INTRACTABILITY RESULTS FOR SOME COMPUTATIONAL PROBLEMS

A Thesis
Presented to
The Academic Faculty

by

Ashok Kumar Ponnuswami

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Algorithms, Combinatorics and Optimization

College of Computing
Georgia Institute of Technology
August 2008

INTRACTABILITY RESULTS FOR SOME COMPUTATIONAL PROBLEMS

Approved by:

Subhash Khot, Advisor
College of Computing
Georgia Institute of Technology

Dana Randall
College of Computing
Georgia Institute of Technology

Robin Thomas
School of Mathematics
Georgia Institute of Technology

Santosh Vempala
College of Computing
Georgia Institute of Technology

H. Venkateswaran
College of Computing
Georgia Institute of Technology

Date Approved: 19 June 2008

ACKNOWLEDGEMENTS

I would like to thank my advisor Subhash Khot for his guidance and encouragement over the past four years. I owe a lot to his immense knowledge of the area and his ability to generate new ideas. Working with Subhash has been a wonderful learning experience that I will value for ever.

I would like to thank H. Venkateswaran for his encouragement and support, especially during my first two years at Georgia Tech. I am also indebted to Ajit Diwan and Sundar Vishwanathan at IIT Bombay. It is due to them that I developed an interest in Theoretical Computer Science.

I am also grateful to the faculty and my colleagues at the Theory group. I am fortunate to have had the opportunity to work with Parikshit Gopalan and Vitaly Feldman. Their ideas have contributed significantly to this thesis.

My stay at Georgia Tech has been made a bit easier by the friends I made here. A special thanks goes to the people who came to play cricket at Home Park and tolerated all the "wides" I bowled, to my Tumlin Street roommates (Aameek, Apurva, Barath and Tejas) and to my seniors Aranyak, Nayantara, Nikhil and Parikshit.

Finally, I would like to thank my parents and my sister Anitha for always supporting me.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

LIST OF TABLES vii

LIST OF FIGURES viii

SUMMARY ix

I INTRODUCTION 1

1.1 Agnostic Learning of Parities and Halfspaces 1

1.1.1 Learning Parities Under the Uniform Distribution 1

1.1.2 Agnostic Learning of Halfspaces 3

1.2 Max-Clique and Chromatic Number 4

1.3 Monotone Boolean Circuits for Bipartite Perfect Matching (BPM) 5

1.4 Contributions of this Thesis 5

1.4.1 Learning Parities under the Uniform Distribution 5

1.4.2 Hardness of Proper Agnostic Learning of Halfspaces 8

1.4.3 Max-Clique and Chromatic Number 11

1.4.4 Monotone Multilinear Boolean Circuits for BPM 13

II LEARNING PARITIES WITH NOISE 14

2.1 Introduction 14

2.2 Preliminaries 14

2.2.1 Learning Models 14

2.2.2 Fourier Transform 16

2.3 Finding Heavy Fourier Coefficients 17

2.4 Learning of Parities with Adversarial Noise 21

2.5 Learning DNF Expressions 22

2.6 Learning Juntas 23

2.7 Learning in the Presence of Random Noise 24

2.8 Conclusions 24

III HARDNESS OF AGNOSTIC LEARNING OF HALFSPACES 25
3.1 Introduction 25
3.2 Preliminaries 27
3.2.1 Approximation Algorithms and Hardness of Approximation 27
3.2.2 Optimization Versions of Learning Problems 28
3.2.3 Expanders 29
3.3 A Small Hardness Factor for Max-Lin-$\mathbb{Q}$ 31
3.4 Amplifying the Gap for Max-Lin-$\mathbb{Q}$ 33
3.4.1 Large constant gap for Max-Lin-$\mathbb{Q}$ 34
3.4.2 Super-constant gap for Max-Lin-$\mathbb{Q}$ 37
3.5 From Max-Lin-$\mathbb{Q}$ to HS-MA 40
3.6 Thresholds of Halfspaces 42
3.6.1 The Cryptosystem of Goldreich et al. [45] 43
3.6.2 Thresholds of Halfspaces are not PAC-learnable 44
3.6.3 Further Hardness of Learning Halfspaces with Adversarial Noise . 46
3.7 Conclusions 47
IV BETTER INAPPROXIMABILITY RESULTS FOR MAX-CLIQUE, CHROMATIC NUMBER AND MIN-3LIN-DELETION 48
4.1 Introduction 48
4.1.1 Max-Clique 48
4.1.2 Chromatic Number 50
4.2 Our Techniques 50
4.2.1 Reduction from Min-3Lin-Deletion to Max-Clique 51
4.2.2 Overview of Our Construction 52
4.3 Preliminaries 56
4.3.1 PCP Verifier of Khot [69] 56
4.3.2 From PCPs to Hardness of Approximating Max-Clique 61
4.3.3 Randomized PCPs and Hardness of Approximating Chromatic Number 61
4.4 Hardness of Approximating Min-Lin-Deletion 64
4.5 Hardness of Approximating Min-3Lin-Deletion 68

4.6 Hardness of Approximating Max-Clique 73
4.7 Hardness of Approximating Chromatic Number 75
4.8 Conclusions 77
V MONOTONE MULTILINEAR BOOLEAN CIRCUITS FOR BIPARTITE PERFECT MATCHING REQUIRE EXPONENTIAL SIZE 78
5.1 Introduction 78
5.2 Upper Bounds on Depth and Size of Monotone Boolean Circuits for PM . 79
5.3 Lower Bound on Size of Restricted Monotone Circuits for BPM 80
5.3.1 Lower Bound for Circuits in Simple Form 81
5.3.2 Lower Bound for Multilinear Circuits 83
5.4 Conclusions 89
APPENDIX A SOME BASIC NOTATION 90
REFERENCES 92

LIST OF TABLES

1 Hardness Results for Max-Clique. 49

LIST OF FIGURES

1 The main steps in proving an improved hardness factor for Min-3Lin-Deletion. 53

2 The hypercube $\mathbb{F}^m$. 70

3 Construction of circuit C_V for showing the depth upper bound. 80

4 Pruning the output from gate g_1 to gate g. 85

5 Disconnecting the output of gate g_1 to gate g in Lemma 5.3.8. 85

SUMMARY

In this thesis, we show results for some well-studied problems from learning theory and combinatorial optimization.

Learning Parities under the Uniform Distribution: We study the learnability of parities in the agnostic learning framework of Haussler [54] and Kearns et al. [65]. We show that under the uniform distribution, agnostically learning parities reduces to learning parities with random classification noise, commonly referred to as the noisy parity problem. Together with the parity learning algorithm of Blum et al. [21], this gives the first nontrivial algorithm for agnostic learning of parities. We use similar techniques to reduce learning of two other fundamental concept classes under the uniform distribution to learning of noisy parities. Namely, we show that learning of DNF expressions reduces to learning noisy parities of just logarithmic number of variables and learning of k-juntas reduces to learning noisy parities of k variables.

Agnostic Learning of Halfspaces: We give an essentially optimal hardness result for agnostic learning of halfspaces over $\mathbb{Q}^n$. We show that for any constant ϵ finding a halfspace that agrees with an unknown function on $1/2+\epsilon$ fraction of examples is NP-hard even when there exists a halfspace that agrees with the unknown function on $1-\epsilon$ fraction of examples. This significantly improves on a number of previous hardness results for this problem. We extend the result to $\epsilon = 2^{-\Omega(\sqrt{\log n})}$ assuming NP $\not\subseteq$ DTIME($2^{(\log n)^{O(1)}}$).

Majorities of Halfspaces: We show that majorities of halfspaces are hard to PAC-learn using any representation, based on the cryptographic assumption underlying the Ajtai-Dwork cryptosystem. This also implies a hardness result for learning halfspaces with a high rate of adversarial noise even if the learning algorithm can output any efficiently computable hypothesis.

Max-Clique, Chromatic Number and Min-3Lin-Deletion: We prove an improved

hardness of approximation result for two problems, namely, the problem of finding the size of the largest clique in a graph (also referred to as the Max-Clique problem) and the problem of finding the chromatic number of a graph. We show that for any constant $\gamma > 0$, there is no polynomial time algorithm that approximates these problems within factor $n/2^{(\log n)^{3/4+\gamma}}$ in an n vertex graph, assuming NP $\not\subseteq$ BPTIME($2^{(\log n)^{O(1)}}$). This improves the hardness factor of $n/2^{(\log n)^{1-\gamma'}}$ for some small (unspecified) constant $\gamma' > 0$ shown by Khot [69]. Our main idea is to show an improved hardness result for the Min-3Lin-Deletion problem.

An instance of Min-3Lin-Deletion is a system of linear equations modulo 2, where each equation is over three variables. The objective is to find the minimum number of equations that need to be deleted so that the remaining system of equations has a satisfying assignment. We show a hardness factor of $2^{\Omega(\sqrt{\log n})}$ for this problem, improving upon the hardness factor of $(\log n)^{\beta}$ shown by Håstad [52], for some small (unspecified) constant $\beta > 0$. The hardness results for Max-Clique and chromatic number are then obtained using the reduction from Min-3Lin-Deletion as given by Khot [69].

Monotone Multilinear Boolean Circuits for Bipartite Perfect Matching: A monotone Boolean circuit is said to be *multilinear* if for any AND gate in the circuit, the minimal representation of the two input functions to the gate do not have any variable in common. We show that monotone multilinear Boolean circuits for computing bipartite perfect matching require exponential size. In fact we prove a stronger result by characterizing the structure of the smallest monotone multilinear Boolean circuits for the problem.

CHAPTER I

INTRODUCTION

1.1 Agnostic Learning of Parities and Halfspaces

Parities and halfspaces are among the most fundamental concept classes in learning theory. Both of these concept classes are long-known to be learnable when examples given to the learning algorithm are guaranteed to be consistent with a function from the concept class [25, 83, 55]. Each example above is a pair $\langle x, b\rangle$ where $b \in \{0, 1\}$ is called the *label* of the point x. The examples labeled 1 are called *positive examples* and those labeled 0 are called *negative examples*. Real data is rarely completely consistent with a simple concept and therefore this strong assumption is a significant limitation of learning algorithms in Valiant's PAC learning model [104]. A general way to address this limitation was suggested by Haussler [54] and Kearns et al. [65] who introduced the *agnostic* learning model. In this model, informally, nothing is known about the process that generated the examples and the learning algorithm is required to do nearly as well as is possible using hypotheses from a given class. This corresponds to a common empirical approach when few or no assumptions are made on the data and a fixed space of hypotheses is searched to find the "best" approximation of the unknown function.

This model can also be thought of as a model of *adversarial classification noise* by viewing the examples as coming from some function f^* in the concept class $\mathcal{C}$, but with labels corrupted on an η^* fraction of examples. It is worth noting that unlike in most other models of noise, the learning algorithm is not required to recover the corrupted labels but only to classify correctly "almost" (in the PAC sense) $1 - \eta^*$ fraction of examples.

1.1.1 Learning Parities Under the Uniform Distribution

Let $x_1, x_2, \ldots, x_n$ be a set of binary variables. A parity function is the XOR of some subset $T = \{x_{i_1}, x_{i_2}, \ldots x_{i_t}\}$ of the set of variables. In the absence of noise, one can identify the set T by running Gaussian elimination on the given examples. The presence of noise in

the labels, however, leads to a number of challenging and important problems. We address learning of parities in the presence of two types of noise: *random classification noise* (each label is flipped with some fixed probability η randomly and independently) and adversarial classification noise (that is the agnostic learning). When learning with respect to the uniform distribution these problems are equivalent to decoding of random linear binary codes (from random and adversarial errors, respectively) both of which are long-standing open problems in coding theory [16, 85, 20]. Below we summarize the known results about these problems.

- **Adversarial Noise:** Without any restrictions on the distribution of examples the problem of *proper* agnostic learning parities is known to be NP-hard (proper here refers to the fact that the algorithm must produce a hypothesis from the same class of functions against which its performance is compared). This follows easily from NP-hardness of maximum-likelihood decoding of linear codes proved by Berlekamp et al. [16] (a significantly stronger version of this result follows from a celebrated result of Håstad [52]). We are unaware of non-trivial algorithms for this problem under any fixed distribution, prior to our work. The problem of learning parities with adversarial noise under the uniform distribution is equivalent to finding a significant Fourier coefficient of a Boolean function and related to the problem of decoding Hadamard codes. If the learner can ask *membership queries* (or queries that allow the learner to get the value of function f at any point), a celebrated result of Goldreich and Levin [46] gives a polynomial time algorithm for this problem. Later algorithms were given by Kushilevitz and Mansour [80], Levin [81], Bshouty et al. [28] and Feldman [38].

- **Random Noise:** The problem of learning parities in the presence of random noise, or the noisy parity problem is a notorious open problem in computational learning theory. Blum, Kalai and Wasserman [21] gave an algorithm for learning parity functions on n variables in the presence of random noise in time $2^{O(\frac{n}{\log n})}$ for any constant noise rate η. Their algorithm works for any distribution of examples. We will also consider a natural restriction of this problem in which the set T is of size at most k. A brute-force algorithm for this problem is to take $O(\frac{1}{1-2\eta}k \log n)$ samples and then find the

parity on k variables that best fits the data through exhaustive search in time $O(n^k)$. While some improvements are possible if η is a sufficiently small constant, this seems to be the best known algorithm for all constant $\eta < \frac{1}{2}$.

1.1.2 Agnostic Learning of Halfspaces

A halfspace is a linear threshold function over the input variables. More formally:

Definition 1.1.1 *Given a set $x_1, x_2, \ldots, x_n$ of variables, a* halfspace *is a function* $\texttt{sign}(\sum_{i\in[n]} w_i x_i - \theta)$*, where $w_1, \ldots, w_n, \theta$ are real numbers, $[n]$ denotes the set $\{1, 2, \ldots, n\}$ and* $\texttt{sign}(a)$ *is the function that is 1 if $a \geq 0$ and 0 otherwise.*

The problem of learning a halfspace is one of the oldest and well-studied problems in machine learning, dating back to the work on Perceptrons in the 1950s [1, 96, 86]. If a halfspace that classifies all the examples correctly does exist, one can find it in polynomial time using efficient algorithms for linear programming. When the positive and negative examples can be separated with a significant margin, simple online algorithms like Perceptron and Winnow are usually used (which also seem to be robust to noise [43, 4]).

In practice, positive examples often cannot be separated from negative examples using a linear threshold. Therefore much of the recent research in this area focuses on finding provably good algorithms when the data is noisy or inconsistent [19, 8, 29, 62]. Halfspaces are properly PAC learnable even in the presence of random noise: Blum et al. [19] showed that a variant of the Perceptron algorithm can be used in this setting (see also [29]). They also explicitly state that even a weak form of agnostic learning for halfspaces is an important open problem.

In this thesis we address proper agnostic learning of halfspaces. Uniform convergence results from Haussler [54] (see also Kearns et al. [65]) imply that learnability of halfspaces in the agnostic model is equivalent to the ability to come up with a function in a concept class $\mathcal{C}$ that has the optimal agreement rate with the given set of examples. It is known that finding a halfspace with the best agreement rate is NP-hard [60, 56]. However, for most practical purposes a hypothesis with any non-trivial (and not necessarily optimal) performance would still be useful. These weaker forms of the agnostic learning of a function

class are equivalent to a natural combinatorial approximation problem or, more precisely, to the following two problems: approximately minimizing the disagreement rate and approximately maximizing the agreement rate (sometimes referred to as *co-agnostic* learning). We refer to these optimization problems as HS-MD and HS-MA respectively.

The HS-MA problem (maximizing agreements with a halfspace) was first considered by Johnson and Preparata [60] who proved that finding a halfspace that has the optimal agreement rate with the given set of examples over $\mathbb{Z}^n$ is NP-hard (see also Hemisphere problem in [44]). In the context of agnostic learning Höffgen et al. [56] showed that the same is true for halfspaces over $\{0,1\}^n$. A number of results are known on hardness of approximating HS-MA. Amaldi and Kann [5], Ben-David et al. [15], and Bshouty and Burroughs [26] proved that HS-MA is NP-hard to approximate within factors $\frac{262}{261}$, $\frac{418}{415}$ and $\frac{85}{84}$ respectively.

The results of Höffgen et al. imply that approximating HS-MD, the problem of minimizing disagreement rate of a halfspace, within $c \log n$ is NP-hard for some constant c. Further Arora et al. [8] improved this factor to $2^{\log^{0.5-\delta} n}$ under the stronger complexity assumption NP $\not\subseteq$ DTIME($2^{(\log n)^{O(1)}}$).

1.2 Max-Clique and Chromatic Number

A *clique* in a graph is a subset of vertices such that any pair of vertices in the subset is connected by an edge. Max-Clique is the problem of finding the size of the largest clique in a graph. It has been a pivotal problem in the field of inapproximability, leading to the development of many important tools in this field.

The best known approximation algorithm for Max-Clique is due to Feige [36]. The algorithm achieves an approximation factor of $O(\frac{n(\log\log n)^2}{\log^3 n})$, where n is the number of vertices in the input graph. There has been a long series of results showing hardness of approximating Max-Clique starting with Feige et al. [33] and leading up to the result of Khot [69] (see Table 4.1.1.1 for details). Khot showed that assuming NP $\not\subseteq$ ZPTIME($2^{(\log n)^{O(1)}}$), Max-Clique cannot be approximated within a factor of $\frac{n}{2^{(\log n)^{1-\gamma'}}}$, for some small (unspecified) constant $\gamma' > 0$.

The *chromatic number* of a graph G is the minimum number of colors required to color the vertices of G such that for any edge, its end-points receive different colors. For this problem too, there has been a series of hardness of approximation results starting with Feige and Kilian [35] and leading to Khot [69]. Khot showed a hardness factor of $\frac{n}{2^{(\log n)^{1-\gamma'}}}$ for some constant $\gamma' > 0$, assuming NP $\not\subseteq$ ZPTIME($2^{(\log n)^{O(1)}}$).

1.3 Monotone Boolean Circuits for Bipartite Perfect Matching (BPM)

A monotone circuit is a Boolean circuit with only AND and OR gates. Let BPM denote the problem of finding whether a bipartite graph has a perfect matching. A Boolean circuit for BPM will have input gates for each pair of vertices of the bipartite graph and the gate is set to 1 if the edge is present and 0 otherwise. The circuit must output 1 if and only if there is a matching in the graph. Razborov [95] showed a super-polynomial lower bound on the size of monotone circuits for this problem, thus establishing a super-polynomial gap between the power of general Boolean circuits (that also have NOT gates) and monotone circuits. Tardos [102] subsequently showed that there are other polynomial-time computable functions for which the gap is exponential. It has been shown by Raz and Wigderson [94] that monotone circuits for perfect matching require linear depth. The upper bound on size of arithmetic circuits for permanent in Jerrum and Snir [59] yields a $2^{O(n)}$ size monotone Boolean circuit for BPM directly (replace the product and plus gates with AND and OR gates respectively). The depth of these circuits is $\Omega(n \log n)$. But it is also possible to construct linear depth monotone circuits for BPM (see Section 5.2). Thus although we know the tight (up to a constant) bound on the depth of monotone circuits for BPM, the question of whether the correct bound on the *size* of monotone circuits for this problem is exponential or only super-polynomial remains an interesting open problem.

1.4 Contributions of this Thesis

1.4.1 Learning Parities under the Uniform Distribution

In this thesis, we reduce a number of fundamental open problems on learning under the uniform distribution to learning noisy parities under the uniform distribution, establishing the central role of noisy parities in this model of learning.

1.4.1.1 Learning Parities with Adversarial Noise

We show that under the uniform distribution, learning parities with adversarial noise reduces to learning parities with random noise. In particular, our reduction and the result of Blum et al. [21] imply the first non-trivial algorithm for learning parities with adversarial noise under the uniform distribution.

Theorem 1.4.1 *For any constant $\eta < 1/2$, parities are learnable under the uniform distribution with adversarial noise of rate η in time $O(2^{\frac{n}{\log n}})$.*

Equivalently, this gives the first non-trivial algorithm for agnostically learning parities. The restriction on the noise rate in the algorithm of Blum et al. [21] translates into a restriction on the optimal agreement rate of the unknown function with a parity (namely it has to be a constant greater than 1/2).

Our main technical contribution is to show that an algorithm for learning noisy parities gives an algorithm that finds significant Fourier coefficients (i.e., correlated parities) of a function from random samples. Thus an algorithm for learning noisy parities gives an analogue of the Goldreich-Levin/Kushilevitz-Mansour algorithm for the uniform distribution, but without membership queries. This result is proved using Fourier analysis.

Subsequent to our work, Kalai et al. [61] gave an algorithm for agnostically learning parities under *any* distribution that runs in time $2^{O(n/\log n)}$. However, their algorithm does not output a parity as a hypothesis. They suggest a alternate definition for weak agnostic learning and show that a weak learning algorithm under this definition can be boosted to obtain an algorithm that outputs a hypothesis which is very close to the best possible. They also give a (non-proper) weak agnostic learning algorithm for parities in this model. These two results together imply their result for agnostically learning parities under an arbitrary distribution.

1.4.1.2 Learning DNF Formulae

Learning of DNF expressions from random examples is a famous open problem originating from Valiant's seminal paper [104] on PAC learning. In this problem we are given access to

examples of a Boolean function f on points randomly chosen with respect to distribution $\mathscr{D}$, and $\epsilon > 0$. The goal is to find a hypothesis that ϵ-approximates f with respect to $\mathscr{D}$ in time polynomial in n, $s = \texttt{DNF-size}(f)$ and $1/\epsilon$, where $\texttt{DNF-size}(f)$ is the number of terms in the DNF formula for f with the minimum number of terms. The best known algorithm for learning DNFs in this model was given by Klivans and Servedio [75] and runs in time $2^{\tilde{O}(n^{1/3})}$.

For learning DNFs under the uniform distribution a simple quasi-polynomial algorithm was given by Verbeurgt [107]. His algorithm essentially collects all the terms of size $\log(s/\epsilon) + O(1)$ that are consistent with the target function, i.e., do not accept negative points and runs in time $O(n^{\log(s/\epsilon)})$. We are unaware of an algorithm improving on this approach. Jackson [58] proved that DNFs are learnable under the uniform distribution if the learning algorithm is allowed to ask membership queries. This breakthrough and influential result gives essentially the only known approach to learning of unrestricted DNFs in polynomial time.

We show that learning of DNF expressions reduces to learning parities of $O(\log(s/\epsilon))$ variables with noise rate $\eta = 1/2 - \tilde{O}(\epsilon/s)$ under the uniform distribution.

Theorem 1.4.2 *Let $\mathcal{A}$ be an algorithm that learns parities of k variables over $\{0,1\}^n$ for every noise rate $\eta < 1/2$ in time $T(n,k,\frac{1}{1-2\eta})$ using at most $S(n,k,\frac{1}{1-2\eta})$ examples. Then there exists an algorithm that learns DNF expressions of size s in time $\tilde{O}(\frac{s^4}{\epsilon^2} \cdot T(n, \log B, B) \cdot S(n, \log B, B)^2)$, where $B = \tilde{O}(s/\epsilon)$.*

1.4.1.3 Learning k-juntas

A Boolean function is a k-junta if it depends only on k out of its n input variables. Learning of k-juntas was proposed by Blum and Langley [23, 17] as a clean formulation of the problem of efficient learning in the presence of irrelevant features. Moreover, for $k = O(\log n)$, a k-junta is a special case of a polynomial-size decision tree or a DNF expression. Thus, learning juntas is a first step toward learning polynomial-size decision trees and DNFs under the uniform distribution. A brute force approach to this problem would be to take $O(k \log n)$

samples and then run through all n^k subsets of possible relevant variables. The first non-trivial algorithm was given only recently by Mossel et al. [87] and runs in time roughly $O(n^{0.7k})$. Their algorithm relies on new analysis of the Fourier transform of juntas. However, even the question of whether one can learn k-juntas in polynomial time for $k = \omega(1)$ still remains open (see Blum [18]).

We give a stronger and simpler reduction from the problem of learning k-juntas to learning noisy parities of size k.

Theorem 1.4.3 *Let $\mathcal{A}$ be an algorithm that learns parities of k variables on $\{0,1\}^n$ for every noise rate $\eta < 1/2$ in time $T(n, k, \frac{1}{1-2\eta})$. Then there exists an algorithm that learns k-juntas in time $O(2^{2k}k \cdot T(n, k, 2^{k-1}))$.*

This reduction also applies to learning k-juntas with random noise. A parity of k variables is a special case of a k-junta. Thus we can reduce the noisy junta problem to a special case, at the cost of an increase in the noise level. By suitable modifications, the reduction from DNFs can also be made resilient to random noise (see Feldman et al. [40] for details).

Even though at this stage our reductions for DNFs and juntas do not yield new algorithms they establish connections between well-studied open problems. Our reductions allow one to focus on functions with known and simple structure viz parities, in exchange for having to deal with random noise. They show that a non-trivial algorithm for learning noisy parities of $O(\log n)$ variables will help make progress on a number of important questions regarding learning under the uniform distribution.

The reductions to learning noisy parity are described in Chapter 2 and are based on joint work with Vitaly Feldman, Parikshit Gopalan and Subhash Khot [40].

1.4.2 Hardness of Proper Agnostic Learning of Halfspaces

In this thesis we give essentially optimal hardness results for agnostic learning of halfspaces. Our result applies to the standard agnostic learning model in which the learning algorithm outputs a hypothesis from the same class as the class against which its performance is

measured. By analogy to learning in the PAC model this restriction is often referred to as *proper* agnostic learning.

We give the optimal (up to the second order terms) hardness result for HS-MA with examples over rationals. Namely we show that even if there is a halfspace that correctly classifies $1-\epsilon$ fraction of the input, it is hard to find a halfspace that is correct on $\frac{1}{2}+\epsilon$ fraction of the inputs for any $\epsilon > 0$ assuming P $\neq$ NP. Under stronger complexity assumptions, we can take ϵ to be as small as $2^{-\sqrt{\Omega(\log n)}}$ where n is the size of the input.

Theorem 1.4.4 *If* P $\neq$ NP *then for any constant* $\epsilon > 0$ *no polynomial time algorithm can distinguish between the following cases of the halfspace problem over* $\mathbb{Q}^n$*:*

- $1-\epsilon$ *fraction of the points can be correctly classified by some halfspace.*
- *No more than* $1/2+\epsilon$ *fraction of the points can be correctly classified by any halfspace.*

Moreover if we assume that NP $\not\subseteq$ DTIME$(2^{(\log n)^{O(1)}})$ *we can take* $\epsilon = 2^{-\Omega(\sqrt{\log n})}$.

Thus HS-MA is NP-hard to approximate within factor $2-\epsilon$ for any constant $\epsilon > 0$. This result implies that even weak agnostic learning of halfspaces is a hard problem. In an independent work Guruswami and Raghavendra [47] showed that an analogous hardness result is true even for halfspaces over points in $\{0,1\}^n$.

The crux of our proof is to first show a hardness result for solving systems of linear equations over the reals. Equations are easier to work with than inequalities since they admit certain *tensoring* and *boosting* operations which can be used for gap amplification. We show that given a system where there is a solution satisfying a $1-\epsilon$ fraction of the equations, it is hard to find a solution satisfying even an ϵ fraction. We then reduce this problem to the halfspace problem.

We note that the approximability of systems of linear equations over various fields is a well-studied problem. Håstad [52] showed that no non-trivial approximation is possible over $\mathbb{Z}_2$. Similar results are known for equations over $\mathbb{Z}_p$ and finite groups [52, 57]. However, to our knowledge this is the first optimal hardness result for equations over $\mathbb{Q}$. A natural question raised by our work is whether a similar hardness result holds for systems

of equations over $\mathbb{Q}$, where each equation involves only constantly many variables. Such a result was proved recently by Guruswami and Raghavendra [48].

1.4.2.1 Hardness of Representation Independent Learning

As opposed to proper-learning results, one could hope to show that a certain concept class is hard to PAC-learn, irrespective of the hypothesis representation, under some cryptographic assumptions [67]. We show such a hardness result for threshold circuits of depth 2. Since a single threshold gate is just a halfspace, these are thresholds of halfspaces. Such circuits correspond to two-level neural networks used in machine learning. They also capture several important concept classes: a convex polytope is an intersection of halfspaces in $\mathbb{R}^n$, whereas a DNF is a union of halfspaces over $\{0,1\}^n$.

There are numerous negative results known for proper learning of such concepts [106, 3], and for learning in the Statistical Query model [77]. Based on certain cryptographic assumptions, Kearns and Valiant [66] showed that constant depth threshold circuits cannot be learned over a certain distribution using any representation. Kharitonov [68] strengthened this result by allowing membership queries, and using the uniform distribution. We obtain a hardness result for threshold circuits of depth 2 independent of the hypothesis representation, based on the cryptographic assumption used in the Ajtai-Dwork lattice-based cryptosystem [2].

Theorem 1.4.5 *Assuming the security of the Ajtai-Dwork cryptosystem, there is no weak PAC-learning algorithm for the concept class of (unweighted) Threshold circuits of depth 2.*

To our knowledge, this is the first such result for depth-2 circuits of any kind. This result follows the general outline for proving inherent unpredictability of [66]. We show that the decryption of a modification of the Ajtai-Dwork cryptosystem [2] by Goldreich et al. [45] can be done by a depth-2 threshold circuit. This result was obtained independently by Klivans and Sherstov [76]. Known algorithms for learning intersections of k-halfspaces typically have running time exponential in k [22, 106, 73]. Our result suggests this is unavoidable. A recent result of Khot and Saket [72] gives another evidence. They show that for any number

l, it is hard to weakly PAC-learning the intersection of two halfspaces using any function of of l linear thresholds.

Finally, using the Discriminator Lemma of Hajnal et al. [49], we show that Theorem 1.4.5 implies the hardness of learning halfspaces with adversarial noise of high rate even when the learning algorithm is allowed to output a hypothesis of its choice.

Theorem 1.4.6 *Assuming the security of the Ajtai-Dwork cryptosystem, there exists a polynomial $p(n)$ such that halfspaces (in fact majorities) are not weakly learnable with adversarial noise of rate $\frac{1}{2} - \frac{1}{p(n)}$.*

This result is incomparable to Theorem 1.4.4, since one hand it is independent of the representation of the hypothesis. On the other hand, it applies only when the noise rate is very close to $\frac{1}{2}$.

The hardness results for learning halfspaces mentioned here are described in Chapter 3 and are based on joint work with Vitaly Feldman, Parikshit Gopalan and Subhash Khot [40, 39].

1.4.3 Max-Clique and Chromatic Number

We believe that it is an important open problem whether inapproximability of Max-Clique can be improved to $\frac{n}{2^{O(\sqrt{\log n})}}$, or even $n/\text{polylog}(n)$ (see Section 4.1.1.2). We show the following inapproximability results for Max-Clique and chromatic number, taking us closer to that goal.

