Efficiency and Computational Limitations of Learning Algorithms

A thesis presented

by

Vitaly Feldman

to

The Division of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

January 2007

Thesis advisor | Author
Leslie G. Valiant | Vitaly Feldman

Efficiency and Computational Limitations of Learning Algorithms

Abstract

This thesis presents new positive and negative results concerning the learnability of several well-studied function classes in the Probably Approximately Correct (PAC) model of learning.

Learning Disjunctive Normal Form (DNF) expressions in the PAC model is widely considered to be the main open problem in Computational Learning Theory. We prove that PAC learning of DNF expressions by an algorithm that produces DNF expressions as its hypotheses is NP-hard. We show that the learning problem remains NP-hard even if the learning algorithm can ask membership queries. We also prove that with an additional restriction on the size of hypotheses the learning remains NP-hard even with respect to the uniform distribution. These last two negative results are the first for learning in the PAC model with membership queries that are not based on cryptographic assumptions.

We complement the hardness results above by presenting a new algorithm for learning DNF expressions with respect to the uniform distribution using membership queries. Our algorithm is attribute-efficient, noise-tolerant, and uses membership queries in a non-adaptive way. In terms of running time, it substantially improves on the best previously known algorithm of Bshouty *et al.*

Learning of parities with random noise with respect to the uniform distribution is a famous open problem in learning theory and is also equivalent to a major open problem in coding theory. We show that an efficient algorithm for this problem would imply efficient algorithms for several other key learning problems with respect to the uniform distribution. In particular, we show that agnostic learning of parities (also referred to as learning with adversarial noise) reduces to learning parities with random classification noise. Together with the parity learning algorithm of Blum *et al.*, this gives the first non-trivial algorithm for agnostic learning of parities. This reduction also implies that learning of DNF expressions reduces to learning noisy parities of just logarithmic number of variables.

A monomial is a conjunction of (possibly negated) Boolean variables and is one of the simplest and most fundamental concepts. We show that even weak agnostic learning of monomials by an algorithm that outputs a monomial is NP-hard, resolving a basic open problem in the model.

The proposed solutions rely heavily on tools from computational complexity and yield solutions to a number of problems outside of learning theory. Our hardness results are based on developing novel reductions from interactive proof systems for NP and known NP-hard approximation problems. Reductions and learning algorithms with respect to the uniform distribution are based on new techniques for manipulating the Fourier Transform of a Boolean function.

Bibliographic Note

Most of the research that appears in this thesis was published elsewhere in some form.

Chapter 3 is based on parts of the papers "Learnability and Automizability" and "Hardness of Approximate Two-level Logic Minimization and PAC Learning with Membership Queries". The paper "Learnability and Automizability" is a joint work with Misha Alekhnovich, Mark Braverman, Adam Klivans, and Toni Pitassi. It appeared in the Proceedings of 45th IEEE Symposium on Foundations of Computer Science, 2004 [ABF^{+}04].

The paper "Hardness of Approximate Two-level Logic Minimization and PAC Learning with Membership Queries" is also the basis of Chapter 4 and appeared in the Proceedings of 38th ACM Symposium on Theory of Computing, 2006 [Fel06a].

Chapter 5 is based on the paper "On Attribute Efficient and Non-adaptive Learning of Parities and DNF Expressions" which appeared in the Proceedings of 18th Annual Conference on Computational Learning Theory, 2005 [Fel05].

Chapter 6 is based on a part of the paper "New Results for Learning Noisy Parities and Halfspaces" which is a joint work with Parikshit Gopalan, Subhash Khot, and Ashok Ponnuswami and appeared in the Proceedings of 47th IEEE Symposium on Foundations of Computer Science, 2006 [FGKP06].

Chapter 7 is based on the paper "Optimal Hardness Results for Maximizing Agreements with Monomials" which appeared in the Proceedings of 21st Annual IEEE Computational Complexity Conference, 2006 [Fel06b].

Contents

Acknowledgements

I am deeply grateful to Leslie Valiant, my advisor, for his endless support and encouragement of my work. I was extremely fortunate to have both his trust in my independent work and his advice whenever I needed it. Les' suggestions and insights are the basis of some of the most important fruits of my research and his own work is always a great source of inspiration.

I thank Nader Bshouty, my advisor during my graduate studies at the Technion. Nader introduced me to research in computational learning theory and was very supportive during my first steps as a researcher. He has also provided me with a concise formula for success in research: "Read papers, write papers! Do nothing else!". I often recall it when failing to implement it.

I thank my collaborators Misha Alekhnovich[1], Mark Braverman, Parikshit Gopalan, Subhash Khot, Adam Klivans, Toni Pitassi, and Ashok Ponnuswami. Their brilliant ideas have greatly contributed to results presented in Chapters 3 and 6. I would also like to thank many colleagues for illuminating discussions and valuable comments on my work. I have especially benefited from the friendship and advice of Rocco Servedio.

Harvard's Division of Engineering and Applied Sciences has been a stimulating and enjoyable place to engage in graduate studies. I am grateful to the wonderful teachers I had during these years: Silvio Micali, Michael Mitzenmacher, Avi Pfeffer, Michael Rabin, Peter Shor, Madhu Sudan, and especially Salil Vadhan for sharing their outstanding knowledge and enthusiasm for science. I would like to thank Avi Pfeffer, Michael Rabin, and Salil Vadhan for also serving on my thesis committee. I thank fellow graduate students Adi Akavia, Yan-Cheng Chang, Kai-Min Chung, Eleni Drinea, Kobi Gal, Alex Healy, Shaili Jain, Minh Huyen-Nguyen, Adam Kirsch, Loizos Michael, Shien Jin Ong, and Emanuele Viola for their friendship and numerous interesting discussions. My special thanks to Alex and Shaili for the careful proofreading and insightful comments on my manuscripts.

I would like to thank my friends both local and distant (and, fortunately, too numerous to list here). Their friendship, unique personalities, and diverse talents have made these years an extraordinary and memorable experience. In particular, my time here would not have been the same without the friendship of my long-time roommates Matthias Röder and Sasha Tokovinine with whom I shared lots of great times and countless conversations on the "meaning of it all".

Finally, my biggest thanks are reserved for my close family – my mother Tatiana, my father Georgiy, my sister Polina, my brother Vladik, and my beloved girlfriend Polinka for their boundless love, care, and understanding. I owe them much more than I could possibly describe. I also feel greatly indebted to Marian Tkach, my high school math teacher, for igniting and nourishing my interest in mathematics as well as for being a close friend and in many ways a personal example for the last 15 years.

[1]Sorrowfully, my thanks are posthumous. Misha died in a tragic accident on August 5, 2006.

This thesis is dedicated to my parents and to Marian Tkach

Chapter 1

Introduction

1.1 Background

1.1.1 Computational Learning Theory

Learning, or the ability to derive useful functionality through experience, is a fundamental computational phenomenon. It is widely believed to be the basis of most human cognitive abilities and is exhibited by a variety of other living organisms. The study of natural learning phenomena and the realization of learning behavior by machines in a wide range of real-life problems are among the most important modern scientific pursuits.

Computational learning theory is the study of learning from the viewpoint of the theory of computation. Its goal is to understand the capabilities and inherent limitations of *learning algorithms* that can be performed by computers as well as to shed light on natural learning phenomena. The basis of this study and one of its important products are formal mathematical models of learning. The great diversity and complexity of real-life learning phenomena make defining realistic and general learning models a very challenging task.

A major step toward more realistic learning models was made by Leslie Valiant who introduced the Probably Approximately Correct (PAC) learning model [Val84]. Unlike previous models, such as identification in the limit [Gol78] and statistical learning theory, the PAC model was defined to study *efficient* (and hence more realistic) learning algorithms. Valiant's elegant and general paradigm of learning from examples was quickly embraced by numerous researchers within both the theory of computer science and machine learning. Their research has lead to a rich theory with connections to many other areas of computer science such as computational complexity theory, statistical pattern recognition, cryptography, and information theory. This theory provides a sound basis for applied machine learning research and has yielded important algorithmic techniques used in practical learning systems.

1.1.2 The PAC Model

In the PAC model the functionality that is being learned is modeled as a binary classifier function (or a concept) over some domain, referred to as the *target concept*. For example, this concept could be "Is it going to rain tomorrow?" defined on the domain of all the possible measurements of temperature, humidity, wind speed, etc. , taken today. A learning algorithm is then given access to points in the domain randomly chosen from some unknown distribution together with values of the target concept on these points, i.e. *examples*. In the above scenario this would correspond to having historical information on both the measurements and the rain. The goal of the learning algorithm is to predict with "high" accuracy the target concept on points it has not previously seen (the accuracy is measured with respect to the same unknown distribution). Note that if we do not make any assumptions on the target concept then its values on the unseen points are not restricted in any way and therefore any non-trivial prediction would be impossible. A general way to model such assumptions suggested by Valiant is to restrict the target concept to belong to a certain *concept class* which is simply a set of functions over the domain. A concept class is considered learnable if there exists an algorithm that can successfully learn every concept from the concept class. In our example one might assume that the points corresponding to positive predictions about the rain form a (multidimensional) ball in the space of all measurements. A learning algorithm for the concept class of all the possible balls could then be used to automatically find the unknown ball with high accuracy.

A major emphasis in Valiant's model is placed on the computational efficiency of the learning algorithms. Specifically, it asks which concept classes can be learned in time polynomial in the description of the problem and the inverse of the desired accuracy (see Section 2.2 for a formal description of the model). Such restrictions are necessary from both practical and theoretical perspectives. From the practical perspective, dealing with large domains and a huge number of candidate functions is only possible by efficient learning algorithms. From the theoretical viewpoint, this restricts the discussion to algorithms that "reveal" the structure of the problem instead of exhaustively searching through all candidate concepts.

The basic PAC model should not be thought of as the universal learning model that applies to any learning scenario (and it was not intended as such [Val84]). Indeed, in most practical settings some of the generality of the model is not required and/or some of the assumptions do not hold [Hau90]. Instead, it provides a general framework for thinking about learning problems that emphasizes the computational problem of finding a "suitable" hypothesis in a large (often exponentially large in the learning parameters) space of candidate hypotheses given examples of the target function. Based on this paradigm, researchers in learning theory have introduced several variants of the basic PAC model a as well as number of other learning models that reflect a wide variety of practical considerations. For example presence of noise is studied by considering various ways to corrupt the examples available to the learning algorithm [AL88, Hau92, Kea98, KL93, KSS94, Val85]; learning in the online setting (in which the learning algorithm updates its hypothesis after each inconsistency is discovered) is studied in the *online mistake-bound* model of Angluin [Ang88] and Littlestone [Lit88]; exact identification of the target function via different types of queries is studied in models introduced by Angluin [Ang87, Ang88] (Sections 2.2.2, 2.2.3, and 2.2.4 describe the models relevant to this thesis; we recommend the textbook by Kearns and Vazirani [KV94b] for a detailed treatment of these models). It is important to note that an array of results is known for automatically transforming between learning algorithms in different models and numerous algorithmic ideas and techniques reappear in several of these seemingly unrelated settings (*cf.* [DHW91, KV94b]). This unusually favorable situation corroborates the universality of the basic insights of Valiant's model and motivates further research into the foundations of computational learning theory.

