mathscr{Y}}|} (k/n)^{s} = 2^{|\mathcal{C}_{\mathscr{Y}}|} \frac{t^{|\mathcal{C}_{\mathscr{Y}}|-1} k^s}{n^{s-a}} \leq 2^{|\mathcal{C}_{\mathscr{Y}}|} \frac{t^{|\mathcal{C}_{\mathscr{Y}}|-1} k^s}{n}.$$

On the other hand, if $|\mathcal{C}^A_{\mathscr{Y}}| > 0$, then

$$\prod_{U \in \mathcal{C}_{\mathscr{Y}}} (\#g_U) \leq 2^{|\mathcal{C}^B_{\mathscr{Y}}|} t^{|\mathcal{C}^A_{\mathscr{Y}}|/2a + |\mathcal{C}^B_{\mathscr{Y}}|} (k/n)^{|\mathcal{C}^B_{\mathscr{Y}}|}.$$

Since $|\mathcal{C}_{\mathscr{Y}}| - (\frac{2a-1}{2a})|\mathcal{C}^A_{\mathscr{Y}}| = |\mathcal{C}^A_{\mathscr{Y}}|/2a + |\mathcal{C}^B_{\mathscr{Y}}|$, observing that $|\mathcal{C}^B_{\mathscr{Y}}| \leq 2^s$ it suffices to show that

$$2^{|\mathcal{C}^B_{\mathscr{Y}}|}k^{2^s}\frac{t^{|\mathcal{C}_{\mathscr{Y}}|}}{n^{\frac{2a-1}{2}|\mathcal{C}^A_y|+|\mathcal{C}^B_{\mathscr{Y}}|}} \leq c \cdot \frac{t^{|\mathcal{C}_{\mathscr{Y}}|-1}k^{2^s}}{\sqrt{n}},$$

which holds when $\frac{2a-1}{2}|\mathcal{C}^A_{\mathscr{Y}}| + |\mathcal{C}^B_{\mathscr{Y}}| \geq a + 1/2$. This inequality follows from the fact that $|\mathcal{C}_{\mathscr{Y}}| \geq 2$ (since $S \notin \mathcal{C}_{\mathscr{Y}}$ and $\cup_{U\in\mathcal{C}_{\mathscr{Y}}} U = S$), $|\mathcal{C}^A_{\mathscr{Y}}| \geq 1$, and $a \geq \frac{3}{2}$. ■

**Proof of Lemma 16.** For a monotone $t$-term DNF $f = T_1 \vee \cdots \vee T_t$, let $f^i$ denote the projected function obtained from $f$ by removing the term $T_i$ from $f$ and restricting all of the variables which were present in term $T_i$ to 1. For $\ell \neq i$ we write $T^i_\ell$ to denote the term obtained by setting all variables in $T_i$ to 1 in $T_\ell$, i.e. $T^i_\ell$ is the term in $f^i$ corresponding to $T_\ell$. Now the probability that $T_i$ is satisfied and no other $T_j$ is satisfied is given by $\Pr[T_i] \cdot \Pr[\overline{T^i_\ell} \text{ for all } \ell \neq i \mid T_i] = \Pr[T_i] \cdot \Pr[\overline{f^i}]$. Since $\Pr[T_i] = \frac{1}{2^k}$, it suffices to bound $\Pr[\overline{f^i}]$ from below. As in [Jackson and Servedio, 2006], we show that the following four facts all hold with probability $1 - \delta_{usat}$:

1. $\Pr[\overline{f^i}] \geq \prod_{\ell:\ell\neq i} \Pr[\overline{T^i_\ell}]$.
2. $\prod_{\ell:T^i_\ell \equiv T_\ell} \Pr[\overline{T^i_\ell}] > 1/16$.
3. $|\{T^i_\ell : \ell \neq i \wedge T^i_\ell \not\equiv T_\ell\}| \leq \frac{2tk^2}{n}$.
4. No term in $f^i$ has fewer than $k - \log\log t$ variables.

Together, these conditions imply that

$$\Pr[\overline{f^i}] \geq \prod_{\ell:T_\ell \equiv T^i_\ell} \Pr[\overline{T^i_\ell}] \prod_{\ell:T_\ell \not\equiv T^i_\ell, \ell\neq i} \Pr[\overline{T^i_\ell}] \geq \frac{1}{16}\left(1 - \frac{\log t}{2^k}\right)^{2tk^2/n} \geq \frac{1}{32}.$$

We now prove (1)–(4). To prove (1) note that

$$\Pr[\overline{f}] = \Pr[\overline{T_1} \wedge \overline{T_2} \wedge \cdots \wedge \overline{T_t}] = \Pr[\overline{T_1} | \overline{T_2} \wedge \cdots \wedge \overline{T_t}] \Pr[\overline{T_2} | \overline{T_3} \wedge \cdots \wedge \overline{T_t}] \cdots \Pr[\overline{T_{t-1}} | \overline{T_t}] \Pr[\overline{T_t}]$$

which is at least $\prod_{i=1}^t \Pr[\overline{T_i}]$ since $f$ is monotone. (Conditioning on terms being unsatisfied can only increase the number of variables set to 0 and thus can only increase the chances a particular term is unsatisfied).