Theorem 1.4.7 *Assuming* NP $\not\subseteq$ BPTIME($2^{(\log n)^{O(1)}}$)*, for any constant $\gamma > 0$, Max-Clique on an n vertex graph cannot be approximated within a factor better than $n/2^{(\log n)^{3/4+\gamma}}$ by any probabilistic polynomial time algorithm.*

Theorem 1.4.8 *Assuming* NP $\not\subseteq$ ZPTIME($2^{(\log n)^{O(1)}}$)*, for any constant $\gamma > 0$, chromatic number of an n vertex graph cannot be approximated within a factor better than $n/2^{(\log n)^{3/4+\gamma}}$ by any probabilistic polynomial time algorithm.*

Our main idea is to show an improved hardness factor for the Min-3Lin-Deletion problem. In the Min-3Lin-Deletion problem we are given a system $\boldsymbol{A}$ of linear equations modulo

2 where each equation is over exactly three variables. The goal is to find the smallest number of equations that must be deleted so that the remaining system has a satisfying assignment. We denote by $Opt(\boldsymbol{A})$ the smallest fraction of equations that must be deleted to accomplish this. The following result was shown by Håstad.

Lemma 1.4.9 ([52]) *For any constants $\epsilon, \delta > 0$, there exists a polynomial time algorithm $\mathcal{A}_1$ that when given a 3SAT formula ϕ of size n produces a Min-3Lin-Deletion instance $\boldsymbol{A}_1$ of size $N_1 = n^{O(1)}$ such that:*

- *(Yes Case:) If ϕ is satisfiable, then there exists an assignment that satisfies all but at most ϵ fraction of the equations. That is, $Opt(\boldsymbol{A}_1) \leq \epsilon$.*
- *(No Case:) If ϕ is not satisfiable, then no assignment satisfies more than $1/2 + \delta$ fraction of the equations. That is, $Opt(\boldsymbol{A}_1) \geq 1/2 - \delta$.*

Håstad also showed that the above result holds with $\epsilon = \delta = (\log N_1)^{-\beta}$ for some (tiny) constant $\beta > 0$ if N_1 and the running time of the reduction are allowed to be slightly super-polynomial in n. In particular Min-3Lin-Deletion-$((\log N_1)^{-\beta}, 0.4)$ is hard. This is the starting point for Khot's [69] hardness results for Max-Clique and chromatic number. Our main contribution is the following improved hardness result for Min-3Lin-Deletion. This in turn implies improved hardness results for Max-Clique and chromatic number:

Theorem 1.4.10 *There exists a $2^{O(\log^2 N_1)}$ time algorithm $\mathcal{A}$ that when given a Min-3Lin-Deletion instance $\boldsymbol{A}_1$ of size N_1 outputs a 7-regular Min-3Lin-Deletion instance $\boldsymbol{A}$ of size $N = 2^{O(\log^2 N_1)}$ such that:*

- *(Yes Case:) If $Opt(\boldsymbol{A}_1) \leq 0.1$, then $Opt(\boldsymbol{A}) \leq 2^{-\Omega(\sqrt{\log N})}$.*
- *(No Case:) If $Opt(\boldsymbol{A}_1) \geq 0.4$, then $Opt(\boldsymbol{A}) \geq \Omega(\log^{-3} N)$.*

The two operations *tensoring* and *boosting* we used to show this result also proved useful for showing hardness of approximating equations over rationals, and in turn yields the hardness result for maximizing agreement with a halfspace (Theorem 1.4.4).

The hardness results for Max-Clique and chromatic number are explained in detail in Chapter 4 and are based on joint work with Subhash Khot [70]

1.4.4 Monotone Multilinear Boolean Circuits for BPM

Since attempts to show an exponential lower bound on the size of monotone circuits for bipartite perfect matching (BPM) have not succeeded, it seems worthwhile to check if such a bound can be shown for restricted monotone circuits. A successful approach in the case of arithmetic circuits has been to consider a restriction called *multilinearity* first defined by Nisan and Wigderson [89]. We consider an analog of multilinear arithmetic circuits in the Boolean setting.

Definition 1.4.11 *We say a variable* influences *a gate if there is an assignment to all the other variables such that changing its value from 0 to 1 changes the output of the gate. We say a Boolean circuit is* multilinear *if at any AND gate with input gates g_1 and g_2, the sets of variables that influence g_1 and g_2 do not have any variable in common.*

This restriction is a bit weaker than that considered in Sengupta and Venkateswaran [99] and Krieger [79]. There the two inputs of any AND gate must be computed from disjoint sets of variables (i.e., every variable must be reachable from at most one of its two input gates). We call such circuits *constrained* multilinear. As stated in Krieger [79], constrained multilinear Boolean circuits generalize non-deterministic read-once branching programs and ordered binary decision diagrams. They are also capable of computing many functions efficiently (for example, for the threshold function $\sum_{i=1}^{n} x_i \geq k$, the DNF representation has $\binom{n}{k}$ size, but there are multilinear Boolean circuits of size $O(nk)$). We show that this is not the case for BPM.

Theorem 1.4.12 *Monotone multilinear Boolean circuits for BPM require exponential size.*

Krieger [79] showed that a constrained multilinear Boolean circuit of the smallest size for a monotone function is also monotone. So the above theorem also shows that all constrained multilinear Boolean circuits for BPM require exponential size.

The proof of Theorem 1.4.12 is given in Chapter 5 and is based on joint work with H. Venkateswaran [91].

CHAPTER II

LEARNING PARITIES WITH NOISE

2.1 Introduction

In this chapter, we describe our reductions from learning of parities with adversarial noise to learning of parities with random noise. We will also show applications of this reduction to learning of DNFs and juntas. We describe the main technical component of our reductions in Section 2.3: an algorithm that using an algorithm for learning noisy parities, finds a heavy Fourier coefficient of a Boolean function if one exists. Following Jackson [58], we call such an algorithm a *weak parity algorithm*.

The high-level idea of the reduction is to modify the Fourier spectrum of a function f so that it is "almost" concentrated at a single point. For this, we introduce the notion of a probabilistic oracle for real-valued functions $f : \{0,1\}^n \to [-1,1]$. We then present a transformation on oracles that allows us to clear the Fourier coefficients of f not belonging to a particular subspace of $\{0,1\}^n$. Using this operation we show that one can simulate an oracle which is close (in statistical distance) to a noisy parity.

2.2 Preliminaries

2.2.1 Learning Models

The learning models discussed in this work are based on Valiant's well-known PAC model [104]. In this model, for a concept $c : X \longrightarrow \{0,1\}$ and distribution $\mathscr{D}$ over X, an *example oracle* $\mathrm{EX}(c,\mathscr{D})$ is an oracle that upon request returns an example $\langle x, c(x)\rangle$ where x is chosen randomly with respect to $\mathscr{D}$. For $\epsilon \geq 0$ we say that function g ϵ-approximates a function f with respect to distribution $\mathscr{D}$ if $\Pr_{\mathscr{D}}[f(x) = g(x)] \geq 1 - \epsilon$.

Definition 2.2.1 *For a concept class $\mathcal{C}$, we say that an algorithm $\mathcal{A}$* PAC learns *$\mathcal{C}$, if for every $\epsilon > 0$, $c \in \mathcal{C}$, and distribution $\mathscr{D}$ over X, $\mathcal{A}$ given access to $EX(c,\mathscr{D})$ outputs, with probability at least $1/2$, a hypothesis h that ϵ-approximates c. The learning algorithm is* efficient *if it runs in time polynomial in $1/\epsilon$, and the* size *s of the learning problem where*

the size of the learning problem is equal to the length of an input to c plus the description length of c in the representation associated with $\mathcal{C}$. An algorithm is said to weakly learn $\mathcal{C}$ *if it produces a hypothesis h that $(\frac{1}{2} - \frac{1}{p(s)})$-approximates (or* weakly approximates*) c for some polynomial p.*

The *random classification noise* model was introduced by Angluin and Laird [6]. In this model for any $\eta \leq 1/2$ called the *noise rate* the regular example oracle $\mathrm{EX}(c, \mathscr{D})$ is replaced with the noisy oracle $\mathrm{EX}^{\eta}(c, \mathscr{D})$. On each call, $\mathrm{EX}^{\eta}(c, \mathscr{D})$, draws x according to $\mathscr{D}$, and returns $\langle x, c(x)\rangle$ with probability $1 - \eta$ and $\langle x, \neg c(x)\rangle$ having the wrong label with probability η. When η approaches $1/2$ the label of the corrupted example approaches the result of a random coin flip, and therefore the running time of algorithms in this model is allowed to polynomially depend on $\frac{1}{1-2\eta}$.

2.2.1.1 Agnostic Learning Model

The *agnostic* PAC learning model was introduced by Haussler [54] and Kearns et al. [65] in order to relax the assumption that examples are labeled by a concept from a specific concept class. In this model no assumptions are made on the function that labels the examples. In other words, the learning algorithm has no prior beliefs about the target concept (and hence the name of the model). The goal of the agnostic learning algorithm for a concept class $\mathcal{C}$ is to produce a hypothesis $h \in \mathcal{C}$ whose error on the target concept is close to the best possible by a concept from $\mathcal{C}$.

Formally, for two Boolean functions f and h and a distribution $\mathscr{D}$ over the domain, we define $\Delta_{\mathscr{D}}(f, h) = \Pr_{\mathscr{D}}[f \neq h]$. Similarly, for a concept class $\mathcal{C}$ and a function f, define $\Delta_{\mathscr{D}}(f, \mathcal{C}) = \inf_{h \in \mathcal{C}}\{\Delta_{\mathscr{D}}(f, h)\}$. Kearns et al. [65] define the agnostic PAC learning model as follows.

Definition 2.2.2 *An algorithm $\mathcal{A}$* agnostically *(PAC) learns a concept class $\mathcal{C}$ if for every $\epsilon > 0$, a Boolean function f and distribution $\mathscr{D}$ over X, $\mathcal{A}$ when given access to $EX(f, \mathscr{D})$ outputs, with probability at least $1/2$, a hypothesis $h \in \mathcal{C}$ such that $\Delta_{\mathscr{D}}(f, h) \leq \Delta_{\mathscr{D}}(f, \mathcal{C}) + \epsilon$. As before, the learning algorithm is* efficient *if it runs in time polynomial in s and $1/\epsilon$.*

The agnostic learning model can also be thought of as a model of adversarial noise. By definition, a Boolean function f differs from some function in $c \in \mathcal{C}$ on $\Delta_{\mathscr{D}}(f, \mathcal{C})$ fraction of the domain (the fraction is measured relative to distribution $\mathscr{D}$). Therefore f can be thought of as c corrupted by noise of rate $\Delta_{\mathscr{D}}(f, \mathcal{C})$. Unlike in the random classification noise model the points on which a concept can be corrupted are unrestricted and therefore we refer to it as *adversarial classification noise*. This noise model is also different from the model of malicious errors defined by Valiant [105] (see also Kearns and Li [64]) where the noise can affect both the label and the point itself, and thus possibly change the distribution of the data-points. Note that an agnostic learning algorithm will not necessarily find a hypothesis that approximates c – any other function in $\mathcal{C}$ that differs from f on at most $\Delta_{\mathscr{D}}(f, \mathcal{C}) + \epsilon$ fraction of the domain is acceptable. This way to view the agnostic learning is convenient when the performance of a learning algorithm depends on the rate of disagreement (that is the noise rate).

Besides algorithms with this strong agnostic guarantee it is natural and potentially useful to consider algorithms that output hypotheses with weaker yet non-trivial guarantees (e.g. having error of at most twice the optimum or within an additive constant of the optimum). We refer to such agnostic learning as *weakly agnostic* (along with a specific bound on the error when concreteness is required).

2.2.2 Fourier Transform

Our reduction uses Fourier-analytic techniques which were first introduced to computational learning theory by Linial et al. [82]. In this context we view Boolean functions as functions $f : \{0,1\}^n \to \{-1,1\}$. All probabilities and expectations are taken with respect to the uniform distribution unless specifically stated otherwise. For a Boolean vector $a \in \{0,1\}^n$ let $\chi_a(x) = (-1)^{a \cdot x}$, where '$\cdot$' denotes an inner product modulo 2, and let $\texttt{weight}(a)$ denote the Hamming weight of a.

We define an inner product of two real-valued functions over $\{0,1\}^n$ to be $\langle f, g \rangle = \mathrm{E}_x[f(x)g(x)]$. The technique is based on the fact that the set of all parity functions $\{\chi_a(x)\}_{a \in \{0,1\}^n}$ forms an orthonormal basis of the linear space of real-valued functions

over $\{0,1\}^n$ with the above inner product. This fact implies that any real-valued function f over $\{0,1\}^n$ can be uniquely represented as a linear combination of parities, that is $f(x) = \sum_{a\in\{0,1\}^n} \hat{f}(a)\chi_a(x)$. The coefficient $\hat{f}(a)$ is called Fourier coefficient of f on a and equals $\mathrm{E}_x[f(x)\chi_a(x)]$; a is called the *index* and $\mathtt{weight}(a)$ is called the *degree* of $\hat{f}(a)$. We say that a Fourier coefficient $\hat{f}(a)$ is θ-heavy if $|\hat{f}(a)| \geq \theta$. Let $L_2(f) = \mathrm{E}_x[(f(x))^2]^{1/2}$. Parseval's identity states that

$$(L_2(f))^2 = \mathrm{E}_x[(f(x))^2] = \sum_a \hat{f}^2(a).$$

For a Boolean function f, this implies that $L_2(f) = 1$.

2.3 Finding Heavy Fourier Coefficients

Given the example oracle for a Boolean function f the main idea of the reduction is to transform this oracle into an oracle for a noisy parity χ_a such that $\hat{f}(a)$ is a heavy Fourier coefficient of f. First we define probabilistic oracles for real-valued functions in the range $[-1,1]$.

Definition 2.3.1 *For any function $f : \{0,1\}^n \to [-1,1]$ a probabilistic oracle $\mathbb{O}(f)$ is the oracle that produces samples $\langle x, b\rangle$, where x is chosen randomly and uniformly from $\{0,1\}^n$ and $b \in \{-1,+1\}$ is a random variable with expectation $f(x)$.*

For a Boolean f this defines exactly $\mathrm{EX}(f, \mathscr{U})$, where $\mathscr{U}$ is the uniform distribution. Random classification noise can also be easily described in this formalism. For $\theta \in [-1,1]$, and $f : \{0,1\}^n \to \{-1,1\}$, define $\theta f : \{0,1\}^n \to [-1,1]$ as $\theta f(x) = \theta \cdot f(x)$. A simple calculation shows that $\mathbb{O}(\theta f)$ is just an oracle for $f(x)$ with random noise of rate $\eta = 1/2 - \theta/2$. Our next observation is that if the Fourier spectra of f and g are close to each other, then their oracles are close in statistical distance.

Claim 2.3.2 *The statistical distance between the outputs of $\mathbb{O}(f)$ and $\mathbb{O}(g)$ is upper-bounded by $L_2(f-g)$.*

Proof: For a given x, the probability that $\mathbb{O}(f)$ outputs $\langle x, 1\rangle$ is $(1+f(x))/2$ and the probability that it outputs $\langle x, -1\rangle$ is $(1-f(x))/2$. Therefore the statistical distance between $\mathbb{O}(f)$ and $\mathbb{O}(g)$ equals $\mathrm{E}_x\left[|f(x) - g(x)|\right]$. By Cauchy-Schwartz inequality,

$$(\mathrm{E}_x[|f(x)-g(x)|])^2 \leq \mathrm{E}_x\left[(f(x)-g(x))^2\right]$$

and therefore the statistical distance is upper bounded by $L_2(f-g)$. □

We now describe the main transformation on a probabilistic oracle that will be used in our reductions. For a function $f:\{0,1\}^n \to [-1,1]$ and a matrix $A \in \{0,1\}^{m\times n}$ define an A-projection of f to be

$$f_A(x) = \sum_{a\in\{0,1\}^n, Aa=1^m} \hat{f}(a)\chi_a(x),$$

where the product Aa is performed mod 2.

Lemma 2.3.3 *For the function f_A defined above:*

1. $f_A(x) = \mathrm{E}_{p\in\{0,1\}^m}[f(x\oplus A^T p)\chi_{1^m}(p)]$.
2. *Given access to the oracle $\mathbb{O}(f)$ one can simulate the oracle $\mathbb{O}(f_A)$.*

Proof: Note that for every $a\in\{0,1\}^n$ and $p\in\{0,1\}^m$,

$$\chi_a(A^T p) \;=\; (-1)^{a^T\cdot(A^T p)} \;=\; (-1)^{(Aa)^T\cdot p} \;=\; \chi_{Aa}(p)$$

Thus if $Aa=1^m$ then $\mathrm{E}_p[\chi_a(A^T p)\chi_{1^m}(p)] = \mathrm{E}_p[\chi_{Aa\oplus 1^m}(p)] = 1$ otherwise it is 0. Now let

$$g_A(x) = \mathrm{E}_{p\in\{0,1\}^m}[f(x\oplus A^T p)\chi_{1^m}(p)].$$

We show that g_A is the same as the function f_A by computing its Fourier coefficients.

$$\begin{aligned}
\widehat{g_A}(a) &= \mathrm{E}_x[\mathrm{E}_p[f(x\oplus A^T p)\chi_{1^m}(p)\chi_a(x)]] \\
&= \mathrm{E}_p[\mathrm{E}_x[f(x\oplus A^T p)\chi_a(x)]\chi_{1^m}(p)] \\
&= \mathrm{E}_p[\hat{f}(a)\chi_a(A^T p)\chi_{1^m}(p)] \\
&= \hat{f}(a)\mathrm{E}_p[\chi_a(A^T p)\chi_{1^m}(p)]
\end{aligned}$$

Therefore $\widehat{g_A}(a) = \hat{f}(a)$ if $Aa=1^m$ and $\widehat{g_A}(a)=0$ otherwise. This is exactly the definition of $f_A(x)$.

For Part 2, we sample $\langle x, b\rangle$, choose random $p\in\{0,1\}^m$ and return $\langle x\oplus A^T p, b\cdot\chi_{1^m}(p)\rangle$. The correctness follows from Part 1 of the Lemma. □

We will use Lemma 2.3.3 to project f in a way that separates one of its significant Fourier coefficients from the rest. We will do this by choosing A to be a random $m \times n$ matrix for appropriate choice of m.

Lemma 2.3.4 *Let $f : \{0,1\}^n \to [-1,1]$ be any function, and let $s \neq 0^n$ be any vector. Choose A randomly and uniformly from $\{0,1\}^{m \times n}$. With probability at least $2^{-(m+1)}$, the following conditions hold:*

$$\widehat{f_A}(s) = \hat{f}(s) \quad (1)$$

$$\sum_{a \in \{0,1\}^n \setminus \{s\}} \widehat{f_A}^2(a) \leq L_2^2(f) 2^{-m+1} \quad (2)$$

Proof: Event (1) holds if $As = 1^m$, which happens with probability 2^{-m}.

For every $a \in \{0,1\}^n \setminus \{s, 0^n\}$ and a randomly uniformly chosen vector $v \in \{0,1\}^n$,

$$\Pr_v[v \cdot a = 1 \mid v \cdot s = 1] = 1/2$$

$$\text{Therefore,} \quad \Pr_A[Aa = 1^m \mid As = 1^m] = 2^{-m}$$

Whereas for $a = 0^n$, $\Pr_A[Aa = 1^m] = 0$. Hence

$$\mathrm{E}_A \left[\sum_{a \in \{0,1\}^n \setminus \{s\}} \widehat{f_A}^2(a) \,\middle|\, As = 1^m \right] \leq \sum_{a \in \{0,1\}^n \setminus \{s\}} 2^{-m} \hat{f}^2(a) \leq 2^{-m} L_2^2(f).$$

By Markov's inequality,

$$\Pr_A \left[\sum_{a \in \{0,1\}^n \setminus \{s\}} \widehat{f_A}^2(a) \geq 2^{-m+1} L_2^2(f) \,\middle|\, As = 1^m \right] \leq 1/2.$$

Thus conditioned on event (1), event (2) happens with probability at least 1/2. So both events happen with probability at least $2^{-(m+1)}$. □

Finally, we show that using this transformation, one can use an algorithm for learning noisy parities to get a weak parity algorithm.

Theorem 2.3.5 *Let $\mathcal{A}$ be an algorithm that learns parities of k variables over $\{0,1\}^n$ for every noise rate $\eta < 1/2$ in time $T(n,k,\frac{1}{1-2\eta})$ using at most $S(n,k,\frac{1}{1-2\eta})$ examples. Then there exists an algorithm* `WP-R` *that for every function $f : \{0,1\}^n \to [-1,1]$ that has a θ-heavy Fourier coefficient s of degree at most k, given access to $\mathbb{O}(f)$, with probability at least $1/2$, finds s. Furthermore,* `WP-R` *runs in time $O(T(n,k,1/\theta) \cdot S^2(n,k,1/\theta))$ and uses $O(S^3(n,k,1/\theta))$ random examples.*

Proof: Let $S = S(n,k,1/\theta)$. The algorithm `WP-R` proceeds in two steps:

1. Let $m = \lceil 2\log S \rceil + 3$. Let $A \in \{0,1\}^{m\times n}$ be a randomly chosen matrix and $\mathbb{O}(f_A)$ be the oracle for A-projection of f. Run the algorithm $\mathcal{A}$ on $\mathbb{O}(f_A)$.

2. If $\mathcal{A}$ stops in $T(n,k,1/\theta)$ steps and outputs r with $\texttt{weight}(r) \leq k$, check that r is at least θ-heavy and if so, output it.

Let s be a θ-heavy Fourier coefficient of degree at most k. Our goal is to simulate an oracle for a function that is close to a noisy version of $\chi_s(x)$.

By Lemma 2.3.4, in Step 1, with probability at least 2^{-m-1} , we create a function f_A such that $|\widehat{f_A}(s)| \geq \theta$ and

$$\sum_{a\neq s} \widehat{f_A}^2(a) \;\leq\; 2^{-m+1}L_2^2(f) \;\leq\; \frac{L_2^2(f)}{4S^2} \;\leq \frac{1}{4S^2}.$$

By Claim 2.3.2, the statistical distance between the oracle $\mathbb{O}(f_A)$ and oracle $\mathbb{O}(\widehat{f_A}(s)\chi_s(x))$ is bounded by

$$L_2(f_A - \widehat{f_A}(s)\chi_s(x)) = \left(\sum_{a\neq s}(\widehat{f_A}^2(a))\right)^{1/2} \leq \frac{1}{2S}\ ,$$

hence this distance is small. Since $\mathcal{A}$ uses at most S samples, with probability at least $\frac{1}{2}$, it will not notice the difference between the two oracles. But $\mathbb{O}(\widehat{f_A}(s)\chi_s(x))$ is exactly the noisy parity χ_s with noise rate $1/2 - \widehat{f_A}/2$. If $\widehat{f_A} \geq \theta$ we will get a parity with $\eta \leq 1/2 - \theta/2 < 1/2$ and otherwise we will get a negation of χ_s with $\eta \leq 1/2 - \theta/2$. Hence we get $(1-2\eta)^{-1} \leq 1/\theta$, so the algorithm $\mathcal{A}$ will learn the parity s when executed either with the oracle $\mathbb{O}(f_A)$ or its negation. We can check that the coefficient produced by $\mathcal{A}$ is indeed heavy using Chernoff bounds, and repeat until we succeed. Using $O(2^m) = O(S^2)$

repetitions, we will get a θ-heavy Fourier coefficient of degree k with probability at least $1/2$. An A-projection always clears the coefficient $\hat{f}(0^n)$ and therefore we need to check whether this coefficient is θ-heavy separately. $\square$

Remark 2.3.6 *A function f can have at most $L_2^2(f)/\theta^2$ θ-heavy Fourier coefficients. Therefore by repeating* `WP-R` $O((L_2^2(f)/\theta^2) \cdot \log(L_2(f)/\theta)) = \tilde{O}(L_2^2(f)/\theta^2)$ *times we can, with high probability, obtain all the θ-heavy Fourier coefficients of f as it is required in some applications of this algorithm.*

2.4 Learning of Parities with Adversarial Noise

A weak parity algorithm is in its essence an algorithm for learning of parities with adversarial noise. In particular, Theorem 2.3.5 gives the following reduction from adversarial to random noise.

Theorem 2.4.1 *The problem of learning parities with adversarial noise of rate $\eta < \frac{1}{2}$ reduces to learning parities with random noise of rate η.*

Proof: Let f be a parity χ_s corrupted by noise of rate η. Then $\hat{f}(s) = \mathrm{E}[f\chi_s] \geq (1-\eta) + (-1)\eta = 1 - 2\eta$. Now apply the reduction from Theorem 2.3.5 setting $k = n$. We get an oracle for the function $\hat{f}(s)\chi_s(x)$, which is $\chi_s(x)$ with random noise of level η. $\square$

Blum et al. [21] give a sub-exponential algorithm for learning noisy parities.

Lemma 2.4.2 ([21]) *Parity functions on $\{0,1\}^n$ can be learned in time and sample complexity $2^{O(\frac{n}{\log n})}$ in the presence of random noise of rate η for any constant $\eta < \frac{1}{2}$.*

This algorithm together with Theorem 2.4.1 gives Theorem 1.4.1.

One can also interpret Theorem 2.4.1 in terms of coding theory problems. Learning a parity function with noise is equivalent to decoding a random linear code from the same type of noise. More formally, we say that a code C is an $[m, n]$ code if C is a binary linear code of block length m and message length n. Any such code can be described by its $n \times m$ *generator matrix* G as follows: $C = \{xG \mid x \in \{0,1\}^n\}$. A *random linear* $[m, n]$ *code* C is produced by choosing randomly and uniformly a generator matrix G for C (that is, each

element of G equals to the outcome of an unbiased coin flip). It is now easy to verify that Theorem 2.3.5 implies the following result.

Theorem 2.4.3 *Assume that there exists an algorithm* `RandCodeRandError` *that corrects a random linear* $[m, n]$ *code from random errors of rate* η *with probability at least* $1/2$ *(over the choice of the code, errors, and the random bits of the algorithm) in time* $T(m, n)$*. Then there exists an algorithm* `RandCodeAdvError` *that corrects a random linear* $[M, n]$ *code from up to* $\eta \cdot M$ *errors with probability at least* $1/2$ *(over the choice of the code and the random bits of the algorithm) in time* $O(m^2 \cdot T(m, n))$ *for* $M = O(m^3)$*.*

The sample bounds in Theorem 2.3.5 correspond to the block length of linear codes. Note that for $\eta \geq 1/4$, there might be more than one codeword within the relative distance η. In this case, by repetitively using `RandCodeAdvError` as in Remark 2.3.6, we can list-decode the random code.

2.5 Learning DNF Expressions

Jackson's [58] celebrated result gives a way to use a weak parity algorithm and Freund's [42] boosting algorithm to build an algorithm for learning DNF expressions with respect to the uniform distribution. His approach can be adapted to our setting. We give an outline of the algorithm and omit the now-standard analysis.

We view a probability distribution $\mathscr{D}$ as a density function and define its L_∞ norm. Jackson's algorithm is based on the following lemma (we use a refinement from Bshouty and Feldman [27]).

Lemma 2.5.1 ([27, Lemma 18]) *For any Boolean function* f *of DNF-size* s *and any distribution* $\mathscr{D}$*, over* $\{0,1\}^n$ *there exists a parity function* χ_a *such that* $|\mathrm{E}_{\mathscr{D}}[f\chi_a]| \geq \frac{1}{2s+1}$ *and*

$$\texttt{weight}(a) \leq \log\left((2s+1)L_\infty(2^n\mathscr{D})\right).$$

This lemma implies that DNFs can be weakly learned by finding a parity correlated with f under distribution $\mathscr{D}(x)$ which is the same as finding a parity correlated with the function $2^n\mathscr{D}(x)f(x)$ under the uniform distribution. The range of $2^n\mathscr{D}(x)f(x)$ is not necessarily

$[-1, 1]$, whereas our WP-R algorithm was defined for functions with this range. So in order to apply Theorem 2.3.5, we first scale $2^n\mathscr{D}(x)f(x)$ to the range $[-1, 1]$ and obtain the function $\mathscr{D}'(x)f(x)$, where $\mathscr{D}'(x) = \mathscr{D}(x)/L_\infty(2^n\mathscr{D})$ ($L_\infty(\mathscr{D})$ is known to the boosting algorithm). We then get the probabilistic oracle $\mathbb{O}(\mathscr{D}'(x)f(x))$ by flipping a ± 1 coin with expectation $\mathscr{D}'(x)f(x)$. Therefore a θ-heavy Fourier coefficient of $2^n\mathscr{D}(x)f(x)$ can be found by finding a $\theta/L_\infty(2^n\mathscr{D})$-heavy Fourier coefficient of $\mathscr{D}'(x)f(x)$ and multiplying it by $L_\infty(2^n\mathscr{D})$. We summarize this generalization in the following lemma.

Lemma 2.5.2 *Let $\mathcal{A}$ be an algorithm that learns parities of k variables over $\{0,1\}^n$ for every noise rate $\eta < 1/2$ in time $T(n, k, \frac{1}{1-2\eta})$ using at most $S(n, k, \frac{1}{1-2\eta})$ examples. Then there exists an algorithm WP-R' that for every real-valued function ϕ that has a θ-heavy Fourier coefficient s of degree at most k, given access to random uniform examples of ϕ, finds s in time $O(T(n, k, L_\infty(\phi)/\theta) \cdot S(n, k, L_\infty(\phi)/\theta)^2)$ with probability at least $1/2$.*

The running time of WP-R' depends on $L_\infty(2^n\mathscr{D})$ (polynomially if T is a polynomial) and therefore gives us an analogue of Jackson's algorithm for weakly learning DNFs. Hence it can be used with a boosting algorithm that produces distributions that are *polynomially-close* to the uniform distribution; that is, the distribution function is bounded by $p2^{-n}$ where p is a polynomial in learning parameters (such boosting algorithms are called *p-smooth*). In Jackson's result [58], Freund's boost-by-majority algorithm [42] is used to produce distribution functions bounded by $O(\epsilon^{-(2+\rho)})$ (for arbitrarily small constant ρ). More recently, Klivans and Servedio have observed [74] that a later algorithm by Freund [41] produces distribution functions bounded by $\tilde{O}(\epsilon)$. By using WP-R' with this boosting algorithm in the same way as in Jackson's DNF learning algorithm, we obtain Theorem 1.4.2.

2.6 Learning Juntas

For the class of k-juntas, we can get a simpler reduction with better parameters for noise. Since there are at most 2^k non-zero coefficients and each of them is at least 2^{-k+1}-heavy, for a suitable choice of m, the projection step is likely to isolate just one of them. This leaves us with an oracle $\mathbb{O}(\hat{f}(s)\chi_s)$. Since $|\hat{f}(s)| \geq 2^{-k+1}$, the noise parameter is bounded

by $\eta < 1/2 - 2^{-k}$. Using Remark 2.3.6, we will obtain the complete Fourier spectrum of f by repeating the algorithm $O(k2^{2k})$ times. The proof of Theorem 1.4.3 follows from these observations. Instead of repeating `WP-R` one can also use a simple recursive procedure of Mossel et al. [87, Sec 3.1] that requires only k invocations of `WP-R`.

