Electronic Colloquium on Computational Complexity, Report No. 59 (2006)

New Results for Learning Noisy Parities and Halfspaces

Vitaly Feldman[*]
Harvard University.
vitaly@eecs.harvard.edu

Parikshit Gopalan
Georgia Tech.
parik@cc.gatech.edu

Subhash Khot
Georgia Tech.
khot@cc.gatech.edu

Ashok Kumar Ponnuswami
Georgia Tech.
pashok@cc.gatech.edu

May 3, 2006

Abstract

We address well-studied problems concerning the learnability of parities and halfspaces in the presence of classification noise.

Learning of parities under the uniform distribution with random classification noise, also called the noisy parity problem is a famous open problem in computational learning. We reduce a number of basic problems regarding learning under the uniform distribution to learning of noisy parities, thus highlighting the central role of this problem for learning under the uniform distribution. We show that under the uniform distribution, learning parities with adversarial classification noise reduces to learning parities with random classification noise. Together with the parity learning algorithm of Blum *et al.* [BKW03], this gives the first nontrivial algorithm for learning parities with adversarial noise. We show that learning of DNF expressions reduces to learning noisy parities of just logarithmic number of variables. We show that learning of k-juntas reduces to learning noisy parities of k variables. These reductions work even in the presence of random classification noise in the original DNF or junta.

We then consider the problem of learning halfspaces over $\mathbb{Q}^n$ with adversarial noise or finding a halfspace that maximizes the agreement rate with a given set of examples. Finding the best halfspace is known to be NP-hard [GJ79, PV88] and many inapproximability results are known for this problem [ABSS97, HSH95, AK95, BDEL00, BB02]. We show that even if there is a halfspace that correctly classifies $1-\epsilon$ fraction of the given examples, it is hard to find a halfspace that is correct on a $\frac{1}{2}+\epsilon$ fraction for any $\epsilon > 0$ assuming $\mathrm{P} \neq \mathrm{NP}$. This gives an essentially optimal inapproximability factor of $2-\epsilon$, improving the factor of $\frac{85}{84}-\epsilon$ due to Bshouty and Burroughs [BB02]. Under stronger complexity assumptions, we can take ϵ to be as small as $2^{-\sqrt{\log n}}$ where n is the size of the input.

Finally, we prove that majorities of halfspaces are hard to PAC-learn using any representation, based on the cryptographic assumption underlying the security of the Ajtai-Dwork cryptosystem. We show that this result implies that learning halfspaces with high levels of adversarial noise is hard, independent of the representation of the hypothesis.

[*]Supported by grants from the National Science Foundation NSF-CCR-0310882, NSF-CCF-0432037, and NSF-CCF-0427129.

ISSN 1433-8092

1 Introduction

Parities and halfspaces are two of the most fundamental concept classes in learning theory. While efficient algorithms are known for learning these classes when the data is guaranteed to be noise-free, presence of noise leads to a number of challenging and important problems. The complexity of the problems depends on the process through which the noise is generated. In this article, we study various kinds of classification noise, where the noise only affects the label of a data point. This is different from the model of malicious errors defined by Valiant [Val85] (see also [KL93]) where the noise can affect both the label and the point itself, and thus possibly change the distribution of the data-points.

The two natural models for classification noise are random noise and adversarial noise. In the former model, the label of each point is flipped independently with probability η for some $\eta < 1/2$ and then given to the learner. As η approaches $1/2$ the labels approach an unbiased coin flip and hence the running time of the learning algorithm is allowed to depend polynomially on $\frac{1}{1-2\eta}$. In the adversarial noise model, an adversary is allowed to flip the labels of an η fraction of the input points. Alternatively, one can view this as learning in the agnostic framework of Haussler [Hau92] and Kearns et al. [KSS94] where we do not make any assumptions about how the data is generated, our goal is to find a hypothesis which does well in comparison to any other hypothesis from a certain class.

1.1 Learning Noisy Parities Under the Uniform Distribution

A parity function is the XOR of some set of variables $T \subseteq [n]$. In the absence of noise, one can identify the set T by taking samples and running Gaussian elimination. The problem of learning parity in the presence of noise has been well studied in both the adversarial noise model and the random noise model. Both these problems are closely related to problems in coding theory regarding the decoding of random linear codes. We summarize the known results about these problem.

- **Adversarial Noise:** In the adversarial noise model, under arbitrary distributions, the problem of learning parity is intractable in the proper learning setting where the learner must produce a parity as the hypothesis. This follows from a celebrated result of Håstad [Hås97]. We are unaware of non-trivial algorithms for this problem under any fixed distribution, prior to our work. The problem of learning parity with adversarial noise under the uniform distribution is related to the problem of decoding Hadamard codes. If the learner is allowed to ask membership queries, a celebrated result of Goldreich and Levin gives a polynomial time algorithm for this problem [GL89]. Later algorithms were given by Kushilevitz and Mansour [KM91] and Levin [Lev93].

- **Random Noise:** The problem of learning parity in the presence of random noise, or the noisy parity problem is a notorious open problem in computational learning theory. Blum, Kalai and Wasserman give algorithms for learning parity functions on n variables in the presence of random noise in time $2^{O(\frac{n}{\log n})}$ for any constant η [BKW03]. Their algorithm works for any distribution. We will also consider the problem under the promise that the set T is of size at most k. A brute force approach to this problem is to take $O(\frac{1}{1-2\eta}k \log n)$ samples and find the parity on k variables that best fits the data through exhaustive search in time $O(n^k)$. We are unaware of a better algorithm for this problem.

In this work, we focus on learning parities under the uniform distribution. We reduce a number of basic problems about learning under the uniform distribution to learning noisy parities, establishing the central role of noisy parities in this model of learning.

Learning Parities with Adversarial Noise

We show that under the uniform distribution, learning parities with adversarial noise reduces to learning parities with random noise at the cost of a small increase in the noise level. More precisely, learning parities with adversarial noise of rate η can be reduced to learning parities with random noise of rate $\frac{1}{2} - (\frac{1}{2} - \eta)^2$. In particular, our reduction and the result of Blum *et al.* imply the first non-trivial algorithm for learning parity with adversarial noise under the uniform distribution. Equivalently, this gives the first non-trivial algorithm under any distribution for agnostically learning parity (see Section 3.1 for the definition of agnostic learning).

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

Our main technical contribution is to show that an algorithm for learning noisy parities gives an algorithm that finds heavy 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 queries.

This result is proved using Fourier analysis. The high-level idea of the reduction is to modify the Fourier spectrum of the function f so that it is concentrated at a single point. For this, we introduce the notion of a probabilistic oracle for real-valued functions $f : \{0,1\}^n \to [-1,1]$. Informally, such an oracle produces a pair $\langle x, b\rangle$ where x is random vector in $\{0,1\}^n$ and $b \in \{-1,1\}$ is a random bit whose expectation is $f(x)$. We present a set of transformation on oracles that allow us to use an oracle for the function f to simulate oracles for many related functions. In particular, one can reduce the influence of the smaller Fourier coefficients, and project f onto Fourier coefficients belonging to a particular subspace of $\{0,1\}^n$. By composing these operations, we show that one can simulate an oracle which is close (in statistical distance) to a noisy parity.

Learning DNF formulae

Learning of DNF expressions from random examples is another famous open problem formulated in Valiant's seminal paper on PAC learning [Val84]. In this problem we are given access to examples of some Boolean function f randomly chosen with respect to distribution $\mathcal{D}$, and $\epsilon > 0$. The goal is to find a hypothesis that ϵ-approximates f with respect to $\mathcal{D}$ 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. A straightforward algorithm for learning DNFs with respect to the uniform distribution collects all the terms of size $\log(s/\epsilon) + O(1)$ that are consistent (in the PAC sense) with the target function, i.e. do not accept negative points. We are unaware of an algorithm improving on the $O(n^{\log(s/\epsilon)})$ running time of this algorithm. Jackson [Jac97] proved that DNFs are learnable under the uniform distribution if the learning algorithm is allowed to get the value of function f at any point (or make *membership queries*). This influential result gives essentially the only known approach to learn unrestricted DNFs in polynomial time.

We establish a connection between these problems proving that learning of DNF expressions reduces to learning parities of $O(\log(s/\epsilon))$ variables with noise rate $\eta = 1/2 - \tilde{O}(\epsilon^2/s^2)$ under the uniform distribution.

Theorem 2 *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 $T(n, k, \frac{1}{1-2\eta})$ 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 $\tilde{O}(\frac{s^4}{\epsilon^2} \cdot T(n, \log B, B^2) \cdot S(n, \log B, B^2))$, where $B = \tilde{O}(s/\epsilon)$.*

Learning k-juntas

A Boolean function on n variables is a k-junta if it depends only on k variables out of n. This problem was proposed by Blum and Langley [BL97], as a clean formulation of the problem of efficient learning in the presence of irrelevant information. In addition, for $k \leq \log n$, a k-junta can be expressed as a decision tree or a DNF of size n. Hence, a polynomial time algorithm for DNFs or decision trees under the uniform distribution would imply an algorithm for the k-junta problem. Thus, learning juntas is a first step towards learning polynomial size decision trees and DNFs under the uniform distribution. A brute force approach 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. [MOS03], and runs in time roughly $O(n^{0.7k})$. Their algorithm relies on some new structural properties of Boolean functions. However, even the question of whether one can learn k-juntas in polynomial time for $k = \omega(1)$ still remains open.