Besides "passive" learning in which the learner has no influence over the choice of examples, learning theory also studies models that allows the learner to ask questions about the unknown function or perform tests on it. This ability is formalized by allowing the learning algorithm to access a *membership query oracle* [Val84]. Upon request of the learner, this oracle returns the value of the target concept on any desired element of the domain. A fundamental question in learning theory is when and how can asking questions help to learn. We address this questions in Chapters 3, 4 and 5.

For a specific learning model and an input domain, the main question addressed in learning theory is which "interesting" concept classes over the domain are efficiently learnable. The domain most widely studied is the set of all objects that can be described using a fixed set of binary features (also referred to as *variables* and *attributes*), or simply $\{0,1\}^n$, where n is the number of features. Many other domains can be naturally reduced to this relatively simple setting. Among the most natural and well-studied concept classes for this domain are (see Section 2.1 for formal definitions of these concept classes):

- *monomials*, or conjunctions of (possibly negated) variables;
- *disjunctive normal form*, or DNF, expressions (which are ORs of monomials);
- *linear threshold functions* (also called *halfspaces*);
- neural networks, or *threshold circuits*;
- *decision trees*;
- *k-juntas*, or functions that depend on only k variables;
- *parities*, or XORs of subsets of the variables.

1.1.3 Learning DNF Expressions

While in this research we address learnability of several of the above concept classes, the primary focus will be on the learnability of DNF expressions. Therefore, we now briefly and informally survey some of the results relevant to this problem. A more thorough treatment of the relevant previous work is postponed until after the models are defined formally and appears in the introductory section of each chapter.

DNF expressions are a succinct way to represent a variety of concepts and are relatively easily understood by humans [Jac95, Hel01]. They are also commonly used to represent Boolean functions in computer hardware where they are referred to as *two-level logic* [UVSV06]. The significance of learning these expressions given random examples of the function was already recognized in Valiant's first paper on the PAC model [Val84] and has since been one of the most well-studied problems in learning theory. Despite numerous attempts to find efficient algorithms for learning DNF in the basic PAC model, only DNF expressions with a constant number of monomials (referred to as the *terms of a DNF expression*) are known to be learnable in polynomial time [Val85]. In fact, even a seemingly much simpler question: "Are $\omega(1)$-juntas efficiently learnable when the distribution of examples is uniform over $\{0,1\}^n$?" is still unresolved (note that ℓ-juntas are a subclass of DNF with 2^ℓ terms).

The lack of progress on the original question has lead to efforts to resolve its restricted or related versions. The variants of the problem that are often considered involve one or more of the following modifications:

1. bound the number terms;
2. bound the number of variables in each term;
3. bound the number of times a variable can appear in the expression;
4. allow only monotone DNF (i.e. without negations);
5. assume that the distribution of examples is uniform (or product);
6. allow membership queries;
7. allow only *proper* learning algorithms, or algorithms that output hypotheses in DNF representation;
8. consider algorithms that run in superpolynomial time (but are more efficient than the straightforward exhaustive search).

Some of the most important and interesting results obtained for these restrictions are as follows.

(1,2) Both DNF expressions with at most k terms and DNF expressions with at most k variables in each term are efficiently learnable for constant k [Val84].

(4+6) Valiant has showed that monotone DNF expressions are learnable using membership queries [Val84].

(4) Kearns *et al.* showed a simple reduction implying that in the basic PAC model learning of monotone DNF expressions is equivalent to learning unrestricted DNF expressions [KLPV87b].

(1+7) Pitt and Valiant showed that learning k-term DNF expressions by an algorithm that outputs a k-term DNF expression as its hypothesis is NP-hard for every $k \geq 2$ [PV88].

(3+5) Hancock proved that DNF formulas in which each variable appears at most k times for a constant k are learnable with respect to the uniform distribution [Han92].

(5+8) Verbeurgt gave an algorithm that learns s-term DNF expressions with respect to the uniform distribution in time $n^{O(\log s)}$.

(6) Angluin and Kharitonov have proved that under certain cryptographic assumptions membership queries do not help to learn unrestricted DNF expressions [AK95b].

(5+6) Jackson has showed that unrestricted DNF expressions are efficiently learnable if the underlying distribution of examples is uniform and membership queries are allowed [Jac97]. The running time of his algorithm was significantly improved by Bshouty *et al.* [BJT04] and Klivans and Servedio [KS03]. Bshouty *et al.* showed that a similar approach can be used to learn from examples on points given by a random walk on $\{0,1\}^n$ (instead of membership queries) [BMOS03].

(1+6) Blum and Rudich proved that $O(\log n)$-term DNF expressions are learnable with membership queries [BR95].

(1+4+5) Sakai and Maruoka gave an algorithm that learns $O(\log n)$-term monotone DNF expressions with respect to the uniform distribution (and is also proper) [SM00]. This was improved by Bshouty and Tamon to $O(\log^2 n/\log^3 \log n)$-term DNF and product distributions [BT96] and by Servedio to $O(2^{\log^{1/2} n})$-term DNF and product distributions [Ser01].

(8) Bshouty gave an algorithm that learns s-term DNF in time $2^{\tilde{O}(\sqrt{n}\log s)}$ [Bsh96]. This was later improved by Klivans and Servedio to $2^{\tilde{O}(n^{1/3}\log s)}$. Alekhnovich *et al.* gave a proper learning algorithm for s-term DNF running in time $2^{\tilde{O}(\sqrt{n}\log s)}$ [ABF⁺04].

An overview of DNF learning results in the related exact learning model of Angluin [Ang87, Ang88] can be found in a survey by Hellerstein [Hel01].

1.2 Summary of Our Contributions

The results presented in this thesis provide answers to several long-standing open questions regarding the limits of learnability by algorithms that use specific representation (e.g. DNF) for their hypotheses. In particular, algorithms that produce hypotheses in the same representation as the most natural representation of the concept class they learn are called *proper*. Many theoretical and applied techniques are based on a specific representation of the hypothesis, such as *decision trees*, *linear thresholds*, or neural networks. Thus, representation-based hardness results elucidate the limitations of such techniques.

A significant part of this thesis is devoted to learning of DNF expressions and parities when the distribution of examples is uniform over $\{0,1\}^n$. We show new algorithms and connections between open problems for learning in this setting. We believe that these results advance the understanding of learnability in this well-studied model.

Some of the results presented in this thesis are for the *agnostic* learning model of Haussler [Hau92] and Kearns *et al.* [KSS94]. This model is an extension of the PAC learning model that deals with the situation in which one cannot assume anything on the target function. Learning in this model is notoriously hard, even though few hardness results are actually known. We give a strong hardness result for proper learning of a basic concept class (monomials) in this model and also the first non-trivial algorithm for agnostic learning another basic class (parities).

Below we summarize the main contributions of this thesis (formal definitions and discussion of the relevant concept classes and learning models can be found in Chapter 2).

Hardness of Proper PAC Learning of DNF Expressions (Chapter 3)

Valiant showed that monotone DNF expressions are learnable properly by a PAC algorithm with access to a membership query oracle and asked whether the same is true for unrestricted DNF expressions [Val84]. We answer his question by proving that, unless $\mathsf{NP} = \mathsf{RP}$, there is no polynomial-time PAC learning algorithm for DNF expressions where the hypothesis is an OR-of-thresholds. As special cases, we show that neither DNF nor OR-of-thresholds are properly learnable unless $\mathsf{NP} = \mathsf{RP}$. Moreover, we show that these results hold even when the learner can make membership queries. While a few hardness results for proper learning are known, all previous results placed significant restrictions on the size of hypotheses and did not allow membership queries.

The proof of this result is based on the inapproximability of the chromatic number of a graph and a delicate reduction from coloring power graphs to learning of DNF formulae by OR-of-thresholds using membership queries.

In addition we prove that it is NP-hard to learn the intersection of $\ell \geq 2$ halfspaces by the intersection of k halfspaces for any constant k. This improves on the fundamental result of Blum and Rivest who proved that intersections of k halfspaces are not learnable by intersections of k halfspaces for every $k \geq 2$ [BR92]. This result is based on a generalization of the reduction of Blum and Rivest, and a strong inapproximability result for coloring 2-colorable hypergraphs of Dinur *et al.* [DRS02].

Hardness of Approximate DNF Minimization and Learning (Chapter 4)

We show that learning of DNF expressions by DNF expressions of a certain, somewhat larger, size is NP-hard even when learning with respect to the uniform distribution and using membership queries. The result is incomparable to that of Chapter 3 since learning often becomes easier when the distribution is uniform but we only rule out a restricted form of proper learning. No hardness results for learning DNF in this model were previously known.

The result is obtained via a new connection to the problem of *two-level logic minimization* or finding a minimal DNF formula consistent with a given complete truth table (TT-MinDNF). This problem was formulated by Quine in 1952 and has been since one of the key problems in logic design. We prove that TT-MinDNF for a function of n variables is NP-hard to approximate within n^γ for some constant $\gamma > 0$, establishing the first inapproximability result for the problem.

The proof of the hardness result for TT-MinDNF is based on a specialized reduction from a multi-prover proof system with certain properties. A powerful low-error PCP by Raz and Safra [RS97] is used to obtain a multi-prover proof system with the desired properties, yielding the desired inapproximability result.

Attribute-Efficient and Non-adaptive Learning of DNF Expressions and Parities (Chapter 5)

Some of the most important considerations in the design of practical learning algorithms are *attribute efficiency* and *noise tolerance*. Attribute-efficient algorithms are algorithms that can learn from a small number of examples even in the presence of numerous irrelevant features in the data. They are of great importance in a common scenario where the data is scarce or expensive to get while each example has a lot of information that is irrelevant to the learned function. Noise-tolerant algorithms are algorithms that are robust to errors in the labels of examples – another common situation in practice.

To contrast the results in Chapter 4, we design the most efficient algorithm for learning DNF expressions with respect to the uniform distribution that uses membership queries. Our algorithm runs in time $\tilde{O}(ns^4/\epsilon)$ and uses $\tilde{O}(s^4 \cdot \log^2 n/\epsilon)$ MQs, where s is the number of terms in the shortest DNF representation of the target concept. The best previously known algorithm for learning DNF runs in time $\tilde{O}(ns^6/\epsilon^2)$ and requires $\tilde{O}(ns^4/\epsilon^2)$ examples [BJT04, KS03]. In addition, our algorithm is attribute-efficient, noise-tolerant, and its membership queries (MQs) are *non-adaptive*, that is, the points at which the algorithm asks MQs do not depend on the target concept. Each of the previously known algorithms had at most one of these properties.

In this chapter we also study attribute-efficient learnability of parity functions with respect to the uniform distribution. We show that attribute-efficient learning of parity functions with respect to the uniform distribution is equivalent to decoding high-rate random linear codes from a low number of errors, a long-standing open problem in coding theory.

Learning Parities with Noise (Chapter 6)

Parity functions are one of most fundamental concept classes in learning theory. In particular, the powerful Fourier transform learning technique is based on representing functions as linear combinations of parities. We address a well-studied problem of learning parities under the uniform distribution with random classification noise, also called the noisy parity problem. We reduce a number of basic problems regarding learning under the uniform distribution to learning of noisy parities, thus establishing the central role of this problem in learning under the uniform distribution.

We show that (under the uniform distribution) agnostic learning of parities reduces to learning parities with random classification noise. Together with the parity learning algorithm of Blum *et al.* [BKW03], this gives the first non-trivial algorithm for agnostic learning of parities. We also show that (under the uniform distribution) PAC learning of DNF expressions reduces to learning noisy parities of just logarithmic number of variables and PAC 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.