2.7 Learning in the Presence of Random Noise

Our reductions from DNFs and k-juntas can be made tolerant to random noise in the original function. This is easy to see in the case of k-juntas. An oracle for f with classification noise η' is the same as an oracle for the function $(1-2\eta')f$. By repeating the reduction used for k-juntas, we get an oracle for the function $\mathbb{O}((1-2\eta')\hat{f}_s\chi_s)$. Hence we have the following theorem:

Theorem 2.7.1 *Let $\mathcal{A}$ be an algorithm that learns parities of k variables on $\{0,1\}^n$ for every noise rate $\eta < 1/2$ in randomized time $T(n,k,\frac{1}{1-2\eta})$. Then there exists an algorithm that learns k-juntas with random noise of rate η' in time $O(k2^{2k} \cdot T(n,k,\frac{2^{k-1}}{1-2\eta'}))$.*

A noisy parity of k variables is a special case of a k-junta. Thus we have reduced the noisy junta problem to a special case viz. noisy parity, at the cost of an increase in the noise level. We refer the reader to Feldman et al. [40] for making the reduction from DNFs tolerant to random noise.

2.8 Conclusions

We have shown connections between several well-studied open problems on learning under the uniform distribution. Our reductions imply that, in a sense, the class of noisy parities is the hardest concept class for this model of learning. A natural question is whether one can reduce learning noisy parities of $O(\log n)$ variables to learning DNF (or juntas). On the positive side, a non-trivial algorithm for learning parities of $O(\log n)$ variables will help make progress on a number of important questions regarding learning under the uniform distribution.

CHAPTER III

HARDNESS OF AGNOSTIC LEARNING OF HALFSPACES

3.1 Introduction

In this chapter we prove a hardness result for agnostic learning of halfspaces over $\mathbb{Q}^n$. We obtain this result by proving hardness of approximately minimizing disagreement with a halfspace. We also show nearly optimal hardness result for maximizing agreement with a halfspace. For this problem, we show that the trivial factor 2 approximation algorithm for this problem is essentially the best one can do. The proof is by reduction from the gap version of 5-regular vertex cover to an intermediate problem called Max-Lin-$\mathbb{Q}$, and then finally to the learning halfspaces problem.

The main step in showing the hardness of approximating Max-Lin-$\mathbb{Q}$ is repeated application of two operations called *tensoring* and *boosting*. We first found these operations handy in the context of equations over $\mathbb{Z}_2$ in order to show hardness for Max-Clique [70] (details follow in Chapter 4). The main technical difference in adapting this technique to work over $\mathbb{Q}$ is keeping track of "error-margins". For the reduction to halfspaces, we need to construct systems of equations where in the 'No' case, many equations are unsatisfiable by a large margin. Indeed our tensoring and boosting operations resemble taking tensor products of codes and concatenation with Hadamard codes over finite fields.

We begin by defining the Max-Lin-$\mathbb{Q}$ problem. Informally, we are given a system of equations over rationals and we are expected to find an assignment that satisfies as many equations as possible. We will show that even if a large fraction, say 99%, of the equations can be satisfied, one can not efficiently find an assignment such that more than 1% of the equations are "almost" satisfied. That is, the difference in the left hand side and right hand side of all but 1% of the equations is "large".

Definition 3.1.1 *Given a system of linear equations with rational coefficients*

$$\{a_{i0} + \sum_{j=1}^{n} a_{ij}x_j = 0\}_{i=1,2,\ldots,M}$$

as input, the objective of the Max-Lin-$\mathbb{Q}$ *problem is to find* $(x_1, x_2, \ldots, x_n) \in \mathbb{Q}^n$ *that satisfies the maximum number of equations. A system of equations is said to be a* (M, c, s, t) *Max-Lin-*$\mathbb{Q}$ *instance if the number of equations in the system is* M *and one of the following conditions holds:*

- *At least* cM *of the equations can be satisfied by some assignment, or*
- *In any assignment,*

  $$|a_{i0} + \sum_{j=1}^{n} a_{ij}x_j| < t$$

  is true for at most sM *values of* $i \in [M]$.

The goal of the Max-Lin-$\mathbb{Q}$ *problem when given such an instance is to find out which of the two cases is true. If the system of equations satisfies the first condition, we say it has* completeness c. *In the other case, we say it has* soundness s *under* tolerance t.

An instance of Max-Lin-$\mathbb{Q}$ can be specified by a matrix

$$\boldsymbol{A} = \begin{bmatrix} a_{10} & a_{11} & \ldots & a_{1n} \\ a_{20} & a_{21} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{M0} & a_{M1} & \ldots & a_{Mn} \end{bmatrix}$$

We will refer to $\boldsymbol{A}$ itself as an instance of Max-Lin-$\mathbb{Q}$. We may also use the rows of $\boldsymbol{A}$ to represent the equations in the instance. The Max-Lin-$\mathbb{Q}$ problem is to find a vector $\boldsymbol{X} = (1, x_1, x_2, \ldots, x_n)$ such that $\boldsymbol{AX}$ has as many zeros as possible. The size of a Max-Lin-$\mathbb{Q}$ instance depends on the number of equations, the number of variables and the size of the largest coefficient. But in all the operations that we define on Max-Lin-$\mathbb{Q}$, the number of equations increases the fastest. Hence, we refer to the number of equations M itself as the size of the instance.

The main steps in the reduction from vertex cover to the learning halfspaces problem are as follows:

- Obtain a (M, c, s, t) instance of Max-Lin-$\mathbb{Q}$ from the vertex cover instance for some fixed constants c, s and t.

- Convert the above instance to a $(M', 1-\epsilon, \epsilon, t')$ instance where ϵ is a very small function of M'. To achieve this, we use two operations called *tensoring* and *boosting*.

- Convert the instance of the Max-Lin-$\mathbb{Q}$ problem obtained to an instance of the halfspace problem.

Our reduction to the learning halfspaces problem can produce a point $x \in \mathbb{Q}^n$ as both positive and negative examples. But as pointed out later in Section 3.2.2.2, this also implies a hardness of approximation when such inconsistencies are not allowed.

3.2 Preliminaries

3.2.1 Approximation Algorithms and Hardness of Approximation

In a maximization (minimization) problem for a function $\texttt{Opt}$ that only takes non-negative values, we are given an an input I and the goal is to find an s that maximizes (minimizes) $\texttt{Opt}(s, I)$. We call the maximum (minimum) attainable by $\texttt{Opt}(\cdot, I)$ the optimum for I. For example, in the Max-Clique problem, given a graph as input, the goal is to find a subset of vertices that forms the largest clique in the graph.

For $\alpha > 1$, we say $\mathcal{A}$ is an α-approximation for the problem of maximizing $\texttt{Opt}$ if $\mathcal{A}$ on any input I finds an s such that $\alpha\texttt{Opt}(s, I) \geq \max_t \texttt{Opt}(t, I)$. For $\alpha > 1$, we say $\mathcal{A}$ is an α-approximation for the problem of minimizing $\texttt{Opt}$ if $\mathcal{A}$ on any input I finds an s such that $\texttt{Opt}(s, I) \leq \alpha \min_t \texttt{Opt}(t, I)$.

We say the problem of maximizing (minimizing) $\texttt{Opt}$ is hard to approximate within α assuming conjecture C if conjecture C implies the existance of a function f such that no polynomial time algorithm $\mathcal{A}$ can always tell which of the following two cases is true for its input I:

- (Yes Case:) The optimum for the input I is at least $\alpha \cdot f(|I|)$.

- (No Case:) The optimum for the input I is at most $f(|I|)$.

In this case, we may also say that the hardness factor of maximizing (minimizing) $\texttt{Opt}$ is α assuming conjecture C.

3.2.2 Optimization Versions of Learning Problems

We have already introduced the agnostic learning model in Section 2.2.1. We now define the optimization problems for halfspaces.

For a set of examples $S \subseteq X \times \{0,1\}$, we denote $S^+ = \{x \mid \langle x, 1\rangle \in S\}$ and similarly $S^- = \{x \mid \langle x, 0\rangle \in S\}$. For any function f and a set of examples S, the *agreement rate* of f with S is $\texttt{AgreeR}(f, S) = \frac{|T_f \cap S^+| + |S^- \setminus T_f|}{|S|}$, where $T_f = \{x \mid f(x) = 1\}$. We allow S, S^+ and S^- to be multisets. For a class of functions $\mathcal{C}$, let $\texttt{AgreeR}(\mathcal{C}, S) = \max_{f \in \mathcal{C}}\{\texttt{AgreeR}(f, S)\}$.

Definition 3.2.1 *For a class of functions $\mathcal{C}$ and domain X, we define the* maximum agreement *problem $\mathcal{C}$-*MA *as follows: The input is a set of examples $S \subseteq X \times \{0,1\}$. The problem is to find a function $h \in \mathcal{C}$ such that $\texttt{AgreeR}(h, S) = \texttt{AgreeR}(\mathcal{C}, S)$.*

So for $\alpha \geq 1$, an α-approximation algorithm for $\mathcal{C}$-MA is an algorithm that returns a hypothesis h such that $\alpha \cdot \texttt{AgreeR}(h, S) \geq \texttt{AgreeR}(\mathcal{C}, S)$. Similarly, an α-approximation algorithm for the *minimum disagreement* problem $\mathcal{C}$-MD is an algorithm that returns a hypothesis $h \in \mathcal{C}$ such that $1 - \texttt{AgreeR}(h, S) \leq \alpha(1 - \texttt{AgreeR}(\mathcal{C}, S))$.

Recall that a *halfspace* is a function that is equal to 1 if $\sum_{i \in [n]} w_i x_i \geq \theta$ and 0 otherwise, where $w_1, \ldots, w_n, \theta$ are real numbers. We denote the concept class of all the halfspaces over by HS. The examples provided to the learning algorithm will be from $\mathbb{Q}^n$.

3.2.2.1 Uniform Convergence

For the hardness results in this work we will deal with samples of fixed size instead of random examples generated with respect to some distribution. One can easily see that these settings are essentially equivalent. In one direction, given an agnostic learning algorithm and a sample S we can just run the algorithm on examples chosen randomly and uniformly from S, thereby obtaining a hypothesis with the disagreement rate on S equal to the error guaranteed by the agnostic learning algorithm. For the other direction, one can use uniform convergence results for agnostic learning given by Haussler [54] (based on the earlier work

in statistical learning theory). They state that for every $c \in \mathcal{C}$ and sample S of size poly(VC-dim($\mathcal{C}$), ϵ) randomly drawn with respect to a distribution $\mathscr{D}$, with high probability the true error of c will be within ϵ of the disagreement rate of c on S. Halfspaces over $\mathbb{Q}^n$ have VC dimension of at most $n+1$. Therefore to show hardness of agnostic learning of halfspaces, we will just show hardness of approximating agreement with halfspaces.

3.2.2.2 Allowing Inconsistencies in the Sample

We work with a more general form of the agnostic learning model (Definition 2.2.2) in which the examples are drawn from an arbitrary distribution over $X \times \{0,1\}$ (and not necessarily consistent with a function). That is, we allow the set of examples S to contain contradictory examples: both $\langle x, 0\rangle$ and $\langle x, 1\rangle$ can be present. We will show that this does not make the learning problem harder. Consider any $h \in \mathcal{C}$. Now suppose we delete all pairs $\langle x, 0\rangle$ and $\langle x, 1\rangle$ of inconsistent examples to get a new set of samples S'. It is obvious that the agreement rate of h with $S \setminus S'$ is $1/2$. Therefore, if h is a hypothesis such that $\alpha \cdot \texttt{AgreeR}(h, S') \geq \texttt{AgreeR}(\mathcal{C}, S')$, then

$$\begin{aligned}
\alpha \cdot \texttt{AgreeR}(h, S) &= \alpha \cdot (\gamma \texttt{AgreeR}(h, S') + (1-\gamma) \cdot \frac{1}{2}) \\
&= \gamma \cdot \alpha \texttt{AgreeR}(h, S') + \alpha \cdot (1-\gamma)/2 \\
&\geq \gamma \texttt{AgreeR}(\mathcal{C}, S') + (1-\gamma)/2 \geq \texttt{AgreeR}(\mathcal{C}, S).
\end{aligned}$$

Therefore a hardness result for approximating the maximum agreement problem with inconsistent samples also implies the same hardness result when no inconsistencies are allowed. A similar argument also holds for minimizing disagreements.

3.2.3 Expanders

Expander graphs are frequently used to get strong guarantees on successfully hitting an unknown subset of a universe when independent trails can be prohibitively expensive. We need expanders to keep the boosting operation from blowing up the size of Max-Lin-$\mathbb{Q}$ instances fast.

Definition 3.2.2 *Let G_M denote the 5-regular Gabber-Galil expander on $M = 2p^2$ vertices.*

The exact kind of expander is not particularly important for our application; we just need them to be polynomially computable.

Definition 3.2.3 *A* walk *of length σ on a graph G is an ordered sequence of vertices $(v_1, v_2, \ldots, v_\sigma)$ such that there is an edge between v_i and v_{i+1} in G for all $1 \leq i < \sigma$.*

Denote by $\boldsymbol{M}_G$ the incidence vector of a graph G. Also define an eigenvalue of a matrix $\boldsymbol{M}$ to be a real number γ such that there exists a vector $\boldsymbol{x}$ that satisfies $\boldsymbol{M}\boldsymbol{x} = \gamma\boldsymbol{x}$. It can be shown that the largest eigenvalue of $\boldsymbol{M}_G$ for a d-regular graph G is d. The second largest eigenvalue is commonly denoted by λ. For expanders, λ is upper bounded by a constant strictly smaller than d. An important consequence of this fact is following lemma stated implicitly in [84]:

Lemma 3.2.4 ([84, Section 15]) *Let λ be the second largest eigenvalue of $\boldsymbol{M}_G$ for a d-regular graph $G = (V, E)$. Let W be a subset of the vertices of G. Then the probability that a walk of length σ picked uniformly at random does not contain a vertex from $\bar{W}$ is at most*

$$(\sqrt{|W|/|V|} + \lambda/d)^\sigma.$$

Informally, the above states that once we use $O(\log M)$ random bits for our first attempt to hit $\bar{W}$ (by picking a random starting vertex for the walk), for every additional $O(1)$ bits (to pick the next $O(1)$ vertices of the walk), we will have a constant factor of hitting $\bar{W}$.

For the 5-regular Gabber-Galil graph G_M, the second eigenvalue of $\boldsymbol{M}_{G_M}$ is $\lambda_2 \leq 5 - \frac{c^2}{1024+2c^2}$ for $c = \frac{2-\sqrt{3}}{4}$ [88, Chapter 6]. Consider the (multi)graph G_M^k obtained by taking all walks of length $k+1$ in G_M and adding an edge from the starting vertex to the ending vertex of the graph. It is easy to check that this graph has degree d^k and that its transition matrix is $(\boldsymbol{M}_{G_M})^k$. This second fact implies that its second eigenvalue is λ_2^k. Using these observations, we get the following corollary to Lemma 3.2.4.

Corollary 3.2.5 *Let W be a subset of G_M containing at most a fraction $1/10$ of the vertices. Then, there exists a constant $r < 1$ such that for sufficiently large M, at most r^σ fraction of all walks of length σ in G_M are contained within W.*

Proof: Pick a constant k large enough so that $\lambda_2^k/d^k \leq (1 - \sqrt{|W|/|V|})/2$. Then apply Lemma 3.2.4 for walks of length $\sigma' = \sigma/k$ on G_M^k. To get a random walk of length σ' on G_M^k, we can take a walk of length σ on G_M and then just take every k^{th} vertex. Since a random walk of length σ' on G_M^k does not visit $\bar{W}$ with probability at most

$$(\sqrt{|W|/|V|} + (\lambda_2/d)^k)^{\sigma'} \leq \left(\frac{1 + \sqrt{|W|/|V|}}{2}\right)^{\sigma'} = 2^{-O(\sigma)},$$

a random walk of length σ on G_M only has lower chance. □

The following is true for any regular graph.

Lemma 3.2.6 *Let W denote a subset of the vertices of a regular graph G. If W contains at least $1 - \epsilon$ fraction of the vertices for some ϵ, then at most $\sigma\epsilon$ fraction of the walks of length σ contain a vertex from $\bar{W}$.*

Proof: Pick a walk uniformly at random from all possible walks of length σ on G_p The probability that the i^{th} vertex of the walk is contained in $\bar{W}$ is at most ϵ. This is because the graph is regular and hence all vertices are equally likely to be visited as the i^{th} vertex. Taking union bound over the σ possible locations for a vertex in the walk, the probability that at least one of the vertices in the walk is contained in $\bar{W}$ is at most $\sigma\epsilon$. □

3.3 A Small Hardness Factor for Max-Lin-$\mathbb{Q}$

We first state the gap version of the NP-hardness result for regular vertex cover.

Lemma 3.3.1 ([90, 10]) *There exist constants d and ζ such that given a 5-regular graph with N vertices, it is* NP*-hard to decide whether there is a vertex cover of size $\leq dN$ or every vertex cover is of size at least $(1 + \zeta)dN$.*

Arora et al. [8] give a reduction from the above gap version of vertex cover of regular graphs to Max-Lin-$\mathbb{Q}$. They show that if there is a "small" vertex cover, the reduction produces a Max-Lin-$\mathbb{Q}$ instance in which a "large" fraction of the equations can be satisfied. But when there is no small vertex cover, only a small fraction of the equations can be exactly satisfied. We show that the proof can be strengthened so that if there is no small vertex cover, only a small fraction of equations can be satisfied even within a certain tolerance.

Lemma 3.3.2 *There exists a polynomial time algorithm that when given a 5-regular graph $G = (V, E)$ with N vertices as input produces a (M, c_0, s_0, t_0) Max-Lin-$\mathbb{Q}$ instance $\boldsymbol{A}$ over N variables as output where $M = N^{O(1)}$, c_0 and s_0 are absolute constants satisfying $s_0 < c_0$, $t_0 = 1/3$ and:*

- *If G has a vertex cover of size dN, then at least c_0 fraction of the equations in $\boldsymbol{A}$ can be satisfied.*
- *If G has no vertex cover smaller than $(1+\zeta)dN$, then for any vector $\boldsymbol{X} = (1, x_1, x_2, \ldots, x_N)$, at least $(1 - s_0)$ fraction of the entries in $\boldsymbol{AX}$ have magnitude $\geq t_0$.*

Proof: The instance $\boldsymbol{A}$ contains one variable x_i for every vertex $v_i \in V$. Corresponding to every vertex, there is a constraint $x_i = 0$. Corresponding to every edge between v_i and $v_{i'}$, we add three constraints

$$-1 + x_i + x_{i'} = 0$$

$$-1 + x_i = 0$$

$$-1 + x_{i'} = 0.$$

In all, $\boldsymbol{A}$ has $N + 3m$ equations, where $m = |E| = 5N/2$. If there is a vertex cover V_0 of size dN, set $x_i = 1$ if $v_i \in V_0$ and $x_i = 0$ otherwise. This satisfies at least $(1 - d)N + 2m$ equations.

Suppose there is no vertex cover smaller than $(1 + \zeta)dN$. We will show that not too many of the $N + 3m$ equations in $\boldsymbol{A}$ can be satisfied under a tolerance of $1/3$. Under a tolerance of $1/3$, the N equations for the vertices relax to $|x_i| < 1/3$, and the equations for an edge relax to

$$|-1 + x_i + x_{i'}| < 1/3$$

$$|-1 + x_i| < 1/3$$

$$|-1 + x_{i'}| < 1/3.$$

Note that no more than two of the three inequalities for an edge can be simultaneously satisfied. We will show that given any rational assignment to the x_is, there is a $\{0, 1\}$ assignment that is just as good or better. Consider any $\boldsymbol{X} = (1, x_1, x_2, \ldots, x_N)$, where $x_i \in \mathbb{Q}$. Set $y_i = 0$ if $x_i < 1/3$ and $y_i = 1$ otherwise. It is clear that y_i satisfies the

inequality for vertex v_i if x_i does. Now suppose at least one of the three inequalities for an edge (v_i, v_i') is satisfied by the x_is. Then, either $x_i > 1/3$ or $x_{i'} > 1/3$. In this case, at least one of y_i and $y_{i'}$ is set to 1. But then two of the equalities

$$y_i + y_{i'} = 1$$

$$y_i = 1$$

$$y_{i'} = 1$$

are satisfied. Therefore, the y_i are at least as good an assignment as the x_i.

Let $\boldsymbol{Y} = (1, y_1, y_2, \ldots, y_N)$. If there is no vertex cover of size less than $(1+\zeta)dN$, $\boldsymbol{AY}$ must contain at least $(1+\zeta)dN + m$ entries that are 1. That is, $\boldsymbol{AY}$ contains at most $(1-(1+\zeta)d)N + 2m$ zeros. The claim about the soundness follows. □

3.4 Amplifying the Gap for Max-Lin-$\mathbb{Q}$

We define two operations called *tensoring* and *boosting*. Tensoring converts a $(M, 1-\epsilon, 1-\delta, t)$ Max-Lin-$\mathbb{Q}$ instance to a $(M^2, 1-\epsilon^2, 1-\delta^2, t^2)$ Max-Lin-$\mathbb{Q}$ instance. We use this to get the completeness close to 1. But as a side-effect, it also gets the soundness close to 1. We use boosting to overcome this problem. A (σ, ρ)-boosting converts a (M, c, s, t) Max-Lin-$\mathbb{Q}$ instance to a $((\rho M)^\sigma, c^\sigma, s^\sigma, t/2)$ Max-Lin-$\mathbb{Q}$ instance. We amplify the (c, s) gap for Max-Lin-$\mathbb{Q}$ in four steps:

- Obtain a $(1-\epsilon, 1-K\epsilon)$ gap for very large constant K using tensoring.
- Obtain a $(1-\epsilon_0, \epsilon_0)$ gap for a very small constant $\epsilon_0 > 0$ by using a boosting operation. This gap is sufficient to prove a $2-\epsilon$ hardness factor for HS-MA for any constant $\epsilon > 0$.
- Improve the completeness even further to $1-o(1)$ while keeping the soundness below a constant, say 1/20. This is done by alternately tensoring and boosting many times. At this stage, it is essential to use a more efficient variation of boosting called pseudo-boosting. A (σ, ρ)-pseudo-boosting converts a (M, c, s, t) Max-Lin-$\mathbb{Q}$ instance to a $(O(\rho)^\sigma M, c^\sigma, s^{\Omega(\sigma)}, t/2)$ Max-Lin-$\mathbb{Q}$ instance. Since we require $c^\sigma > s^{\Omega(\sigma)}$ for the

reduction to be meaningful, we need some minimum gap between c and s. This is guaranteed by the first two steps.

- Using one more boosting operation, decrease the soundness. This gives the $(M', 1-\epsilon, \epsilon, t')$ instance where $\epsilon = 2^{-\Omega(\sqrt{\log M'})}$ as desired.

3.4.1 Large constant gap for Max-Lin-$\mathbb{Q}$

We define the first operation called tensoring. This operation is similar to an operation defined by Dumer et al. [30] on linear codes. Informally, the tensoring of a system of equations contains one equation for the "product" of every pair of equations. In this product, we replace the occurrence of $x_{j_1}x_{j_2}$ with $x_{j_1 j_2}$ and x_j with x_{0j} respectively.

Definition 3.4.1 *The tensoring of the system of equations*

$$\{a_{i0} + \sum_{j=1}^{n} a_{ij}x_j = 0\}_{i=1,2,\ldots,M}$$

is the system

$$\{a_{i_1 0}a_{i_2 0} + a_{i_1 0}(\sum_{j_2=1}^{n} a_{i_2 j_2}x_{0 j_2}) + a_{i_2 0}(\sum_{j_1=1}^{n} a_{i_1 j_1}x_{0 j_1}) + (\sum_{j_1=1}^{n}\sum_{j_2=1}^{n} a_{i_1 j_1}a_{i_2 j_2}x_{j_1 j_2}) = 0\}_{i_1,i_2=1,\ldots,M}.$$

In the matrix representation, the tensoring of

$$\begin{bmatrix} a_{10} & a_{11} & \ldots & a_{1n} \\ a_{20} & a_{21} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{M0} & a_{M1} & \ldots & a_{Mn} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = 0$$

is the system

$$\begin{bmatrix} a_{10} & a_{11} & \ldots & a_{1n} \\ a_{20} & a_{21} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{M0} & a_{M1} & \ldots & a_{Mn} \end{bmatrix} \begin{bmatrix} 1 & x_{01} & \ldots & x_{0n} \\ x_{01} & x_{11} & \ldots & x_{1n} \\ \vdots & \vdots & & \vdots \\ x_{0n} & x_{n1} & \ldots & x_{nn} \end{bmatrix} \begin{bmatrix} a_{10} & a_{20} & \ldots & a_{M0} \\ a_{11} & a_{21} & \ldots & a_{M1} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \ldots & a_{Mn} \end{bmatrix} = 0$$

where the x_{ij}s in the second matrix are the variables in the new instance. Note that the number of equations, the number of variables and the magnitude of the largest coefficient all roughly get squared.

Lemma 3.4.2 *Let $\boldsymbol{A}$ be a (M, c, s, t) instance of Max-Lin-$\mathbb{Q}$. Let $\boldsymbol{B}$ be obtained by tensoring $\boldsymbol{A}$. Then $\boldsymbol{B}$ is a $(M^2, 1-(1-c)^2, 1-(1-s)^2, t^2)$ instance*

Proof: We first prove the claim about the soundness. Suppose that for any vector $\boldsymbol{X} = (1, x_1, x_2, \ldots, x_n)$, at least $(1-s)$ fraction of the entries in $\boldsymbol{AX}$ have magnitude greater than or equal to t. Consider any assignment to the variables $(x_{j_1 j_2})$ in $\boldsymbol{B}$. Let $\boldsymbol{X}^*$ denote the matrix

$$\begin{bmatrix} 1 & x_{01} & \ldots & x_{0n} \\ x_{01} & x_{11} & \ldots & x_{1n} \\ \vdots & \vdots & & \vdots \\ x_{0n} & x_{n1} & \ldots & x_{nn} \end{bmatrix}$$

We will show that at least $(1-s)^2 M^2$ entries in $\boldsymbol{AX}^*\boldsymbol{A}^T$ have magnitude $\geq t^2$. Let $\boldsymbol{X} = (1, x_{01}, x_{02}, \ldots, x_{0n})$. The vector $\boldsymbol{AX}$ has at least $(1-s)M$ entries with magnitude $\geq t$. Let J be the set of indices of these entries. Let $\boldsymbol{V} = (\boldsymbol{AX}^*)^T$. Note that since the first column of $\boldsymbol{X}^*$ is $\boldsymbol{X}$, $\boldsymbol{V}$ has at least $(1-s)M$ entries in the first row that have magnitude $\geq t$. Let $\boldsymbol{V}_j$ denote the j^{th} column of $\boldsymbol{V}$. Note that if $j \in J$, $\boldsymbol{AV}_j$ contains at least $(1-s)M$ entries that have magnitude $\geq t^2$. Therefore, $\boldsymbol{AX}^*\boldsymbol{A}^T = \boldsymbol{V}^T\boldsymbol{A}^T = (\boldsymbol{AV})^T$ has at least $(1-s)^2M^2$ entries with magnitude $\geq t^2$. It remains to show the claim about the completeness.

Suppose there is a vector $\boldsymbol{X} = (1, x_1, x_2, \ldots, x_n)$ such that $\boldsymbol{AX}$ has a zero in cM entries. Define $x_{0j} = x_j$ and $x_{j_1 j_2} = x_{j_1} x_{j_2}$ for $j_1 \geq 1$. This satisfies all but $(1-c)^2 M^2$ of the equations in $\boldsymbol{B}$. □

We now define an operation called boosting. Roughly speaking, we pick σ equations at a time from the Max-Lin-$\mathbb{Q}$ instance $\boldsymbol{A}$. We add ρ^σ linear combinations of these to the boosted instance $\boldsymbol{B}$. The intention is that even if one of the σ equations fails under some assignment, a lot of the ρ^σ corresponding equations in $\boldsymbol{B}$ must fail. This is accomplished by using a construction similar to Hadamard code.

Definition 3.4.3 *Let $\boldsymbol{A}$ be a Max-Lin-$\mathbb{Q}$ instance with M equations. Let ρ, σ be two arbitrary natural numbers. We define the (ρ, σ)-boosting to be the Max-Lin-$\mathbb{Q}$ instance $\boldsymbol{B}$ obtained as follows. For every possible choice $(\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2}, \ldots, \boldsymbol{A}_{i_\sigma})$ of σ rows of $\boldsymbol{A}$ and a vector $(\rho_1, \rho_2, \ldots, \rho_\sigma) \in [\rho]^\sigma$, add a row $\rho_1 \boldsymbol{A}_{i_1} + \rho_2 \boldsymbol{A}_{i_2} + \ldots + \rho_\sigma \boldsymbol{A}_{i_\sigma}$ to $\boldsymbol{B}$. We call the ρ^σ rows of $\boldsymbol{B}$ that correspond to a choice of $(\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2}, \ldots, \boldsymbol{A}_{i_\sigma})$ a cluster.*

The idea behind adding ρ^σ equations to each cluster is the following. If $b_1 \geq t$, then for any b, $\rho_1 b_1 + b$ lies in the interval $(-t/2, t/2)$ for at most one value of $\rho_1 \in [\rho]$. Similarly, for any given values of $\rho_2, \ldots, \rho_\sigma$ and $b_2, \ldots, b_\sigma$, $\sum_{i=1}^{\sigma} \rho_i b_i$, lies in the interval $(-t/2, t/2)$ for at most one value of $\rho_1 \in [\rho]$. An analogy to Hadamard codes is that if a bit in a string is 1, then half of the positions in its Hadamard code are 1.

Lemma 3.4.4 *Let $\boldsymbol{A}$ be a (M, c, s, t) Max-Lin-$\mathbb{Q}$ instance. Let $\boldsymbol{B}$ be a (ρ, σ)-boosting of $\boldsymbol{B}$. Then $\boldsymbol{B}$ is a $((\rho M)^\sigma, 1 - \sigma(1-c), s^\sigma + \rho^{-1}, t/2)$ instance.*

Proof: There are M^σ choices for $(\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2}, \ldots, \boldsymbol{A}_{i_\sigma})$ and ρ^σ choices for $(\rho_1, \rho_2, \ldots, \rho_\sigma)$. This proves the claim about the size of $\boldsymbol{B}$.

Fix an assignment that satisfies c fraction of the equations in $\boldsymbol{A}$. Let W denote the set of equations in $\boldsymbol{A}$ that are satisfied by this assignment. The probability that all of the σ equations in a random choice of $(\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2}, \ldots, \boldsymbol{A}_{i_\sigma})$ are in W is at least $c^\sigma \geq 1 - \sigma(1-c)$. When this happens, all the equations in the cluster corresponding to the choice of $(\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2}, \ldots, \boldsymbol{A}_{i_\sigma})$ are satisfied by the same assignment.