For the problem of learning k-juntas, we give a simpler reduction to learning noisy parities of size k, which also gives a lower noise rate η.

Theorem 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 $O(2^{2k} \cdot T(n, k, 2^k))$.*

This reduction also applies for learning k-juntas with random noise. A noisy 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 DNF can also be made resilient to random noise.

Even though our reductions for DNFs and juntas do not give an improved algorithm for at this stage, 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 parities of $O(\log n)$ variables will help make progress on a number of important questions regarding learning under the uniform distribution.

1.2 Hardness of Learning Halfspaces with Adversarial Noise

In the problem of learning a halfspace, we are given a set of points in $\mathbb{R}^n$ labeled '$+$' or '$-$'. Our goal is to find a halfspace that separates the '$+$'s and the '$-$'s, if such a halfspace exists, else to find one that classifies most of the data correctly. This is one of the oldest and best-studied problems in machine learning, dating back to work on Perceptrons in the 1950s [Agm64, Ros64, MP69]. If such a halfspace does exist, one can find it in polynomial time using efficient algorithms for Linear Programming. In practice, simple greedy algorithms like the Perception and Winnow are used, which also seem to be robust to noise [Gal90, Ama94]. Much of the recent research in this area focuses on finding provably good algorithms when the data is noisy [BFKV96, ABSS97, Coh97, KKMS05]. Halfspaces are PAC-learnable even in the presence of random noise: Blum et al. [BFKV96] show that a variant of the Perceptron algorithm can be used in this setting (see also [Coh97]).

In the adversarial noise scenario, there is no halfspace that correctly classifies all the data points and our goal is to find one that does as well as possible. This version of the halfspace problem arises frequently in practice, where the data is often inconsistent. Also, this problem has important implications for some central open problems in computational learning, namely PAC learning of DNFs and AC^0 circuits.

This problem was known to be NP-complete [GJ79], even when the points lie in $\{0,1\}^n$ [PV88]. A number of results are known on hardness of producing a halfspace whose rate of agreement is within a certain constant factor of the optimal. Amaldi and Kann [AK95], Ben-David *et al.* [BDEL00], and Bshouty and Burroughs [BB02] prove hardness of approximating agreements with halfspaces - factors $\frac{262}{261}$, $\frac{418}{415}$, and $\frac{85}{84}$, respectively. Further Arora et al. [ABSS97] showed that for any constant C, the problem of minimizing the number of points that are wrongly classified by the halfspace is hard to approximate within factor C.

The problem we resolve is the following: if there is a halfspace that correctly classifies 99% of the data, can one find a halfspace that is correct on even 51%? Note that by taking any hyperplane, and deciding which halfspace to label '$+$', one can get a 50% success rate. An algorithm with a non-trivial guarantee would be a tremendous breakthrough in computational learning. It is known that any AC^0 circuit can be approximated by the sign of a low-degree polynomial over the reals under any distribution [BRS91, ABFR91]. Thus, a non-trivial algorithm for learning a halfspace would imply that AC^0 circuits are weakly PAC-learnable. One can then boost this to get a PAC-learning algorithm for AC^0 circuits in quasi-polynomial time [KV94]. In contrast, at present the best PAC-learning algorithm even for DNFs (which are a special case of AC^0 circuits) runs in time $2^{\tilde{O}(n^{1/3})}$ [KS01]. This connection is well-known, Blum et al. [BFKV96] observe this connection, and state the question of learning halfspaces with adversarial noise as an important open problem.

We prove a negative result which is essentially optimal. 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 a $\frac{1}{2}+\epsilon$ fraction of the inputs for any $\epsilon > 0$ assuming $\mathrm{P} \neq \mathrm{NP}$. Under stronger complexity assumptions, we can take ϵ to be as small as $2^{-\sqrt{\log n}}$ where n is the size of the input.

Theorem 4 *Assuming* $\mathrm{NP} \nsubseteq \mathrm{DTIME}(2^{(\log n)^{O(1)}})$, *no polynomial time algorithm can distinguish between the following cases of the halfspace problem over N points:*

- $1 - 2^{-\Omega(\sqrt{\log N})}$ *fraction of the points can be correctly classified by some halfspace.*
- *No more than* $1/2 + 2^{-\Omega(\sqrt{\log N})}$ *fraction of the points can be correctly classified by any halfspace.*

Similar optimal results were only known for the class of parities [Hås97] and, recently, for monomials [Fel06b].

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 some 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. The idea of repeated tensoring and boosting was used by Khot and Ponnuswami for equations over $\mathbb{Z}_2$ in order to show hardness for Max-Clique [KP06]. 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. This result was proved independently by Guruswami and Raghavendra [GR06], their result holds even if the data points are restricted to lie in $\{0,1\}^n$.

We note that the approximability of systems of linear equations over various fields is a well-studied problem. Håstad shows that no non-trivial approximation is possible over $\mathbb{Z}_2$ [Hås97]. Similar results are known for equations over $\mathbb{Z}_p$ and finite groups [Hås97, HER04]. However, to our knowledge this is the first optimal hardness result for equations over $\mathbb{Q}$. On one hand, the Fourier analytic techniques that work well for finite groups and fields do not seem to apply over $\mathbb{Q}$. On the other hand, the fact that we are not restricted to equations with constantly many variables makes our task much simpler. A natural open question is whether a similar hardness result holds for equations of constant size over $\mathbb{Q}$.

We note that this is a hardness result for proper learning, where the algorithm must output a halfspace as a hypothesis. In order to PAC-learn AC^0 circuits, any hypothesis suffices provided it can be evaluated efficiently. On one hand, halfspaces are the most natural hypothesis to begin with, since many of the known algorithms for learning halfspaces are proper learning algorithms, including the algorithms for the non-noisy case, as well as for random classification noise [Coh97]. On the other hand, a recent algorithm for agnostically learning halfspaces under certain distributions due to Kalai et al. [KKMS05] uses the sign of a low degree polynomial as a hypothesis, hence it is not a proper learning algorithm. This suggests a natural question of whether it is possible to learn a halfspace under adversarial noise using a low-degree polynomial. To our knowledge, even NP-hardness is not known for this problem, even if we restrict the hypothesis to quadratic polynomials.

1.2.1 Thresholds of Halfspaces are not PAC-learnable

As opposed to proper-learning results, once could hope to show that a certain concept class is hard to PAC-learn, irrespective of the hypothesis representation. Such results typically rely on specific cryptographic assumptions [KV94]. 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, which are used in machine learning [Mit97]. 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 $\{0,1\}^n$.

Thresholds of halfspaces are well studied in the literature [BK97, Vem04, KOS02, ABF$^+$04, KS06a]. There are known algorithms for learning various thresholds of k halfspaces both over $\mathbb{R}^n$ [BK97, Vem04] and $\{0,1\}^n$ [KOS02] under many distributions. Typically, the running time of these algorithm is exponential in k. There are numerous negative results known for proper learning of such concepts [Vem04]. Recent results by Alekhnovich et al. [ABF$^+$04] show that it is NP-hard to learn the intersection of $k \geq 2$ halfspaces by ℓ halfspaces for any constant ℓ. They also show that one cannot learn a union by halfspaces (even a DNF) by a union of halfspaces in polynomial time assuming RP$\neq$ NP. Klivans and Sherstov show lower bounds for learning intersections of halfspaces in the Statistical Query model [KS06a]. There has also been work showing that constant depth threshold circuits are hard for PAC-learning. Based on certain cryptographic assumptions, Kearns and Valiant showed that constant depth threshold circuits cannot be learned over a certain distribution using any representation [KV89]. Kharitonov strengthened this result by allowing membership queries, and using the uniform distribution [Kha95].

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 [AD97].

Theorem 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.*

For this result, one can even take all the gates to be Majority gates. 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 [KV89]. We show that the decryption of the Ajtai-Dwork cryptosystem [AD97], (specifically a modification by Goldreich et al. [GGH97]) can be done by a depth-2 threshold circuit. This result was obtained independently by Klivans and Sherstov [KS06b], their reduction allows one to use an AND gate at the second level.

Finally, using the Discriminator Lemma of Hajnal *et al.* [HMP$^+$93], we show that Theorem 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 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 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 $\frac{1}{2}$.

This paper is organized as follows: we present our main technical Lemma regarding finding significant Fourier coefficients in Section 2. We derive various consequences of this Lemma in Section 3. We present our hardness result for proper learning of halfspaces in Section 4. In Section 5, we show the hardness of learning majorities of halfspaces under cryptographic assumptions.

2 Learning Parities with Noise

In this section, we describe the main component of our reductions: an algorithm that using an algorithm for learning noisy parities, finds a heavy Fourier coefficient of a Boolean function if one exists. Following Jackson, we call such an algorithm a *weak parity algorithm*.

2.1 Fourier Transform

Our reduction uses Fourier-analytic techniques which were first used in computational learning by Linial *et al.* [LMN93]. 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 $\mathtt{weight}(a)$ denote the Hamming weight of a. For any real-valued function f the Fourier coefficient of f with *index* a is $\hat{f}(a) = \mathrm{E}_x[f(x)\chi_a(x)]$ and its *degree* is $\mathtt{weight}(a)$. We say that a Fourier coefficient $\hat{f}(a)$ is θ-heavy if $|\hat{f}(a)| \geq \theta$. The function f can be written as $f(x) = \sum_a \hat{f}(a)\chi_a(x)$. Let $L_2(f) = E_x[(f(x))^2]^{1/2}$. Parseval's identity states that

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

We also define the quantity $L_1(f)$ as

$$L_1(f) \; = \; \sum_a |\hat{f}(a)|.$$

2.2 Oracle Transformations

Given an 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.