Our main reduction can also be stated in terms of coding theory. Specifically, it implies that if there exists an algorithm that can efficiently decode random linear codes from random errors of rate η, then there exists an algorithm that can efficiently (list)decode random linear codes from adversarial errors of rate η.

Hardness of Proper Agnostic Learning of Monomials (Chapter 7)

A monomial or a conjunction of Boolean variables is one of the simplest and most fundamental concepts. The concept class of monomials is long-known to be learnable when examples given to the learning algorithm are guaranteed to be consistent with a monomial (as in the PAC model) [Val84, Lit88]. Real data is rarely completely consistent with a simple concept and therefore this assumption is a limitation of learning algorithms in Valiant's PAC learning model [Val84]. A general way to address this limitation was suggested by Haussler [Hau92] and Kearns *et al.* [KSS94] who introduced the agnostic learning model. In this model, informally, nothing is known about the process that generated the examples and the learning algorithm is required to do nearly as well as is possible using hypotheses from a given class. This corresponds to a common empirical approach when few or no assumptions are made on the data and a fixed space of hypotheses is searched to find the "best" approximation of the unknown function.

We show that for any constant ϵ finding a monomial that agrees with an unknown function on $1/2 + \epsilon$ fraction of examples is NP-hard even when there exists a monomial that agrees with the unknown function on $1 - \epsilon$ fraction of examples. This implies that even weak agnostic learning of monomials by an algorithm that outputs a monomial is NP-hard, resolving one of the basic open problems in the model [Blu98].

The result is obtained by showing that the maximum agreement rate of a monomial with a given set of examples is NP-hard to approximate within $2-\epsilon$ for any constant $\epsilon > 0$. Factor 2 approximation is trivial and therefore this result is optimal. It substantially improves on previously known results of Ben-David *et al.* [BDEL03], and Bshouty and Burroughs [BB06] (who prove hardness for factors $\frac{770}{767}$ and $\frac{59}{58}$, respectively).

Under the assumption that $\mathsf{NP} \not\subseteq \mathsf{RTIME}(n^{poly\log(n)})$, we also obtain an inapproximability factor of $2^{\log^{1-\lambda} n}$ for the symmetric problem of approximating the minimum disagreement rate. This improves on the $\log n$ hardness of approximation factor due to Kearns *et al.* [KSS94] and Hoffgen *et al.* [HvHS95].

To prove these results we design a new probabilistically checkable proof system that is of independent interest. Previous approaches to this and similar problems used known inapproximability results and in this sense this is the first direct application of powerful PCP techniques to learning theory. This approach was recently used by Feldman *et al.* [FGKP06] and Guruswami and Raghavendra [GR06] to prove equally strong hardness results for agnostic learning of halfspaces.

Chapter 2

Preliminaries

In this chapter we define the learning models and the classes of functions that are discussed in this thesis. For detailed introduction of the models and great examples of learning algorithms we refer the reader to the textbook by Kearns and Vazirani [KV94b].

We assume the reader is familiar with the basic notions of computational complexity theory. For the background and definitions related to such topics as circuits, randomized computation, NP-completeness and interactive proofs we refer the reader to [Pap94].

2.1 Concepts and Their Representations

Let X be a set which is the domain or the *input space* of the unknown function to be learned. The domain of the learning problems that we study is $\{0,1\}^n$, or the n-dimensional *Boolean hypercube*. In this domain a point is described using n Boolean variables (or, *attributes*) denoted $x_1, x_2, \ldots, x_n$. The dimension of the domain n is a parameter of our learning problems used for asymptotic analysis of the efficiency of learning. A *concept* over X is a Boolean function over the domain and a *concept class* $\mathcal{C}$ is a set of concepts over X. The unknown function $c \in \mathcal{C}$ that a learning algorithm is trying to learn is referred to as the *target concept*.

In order to discuss algorithms that learn and output functions we need to define how these functions are represented. A representation for a concept class $\mathcal{C}$ is a way to describe functions from $\mathcal{C}$ that defines a procedure to evaluate a function in $\mathcal{C}$ on any input in the domain. For example, we can represent a concept class of all the conjunctions of input variables by the listing the variables in a conjunction. More formally, we define

Definition 2.1.1 *A* representation class $\mathcal{F}$ *over* X *is a pair* $(L, \mathcal{R})$ *where*

- *L is a language over some fixed finite alphabet (e.g. $\{0,1\}$);*

- *$\mathcal{R}$ is an algorithm that for $\sigma \in L$, on input $(\sigma, 1^n)$ returns a Boolean circuit over $\{0,1\}^n$.*

For the purposes of this work we only consider efficient representations, in other words, representations for which $\mathcal{R}$ is a polynomial-time algorithm. The concept class represented by $\mathcal{F}$ is set of functions over $\{0,1\}^n$ defined by the circuits in $\{\mathcal{R}(\sigma, 1^n) \mid \sigma \in L\}$ and is denoted by $\mathcal{C}_{\mathcal{F}}$. For a Boolean function f we say that $f \in \mathcal{F}$ if $f \in \mathcal{C}_{\mathcal{F}}$ and is given by its representation $\sigma \in L$. For most of the representations we use, it is straightforward to construct a language L and the translating function $\mathcal{R}$, and we do not specify them explicitly.

Associated with each representation is the complexity of describing a Boolean function using this representation. More formally for a Boolean function $c \in \mathcal{C}_{\mathcal{F}}$ we define $\mathcal{F}\texttt{-size}(c)$ to be the length the shortest way to represent c using $\mathcal{F}$, or $\min\{|\sigma| \mid \sigma \in L,\ \mathcal{R}(\sigma, 1^n) = c\}$.

The representation class that we study most frequently is *Disjunctive Normal Form* (DNF) expressions (or formulae). A DNF formula is defined as an OR of ANDs of literals, where a *literal* is a possibly negated input variable. We refer to the ANDs of a DNF formula as its *terms*. For example, $(x_1 \wedge \bar{x}_2 \wedge x_5) \vee (x_3 \wedge x_5)$ is a DNF expression with two terms. The representation class of s-term DNF includes all the DNF expressions with at most s terms. Note that every Boolean function can be represented as a DNF expression (possibly of exponential in n size).

A *linear threshold function*(LTF) or a *halfspace* is a function $f = (\sum_{i=1}^{n} \alpha_i x_i \geq \theta)$, where α_i (for all i) and θ are integers (and it is represented by these integers). An *AND-of-thresholds*, or *intersection of halfspaces* is a function $g = \wedge_{i=1}^{k} h_i$, for some k, where each h_i is a halfspace. Similarly, an *OR-of-thresholds* (or a union of halfspaces) is a function equal to $\vee_{i=1}^{k} h_i$.

In the context of DNF we use the number of terms to measure $\texttt{DNF-size}(f)$ instead of the length of description as in the general definition. Similarly, for unions and intersections of halfspaces we use the number of halfspaces.

Some other simple concept and representation classes that we study are parities, monomials, and juntas. A *monomial* is a function equal to a conjunction of any subset of literals. We denote the representation class of monomials by Mon.

A parity function is a function equal to the *XOR* of some subset of variables, where XOR is the exclusive OR operation denoted by $\oplus$. For a Boolean vector $a \in \{0,1\}^n$ we define the parity function $\chi_a(x)$ as $\chi_a(x) = a \cdot x = \oplus_{i \leq n} a_i x_i$. We denote the representation class of parity functions $\{\chi_a \mid a \in \{0,1\}^n\}$ by PAR and the class of all the parities on at most k literals by PAR(k).

A *k-junta* is a function that depends only on k variables. We represent a parity function or a monomial by listing the indices of the variables on which it depends. For a junta we also fully describe the function of the k variables (that is, we give its truth table).

A *decision tree* is a binary tree with labels chosen from $x_1, \ldots, x_n$ on the internal nodes, and labels from $\{0,1\}$ on the leaves. Each internal node's left branch is viewed as the 0-branch; the right branch is the 1-branch. For a decision tree T, a vector $a \in \{0,1\}^n$ defines a path in the tree from the root to a specific leaf by choosing a_i-branch at each node x_i; the value of T at a is defined to be the label of the leaf.

2.2 PAC Learning

In this section we review the basic PAC learning model and a number of related models.

2.2.1 The Basic Model

The *Probably Approximately Correct* (PAC) model of learning was introduced by Valiant in 1984 and has since become one of the most widely studied learning models [Val84]. In this model the learning algorithm is given access to random examples of the target concept drawn from some distribution $\mathcal{D}$ over X, and is supposed to produce, with high probability, a hypothesis h that approximates c (hence the name of the model). More formally, when learning the target concept c with respect to an unknown distribution $\mathcal{D}$ the learning algorithm has access to an *example oracle* $\mathrm{EX}(c,\mathcal{D})$. On request of the learning algorithm, the oracle $\mathrm{EX}(c,\mathcal{D})$ returns an *example* $\langle x, c(x)\rangle$ where x is chosen randomly with respect to $\mathcal{D}$ independently of any previous examples.

Learning in the PAC model in its general form is described by a number of parameters. The first parameter is the representation class $\mathcal{F}$ that is learned. This representation class expresses the prior belief of the learning algorithm about the nature of the unknown function. This parameter has two aspects. First, it means that the target concept belongs to $\mathcal{C}_\mathcal{F}$ (the concept class represented by $\mathcal{F}$). Note that in the case of learning DNF expressions (or other universal representation), $\mathcal{C}_\mathcal{F}$ is the set of all the Boolean functions and therefore this is not a meaningful restriction. The second aspect is that the representation is used to define the complexity of the target concept. Specifically, it defines the size of the unknown function in representation $\mathcal{F}$ or $\mathcal{F}\texttt{-size}(c)$, usually denoted by s. In the PAC model the learning algorithm is allowed to use more resources (such as running time or examples) when it is learning a more complicated hypothesis in terms of $\mathcal{F}\texttt{-size}$. Therefore, when learning DNF expressions the learning algorithm is allowed to use more resources on functions whose minimal DNF representation has more terms.

The second parameter is the representation class of hypotheses $\mathcal{H}$. The learning algorithm is said to learn *by* $\mathcal{H}$ if it outputs a hypothesis $h \in \mathcal{H}$. The significance of this parameter was identified by Pitt and Valiant who showed that the choice of $\mathcal{H}$ can have dramatic impact on the computational complexity of the learning problem [PV88] (Section 3.1.1 has a more detailed description of their result). Algorithms for which $\mathcal{H} = \mathcal{F}$ are called *proper*.

The desired *accuracy* of learning, denoted by ϵ, is another parameter given to the learning algorithm. One cannot expect a polynomial time algorithm to predict the target concept exactly since, intuitively, a portion of the input space may have a very low weight under $\mathcal{D}$ and therefore the learning algorithm will not see any examples of the target function's behavior on that portion of the space. This implies that the hypothesis will not be correct on some of the points. The accuracy parameter specifies the maximum permissible error of the learning algorithm (relative the same distribution $\mathcal{D}$) and allows the learning algorithm to use more resources to achieve higher accuracy. For $\epsilon \geq 0$, we say that function g ϵ-approximates function f with respect to distribution $\mathcal{D}$ if $\mathbf{Pr}_{\mathcal{D}}[f(x) = g(x)] \geq 1 - \epsilon$. Alternatively, one can say that the error of h on f with respect to $\mathcal{D}$ is at most ϵ.