Now suppose for any $\boldsymbol{X} = (1, x_1, x_2, \ldots, x_n)$, at least sM fraction of the entries in $\boldsymbol{AX}$ have magnitude $\geq t$. Fix any assignment $\boldsymbol{X}$ to the variables in $\boldsymbol{A}$. Consider σ rows $\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2}, \ldots, \boldsymbol{A}_{i_\sigma}$ from $\boldsymbol{A}$. Now suppose $|\boldsymbol{A}_{i_1}\boldsymbol{X}| \geq t$. Let $b \in \mathbb{Q}$. Then, for at most one value of $\rho_1 \in [\rho]$, $\rho_1 \boldsymbol{A}_{i_1}\boldsymbol{X} + b$ has magnitude less than $t/2$. Therefore, for all but a $1/\rho$ fraction of $(\rho_1, \rho_2, \ldots, \rho_\sigma) \in [\rho]^\sigma$,

$$|(\rho_1 \boldsymbol{A}_{i_1} + \rho_2 \boldsymbol{A}_{i_2} + \ldots + \rho_\sigma \boldsymbol{A}_{i_\rho})\boldsymbol{X}| \geq t/2$$

If $(\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2}, \ldots, \boldsymbol{A}_{i_\sigma})$ are σ random rows of $\boldsymbol{A}$, the probability that none of $\boldsymbol{A}_{i_1}\boldsymbol{X}, \boldsymbol{A}_{i_2}\boldsymbol{X}$, $\ldots, \boldsymbol{A}_{i_k}\boldsymbol{X}$ have magnitude $\geq t$ is at most s^σ. Therefore, at most $s^\sigma + (1 - s^\sigma)\rho^{-1} \leq s^\sigma + \rho^{-1}$

fraction of the entries in $\boldsymbol{BX}$ have magnitude less than $t/2$. This proves the claim about the soundness. □

We now use tensoring and boosting to obtain a $1-\epsilon_0$ versus ϵ_0 gap for Max-Lin-$\mathbb{Q}$.

Lemma 3.4.5 *For any constants $\epsilon_0 > 0$, $0 < s < c \leq 1$ and $t > 0$, there exists a polynomial time algorithm that when given a (M, c, s, t) Max-Lin-$\mathbb{Q}$ instance $\boldsymbol{A}$ as input produces a $(M_1, 1-\epsilon_0, \epsilon_0, t_1)$ instance where $M_1 = M^{O(1)}$ and $t_1 > 0$ is a constant.*

Proof: Let $\boldsymbol{B}$ be the instance obtained by repeatedly tensoring $\boldsymbol{A}$ q times. Then, $\boldsymbol{B}$ is a $(M^Q, 1-(1-c)^Q, 1-(1-s)^Q, t^Q)$ Max-Lin-$\mathbb{Q}$ instance, where $Q = 2^q$. Choose q large enough so that

$$\left\lceil \frac{\ln(2/\epsilon_0)}{(1-s)^Q} \right\rceil \leq \frac{\epsilon_0}{(1-c)^Q}.$$

Now we use (ρ, σ)-boosting on $\boldsymbol{B}$ where $\rho = \lceil 2/\epsilon_0 \rceil$ and

$$\sigma = \left\lceil \frac{\ln(2/\epsilon_0)}{(1-s)^Q} \right\rceil.$$

The result is a (M_1, c_1, s_1, t_1) instance where

$$c_1 \geq 1 - \sigma(1-c)^Q \geq 1 - \epsilon_0$$

and

$$(1-(1-s)^Q)^\sigma \leq (1/e)^{\sigma(1-s)^Q} \leq e^{-\ln(2/\epsilon_0)} = \epsilon_0/2.$$

Therefore, $s_1 = (1-(1-s)^Q)^\sigma + \rho^{-1} \leq \epsilon_0$ and $t_1 = t^Q/2$. □

Note that combining Lemma 3.4.5 with Lemma 3.5.1 suffices to show a $2-\epsilon$ hardness factor for HS-MA for any constant $\epsilon > 0$. We now focus on obtaining an improved hardness result where ϵ is sub-constant.

3.4.2 Super-constant gap for Max-Lin-$\mathbb{Q}$

We will now prove a $(1-\epsilon, \epsilon)$ hardness for Max-Lin-$\mathbb{Q}$ for a sub-constant (as a function of the size M) value of ϵ. One hurdle to be overcome is the rapid increase in the size of the instance produced by both tensoring (from M to M^2) and boosting (from M to M^σ). To overcome this problem we now define a pseudo-random boosting, or simply *pseudo-boosting*,

that achieves a similar improvement in soundness (with a similar expense in completeness) as normal boosting does, but increases the size by only a constant factor.

Definition 3.4.6 *Let $\boldsymbol{A}$ be a Max-Lin-$\mathbb{Q}$ instance with M equations. Let ρ, σ be two arbitrary numbers. We define the (ρ, σ)-pseudo-boosting to be the Max-Lin-$\mathbb{Q}$ instance $\boldsymbol{B}$ obtained as follows. Let G_M be the 5-regular Gabber-Galil graph on M vertices. Associate every vertex v of G_M to an equation $\boldsymbol{A}_v$ in $\boldsymbol{A}$. For every possible walk $(v_1, v_2, \ldots, v_\sigma)$ of length σ on G_M and a vector $(\rho_1, \rho_2, \ldots, \rho_\sigma) \in [\rho]^\sigma$, add a row $\rho_1 \boldsymbol{A}_{v_1} + \rho_2 \boldsymbol{A}_{v_2} + \ldots + \rho_\sigma \boldsymbol{A}_{v_\sigma}$ to $\boldsymbol{B}$. We call the ρ^σ rows of $\boldsymbol{B}$ that correspond to a walk on the rows of $\boldsymbol{A}$ a cluster.*

The specific kind of expander used in pseudo-boosting is not important. We would like to point out that since Gabber-Galil graphs are defined only for integers of the form $2p^2$, we might have to add some trivially satisfied equations to $\boldsymbol{A}$. This only improves the completeness of $\boldsymbol{A}$. The soundness suffers by at most $O(1/\sqrt{M})$, which is a negligible increase if the soundness of the instance $\boldsymbol{A}$ were constant. Hence, we ignore this issue from now on.

Lemma 3.4.7 *Let $\boldsymbol{A}$ be a $(M, c, 1/10, t)$ Max-Lin-$\mathbb{Q}$ instance. Let $\boldsymbol{B}$ be a (ρ, σ)-pseudo-boosting of $\boldsymbol{B}$. Then $\boldsymbol{B}$ is a $(5^{\sigma-1}\rho^\sigma M, 1 - \sigma(1-c), r^\sigma + \rho^{-1}, t/2)$ instance, where r is the constant guaranteed by Corollary 3.2.5.*

Proof: The proof of the lemma will closely parallel that of Lemma 3.4.4. The number of walks of length σ beginning from each vertex in a graph G_M is $5^{\sigma-1}$. Corresponding to each walk, we add ρ^σ rows to $\boldsymbol{B}$. This proves the claim about the size of $\boldsymbol{B}$.

Fix an assignment that satisfies c fraction of the equations in $\boldsymbol{A}$. Let W denote the set of equations in $\boldsymbol{A}$ that are satisfied by this assignment. From Lemma 3.2.6, we know that at most $\sigma(1-c)$ fraction of walks of length σ visit a row from $\bar{W}$. If all of the σ rows visited by a walk are satisfied, then all the equations of $\boldsymbol{B}$ in the cluster corresponding to this walk are also satisfied under the same assignment.

Now suppose for any $\boldsymbol{X} = (1, x_1, x_2, \ldots, x_n)$, at least $M/10$ fraction of the entries in $\boldsymbol{AX}$ have magnitude $\geq t$. Fix any assignment $\boldsymbol{X}$ to the variables in $\boldsymbol{A}$. If $(\boldsymbol{A}_{v_1}, \boldsymbol{A}_{v_2}, \ldots, \boldsymbol{A}_{v_\sigma})$

is a random walk on G_M, then from Corollary 3.2.5, the probability that none of $\boldsymbol{A}_{v_1}\boldsymbol{X}$, $\boldsymbol{A}_{v_2}\boldsymbol{X}, \ldots, \boldsymbol{A}_{v_k}\boldsymbol{X}$ have magnitude $\geq t$ is at most r^σ for large enough M. Therefore, as in Lemma 3.4.4, at most $r^\sigma + (1 - r^\sigma)\rho^{-1} \leq r^\sigma + \rho^{-1}$ fraction of the entries in $\boldsymbol{BX}$ have magnitude less than $t/2$. This proves the claim about the soundness. □

We now use tensoring with pseudo-boosting to obtain a super-constant hardness factor for Max-Lin-$\mathbb{Q}$.

Theorem 3.4.8 *There exists a $2^{(\log N)^{O(1)}}$ time reduction that when given a 5-regular graph G on N vertices outputs a Max-Lin-$\mathbb{Q}$ instance $\boldsymbol{A}_3$ of size $M_3 = 2^{(\log N)^{O(1)}}$ such that*

- *If there is a vertex cover of size dN, then there is an assignment that satisfies $1 - 2^{-\Omega(\sqrt{\log M_3})}$ fraction of the equations.*
- *If every vertex cover is of size $\geq (1 + \zeta)dN$, then under any assignment, at most $2^{-\Omega(\sqrt{\log M_3})}$ fraction of the equations can be satisfied within a tolerance as large as $2^{-O(\sqrt{\log M_3})}$.*

where d and ζ are the constants mentioned in Lemma 3.3.1. The number of variables n in $\boldsymbol{A}_3$ is $M_3^{O(1)}$.

We first use Lemma 3.3.2 and Lemma 3.4.5 to convert a vertex cover instance to a $(M_1, 1 - \epsilon_0, \epsilon_0, t_1)$ Max-Lin-$\mathbb{Q}$ instance $\boldsymbol{A}_1$. We then alternately tensor and pseudo-boost $\boldsymbol{A}_1$ so that the soundness stays below $1/20$, but the completeness progressively comes closer to 1. As a final step, we pseudo-boost once more so that the completeness is $1 - \epsilon$ and the soundness is ϵ for a small value ϵ as desired.

Proof: Fix

$$\epsilon_0 = \min\left\{\frac{\log r^{-1}}{4\log 40}, \frac{1}{20}\right\},$$

$\sigma_0 = \lceil (4\epsilon_0)^{-1} \rceil$ and $\rho_0 = 40$.

We first use Lemma 3.3.2 and Lemma 3.4.5 to convert the graph to a $(M_1, 1-\epsilon_0, 1/20, t_1)$ Max-Lin-$\mathbb{Q}$ instance $\boldsymbol{A}_1$, where $M_1 = N^{O(1)}$. Suppose $\boldsymbol{B}_1$ is the result of tensoring and (ρ_0, σ_0)-pseudo-boosting $\boldsymbol{A}_1$ once. Then $\boldsymbol{B}_1$ is a $(O(M_1)^2, 1 - \sigma_0\epsilon_0^2, r^{\sigma_0} + \rho_0^{-1}, t_1^2/2)$ instance (The claim about soundness follows since the soundness after tensoring is $1 - (1 - 1/20)^2 \leq$

1/10 and we can apply Lemma 3.4.7 to bound the soundness after the pseudo-boosting). Since $\sigma_0\epsilon_0 \leq 1/2 < 1$, after one round of tensoring and pseudo-boosting, the completeness comes closer to 1. Also, the soundness stayed below 1/20 after one round since $r^{\sigma_0} + \rho_0^{-1} \leq 2^{\log r/(4\epsilon_0)} + 1/40 \leq 1/20$. Now, let $\boldsymbol{A}_2$ be the result of repeatedly tensoring and (ρ_0, σ_0)-pseudo-boosting $\boldsymbol{A}_1$ q_1 times. Let $Q_1 = 2^{q_1}$. Then $\boldsymbol{A}_2$ is a $(M_2, c_2, 1/20, t_2)$ instance where $M_2 = O(M_1)^{Q_1}$, $c_2 = 1 - O(1)^{Q_1}$ and $t_2 = \Omega(1)^{Q_1}$.

As a final step, we now use (ρ_2, σ_2)-pseudo-boosting on $\boldsymbol{A}_2$ where $\rho_2 = 3^{Q_1}$, $\sigma_2 = \left\lceil \frac{1+Q_1}{\log(1/r)} \right\rceil$. This produces a (M_3, c_3, s_3, t_3) instance where $M_3 = O(\rho_2)^{\sigma_2} M_2 = 2^{O({Q_1}^2)} M_1^{Q_1}$, $c_3 = 1 - O(Q_1)O(1)^{Q_1} = 1 - O(1)^{Q_1}$, $s_3 = r^{\sigma_2} + \rho_2^{-1} \leq 2^{-Q_1}$ and $t_3 = \Omega(1)^{Q_1}$. Choose the smallest q_1 so that $Q_1 \geq \log M_1$. Then $\log M_3 = O(\log^2 M_1)$, which implies $Q_1 = \Omega(\sqrt{\log M_3})$. That is, $\boldsymbol{A}_3$ is a $(M_3, 1 - 2^{-\Omega(\sqrt{\log M_3})}, 2^{-\Omega(\sqrt{\log M_3})}, 2^{-O(\sqrt{\log M_3})})$ instance.

The total number of times we tensor is $O(q_1)$ (a constant number of times in Lemma 3.4.5 and q_1 times here). Therefore the number of variables is $N^{O(2^{q_1})} = N^{O(Q_1)} = M_3^{O(1)}$, since boosting and pseudo-boosting don't increase the number of variables.

□

3.5 From Max-Lin-$\mathbb{Q}$ to HS-MA

Lemma 3.5.1 *There exists a polynomial time algorithm that when given a (M, c, s, t) instance $\boldsymbol{A}$ of Max-Lin-$\mathbb{Q}$ over n variables produces a instance of the halfspace problem with $2M$ points over $\mathbb{Q}^n$ such that:*

- *If there is a solution to the Max-Lin-$\mathbb{Q}$ instance that satisfies $\geq cM$ of the equations, there is a halfspace that correctly classifies $\geq 2cM$ of the points.*
- *If $\boldsymbol{A}$ has soundness s under tolerance t, then no halfspace can correctly classify more than $(1+s)M$ of the points.*

Proof: We can rewrite each equation of $\boldsymbol{A}$ as two inequalities

$$-t' \leq a_{i0} + \sum_{j=1}^{n} a_{ij} x_j \leq t'$$

or simply

$$\begin{aligned}(a_{i0} + t') + \textstyle\sum_{j=1}^{n} a_{ij}x_j \geq 0 \\ (a_{i0} - t') + \textstyle\sum_{j=1}^{n} a_{ij}x_j \leq 0\end{aligned} \tag{3}$$

for any $t' \in \mathbb{Q}$ satisfying $0 < t' \leq t/2 < t$. We choose a t' also satisfying $t' < \min_{i:a_{i0}\neq 0}|a_{i0}|$. This ensures that the constant term in every constraint is non-zero and if $a_{i0} \neq 0$, then both $a_{i0} + t'$ and $a_{i0} - t'$ have the same sign. We now divide by the appropriate number (and flip the direction of the inequality if necessary) to make all the constant terms equal to 1. We get $2M$ constraints of the form

$$1 + \sum_{j=1}^{n} h_{ij}x_j \geq 0 \quad \text{or} \quad 1 + \sum_{j=1}^{n} h_{ij}x_j \leq 0 \tag{4}$$

where $i \in \{1, 2, \ldots, 2M\}$. At this stage, the set of inequalities (3) and (4) are equivalent. Next we convert the $\leq$ constraints in (4) to $<$ and homogenize the resulting set of constraints to get

$$x_0 + \sum_{j=1}^{n} h_{ij}x_j \geq 0 \quad \text{or} \quad x_0 + \sum_{j=1}^{n} h_{ij}x_j < 0. \tag{5}$$

If we could satisfy cM equations in $\boldsymbol{A}$, (5) has a solution satisfying $2cM$ inequalities (with strict inequality since $t' > 0$) by setting x_0 to 1. Suppose there is a solution $(x_0, x_1, \ldots x_n)$ satisfying more than $(1 + s)M$ of the inequalities in (5). Note that if $x_0 < 0$, then since $t' > 0$, we will satisfy at most one of the two inequalities corresponding to an equation of $\boldsymbol{A}$. If $x_0 = 0$, we satisfy one constraint from the pair corresponding to row i in $\boldsymbol{A}$ if $a_{i0} \neq 0$, and maybe both constraints if $a_{i0} = 0$ (If $a_{i0} = 0$, the two corresponding inequalities in (5) are $\geq$ inequalities). But not more than sM of the a_{i0}s can be zero (otherwise more than sM equations can be satisfied by setting all variables to zero in $\boldsymbol{A}$). So we can assume $x_0 > 0$. Now we can scale the values of x_is so that x_0 is 1. Then, $(x_1, x_2, \ldots x_n)$ is a solution to $\boldsymbol{A}$ that satisfies more than s fraction of the equalities within tolerance t, a contradiction.

We now define the halfspace instance. The halfspace instance produced is over $\mathbb{Q}^n$. For an inequality of the first form in (5), add the point $(h_{i1}, h_{i2}, \ldots, h_{in})$ to S^+. For an inequality of the second form add the point $(h_{i1}, h_{i2}, \ldots, h_{in})$ to S^-. The correspondence between a constraint in (5) being satisfied by an assignment $(x_0, x_1, \ldots, x_n)$ and a point

from (S^+, S^-) being classified correctly by the hyperplane $x_0 + \sum_{j=1}^{n} x_j l_j \geq 0$ is clear. This completes the proof of the lemma. □

Proof of Theorem 1.4.4: We give a reduction from the vertex cover problem on 5-regular graphs mentioned in Lemma 3.3.1. The reduction will have running time $2^{(\log N)^{O(1)}}$ for N vertex graphs.

Let G be the input graph with N vertices. We use the reduction mentioned in Theorem 3.4.8 to produce a (M_3, c_3, s_3, t_3) Max-Lin-$\mathbb{Q}$ instance $\boldsymbol{A}$, where $c_3 = 1 - \epsilon$, $s_3 = \epsilon$, $\epsilon = 2^{-\Omega(\sqrt{\log M_3})}$ and $t = 2^{-O(\sqrt{\log M_3})} > 0$. We transform $\boldsymbol{A}_3$ to a halfspace instance (S^+, S^-) as described in Lemma 3.5.1. Note that the number of examples is $M' = 4M_3$. We now analyze the completeness and soundness.

- (Yes Case:) If there is a vertex cover of size $\leq dN$ in G, there is a halfspace that correctly classifies $c_3 = 1 - \epsilon$ fraction of the points.
- (No Case:) If there is no vertex cover of size smaller than $(1 + \zeta)dN$ in G, there is no halfspace that correctly classifies $\geq \frac{1}{2}(1 + s_3) = \frac{1}{2}(1 + \epsilon)$ fraction of the points.

Finally, observe that the number of dimensions n in the halfspace instance produced is $O(M_3) = O(M')$. □

For the HS-MA problem the gap obtained is $c_3/(\frac{1}{2}(1 + s_3)) = 2(1 - \epsilon)/(1 + \epsilon) = 2(1 - O(\epsilon)) = 2 - 2^{-\Omega(\sqrt{\log n})}$. The gap for HS-MD is $(1 - \frac{1}{2}(1 + s_3))/(1 - c_3) = (1/2 - s_3/2)/\epsilon = 2^{\Omega(\sqrt{\log n})}$.

3.6 Thresholds of Halfspaces

We show that under certain cryptographic assumptions, it is not possible to learn a circuit of depth 2 with unweighted threshold gates.

Definition 3.6.1 *An unweighted threshold gate (or majority gate) is a gate that when given $(x_1, x_2, \ldots, x_n) \in \{0, 1\}^n$ outputs 1 if more than half of its inputs are 1 and outputs 0 otherwise. Let TC_2^0 denote the class of depth two polynomial (in the number of inputs) sized circuits with majority gates.*

3.6.1 The Cryptosystem of Goldreich et al. [45]

The result relies on an assumption regarding the hardness of a certain lattice problem, which underlies the security of the Ajtai-Dwork cryptosystem [2].

Definition 3.6.2 *Given a set of vectors $a_1, a_2, \ldots, a_n \in \mathbb{R}^m$, the objective of SVP is to find the shortest non-zero vector in the set $S = \{\sum_{i=1}^{n} c_i a_i | c_1, c_2, \ldots, c_n \in \mathbb{Z}\}$. The set of points S is called the lattice generated by the basis vectors $a_1, a_2, \ldots, a_n$.*

We say a lattice is $p(n)$-unique if every vector of the lattice which is of length at most $p(n)$ times the shortest vector is an integer multiple of the shortest vector. The $p(n)$-unique SVP is the problem of finding the shortest vector in a $p(n)$-unique lattice.

Assumption 3.6.3 *There does not exist a randomized polynomial time algorithm for n^8-unique SVP.*

Under the above assumption, building on the Ajtai-Dwork cryptosystem, Goldreich, Goldwasser and Halevi (GGH) [45] construct a public-key encryption scheme and show it to be secure. In their cryptosystem, the decryption is error-free, which is convenient for proving our hardness result. The cryptosystem is specified by a triple $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$, which are the key generation, encryption and decryption algorithms respectively. We only sketch the first two, since they are not directly needed to prove our result.

The key generation algorithm $\mathcal{K}$ when given a security parameter n as input generates private-key public-key pair (u, e) as the output. Here, u is a vector picked uniformly from the unit sphere in $\mathbb{R}^n$. The coordinates of vector u and all real numbers that follow need to be specified only to n bits of precision. The encryption of a bit 0 or 1 using the public-key e is a random vector from $\mathbb{R}^n$ contained in a sphere of radius $2^{O(n \log n)}$ around the origin. Let $E_e(x)$ denote the distribution on cipher-texts when the bit x is encrypted by $\mathcal{E}$ using public-key e.

If $A, B \subseteq \mathbb{R}$, let $A + B = \{a + b | a \in A, b \in B\}$. To decrypt a cipher-text v' using private-key u, Algorithm $\mathcal{D}$ computes $u \cdot v'$.

1. If $u \cdot v' \in \mathbb{Z} + [-2/n, 2/n]$, v' is decrypted as a 0.

2. Otherwise, $u \cdot v'$ is guaranteed to lie in the set $\mathbb{Z}+1/2+[-2/n, 2/n]$ and v' is decrypted as a 1.

Goldreich et al. prove that if a randomized algorithm can differentiate between the encryption of 0 and 1 with reasonable success with knowledge of only the public key, then there a randomized polynomial time algorithm for n^8-unique SVP. Our goal is now to show that the decryption can be implemented using a depth two threshold circuits.

3.6.2 Thresholds of Halfspaces are not PAC-learnable

Lemma 3.6.4 *For any key to the GGH encryption scheme, the decryption can be done by a* TC^0_2 *circuit of size polynomial in n, the security parameter.*

Proof: Let the cipher-text be the vector $x = (x_1, x_2, \ldots, x_n)$, where each of the coordinates x_i is specified up to a precision of n bits. That is we are given bits x_{ij} where $x_i = \sum_j 2^j x_{ij}$. The decoding algorithm $\mathcal{D}$ needs to compute $u \cdot x$ and check if it is strictly within $2/n$ of an integer. The quantity $u \cdot x$ can be computed as

$$u \cdot x = \sum_i u_i x_i = \sum_i u_i \sum_j 2^j x_{ij} = \sum_{i,j} u_i 2^j x_{ij}$$

Let f_{ij} denote the fractional part of $u_i 2^j$. Then the fractional part of $u \cdot x$ is the same as that of $\sum f_{ij} x_{ij}$. Also, $a_x = \sum f_{ij} x_{ij}$ is a quantity in the range from 0 to N for some $N = O(n^2)$. Since either $a_x \in \mathbb{Z} + [-2/n, 2/n]$ or $a_x \in \mathbb{Z} + 1/2 + [-2/n, 2/n]$, we can round off all the weights f_{ij} to $4 \log n$ decimal places. This results in an error of at most $\frac{N}{n^4} = O(1/n^2)$ in calculating a_x. Therefore, x is the cipher-text of 0 if and only if a_x is within $O(1/n)$ of an integer.

We now specify how to check if the number a_x is close to an integer using a TC^0_2 circuit with $2N+3$ gates, provided $a_x \in [0, N]$. Note that for any $z \in \mathbb{Z}$ and any $a \in \mathbb{R}$, at least one of the two constraints $a < z+1/8$ and $a > z-1/8$ is satisfied. Both the constraints are met if and only if $a \in (z-1/8, z+1/8)$. Therefore, for any $a \in [0, N]$, at least $N+1$ of the $2N+2$ constraints $a < z+1/8$ and $a > z-1/8$ are satisfied, where $z \in \{0, 1, 2, \ldots, N\}$. If a is within $1/8$ of some integer, then exactly $N+2$ of the constraints are satisfied. Otherwise exactly $N+1$ of the constraints are satisfied. Each of the constraints $\sum f_{ij} x_{ij} < z + 1/8$ (and by

a similar argument, the constraints $\sum f_{ij}x_{ij} > z - 1/8)$ can be checked by a majority gate. This is because $2^{4\log n} f_{ij}$ is an integer in the range 0 through n^4. Consider a polynomial sized list in which we add $n^4 f_{ij}$ copies of the variable x_{ij}. We want to check if less than $n^4(z + 1/8)$ entries in the list are 1. This can be done using a majority gate. Checking whether $N + 1$ or $N + 2$ of these $2N + 2$ gates evaluate to true can be done using another majority gate. □

We use the notation of Kearns and Valiant [66] to define our learning problem for depth 2 threshold circuits.

Definition 3.6.5 *We say the concept class $\mathcal{C}$ is weakly PAC learnable if there exists a randomized polynomial time algorithm $\mathcal{A}$ and a polynomially evaluatable hypothesis class $\mathcal{H}$ with the following properties:*

- *(Input:) The algorithm is given access to oracles POS and NEG that generate points from some distributions $\mathscr{D}_c^+$ and $\mathscr{D}_c^-$ over the positive and negative examples respectively of a concept c.*
- *(Output:) With probability at least $1/2$, the algorithm outputs a hypothesis $h \in \mathcal{H}$ such that*

$$\Pr[h(x) = 1 \text{ when } c(x) = 1] \geq \frac{1}{2} + \frac{1}{p(n)}$$
$$\Pr[h(x) = 0 \text{ when } c(x) = 0] \geq \frac{1}{2} + \frac{1}{p(n)}$$

for some polynomial $p(n)$.

Proof of Theorem 1.4.5: Suppose there exists a weakly PAC learning algorithm $\mathcal{A}$ for the class TC^0_2. We construct an algorithm $\mathcal{D}'$ for decoding a bit encrypted using the scheme of Goldreich et al. [45] with a reasonable success probability.

Let e be the public-key used generated by the key-generation algorithm $\mathcal{K}$. This key specifies a distribution $\mathscr{D}_e(0)$ and $\mathscr{D}_e(1)$ on the cipher-texts of 0 and 1 (the positive and negative examples) respectively. It is easy to sample from these distributions since the public-key and the encryption algorithm are known. We also know from Lemma 3.6.4 that there is a TC^0_2 circuit that correctly identifies if its input is the encryption of a 0 or 1.

Algorithm $\mathcal{D}'$ uses Algorithm $\mathcal{A}$ to learn this circuit. Suppose that with probability $\geq 1/2$, Algorithm $\mathcal{A}$ outputs a hypothesis h such that

$$\Pr[h(x) = 0 | x \in_R \mathscr{D}_e(0)] - \Pr[h(x) = 0 | x \in_R \mathscr{D}_e(1)] \geq \epsilon(n)$$

for some $\epsilon(n)$ that is inverse polynomial in n, where $x \in_R \mathscr{D}$ is used to denote that x is a random element from the distribution $\mathscr{D}$. Then, $\mathcal{D}'$ when given a string that is the random encryption of a 0 or a 1 can use h to decode the input with reasonable success:

$$\begin{aligned} &\Pr[\mathcal{D}\text{' outputs 0 when given random encryption of 0}] \\ &\quad - \Pr[\mathcal{D}\text{' outputs 0 when given random encryption of 1}] \\ &\geq \epsilon(n) \end{aligned}$$

This contradicts the security assumption of the GGH encryption scheme. □

3.6.3 Further Hardness of Learning Halfspaces with Adversarial Noise

Our result on hardness of learning TC^0_2 circuits also implies hardness of learning halfspaces with adversarial noise of high rate even when the learning algorithm is allowed to output any circuit. The proof is immediate from the "discriminator lemma" due to Hajnal et al. [49].

Lemma 3.6.6 ([49]) *For any Boolean functions $g_1, \ldots, g_k$ on X, $f = MAJ(g_1, g_2, \ldots, g_k)$ and any distribution $\mathscr{D}$ on X there exists $i \leq k$ such that $|\Pr_{\mathscr{D}}[f = g_i] - \frac{1}{2}| \geq \frac{1}{2k}$.*

If it holds that $\Pr_{\mathscr{D}}[f = g_i] \geq \frac{1}{2} + \frac{1}{2k}$, then examples of f drawn from distribution $\mathscr{D}$ can be seen as examples of g_i with adversarial noise of rate $\frac{1}{2} - \frac{1}{2k}$. Similarly, if $\Pr_{\mathscr{D}}[f = g_i] \leq \frac{1}{2} - \frac{1}{2k}$, then the examples are equivalent to examples of $\neg g_i$ with adversarial noise of rate $\frac{1}{2} - \frac{1}{2k}$. The negation of a halfspace is a halfspace (in fact the negation of a majority is a majority of negated variables). Thus the discriminator lemma implies that a TC^0_2 circuit is equivalent to a halfspace with adversarial noise. Hence Theorem 1.4.5 implies Theorem 1.4.6.

3.7 Conclusions

Our hardness results essentially resolve the status of proper agnostic learning of halfspaces. A challenging open problem is to allow more general hypothesis classes. A recent result of Khot and Saket [72] shows such a result for intersection of halfspaces. They show that for any number l, it is not possible to even weakly PAC-learn intersection of two halfspaces using any function of l linear thresholds. We took a step in this direction for halfspaces and showed that at least when the noise rate is high, weak non-proper learning of halfspaces is hard.

CHAPTER IV

BETTER INAPPROXIMABILITY RESULTS FOR MAX-CLIQUE, CHROMATIC NUMBER AND MIN-3LIN-DELETION

4.1 Introduction

4.1.1 Max-Clique

As mentioned before, the best approximation algorithm for Max-Clique is an $O(\frac{n(\log\log n)^2}{\log^3 n})$-approximation algorithm by Feige [36]. It was conjectured that the Lovász θ-function might be an $O(\sqrt{n})$ approximation for Max-Clique (see [78] for details). Since the Lovász θ-function can be computed to any desired degree of accuracy in polynomial time, the conjecture implies an $O(\sqrt{n})$ approximation algorithm for Max-Clique. For "perfect" graphs, Lovász θ-function equals the size of the largest clique. For random graphs, the gap between these two values can be as bad as $\Omega(\sqrt{n}/\log n)$. The conjecture says that this may essentially be the worst possible gap. Feige [32] disproved the conjecture by showing that the Lovász θ-function does not approximate Max-Clique better than $\frac{n}{2^{\sqrt{c\log n}}}$, where $c > 0$ is a constant.

4.1.1.1 Previous Results for Hardness of Approximating Max-Clique

The first inapproximability result for Max-Clique was obtained by Feige et al. [33] who discovered the connection between hardness of approximation and Probabilistically Checkable Proofs(PCPs). We summarize the progress on showing hardness results for Max-Clique in Table 4.1.1.1. Let $\mathrm{PCP}_{c,s}(r(n), q(n))$ denote the class of languages that have a non-adaptive verifier with the following properties. For an input string of length n, the verifier uses $r(n)$ random bits and queries $q(n)$ bits from the proof. If the input belongs to the language, there is a correct proof that is accepted with probability c. Otherwise, no proof is accepted with probability more than s. Feige et al. [33] showed that $\mathrm{NP} \subseteq \mathrm{PCP}_{1,1/2}(O(\log n \log\log n), O(\log n \log\log n))$. Arora and Safra [11] and Arora et al. [10] improved this result to show that $\mathrm{NP} \subseteq \mathrm{PCP}_{1,1/2}(O(\log n), O(1))$, a result known as

Table 1: Hardness Results for Max-Clique.