Definition 1 *For any function $f : \{0,1\}^n \to [-1,1]$ an 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 $f(x)$.*

Our first observation is that for $\theta \in [-1,1]$, $\mathbb{O}(\theta\chi_a(x))$ is exactly the oracle for parity $\chi_a(x)$ with noise of rate $\eta = 1/2 - \theta/2$. Our second key 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 1 *The statistical distance between the outputs of $\mathbb{O}(f)$ and $\mathbb{O}(g)$ is upper-bounded by $L_1(f-g)$.*

PROOF: The probability that $\mathbb{O}(f)$ outputs $\langle x, 1\rangle$ is $\frac{1+f(x)}{2}$ and the probability that it outputs $\langle x, -1\rangle$ is $\frac{1-f(x)}{2}$. Therefore the statistical distance between $\mathbb{O}(f)$ and $\mathbb{O}(g)$ equals

$$\mathrm{E}_x\left[|f(x) - g(x)|\right] \; = \; \mathrm{E}_x\left[\left|\sum_a (\hat{f}(a) - \hat{g}(a))\chi_a(x)\right|\right] \; \leq \; \sum_a \left|\hat{f}(a) - \hat{g}(a)\right| \; = \; L_1(f-g)$$

□

We now describe the transformations on probabilistic oracles that will be used in our reductions. For a function $f : \{0,1\}^n \to [-1,1]$ we define $f_2(x)$ as

$$f_2(x) = \sum_a (\hat{f}(a))^2 \chi_a(x)$$

Intuitively, the function f_2 reduces the influence of the smaller Fourier coefficients in f. Note that if f is a Boolean function, then $L_1(f)$ could be large, however $L_1(f_2) = 1$.

Lemma 1 *For the function f_2 defined above:*

1. $f_2(x) = \mathrm{E}_{y_1 \oplus y_2 = x}[f(y_1)f(y_2)]$.
2. *Given access to the oracle $\mathbb{O}(f)$ one can simulate the oracle $\mathbb{O}(f_2)$.*

Proof: Part 1 follows from the standard Fourier analysis. To simulate the oracle $\mathbb{O}(f_2)$ get two samples $\langle y_1, b_1\rangle$ and $\langle y_2, b_2\rangle$ from $\mathbb{O}(f)$ and return $\langle y_1 \oplus y_2, b_1 b_2\rangle$. By Part 1, this simulates the oracle for f_2. $\square$

For a matrix $A \in \{0,1\}^{m\times n}$ define an A-projection of f to be

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

Lemma 2 *For the function f_A defined above:*

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

Proof: First 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 $Aa = 0^m$ then $E_p[\chi_a(A^T p)] = 1$ otherwise it is 0. Now let

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

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

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

Therefore $\widehat{g_A}(a) = \hat{f}(a)$ if $Aa = 0^m$ 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 $\langle x \oplus A^T p, b\rangle$. The correctness follows from Part 1 of the Lemma. $\square$

We will use this Lemma 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 3 *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:*

$$\begin{aligned}
\widehat{f_A}(s) &= \hat{f}(s) &\quad (1)\\
\sum_{a\in\{0,1\}^n\setminus\{s,0^n\}} |\widehat{f_A}(a)| &\leq L_1(f)2^{-m+1} &\quad (2)
\end{aligned}$$

Proof: Event (1) holds if $As = 0^m$, which happens with probability 2^{-m}.

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

$$\begin{aligned} \Pr_v[v \cdot a = 0 \mid v \cdot s = 0] &= 1/2 \\ \text{Therefore} \quad \Pr_A[Aa = 0^m \mid As = 0^m] &= 2^{-m} \end{aligned}$$

This implies that

$$\mathrm{E}_A\left[\sum_{a \in \{0,1\}^n \setminus \{s,0^n\}} |\widehat{f_A}(a)| \;\middle|\; As = 0^m\right] = \sum_{a \in \{0,1\}^n \setminus \{s,0^n\}} 2^{-m}|\hat{f}(a)| \;\leq\; 2^{-m}L_1(f)$$

By Markov's inequality,

$$\Pr_A\left[\sum_{a \in \{0,1\}^n \setminus \{s,0^n\}} |\widehat{f_A}(a)| \geq 2^{-m+1}L_1(f) \;\middle|\; As = 0^m\right] \leq 1/2$$

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

This projection does not influence $\hat{f}(0^n)$. But one can clear the coefficient $\hat{f}(0)$ as follows. Define the function f^{-0} as

$$f^{-0}(x) = \frac{1}{2}\sum_{a \neq 0^n} \hat{f}(a)\chi_a(x)$$

Lemma 4 *For the function f^{-0} defined above, given access to $\mathbb{O}(f)$ one can simulate the oracle $\mathbb{O}(f^{-0})$.*

Proof: By definition, $\hat{f}(0^n) = \mathrm{E}_x[f(x)]$. Therefore

$$f^{-0}(x) \;=\; \frac{1}{2}f(x) - \frac{1}{2}\mathrm{E}_y[f(y)] \;=\; \frac{1}{2}f(x) + \frac{1}{2}\mathrm{E}_y[-f(y)] \tag{3}$$

Let $\langle x, b_1\rangle$ and $\langle y, b_2\rangle$ be two random examples from $\mathbb{O}(f)$. With probability $\frac{1}{2}$ return example $\langle x, b_1\rangle$ and with probability $\frac{1}{2}$ return example $\langle x, -b_2\rangle$. By Equation 3, this simulates $\mathbb{O}(f^{-0})$. $\square$

Finally, we show that by composing these transformations, one can use an algorithm for learning noisy parities to get a weak parity algorithm.

Theorem 7 *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 $\mathbb{O}(f)$ finds s in time $O(T(n, k, 2\theta^{-2}) \cdot S(n, k, 2\theta^{-2}))$ with probability at least $1/2$.*

Proof: The algorithm `WP-R` proceeds in the following steps:

1. Square the coefficients of f and obtain $\mathbb{O}(f_2)$.
2. Clear $\widehat{f_2}(0^n)$ and obtain the oracle $\mathbb{O}(f_2^{-0})$.
3. Let $m = \lceil \log S(n, k, 2\theta^{-2})\rceil + 2$. Let $A \in \{0,1\}^{m \times n}$ be a randomly chosen matrix and $\mathbb{O}(f_{2,A}^{-0})$ be the oracle for A-projection of f_2^{-0}. Run the algorithm $\mathcal{A}$ on $\mathbb{O}(f_{2,A}^{-0})$.

4. If $\mathcal{A}$ stops in $T(n, k, 2\theta^{-2})$ steps and outputs r with $\texttt{weight}(r) \leq k$, check that r is at least $\theta/2$-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 $\chi_s(x)$.

The first step transforms f into a function f_2 with small L_1 norm ($L_1(f_2) = 1$), while at the same time $f_2(s) = \theta^2$ so s stays a reasonably heavy coefficient. The second step clears the coefficient $\widehat{f_2}(0^n)$ and halves the rest of the coefficients. By Lemma 3, in the third step, with probability at least 2^{-m-1} , we create a function $f_{2,A}^{-0}$ such that $\widehat{f_{2,A}^{-0}}(s) \geq \theta^2/2$ and

$$\sum_{a \neq s} |\widehat{f_{2,A}^{-0}}(a)| \;\leq\; 2^{-m+1} L_1(f_2^{-0}) \;\leq\; \frac{1}{2S(n, k, 2\theta^{-2})}$$

By Claim 1, the statistical difference between the oracle $\mathbb{O}(f_{2,A}^{-0})$ and oracle $\mathbb{O}(\widehat{f_{2,A}^{-0}}(s)\chi_s(x))$ is small. Since $\mathcal{A}$ uses at most $S(n, k, 2\theta^{-2})$ samples, with probability at least 1/2 it will not notice the difference between the two oracles. But, $\mathbb{O}(\widehat{f_{2,A}^{-0}}(s)\chi_s(x))$ is exactly the noisy parity χ_s with noise rate

$$\eta \;=\; 1/2 - \widehat{f_{2,A}^{-0}}/2 \;\leq\; 1/2 - \theta^2/4$$

Hence we get $(1 - 2\eta)^{-1} \leq 2\theta^{-2}$, so the algorithm $\mathcal{A}$ will learn the parity s. We can check that the coefficient produced by $\mathcal{A}$ is indeed heavy using Chernoff bounds, and repeat until we succeed. Using $O(2^m) = O(S(n, k, 2\theta^{-2}))$ repetitions, we will get a θ-heavy Fourier coefficient of degree k with probability at least 1/2.
□

Remark 1 *A function f can have at most $L_2^2(f)/\theta^2$ θ-heavy Fourier coefficients. Therefore by repeating* `WP-R` *$O(\log(L_2(f)/\theta))$ 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.*

3 Applications

3.1 Agnostic Learning of Parities and Adversarial Noise

We start by defining the agnostic PAC model of Haussler [Hau92] and Kearns *et al.* [KSS94] more precisely. For two Boolean functions f and h and a distribution $\mathcal{D}$ over $\{0,1\}^n$ we define $\Delta_{\mathcal{D}}(f, h) = \Pr_{\mathcal{D}}[f \neq h]$. Similarly, for a class of Boolean functions $\mathcal{C}$ and a function f define $\Delta_{\mathcal{D}}(f, \mathcal{C}) = \min_{h \in \mathcal{C}}\{\Delta_{\mathcal{D}}(f, h)\}$. We say that a class $\mathcal{C}$ is PAC learnable in the agnostic model under the distribution $\mathcal{D}$ if there exists an algorithm that for every Boolean function f and $\epsilon > 0$, runs in time polynomial in n and $1/\epsilon$ and, with probability at least 1/2, produces a hypothesis h such that $\Delta_{\mathcal{D}}(f, h) \leq \Delta_{\mathcal{D}}(f, \mathcal{C}) + \epsilon$.