Finally, the learning algorithm depends on randomly chosen examples and therefore will likely fail altogether if the examples happen to be not representative (such as always being equal to the same point). The *confidence* parameter δ specifies the probability with which the learning algorithm is allowed to fail.

We are now ready to define the model formally.

Definition 2.2.1 *We say that an algorithm $\mathcal{A}$* learns *representation class $\mathcal{F}$ by representation class $\mathcal{H}$ if for every $\epsilon > 0$, $\delta > 0$, n, $c \in \mathcal{C}_{\mathcal{F}}$, and distribution $\mathcal{D}$ over X, $\mathcal{A}(n, \epsilon, \delta)$, given access to $EX(c, \mathcal{D})$, outputs, with probability at least $1 - \delta$, a hypothesis $h \in \mathcal{H}$ that ϵ-approximates c. Furthermore, the learning algorithm is* efficient *if it runs in time polynomial in $n, s = \mathcal{F}\text{-}\mathtt{size}(c), 1/\epsilon$ and $1/\delta$.*

We are primarily interested in efficient algorithms and will often use "learn" to mean "learn efficiently". When we do not specify $\mathcal{H}$ explicitly it is meant to be unrestricted. Such learning is referred to as *representation-independent* (or *PAC prediction* in some other works). Effectively this means that $\mathcal{A}$ can output any Boolean circuit.

Throughout this work we will usually fix $\delta = 1/2$. In order to obtain an algorithm with success probability $1 - \delta$, one can always use a standard confidence boosting procedure (*cf.* [KV94b]). The boosting procedure consists of repeating the original algorithm $k = \log(1/\delta) + 1$ times with slightly increased accuracy (e.g. $\epsilon/2$), each time on new examples and independent coin flips. The hypotheses obtained from these runs are then tested on an independent sample of size $O(\epsilon^{-1} \log(1/\delta))$ and the best one is chosen. This procedure increases the running time and the number of examples used by the algorithm by a factor

$O(\log(1/\delta))$.

2.2.2 Extensions

A number of variants of the PAC learning model have been defined to reflect a variety of considerations. Here we briefly review the ones that are relevant to this work.

In *distribution-specific* PAC learning we assume that the distribution is fixed and known to the learning algorithm. One of the most common distributions that we will consider is the uniform distribution $\mathcal{U}$ over the whole hypercube. We will denote the model of learning with respect to a specific distribution $\mathcal{D}$ by $\text{PAC}_{\mathcal{D}}$. To emphasize the difference between these two settings, the usual PAC learning model is often referred to as *distribution-independent* or *distribution-free*.

Valiant has also introduced an augmented version of his model in which a learning algorithm has access to a *membership query oracle* for the target concept [Val84]. A membership query(MQ) oracle MEM(c) is the oracle that, given any point $x \in \{0,1\}^n$, returns the value $c(x)$. Examples of real-world learning tasks that are modeled by MQs include learning a certain skill from a human expert and learning the influence of genes on a certain condition by conducting biological tests on synthesized DNA sequences.

Achieving any desired accuracy is a strong requirement on a learning algorithm, sometimes referred to as *strong* learning. A relaxation of this requirement is the notion of *weak learning* introduced by Kearns and Valiant [KV94a]. A *weak* learning algorithm is a learning algorithm that can produce a hypothesis whose error on the target concept is noticeably less than 1/2 (either 0 or 1 constant hypothesis predicts the target on 1/2 of the points). More precisely, a weak learning algorithm produces a hypothesis h such that $\mathbf{Pr}_{\mathcal{D}}[c(x) \neq h(x)] \leq 1/2 - 1/p(n,s)$ for some fixed polynomial p (the rest of Definition 2.2.1 is unchanged).

In a celebrated result Schapire showed that weak PAC learning in a distribution-independent setting implies strong PAC learning [Sch90]. His proof is based on using the weak learning algorithm on different distributions to construct a strong learning algorithm. This general technique, referred to as *boosting*, has become a subject of intensive theoretical and practical research. We use a boosting algorithm of Freund in Chapter 5 [Fre92].

In numerous real-life situations examples are an "expensive" resource. It is therefore important to consider the number of examples used by an algorithm to learn, referred to as its *sample complexity*. An algorithm $\mathcal{A}$ is said to be *attribute-efficient* if the number of examples (both random and received from a MQ oracle) it uses is polynomial in the size of the representation of the target concept and $1/\epsilon$. Attribute-efficient learning is important when the number of variables on which the target concept depends is small in comparison with the total number of variables n. We say that a variable x_i is *relevant* for a function f if there exists $y \in \{0,1\}^n$ such that $f(y) \neq f(y^i)$, where y^i is y with the bit in i-th position flipped. The number of relevant variables of the target concept is denoted by r. Attribute-efficiency does not allow the number of examples to depend polynomially on n. Instead the number of examples used can depend polynomially on r and $\log n$ since for most representations (including the ones considered in this work) the size of the representation of f is lower bounded by both $\log n$ and r.

Note that these extensions are generally independent of each other and various combinations are often considered in the learning literature. For example, in Section 5.5 we will present an attribute-efficient weak $\mathrm{PAC}_{\mathcal{U}}$ learning algorithm that uses membership queries.

2.2.3 Noise Models

Learning in the basic PAC model and its extensions defined in the previous section relies on the assumption that the examples that are given to the learning algorithm are always labeled by the correct value of the target concept $c(x)$. Several ways of dealing with the situation when the labels are noisy have been suggested and widely-studied. The first one that we consider is the simplest type of noise called *random classification noise*. It was introduced by Angluin and Laird [AL88]. In this model for any $\eta \leq 1/2$, called the *noise rate*, the regular example oracle $\mathrm{EX}(c, \mathcal{D})$ is replaced with the noisy oracle $\mathrm{EX}^\eta(c, \mathcal{D})$. On each call, $\mathrm{EX}^\eta(c, \mathcal{D})$, draws x according to $\mathcal{D}$, and returns $\langle x, c(x)\rangle$ with probability η and $\langle x, \neg c(x)\rangle$ with probability $1-\eta$. When η approaches $1/2$, the label of the corrupted example approaches the result of a random coin flip, and therefore the running time of algorithms in this model is allowed to polynomially depend on $\frac{1}{1-2\eta}$.

This model of noise is not suitable for corrupting labels returned by $\mathrm{MEM}(c)$ since a learning algorithm can, with high probability, find the correct label at point x by asking for the label of x polynomial (in $\frac{1}{1-2\eta}$) number of times and then returning the label which appeared in the majority of answers (the Chernoff bound given in Lemma 2.3.1 implies that the result will, with high probability, be equal to the correct label). An appropriate modification of the noise model is the introduction of *persistent classification noise* by Goldman, Kearns and Shapire [GKS93]. In this model, as before, the answer to a query at each point x is flipped with probability $1-\eta$. However, if the membership oracle was already queried about the value of f at some specific point x, or if x was already generated as a random example, the returned label has the same value as in the first occurrence (i.e., in such a case the noise persists and is not purely random). If the learner does not ask for the label of a point more than once then this noise can be treated as the usual independent random classification noise. There is another subtle but important feature of this model. We cannot require learning algorithms in this model to work for an arbitrarily small confidence parameter δ because with some (though negligible) probability the target function can be persistently totally corrupted (for example negated), making learning absolutely infeasible. So we must impose a positive lower bound on the confidence δ that can be required from the learning algorithm in the persistent noise model, although this bound will be negligible.

2.2.4 Agnostic Learning

Another way of relaxing the assumption that examples are labeled by a concept from a specific concept class is the introduction of the *agnostic* PAC learning model by Haussler [Hau92] and Kearns *et al.* [KSS94]. In this model no assumptions are made on the function that labels the examples, in other words, the learning algorithm has no prior beliefs about the target concept (and hence the name of the model). The goal of the agnostic learning algorithm for a concept class $\mathcal{C}$ is to produce a hypothesis h whose error on the target concept is close to the best possible by a concept from $\mathcal{C}$. This model reflects a common empirical approach to learning, where few or no assumptions are made on the unknown function and a limited space of candidate hypothesis functions is searched in an attempt to find the best approximation to the target function.

Formally, for two Boolean functions f and h and a distribution $\mathcal{D}$ over the domain, we define $\Delta_{\mathcal{D}}(f,h) = \mathbf{Pr}_{\mathcal{D}}[f \neq h]$. Similarly, for a concept class $\mathcal{C}$ and a function f define $\Delta_{\mathcal{D}}(f,\mathcal{C}) = \min_{h\in\mathcal{C}}\{\Delta_{\mathcal{D}}(f,h)\}$.

Definition 2.2.2 *We say that an algorithm $\mathcal{A}$* agnostically *learns representation class $\mathcal{F}$ by representation class $\mathcal{H}$ if for every $\epsilon > 0$, $\delta > 0$, n, Boolean function f and distribution $\mathcal{D}$ over X, $\mathcal{A}(n,\epsilon,\delta)$, given access to $EX(f,\mathcal{D})$, outputs, with probability at least $1-\delta$, a hypothesis $h \in \mathcal{H}$ such that $\Delta_{\mathcal{D}}(f,h) \leq \Delta_{\mathcal{D}}(f,\mathcal{C}_{\mathcal{F}}) + \epsilon$. As before, the learning algorithm is* efficient *if it runs in time polynomial in $n, s = \mathcal{F}\text{-}\mathtt{size}(f), 1/\epsilon$ and $1/\delta$.*

It is easy to see that if we restrict the target function to be in $\mathcal{C}_{\mathcal{F}}$ then $\Delta_{\mathcal{D}}(f,\mathcal{C}_{\mathcal{F}}) = 0$ and we obtain the usual PAC learning model. In this sense the agnostic PAC model generalizes the basic model. Furthermore, all the extensions of the PAC model discussed in Section 2.2.2 can also be applied to the agnostic setting.

The agnostic learning model can also be thought of as a model of adversarial noise. By definition, a Boolean function f differs from some function in $c \in \mathcal{C}_{\mathcal{F}}$ on $\Delta_{\mathcal{D}}(f,\mathcal{C}_{\mathcal{F}})$ fraction of the domain. Therefore f can be thought of as c corrupted by *adversarial classification noise* of rate $\Delta_{\mathcal{D}}(f,\mathcal{C}_{\mathcal{F}})$. Note that an agnostic learning algorithm will not necessarily find a hypothesis that approximates c – any other function that differs from f on at most $\Delta_{\mathcal{D}}(f,\mathcal{C}_{\mathcal{F}}) + \epsilon$ fraction of the domain is acceptable.

A minor difference arises from the fact that the noise rate is not explicitly mentioned in the agnostic setting and, in particular, agnostic algorithms are not allowed polynomial time dependence on $\frac{1}{1-2\eta}$. But we note that if $\epsilon > 1/2-\eta$ then a constant function (either 0 or 1) will be a good agnostic hypothesis. Therefore we can assume that $\epsilon \leq 1/2-\eta$ and any algorithm that is polynomial in $\frac{1}{1-2\eta}$ will also be polynomial in $1/\epsilon$.

2.3 Estimating Random Variables

As can be seen from the definitions above, the PAC model is intrinsically probabilistic. Therefore, the analysis of learning algorithms often involves evaluation of the accuracy and confidence of estimates produced by random sampling. We do this by using the standard inequalities of Chernoff and Bienaymé-Chebyshev.