For any $i$ and $\ell$ such that $T^i_\ell \equiv T_\ell$, we have $\Pr[\overline{T^i_\ell}] = \Pr[\overline{T_\ell}] = 1 - \Pr[T_\ell] = 1 - \frac{1}{2^k}$. Certainly there are at most $t$ such $T^i_\ell$, so (2) follows from the fact that $k = \lfloor \log t \rfloor$ so $(1 - \frac{1}{2^k})^t > 1/16$.

For (3), first we prove that with probability at least $1 - \exp(\frac{-tk}{3n})$, any variable appears in at most $\frac{2tk}{n}$ many terms. Each variable $v_j$ appears in each fixed term $T_\ell$ with probability $k/n$. Since the terms are chosen independently, the number of occurrences of $v_j$ is binomially distributed according to $B(t,p)$ with $p = k/n$. Now recall that the Chernoff bound gives that $\Pr[X \geq (1+\zeta)\mathbf{E}[X]] \leq e^{-\zeta^2 tp/3}$ where $X$ is drawn from $B(t,p)$. Taking $\zeta = 1$, we get that $\Pr[X > \frac{2tk}{n}] < \exp(\frac{-tk}{3n})$. If $T_\ell \not\equiv T^i_\ell$ then $T_\ell$ must contain some variable from $T_i$. Assuming every variable appears in at most $2tk/n$ terms, and term $T_i$ has at most $k$ variables, there can be at most $k \cdot 2tk/n$ such terms.

Finally, Lemma 3.5 of [Jackson and Servedio, 2006] gives that (4) holds with probability at least $1 - t^2(\frac{k^2}{n})^{\log\log t}$. Thus we have that conditions (1)–(4) all hold with probability at least $1 - \delta_{usat}$. ■

**Proof of Lemma 17.** Fix any sequence $\iota_1 < \cdots < \iota_j$ of $j$ terms. Let $v \leq jk$ be the number of distinct variables that occur in these terms. First, we will bound the probability that $v > w := jk - \log k$. Consider any particular fixed set of $w$ variables. The probability that none of the $j$ terms includes any variable outside of the $w$ variables is precisely $(\binom{w}{k}/\binom{n}{k})^j$. Thus, the probability that $v \leq w$ is by the union bound:

$$\Pr[v \leq w] \leq \binom{n}{w}\left(\frac{\binom{w}{k}}{\binom{n}{k}}\right)^j \leq \left(\frac{en}{w}\right)^w \left(\frac{w}{n}\right)^{jk} \leq \frac{e^{jk-\log k}(jk-\log k)^{\log k}}{n^{\log k}}.$$

Taking a union bound over all (at most $t^j$) sequences $1 \leq \iota_1 < \cdots < \iota_j \leq t$, we have that with probability $1 - \delta_{\text{simult}}$, every sequence of $j$ terms contains at least $w$ distinct variables, and thus for every sequence we have $\Pr[T_{\iota_1} \wedge \cdots \wedge T_{\iota_j}] \leq 2^{-w} = k/2^{jk}$. ■

**Proof of Lemma 18.** For any fixed $r \in \{1,\ldots,t\}$ and any fixed $S$ such that $|S| = s$, we have $\Pr[\text{all variables in } S \text{ occur in } T_r] = \frac{k(k-1)\cdots(k-s+1)}{n(n-1)\cdots(n-s+1)} \leq \left(\frac{k}{n}\right)^s$. Since terms are chosen independently, the probability that the variables in $S$ co-occur in a fixed collection of $\gamma+1$ terms is at most $\left(\frac{k}{n}\right)^{s(\gamma+1)}$. By the union bound, the probability that these variables co-occur in any collection of $\gamma+1$ terms is at most $\binom{t}{\gamma+1} \cdot (\frac{k}{n})^{s(\gamma+1)} \leq \left(\frac{tk^s}{n^s}\right)^{\gamma+1}$. Using the

union bound again, we have that the probability that any set of $s$ variables co-occurs in more than $\gamma$ terms is at most $\binom{n}{s} \cdot \left(\frac{tk^s}{n^s}\right)^{\gamma+1}$. Recalling that $t = n^a$, that $s = \lfloor a \rfloor + 2$, and that $k = \lfloor \log t \rfloor = O(\log n)$, we have that this probability is at most $\text{polylog}(n) \cdot n^{a(\gamma+1)-(a+1)\gamma} = \text{polylog}(n) \cdot n^{a-\gamma}$. By our choice of $\gamma$ this is at most $\delta_\gamma$, and the proof is done. ■