An equivalent way to think of function f is as a function in $\mathcal{C}$ corrupted by adversarial classification noise of rate $\Delta_{\mathcal{D}}(f, \mathcal{C})$. A minor difference arises from the fact that the noise rate is not explicitly mentioned in the agnostic setting. But we note that if $\epsilon < 1/2 - \eta$ then a constant function will be a good agnostic hypothesis. Therefore any algorithm that is polynomial in $1/\epsilon$ will also be polynomial in $\frac{1}{1-2\eta}$.

Theorem 7 gives the following reduction from adversarial to random noise.

Theorem 8 *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 parities with adversarial noise η' are learnable in time $O(T(n, k, \frac{2}{(1-2\eta')^2}) \cdot S(n, k, \frac{2}{(1-2\eta')^2}) + \epsilon^{-2})$ with probability at least $1/2$.*

Proof: Let f be a parity χ_s corrupted by noise of rate η'. Then $\hat{f}(s) = \mathrm{E}[f\chi_s] \geq (1-\eta') + (-1)\eta' = 1 - 2\eta'$. We now apply the reduction from Theorem 7 with a minor modification. We make the estimates of Fourier coefficients within ϵ to make sure that we return a coefficient that is at least $1 - 2\eta' - \epsilon$, meaning that the error of our hypothesis is at most $\eta' + \epsilon$ (as required by the definition). □

Blum *et al.* give a sub-exponential algorithm for learning noisy parity.

Lemma 5 *[BKW03] Parity functions on $\{0,1\}^n$ can be learned in computation-time and sample-complexity $2^{O(\frac{n}{\log n})}$ in the presence of random noise of rate η for any constant $\eta < \frac{1}{2}$.*

If the rate of adversarial noise η' is a constant then the rate of random noise that our reduction produces $\eta = \frac{1}{2} - (\frac{1}{2} - \eta')^2$ is also a constant. Therefore Theorem 8 together with Lemma 5 gives Theorem 1.

3.2 Learning DNF Expressions

Jackson [Jac97] in his breakthrough result on learning DNF expressions with respect to the uniform distribution gives a way to use an algorithm for locating correlated parities and the boosting algorithm due to Freund [Fre90] to build a DNF learning algorithm. We can adapt Jackson's approach to our setting. We give an outline of the algorithm and omit the now-standard analysis.

We view a probability distribution $\mathcal{D}$ as a density function and define its L_∞ norm. Jackson's algorithm is based on the following Lemma (we use a refinement from [BF02]).

Lemma 6 ([BF02](Lemma 18)) *For any Boolean function f of DNF-size s and any distribution $\mathcal{D}$ over $\{0,1\}^n$ there exists a parity function χ_a such that $|\mathrm{E}_{\mathcal{D}}[f\chi_a]| \geq \frac{1}{2s+1}$ and $\mathtt{weight}(a) \leq \log((2s+1)L_\infty(2^n\mathcal{D}))$.*

This lemma implies that DNFs can be weakly learned by finding a parity correlated with f under distribution $\mathcal{D}(x)$ which is the same as finding a parity correlated with the function $2^n\mathcal{D}(x)f(x)$ under the uniform distribution. The range of $2^n\mathcal{D}(x)f(x)$ is not necessarily $[-1,1]$, whereas our `WP-R` algorithm was defined for functions with this range. So in order to apply Theorem 7, we first scale $2^n\mathcal{D}(x)f(x)$ to the range $[-1,1]$ and obtain the function $\mathcal{D}'(x)f(x)$, where $\mathcal{D}'(x) = \mathcal{D}(x)/L_\infty(2^n\mathcal{D})$ ($L_\infty(\mathcal{D})$ is known to the boosting algorithm). We then get the probabilistic oracle $\mathbb{O}(\mathcal{D}'(x)f(x))$ by flipping a ± 1 coin with expectation $\mathcal{D}'(x)f(x)$. Therefore a θ-heavy Fourier coefficient of $2^n\mathcal{D}(x)f(x)$ can be found by finding a $\theta/L_\infty(2^n\mathcal{D})$-heavy Fourier coefficient of $\mathcal{D}'(x)f(x)$ and multiplying it by $L_\infty(2^n\mathcal{D})$. We summarize this generalization in the following lemma.

Lemma 7 *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 $O(T(n,k,2(L_\infty(\phi)/\theta)^2) \cdot S(n,k,2(L_\infty(\phi)/\theta)^2))$ with probability at least $1/2$.*

The running time of `WP-R'` depends on $L_\infty(2^n\mathcal{D})$ (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 [Jac97], Freund's boost-by-majority algorithm [Fre90] is used to produce distribution functions bounded by $O(\epsilon^{-(2+\rho)})$ (for arbitrarily small constant ρ). More recently, Klivans and Servedio have observed [KS03] that a later algorithm by Freund [Fre92] produces distribution functions bounded by $\tilde{O}(\epsilon)$. Putting the two components together, we get the proof of Theorem 2.

3.3 Learning Juntas

For the class of k-juntas, we can get a simpler reduction with better parameters for noise. We first clear the coefficient $\hat{f}(0)$. 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 1 we will obtain the complete Fourier spectrum of f by repeating the algorithm $O(k)$ times. The proof of Theorem 3 follows from these observations.

3.4 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 9 *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 $O(2^{2k} \cdot T(n,k,\frac{2^k}{1-2\eta'}))$.*

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.

Handling noise in the DNF reduction is more subtle since Freund's boosting algorithms do not necessarily work in the presence of noise, in particular Jackson's original algorithm does not handle noisy DNFs. Nevertheless, as shown by Feldman [Fel06a], the effect of noise can be offset if the weak parity algorithm can handle a "noisy" version of $2^n\mathcal{D}(x)f(x)$. More specifically, we need a generalization of the `WP-R` algorithm that for any real-valued function $\phi(x)$, finds a heavy Fourier coefficient of $\phi(x)$ given access to $\Phi(x)$, where $\Phi(x)$ is an independent random variable with expectation $\phi(x)$ and $L_\infty(\Phi(x)) \leq \frac{2L_\infty(\phi)}{1-2\eta}$. It is easy to see that `WP-R'` can handle this case. Scaling by $L_\infty(\Phi(x))$ will give us a random variable $\Phi'(x)$ in the range $[-1,1]$ with expectation $\phi(x)/L_\infty(\Phi(x))$. By flipping a ± 1 coin with expectation $\Phi'(x)$ we will get a ± 1 random variable with expectation $\phi(x)/L_\infty(\Phi(x))$. Therefore `WP-R` algorithm will find a heavy Fourier coefficient of $\phi(x)$ (scaled by $L_\infty(\Phi(x)) \leq \frac{2L_\infty(\phi)}{1-2\eta}$). Altogether we obtain the following theorem for learning noisy DNFs.

Theorem 10 *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 $T(n,k,\frac{1}{1-2\eta})$ using at most $S(n,k,\frac{1}{1-2\eta})$ examples. Then there exists an algorithm that learns DNF expressions of size s with random noise of rate η' in time $\tilde{O}(\frac{s^4}{\epsilon^2} \cdot T(n, \log B, (\frac{B}{1-2\eta'})^2) \cdot S(n, \log B, B^2))$ where $B = \tilde{O}(s/\epsilon)$.*

4 Hardness of Learning a Halfspace with Adversarial Noise

The following is the combinatorial version of the problem of learning a halfspace over $\mathbb{Q}^m$ with adversarial noise (as stated in Arora et al. [ABSS97]).

Definition 2 *Let S^+ and S^- be two sets of points from $\mathbb{Q}^m$. Given a pair (S^+, S^-) as input, the goal of the halfspace learning problem is to find a hyperplane $\sum_{j=1}^m a_j x_j = b$ that correctly classifies as many points as possible. A point $x \in S^+$ is said to be correctly classified if $a.x > b$ and a point $x \in S^-$ is said to be correctly classified if $a.x < b$.*

A trivial factor 2 algorithm for this problem is to just take an arbitrary hyperplane and decide which one of the two sides of the hyperplane to mark as positive. We show that this is essentially the best approximation possible. The proof is by reduction from the gap version of 5-regular vertex cover to the learning halfspaces problem. The main steps in the reduction are as follows:

- Obtain a (N, c, s, t) instance of MaxLin-$\mathbb{Q}$ from the vertex cover instance for some fixed constants c, s and t (see Definition 3).
- Convert the above instance to a $(N', 1-\epsilon, \epsilon, t')$ gap where ϵ is a very small function of N'. To achieve this, we use two operations called *tensoring* and *boosting*.
- Convert the instance of the MaxLin-$\mathbb{Q}$ problem obtained to an instance of the halfspace problem.