	Hardness Factor	Assumption
Feige et al. [33]	$2^{\log^{1-\epsilon} n}$, for any $\epsilon > 0$	NP $\not\subseteq$ DTIME($2^{(\log n)^{O(1)}}$)
Arora and Safra [11]	$2^{(\log n)^{1/2-\epsilon}}$	P $\neq$ NP
Arora et al. [10]	n^c, for some $c > 0$	P $\neq$ NP
Bellare et al. [13]	$n^{1/30}$	NP $\not\subseteq$ BPP
Bellare et al. [13]	$n^{1/25}$	NEXP $\not\subseteq$ BPEXP
Feige and Kilian [34]	$n^{1/15}$	NP $\not\subseteq$ coRP
Bellare and Sudan [14]	$n^{1/4-\epsilon}$	NP $\not\subseteq$ ZPP
Bellare et al. [12]	$n^{1/3-\epsilon}$	NP $\not\subseteq$ ZPP
Håstad [51]	$n^{1/2-\epsilon}$	NP $\not\subseteq$ coRP
Håstad [50]	$n^{1-\epsilon}$	NP $\not\subseteq$ ZPP
Engebretsen and Holmerin [31]	$\frac{n}{2^{O(\log n/\sqrt{\log\log n})}}$	NP $\not\subseteq$ ZPTIME($2^{O(\log n(\log\log n)^{3/2})}$)
Khot [69]	$\frac{n}{2^{(\log n)^{1-\gamma'}}}$, for some $\gamma' > 0$	NP $\not\subseteq$ ZPTIME($2^{(\log n)^{O(1)}}$)

the *PCP Theorem*. Since then, many different PCP constructions for languages in NP have led to inapproximability results for several other problems in addition to Max-Clique.

Bellare and Sudan [14] defined a parameter called *amortized free bits* for PCPs. They showed that if problems in NP have PCPs that use logarithmic randomness and $\bar{f}$ amortized free bits, then Max-Clique is hard to approximate within a factor of $n^{1/(1+\bar{f})-\epsilon}$ unless NP $\subseteq$ ZPP. They constructed PCPs with $3 + \delta$ amortized free bits for arbitrarily small $\delta > 0$. This implies a hardness factor of $n^{1/4-\epsilon}$ for Max-Clique. The result was improved by Bellare et al. [12] by constructing PCPs with $2 + \delta$ amortized free bits. Finally, Håstad [50] gave a construction that achieved an amortized free bit complexity of δ for any constant $\delta > 0$, proving $n^{1-\epsilon}$ hardness for Max-Clique. Simpler proofs of Håstad's result were given by Samorodnitsky and Trevisan [97] and Håstad and Wigderson [53]. Both these results achieved amortized free bit complexity δ and *amortized query complexity* $1 + \delta$ for any constant $\delta > 0$ (both parameters are optimal).

Khot [69] showed that Max-Clique cannot be approximated within a factor of $\frac{n}{2^{(\log n)^{1-\gamma'}}}$ for some small constant $\gamma' > 0$, assuming NP $\not\subseteq$ ZPTIME($2^{(\log n)^{O(1)}}$).

4.1.1.2 Significance of Improved Hardness Results for Max-Clique

We consider it to be a significant problem to prove a hardness factor of $\frac{n}{2^{O(\sqrt{\log n})}}$ for Max-Clique. As mentioned before, Feige [32] showed that the Lovász θ-function can have an approximation ratio as bad as $\frac{n}{2^{c\sqrt{\log n}}}$ for some constant $c > 0$. It would be interesting to prove the same lower bound for *any* polynomial time algorithm. It would also fit in nicely with Trevisan's [103] lower bound of $\frac{d}{2^{O(\sqrt{\log d})}}$ for Max-Clique on degree d graphs (d thought of as a large constant). Blum [24] showed that if there exists a factor $\frac{n}{2^{\sqrt{b \log n}}}$ quasi-polynomial time approximation algorithm for Max-Clique, then there exists a quasi-polynomial time algorithm to color a 3-colorable graph with n^{ϵ} colors, where $\epsilon = O(1/b)$. Therefore, strong lower bounds for Max-Clique give evidence that the graph coloring problem is hard. Another motivation comes from a result of Feige and Kogan [37] who showed that if the *balanced bipartite clique problem* can be approximated within a constant factor, then there is a $\frac{n}{2^{O(\sqrt{\log n})}}$ approximation for Max-Clique. We refer to Srinivasan's paper [100] for several other interesting consequences of proving strong hardness results for Max-Clique.

4.1.2 Chromatic Number

Feige and Kilian [35] showed the connection between *randomized PCPs* and inapproximability of chromatic number. Using this result, they prove that it is hard to approximate chromatic number within a factor better than $n^{1-\epsilon}$ for any constant $\epsilon > 0$, assuming NP $\not\subseteq$ ZPP. Khot [69] constructs a more efficient verifier and obtains a hardness factor of $\frac{n}{2^{(\log n)^{1-\gamma'}}}$ for some constant $\gamma' > 0$, assuming NP $\not\subseteq$ ZPTIME($2^{(\log n)^{O(1)}}$). We would like to emphasize that the constant γ' in Khot's hardness results for Max-Clique and chromatic number is a non-explicit (possibly extremely tiny) constant that depends on the proof of Raz's Parallel Repetition Theorem [92].

4.2 Our Techniques

Our main step to prove the improved inapproximability result for Max-Clique and chromatic number is to first to show an improved inapproximability result for Min-3Lin-Deletion.

Definition 4.2.1 *Let $\oplus$ denote addition modulo 2. Given a system of linear equations*

modulo 2

$$\{a_{i0} \oplus (\bigoplus_{j=1}^{l} a_{ij}x_j) = 0\}_{i=1,2,\ldots,M}$$

as an input, Min-Lin-Deletion *is the problem of finding the minimum number of equations that need to be deleted so that the remaining system of equations has a satisfying assignment.* Min-3Lin-Deletion *is the special case where exactly three of the coefficients $a_{i1}, a_{i2}, \ldots, a_{il}$ are non-zero for all i (i.e., each equation is over exactly 3 variables). An instance of the Min-Lin-Deletion problem can be specified by a $(M, l+1)$ matrix*

$$\boldsymbol{A} = \begin{bmatrix} a_{10} & a_{11} & \ldots & a_{1l} \\ a_{20} & a_{21} & \ldots & a_{2l} \\ \vdots & \vdots & & \vdots \\ a_{M0} & a_{M1} & \ldots & a_{Ml} \end{bmatrix}$$

In this case, we say that $\boldsymbol{A}$ is a (M, l)-Min-Lin-Deletion instance. We refer to the minimum fraction of equations that need to be deleted to find a satisfying assignment as the optimum of $\boldsymbol{A}$, denoted by $Opt(\boldsymbol{A})$. That is, $Opt(\boldsymbol{A})$ is the minimum possible fraction of 1s in $\boldsymbol{AX}$, where the minimum is taken over all vectors $\boldsymbol{X} = (1, x_1, \ldots, x_l)$.

Min-Lin-Deletion-(c, s) is the problem of deciding whether the optimum of the input is at most c or at least s (we let c and s depend on the size of the input). The parameters c and s are called the completeness *and* soundness *of this problem.*

*We say a Min-Lin-Deletion instance is k-*restricted *if each equation is over at most k variables. We say a Min-Lin-Deletion instance is k-*regular *if every variable appears in exactly k equations.*

All instances of Min-Lin-Deletion considered in this chapter have the property that the maximum number of variables in an equation is at most the number of linear equations in that instance. Therefore, for simplicity, we assume that the size of a Min-Lin-Deletion instance is the number of equations in it.

4.2.1 Reduction from Min-3Lin-Deletion to Max-Clique

We briefly explain here how the improved hardness result for the Min-3Lin-Deletion problem leads to improved hardness results for Max-Clique (and similarly for chromatic number).

Khot's [69] reduction from Min-3Lin-Deletion to Max-Clique proceeds in two steps. (see Section 4.3.1 and Section 4.3.2 for a detailed description of these steps). First, the Min-3Lin-Deletion instance is reduced to the so-called *Raz Verifier* and a PCP is built on top of the Raz Verifier. Then, the hardness result for Max-Clique follows from the PCP construction using known techniques (see Lemma 4.3.5).

The strength of the hardness result for Max-Clique depends directly on the strength of the Raz Verifier. To be precise, one would like to have a Raz Verifier with as low soundness as possible, without losing *much* in completeness. Khot [69] starts with a size N instance of Min-3Lin-Deletion-$((\log N)^{-\beta}, 0.4)$ which is shown to be hard by Håstad [52]. The Raz Verifier is obtained via Parallel Repetition of a certain protocol constructed from the Min-3Lin-Deletion instance. If u is the number of repetitions, then the soundness of the Raz Verifier is $2^{-\Omega(u)}$. Thus the soundness can be lowered by taking u large enough. However, the completeness of the Raz Verifier suffers with parallel repetition. The completeness of the Min-3Lin-Deletion instance is $(\log N)^{-\beta}$ and this limits u to be at most $(\log N)^{\beta}$. Note that $\beta > 0$ is a tiny constant.

On the other hand, we start with the Min-3Lin-Deletion-$(2^{-\Omega(\sqrt{\log N})}, \Omega(\log^{-3} N))$ instance given by Theorem 1.4.10. The completeness is good enough so that we may take up to $u = 2^{\Omega(\sqrt{\log N})}$ repetitions (we however take much fewer repetitions since we do not want to blow up the size of the Raz Verifier). For some fixed constant c_0, the soundness of the Raz Verifier is $(1-(1/\log^3 N)^{c_0})^u$, which is roughly $2^{-u/\log^{3c_0} N}$. We pick $u = (\log N)^{K+3c_0}$ for a large constant K and achieve a Raz Verifier with much lower soundness than earlier.

4.2.2 Overview of Our Construction

The main steps involved in showing inapproximability of Min-3Lin-Deletion are shown in Figure 1. We start with the Min-3Lin-Deletion-$(10^{-10}, 0.4)$ problem shown to be NP-hard by Håstad [52]. We repeatedly perform two operations called *tensoring* and *boosting* on this problem. This gives a reduction to a version of Min-Lin-Deletion that has a big gap between completeness and soundness. But the instances of Min-Lin-Deletion produced by the reduction can have equations with large number of variables. We first reduce the number

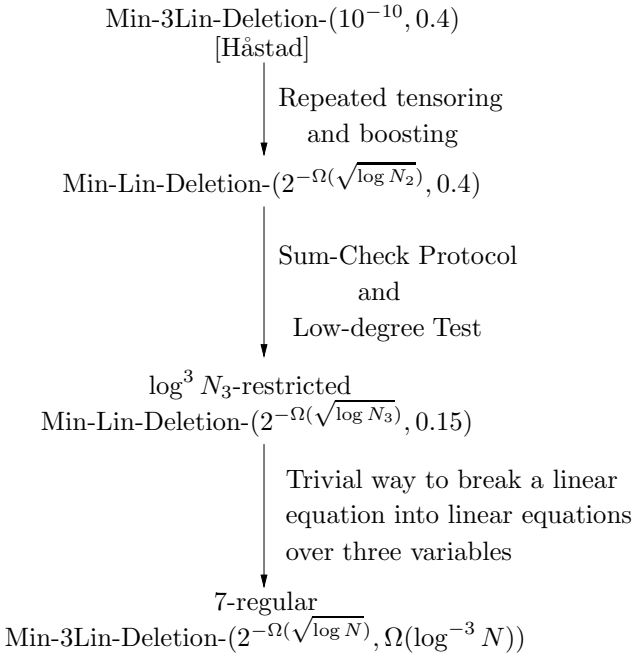

Figure 1: The main steps in proving an improved hardness factor for Min-3Lin-Deletion.

of variables appearing in an equation significantly by using the Sum-Check protocol and the Low-degree Test. We then break each of the linear equations into equations over at most three variables in a trivial way by introducing auxiliary variables. We now describe these steps in more detail and explain the new ideas involved.

4.2.2.1 Hardness of Approximation Result for Min-Lin-Deletion

We define two operations called *tensoring* and *boosting* on Min-Lin-Deletion. These are similar to the operations defined on Max-Lin-$\mathbb{Q}$ in Chapter 3. Tensoring involves taking all possible pairs of linear equations, computing their "product", and then replacing the terms of the form $x_i x_j$ with x_{ij} and x_i with x_{0i} to get back a linear equation. This converts a Min-Lin-Deletion-(c, s) instance to a Min-Lin-Deletion-(c^2, s^2) instance. Our aim is to bring the completeness close to zero, while keeping the soundness close to $1/2$. Therefore, we cannot use tensoring repeatedly by itself (otherwise the soundness would also tend to zero). We use a boosting step after every tensoring operation to work around this problem. Given a Min-Lin-Deletion instance, boosting produces a new Min-Lin-Deletion instance by picking $O(1)$ equations from its input and adding all possible linear combinations of these

equations to the output instance. The idea is that even if one the $O(1)$ equations are not satisfied by some assignment, half of the linear combinations of these equations will also not be satisfied by the assignment. To keep the size of the output Min-Lin-Deletion instance small, we pseudo-randomly generate only $O(n)$ of the $n^{O(1)}$ possible ways to pick $O(1)$ equations from n equations. In this sense, this is similar to pseudo-boosting for Max-Lin-$\mathbb{Q}$. The reason we don't need a separate kind of boosting here to get the initial gap is that Lemma 1.4.9 already lets us choose any initial gap we want for Min-Lin-Deletion. When given a Min-Lin-Deletion-$(c, 0.16)$ instance as an input, boosting produces a Min-Lin-Deletion-$(\sigma c, 0.4)$ instance as the output, for some absolute constant σ. Here, σ is the length of a random walk we need to perform on the expander so that the probability of visiting a subset containing 0.16 fraction of the vertices is at least 0.8.

After applying tensoring and boosting once to a Min-Lin-Deletion-$(c, 0.4)$ instance, we get a Min-Lin-Deletion-$(\sigma c^2, 0.4)$. If we start with the completeness $c = 1/(2\sigma)$, we could apply tensoring and boosting repeatedly. The completeness will decrease each time while the soundness stays at 0.4.

Both tensoring and boosting increase the number of variables appearing in each equation. As a result, even though we start with a Min-3Lin-Deletion instance, the final instance of Min-Lin-Deletion has a large number of variables appearing in an equation. To obtain the inapproximability result for Min-3Lin-Deletion, we cannot simply break the equations into smaller equations with at most three variables in the trivial way by introducing auxiliary variables. The reason is that the gap between the completeness and soundness of the Min-Lin-Deletion problem will then become insignificant. We instead use a technique based on the Sum-Check Protocol.

4.2.2.2 Reducing the Size of Equations in a Min-Lin-Deletion Instance

We use the Sum-Check Protocol combined with the Low-degree Test (see Arora [7]) to construct a PCP verifier for Min-Lin-Deletion. A typical constraint of the Min-Lin-Deletion instance looks like:

$$x_1 \oplus x_2 \oplus \ldots \oplus x_n = a \tag{6}$$

The verifier tries to verify that this constraint is satisfied using $\log^{O(1)} n$ queries (with access to auxiliary tables as explained below). The test of the verifier is a linear predicate in the $\log^{O(1)} n$ bits read. This gives a reduction to the Min-Lin-Deletion problem in which every equation is over at most $\log^{O(1)} n$ variables. We can then break the equations into smaller equations in the trivial way.

Let $\mathbb{F}$ be a field. Given a r-variate degree d polynomial f and $a \in \mathbb{F}$, the Sum-Check Protocol can be used to verify if the sum of the values of f on the sub-hypercube S^r of $\mathbb{F}^r$ is equal to a, without having to read the value of f at all points on S^r. The prover needs to provide some auxiliary data in the form of a *Partial Sums Table*. The non-adaptive verifier under this protocol randomly reads a few values from the Partial Sums Table and uses the value of f at one point in $\mathbb{F}^r$ and accepts or rejects the proof based on these values. We use this protocol to check if a constraint of the input Min-Lin-Deletion instance (such as (6)) is satisfied. We fix a field $\mathbb{F}$ of characteristic 2 and associate the values of the variables to points in S^r, for some appropriately chosen values of the parameters $|\mathbb{F}|$, $|S|$ and r. There exists a polynomial f of "low" degree that takes these values on S^r. Checking if a linear equation is satisfied is then basically the task of checking if the sum of the values of f on the points in S^r (weighted by the coefficients of the variables in the equation) is a certain target value. We use the Sum-Check Protocol for this purpose. The polynomial f is not known to the verifier (since the values of the variables are not known to the verifier). Hence we expect the prover to also provide a *Points Table*, a table with the value of f at all the points in $\mathbb{F}^r$. For the protocol to work, we need to make sure that the Points Table is in fact "close" to a low degree polynomial. We use the Low-degree test for this purpose.

The Low-degree test expects a *Lines Table* as an auxiliary input. The Lines Table is supposed to contain the restriction of f to every line in $\mathbb{F}^r$. The test picks a random point and a random line through the point. It then checks that the value of the point as reported in the Points Table and the value of the line as reported in the Lines Table are consistent. All the tests performed by the Sum-Check protocol and the Low-degree Test are linear in the field elements. Since the field $\mathbb{F}$ has characteristic 2, these can be replaced by linear tests over Boolean values if the field elements are encoded as appropriate bit strings. The

number of queries is $\log^{O(1)} n$ and hence we get linear constraints with $\log^{O(1)} n$ size.

Layout of the Chapter

We provide an overview of the construction of Khot's [69] PCP for Min-3Lin-Deletion in Section 4.3.1. We explain how this implies inapproximability of Max-Clique using a result from Zuckerman [108] and Bellare et al. [12] in Section 4.3.2. We mention the tools from Feige and Kilian [35] and Khot [69] required to show our inapproximability result for chromatic number in Section 4.3.3.

Section 4.4 shows the gap amplifying reduction from Min-3Lin-Deletion to Min-Lin-Deletion. We use this to show a improved hardness factor for Min-3Lin-Deletion in Section 4.5. We combine these results in Section 4.6 and Section 4.7 to show an improved hardness for Max-Clique and chromatic number respectively.

4.3 Preliminaries

4.3.1 PCP Verifier of Khot [69]

In this section, we explain the main steps involved in the construction of Khot's PCP verifier. As with other PCP based constructions, the first step is the construction of a *Raz Verifier* (also known as the *outer verifier*) based on a *2-Prover 1-Round game*. Khot [69] uses a Raz Verifier for the gap version of Min-3Lin-Deletion shown to be hard by Håstad [52]. The Raz Verifier reads a constant number of locations from the proof which is over a large alphabet. The PCP verifier is obtained from this verifier by encoding the proof using Hadamard Codes.

4.3.1.1 The Raz Verifier

We first define a 2-Prover 1-Round game based on the 7-regular Min-3Lin-Deletion problem. The game consists of a verifier and two provers. The aim of the provers is to convince the verifier that there exists an assignment to the variables in the Min-3Lin-Deletion instance $\boldsymbol{A}$ which satisfies almost all equations. The verifier picks a random variable x and a random equation that contains the variable x. The verifier then passes the variable and the equation to the first and the second prover respectively. The first prover returns an assignment to

x and the second prover returns a assignment to the three variables in the equation. The verifier accepts if the assignments to x returned by both the provers are equal and the equation is satisfied by the values returned by the second prover. It is easy to check that if $Opt(\boldsymbol{A}) \leq \epsilon$, then there exists a proof that is accepted with probability at least $1 - \epsilon$. If $Opt(\boldsymbol{A}) \geq s$, then the verifier rejects any proof with probability at least $s/3$.

The Raz Verifier is obtained by repeating this protocol u times in parallel. It picks u variables $U = (x_1, x_2, \ldots x_u)$ independently at random and u equations $W = (C_1, C_2, \ldots C_u)$, where each equation C_i is a random equation containing the variable x_i. The proof is supposed to consist of two parts: one part consisting of a u-bit string $P(U)$ corresponding to every u-tuple U of the variables, and another part consisting of a $3u$-bit string $Q(W)$ corresponding to every u-tuple W of the equations. The string $P(U)$ is interpreted as an assignment to the u variables in U and the string $Q(W)$ is interpreted as an assignment to the $3u$ variables in the u equations of W. In any honest proof, $P(U)$ will be the projection of u coordinates from $Q(W)$. Define $\pi_{U,W} : \mathbb{F}_2^{3u} \rightarrow \mathbb{F}_2^u$ to be the projection that maps an assignment to the variables in W to an assignment to the variables in U. The verifier performs two tests with the strings $P(U)$ and $Q(W)$. It accepts if $\pi_{U,W}(Q(W)) = P(U)$ and $Q(W)$ satisfies the u equations in W.

It follows that if $Opt(\boldsymbol{A}) \leq \epsilon$, then there exists a proof that is accepted with probability at least $(1 - \epsilon)^u \geq 1 - \epsilon u$. To upper-bound the acceptance probability of the verifier, we need the following version of the Parallel Repetition Theorem which is implicit in Raz [92] (see remark about Theorem 1.1 in [92]).

Lemma 4.3.1 *If no proof for the 2-Prover 1-Round game is is accepted with probability greater than $1 - s$, then the Raz Verifier does not accept any proof with probability greater than $(1 - s^{c_0})^u$ for some absolute constant c_0.*

4.3.1.2 The Hadamard Code Based Verifier

Based on the Raz Verifier defined above, Khot [69] constructs a Hadamard code based verifier $\mathcal{V}_{lin}$ for Min-3Lin-Deletion which is more efficient and easier to analyze than Long code based verifiers for 3SAT. We next define Hadamard codes. Recall χ_α defined in Section

2.2.2.

Definition 4.3.2 *Let $\mathbb{F}_2 = \{0, 1\}$ denote the field over two elements.*

The Hadamard code of a u-bit string $a \in \mathbb{F}_2^u$, denoted by $Hadamard(a)$, is the string of length 2^u given by $(\chi_\alpha(a))_{\alpha \in \mathbb{F}_2^u}$.

The construction and analysis of the verifier $\mathcal{V}_{lin}$ are based on the verifiers of Sudan and Trevisan [101] and Samorodnitsky and Trevisan [97]. The queries of the verifier are chosen so as to minimize a parameter known as the amortized free bit complexity.

Definition 4.3.3 *A PCP verifier is said to use f* free bits *if there exists a set of f bits from its queries such that for any possible answer for these queries, there is a unique answer to the remaining queries that makes the verifier accept. The* amortized free bit *complexity $\bar{f}$ is defined to be $\frac{f}{\log(c/s)}$, where c and s are the completeness and soundness of the verifier.*

Both the tests of the outer verifier, checking if $P(U)$ and $Q(W)$ are consistent and checking if $Q(W)$ satisfies the u equations corresponding to W, have to be performed by reading a few bits from the proof. Instead of the values of the strings $P(U)$ and $Q(W)$, the verifier $\mathcal{V}_{lin}$ expects the proof to contain Hadamard codes of $P(U)$ and $Q(W)$. The second test is performed implicitly by a technique known as *folding*.

Folding: Let $W = (C_1, C_2, \ldots C_u)$ be a list of u linear constraints. The $3u$-bit string $y = Q(W)$ provided in the proof is supposed to satisfy the u constraints C_i. That is, the Raz verifier checks if the string y satisfies $h_i y = \zeta_i$ for $i = 1, 2, \ldots, u$. Let B be the Hadamard code of y. Let $h = \oplus \rho_i h_i$ for any $\rho_i \in \{0, 1\}$. That is, h is a vector in the linear span H of the vectors $\{h_i\}_{i=1,2,\ldots,u}$. Then,

$$B(b \oplus h) = (-1)^{y(b \oplus h)} = (-1)^{yb}(-1)^{\sum_i \rho_i y h_i} = B(b)(-1)^{\sum_i \rho_i \zeta_i}$$

Since the above statement is true as long as B is a valid Hadamard code, instead of reading $B(b \oplus h)$ from the proof, the verifier $\mathcal{V}_{lin}$ can read $B(b)$ and compute $B(b \oplus h)$ from it. The verifier identifies one element (say, the lexicographically smallest element) from each of the set-theoretic cosets of H in $\mathbb{F}_2^{3u}$ as the representative of the coset. The prover is expected

to provide the value of B only at these points. We call this the folding of B with respect to the constraints $W = (C_1, C_2, \ldots C_u)$. If b is a representative of a coset, then the value of B at some other point $b \oplus h$ in the coset is computed using the identity above.

We now mention an important consequence of folding of Hadamard codes shown by Khot [69]. Recall from Section 2.2.2 that Parseval's identity implies that for any function $A : \mathbb{F}_2^u \to \{-1, +1\}$, $\sum_{\alpha \in \mathbb{F}_2^u} \hat{A}_\alpha^2 = 1$. Khot [69] shows that if B is folded with respect to W, then $\hat{B}_\beta \neq 0$ for some $\beta \in \mathbb{F}_2^{3u}$ only if β satisfies the u equations corresponding to W, i.e., $h_i\beta = \zeta_i$ for all $i = 1, 2, \ldots, u$. This property is used to "decode" a "good" proof for the verifier $\mathcal{V}_{lin}$ to obtain a "good" proof for the Raz Verifier. Given a purported Hadamard code A (or B), it is decoded to the string α (or β) with probability $\hat{A}_\alpha$ (or $\hat{B}_\beta$, respectively). This defines a natural way to convert a proof for $\mathcal{V}_{lin}$ to a proof for the Raz verifier. A consequence of folding is that such a proof will always succeed the second test of the Raz Verifier. That is, the label $Q(W)$ will always satisfy the u equations in W.

The Test of the Verifier: We now define the verifier $\mathcal{V}_{lin}$ for Min-3Lin-Deletion. $\mathcal{V}_{lin}$ expects the Hadamard codes of $P(U)$ and $Q(W)$ for all U and W. The Hadamard code for $Q(W)$ is assumed to be folded over W. The verifier performs the following test.

1. Pick a random u-tuple U of the variables.

2. Define two nodes U and W to be neighbors if $\forall i \in \{1, 2, \ldots u\}$, the i^{th} equation in W contains the i^{th} variable in U. Pick k u-tuples $(W_j)_{j \in 1,\ldots,k}$ of equations independently from the set of neighbors of U. Let $\pi_j = \pi_{U,W_j}$ be the projection from W_j to U. Let $\pi_j^{-1} : \mathbb{F}_2^u \to \mathbb{F}_2^{3u}$ be the function such that $\pi_j^{-1}(a)$ is the unique string that maps to a under the projection π_j and contains zeros at the $2u$ coordinates that get mapped out. Let A be the purported Hadamard code of $P(U)$. Let B_j be the purported Hadamard code of W_j. As mentioned above, the B_j are folded over W_j.

3. Pick k strings $a_1, a_2, \ldots, a_k \in \mathbb{F}_2^u$ independently at random. Pick k strings $b_1, b_2, \ldots,$ $b_k \in \mathbb{F}_2^{3u}$ independently at random.

4. Test whether $\forall i, j \in \{1, 2, \ldots k\}$

$$A(a_i)B_j(b_j)B_j(\pi_j^{-1}(a_i) \oplus b_j) = 0$$

and accept if all the k^2 linear tests above succeed.

The following is a slightly modified version of Theorem 3.1 in [69].

Lemma 4.3.4 *The verifier $\mathcal{V}_{lin}$ when given a 7-regular Min-3Lin-Deletion instance $\boldsymbol{A}$ of size N*

- *Uses $r = u \log N + O(ku)$ random bits.*
- *Queries $2k + k^2$ bits from the proof. Of these, $f = 2k$ are free query bits.*
- *If $Opt(\boldsymbol{A}) \leq \epsilon$, then the verifier accepts with probability $\geq 1 - \epsilon ku$.*
- *If $Opt(\boldsymbol{A}) \geq s$ and $(1 - (s/3)^{c_0})^u \leq \delta^2$, then the verifier accepts any proof with probability at most $2^{-k^2} + \delta$, where c_0 is the constant mentioned in Lemma 4.3.1.*

The first two claims about the verifier are easy to check. We now justify the claim about the completeness of the verifier. Suppose there exists an assignment satisfying $1 - \epsilon$ fraction of the equations in the Min-3Lin-Deletion instance. Consider a proof for $\mathcal{V}_{lin}$ constructed from one such assignment honestly. The ku equations corresponding to the W_js are all satisfied with probability $(1 - \epsilon)^{ku} \geq 1 - \epsilon ku$. The projection functions π_j are always satisfied for a honest proof. Let $x = P(U)$ and $y_j = Q(W_j)$ so that $\pi_j(y_j) = x \; \forall \; 1 \leq j \leq k$. Then,

$$\begin{aligned} A(a_i)B_j(b_j)B_j(\pi_j^{-1}(a_i) \oplus b_j) &= (-1)^{xa_i}(-1)^{y_j b_j}(-1)^{y_j(\pi_j^{-1}(a_i) \oplus b_j)} \\ &= (-1)^{xa_i}(-1)^{\pi_j(y_j)a_i} = 1 \end{aligned}$$

The proof of claim about the soundness of $\mathcal{V}_{lin}$ is based on the analysis of Samorodnitsky and Trevisan [97] and Fourier analysis of Hadamard codes. Khot [69] shows that if there exists a proof that is accepted by $\mathcal{V}_{lin}$ with probability greater than $2^{-k^2} + \delta$ for any $\delta > 0$, then it can be decoded to get a proof for the Raz Verifier that is accepted with probability δ^2. If $Opt(\boldsymbol{A}) \geq s$ and $(1 - (s/3)^{c_0})^u \leq \delta^2$, where c_0 is the constant defined in Lemma 4.3.1, this contradicts the soundness of the Raz Verifier. Khot [69] only considers the special case when s is a constant, like $4/10$.

4.3.2 From PCPs to Hardness of Approximating Max-Clique

The following theorem implicit in [108] and [12] relates the properties of a verifier for 3SAT to the inapproximability of Max-Clique. The proof of the theorem involves two main steps. The first step is the construction a FGLSS graph (see [33]) based on the verifier. The second step involves boosting the gap in the size of the largest clique in the FGLSS graph using a technique of Zuckerman [108] based on dispersers. The reader is referred to Engebretsen and Holmerin [31, Lemma 6.3] for the explicit statement and proof of the theorem.

Lemma 4.3.5 *Assume there exists a verifier for 3SAT that uses r random bits, makes f free queries and achieves completeness c and soundness s. Let $R > r$ be any integer. Let $D = (R+2)/\log s^{-1}$. Then, there exists a reduction from 3SAT to Max-Clique that when given a 3SAT instance ϕ produces a graph G with $N' = 2^{R+Df}$ vertices such that:*

- *If ϕ is satisfiable, then with probability 2/3, there is a clique of size at least $c^D 2^R/2$ in G.*
- *If ϕ is not satisfiable, then with probability 2/3, the size of the largest clique in G is at most 2^r.*

The running time of the reduction is polynomial in N' and the running time of the verifier.

In the same way as in Khot [69], we combine it with Lemma 4.3.4 which implies a verifier for 3SAT with low amortized free bit complexity and obtain the inapproximability result for Max-Clique.