Lemma 2.3.1 (Chernoff [Che52](see also [MR95])) *Let $X_1, \ldots, X_m$ be a sequence of m independent Bernoulli trials, each with probability of success $\mathbf{E}[X_i] = p$ and let $S = \sum_{i=1}^m X_i$. Then for $0 \leq \gamma \leq 1$,*

$$\mathbf{Pr}[S > (1+\gamma)pm] \leq e^{-mp\gamma^2/3}$$

and

$$\mathbf{Pr}[S < (1-\gamma)pm] \leq e^{-mp\gamma^2/2} \ .$$

Lemma 2.3.2 (Bienaymé-Chebyshev (c.f. [MR95])) *Let $X_1, \ldots, X_m$ be pairwise independent random variables all with mean μ and variance σ^2. Then for any $\lambda \geq 0$,*

$$\mathbf{Pr}\left[\left|\frac{1}{m}\sum_{i=1}^m X_i - \mu\right| \geq \lambda\right] \leq \frac{\sigma^2}{m\lambda^2} \ .$$

Chapter 3

Hardness of Proper PAC Learning of DNF Expressions

In this chapter we prove that unless $\mathsf{NP} = \mathsf{RP}$, there is no polynomial-time PAC learning algorithm for DNF formulae where the hypothesis is an OR-of-thresholds. As special cases, we show that neither DNF nor OR-of-thresholds are properly learnable unless $\mathsf{NP} = \mathsf{RP}$. Moreover, we show that these results hold even when the learner can make membership queries. We also prove that it is NP-hard to learn the intersection of $\ell \geq 2$ halfspaces by the intersection of k halfspaces for any constant $k \geq 0$.

3.1 Introduction

A fundamental goal of computational learning theory is to establish hardness results for PAC learning concept classes. Seminal work by Kearns and Valiant [KV94a] has shown that, under the assumption that certain cryptographic primitives are computationally intractable (e.g. inverting one-way functions), there are no polynomial-time learning algorithms for concept classes which are expressive enough to compute pseudorandom-functions. Subsequent work [Kha93, NR97, JKS02] has shown that even constant depth, polynomial size circuits (often referred to as AC^0) are capable of computing pseudo-random objects and are unlikely to be learnable in polynomial time.

Still, several well-studied concept classes seem too weak to compute cryptographic primitives, such as polynomial-size DNF formulae, intersections of halfspaces, and decision trees. For all of these concept classes the existence of a polynomial-time PAC learning algorithm remains a challenging open problem. The primary contribution of this chapter is new negative results for learning DNF formulae and intersections of halfspaces. Our hardness results apply to *representation dependent* learning algorithms, algorithms where the output hypothesis is represented in a certain specific way.

3.1.1 Previous Work

Previous representation dependent hardness results for learning concept classes applied to *proper* learning algorithms and required strong restrictions on the size of the hypothesis output by the learning algorithm [BR92, Gol78, KLPV87a, PV88, NJS98]. In each case, the hardness assumption required is not cryptographic, but a worst-case assumption on the complexity of NP (e.g. $\mathsf{NP} \neq \mathsf{RP}$).

Initial hardness results for properly learning DNF formulae due to Pitt and Valiant [PV88] show that coloring a k-colorable graph on n vertices using ℓ colors can be reduced to learning k-term DNF formulae over n variables by ℓ-term DNF formulae. In particular, combined with the hardness results on chromatic number due to Feige and Kilian [FK96], their result implies that unless $\mathsf{NP} = \mathsf{RP}$ it is hard to learn n^{γ}-term DNF by $n^{1-\gamma}$-term DNF. Pitt and Valiant also show a similar reduction from k-NM-Colorability [GJ79] (also called hypergraph coloring) to learning of DNF formulae by DNF formulae. They used this reduction to show that 2-term DNF formulae are not learnable properly (unless $\mathsf{NP} = \mathsf{RP}$). Combined with recent results on the hardness of approximate hypergraph coloring [DRS02], their reduction implies that 2-term DNFs are NP-hard to learn by k-term DNFs for any constant k (we include this reduction for completeness in Section 3.3.1).

The best result along these lines is due to Nock *et al.* [NJS98] who have used reductions from generalized coloring problems to show that it is hard to output a DNF formula whose size is at most $O(k^a n^b)$ times the size of the unknown k-term DNF formula for $a \leq 2, b \geq 0$ and $k = \Omega(n^{\gamma})$ for any $\gamma > 0$.

The best hardness result for learning intersections of halfspaces is due to Blum and Rivest [BR92]. They prove that unless $\mathsf{NP} = \mathsf{RP}$, it is hard to learn intersections of k halfspaces by intersections of k halfspaces for any $k \geq 2$.

We note here that the above hardness results hold for proper *Occam* algorithms, learning algorithms which work by receiving a suitably large set of training examples and outputting a small hypothesis consistent with the examples. It is not known, in general, if the existence of a proper PAC learning algorithm for a concept class implies the existence of a proper Occam algorithm for the class [PB90]. In particular, it is not known for the classes of DNF formulae and intersections of halfspaces. Our hardness results for DNF formulae and intersections of halfspaces hold for any proper PAC learning algorithm and overcome this limitation.

Angluin and Kharitonov prove that if non-uniform one-way functions exist then MQs do not help predicting DNF formulae [AK95b]. However, their reduction does not preserve the representation of a hypothesis and therefore cannot be used to obtain hardness of proper learning with MQs.

Hardness results for learning of DNF expressions with MQs are only known for the *exact* model of learning (which is weaker than PAC learning) and only for strong proper learning (or slight relaxations similar to the one we prove for PAC learning with respect to the uniform distribution). The strongest results in this model are due to Hellerstein and Raghavan [HR02] and are based on information-theoretic hardness.

Our hardness results for learning DNF expressions are contrasted by the fact that monotone DNF expressions are known to be strongly properly PAC learnable with MQs [Val84]. In addition to that, DNF expressions with k terms are known to be learnable by DNF expressions with 2^k terms when MQs are available [BR95]. It is also interesting to note that known non-trivial algorithms for learning unrestricted DNF formulae (running in time $2^{\tilde{O}(n^{\frac{1}{3}})}$ [KS04a] and in time $2^{\tilde{O}(\sqrt{n})}$ with DNF hypotheses [ABF+04]) use only random examples and it is unknown whether they could be sped-up by using MQs.

3.1.2 Our Results

Learning DNF formulae is one of the central challenges in computational learning theory. We give strong evidence that there are no polynomial-time learning algorithms for DNF formulae which output DNF formulae or unions of halfspaces as output hypotheses:

Theorem 3.1.1 *If there exists an algorithm $\mathcal{A}$ such that for every Boolean function c, distribution $\mathcal{D}$ and error parameter ϵ, $\mathcal{A}$, given access to the example and the membership query oracles for c, runs in time $poly(n, \texttt{DNF-size}(c), 1/\epsilon)$ and with probability $3/4$ outputs an OR-of-thresholds formula h such that $\mathbf{Pr}_{x \in \mathcal{D}}[h(x) = c(x)] \geq 1 - \epsilon$, then $\mathsf{NP} = \mathsf{RP}$.*

Access to membership queries plays an instrumental role in numerous learning algorithms (many of which are proper), but hardness results for learning with MQs are still very scarce. This is the first result showing that PAC learning can be NP-hard even when MQs are available.

Hardness Results for Learning Intersections of Halfspaces

Let $h = \mathsf{sign}(\sum_{i=1}^{n} w_i x_i - \theta)$ where each w_i and θ are integers; h naturally induces two halfspaces: the set of points in $\{0,1\}^n$ which make h positive and the set of points which make h negative (h is often referred to as a linear threshold function). Although several polynomial-time algorithms for learning halfspaces are known (e.g. [BEHW87]), a long-standing open problem in learning theory is to develop polynomial-time algorithms for learning *intersections* of halfspaces (i.e. functions of the form $h = \wedge_{i=1}^{k} h_i$ where each h_i is a linear threshold function).

The above theorem proves as a special case that intersections of halfspaces are not properly learnable unless $\mathsf{NP} = \mathsf{RP}$. If we wish to restrict the concept class to intersections of just two halfspaces (even for this case no polynomial-time learning algorithms are known), we can prove the following hardness result:

Theorem 3.1.2 *Let $\mathcal{C}$ be the concept class of intersections of two halfspaces. If there exists an algorithm $\mathcal{A}$ such that for every $c \in \mathcal{C}$, distribution $\mathcal{D}$ and error parameter ϵ, $\mathcal{A}$ runs in time poly$(n, 1/\epsilon)$ and with probability $3/4$ outputs h, an intersection of k halfspaces for any constant $\ell \geq 0$ such that $\mathbf{Pr}_{x \in \mathcal{D}}[h(x) = c(x)] \geq 1 - \epsilon$, then* $\mathsf{NP} = \mathsf{RP}$.

The result of Blum and Rivest [BR92] only implies hardness of learning the intersection of two halfspaces by the intersection of two halfspaces.

3.1.3 Our Approach

Our results are based on the intractability of approximate graph and hypergraph coloring.

Amplifying Hardness Results for Approximate Graph Coloring

For proving hardness results for properly learning DNF and intersections of halfspaces we amplify known hardness results for the problem of distinguishing between graphs with small and large chromatic number. Feige and Kilian [FK96] have proved that for any $\gamma > 0$ it is NP-hard (under randomized reductions) to distinguish between graphs with chromatic number $O(n^{\gamma})$ and graphs with chromatic number $\Omega(n^{1-\gamma})$. This result combined with known reductions from graph coloring to properly learning DNF formulae (i.e. [PV88]) implies that it is NP-hard to distinguish between distributions induced by n^{γ}-term DNF formulae and $n^{1-\gamma}$-term DNF formula.

We wish to amplify this $n^{1-\gamma}$ bound and prove hardness results for n^{a}-term DNF formulae (and intersections of n^{a} halfspaces) for any $a \geq 0$. To do this we apply specialized graph products (along the lines of Linial and Vazirani [LV89]) to create distributions which amplify the size of the underlying chromatic number. In addition, we provide an accompanying transformation of DNF formulae and intersections of halfspaces into "normal forms" which satisfy only examples derived from subsets of independent sets from the product. Many terms or halfspaces are required for a good approximation to these distributions if and only if the original graph had large chromatic number.

For proving hardness results for learning the intersection of two halfspaces, we make critical use of recent hardness results due to Dinur *et al.* [DRS02] on the hardness of coloring 2-colorable, 3-uniform hypergraphs. We give a reduction from ℓ-coloring k-colorable, 3-uniform hypergraphs to properly learning intersections of k halfspaces by ℓ halfspaces.

3.2 Preliminaries

Our learning model is Valiant's well known Probably Approximately Correct (PAC) learning model [Val84] defined in Section 2.2. Recall that a DNF formula is a logical formula equal to the OR of a number of ANDs. A halfspace is a function $f = \mathsf{sign}(\sum_{i=1}^{n} \alpha_i x_i - \theta)$ where α_i (for all i) and θ are integers. An intersection of k halfspaces is a function $g = \wedge_{i=1}^{k} h_i$ where each h_i is a halfspace. A neural network with k hidden nodes is a function $g = f(h_1(\vec{x}), \ldots, h_k(\vec{x}))$ where each h_i is a halfspace and f is an arbitrary Boolean function. Each h_i is called a *hidden* node. The halfspace h_i is *origin-centered* if the corresponding $\theta = 0$.