We begin by defining the MaxLin-$\mathbb{Q}$ 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 *Given a system of linear equations with rational coefficients*

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

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

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

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

  is true for at most sN values of $i \in 1, 2, \ldots, N$.

The goal of the MaxLin-$\mathbb{Q}$ 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 MaxLin-$\mathbb{Q}$ can be specified by a matrix

$$\boldsymbol{A} = \begin{bmatrix} a_{10} & a_{11} & \ldots & a_{1m} \\ a_{20} & a_{21} & \ldots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{N0} & a_{N1} & \ldots & a_{Nm} \end{bmatrix}$$

We will refer to $\boldsymbol{A}$ itself as an instance of MaxLin-$\mathbb{Q}$. We may also use the rows of $\boldsymbol{A}$ to represent the equations in the instance. The MaxLin-$\mathbb{Q}$ 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. In all the instances of MaxLin-$\mathbb{Q}$ that we consider, the number of variables will be less than the number of equations in the system. Also, the size of each entry of the matrix will be proportional to the number of equations. Hence, we refer to N itself as the size of the instance.

4.1 A Small Hardness Factor for MaxLin-$\mathbb{Q}$

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

Lemma 8 *[PY91, ALM$^+$98] 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. [ABSS97] give a reduction from the above gap version of vertex cover of regular graphs to MaxLin-$\mathbb{Q}$. They show that if there is a "small" vertex cover, the reduction produces a MaxLin-$\mathbb{Q}$ 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 9 *There exists a polynomial time algorithm that when given a 5-regular graph $G = (V, E)$ with n vertices as input produces a (N, c_0, s_0, t_0) MaxLin-$\mathbb{Q}$ instance $\boldsymbol{A}$ over n variables as output where $N = n^{O(1)}$, 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

$$\begin{aligned} -1 + x_i + x_{i'} &= 0 \\ -1 + x_i &= 0 \\ -1 + x_{i'} &= 0 \end{aligned}$$

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

$$\begin{aligned} |-1 + x_i + x_{i'}| &< 1/3 \\ |-1 + x_i| &< 1/3 \\ |-1 + x_{i'}| &< 1/3 \end{aligned}$$

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. 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

$$\begin{aligned} y_i + y_{i'} &= 1 \\ y_i &= 1 \\ y_{i'} &= 1 \end{aligned}$$

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. $\square$

4.2 Amplifying the Gap for MaxLin-$\mathbb{Q}$

We define two operations called *tensoring* and *boosting*. Tensoring converts a $(N, 1-\epsilon, 1-\delta, t)$ MaxLin-$\mathbb{Q}$ instance to a $(N^2, 1-\epsilon^2, 1-\delta^2, t^2)$ MaxLin-$\mathbb{Q}$ 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 (N, c, s, t) MaxLin-$\mathbb{Q}$ instance to a $(O(\rho)^\sigma N, c^\sigma, s^{\Omega(\sigma)}, t/2)$ MaxLin-$\mathbb{Q}$ instance. Note that there must be a reasonable gap between the completeness and soundness before we can use boosting. We amplify the (c, s) gap for MaxLin-$\mathbb{Q}$ in four steps:

- Obtain a $(1-\epsilon, 1-K\epsilon)$ gap for very large constant K.
- Obtain a $(1-\epsilon_0, \epsilon_0)$ gap for a very small constant $\epsilon_0 > 0$ by a boosting operation. The first step guarantees that we can use boosting without destroying the gap.
- Improve the completeness even further to $1 - o(1)$ while keeping the soundness at ϵ_0. This is done by alternately tensoring and boosting many times.
- Using one more boosting operation, decrease the soundness. This gives the $(N', 1-\epsilon, \epsilon, t')$ instance where $\epsilon = 2^{-\Omega(\sqrt{\log N'})}$ as desired.

We define the first operation called tensoring. This operation is similar to an operation defined by Dumer et al. [DMS99] 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 occurance of $x_{j_1}x_{j_2}$ with $x_{j_1j_2}$ and x_j with x_{0j} respectively.

Definition 4 *The tensoring of the system of equations*

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

is the system

$$\{a_{i_10}a_{i_20} + a_{i_10}(\sum_{j_2=1}^{m} a_{i_2j_2}x_{0j_2}) + a_{i_20}(\sum_{j_1=1}^{m} a_{i_1j_1}x_{0j_1}) + (\sum_{j_1=1}^{m}\sum_{j_2=1}^{m} a_{i_1j_1}a_{i_2j_2}x_{j_1j_2}) = 0\}_{i_1,i_2=1,\ldots,N}$$

In the matrix representation, the tensoring of

$$\begin{bmatrix} a_{10} & a_{11} & \ldots & a_{1m} \\ a_{20} & a_{21} & \ldots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{N0} & a_{N1} & \ldots & a_{Nm} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_m \end{bmatrix} = 0$$

is the system

$$\begin{bmatrix} a_{10} & a_{11} & \ldots & a_{1m} \\ a_{20} & a_{21} & \ldots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{N0} & a_{N1} & \ldots & a_{Nm} \end{bmatrix} \begin{bmatrix} 1 & x_{01} & \ldots & x_{0m} \\ x_{01} & x_{11} & \ldots & x_{1m} \\ \vdots & \vdots & & \vdots \\ x_{0m} & x_{m1} & \ldots & x_{mm} \end{bmatrix} \begin{bmatrix} a_{10} & a_{20} & \ldots & a_{N0} \\ a_{11} & a_{21} & \ldots & a_{N1} \\ \vdots & \vdots & & \vdots \\ a_{1m} & a_{2m} & \ldots & a_{Nm} \end{bmatrix} = 0$$

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

Lemma 10 *Let $\boldsymbol{A}$ be a (N, c, s, t) instance of MaxLin-$\mathbb{Q}$. Let $\boldsymbol{B}$ be obtained by tensoring $\boldsymbol{A}$. Then $\boldsymbol{B}$ is a $(N^2, 1-(1-c)^2, 1-(1-s)^2, t^2)$ instance*

Proof: Suppose there is a vector $\boldsymbol{X} = (1, x_1, x_2, \ldots, x_m)$ such that $\boldsymbol{AX}$ has a zero in cN fraction of the entries. Define $x_{0j} = x_j$ and $x_{j_1j_2} = x_{j_1}x_{j_2}$ for $j_1 \geq 1$. This satisfies all but $(1-c)^2N^2$ of the equations in $\boldsymbol{B}$. It remains to show the claim about the soundness.

Suppose that for any vector $\boldsymbol{X} = (1, x_1, x_2, \ldots, x_m)$, at least s fraction of the entries in $\boldsymbol{AX}$ have magnitude greater than or equal to t. Consider any assignment to the variables $(x_{j_1j_2})$ in $\boldsymbol{B}$. Let $\boldsymbol{X}^*$ denote the matrix

$$\begin{bmatrix} 1 & x_{01} & \ldots & x_{0m} \\ x_{01} & x_{11} & \ldots & x_{1m} \\ \vdots & \vdots & & \vdots \\ x_{0m} & x_{m1} & \ldots & x_{mm} \end{bmatrix}$$

We will show that at least $(1-s)^2N^2$ entries in $\boldsymbol{AX}^*\boldsymbol{A}^T$ have magnitude $\geq t^2$. Let $\boldsymbol{X} = (1, x_{01}, x_{02}, \ldots, x_{0m})$. The vector $\boldsymbol{AX}$ has at least $(1-s)N$ 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 $(1-s)N$ 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 $(1-s)N$ 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 $(1-s)^2N^2$ entries with magnitude $\geq t^2$. □

We now define an operation called boosting. Roughly speaking, we pick σ equations at a time from the MaxLin-$\mathbb{Q}$ 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. If $\boldsymbol{A}$ has N equations, $\boldsymbol{B}$ will have $(\rho N)^\sigma$ equations. Though this suffices to prove a $2-\epsilon$ hardness factor for the halfspace problem for constant ϵ, the size of the boosted system becomes crucial when we want to achieve $\epsilon = o(1)$. Therefore, instead of generating all the N^σ possible combinations of σ equations, we pseudorandomly generate only $2^{O(\sigma)}N$ of them. To accomplish this, we construct an expander on the equations of $\boldsymbol{A}$ and perform random walks of length σ on it.

Definition 5 *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 $1 \leq i < \sigma$.*

Definition 6 *Let $\boldsymbol{A}$ be a MaxLin-$\mathbb{Q}$ instance with N equations. Let ρ, σ be two arbitrary numbers. We define the (ρ, σ)-boosting to be the MaxLin-$\mathbb{Q}$ instance $\boldsymbol{B}$ obtained as follows. Let G_N be the 5-regular Gabber-Galil graph on N vertices. Associate every vertex v of G_N 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_N 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 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.

The specific kind of expander used in 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 $O(1/\sqrt{N})$, which is a negligible increase if the soundness of the instance $\boldsymbol{A}$ were constant. Hence, we ignore this issue from now on. Before we analyze boosting, we mention some results about expanders that will be useful.