4.3.3 Randomized PCPs and Hardness of Approximating Chromatic Number

Feige and Kilian [35] showed the connection between *randomized PCPs* and inapproximability of chromatic number. We first define randomized PCPs before we state their formal result.

Definition 4.3.6 *A pair (S, ν) is said to be an accepting pattern of a PCP verifier on input x if for some random string, S is the set of bits read from the proof and ν is an assignment to these bits that makes the verifier accept. We denote by $\mathcal{T}_x$ the set of all accepting patterns*

for input x. A proof Π is said to be consistent with $\tau = (S, \nu)$ if the value specified by Π for the positions in S is ν.

A PCP verifier V is said to be a randomized PCP for language L with parameters (r, f, ρ, s) if the verifier uses r random bits, uses f free bits and satisfies the following conditions:

- *If the input x to the verifier V belongs to L, then there exists a set of proofs $\{\Pi_1, \Pi_2, \ldots\}$ and a probability distribution on the proofs such that for all $\tau \in \mathcal{T}_x$,*

$$\Pr_i[\Pi_i \text{ is consistent with } \tau] \geq \rho.$$

  In this case, we say that x has a ρ-covering set of proofs.

- *If the input x to V does not belong to L, then for any proof Π,*

$$\Pr[V \text{ accepts the proof } \Pi] \leq s$$

Feige and Kilian [35] start with the Long code based PCP verifier for 3SAT given by Håstad [50] and then *randomize* it. Given an instance ϕ of 3SAT, they construct the FGLSS graph G (see [33]) based on this verifier, construct the product graph G^t and take its randomly induced subgraph G' of appropriate size. For our purposes, we will need the following result of [35] that relates the satisfiability of ϕ to the chromatic number of G'. This theorem is implicit in [35] and the reader is referred to Khot [69] for the explicit statement and proof.

Lemma 4.3.7 *If 3SAT has a randomized PCP with parameters (r, f, ρ, s), then for any integer h, there exists a randomized reduction from 3SAT to chromatic number that when given a 3SAT instance ϕ produces a graph G' over $N' = (2^r/s)^h$ vertices satisfying the following conditions:*

- *If ϕ is satisfiable, $\chi(G') \leq \frac{2 \ln N'}{\rho}$.*

- *If ϕ is unsatisfiable, with probability 1/2, $\chi(G') \geq \frac{N'}{h 2^{r+f}}$, where $\chi(G')$ denotes the chromatic number of the graph G'.*

The running time of the reduction is polynomial in N' and the running time of the randomized PCP verifier for 3SAT.

Khot [69] gives an alternative approach to obtain a randomized PCP based on his Hadamard code based verifier for Min-3Lin-Deletion. This verifier is easier to randomize and yields a stronger hardness factor for chromatic number. The first step is to show a stronger reduction from 3SAT to Min-3Lin-Deletion such that if the 3SAT instance is satisfiable, then instead of just one good assignment to the Min-3Lin-Deletion instance, we have a collection of assignments that "cover" all the equations in it.

Lemma 4.3.8 ([69]) *For any constants $\epsilon, \delta > 0$, there exists a polynomial time algorithm $\mathcal{A}'_1$ that when given a 3SAT formula ϕ of size n produces a Min-3Lin-Deletion instance $\boldsymbol{A}_1$ such that:*

- *(Yes Case:) If ϕ is satisfiable, there exists a set of assignments Λ for $\boldsymbol{A}_1$ such that for every equation in $\boldsymbol{A}_1$, at least $(1-\epsilon)$ fraction of the assignments satisfy it. We say that $\boldsymbol{A}_1$ is* $(1-\epsilon)$-coverable *and Λ is a* $(1-\epsilon)$-covering *for $\boldsymbol{A}_1$.*
- *(No Case:) If ϕ is not satisfiable, then no assignment satisfies more than $1/2 + \delta$ fraction of the equations.*

We finally need the following result from Khot [69] to construct a randomized PCP for coverable Min-3Lin-Deletion.

Lemma 4.3.9 ([69]) *For any $\epsilon, k, u > 0$ (where all the parameters can be a function of N), there exists a randomized PCP verifier $\mathcal{V}_{rand}$ for Min-3Lin-Deletion such that $\mathcal{V}_{rand}$ when given an input $\boldsymbol{A}$ of size N:*

- *Uses $r = u \log N + O(ku)$ random bits and $f = 2k$ free query bits.*
- *If $\boldsymbol{A}$ is $(1-\epsilon)$-coverable and $\epsilon k u \leq 1/2$, then there exists a ρ-covering proof for $\boldsymbol{A}$ where $\rho = 2^{-(2k+1)}$.*
- *If no assignment satisfies more than $1-s$ fraction of the equations in $\boldsymbol{A}$ and $(1-(s/3)^{c_0})^u \leq \delta^2$, then the verifier accepts any proof with probability at most $2^{-k^2} + \delta$, where c_0 is the constant mentioned in Lemma 4.3.1.*

The running time of $\mathcal{V}_{rand}$ is polynomial in 2^r.

4.4 Hardness of Approximating Min-Lin-Deletion

We begin by analyzing two operations on Min-Lin-Deletion necessary to accomplish Step 1 in Figure 1. The first operation called *tensoring* is similar to the operation of tensor product of two linear codes used in [30].

Definition 4.4.1 *The tensoring of a system of linear equations modulo 2*

$$\{a_{i0} \oplus (\bigoplus_{j=1}^{l} a_{ij}x_j) = 0\}_{i=1,2,\ldots,M}$$

is the system

$$\{a_{i_1 0}a_{i_2 0} \oplus (\bigoplus_{j_2=1}^{l} a_{i_1 0}a_{i_2 j_2}x_{0j_2}) \oplus (\bigoplus_{j_1=1}^{l} a_{i_1 j_1}a_{i_2 0}x_{0j_1}) \oplus (\bigoplus_{j_1=1}^{l}\bigoplus_{j_2=1}^{l} a_{i_1 j_1}a_{i_2 j_2}x_{j_1 j_2}) = 0\}_{i_1,i_2=1,\ldots,M}$$

In other words, for every pair of equations, the tensoring contains a linear equation that is obtained by taking the product of the two equations and replacing $x_{j_1}x_{j_2}$ in each term with the variable $x_{j_1 j_2}$ and x_j with x_{0j}. In the matrix form, the tensoring of the Min-Lin-Deletion instance

$$\begin{bmatrix} a_{10} & a_{11} & \ldots & a_{1l} \\ a_{20} & a_{21} & \ldots & a_{2l} \\ \vdots & \vdots & & \vdots \\ a_{M0} & a_{M1} & \ldots & a_{Ml} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_l \end{bmatrix} = 0$$

is

$$\begin{bmatrix} a_{10} & a_{11} & \ldots & a_{1l} \\ a_{20} & a_{21} & \ldots & a_{2l} \\ \vdots & \vdots & & \vdots \\ a_{M0} & a_{M1} & \ldots & a_{Ml} \end{bmatrix} \begin{bmatrix} 1 & x_{01} & \ldots & x_{0l} \\ x_{01} & x_{11} & \ldots & x_{1l} \\ \vdots & \vdots & & \vdots \\ x_{0l} & x_{l1} & \ldots & x_{ll} \end{bmatrix} \begin{bmatrix} a_{10} & a_{20} & \ldots & a_{M0} \\ a_{11} & a_{21} & \ldots & a_{M1} \\ \vdots & \vdots & & \vdots \\ a_{1l} & a_{2l} & \ldots & a_{Ml} \end{bmatrix} = 0$$

where the the x_{ij}s in the second matrix are the variables in the new instance.

Lemma 4.4.2 *Let $Opt(\boldsymbol{A}) \leq \epsilon$ for some instance $\boldsymbol{A}$ of Min-Lin-Deletion. Let $\boldsymbol{B}$ be the Min-Lin-Deletion instance obtained by tensoring $\boldsymbol{A}$. Then $Opt(\boldsymbol{B}) \leq \epsilon^2$*

Proof: Fix an assignment to the variables $x_1, x_2, \ldots, x_l$ in the Min-Lin-Deletion instance

$$\{(\bigoplus_{j=1}^{l} a_{ij}x_j) \oplus a_{i0} = 0\}_{i=1,2,\ldots,M}$$

that satisfies at least a fraction $1-\epsilon$ of the equations. Define $x_{j_1j_2} = x_{j_1}x_{j_2}$ and $x_{0j_1} = x_{j_1}$ for all $j_1, j_2 \in \{1, 2, \ldots, l\}$. Then,

$$\begin{aligned}
&a_{i_10}a_{i_20} \oplus (\bigoplus_{j_2=1}^{l} a_{i_10}a_{i_2j_2}x_{0j_2}) \oplus (\bigoplus_{j_1=1}^{l} a_{i_1j_1}a_{i_20}x_{0j_1}) \oplus (\bigoplus_{j_1=1}^{l}\bigoplus_{j_2=1}^{l} a_{i_1j_1}a_{i_2j_2}x_{j_1j_2}) \\
&= a_{i_10}a_{i_20}1.1 \oplus (\bigoplus_{j_2=1}^{l} a_{i_10}a_{i_2j_2}.1.x_{j_2}) \oplus (\bigoplus_{j_1=1}^{l} a_{i_1j_1}a_{i_20}x_{j_1}.1) \\
&\quad \oplus (\bigoplus_{j_1=1}^{l}\bigoplus_{j_2=1}^{l} a_{i_1j_1}a_{i_2j_2}x_{j_1}x_{j_2}) \\
&= (a_{i_10} \oplus (\bigoplus_{j_1=1}^{l} a_{i_1j_1}x_{j_1}))(a_{i_20} \oplus (\bigoplus_{j_2=1}^{l} a_{i_2j_2}x_{j_2}))
\end{aligned}$$

Clearly this evaluates to 1 only if the i_1^{th} equation and the i_2^{th} equation of the Min-Lin-Deletion instance $\boldsymbol{A}$ are both not satisfied. Therefore if at most ϵM equations of the instance $\boldsymbol{A}$ are not satisfied, then at most $\epsilon^2 M^2$ equations of the instance $\boldsymbol{B}$ are not satisfied. □

Lemma 4.4.3 *Let $Opt(\boldsymbol{A}) \geq s$ for some instance $\boldsymbol{A}$ of (M, l)-Min-Lin-Deletion. Let $\boldsymbol{B}$ be the Min-Lin-Deletion instance obtained by tensoring $\boldsymbol{A}$. Then $Opt(\boldsymbol{B}) \geq s^2$*

Proof: Fix any assignment $(x_{j_1j_2})$ to the tensoring of $\boldsymbol{A}$. Denote by $\boldsymbol{X}^*$ the matrix

$$\begin{bmatrix} 1 & x_{01} & \ldots & x_{0l} \\ x_{01} & x_{11} & \ldots & x_{1l} \\ \vdots & \vdots & & \vdots \\ x_{0l} & x_{l1} & \ldots & x_{ll} \end{bmatrix}$$

We want to show that the $M \times M$ matrix $\boldsymbol{A}\boldsymbol{X}^*\boldsymbol{A}^T$ has a 1 in at least s^2 fraction of the entries.

Let $\boldsymbol{X} = (1, x_{01}, x_{02}, \ldots, x_{0l})$. Since $Opt(\boldsymbol{A}) \geq s$, $\boldsymbol{A}\boldsymbol{X}$ has at least sM non-zero entries. Define $\boldsymbol{V} = \boldsymbol{A}\boldsymbol{X}^*$ Then, $\boldsymbol{V}$ is an $M \times (l+1)$ matrix with at least sM 1s in the first column. That is, $\boldsymbol{V}^T$ has at least sM columns that have a 1 in the first row. Again, since $Opt(\boldsymbol{A}) \geq s$ the product of each of these sM column vectors with $\boldsymbol{A}$ results in a column vector with sM 1s. Therefore, $\boldsymbol{A}\boldsymbol{V}^T$ has a 1 in s^2M^2 entries, and hence so does $\boldsymbol{A}\boldsymbol{X}^*\boldsymbol{A}^T = \boldsymbol{V}\boldsymbol{A}^T = (\boldsymbol{A}\boldsymbol{V}^T)^T$. □

We next define an operation called boosting that brings the optimum of a Min-Lin-Deletion instance closer to 1/2.

Definition 4.4.4 *Let $\boldsymbol{A}$ be a (M, l)-Min-Lin-Deletion instance. Let G_M be the 5-regular Gabber-Galil graph with M vertices (If M is not of the form $2p^2$, add a sufficient number of all-zeros rows to $\boldsymbol{A}$). Associate each vertex v of G_M with a row $\boldsymbol{A}_v$ of $\boldsymbol{A}$. Let $\sigma = 5 \times 10^9$. We define the boosting of $\boldsymbol{A}$ to be the $(l5^{\sigma-1}2^{\sigma}, l)$-Min-Lin-Deletion instance $\boldsymbol{B}$ obtained as follows. For each of the $l5^{\sigma-1}$ possible walks $(v_1, v_2, \ldots, v_\sigma)$ on G_p of length σ, add 2^σ rows to $\boldsymbol{B}$ where each row is one of the 2^σ possible linear combinations of the row-vectors $\boldsymbol{A}_{v_1}, \boldsymbol{A}_{v_2}, \ldots, \boldsymbol{A}_{v_\sigma}$. We call each such group of 2^σ rows a cluster of rows.*

In the above definition, we generate cluster of rows corresponding to only $O(M)$ of all possible elements of V^σ. We do so in order to keep the size of the Min-Lin-Deletion instances produced small. We also note that, for our purposes, any class of expander graphs that can be generated in polynomial time will do in the above definition.

Lemma 4.4.5 *Let $\boldsymbol{A}$ be a (M, l)-Min-Lin-Deletion instance with $Opt(\boldsymbol{A}) \leq \epsilon$, and let $\boldsymbol{B}$ be its boosting. Then, $Opt(\boldsymbol{B}) \leq \sigma\epsilon$.*

Proof: Adding zero-rows to $\boldsymbol{A}$ to make the number of rows of the form $2p^2$ only decreases $Opt(\boldsymbol{A})$. Hence we ignore that issue in the proof. Fix a vector $\boldsymbol{X} = (1, x_1, \ldots, x_l)$ such that $\boldsymbol{AX}$ has the minimum number of 1s possible. Let $(v_1, v_2, \ldots, v_\sigma)$ be a walk on G_M. Suppose $\boldsymbol{A}_{v_i}\boldsymbol{X} = 0$ for all $1 \leq i \leq \sigma$. Then any linear combination of the $\boldsymbol{A}_{v_i}$s also has a dot-product zero with $\boldsymbol{X}$. That is, all rows of $\boldsymbol{B}$ in the cluster corresponding to this walk have dot-product zero with $\boldsymbol{X}$. It is a easy consequence of Lemma 3.2.6 that at least a fraction $1 - \sigma\epsilon$ of the clusters in $\boldsymbol{B}$ satisfy this property. This implies that $\boldsymbol{BX}$ is a vector with at least $1 - \sigma\epsilon$ fraction of the entries being zero. □

We use the following corollary to Lemma 3.2.4 to analyze the soundness of the boosted instance. The proof is similar to that of Corollary 3.2.5.

Corollary 4.4.6 *Let W be a subset of G_M containing at most a fraction 0.85 of the vertices. Then, for $\sigma = 5 \times 10^9$, the fraction of walks of length σ that do not contain a vertex from $\bar{W}$ is at most 0.2.*

Lemma 4.4.7 *Let $\boldsymbol{A}$ be a (M,l)-Min-Lin-Deletion instance with $Opt(\boldsymbol{A}) \geq 0.16$, and let $\boldsymbol{B}$ be its boosting. Then, for M large enough, $OPT(\boldsymbol{B}) \geq 0.4$.*

Proof: Since we need to add only $O(\sqrt{M})$ zero-rows to $\boldsymbol{A}$ to make the number of rows to be of the form $2p^2$, we can assume $Opt(\boldsymbol{A}) \geq 0.16 - O(\sqrt{M}/M) \geq 0.15$ for M large enough. Fix any vector $\boldsymbol{X} = (1, x_1, \ldots, x_l)$. Then, $\boldsymbol{AX}$ has a 1 at at least 0.15 fraction of the entries. Consider a walk $(v_1, v_2, \ldots, v_\sigma)$ on G_p. Suppose $\boldsymbol{A}_{v_i}\boldsymbol{X} \neq 0$ for some $1 \leq i \leq \sigma$. Then, exactly half of the 2^σ linear combinations of the $\boldsymbol{A}_{v_i}$s have non-zero dot-product with $\boldsymbol{X}$. From Corollary 4.4.6, at least 0.8 fraction of the walks of length σ visit a row that has a non-zero dot product with $\boldsymbol{X}$. Since half of the rows in the cluster corresponding to such a walk have a non-zero dot-product with $\boldsymbol{X}$, $\boldsymbol{BX}$ has at least $1/2 \times 0.8 = 0.4$ fraction of the entries as non-zero. Since this statement is true for any arbitrary vector $\boldsymbol{X} = (1, x_1, \ldots, x_l)$, $Opt(\boldsymbol{B}) \geq 0.4$. □

Theorem 4.4.8 *There exists a $2^{O(\log^2 n)}$ time algorithm $\mathcal{A}_2$ that when given a Min-3Lin-Deletion instance $\boldsymbol{A}_1$ of size N_1 outputs a Min-Lin-Deletion instance $\boldsymbol{A}_2$ of size $N_2 = 2^{O(\log^2 n)}$ such that:*

- *(Yes Case:) If $Opt(\boldsymbol{A}_1) \leq 10^{-10}$, then $Opt(\boldsymbol{A}_2) \leq 2^{-\Omega(\sqrt{\log N_2})}$*
- *(No Case:) If $Opt(\boldsymbol{A}_1) \geq 0.4$, then $Opt(\boldsymbol{A}_2) \geq 0.4$*

Proof: The algorithm $\mathcal{A}_2$ works by repeatedly tensoring and boosting $\boldsymbol{A}_1$. Let $\boldsymbol{B}_1$ be the Min-Lin-Deletion instance obtained by tensoring $\boldsymbol{A}_1$ and let $\boldsymbol{B}'_1$ be the boosting of $\boldsymbol{B}_1$. If $Opt(\boldsymbol{A}_1) \leq \epsilon$, then $Opt(\boldsymbol{B}'_1) \leq \sigma\epsilon^2$. On the other hand, if $Opt(\boldsymbol{A}_1) \geq 0.4$, then $Opt(\boldsymbol{B}'_1) \geq 0.4$ too since $\boldsymbol{B}_1$ will have soundness $0.4^2 = 0.16$. $\boldsymbol{B}_1$ has N_1^2 equations and hence $\boldsymbol{B}'_1$ has at most $O(N_1^2)$ equations.

Let $\boldsymbol{A}_2$ be obtained from $\boldsymbol{A}_1$ by repeatedly tensoring and boosting it $q = \log\log N_1$ times. Then, $\boldsymbol{A}_2$ has $N_2 = (O(N_1))^{2^q}$ equations. The construction of $\boldsymbol{A}_2$ from $\boldsymbol{A}_1$ takes $2^{O(\log^2 n)}$ time since tensoring and boosting are both polynomial time operations in their input size. We also have:

- *(Yes Case:)* If $Opt(\boldsymbol{A}_1) \leq 10^{-10}$, we get $Opt(\boldsymbol{A}_2) \leq (O(1))^{2^q}$ since $\epsilon\sigma \leq 1/2$. This implies $\log N_2 = O(2^q \log N_1)$, and hence $\log N_2 = O(\log^2 N_1)$. Therefore, $2^q = \log N_1 \geq \Omega(\sqrt{\log N_2})$.

- *(No Case:)* If $Opt(\boldsymbol{A}_1) \geq 0.4$, then $Opt(\boldsymbol{A}_2) \geq 0.4$.

□

4.5 Hardness of Approximating Min-3Lin-Deletion

In this section, we formally state the second step mentioned in Figure 1. The equations in $\boldsymbol{A}_2$ produced by $\mathcal{A}_2$ in Theorem 4.4.8 can be over a large number of variables. We use the Sum-Check Protocol combined with the Low-degree Test (see Arora [7]) to get the number of variables in an equation down to a poly-logarithmic number.

Theorem 4.5.1 *There exists a polynomial-time reduction $\mathcal{A}_3$ from a Min-Lin-Deletion instance $\boldsymbol{A}_2$ of size N_2 to a Min-Lin-Deletion instance $\boldsymbol{A}_3$ such that:*

- *$\boldsymbol{A}_3$ has size $N_3 = N_2^{O(1)}$. Also, $\boldsymbol{A}_3$ is $\log^3 N_3$-restricted. That is, each equation of $\boldsymbol{A}_3$ is over at most $\log^3 N_3$ variables.*

- *$Opt(\boldsymbol{A}_3) \leq Opt(\boldsymbol{A}_2)$. This implies that $Opt(\boldsymbol{A}_3) \leq 2^{-\Omega(\sqrt{\log N_3})}$ if we have $Opt(\boldsymbol{A}_2) \leq 2^{-\Omega(\sqrt{\log N_2})}$.*

- *Assuming N_2 is large enough, $Opt(\boldsymbol{A}_2) \geq 0.4$ implies $Opt(\boldsymbol{A}_3) \geq 0.15$.*

Proof: To prove the theorem, we first construct a probabilistic polynomial-time verifier $\mathcal{V}$ for Min-Lin-Deletion with the following properties:

1. The acceptance test is non-adaptive. That is, $\mathcal{V}$ determines the test only based on $\boldsymbol{A}_2$ and its random string.

2. Each test of $\mathcal{V}$ is a logical AND of k smaller tests, where $k = O(\log N_2)$ and each of the smaller tests is a linear equation mod 2 over $O(\log^2 N_2)$ bits of the certificate/proof. We call these the basic tests corresponding to the random string.

3. $\mathcal{A}_3$ uses $O(\log N_2)$ random bits.

4. *(Yes Case:)* There exists a certificate for $\boldsymbol{A}_2$ that is accepted with probability $\geq 1 - Opt(\boldsymbol{A}_2)$.

5. *(No Case:)* If $Opt(\boldsymbol{A}_2) \geq 0.4$, then any certificate for $\boldsymbol{A}_2$ is accepted with probability at most 0.7.

The variables in the output Min-Lin-Deletion instance $\boldsymbol{A}_3$ then correspond to the bits in the certificate. For each possible random string of length $O(\log N_2)$ to $\mathcal{V}$, $\mathcal{A}_3$ adds $2^k = 2^{O(\log N_2)}$ equations, corresponding to all possible linear combinations of the basic tests mentioned in Property 2. Therefore, $\boldsymbol{A}_3$ has $N_3 = N_2^{O(1)}$ linear equations where each equation is over $kO(\log^2 N_2) = O(\log^3 N_2)$ variables. By Property 4, there is an assignment to the variables in $\boldsymbol{A}_3$ such that all the 2^k equations corresponding to $1 - Opt(\boldsymbol{A}_2)$ fraction of the random strings are accepted. If $Opt(\boldsymbol{A}_2) \geq 0.4$, then it follows from Property 5 that for at least a 0.3 fraction of the random strings, one or more of the $O(\log N_2)$ basic tests corresponding to each of these strings must fail. This implies that if $Opt(\boldsymbol{A}_2) \geq 0.4$, then $Opt(\boldsymbol{A}_3) \geq 0.3 \times 1/2 = 0.15$. We next describe the verifier.

Let the number of variables in $\boldsymbol{A}_2$ be n_2. Then, $n_2 \leq N_2^2$ since each equation in $\boldsymbol{A}_2$ is over at most N_2 variables. Define $h = \lceil \log n_2 \rceil$, $m = \lceil \log n_2 / \log \log n_2 \rceil$ and $d = (h-1)m$. Then, $h^m \geq n_2$. The verifier $\mathcal{V}$ picks a field $(\mathbb{F}, +, \cdot)$ of characteristic 2 with 2^q elements, where $2^q \geq d^3 m$. Let S be any subset of $\mathbb{F}$ of size h. The elements in $\mathbb{F}$ can be represented by bit strings of length q such that for any $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_q)$ and $\alpha' = (\alpha'_1, \alpha'_2, \ldots, \alpha'_q)$ in $\mathbb{F}$, we have

- $\alpha + \alpha'$ is the bitwise xor of the two strings.
- The k^{th} bit of $\alpha \cdot \alpha'$ is $\oplus_{i,j=1,2,\ldots,q} \alpha_i c_{ijk} \alpha'_j$, where the c_{ijk} only depends on the field.

Let the prover wish to prove that a certain assignment to the variables $x_1, x_2, \ldots, x_{n_2}$ in $\boldsymbol{A}_2$ satisfies a "large" number of linear equations. The prover is expected to provide a certificate consisting of:

- The values of a multivariate polynomial $f(y_1, y_2, \ldots, y_m)$ on the points in $\mathbb{F}^m$, where f has degree $h-1$ in each of the m variables. This is called the *Points Table*.

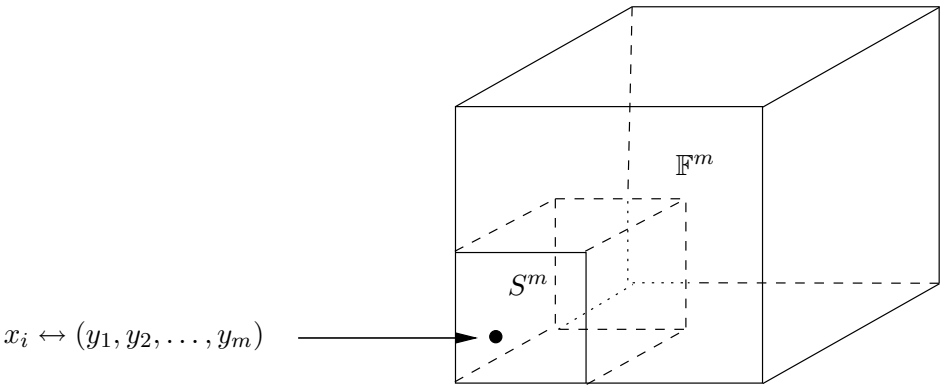

Figure 2: The hypercube $\mathbb{F}^m$.

- For every line l in $\mathbb{F}^m$, a univariate degree $d = m(h-1)$ polynomial $g_l(t)$. This is called the *Lines Table*.

- For the i^{th} line in $\mathbb{F}^m$, for every $k = 0, 1, \ldots, m-1$ and every $(\theta_1, \theta_2, \ldots, \theta_k) \in \mathbb{F}^k$, the coefficients of a univariate polynomial $p_{i,\theta_1,\theta_2,\ldots,\theta_k}(y_{k+1})$ with degree $2(h-1)$. This is called the *Partial Sums Table*. The polynomials for $k = 0$ are denoted as $p_{i,\emptyset}(y_1)$.

All the field elements are expected to be provided as bit-strings with the aforementioned properties. $\mathcal{V}$ associates the variables in $\boldsymbol{A}_2$ to different points in S^m (see Figure 2). The last bit of $f(y_1, y_2, \ldots, y_m)$ is supposed to be the value to be assigned to the variable x_j corresponding to $(y_1, y_2, \ldots, y_m)$. It can be verified that the degree of f is large enough to represent any assignment to the variables x_j. Let $c_i(y_1, y_2, \ldots, y_m)$ be the unique multivariate polynomial of degree $\leq h-1$ in each variable such that $c_i(y_1, y_2, \ldots, y_m)$ is the coefficient of the variable x_j corresponding to $(y_1, y_2, \ldots, y_m)$ in the i^{th} equation of $\boldsymbol{A}_2$ (If no variable x_j corresponds to $(y_1, y_2, \ldots, y_m) \in S^m$, then $c_i(y_1, y_2, \ldots, y_m)$ is defined to be zero). Note that $c_i(y_1, y_2, \ldots, y_m)$ can be computed from $\boldsymbol{A}_2$. Then testing that an equation $a_{i0} \oplus (\bigoplus_{j=1}^{m} a_{ij} x_j) = 0$ is satisfied reduces to checking if

$$\sum_{(y_1,y_2,\ldots,y_m) \in S^m} c_i(y_1, y_2, \ldots, y_m) f(y_1, y_2, \ldots, y_m) = a_{i0}$$

Note that $c_i(y_1, y_2, \ldots, y_m) f(y_1, y_2, \ldots, y_m)$ is a degree $2(h-1)m$ polynomial. The polynomial $p_{i,\theta_1,\theta_2,\ldots,\theta_k}(y_{k+1})$ is supposedly the unique degree $2(h-1)$ univariate polynomial such

that

$$p_{i,\theta_1,\theta_2,\ldots,\theta_k}(y_{k+1}) =$$

$$\sum_{y_{k+2},\ldots,y_m \in S^{m-k-1}} c_i(\theta_1,\ldots,\theta_k,y_{k+1},\ldots,y_m) f(\theta_1,\ldots,\theta_k,y_{k+1},\ldots,y_m)$$

Also, $g_l(t)$ is the supposed restriction of the polynomial f to the line l. That is, if $l(t) = \boldsymbol{\theta} + \boldsymbol{\theta}' t$ is the parametric representation of the line l for some $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{F}^m$, then $g_l(t) = f(l(t))$.

The verifier performs the following tests:

1. (The Low-degree Test) Repeat the following $4\delta^{-1}$ times, where $\delta = 10^{-4}/2$. Pick a line l in $\mathbb{F}^m$ and $t \in \mathbb{F}$ uniformly at random. Check that $g_l(t) = f(l(t))$.

2. (The Sum-Check Protocol) Pick $i \in \{1, 2, \ldots, N_2\}$ uniformly at random (i.e., the verifier is trying to verify the i^{th} equation of $\boldsymbol{A}_2$). Pick $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_m)$ from $\mathbb{F}^m$ uniformly at random.

   (a) Check that $\sum_{y_1 \in S} p_{i,\emptyset}(y_1) = a_{i0}$

   (b) Check that $\forall j : 1 \leq j \leq m-1$,

   $$\sum_{y_{j+1} \in S} p_{i,\theta_1,\ldots,\theta_j}(y_{j+1}) = p_{i,\theta_1,\ldots,\theta_{j-1}}(\theta_j)$$

   (c) Check that $p_{i,\theta_1,\ldots,\theta_{m-1}}(\theta_m) = c_i(\theta_1,\ldots,\theta_m) f(\theta_1,\ldots,\theta_m)$

The verifier accepts if all the above tests succeed. The test is clearly non-adaptive. All the $O(1)+1+O(m)+1 = O(m)$ basic tests mentioned above are linear in the field elements read, and hence can be broken down into $k = O(m \log |\mathbb{F}|) = O(m \log(d^3 m)) = O(\log N_2)$ smaller linear tests over the bits read. Each of these smaller tests is over at most $O(d) \log |\mathbb{F}| = O(\log^2 N_2)$ bits read. The randomness used is $O(\log |\mathbb{F}|) + O(\log N_2) + O(m \log |\mathbb{F}|) = O(\log N_2)$. It is easy to see that if the prover selects the best assignment to $\boldsymbol{A}_2$ and constructs the proof honestly as expected, it is accepted with probability $\geq 1 - Opt(\boldsymbol{A}_2)$. On the other hand, if $Opt(\boldsymbol{A}_2) \geq 0.4)$, then no proof is accepted with probability ≥ 0.7. The analysis of Sum-Check Protocol and Low-degree Test are based on Theorem 4.15 and Lemma 4.12 of Arora [7].