3.3 Hardness of Learning DNF and Intersections of Halfspaces

In this section we prove our main hardness result for DNF formulae, namely that an algorithm for learning DNF in polynomial-time by ORs of threshold functions can be used to approximate the chromatic number of a graph. We will actually prove the equivalent hardness result for CNF formulae and ANDs of thresholds (intersections of halfspaces). It is easy to see that this will imply the intractability of properly learning both DNF formulae and intersections of halfspaces. We begin by reviewing the reduction of Pitt and Valiant that gives a simple way to reduce a coloring problem to proper learning of DNF [PV88].

3.3.1 The Reduction of Pitt and Valiant

Pitt and Valiant [PV88] gave a reduction from coloring k-colorable hypergraphs to learning k-term DNF formulae (in their work this problem is referred to as k-NM-Colorability). We include this reduction for completeness and also observe that, combined with a recent hardness result for coloring hypergraphs due to Dinur *et al.* [DRS02], the reduction implies that 2-term DNF in not learnable by k-term DNF for any constant k.

Recall that k-coloring a hypergraph means finding a mapping from the vertices to $\{1, \ldots, k\}$ such that no edge has all of its vertices assigned the same integer.

Theorem 3.3.1 ([PV88]) *Coloring a k-colorable hypergraph $H = (V, E)$ using ℓ colors reduces to learning k-term DNF formulae by outputting an ℓ-term DNF formulae.*

Proof: Let $\mathcal{A}$ be an algorithm for learning k-term DNF formulae by ℓ-term DNF formulae and let $H = (V, E)$ be any k-colorable hypergraph on n vertices. For a vertex $v_i \in V$ let $a(v_i)$ be the a vector of length n which is equal to 0 in position i and 1 elsewhere. For an edge $e \in E$ let $a(e) = \bigwedge_{v \in e} a(v)$ (conjunction is applied bitwise).

We construct a set of examples S as follows:

- Vertex examples: for each $v \in V$, $(a(v), +)$.
- Edge examples: for each $e \in E$, $(a(e), -)$.

We now claim that any k-coloring of H can be efficiently translated into a k-term DNF formula consistent with the given examples and vice versa. Let χ be a k-coloring of H. For every color $c \leq k$ we define

$$t_c = \bigwedge_{\chi(v_i) \neq c} x_i \ ,$$

that is, t_c is the conjunction of all the variables whose corresponding vertices are not colored in color c. We set $h = t_1 \vee t_2 \vee \ldots \vee t_k$. Clearly h is a k-term DNF formula and the translation is efficient. For every vertex example $a(v_i)$, $t_{\chi(v_i)}(a(v_i)) = 1$ and hence $h(a(v_i)) = 1$. For any edge example $a(e)$, vertices in e are colored in at least two different colors and hence every term t_c will contain at least one variable x_i such that $v_i \in e$. This means that h will not satisfy $a(e)$.

Now let $h = t_1 \vee t_2 \vee \ldots \vee t_\ell$ be a DNF expression consistent with the given examples. For every vertex v, we define $\chi(v) = c$ if $a(v)$ is satisfied by t_c (if there are several terms that satisfy $a(v)$ we choose the one with the smallest c). Clearly this defines a mapping of vertices into ℓ colors. Take $e \in E$ and assume that all the vertices in it are colored in color c. That is, for each $v \in e$, $t_c(a(v)) = 1$. This implies that

$$t_c(a(e)) = t_c(\bigwedge_{v \in e} a(v)) = \bigwedge_{v \in e} t_c(a(v)) = 1$$

contradicting the consistency with example $(a(e), -)$. $\square$

This reduction can be used with the following hardness result due to Dinur *et al.* [DRS02]:

Theorem 3.3.2 ([DRS02]) *It is* NP*-hard to k-color a* 2*-colorable* 3*-uniform hypergraph for any constant $k \geq 0$.*

We therefore obtain the following hardness result.

Theorem 3.3.3 *Assuming* $\mathsf{NP} \neq \mathsf{RP}$ *there is no polynomial-time algorithm for learning* 2*-term DNF formulae by k-term DNF formulae for any constant k.*

To contrast this with known results for learning k-term DNF, note that a k-term DNF is learnable in time $O(n^k)$ where the hypothesis is a CNF of size $O(n^k)$ [Val85].

3.3.2 The Target Function and Distribution

We now present our reduction from coloring a graph to learning of CNF. Given a graph $G = (V, E)$, construct a target function f and a distribution D as follows.

Fix some positive integer parameter r. The examples are from $\left(\{0,1\}^V\right)^r = \{0,1\}^{|V|\cdot r}$.

Definition 3.3.4 *Let $G(V,E)$ be a graph with n vertices and m edges. For a vertex $v \in V$, let $z(v)$ denote the vector with a 1 in the v-th position and 0 everywhere else. For an edge $e = (u,v)$ of G, let $z(e)$ be the vector with a 1 in positions u and v.*

With each vector $(v_1, v_2, \ldots, v_r) \in V^r$, we associate a negative example $\langle z(v_1), \ldots, z(v_r), 0\rangle$. For each choice of k_1, k_2, such that $1 \le k_1 \le r$, $1 \le k_2 \le r$, $k_1 \ne k_2$, $e = (u,w) \in E$ and $v_i \in V$ for each $i = 1, 2, \ldots, r$, $i \ne k_1, k_2$, we associate a positive example

$$\langle z(v_1), \ldots, z(v_{k_1-1}), z(e), z(v_{k_1+1}), \ldots, z(v_{k_2-1}), \overline{0}, z(v_{k_2+1}), \ldots, z(v_r), 1\rangle \ .$$

Let S^+ denote the positive examples and S^- denote the negative examples. Set $S = S^+ \cup S^-$.

There are r ways to choose k_1, $r-1$ ways to choose k_2, $|E|$ ways to choose e, and $|V|^{r-2}$ ways to choose the rest of v_i's. Hence there is a total of $r\cdot(r-1)\cdot|E|\cdot n^{r-2}$ positive examples.

Distribution D is uniform over the above set of examples S, so the probability of each negative example is $\frac{1}{2\cdot n^r}$ and the probability of each positive example is $\frac{1}{2\cdot r\cdot(r-1)\cdot|E|\cdot n^{r-2}}$.

These examples define the values of f on points in S. In order to answer membership query we also need to define f on the rest of the hypercube. Let $x = (x^1, \ldots, x^r)$ be a point not in $S^+ \cup S^-$. If for all i, $x^i \in \{\bar{0}\} \cup \{z(v) \mid v \in V\}$ then $f(x) = 0$. We refer to this set of points as *0-vertex points*. If for some $i \in [r]$, there exists $(u,v) \notin E$ such that $x^i_u = x^i_v = 1$, then $f(x) = 0$. We call this set of points *non-edge* points. Otherwise, let $f(x) = 1$. We first note that the example oracle for f with respect to the distribution D and the membership query oracle for f can be simulated efficiently by a randomized algorithm with input G.

3.3.3 The Case of Small Chromatic Number

We now prove that if the chromatic number $\chi(G)$ is small, then there exists a small CNF formula equal to f. Set $r = g(n)/\gamma = g/\gamma$, for some function g such that $g(n) \le n$ and constant $\gamma < 1$. Hence $\gamma = g/r$. Then we have

Lemma 3.3.5 *If $\chi(G) \leq n^{\gamma} = n^{g/r}$, then there is a CNF formula of size at most $n^g + r|E|$ equal to f.*

Proof: Suppose $V = \bigcup_{i=1}^{\chi} V_i$, where V_i are independent sets. Such sets must exist by the definition of χ. Define the CNF formula

$$g(x_1, x_2, \ldots, x_n) = \bigwedge_{i=1}^{\chi} \bigvee_{v \notin V_i} x_v \ .$$

This formula rejects all points in $\{\bar{0}\} \cup \{z(v) \mid v \in V\}$ and accepts all points in $\{z(e) \mid e \in E\}$.

We then define an expression F that rejects all the points in S^- and 0-vertex points

$$F(x^1, \ldots, x^r) = \bigvee_{k=1}^{r} g(x_1^k, \ldots, x_n^k) = \bigvee_{k=1}^{r} \bigwedge_{i=1}^{\chi} \bigvee_{v \notin V_i} x_v^k \ ,$$

and a CNF formula H on $r \cdot n$ variables that is negative on all the non-edge points and positive elsewhere

$$H(x^1, \ldots, x^r) = \bigwedge_{k \in [r];\ (u,v) \notin E} (\overline{x_u^k} \vee \overline{x_v^k}) \ .$$

We claim that in addition to S^- and 0-vertex points F rejects only non-edge points. By the definition of F, if F rejects a point $x^1, \ldots, x^r$ then for all $k \in [r]$, there exists $i \in [\chi]$ such that for all $v \notin V_i$, $x_v = 0$. Therefore if for some $k \in [r]$ and $u, v \in V$, $x_u^k = x_v^k = 1$ then $u, v \in V_i$ for some i. In particular, $(u, v) \notin E$ since V_i is an independent set. We therefore obtain that $F \wedge H$ rejects exactly points in S^-, 0-vertex points and the non-edge points, in other words, is identical to f. We remark that in order to answer membership queries to F we would need to know which non-edge points it accepts and which rejects. This would not be possible without knowing the "small" coloring. Therefore f is defined as $F \wedge H$ in order to hide all the coloring-dependent information in F by using H that rejects all the non-edge points.

Note that F above is not written as a CNF formula. It is, however, a disjunction of r CNF formulas, each having at most $\chi(G)$ clauses. Hence expanding the expression for F yields a CNF formula with at most $\chi(G)^r \leq n^g$ clauses. So $F \wedge H$ can be written as a CNF formula satisfying the conditions of the lemma. □

3.3.4 The Case of Large Chromatic Number

In this section we assume that $\chi(G) \geq n^{1-\lambda}$, and we prove that no small AND-of-thresholds formula gives a good approximation to the learning problem.

Theorem 3.3.6 *Let G be a graph such that $\chi(G) \geq n^{1-\lambda}$. Let $F = \wedge_{i=1}^{\ell} h_i$ where $\ell < \frac{1}{2\chi r}\left(\frac{\chi-1}{\log n}\right)^r$. Then F has error at least $\frac{1}{n^{2g+4}}$ with respect to D.*

We will need the following covering lemma which was first proved by Linial and Vazirani [LV89] and is a special case of a result due to Feige on randomized graph products (Corollary 2.9 of [Fei95]):

Lemma 3.3.7 *[LV89] One needs at least $\left(\frac{\chi-1}{\ln n}\right)^r$ products of the form $I_1 \times I_2 \times \ldots \times I_r$, where the I_i's are independent sets, to cover $V^r = V \times V \times \ldots \times V$.*

Let a product in the above form be called a product of independent sets. At a high level, we will argue that any $h_k \in F$ correctly classifies very few negative examples that lie outside a particular product of independent sets. Then using the above lemma, it will follow that we need many h_k's to cover (correctly classify) all negative examples. We now proceed to the details.