Lemma 11 *Let W denote a subset of the vertices of a regular graph G. If $|W| \geq (1-\epsilon)N$, 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. 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. Applying union bound over all 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 no more than $\sigma\epsilon$. $\Box$

Lemma 12 *[LW95, Section 15] Let W be a subset of the vertices of G_N. Let $|W| \leq sN$ for some constant s. There exists an absolute constant β such that for sufficiently large N, at most $s^{\beta\sigma}$ fraction of all walks of length σ in G_N do not contain a vertex from $\bar{W}$.*

Lemma 13 *Let $\boldsymbol{A}$ be a (N, c, s, t) MaxLin-$\mathbb{Q}$ instance. Let $\boldsymbol{B}$ be a (ρ, σ) boosting of $\boldsymbol{B}$. Then $\boldsymbol{B}$ is a $(5^{\sigma-1}\rho^{\sigma}N, 1-\sigma(1-c), s^{\beta\sigma}+\rho^{-1}, t/2)$ instance.*

The number of walks of length σ beginning from each vertex in a graph G_N 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 11, 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 $\boldsymbol{X} = (1, x_1, x_2, \ldots, x_m)$, at least sN 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 $\boldsymbol{A}_{v_1}, \boldsymbol{A}_{v_2}, \ldots, \boldsymbol{A}_{v_\sigma}$ from $\boldsymbol{A}$. Now suppose $|\boldsymbol{A}_{v_1}\boldsymbol{X}| \geq t$. Let $b \in \mathbb{Q}$. Then, for at most one value of $\rho_1 \in [\rho]$, $\rho_1\boldsymbol{A}_{v_1}\boldsymbol{X} + b$ has magnitude less than $t/2$. Therefore, for all but a $1/\rho$ fraction of $(\rho_1, \rho_2, \ldots, \rho_\sigma) \in [\rho]^{\sigma}$,

$$|(\rho_1\boldsymbol{A}_{v_1} + \rho_2\boldsymbol{A}_{v_2} + \ldots + \rho_\sigma\boldsymbol{A}_{v_\rho})\boldsymbol{X}| \geq t/2$$

If $(\boldsymbol{A}_{v_1}, \boldsymbol{A}_{v_2}, \ldots, \boldsymbol{A}_{v_\sigma})$ is a random walk on G_N, then from Lemma 12, 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 $s^{\beta\sigma}$. Therefore, at most $s^{\beta\sigma} + (1-s^{\beta\sigma})\rho^{-1} \leq s^{\beta\sigma} + \rho^{-1}$ fraction of the entries in $\boldsymbol{BX}$ have magnitude less than $t/2$. This proves the claim about the soundness. $\Box$

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

Lemma 14 *For any constants $\epsilon_0 > 0$, $0 < s < c \leq 1$ and $t > 0$, there exists a polynomial time algorithm that when given a (N, c, s, t) MaxLin-$\mathbb{Q}$ instance $\boldsymbol{A}$ as input produces a $(N_1, 1-\epsilon_0, \epsilon_0, t_1)$ instance where $t_1 > 0$ is a constant.*

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

$$\left(\frac{1-c}{1-s}\right)^L \frac{1}{\beta}\ln(2/\epsilon_0) \leq \epsilon_0$$

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

$$\sigma = \frac{\ln(2/\epsilon_0)}{\beta(1-s)^L}$$

The result is a (N_1, c_1, s_1, t_1) instance where

$$c_1 \geq 1 - \sigma(1-c)^L \geq 1 - \frac{\ln(2/\epsilon_0)}{\beta(1-s)^L}(1-c)^L \geq 1-\epsilon_0$$

and

$$(1-(1-s))^{\beta\sigma} \le (1/e)^{\beta\sigma(1-s)^L} = e^{-\ln(2/\epsilon_0)} = \epsilon_0/2$$

Therefore, $s_1 = (1-(1-s)^L)^{\beta\sigma} + \rho^{-1} \le \epsilon_0$ and $t_1 = t^L/2$. $\square$

We would like to point out that combining Lemma 14 with Lemma 15 suffices to show a $2-\epsilon$ hardness factor for the halfspace problem for any constant $\epsilon > 0$. The next theorem makes this parameter sub-constant.

Theorem 11 *There exists a $2^{(\log n)^{O(1)}}$ time reduction that when given a 5-regular graph G on n vertices outputs a MaxLin-$\mathbb{Q}$ instance $\boldsymbol{A}_2$ of size $N_2 = 2^{(\log n)^{O(1)}}$ such that*

- *If there is a vertex cover of size dn, then there is an assignment that satisfies $1 - 2^{-\Omega(\sqrt{\log N_2})}$ fraction of the equations.*
- *If every vertex cover is of size $\ge (1+\zeta)dn$, then under any assignment, at most $2^{-\Omega(\sqrt{\log N_2})}$ fraction of the equations can be satisfied within a tolerance as large as $2^{-O(\sqrt{\log N_2})}$.*

where d and ζ are the constants mentioned in Lemma 8

We first use Lemma 9 and Lemma 14 to convert a vertex cover instance to a $(N_1, 1-\epsilon_0, \epsilon_0, t_1)$ MaxLin-$\mathbb{Q}$ instance $\boldsymbol{A}_1$. We alternately tensor and boost $\boldsymbol{A}_1$ so that the soundness stays at ϵ_0, but the completeness progressively comes closer to 1. As a final step, we boost once more so that the completeness is $1-\epsilon$ and the soundness is ϵ for a small value ϵ as desired.

Proof: Fix ϵ_0 large enough such that

$$\frac{\epsilon_0 \log(\epsilon_0/2)}{\beta \log(2\epsilon_0 - \epsilon_0^2)} \le 1/2$$

Fix σ_0 such that

$$\sigma_0 = \frac{\log(\epsilon_0/2)}{\beta \log(2\epsilon_0 - \epsilon_0^2)}$$

and let $\rho_0 = 2/\epsilon_0$.

We first use Lemma 9 and Lemma 14 to convert the graph to a $(N_1, 1-\epsilon_0, \epsilon_0, t_1)$ MaxLin-$\mathbb{Q}$ instance $\boldsymbol{A}_1$, where $N_1 = n^{O(1)}$. Suppose $\boldsymbol{B}_1$ is the result of tensoring and (ρ_0, σ_0)-boosting $\boldsymbol{A}_1$ once. Then $\boldsymbol{B}_1$ is a $(O(N_1)^2, 1-\sigma_0\epsilon_0^2, (1-(1-\epsilon_0)^2)^{\beta\sigma_0} + \rho_0^{-1}, t_1^2/2)$ instance. Note that $\sigma_0\epsilon_0 \le 1/2 < 1$ and $(1-(1-\epsilon_0)^2)^{\beta\sigma_0} + \rho_0^{-1} = (2\epsilon_0 - \epsilon_0^2)^{\beta\sigma_0} + \rho_0^{-1} \le 2/\epsilon_0 + 2/\epsilon_0 = \epsilon_0$. This implies that after one round of tensoring and boosting, the completeness comes closer to 1 and the soundness does not get worse. Now, let $\boldsymbol{A}_2$ be the result of repeatedly tensoring and (ρ_0, σ_0)-boosting $\boldsymbol{A}_1$ l times. Let $L = 2^l$. Then $\boldsymbol{A}_2$ is a (N_2, c_2, s_2, t_2) instance where $N_2 = O(N_1)^L$, $c_2 = 1 - O(1)^L$, $s_2 = \epsilon_0$ and $t_2 = \Omega(1)^L$.

As a final step, we now use (ρ_2, σ_2)-boosting on $\boldsymbol{A}_2$ where $\rho_2 = 2.2^L$, $\sigma_2 = \dfrac{1+L}{\beta \log(1/\epsilon_0)}$. This produces a (N_3, c_3, s_3, t_3) instance where $N_3 = O(\rho_2)^{\sigma_2} N_2 = 2^{O(L^2)} N_1^L$, $c_3 = 1 - O(L)O(1)^L = 1 - O(1)^L$, $s_3 = \epsilon_0^{\beta\sigma_2} + \rho_2^{-1} \le 2^{-L}$, and $t_3 = \Omega(1)^L$. Choose $L = \log N_1$. Then $\log N_3 = O(\log^2 N_1)$, which implies $L = \Omega(\sqrt{\log N_3})$. That is, $\boldsymbol{A}_3$ is a $(N_3, 1 - 2^{-\Omega(\sqrt{\log N_3})}, 2^{-\Omega(\sqrt{\log N_3})}, 2^{-O(\sqrt{\log N_3})})$ instance. $\square$

4.3 From MaxLin-$\mathbb{Q}$ to the Halfspace Problem

Lemma 15 *There exists a polynomial time algorithm that when given a (N, c, s, t) instance $\boldsymbol{A}$ of MaxLin-$\mathbb{Q}$ produces a instance of the halfspace problem with $2N$ points such that:*

- *If there is a solution to the MaxLin-$\mathbb{Q}$ instance that satisfies $\ge cN$ of the equations, there is a halfspace that correctly classifies $\ge 2cN$ of the points.*

- *If $\boldsymbol{A}$ has soundness s under tolerance t, then no halfspace can correctly classify more than $(1+s)N$ of the points.*

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

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

for any $t' \in \mathbb{Q}$ satisfying $0 < t' \leq t$. Homogenizing the above, one can rewrite $\boldsymbol{A}$ as a system of $2N$ inequalities

$$\begin{array}{c}(a_{i0}+t')x_0 + \sum_{j=1}^{n} a_{ij}x_j > 0 \\ (a_{i0}-t')x_0 + \sum_{j=1}^{n} a_{ij}x_j < 0\end{array} \tag{4}$$

where $i \in \{1,2,\ldots,N\}$. If we could satisfy cN equations in $\boldsymbol{A}$, the new system has a solution satisfying $2cN$ inequalities by setting x_0 to 1. Note that if we set x_0 to some value ≤ 0, then we can not satisfy more than half of the inequalities. Suppose there is a solution satisfying $(1+s)N$ of the inequalities in (4). Since $x_0 > 0$, 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 s fraction of the equalities within tolerance $t' \leq t$.