Suppose the prover selects the best assignment to the variables in $\boldsymbol{A}_2$ and constructs the certificate honestly as expected. The Low-degree Test always succeeds. The Sum-Check Protocol also succeeds when the equation of $\boldsymbol{A}_2$ that is selected is satisfied by the assignment. This happens with probability at least $1 - Opt(\boldsymbol{A}_2)$. Therefore, there is always a certificate that is accepted with probability $1 - Opt(\boldsymbol{A}_2)$.

Let us now consider the case when $Opt(\boldsymbol{A}_2) \geq 0.4$. Assume the Low-degree Test succeeds with probability $\geq 1/4$. Since $|\mathbb{F}| = \Omega(d^3 m)$ and $\delta \leq 10^{-4}$, Arora [7, Theorem 4.15] implies that f agrees with some degree $d = m(h-1)$ polynomial F on all but δ fraction of the points in $\mathbb{F}^m$. Therefore, with probability $1 - \delta$, f and F agree on the point $(\theta_1, \ldots, \theta_m)$ used in the Sum-Check Protocol.

We now analyze the Sum-Check Protocol. Pick a value of i uniformly at random. With probability at least 0.4,

$$\sum_{(y_1, y_2, \ldots, y_m) \in S^m} c_i(y_1, y_2, \ldots, y_m) F(y_1, y_2, \ldots, y_m) \neq a_{i0}$$

Assume this is the case. From Lemma 4.12 in [7], with probability at least $1 - 2dm/|\mathbb{F}|$, either one of the tests in Test 2a or Test 2b fails, or

$$p_{i,\theta_1,\ldots,\theta_{m-1}}(\theta_m) \neq c_i(\theta_1, \ldots, \theta_m) F(\theta_1, \ldots, \theta_m)$$

With probability $1 - \delta$, f and F agree at $(\theta_1, \ldots, \theta_m)$. Therefore, with probability $\geq 0.4(1 - 2dm/|\mathbb{F}|) - \delta = 0.4(1 - \Omega(d^{-2})) - \delta$, the Sum-Check protocol fails.

Therefore, assuming N_2 is large enough, either the Low-degree Test fails with probability $\geq 3/4$, or the Sum-Check Protocol fails with probability ≥ 0.3. That is, no proof is accepted with probability greater than $1 - 0.3 = 0.7$. □

Proof of Theorem 1.4.10: The algorithm $\mathcal{A}$ first uses $\mathcal{A}_2$ and $\mathcal{A}_3$ to convert a Min-3Lin-Deletion instance $\boldsymbol{A}_1$ of size N_1 to a Min-Lin-Deletion instance $\boldsymbol{A}_3$ of size $N_3 = 2^{O(\log^2 n)}$. $\mathcal{A}$ produces its output by breaking each linear equation in $\boldsymbol{A}_3$ into $O(\log^3 N_3)$ smaller equations having at most 3 variables by introducing auxiliary variables. An equation of the form

$$x_1 \oplus x_2 \oplus \ldots \oplus x_k = a_0$$

is broken into

$$x_1 \oplus x_2 \oplus y_1 = 0$$

$$y_1 \oplus x_3 \oplus y_2 = 0$$

$$\vdots$$

$$y_{i-2} \oplus x_i \oplus y_{i-1} = 0$$

$$\vdots$$

$$y_{k-3} \oplus x_{k-1} \oplus x_k = a_0$$

We call the smaller equations added for an equation in $\boldsymbol{A}_3$ to be a cluster of equations corresponding to it. For simplicity, we will assume that each cluster has the same number of equations (just add enough number of trivially satisfied equations). The size of $\boldsymbol{A}$ is $N = O(N_3 \log^3 N_3)$.

It is easy to see that all the equations in a cluster can be satisfied if the equation in $\boldsymbol{A}_3$ corresponding to the cluster is satisfied. It then follows immediately that $Opt(\boldsymbol{A}) \leq Opt(\boldsymbol{A}_3)$. Also, if $Opt(\boldsymbol{A}_3) \geq 0.15$, then $Opt(\boldsymbol{A}) \geq \Omega(\log^{-3} N_3)$. This is because for any assignment, at least 0.15 fraction of equations in $\boldsymbol{A}_3$ are unsatisfied, and for each unsatisfied equation, at least one equation in its cluster is not satisfied.

The Min-3Lin-Deletion instance $\boldsymbol{A}$ as defined above is not regular. But it can be converted to a 7-regular Min-3Lin-Deletion instance as in the proof of Theorem 10.2 in [9]. This only incurs a constant factor loss in the soundness. □

4.6 Hardness of Approximating Max-Clique

We now prove Theorem 1.4.7. The proof is by reduction from 3SAT to Max-Clique. We first construct a verifier for 3SAT. We then apply Lemma 4.3.5 to this verifier.

Proof of Theorem 1.4.7: Fix any constant $\gamma > 0$. Let $\beta > 0$ be a constant that will later be fixed to a large enough value. Let $\mathcal{A}_1$ be a polynomial time reduction from 3SAT to Min-3Lin-Deletion-$(10^{-10}, 0.4)$ whose existence is guaranteed in Lemma 1.4.9. We first compose the algorithms $\mathcal{A}_1$ and $\mathcal{A}$ to obtain a $2^{O(\log^2 n)}$ time algorithm that when given a 3SAT instance ϕ of size n produces a 7-regular Min-3Lin-Deletion instance $\boldsymbol{A}$ of size $N = 2^{O(\log^2 n)}$ such that:

- If ϕ is satisfiable, then $Opt(\boldsymbol{A}) \leq \epsilon$, where $\epsilon = \log^{-\beta} N$ (note that $2^{-\Omega(\sqrt{\log N})} \leq \log^{-\beta} N$).

- If ϕ is not satisfiable, then $Opt(\boldsymbol{A}) \geq s'$, where $s' = \Omega(\log^{-3} N)$ can be achieved.

We next use the verifier $\mathcal{V}_{lin}$ mentioned in Lemma 4.3.4 with the parameters as $u = \frac{1}{2}\sqrt[3]{\frac{5}{\epsilon^2 \log(1-s'^{c_0})^{-1}}}$ and $k = \sqrt[3]{\frac{\log(1-s'^{c_0})^{-1}}{5\epsilon}}$ to obtain a verifier for 3SAT with the following parameters:

- The constant β can be made large enough so that $r = O(ku) = O(\epsilon^{-1}) = O(\log^{\beta} N) = (\log n)^{O(1)}$. The number of free bits is $f = 2k$.

- If ϕ is satisfiable, the verifier accepts with probability $c \geq 1 - \epsilon k u \geq 1/2$.

- Let $\delta = 2^{-k^2}$. Then $(1-s'^{c_0})^u \leq \delta^2$, which implies the soundness s of the verifier is at most $2 \cdot 2^{-k^2}$.

Note that $k = O((\log^{\beta/3} N) \times (\log^{-c_0} N)) = (\log n)^{O(1)}$ and $u = O((\log^{2\beta/3} N) \times (\log^{c_0} N)) = (\log n)^{O(1)}$. Therefore, $u = O(k^2 \log^{3c_0} N)$.

As the final step we use Lemma 4.3.5 on the above verifier for 3SAT with the parameter $R = rk$. Then, $D = (R+2)/(k^2-1)$ and $N' = 2^{R+Df} \leq 2^{2R} = 2^{2rk} \leq 2^{O(k^2 u)} \leq 2^{(\log n)^{O(1)}}$. Then, assuming NP $\not\subseteq$ BPTIME$(2^{(\log n)^{O(1)}})$, there cannot exist a polynomial time algorithm that can distinguish if there is a clique of size at least $c^D 2^R/2$ or there is no clique of size more than 2^r. This implies that the size of the largest clique in a graph with N' vertices cannot be approximated within a factor N'^{α} where

$$\alpha = \frac{\log(c^D 2^R/2) - \log 2^r}{\log N'} = \frac{D \log c + R - 1 - r}{R + Df} = 1 - \frac{Df + D\log(1/c) + r + 1}{R + Df}$$
$$\geq 1 - \frac{\left(\frac{R+2}{k^2+1}\right)(2k+1) + r + 1}{R + \left(\frac{R+2}{k^2+1}\right) 2k} \geq 1 - O(r/R) = 1 - O(1/k)$$

The fact that $N' \leq 2^{O(k^2 u)}$ implies $\log N' = O(k^4 \log^{3c_0} N) = O((\log N)^{4\beta/3 - c_0})$. For a large enough value of β, $\beta/3 - c_0 > (4\beta/3 - c_0)(1/4 - \gamma)$ which implies $\alpha \geq 1 - O(1/k) \geq 1 - (\log N')^{-1/4+\gamma}$. □

4.7 Hardness of Approximating Chromatic Number

It turns out that repeated applications of tensoring and boosting on a Min-Lin-Deletion instance improve its "coverability" (see Lemma 4.3.8) and our process for converting Min-Lin-Deletion to Min-3Lin-Deletion preserves the "coverability".

Proof of Theorem 1.4.8: We first give a reduction from 3SAT to the covering version of Min-3Lin-Deletion. Given a 3SAT formula ϕ of size n, we produce a Min-3Lin-Deletion instance $\boldsymbol{A}_1$ of size $N_1 = n^{O(1)}$ using the reduction given by Lemma 4.3.8 with the parameters $\epsilon = 10^{-10}$ and $\delta = 0.4$. We use the reduction $\mathcal{A}$ defined in Theorem 1.4.10 to convert $\boldsymbol{A}_1$ to a 7-regular Min-3Lin-Deletion instance $\boldsymbol{A}$ of size $N = 2^{O(\log^2 N_1)} = 2^{O(\log^2 n)}$. If ϕ is unsatisfiable, the analysis of Theorem 1.4.10 shows that $\boldsymbol{A}$ has no assignment satisfying more that $1 - s'$ fraction of the equations for some $s' = \Omega(\log^{-3} N)$. We now consider the case when ϕ is satisfiable. Lemma 4.3.8 guarantees that $\boldsymbol{A}_1$ is $(1 - 10^{-10})$-coverable. We now show how the various steps of reduction $\mathcal{A}$ change the covering parameter. Let $\boldsymbol{B}$ be a Min-Lin-Deletion instance and let Λ be a ρ-covering for $\boldsymbol{B}$.

- *Tensoring:* Any pair of assignments in Λ naturally translates to an assignment to the tensoring $\boldsymbol{B}'$ of $\boldsymbol{B}$. It can be verified that the set Λ' of assignments to the variables in $\boldsymbol{B}'$ that consists of these $|\Lambda|^2$ assignments is a $1 - (1 - \rho)^2$-covering for $\boldsymbol{B}'$.

- *Boosting:* If $\boldsymbol{B}'$ is the boosting of $\boldsymbol{B}$, then Λ is a $(1 - \sigma(1 - \rho))$-covering for $\boldsymbol{B}'$. This is because given any set of σ equations in $\boldsymbol{B}$, the probability that a randomly picked assignment from Λ does not satisfy at least one of them is at most $\sigma(1 - \rho)$.

- *Sum-Check Protocol and Low-degree Test:* Let $\boldsymbol{B}'$ be the output of the reduction $\mathcal{A}_3$ defined in the proof of Theorem 4.5.1 on input $\boldsymbol{B}$. We can construct $|\Lambda|$ proofs for the verifier $\mathcal{V}$ based on each assignment in Λ, which eventually gives a set Λ' of $|\Lambda|$ assignments to the variables in $\boldsymbol{B}'$. If the equation from $\boldsymbol{B}$ picked by $\mathcal{V}$ for some random string is satisfied by the assignment used to construct the proof, the $\mathcal{V}$ will accept the proof. That is, all the linear predicates of $\mathcal{V}$ are satisfied in this case. It follows from this argument that that Λ' is a ρ-covering for $\boldsymbol{B}'$.

- *Breaking equations to obtain a Min-3Lin-Deletion instance:* Let $\boldsymbol{B}'$ be the result of breaking the equations in $\boldsymbol{B}$ as stated in the proof of Theorem 1.4.10 by introducing auxiliary variables. Let x_is be the variables in $\boldsymbol{B}$, and let y_js be the auxiliary variables introduced in $\boldsymbol{B}'$. For any given assignment to the x_is, there exists an assignment to y_js such that if an equation in $\boldsymbol{B}$ is satisfied, then all the equations in the cluster corresponding to it in $\boldsymbol{B}'$ are satisfied.

This shows that if ϕ is satisfiable, then $\boldsymbol{A}$ has a $(1 - 2^{-\Omega(\sqrt{\log N})})$-covering. We now use Lemma 4.3.9 to obtain a randomized PCP for 3SAT. Let $\gamma > 0$ be any constant. Let $\beta > 0$ be a constant that will be fixed to a large enough value later. As in the proof of Theorem 1.4.7, set $\epsilon = \log^{-\beta} N$, $u = \frac{1}{2}\sqrt[3]{\frac{5}{\epsilon^2 \log(1 - s'^{c_0})^{-1}}}$ and $k = \sqrt[3]{\frac{\log(1 - s'^{c_0})^{-1}}{5\epsilon}}$. This gives a randomized PCP for 3SAT with the following parameters:

- The constant β can be made large enough so that $r = O(ku) = O(\epsilon^{-1}) = O(\log^{\beta} N) = (\log n)^{O(1)}$. The number of free bits is $f = 2k$.
- If ϕ is satisfiable, then it has a $2^{-(2k+1)}$-covering proof since $1 - \epsilon ku \geq 1/2$.
- Let $\delta = 2^{-k^2}$. Then $(1 - s'^{c_0})^u \leq \delta^2$, which implies the soundness s of the verifier is at most 2.2^{-k^2}.

Let $\rho = 2^{-(2k+1)}$. As the final step, we use Lemma 4.3.7 for this (r, f, ρ, s) randomized PCP for 3SAT with $h = u$ to obtain a reduction from 3SAT to chromatic number. When given a 3SAT formula ϕ of size n, the reduction produces a graph G' of size $N' = (2^f/s)^h = 2^{h(2k+k^2-1)} = 2^{O(k^2 u)} = 2^{(\log n)^{O(1)}}$ such that:

- If ϕ is satisfiable, $\chi(G') \leq \frac{2 \ln N'}{\rho^h}$.
- If ϕ is unsatisfiable, with probability $1/2$, $\chi(G') \geq \frac{N'}{h 2^{r+f}}$, where $\chi(G')$ denotes the chromatic number of the graph G'.

Therefore, assuming NP $\not\subseteq$ coRTIME$(2^{(\log n)^{O(1)}})$ or equivalently NP $\not\subseteq$ ZPTIME$(2^{(\log n)^{O(1)}})$, no randomized polynomial time algorithm can approximate chromatic number in a graph

with N' vertices better than N'^{α} where

$$\alpha = \frac{\log(N'\rho^h) - \log(h2^{r+f}2\ln N')}{\log N'} = 1 - \frac{h\log(1/\rho) + \log h + r + f + 1 + \log(\ln N')}{\log N'}$$
$$\geq 1 - O\left(\frac{hk}{h(2k+k^2-1)}\right) \geq 1 - O(1/k)$$

where the first inequality follows since $\log(1/\rho) = O(k)$, $r = O(ku) = O(hk)$ and $f = O(k)$. Once again as in Theorem 1.4.7, we can make β large enough so that $\beta/3 - c_0 > (4\beta/3 - c_0)(1/4 - \gamma)$ which implies $\alpha \geq 1 - O(1/k) \geq 1 - (\log N')^{-1/4+\gamma}$. □

4.8 Conclusions

Recently, Samorodnitsky and Trevisan [98] showed that, assuming the Unique Games Conjecture of Khot [71], it is hard to approximate Max-Clique in degree d graphs better than $d/\text{polylog}(d)$. This suggests that Max-Clique on general graphs could be hard to approximate within $n/\text{polylog}(n)$. We think it is a challenging (and important) open problem to prove such a hardness result, or even to improve the hardness result to $\frac{n}{2^{O(\sqrt{\log n})}}$.

CHAPTER V

MONOTONE MULTILINEAR BOOLEAN CIRCUITS FOR BIPARTITE PERFECT MATCHING REQUIRE EXPONENTIAL SIZE

5.1 Introduction

Let BPM (PM) denote the problem of finding whether a bipartite (general) graph has a perfect matching. In this chapter, we show that under two different restrictions on the function calculated by AND gates and OR gates, monotone circuits for BPM require exponential size.

As defined by Nisan and Wigderson [89], an arithmetic circuit is multilinear if at each gate the power of any variable in minimal representation of the polynomial computed is at most 1. Equivalently, for any product gate, the minimal representation of the polynomials of its two input gates have no variable in common. A recent result of Raz [93] shows a super-polynomial lower bound on multilinear arithmetic formulas for the permanent. Multilinearity for arithmetic circuits has been extensively studied due to the lack of strong lower bounds for general arithmetic circuits and because they seem to be the most intuitive circuits for multilinear functions [89, 93].

We already defined multlinearity for Boolean circuits (Definition 1.4.11). To the best of our knowledge, our lower bounds are not implied by any of the known lower bounds for arithmetic and Boolean circuits. When a multilinear Boolean circuit for BPM is converted to an arithmetic circuit in the natural way (by replacing the AND and OR gates by product and plus gates), it does not necessarily yield a multilinear arithmetic circuit for permanent because Boolean circuits can use idempotence.

In what follows, we will assume that all circuits are monotone Boolean circuits in which the AND and OR gates have fan-in 2 (but the fan-out can be unbounded). The size of a circuit is the number of gates in it. The inputs to the circuit correspond to potential edges of a graph G on a set $V = \{1, 2, \ldots n\}$ of n vertices, where n is even. So in the

BPM problem there is an input for each pair $\{i, j\}$ in BPM where $i \in \{1, 2, \ldots, n/2\}$ and $j \in \{n/2 + 1, n/2 + 2, \ldots, n\}$. In PM we have a input for all pairs $\{i, j\}$. The input corresponding to edge $\{i, j\}$ is 1 if the edge is present in G and 0 otherwise.

5.2 Upper Bounds on Depth and Size of Monotone Boolean Circuits for PM

Let S be an even sized subset of V. A subset m of edges is said to be an *S-matching* if m is a matching with an edge incident on each vertex of S and no edge in m has one end point in S and the other in $\bar{S} = V - S$ (m may contain edges that have both end-points in $\bar{S}$). We say that m is an *exact S-matching* if m is an S-matching and m has no edge incident on a vertex in $V - S$.

We first describe the depth upper bound for PM.

Lemma 5.2.1 *PM has monotone circuits of $O(n)$ depth.*

Proof: We first give the construction when $n = 2^k$ for some number k. If $n = 2$, the construction is trivial. So assume $n > 2$. Suppose for each $S \subseteq V, |S| = n/2$, we are given a circuit C_S that evaluates to 1 iff there is a S-matching in the input graph G. If we take the AND of C_S and $C_{\bar{S}}$, we get a circuit C_P that evaluates to 1 iff there is a perfect matching that does not cross the partition $P = \{S, \bar{S}\}$. If we take the OR of all circuits corresponding to the partitions of V into two sets of $n/2$ vertices, we get a circuit C_V for perfect matching on V (see Figure 3). Since the number of partitions of V into $n/2$ vertices is $\frac{n!}{2!(n/2)!^2}$, the depth of the OR gates at the output of C_V is $\lfloor \log n! - \log 2! - 2 \cdot \log(n/2)! \rfloor \leq d(n)$ for some function $d(n) = O(n)$. Therefore the depth of C_V is $d(n) + \max_S \texttt{depth}(C_S)$ where $\texttt{depth}(C)$ is the depth of circuit C. The C_S may be recursively constructed the same way as C_V since C_S is a circuit for perfect matching on the graph induced by S. Therefore the depth of $C \leq d(n) + d(n/2) + \ldots = O(n)$. If n were not a power of 2, at each level of recursion, split the set of vertices into two sets of even sizes in the most balanced way. For example if $n = 36$, we need to consider all subsets of size 18 at the first level, all subsets of size 8 and 10 at the second level, all subsets of size 4 and 6 at the third level and so on. We still get that $d(C_S) = O(|S|)$ and gives $\texttt{depth}(C_V) = O(n)$. □

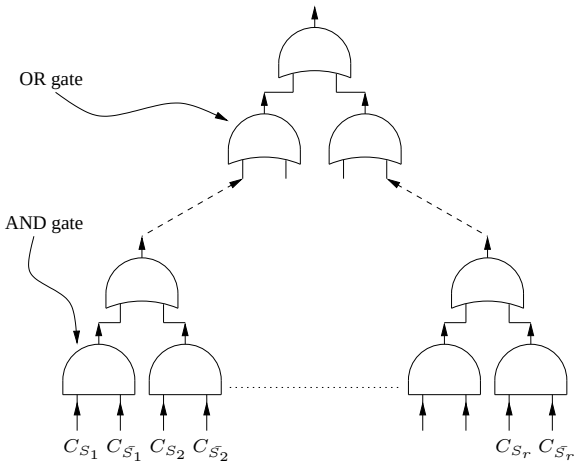

Figure 3: Construction of circuit C_V for showing the depth upper bound.

The upper bound on size for PM obtained above is $n^{O(1)}2^{3n/2}$ as the level of recursion that contains the maximum number of gates is the second level with $O(\binom{n}{n/2}\binom{n/2}{n/4})$ gates. Since finding if there is a perfect matching in a bipartite graph on n vertices is the same as checking whether the permanent of the $n/2 \times n/2$ incidence matrix is not zero, we can obtain a upper bound of $n^{O(1)}2^{n/2}$ on size for BPM by replacing the product and sum gates in the arithmetic circuit for permanent given in Jerrum and Snir [59]. The same approach can be generalized to obtain a better size upper bound for PM as follows. In the construction in Lemma 5.2.1, instead of considering partitions $\{S, \bar{S}\}$ into the most balanced even sets, consider all partitions such that S is a set of two vertices, one of which is the vertex of smallest index in V. Now the circuit for each $\bar{S}$ can be constructed recursively. At level 2, we need at most $\binom{n-1}{n-2}$ subcircuits to compute the different possible values for $\bar{S}$. In general, at level i, we need at most $\binom{n-i}{n-2i}$ subcircuits. Solving for the value for i that maximizes $\binom{n-i}{n-2i}$, we get that the circuit constructed has size $O(2^{0.695n})$.

5.3 Lower Bound on Size of Restricted Monotone Circuits for BPM

Throughout this section, we assume that the inputs to a circuit correspond to the edges of a bipartite graph G on a bipartition $\{V', \bar{V}'\}$ of V into equal sets. We use the notion of *minterm* from Karchmer and Wigderson [63] to analyze the circuits.

Definition 5.3.1 *A* minterm *of a monotone Boolean function is a minimal set of variables*

that when set to 1*, the function evaluates to* 1 *irrespective of the value of the other variables. Let* C *be a circuit and let* g *be a gate in it. The function computed by* g *is defined as the Boolean function representing the output of* g *in terms of the input gates. The set of minterms of* g *in circuit* C*, denoted by* $minterm_C(g)$ *(or* $minterm(g)$ *if the circuit is clear from the context) is the set of minterms of the function computed by* g*.*

The edge set *of gate* g *is the set of all edges that appear in some minterm of* g*. The* vertex set *of a subset* m *of the edges is the set of all end points of the edges in* m*. The* vertex set *of* g *is the vertex set of its edge set. The edge set and vertex set of a subcircuit of* C *are defined to be the respective values for the output gate of the subcircuit.*

We will use the following easy to verify facts about minterms (below g is a gate with input gates g_1 and g_2):

- If g is an OR gate and m is a minterm of g, then m is a minterm of either g_1 or g_2.
- If g is an OR gate and m_1 is a minterm of g_1, then there exists a minterm m of g such that $m \subseteq m_1$.
- If g is an AND gate and m is a minterm of g, then there exist a minterm m_1 of g_1 and minterm m_2 of g_2 such that $m = m_1 \bigcup m_2$.
- If g is an AND gate and m_1 and m_2 are minterms of g_1 and g_2 respectively, then there exists a minterm m of g such that $m \subseteq m_1 \bigcup m_2$.

5.3.1 Lower Bound for Circuits in Simple Form

Definition 5.3.2 *A circuit is said to be in* simple form *if for any OR gate* g *in the circuit with input gates* g_1 *and* g_2*, the vertex sets of* g*,* g_1 *and* g_2 *are the same.*

It can be seen by induction that in a circuit in simple form, the vertex set of any minterm of a gate is the same as the vertex set of the gate. The statement is true for input gates of the circuit. If it is true for the input gates g_1 and g_2 of an OR gate g, it is true for g as well since minterms of g are minterms of either g_1 or g_2. If the statement is true for the input gates g_1 and g_2 of an AND gate g, all the minterms of g have the same vertex set since any

minterm of g is the union of some minterm of g_1 with some minterm of g_2 and all minterms of g_1 have the same vertex set, as do minterms of g_2.

Lemma 5.3.3 *Assume $n > 2$. Let C be a monotone circuit for BPM in simple form. For any perfect matching m, $\exists V_m \subseteq V$, $|V|/3 \leq |V_m| \leq 2|V|/3$ such that V_m is the vertex set of some gate g and m is a V_m-matching.*

Proof: Set $U = V$, $m' = m$ and let g be the output gate of C. At any stage later, we will ensure that (U, m', g) satisfy the following constraints:

(1) U is the vertex set of g.

(2) m' is a subset of m, m' is a minterm of g and U is the vertex set of m' (and hence m is a U-matching).

Also, let $U > 2|V|/3$, which is true initially when $U = V$. Since $|V| = n > 2$, the gate g can not be an input gate.

- If g is an OR gate, one of the two input gates g_1 or g_2 of g, say g_1, has m' as a minterm. Set $g \leftarrow g'$ and repeat.
- If g is an AND gate, let V_1 and V_2 be the vertex sets of its input gates g_1 and g_2 respectively. Without loss of generality, let $|V_1| \geq |V_2|$. There must be some minterm m_1 of g_1 and m_2 of g_2 such that $m' = m_1 \bigcup m_2$. From a remark above the lemma, we know that the vertex sets of m_1 and m_2 are V_1 and V_2 respectively, and hence $U = V_1 \bigcup V_2$. Therefore (V_1, m_1, g_1) satisfy conditions (1) and (2).
  - If $|V|/3 \leq |V_1| \leq 2|V|/3$, then V_1 is the required value for V_m.
  - If $|V_1| > 2|V|/3$, set $g \leftarrow g_1$ and $U \leftarrow V_1$ and repeat.
  - The case $|V_1| < |V|/3$ is not possible since otherwise $|V_1| + |V_2| \leq 2|V|/3$, contradicting $|U| > 2|V|/3$.

Since C has a finite depth, and input gates have vertex sets of size 2, we will successfully find a value for V_m. □

The problem with proving the above lemma for general monotone circuits for BPM is that for a particular gate, there can be two minterms with different vertex sets. So we can not associate a set of vertices with each gate such that all its minterms are exact matchings for the set.

Lemma 5.3.4 *Monotone circuits for BPM in the simple form require* $\Omega(2^{.459n})$ *size.*

Proof: Let $n > 2$ and let C be a monotone circuit in simple form for BPM. Enumerate one set V_m satisfying the conditions of Lemma 5.3.3 for each possible perfect matching m on the bipartition $\{V', \bar{V}'\}$. For each $U \subseteq V$ of size p, there are at most $(p/2)!\frac{n-p}{2}!$ perfect matchings that do not cross it (assuming G is a complete bipartite graph, this number is exactly $(p/2)!\frac{n-p}{2}!$ if U contains the same number of vertices from V' and $\bar{V}'$ and zero otherwise). Therefore, each $U \subseteq V$ such that $|V|/3 \leq |U| \leq 2|V|/3$ corresponds to at most $(n/6)!(n/3)!$ perfect matchings. Since the number of perfect matchings in a complete bipartite graph is $(n/2)!$, we must have enumerated

$$\frac{(n/2)!}{(n/6)!(n/3)!} = n^{\Omega(1)}2^{(n/2\log 3 - n/3)} = \Omega(2^{.459n})$$

distinct subsets of V, each of them corresponding to a different gate of C. Therefore, the number of gates in C is $\Omega(2^{.459n})$. □

5.3.2 Lower Bound for Multilinear Circuits

An equivalent way of stating Definition 1.4.11 for circuits for BPM is that a circuit is multilinear if the edge set of any AND gate is the disjoint union of the edge set of its input gates. Unlike circuits in simple form, multilinear circuits are expressive enough to compute any monotone Boolean function.

Definition 5.3.5 *A circuit is said to be in the* simplest form *if it is in simple form and the vertex set of any AND gate is the disjoint union of the vertex set of its input gates.*

It can be seen that the circuits used to show the upper bounds on depth and size in Section 5.2 are all in the simplest form. Also note that circuits in simplest form are also multilinear.

Theorem 5.3.6 *A multilinear circuit of smallest possible size for BPM is also in the simplest form.*

Idea: Let C be a multilinear circuit for bipartite perfect matching on $\{V', \bar{V}'\}$ of smallest possible size. We will define a *required vertex set* V_g for each gate g such that (g, V_g) satisfies the following:

(1) If g is an AND gate with input gates g_1 and g_2, the required vertex of g is the disjoint union of the required vertex sets of g_1 and g_2.

(2) If g is an OR gate with input gates g_1 and g_2, the required vertex sets of g, g_1 and g_2 are the same.

(3) There is at least one minterm of g that is an exact V_g-matching.

(4) All minterms of g have an edge to each vertex in V_g.

Definition 5.3.7 *For two functions f and h, we say h is a pruning of f with respect to $V_0 \subseteq V$ if every minterm of h is the superset of some minterm of f and every minterm of f that is an exact V_0-matching is also a minterm of h.*

Note that pruning f multiple times with respect to the same set V_0 results in a pruning of f with respect to V_0. Continuing with the properties of (g, V_g):

(5) Let g be a gate with an input g_1. Let the output from g_1 to g be replaced by the output of a new subcircuit that computes a function f (see Figure 4). Assume that f is a pruning of the output of g with respect to V_{g_1}. Then the new output of g is a pruned function of the original output of g.

Intuitively, if an input gate g_1 of an OR gate g has no minterm that is an exact V_g-matching, we can replace the output from g_1 to g with a zero input without affecting the function calculated by the circuit. For gate g, the effect of this change is to increase the size of some minterms while some minterms drop out. But all minterms that are exact V_g-matchings are unaffected as they must have been minterms of the other input gate to

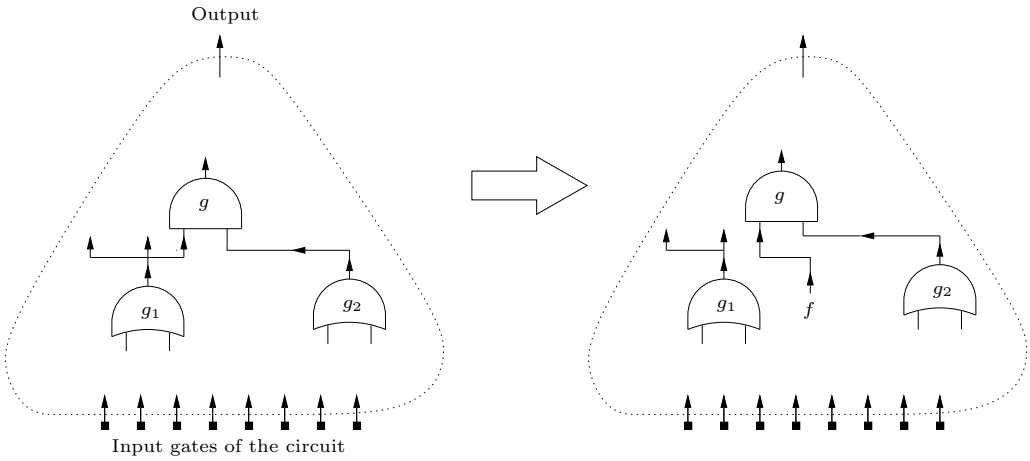

Figure 4: Pruning the output from gate g_1 to gate g.