Fix a particular $h_k \in F$. Let $h_k = \sum_{i=1}^{r}\sum_{j=1}^{n} \alpha_j^i x_j^i \geq \beta$. For each $i \leq r$, the i-coefficients in h are the coefficients of the form α_j^i, $j \leq n$. For each $i \leq r$, let I_i be the set of all $j \leq n$ such that there is no edge $(k, j) \in E$ such that α_k^i is less than α_j^i. (That is, we order all i-coefficients in nondecreasing order, and take the coefficients in order that are independent). Note that I_i is an independent set of G. Let $S_1^k = V \times I_2 \times \ldots \times I_r$, $S_2^k = I_1 \times V \times I_3 \times \ldots \times I_r$, and so forth. Let $S^k = \cup_{i=1}^{r} S_i^k$. The following lemma shows that h_k either misclassifies many positive examples, or misclassifies almost all negative examples outside of S^k.

Lemma 3.3.8 *Let $\{h_k\}_{k=1}^{\ell}$ be a family of halfspaces, and S^k as above. Let N denote the number of negative examples outside of $\cup_{k=1}^{\ell} S^k$ that $\wedge h_k$ classifies correctly. Then the number of positive examples that $\wedge h_k$ misclassifies is at least $N/2n$.*

Proof: Fix h_k and $I_1, I_2, \ldots, I_r$ as above. Let $\alpha = z(j_1), \ldots, z(j_r)$ be a negative example such that α is not in S^k, and $h_k(\alpha) = 0$. Thus $h_k(\alpha) = \alpha_{j_1}^1 + \alpha_{j_2}^2 + \ldots + \alpha_{j_r}^r < \beta$. Since α is not in S^k, there exist two j_i's, say j_1 and j_2 such that $j_1 \notin I_1$ and $j_2 \notin I_2$. Since j_1 is not in I_1, there is some vertex k_1 in I_1 such that the edge (j_1, k_1) is present in E_1 and similarly there is a vertex k_2 in I_2 such that the edge (j_2, k_2) is in E_2. By the way we chose I_1 and I_2, it follows that $\alpha_{k_1}^1 \leq \alpha_{j_1}^1$ and $\alpha_{k_2}^2 \leq \alpha_{j_2}^2$. Either (a) $\alpha_{j_1}^1 \leq \alpha_{j_2}^2$, or (b) $\alpha_{j_2}^2 < \alpha_{j_1}^1$. If (a) holds, then $\alpha_{k_1}^1 + \alpha_{j_1}^1 + \alpha_{j_3}^3 + \ldots + \alpha_{j_r}^r < \beta$. But this corresponds to the positive example $\alpha' = (z(j_1, k_1), \overline{0}, z(j_3), \ldots, z(j_r))$ and thus h_k (and $\wedge h_k$) misclassifies α'. Similarly if (b) holds, then h_k (and $\wedge h_k$) misclassifies the positive example $\alpha' = (\overline{0}, z(j_2, k_2), z(j_3), \ldots, z(j_r))$. Thus we have a mapping from the set of all correctly classified negative examples outside of $\cup_{k=1}^{\ell} S^k$ to incorrectly classified positive examples. Since each positive example is mapped onto by at most $2n$ negative examples (each misclassified positive example can be obtained from starting with a negative example that falls into either cae (a) or case(b)), it follows that the number of positive examples misclassified by $\wedge h_k$ is at least $N/2n$. □

Recall that F is the conjunction of ℓ threshold formulas, $h_1, \ldots, h_\ell$. For each h_k, let S^k be the associated set of cross products. Let the negative examples that h_k correctly classifies be denoted by $In_k \cup Out_k$, where In_k are those correctly classified negative examples in S^k, and Out_k are the remaining correctly classified negative examples.

Lemma 3.3.9 *Let S^k, $k \leq \ell$ be defined as above. If $\ell \leq \frac{1}{2\chi r} \cdot \left(\frac{\chi-1}{\ln n}\right)^r$ then*

$$n^r - |\cup_{k=1}^{\ell} S^k| \geq \frac{1}{2} \cdot \left(\frac{\chi - 1}{\ln n}\right)^r .$$

Proof: If this were not the case, we would have a collection of $\ell \cdot \chi \cdot r \leq \frac{1}{2} \cdot \left(\frac{\chi-1}{\ln n}\right)^r$ products of independent sets which cover all but $m < \frac{1}{2} \cdot \left(\frac{\chi-1}{\ln n}\right)^r$ points of V^r. (To see this, replace the cross product $I_1 \times \ldots I_{i-1} \times V \times \ldots \times I_{i+1} \times \ldots \times I_r$ by χ cross products $I_1 \times \ldots I_{i-1} \times J_k \times I_{i+1} \times \ldots \times \ldots \times I_r$, where $k \leq \chi$, and $J_1, J_2, \ldots J_\chi$ is a partition of the vertices in G into χ independent sets.) Then by adding m singletons (which are trivially products of independent sets) we obtain a cover of V^r by $\ell\chi r + m < \left(\frac{\chi-1}{\ln n}\right)^r$ products of independent sets, which contradicts the above covering lemma (Lemma 3.3.7). □

We can now analyze the overall error with respect to D. Let $F = \wedge_{k=1}^{\ell} h_k$, where each h_k is a threshold formula, and $\ell < \frac{1}{2\chi r}\left(\frac{\chi-1}{\ln n}\right)^r$

Let $R = \frac{1}{4} \cdot \left(\frac{\chi-1}{\ln n}\right)^r$. There are two cases to consider. The first case is when $|\cup_{k=1}^{\ell} Out_k| \geq R$. Then by Lemma 3.3.8, the number of positive examples that F misclassifies is at least $\frac{R}{2n}$. Thus the probability of error with respect to D is at least $\frac{R}{4n \cdot r \cdot (r-1) \cdot |E| \cdot n^{r-2}}$ which, for sufficiently large n, is at least:

$$\begin{aligned} R/n^{r+4} &= \frac{\frac{1}{4} \cdot \left(\frac{\chi-1}{\ln n}\right)^r}{n^{r+4}} \geq \frac{\frac{1}{4} \cdot \left(\frac{n^{1-g/r}-1}{\ln n}\right)^r}{n^{r+4}} \\ &> \frac{\left(n^{1-2g/r}\right)^r}{n^{r+4}} = n^{-2g-4} = \frac{1}{n^{2g+4}}. \end{aligned}$$

In the second case, $|\cup_{k=1}^{\ell} Out_k| < R$. But then by Lemma 3.3.9, the number of negative examples misclassified is at least $\frac{1}{2} \cdot \left(\frac{\chi-1}{\ln n}\right)^r - R$ which is equal to R. Thus the probability of an error with respect to D is at least $\frac{R}{2n^r}$, which again is at least $\frac{1}{n^{2g+4}}$ for sufficiently large n.

Finally, we have reduced the problem of approximating $\chi(G)$ to learning CNF:

Theorem 3.3.10 *Suppose that CNF is efficiently learnable with membership queries by ANDs-of-thresholds in time $O(n^{kg(n)/2} \cdot s^k \cdot (\frac{1}{\epsilon})^k)$, where $k \geq 2$, and $2 \leq g(n) \leq n$ (recall s is the size of the CNF). Then there exists a randomized algorithm for approximating the chromatic number of a graph within a factor of $n^{1-1/8k}$ in time $O(n^{7kg(n)})$. Moreover, the algorithm will always give a valid answer for $\chi \geq n^{1-1/8k}$.*

Proof: Set $\epsilon = \frac{1}{n^{2g+4}}$ and $r = 8kg$. Let G be a graph and let f and $\mathcal{D}$ be the target function and the distribution induced from G as described previously. Run the learning algorithm with respect to distribution $\mathcal{D}$ using f to answer membership queries. If it does not terminate after n^{7kg} steps output "$\chi \geq n^{1-1/8k}$". Otherwise, let h be the hypothesis the algorithm outputs. Calculate the error ϵ_h of h with respect to the distribution $\mathcal{D}$. If $\epsilon_h < \frac{1}{n^{2g+4}}$ output "$\chi \leq n^{1/8k}$", otherwise output "$\chi \geq n^{1-1/8k}$". We claim that this algorithm works with probability at least 3/4 for sufficiently large n in approximating $\chi \leq n^{1/8k}$ and works perfectly for $\chi \geq n^{1-1/8k}$.

If $\chi \leq n^{1/8k}$, by Lemma 3.3.5, $s \leq n^g + r|E| \leq n^g + n^3 \leq 2n^{g+1}$. The number of variables in the underlying learning problem is $r \cdot n \leq n^2$. Hence the running time with probability $\geq 3/4$ is at most $O(n^{2\cdot kg/2} n^{(g+1)\cdot k} n^{(2g+4)k}) \leq O(n^{4kg+5k}) < n^{7kg}$ for sufficiently large n, and the output is supposed to have an error $< \epsilon = \frac{1}{n^{2g+4}}$. Hence the algorithm outputs "$\chi \leq n^{1/8k}$" with probability at least 3/4 in this case.

If $\chi \geq n^{1-1/8k}$, by Lemma 3.3.6 the output of the algorithm must contain at least $\frac{1}{2\chi r}\left(\frac{\chi-1}{\ln n}\right)^r$ terms in order to have an error $< \epsilon = \frac{1}{n^{2g+4}}$. In this case the running time of the algorithm (for n sufficiently large) is at least:

$$\frac{1}{2\chi r}\left(\frac{\chi-1}{\ln n}\right)^r \geq \frac{1}{16kn^2}\left(\frac{n^{1-1/8k}-1}{\ln n}\right)^r \geq \frac{1}{16kn^2}\frac{n^{8kg-g}}{(2\ln n)^{8kg}} > n^{7kg} .$$

Hence if the algorithm terminates in n^{7kg} steps, its error will be bigger than ϵ, and the algorithm outputs "$\chi \geq n^{1-1/8k}$" with probability 1 in this case. □

Remark 3.3.11 *By negating the CNFs and the ANDs-of-thresholds in Theorem 3.3.10, we obtain the following:*

Suppose that DNFs are learnable with MQs by ORs-of-thresholds in time $O(n^{kg(n)/2} \cdot s^k \cdot (\frac{1}{\epsilon})^k)$, where $k > 1$, and $2 \leq g(n) \leq n$. Then there exists a randomized algorithm for approximating the chromatic number of a graph within a factor of $n^{1-1/8k}$ in time $O(n^{7kg(n)})$. Moreover, the algorithm will always give a valid answer for $\chi \geq n^{1-1/8k}$.