Select a value of t' such that $0 < t' \leq t$ and $t' \notin \{\pm a_{i0}\}_{i=1,2\ldots N}$. The second condition on t' ensures that the coefficient of x_0 in none of the $2N$ inequalities of (4) is zero. Now divide each inequality by the coefficient of x_0 and flip the direction of the inequality if we divided by a negative number. This way, we can convert the system (4) to an equivalent system of $2N$ inequalities, where each inequality is of the form

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

where $i \in \{1,2,\ldots,2N\}$. We now define the halfspace instance. The halfspace instance produced is over $\mathbb{R}^n$. 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^-.

Suppose that there is an assignment $(x_0, x_1, \ldots, x_n)$ satisfying $2cN$ inequalities in (4). Then the hyperplane $x_0 + \sum_{j=1}^{n} x_j h_j = 0$ correctly classifies cN points in (S^+, S^-).

Now suppose there is a hyperplane $x_0 + \sum_{j=1}^{n} x_j h_j = 0$ that correctly classifies $(1+s)N$ points in (S^+, S^-) for some $s > 0$. Clearly, $x_0 > 0$. Scale the x_i so that $x_0 = 1$. Now, $(x_1, x_2, \ldots, x_n)$ is an assignment satisfying $(1+s)N$ inequalities in (5), and equivalently in (4). This completes the proof of the lemma. $\square$

We can now prove Theorem 4.

Proof: We give a reduction from the vertex cover problem on 5-regular graphs mentioned in Lemma 8. 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 11 to produce a (N_2, c_2, s_2, t_2) MaxLin-$\mathbb{Q}$ instance $\boldsymbol{A}$, where $c_2 = 1-\epsilon$, $s_2 = \epsilon$, $\epsilon = 2^{-\Omega(\sqrt{\log N_2})}$ and $t = 2^{-O(\sqrt{\log N_2})} > 0$. We transform $\boldsymbol{A}_2$ to a halfspace instance (S^+, S^-) as described in Lemma 15. Note that $N' = |S^+| + |S^-| = 4N_2$. Now, if there is a vertex cover of size $\leq dn$ in G, there is a halfspace that correctly classifies c_2 fraction of the points. On the other hand, if there is no vertex cover of size smaller than $(1+\zeta)dn$ in G, there is no halfspace that correctly classifies $\geq 1/2(1+s_2)$ fraction of the points.

Therefore the gap obtained is $c_2/(1/2(1+s_2)) = 2(1-\epsilon)/(1+\epsilon) = 2(1-O(\epsilon)) = 2 - 2^{-\Omega(\sqrt{\log N'})}$. $\square$

5 Thresholds of Halfspaces are not PAC-learnable

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

Definition 7 *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 1 *There does not exist a randomized polynomial time algorithm for n^8-unique SVP.*

Under the above assumption, Goldreich et al. [GGH97] construct a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ and show it to be secure. Here, $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$ are the key generation, encryption and decryption algorithms respectively. The encryption scheme of Goldreich et al. [GGH97] builds on that of Ajtai-Dwork [AD97]. We use the former because decryption in that scheme is error-free. This is particularly convenient for getting hardness results for noise-free learning models. We next briefly describe the encryption scheme of Goldreich et al. [GGH97].

$\mathcal{K}$ when given a security parameter n as input generates a uniformly random vector $u \in \mathbb{R}^n$ from the unit sphere in n dimensions. The coordinates of u and all real numbers that follow need to be only specified up to n bits of precision. Let $m = n^3$. It then picks random vectors $a_1, a_2, \ldots, a_m$ from the set $\{x | x.u \in \mathbb{Z}\} \bigcap B$, where B is a ball of "big" radius $R = 2^{O(n \log n)}$ centered at the origin. Each vector a_i is perturbed by a "small" vector δ_i to obtain vector v_i. Let i_0 denote the smallest index i for which the parallelepiped generated by vectors $v_{i+1}, v_{i+2}, v_{i+n}$ has width $n^{-2}R$. It is shown in [AD97] that such an index exists with high probability and is less than $m/2$. Let $w_1 = v_{i_0+1}, w_2 = v_{i_0+2}, \ldots, w_n = v_{i_0+n}$. Let i_1 be an index such that $u.a_{i_1}$ is an odd integer (such an index exists with probability close to $1 - 2^{-m}$). Then, (u, e) is the private-key public-key pair where $e = (v_1, v_2, \ldots, v_m, i_0, i_1)$.

To encrypt a 0, $\mathcal{E}$ picks bits $b_1, b_2, \ldots, b_m$ at random and reduces the vector $v = \sum_{i=1}^m b_i v_i$ modulo the parallepiped P generated by $w_1, w_2, \ldots, w_n$. By reducing v modulo P, we mean finding a vector $v' \in P$ such that $v = v' + \sum c_i w_i$, where $c_i \in \mathbb{Z}$. The cipher-text is then the vector $v' \in P$. To encrypt a 1, $\mathcal{E}$ picks bits $b_1, b_2, \ldots, b_m$ as before. The vector v in this case is computed as $v_{i_1}/2 + \sum_{i=1}^m b_i v_i$. The cipher-text v' is once again computed as v modulo P. Let $E_e(x)$ denote the distribution on cipher-texts when the bit x is encrypted by algorithm $\mathcal{E}$ using public key e.

To decrypt a cipher-text v' using private key u, Algorithm $\mathcal{D}$ computes $u.v'$. If $u.v' \in \mathbb{Z} + [-2/n, 2/n]$, v' is decrypted as a 0 (If $A, B \subseteq \mathbb{R}$, we denote by $A + B$ the set $\{a + b | a \in A, b \in B\}$). Otherwise, $u.v'$ is guaranteed to satisfy $u.v' \in \mathbb{Z} + 1/2 + [-2/n, 2/n]$ and v' is decrypted as a 1. Goldreich et al. [GGH97] prove the following about the encryption scheme.

Lemma 16 *Suppose there exists a randomized polynomial time algorithm $\mathcal{D}'$ that when given (e, p) as input, where e is a random public key generated by Algorithm $\mathcal{K}$ on parameter n, and p is either the encryption of a 0 or 1 outputs a bit denoted by $\mathcal{D}'(e, p)$ satisfying*

$$\Pr[\mathcal{D}'(e, E_e(0)) = 0] - \Pr[\mathcal{D}'(e, E_e(1)) = 0] \geq \epsilon(n)$$

where $\epsilon(n)$ is an inverse polynomial in n, and the probabilities are taken over the key generation, encryption and the computation of $\mathcal{D}'$. Then n^8-unique SVP has a randomized polynomial time algorithm.

Definition 8 *An unweighted threshold gate (or majority gate) is a gate that when given $(x_1, x_2, \ldots, x_k) \in \{0, 1\}^k$ 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.*

Lemma 17 *For any key to the encryption scheme of [GGH97], the decryption can be done by a TC_2^0 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 $x_i = \sum_j 2^j x_{ij}$. Then the decoding algorithm $\mathcal{D}$ just computes $u.x$ and checks if it is strictly within $2/n$ of an integer. The quantity $u.x$ can be computed as

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

Let f_{ij} denote the fractional part of $u_i 2^j$. Then the fractional part of $u.x$ 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 $N.1/n^4 = O(1/n^2)$ 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_2^0 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. $\square$

We use the notation of Kearns and Valiant [KV89] to define our learning problem for depth 2 threshold circuits.

Definition 9 *We say the concept class C is weakly PAC learnable if there exists a randomized polynomial time algorithm $\mathcal{A}$ and a polynomially evaluatable hypothesis class H with the following properties:*

- *(Input:) The algorithm is given access to oracles POS and NEG that generate points from some distributions D_c^+ and D_c^- over the positive and negative examples respectively of a concept c. It is also given a confidence parameter δ.*
- *(Output:) With probability $1 - \delta$, the algorithm outputs a hypothesis $h \in H$ such that*

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

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

for some polynomial $p(n)$.

Theorem 12 *If the concept class TC_2^0 is weakly PAC learnable, then there exists a polynomial time randomized algorithm for n^8-unique SVP.*

Proof: Suppose there exists a weakly PAC learning algorithm $\mathcal{A}$ for the class TC_2^0. We construct an algorithm $\mathcal{D}'$ for decoding a bit encrypted using the scheme of Goldreich et al [GGH97] 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 $E_e(0)$ and $E_e(1)$ 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 that there is a TC_2^0 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 $\geq 1-\delta$, Algorithm $\mathcal{A}$ outputs a hypothesis h such that

$$\Pr[h(x)=0|x \in_R E_e(0)] - \Pr[h(x)=0|x \in_R E_e(1)] \geq \epsilon(n)$$

for some $\epsilon(n)$ that is inverse polynomial in n. 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:

$$\begin{aligned}&\Pr[\mathcal{D}' \text{ outputs } 0 \text{ when given random encryption of } 0]\\ &\quad - \Pr[\mathcal{D}' \text{ outputs } 0 \text{ when given random encryption of } 1] \geq 2(1-\delta)\epsilon(n)\end{aligned}$$

Using Lemma 16, this implies an algorithm for n^8-unique SVP. □

5.1 Further Hardness of Learning Halfspaces with Adversarial Noise

We now show that our result on hardness of learning TC_2^0 circuits easily 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.* [HMP$^+$93].