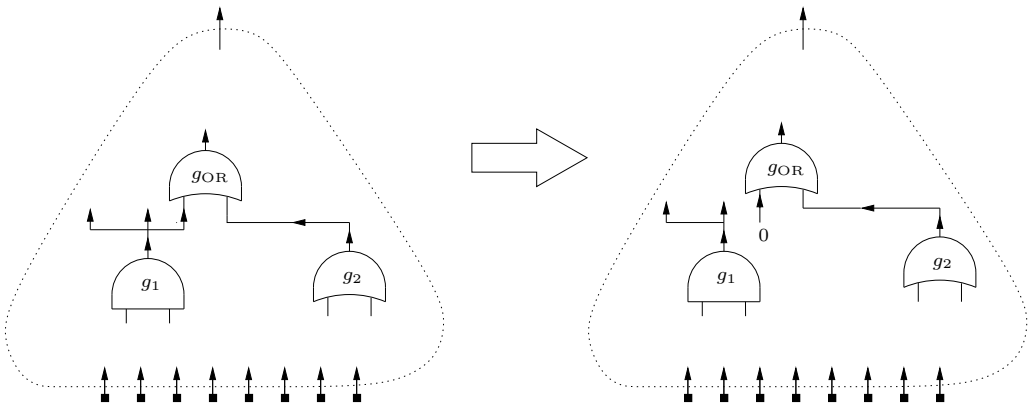

Figure 5: Disconnecting the output of gate g_1 to gate g in Lemma 5.3.8.

g. This pruning effect cascades all the way to the output gate. But the function calculated by the output gate will remain the same as all its minterms will be exact matchings on its required vertex set, namely V. We first formalize the intuition in the following lemma.

Lemma 5.3.8 *Let C be a circuit for BPM. Let g_{OR} be an OR gate in circuit C with children g_1 and g_2. Assume there exists a set V_g defined for each gate g on the path from g_{OR} to the output (g_{OR} inclusive) such that (g, V_g) satisfies (5). If g_1 does not have any minterm that is an exact $V_{g_{\mathrm{OR}}}$-matching, then replacing the input from g_1 to g_{OR} with a zero input (as shown in Figure 5) does not change the function calculated by C.*

Proof: If $m \in minterm_C(g_{\mathrm{OR}})$ is an exact g_{OR}-matching, then $m \notin minterm_C(g_1)$. Therefore $m \in minterm_C(g_2)$. This implies $m \in minterm_{C'}(g_{\mathrm{OR}})$. If $m \in minterm_{C'}(g_{\mathrm{OR}})$, then

$m \in minterm_C(g_2)$. Therefore, $\exists m' \in minterm_C(g_{\text{OR}})$ such that $m' \subseteq m$. That is, the output of g_{OR} in C' is a pruned function of the output of g_{OR} in C. Therefore, applying property (5) to gate g_{OR} and subsequently to all the gates along the paths to the output gate, we see that the outputs of circuits C and C' are the same. □

We will apply a series of such pruning changes. One subtle point is that when pruning is applied, it may violate multilinearity for AND gates (only) on the path between the gate pruned and the output gate. But we get back multilinearity once all possible prunings are performed since all minterms that are not exact V_g-matchings for a gate g will drop out (hence the name pruning).

In the proof below, we might modify C by removing the connection between some gates. If at any time, there is no path from a gate to the output gate, we assume that the gate has been deleted without loss of generality (since such a gate can no longer affect the output of the circuit).

Proof of Theorem 5.3.6: We first define the required vertex set in a top-down fashion. The required vertex set of the output gate is defined to be its vertex set V. It can be seen that it satisfies conditions (3) and (4). We define the required vertex set of a gate based on the required vertex sets of its parents (gates to which it supplies an input) in C. For this purpose, once the required vertex set of a gate g is defined it passes a *requirement*, a subset of V, to each of its children g_1 and g_2 as defined below. We will not alter the subcircuit rooted at a gate g before we define its required vertex set and pass the requirements to its children. We will finally show that if C is in fact the smallest multilinear circuit, then we could not have altered the circuit by pruning. Suppose the required vertex of a gate g has been defined.

- **Case 1:** g is an AND gate: By property (3), $\exists m \in minterm(g)$ such that m is an exact V_g-matching. Therefore, $\exists m_1 \in minterm(g_1)$ and $\exists m_2 \in minterm(g_2)$ such that $m = m_1 \bigcup m_2$ and $m_1 \bigcap m_2 = \emptyset$ (since the edge sets of g_1 and g_2 are disjoint. This is because of the multilinearity of the original circuit and the fact that the subcircuit rooted at g has not yet been modified). The requirements passed to g_1 and g_2 are the

vertex sets of m_1 and m_2, say V_1 and V_2, respectively. We now show some properties of (g_1, V_1) (analogous properties hold for (g_2, V_2)).

- Since V_1 is the vertex set of m_1, m_1 is an exact V_1-matching.
- Suppose some $m' \in minterm(g_1)$ does not have an edge to some $v \in V_1$. Then some subset m'' of $m' \bigcup m_2$ is a minterm of g. Also, m_2 does not have an edge to v since $m_1 \bigcup m_2$ was a $V_1 \bigcup V_2$-matching and m_1 has an edge to v. But then m'' does not have an edge to a vertex v in the required vertex set of g, contradicting (4) for (g, V_g). Therefore, all $m' \in minterm(g_1)$ have an edge to each vertex of V_1.
- This also means that any $m' \in minterm(g)$ which is an exact V_g-matching is produced by the (disjoint) union of $m'_1 \in minterm(g_1)$ and $m'_2 \in minterm(g_2)$ where m'_1 and m'_2 are exact V_1-matching and exact V_2-matching respectively (Since $m' = m'_1 \bigcup m'_2$ is an exact V_g-matching, if m'_1 is not an exact V_1-matching, then $\exists e_1 \in m'_1$ such that e_1 has an endpoint $v_2 \in V_2$. But m'_2 has some edge e_2 incident on v_2. For $m'_1 \bigcup m'_2$ to be a matching, e_1 and e_2 must be the same edge. But this contradicts the multilinearity of the original circuit).
- We will now show that pruning the output from g_1 to g with respect to V_1 prunes the output of g with respect to V_g. Suppose we replace the output from g_1 to g with the output from a new subcircuit C_1 having output gate g'_1 to obtain a new circuit C'. Let the output of g'_1 be a pruned function (with respect to V_1) of the output of g_1. Then if $m \in minterm_{C'}(g)$, then $\exists m_1 \in minterm_{C'}(g'_1)$ and $m_2 \in minterm_{C'}(g_2) = minterm_C(g_2)$ such that $m = m_1 \bigcup m_2$. Since m_1 is the superset of some minterm of g_1 in C, m is the superset of some minterm of g in C. Also, if $m \in minterm_C(g)$ and m is an exact V_g-matching, then m is in $minterm_{C'}(g)$ too (this follows from the previous property). That is, the output of g in C' is a pruned function of that in C.

- **Case 2:** g is an OR gate: If g_1 does not have any minterm that is an exact V_g-matching, then we can modify C using Lemma 5.3.8; this will disconnect g_1 from g.

Gate g passes V_g as the requirement to g_1 if it was not disconnected. We do the same for g_2 also. For now, assume that we don't disconnect g_1 from g. We will show some properties for (g_1, V_g) (analogous properties hold for (g_2, V_g) if g_2 was not disconnected from g). If g_2 were disconnected from g, treat g_2 below to be the zero-gate.

- Obviously, there is a minterm of g_1 that is an exact V_g matching, say m_1 (otherwise we would have disconnected g_1 from g).
- Since every minterm of g_1 is the superset of some minterm of g, all minterms of g_1 have an edge to every vertex in V_g.
- We will now show that if V_g were defined to be the required vertex set of g_1, then pruning the output from g_1 to g with respect to V_g prunes the output of g with respect to V_g. Suppose we replace the input to g from g_1 with the output from a new subcircuit C_1 having output gate g_1' to obtain a new circuit C'. Let the output of g_1' be a pruned function (with respect to V_g) of the output of g_1. Then if $m \in minterm_{C'}(g)$, it is a superset of some minterm of g in C since any minterm of g_1' in C' is a superset of some minterm of g_1 in C and the set of minterms of g_2 is the same in C and C'. If $m \in minterm_C(g)$ is an exact V_g-matching, then this is still a minterm in C' since all minterms of g_1 (respectively, g_2) in C that are exact V_g-matchings are minterms of g_1' (respectively, g_2) in C' and the other minterms either did not shrink or they dropped out.

We have shown that for any requirement V_{g_1} passed to a gate g_1, (g_1, V_{g_1}) satisfies (3) and (4). But this must mean that all requirements passed to g_1 from its parents must be the same. We can then define this to be the required vertex set of g_1.

We will now show that the required vertex set of a gate is in fact also the vertex set of the gate. Let g be a gate whose required vertex set is not the same as its vertex set. Therefore $\exists m \in minterm(g)$ with an edge e to a vertex outside its required vertex set. If g is an OR gate, one of its two input gates, say g', has m as a minterm. Since the required vertex sets of g and g' are the same, g' too has an edge to a vertex v outside its required vertex set. If g is an AND gate with inputs g_1 and g_2, then $\exists m_1 \in minterm(g_1)$ and $m_2 \in minterm(g_2)$

such that $m_1 \bigcup m_2 = m$. Let $e \in m_1$. Then g_1 has a minterm m_1 that has an edge to a vertex v outside its required vertex set (since the requirement passed by an AND gate to its input gate is a subset of its required vertex set). Since C has finite depth, we reach an input gate of the circuit whose required vertex set is not the same as its vertex set. But this is a contradiction of property (3).

Therefore, from conditions (1) and (2), circuit C is now in simplest form. But if we removed the connection between an OR gate and its child using Lemma 5.3.8, we would have decreased the size of the circuit (because we can eliminate the OR gate as explained before the proof). This would contradict the assumption that the original circuit was a multilinear circuit of smallest size for BPM. Hence a multilinear circuit for BPM of the smallest size is also in the simplest form. □

Proof of Theorem 1.4.12: This is easily seen from Lemma 5.3.4 and Theorem 5.3.6, since circuits in simplest form are also in simple form. □

5.4 Conclusions

We gave an analogue of Raz's [93] lower bound for arithmetic formulas computing the permanent in the Boolean setting. The class of circuits in simplest form that attain the lower bound for multilinear circuits for BPM also happen to achieve all the upper bounds we mentioned in Section 5.2. The upper bound on size mentioned in Section 5.2 and the lower bound shown in Lemma 5.3.4 for BPM are very close showing that the analysis of the lower bound for monotone multilinear Boolean circuits is quite tight.

As justified above, circuits in simplest form seem to be most natural circuits for PM and BPM. An interesting open problem is to try and generalize our result to show that circuits in simplest form are also the smallest (or close to the smallest) circuits for other kinds of monotone circuits for BPM. Another interesting problem is to construct circuits not in the simplest form that beat the upper bounds shown in Section 5.2.

APPENDIX A

SOME BASIC NOTATION

The vertex cover of a graph is a subset of the vertices such that each edge of the graph has at least one endpoint in the susbset.

A graph is said to be d-regular if the degree of every vertex is d.

A 3SAT instance is a conjunction of disjunctions where each disjunction is over exactly three variables. The 3SAT problem is to decide whether a given 3SAT instance is satisfiable.

poly(n) (or simply poly$n$) is defined to be the set of all finite polynomials on n. Also, $\tilde{O}(f(n)) = O(f(n)\text{poly}\log f(n))$.

We denote by $[n]$ the set $\{1, 2, \ldots, n\}$.

$\mathbb{Q}$ denotes the set of rational numbers. $\mathbb{R}$ denotes the set of real numbers.

We denote the complement of a set S by $\bar{S}$.

A.1 Complexity Classes

We give a brief description of the complexity classes that we used. Below, n is used to denote the size of the input to an algorithm.

- P: The class of languages that have deterministic polynomial time algorithms.
- NP: The class of languages that have a polynomial time "verifier". The verifier takes takes a pair of strings (x, y) as input and satisfies the following:
  - If x is in the language, there exists a "proof" y satisfying $|y| \leq p(|x|)$ that causes the verifier to accept (x, y), where p is a polynomial that depends on the verifier.
  - If x is not in the language, (x, y) is rejected for all y satisfying $|y| \leq p(|x|)$.
- DTIME($f(n)$): The class of languages that have deterministic algorithms with running time $f(n)$.

- BPTIME($f(n)$): The class of languages that have algorithms that make (two-sided) error with probability at most 1/3 and have running time $f(n)$. BPP is defined as BPTIME($n^{O(1)}$).

- ZPTIME($f(n)$): The class of languages that have randomized algorithms that have expected running time $f(n)$ and make no error. ZPP is defined as ZPTIME($n^{O(1)}$).

REFERENCES

[1] Agmon, S., "The relaxation method for linear inequalities," *Canadian J. of Mathematics*, vol. 6(3), pp. 382–392, 1964.

[2] Ajtai, M. and Dwork, C., "A public-key cryptosystem with worst-case/average-case equivalence," in *Proc. 29th ACM Symposium on the Theory of Computing*, pp. 284–293, 1997.

[3] Alekhnovich, M., Braverman, M., Feldman, V., Klivans, A., and Pitassi, T., "Learnability and automizability," in *Proceeding of FOCS*, pp. 621–630, 2004.

[4] Amaldi, E., "From finding feasible susbsystems of linear systems to feedforward neural network design," *Ph.D Thesis, Swiss Federal Institute of Technology at Lausanne (EPFL)*, 1994.

[5] Amaldi, E. and Kann, V., "The complexity and approximability of finding maximum feasible subsystems of linear relations.," *Theoretical Computer Science*, vol. 147, no. 1&2, pp. 181–210, 1995.

[6] Angluin, D. and Laird, P., "Learning from noisy examples," *Machine Learning*, vol. 2, pp. 343–370, 1988.

[7] Arora, S., *Probabilistic checking of proofs and the hardness of approximation problems*. Ph.D. thesis, UC Berkeley, 1994.

[8] Arora, S., Babai, L., Stern, J., and Sweedyk, Z., "The hardness of approximate optima in lattices, codes, and systems of linear equations," *Journal of Computer and System Sciences*, vol. 54, no. 2, pp. 317–331, 1997.

[9] Arora, S. and Lund, C., *Approximation Algorithms for NP-hard Problems, editor : D. Hochbaum*. PWS Publishing, 1996.

[10] Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M., "Proof verification and the hardness of approximation problems," *Journal of the ACM*, vol. 45, no. 3, pp. 501–555, 1998.

[11] Arora, S. and Safra, S., "Probabilistic checking of proofs : A new characterization of NP," *Journal of the ACM*, vol. 45, no. 1, pp. 70–122, 1998.

[12] Bellare, M., Goldreich, O., and Sudan, M., "Free bits, PCPs and non-approximability," *Electronic Colloquium on Computational Complexity, Technical Report TR95-024*, 1995.

[13] Bellare, M., Goldwasser, S., Lund, C., and Russell, A., "Efficient probabilistically checkable proofs and applications to approximations," in *Proc. 25th ACM Symposium on Theory of Computing*, pp. 294–304, 1993.

[14] Bellare, M. and Sudan, M., "Improved non-approximability results," in *Proc. 26th ACM Symposium on Theory of Computing*, pp. 184–193, 1994.

[15] Ben-David, S., Eiron, N., and Long, P. M., "On the difficulty of approximately maximizing agreements.," *Journal of Computer and System Sciences*, vol. 66, no. 3, pp. 496–514, 2003.

[16] Berlekamp, E., McEliece, R., and van Tilborg, H., "On the inherent intractability of certain coding problems," *IEEE Transactions on Information Theory*, vol. 24, 1978.

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

[18] Blum, A., "Open problem: Learning a function of r relevant variables.," in *Proceeding of COLT*, pp. 731–733, 2003.

[19] Blum, A., Frieze, A., Kannan, R., and Vempala, S., "A polynomial time algorithm for learning noisy linear threshold functions," *Algorithmica*, vol. 22, no. 1/2, pp. 35–52, 1997.

[20] Blum, A., Furst, M., Kearns, M., and Lipton, R. J., "Cryptographic primitives based on hard learning problems," in *Proceedings of International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pp. 278–291, 1993.

[21] Blum, A., Kalai, A., and Wasserman, H., "Noise-tolerant learning, the parity problem, and the statistical query model," *Journal of the ACM*, vol. 50(4), pp. 506–519, 2003.

[22] Blum, A. and Kannan, R., "Learning an intersection of a constant number of halfspaces under a uniform distribution," *Journal of Computer and System Sciences*, vol. 54, no. 2, pp. 371–380, 1997.

[23] Blum, A. and Langley, P., "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 245–271, 1997.

[24] Blum, A. L., *Algorithms for approximate graph coloring*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1992.

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

[26] Bshouty, N. and Burroughs, L., "Maximizing agreements and coagnostic learning," *Theoretical Computer Science*, vol. 350, no. 1, pp. 24–39, 2006.

[27] Bshouty, N. and Feldman, V., "On using extended statistical queries to avoid membership queries," *Journal of Machine Learning Research*, vol. 2, pp. 359–395, 2002.

[28] Bshouty, N., Jackson, J., and Tamon, C., "More efficient PAC-learning of DNF with membership queries under the uniform distribution.," *Journal of Computer and System Sciences*, vol. 68, no. 1, pp. 205–234, 2004.

[29] Cohen, E., "Learning noisy perceptrons by a perceptron in polynomial time," in *Proc. 38th IEEE Symp. on Foundations of Computer Science*, pp. 514–523, 1997.

[30] Dumer, I., Micciancio, D., and Sudan, M., "Hardness of approximating the minimum distance of a linear code," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 22–37, 2003.

[31] Engebretsen, L. and Holmerin, J., "Towards optimal lower bounds for clique and chromatic number," *Electronic Colloquium on Computational Complexity (ECCC)*, no. TR01-003, 2001.

[32] Feige, U., "Randomized graph products, chromatic numbers, and the Lovász θ-function," in *Proc. 27th ACM Symposium on Theory of Computing*, pp. 635–640, 1995.

[33] Feige, U., Goldwasser, S., Lovász, L., Safra, S., and Szegedy, M., "Interactive proofs and the hardness of approximating cliques," *Journal of the ACM*, vol. 43, no. 2, pp. 268–292, 1996.

[34] Feige, U. and Kilian, J., "Two prover protocols: low error at affordable rates," in *Proc. 26th ACM Symposium on Theory of Computing*, pp. 172–183, 1994.

[35] Feige, U. and Kilian, J., "Zero knowledge and the chromatic number," in *Proceedings of Conference on Computational Complexity (CCC-96)*, pp. 278–289, May 24–27 1996.

[36] Feige, U., "Approximating maximum clique by removing subgraphs," *SIAM J. Discret. Math.*, vol. 18, no. 2, pp. 219–225, 2005.

[37] Feige, U. and Kogan, S., "Hardness of approximation of the balanced complete bipartite subgraph problem," Tech. Rep. MCS04-04, The Weizmann Institute of Science, 2004.

[38] Feldman, V., "Attribute Efficient and Non-adaptive Learning of Parities and DNF Expressions," *Journal of Machine Learning Research*, no. 8, pp. 1431–1460, 2007.

[39] Feldman, V., Gopalan, P., Khot, S., and Ponuswami, A., "New Results for Learning Noisy Parities and Halfspaces," in *Proceedings of FOCS*, pp. 563–574, 2006.

[40] Feldman, V., Gopalan, P., Khot, S., and Ponuswami, A., "On agnostic learning of parities, monomials and halfspaces," 2007. To appear in SIAM Journal on Computing. Available at `http://www.cc.gatech.edu/~pashok/publications/hs-parity-monomial_SIAM.p%df`.

[41] Freund, Y., "An improved boosting algorithm and its implications on learning complexity," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 391–398, 1992.

[42] Freund, Y., "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121, no. 2, pp. 256–285, 1995.

[43] Galant, S., "Perceptron based learning algorithms," *IEEE Trans. on Neural Networks*, vol. 1(2), 1990.

[44] Garey, M. and Johnson, D. S., *Computers and Intractability.* W. H. Freeman, San Francisco, 1979.

[45] Goldreich, O., Goldwasser, S., and Halevi, S., "Eliminating decryption errors in the ajtai-dwork cryptosystem," in *Advances in Cryptology, Proc. of Crypto '97, LNCS, Springer Verlag.*, 1997.

[46] Goldreich, O. and Levin, L., "A hard-core predicate for all one-way functions," in *Proceedings of STOC*, pp. 25–32, 1989.

[47] Guruswami, V. and Raghavendra, P., "Hardness of Learning Halfspaces with Noise," in *Proceedings of FOCS*, pp. 543–552, 2006.

[48] Guruswami, V. and Raghavendra, P., "A 3-query pcp over integers," in *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pp. 198–206, 2007.

[49] Hajnal, A., Maass, W., Pudlak, P., Szegedy, M., and Turan, G., "Threshold circuits of bounded depth," *Journal of Computer and System Sciences*, vol. 46, pp. 129–154, 1993.

[50] Hastad, J., "Clique is hard to approximate within $n^{1-\epsilon}$," in *Proc. 37th IEEE Symposium on Foundations of Computer Science*, pp. 627–636, 1996.

[51] Håstad, J., "Testing of the long code and hardness for clique," in *Proc. 28th ACM Symposium on Theory of Computing*, pp. 11–19, 1996.

[52] Hastad, J., "Some optimal inapproximability results," *Journal of the ACM*, vol. 48, no. 4, pp. 798–859, 2001.

[53] Hastad, J. and Wigderson, A., "Simple analysis of graph tests for linearity and PCP," in *Proc. 16th IEEE Conference on Computational Complexity*, 2001.

[54] Haussler, D., "Decision theoretic generalizations of the PAC model for neural net and other learning applications," *Information and Computation*, vol. 100, no. 1, pp. 78–150, 1992.

[55] Helmbold, D., Sloan, R., and Warmuth, M., "Learning integer lattices," *SIAM Journal on Computing*, vol. 21, no. 2, pp. 240–266., 1992.

[56] Hoffgen, K., van Horn, K., and Simon, H. U., "Robust trainability of single neurons," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 114–125, 1995.

[57] Holmerin, J., Engebretsen, L., and Russell, A., "Inapproximability results for equations over finite groups," *Theoretical Computer Science*, vol. 312(1):17-45, 2004.

[58] Jackson, J., "An efficient membership-query algorithm for learning DNF with respect to the uniform distribution," *Journal of Computer and System Sciences*, vol. 55, pp. 414–440, 1997.

[59] Jerrum, M. and Snir, M., "Some exact complexity results for straight-line computations over semirings," *J. ACM*, vol. 29, pp. 874–897, 1982.

[60] Johnson, D. S. and Preparata, F. P., "The densest hemisphere problem.," *Theoretical Computer Science*, vol. 6, pp. 93–107, 1978.

[61] Kalai, A., Mansour, Y., and Verbin, E., "On agnostic boosting and parity learning," in *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of Computing*, 2008.

[62] Kalai, A. T., Klivans, A., Mansour, Y., and Servedio, R., "Agnostically learning halfspaces.," in *Proceedings of FOCS*, pp. 11–20, 2005.

[63] Karchmer, M. and Wigderson, A., "Monotone circuits for connectivity require super-logarithmic depth," in *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pp. 539–550, ACM Press, 1988.

[64] Kearns, M. and Li, M., "Learning in the presence of malicious errors," *SIAM Journal on Computing*, vol. 22, no. 4, pp. 807–837, 1993.

[65] Kearns, M., Schapire, R., and Sellie, L., "Toward efficient agnostic learning.," *Machine Learning*, vol. 17, no. 2-3, pp. 115–141, 1994.

[66] Kearns, M. and Valiant, L., "Cryptographic limitations on learning Boolean formulae and finite automata," *Journal of the ACM*, vol. 41, no. 1, pp. 67–95, 1994.

[67] Kearns, M. and Vazirani, U., *An introduction to computational learning theory.* Cambridge, MA: MIT Press, 1994.

[68] Kharitonov, M., "Cryptographic lower bounds for learnability of Boolean functions on the uniform distribution," *Journal of Computer and System Sciences*, vol. 50, pp. 600–610, 1995.

[69] Khot, S., "Improved inapproximability results for maxclique, chromatic number and approximate graph coloring," in *Proc. 42nd IEEE Annual Symposium on Foundations of Computer Science*, 2001.

[70] Khot, S. and Ponnuswami, A. K., "Better inapproximability results for MaxClique, Chromatic Number and Min-3Lin-Deletion," in *Proceedings of ICALP*, pp. 226–237, 2006.

[71] Khot, S., "On the power of unique 2-prover 1-round games," in *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, (New York, NY, USA), pp. 767–775, ACM Press, 2002.

[72] Khot, S. and Saket, R., "On hardness of learning intersection of two halfspaces," 2008. To appear in Proceedings of STOC.

[73] Klivans, A., O'Donnell, R., and Servedio, R., "Learning intersections and thresholds of halfspaces," in *Proceedings of the Forty-Third Annual Symposium on Foundations of Computer Science*, pp. 177–186, 2002.

[74] Klivans, A. and Servedio, R., "Boosting and hard-core set construction," *Machine Learning*, vol. 51, no. 3, pp. 217–238, 2003.

[75] Klivans, A. and Servedio, R., "Learning DNF in time $2^{\tilde{O}(n^{1/3})}$," *Journal of Computer and System Sciences*, vol. 68, no. 2, pp. 303–318, 2004.

[76] Klivans, A. and Sherstov, A., "Cryptographic hardness for learning intersections of halfspaces," in *Proceedings of FOCS*, pp. 553–562, 2006.

[77] Klivans, A. and Sherstov, A., "Improved lower bounds for learning intersections of halfspaces," in *Proc.* 19^{th} *Conference on Learning Theory (COLT)*, 2006.

[78] Knuth, D. E., "The sandwich theorem.," *Electr. J. Comb.*, vol. 1, 1994.

[79] Krieger, M. P., "On the incompressibility of monotone dnfs," *Theor. Comp. Sys.*, vol. 41, no. 2, pp. 211–231, 2007.

[80] Kushilevitz, E. and Mansour, Y., "Learning decision trees using the Fourier spectrum," *SIAM Journal on Computing*, vol. 22, no. 6, pp. 1331–1348, 1993.

[81] Levin, L., "Randomness and non-determinism," *Journal of Symbolic Logic*, vol. 58(3), pp. 1102–1103, 1993.

[82] Linial, N., Mansour, Y., and Nisan, N., "Constant depth circuits, Fourier transform and learnability," *Journal of the ACM*, vol. 40, no. 3, pp. 607–620, 1993.

[83] Littlestone, N., "Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm," *Machine Learning*, vol. 2, pp. 285–318, 1988.

[84] Luby, M. and Wigderson, A., "Pairwise independence and derandomization," Tech. Rep. 95-035, International Computer Science Institute, 1995.

[85] McEliece, R. J., "A public-key cryptosystem based on algebraic coding theory," *DSN progress report*, vol. 42-44, 1978.

[86] Minsky, M. and Papert, S., *Perceptrons: an introduction to computational geometry.* Cambridge, MA: MIT Press, 1969.

[87] Mossel, E., O'Donnell, R., and Servedio, R., "Learning functions of k relevant variables," *Journal of Computer and System Sciences*, vol. 69, no. 3, pp. 421–434, 2004.

[88] Motwani, R. and Raghavan, P., *Randomized Algorithms.* Cambridge University Press, 1995.

[89] Nisan, N. and Wigderson, A., "Lower bounds on arithmetic circuits via partial derivatives," *Comput. Complex.*, vol. 6, no. 3, pp. 217–234, 1997.

[90] Papadimitriou, C. and Yannakakis, M., "Optimization, approximation, and complexity classes," *Journal of Computer and System Sciences*, vol. 43, pp. 425–440, 1991.

[91] Ponnuswami, A. K. and Venkateswaran, H., "Monotone multilinear boolean circuits for bipartite perfect matching require exponential size," in *Proceedings of Foundations of Software Technology and Theoretical Computer Science*, pp. 460–468, 2004.

[92] Raz, R., "A parallel repetition theorem.," *SIAM Journal on Computing*, vol. 27, no. 3, pp. 763–803, 1998.

[93] Raz, R., "Multi-linear formulas for permanent and determinant are of super-polynomial size," in *Electronic Colloquium on Computational Complexity*, vol. 67, 2003.

[94] Raz, R. and Wigderson, A., "Monotone circuits for matching require linear depth," in *ACM Symposium on Theory of Computing*, pp. 287–292, 1990.

[95] Razborov, A. A., "Lower bounds on the monotone complexity of some boolean functions," *Doklady Akademii Nauk SSSR*, vol. 281, pp. 798–801, 1985. In Russian. English translation in *Soviet Mathematics Doklady*, 31:354–357, 1985.

[96] Rosenblatt, F., *Principles of neurodynamics*. New York: Spartan Books, 1962.

[97] Samorodnitsky, A. and Trevisan, L., "A PCP characterization of NP with optimal amortized query complexity," in *Proc. 32nd ACM Symposium on Theory of Computing*, pp. 191–199, 2000.

[98] Samorodnitsky, A. and Trevisan, L., "Gowers uniformity, influence of variables, and pcps," *Electronic Colloquium on Computational Complexity (ECCC)*, no. TR05-116, 2005.

[99] Sengupta, R. and Venkateswaran, H., "Multilinearity can be exponentially restrictive (preliminary version)," Tech. Rep. GIT-CC-94-40, College of Computing, Georgia Institute of Technology, 1994.

[100] Srinivasan, A., "The value of strong inapproximability results for clique," in *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pp. 144–152, 2000.

[101] Sudan, M. and Trevisan, L., "Probabilistically checkable proofs with low amortized query complexity," in *Proc. 39th IEEE Symposium on Foundations of Computer Science*, 1998.

[102] Tardos, E., "The gap between monotone and non-monotone circuit complexity is exponential," *Combinatorica*, vol. 8, pp. 141–142, 1988.

[103] Trevisan, L., "Non-approximability results for optimization problems on bounded degree instances," in *Proc. 33rd ACM Symposium on Theory of Computing*, pp. 453–461, 2001.

[104] Valiant, L. G., "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[105] Valiant, L. G., "Learning disjunctions of conjunctions," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 560–566, 1985.

[106] Vempala, S., *The Random Projection Method*. AMS, 2004.

[107] Verbeurgt, K., "Learning DNF under the uniform distribution in quasi-polynomial time," in *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pp. 314–326, 1990.

[108] Zuckerman, D., "On unapproximable versions of NP-complete problems," *SIAM J. on Computing*, pp. 1293–1304, 1996.