Lemma 18 ([HMP$^+$93]) *For any Boolean functions $g_1, \ldots, g_k$ on X, $f = MAJ(g_1, g_2, \ldots, g_k)$ and any distribution $\mathcal{D}$ on X there exists $i \leq k$ such that $|\Pr_{\mathcal{D}}[f = g_i] - \frac{1}{2}| \geq \frac{1}{2k}$.*

If it holds that $\Pr_{\mathcal{D}}[f = g_i] \geq \frac{1}{2} + \frac{1}{2k}$, then examples of f drawn from distribution $\mathcal{D}$ can be seen as examples of g_i with adversarial noise of rate $\frac{1}{2} - \frac{1}{2k}$. Similarly, if $\Pr_{\mathcal{D}}[f = g_i] \leq \frac{1}{2} - \frac{1}{2k}$, 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_2^0 circuit is equivalent to a halfspace with adversarial noise. Hence Theorem 5 implies Theorem 6.

6 Conclusions

We have shown connections between some well-studied open problems in 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. 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. Indeed it is plausible that there exists a better algorithm than exhaustive search for this variant of the problem, as in the case of (unrestricted) noisy parity [BKW03].

For halfspaces, a natural question is whether one can extend our hardness result for learning halfspaces to more general concept classes. One possible generalization would be to allow the sign of a low-degree polynomial as hypothesis. Kalai et al. [KSS94] use this hypothesis class to design algorithms for agnostic learning of halfspaces under some natural distributions. Similarly, for the problem of learning parity with adversarial noise, one could allow the algorithm to produce a low degree polynomial over $\mathbb{Z}_2$ as hypothesis. To the best of our knowledge, there are no negative results known for these problems.

Acknowledgments

We would like to thank Avrim Blum and Leslie Valiant for useful comments and suggestions.

References

[ABF$^+$04] M. Alekhnovich, M. Braverman, V. Feldman, A. Klivans, and T. Pitassi. Learnability and automatizability. In *Proc. 45^{th} IEEE Symp. on Foundations of Computer Science*, 2004.

[ABFR91] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. In *Proc. 23^{rd} ACM Symp. on Theory of Computation*, pages 402–409, 1991.

[ABSS97] S. Arora, L. Babai, J. Stern, and E.Z. Sweedyk. The hardness of approximate optima in lattices, codes and systems of linear equations. *J. Comput. Syst. Sci.*, 54:317–331, 1997.

[AD97] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proc. 29th ACM Symposium on the Theory of Computing*, pages 284–293, 1997.

[Agm64] S. Agmon. The relaxation method for linear inequalities. *Canadian J. of Mathematics*, 6(3):382–392, 1964.

[AK95] E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147(1&2):181–210, 1995.

[ALM$^+$98] S. Arora, C. Lund, R. Motawani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. of the ACM*, 45(3):501–555, 1998.

[Ama94] E. Amaldi. From finding feasible susbsystems of linear systems to feedforward neural network design. *Ph.D Thesis, Swiss Federal Institute of Technology at Lausanne (EPFL)*, 1994.

[BB02] N. Bshouty and L. Burroughs. Maximizing agreements and coagnostic learning. In *Proceedings of ALT '02*, pages 83–97, 2002.

[BDEL00] S. Ben-David, N. Eiron, and P. M. Long. On the difficulty of approximately maximizing agreements. In *Proceedings of COLT '00*, pages 266–274, 2000.

[BF02] N. Bshouty and V. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machince Learning Research*, 2:359–395, 2002.

[BFKV96] A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. In *Proc. 37^{th} IEEE FOCS*, pages 330–338, 1996.

[BK97] A. Blum and R. Kannan. Learning an intersection of a constant number of halfspaces over a uniform distribution. *J. Comput. Syst. Sci.*, 54(2):371–380, 1997.

[BKW03] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. of the ACM*, :50(4):506–519, 2003.

[BL97] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[BRS91] R. Beigel, N. Reingold, and D. A. Spielman. The perceptron strikes back. In *Structure in Complexity Theory Conference*, pages 286–291, 1991.

[Coh97] E. Cohen. Learning noisy perceptrons by a perceptron in polynomial time. In *Proc. 38^{th} IEEE Symp. on Foundations of Computer Science*, pages 514–523, 1997.

[DMS99] I. Dumer, D. Micciancio, and M. Sudan. Hardness of approximating the minimum distance of a linear code. In *Proc. 40^{th} IEEE Symp. on Foundations of Computer Science*, 1999.

[Fel06a] V. Feldman. On Attribute Efficient and Non-adaptive Learning of Parities and DNF Expressions, 2006. Manuscript (preliminary version appeared in proceedings of COLT '05).

[Fel06b] V. Feldman. Optimal hardness results for maximizing agreements with monomials. *Electronic Colloquium on Computational Complexity (ECCC)*, (032), 2006.

[Fre90] Y. Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 202–216, 1990.

[Fre92] Y. Freund. An improved boosting algorithm and its implications on learning complexity. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 391–398, 1992.

[Gal90] S. Galant. Perceptron based learning algorithms. *IEEE Trans. on Neural Networks*, 1(2), 1990.

[GGH97] O. Goldreich, S. Goldwasser, and S. Halevi. Eliminating decryption errors in the ajtai-dwork cryptosystem. In *Advances in Cryptology, Proc. of Crypto '97, LNCS, Springer Verlag.*, 1997.

[GJ79] M. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman, San Francisco, 1979.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of STOC '89*, pages 25–32, 1989.

[GR06] V. Guruswami and P. Raghavendra. Hardness of learning halfspaces with noise. In *Manuscript*, April 2006.

[Hås97] J. Håstad. Some optimal inapproximability results. In *Proc. 29^{th} Annual ACM Symposium on Theory of Computing*, pages 1–10, 1997. El Paso, Texas, 4–6 May 1997.

[Hau92] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

[HER04] J. Holmerin, L. Engebretsen, and A. Russell. Inapproximability results for equations over finite groups. *Theoretical Computer Science*, 312(1):17-45, 2004.

[HMP$^+$93] A. Hajnal, W. Maass, P. Pudlak, G. Turan, and M. Szegedy. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, Volume 46(2):129 – 154, 1993.

[HSH95] K. Hoffgen, H. Simon, and K. Van Horn. Robust trainability of single neurons. *Journal of Computer and System Sciences*, 50(1):114–125, 1995.

[Jac97] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55:414–440, 1997.

[Kha95] M. Kharitonov. Cryptographic lower bounds for learnability of boolean functions on the uniform distribution. *J. Comput. Syst. Sci.*, 50(3):600–610, 1995.

[KKMS05] A. T. Kalai, A. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. In *Proc. 46^{th} IEEE Symp. on Foundations of Computer Science*, pages 11–20, 2005.

[KL93] M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.

[KM91] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. In *Proceedings of STOC '91*, pages 455–464, 1991.

[KOS02] A. Klivans, R. O'Donnell, and R. Servedio. Learning intersections and thresholds of halfspaces. In *Proc. 43^{rd} IEEE Symp. on Foundations of Computer Science*, pages 177–186, 2002.

[KP06] S. Khot and A. K. Ponnuswami. Better inapproximability results for maxclique, chromatic number and min-3lin-deletion. In *Proc. ICALP'06*, 2006. To appear.

[KS01] A. Klivans and R. Servedio. Learning dnf in time $2^{n^{1/3}}$. In *Proc. 33^{rd} ACM Symp. on the Theory of Computing*, pages 258–265, 2001.

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

[KS06a] A. R. Klivans and A. Sherstov. Improved lower bounds for learning intersections of halfspaces. In *Proc. 19^{th} Conference on Learning Theory (COLT)*, 2006.

[KS06b] A. R. Klivans and A. Sherstov. Personal communication, 2006.

[KSS94] M. Kearns, R. Schapire, and L. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2/3):115–142, 1994.

[KV89] M. Kearns and L. G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. In *Proc. 21^{st} ACM Symp. on Theory of Computing*, pages 433–444, 1989.

[KV94] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[Lev93] L. Levin. Randomness and non-determinism. *Journal of Symbolic Logic*, 58(3):1102–1103, 1993.

[LMN93] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

[LW95] M. Luby and A. Wigderson. Pairwise independence and derandomization. Technical Report 95-035, International Computer Science Institute, 1995.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[MOS03] E. Mossel, R. O'Donnell, and R. Servedio. Learning juntas. In *Proc. 35th Ann. ACM Symp. on the Theory of Computing, 2003.*, 2003.

[MP69] M.Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[PV88] L. Pitt and L.G. Valiant. Computational limitations on learning from examples. *J. of the ACM*, 35:965–984, 1988.

[PY91] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43:425–440, 1991.

[Ros64] F. Rosenblatt. *Principles of Neurodynamics*. Spartan Books, 1964.

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

[Val85] L. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 560–566, 1985.

[Vem04] S. Vempala. *The Random Projection Method*. AMS, 2004.