We will require the following hardness result due to Feige and Kilian [FK96]:

Theorem 3.3.12 *[FK96] For any constant $\lambda > 0$, there exists a polynomial-time randomized reduction mapping instances f of SAT of length n to graphs G with $N = \mathsf{poly}(n)$ vertices with the property that if f is satisfiable then $\chi(G) \leq O(N^{\lambda})$ and if f is unsatisfiable then $\chi(G) \geq \Omega(N^{1-\lambda})$. The reduction has zero-sided error.*

An immediate corollary is that approximating the chromatic number is hard:

Corollary 3.3.13 *[FK96] Let $\lambda > 0$ be a constant. Assume there exists an algorithm which approximates the chromatic number of a graph with n vertices within a factor of $n^{1-\lambda}$ in $\mathsf{RTIME}(t(n))$ (with zero error if $\chi \geq n^{1-\lambda}$). Then $\mathsf{NP} \subseteq \mathsf{RTIME}(t(n^a))$ for some constant $a \geq 1$.*

Now we can combine Theorem 3.3.10 and Corollary 3.3.13 to prove Theorem 3.1.1.
Proof:(of Theorem 3.1.1) If DNF formulae are efficiently learnable with MQs by ORs-of-thresholds, we show how to approximate the chromatic number of a graph in polynomial-time to within a factor of n^{λ} for some small constant $\lambda > 0$. Let G be a graph on n vertices. From Remark 3.3.11 setting $g = 2$, we can approximate $\chi(G)$ within a factor of $n^{1-1/8k}$ in time $O(n^{14k})$ where k is a constant, with zero error for $\chi \geq n^{1-1/8k}$. Hence, by Corollary 3.3.13, $\mathsf{NP} \subseteq \mathsf{RTIME}(n^{O(1)}) = \mathsf{RP}$. □

From the proof of Theorem 3.3.10 we can see that it is hard to learn even n^{λ}-term DNF by n^b-term OR-of-thresholds in time n^b for any constant $b \geq 0$. We can, under a stronger hardness assumption, prove stronger hardness results for learning superpolynomial size DNF formulae (i.e., if we do not restrict our concept class to be polynomial-size DNF formulae):

Corollary 3.3.14 *Suppose that $\mathsf{SAT} \notin \mathsf{RTIME}(O(n^{n^{\beta}}))$ for some β. Then for any $k > 0$ there is $\alpha > 0$ such that DNF formulas are not properly learnable with MQs in time $O(n^{n^{\alpha}} \cdot s^k \cdot (\frac{1}{\epsilon})^k)$.*

Notice that if we assume $\mathsf{SAT} \notin \mathsf{RTIME}(2^{n^{\beta}})$ for some β and substitute $k = 2$ in Corollary 3.3.14 then we can conclude that DNF formulae are not properly learnable in time $O(n^{n^{\alpha}} \cdot (\frac{s}{\epsilon})^2)$ for some $\alpha < 1$. Alekhnovich *et al.* prove that DNF formula are properly learnable in time $2^{O((n \log s)^{1/2} \log n)}/\epsilon$, so our lower bound is fairly tight [ABF+04].

3.4 Intersections of Two Halfspaces

In this section we show that it is hard to learn the intersection of two halfspaces by the intersection of any constant number of halfspaces. This result may be especially interesting in light of the fact that it is not known how to learn (even non-properly) the intersection of two n-dimensional halfspaces in time less than $2^{O(n)}$.

The main idea is to apply recent hardness results on the hardness of *hypergraph* coloring. Recently, several researchers [GHS00, Kho02, DRS02] have shown that it is hard to color *uniform* hypergraphs, i.e., hypergraphs where each hyperedge is of equal size.

We start by reducing the problem of coloring a 3-uniform hypergraph to a consistency problem for intersections of halfspaces:

Theorem 3.4.1 *The problem of ℓ-coloring a k-colorable hypergraph on n vertices reduces to finding an intersection of ℓ halfspaces over n variables consistent with examples labelled by an intersection of k halfspaces.*

Proof: Let $H = (V, E)$ be a k-colorable hypergraph with n vertices. We construct a set of examples S classified by the intersection of k halfspaces such that any intersection of ℓ halfspaces consistent with S can be used to ℓ-color H (the reduction is an extension of the reduction by Blum and Rivest [BR92]).

Denote the vertices of H by $v_1, v_2, \ldots, v_n$. For a vertex $v_i \in V$, let $a(v_i)$ denote the vector of length n with a 1 in the ith position and 0 everywhere else. For an edge $e \subseteq V$ of G let $a(e)$ be the vector equal to $\sum_{v \in e} a(v)$ (that is, the characteristic vector for set e). Let 0^n denote the all zeroes vector of length n. Create the following set S of examples

- The example $(0^n, +)$.
- For every vertex $v \in V$, the example $(a(v), -)$.
- For every edge $e \in E$, the example $(a(e), +)$.

Assume H is colorable by k colors according to the function χ. We construct an intersection of k halfspaces consistent with S. Let $h_i = \mathsf{sign}(w_i \cdot x - \theta_i)$ where $\theta_i = -1/2$ and $w_i = (w_{i,1}, \ldots, w_{i,n})$ such that $w_{i,j}$ equals -1 if $\chi(v_j) = i$ and n otherwise. Set $h = \wedge_{i=1}^{k} h_i$.

Checking that $h_1 \wedge h_2, \ldots, \wedge h_k$ is consistent with S is straightforward; the example $(0^n, +)$ is satisfied and each $(a(v), -)$ example is consistent with the intersection. Finally given an edge e each $(a(e), +)$ is satisfied by each h_i since there exist two vertices in e which are colored in different colors and hence at least one of two vertices will contribute n to the weighted sum making the total positive.

For the other direction, assume there exists an intersection of ℓ halfspaces $h = h_1 \wedge \cdots \wedge h_\ell$ consistent with the examples in S. Construct a coloring for H as follows. Let $\chi(v) = t$ where t is the first halfspace h_t such that $h_t(a(v))$ is negative. This assigns a color to each vertex. Now let $e \in E$. If there exists a color c and edge e such that $\forall v \in e,\ \chi(v) = c$ then for all $v \in e,\ h_c(a(v)) = 0$. Since $h(0^n)$ is positive, it implies that $h_c(0^n)$ is positive, and hence its threshold θ_c is negative. But for all i such that $v_i \in e$, $h_c(a(v_i)) = 0$ and thus $w_{c,i} < \theta_c$. This implies $\sum_{v_i \in e} w_{c,i} < \theta_c$ and therefore $h_c(a(e)) = 0$ contradicting the consistency with S. □

Applying Theorem 3.4.1 and Theorem 3.3.2 we obtain Theorem 3.1.2, our main hardness result for this section.

Proof:(of Theorem 3.1.2) Assume there exists a polynomial time algorithm for learning the intersection of two halfspaces which outputs a hypothesis equal to an intersection of k halfspaces. Given a hypergraph H construct a set of examples S as above. Consider a distribution $\mathcal{D}$ which is uniform over this set of examples. Setting the error parameter $\epsilon = 1/(|S| + 1)$, run the algorithm to obtain with probability $3/4$ a hypothesis h equal to the intersection of k halfspaces. The algorithm will run in time polynomial in n and $1/\epsilon = |V| + |E| + 2$, that is, will be polynomial in the size of H. Since $\epsilon < 1/|S|$, h must be consistent with S. Hence from Theorem 3.4.1 we can reconstruct a coloring for H. □

Dinur *et al.* [DRS02] also give the following hardness result under a slightly stronger assumption:

Theorem 3.4.2 ([DRS02]) *If* $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{\log^{O(1)} n})$ *then there is no polynomial time algorithm for coloring a* 2*-colorable* 3*-uniform hypergraph using* $O((\log\log n)^{1/3})$ *colors.*

We obtain a corresponding hardness result:

Corollary 3.4.3 *There is no efficient algorithm for learning intersections of two halfspaces by intersections of* $O((\log\log n)^{1/3})$ *halfspaces unless* $\mathsf{NP} \subseteq \mathsf{RTIME}(2^{\log^{O(1)} n})$.

We can also prove a hardness result for learning two-node neural networks by neural networks with a constant number of origin-centered hidden nodes:

Theorem 3.4.4 *Coloring a* k*-colorable hypergraph with* 2^ℓ *colors reduces to the problem of finding any function of* ℓ *origin-centered halfspaces consistent with a data set labelled by the intersection of* k *origin-centered halfspaces.*

Proof: Let $H = (V, E)$ and use the same reduction as in the proof of Theorem 3.4.1 to obtain a set of examples S without the 0^n example. We first note that if we define $h = \wedge_{i=1}^{\ell} h_i$ where h_i's are defined as before but with thresholds $\theta_i = 0$ we will get an intersection of k origin-centered halfspaces consistent with S.

For the other direction let f be a Boolean function of ℓ origin-centered halfspaces $h_1, \ldots, h_\ell$ consistent with S. Put 2^ℓ colors in correspondence with every subset of ℓ

halfspaces. Color v the color corresponding to which subset of the ℓ halfspaces are negative on input $a(v)$. Assume e is a monochromatic edge. Then each $v \in e$ is set negative by the same subset of halfspaces. That is for every $j \leq \ell$ and $v_{i_1}, v_{i_2} \in e$, $h_j(a(v_{i_1})) = h_j(a(v_{i_2}))$. h_j is zero centered and hence $\mathsf{sign}(w_{j,i_1}) = \mathsf{sign}(w_{j,i_2})$. This implies that for every $v \in e$,

$$h_j(a(e)) = \mathsf{sign}(\sum_{v_i \in e} w_{j,i}) = h_j(a(v))$$

and thus $f(a(e)) = f(a(v))$ which contradicts the consistency with S. $\square$

Now we can apply the hardness result due to Dinur *et al.* [DRS02]. For any learning algorithm outputting some representation of a function of ℓ halfspaces in polynomial time we have the following:

Corollary 3.4.5 *It is* NP*-hard to learn three-node neural networks by outputting a neural network with a constant number of hidden, origin-centered nodes.*

Chapter 4

Hardness of Approximate DNF Minimization and Learning

In this chapter we prove that learning of DNF expressions by DNF expressions of somewhat larger size is NP-hard even when learning with respect to the uniform distribution and using membership queries. The result and techniques are incomparable to those of Chapter 3 since learning often becomes easier when the distribution is uniform but we only rule out a restricted form of proper learning.

The result is obtained via a new connection to the problem of *two-level logic minimization* or finding a minimal DNF formula consistent with a given complete truth table (TT-MinDNF). This problem was formulated by Quine in 1952 and has been since one of the key problems in logic design. We prove that TT-MinDNF for a function of d variables is NP-hard to approximate within d^{γ} for some constant $\gamma > 0$, establishing the first inapproximability result for the problem. This result is of independent interest and will be the main focus of this chapter.

4.1 Introduction

The problem of finding a minimal-size disjunctive normal form expression consistent with a given truth table (TT-MinDNF) is one of the oldest problems in computer science. It was formulated by the famous logician and philosopher Willard Van Quine in his work on mathematical logic [Qui52, Qui56]. His algorithm for simplifying logical steps was also discovered in 1956 by Edward McCluskey in the context of circuit design [McC56]. Besides its important role in circuit design (in particular, two-level and multi-level logic synthesis for VLSI design of ASICs and Programmable Gate Arrays [CS01]) the problem has more recently appeared in reliability analysis [CM94], IP routing table compaction

[Liu02], and high-dimensional data representation [AGGR05]. This array of applications has led to an ongoing effort by many researchers to seek efficient heuristic and exact minimization procedures. We direct the interested reader to [CS01] for an overview of a large number of publications and some software tools. In the original Quine-McCluskey algorithm and in most of the later approaches, after a number of simplification steps the problem is reduced to an instance of the classical SET-COVER problem. Then, either an exact solution is found via the brute-force search, or an approximate solution is found using a certain heuristic. In the former case the size of the search space is not theoretically analyzed and in the latter no guarantees on the quality (i.e. size) of the output are given (both are usually measured empirically).

Far less work has been done on the theoretical side of this problem. Gimpel [Gim65] and Paul [Pau74] showed that Quine-McCluskey method can produce instances of SET-COVER that are NP-hard to solve. Then, in 1979, the full truth table version was proven NP-complete by Masek [Mas79] (his manuscript was not published but the proof can be found in surveys by Czort [Czo99] and Umans *et al.* [UVSV06]). Inapproximability results are only known for a generalization of TT-MinDNF that allows "don't care" values in the truth table (i.e., the truth table is partial). Allender *et al.* prove that this problem (we denote it by PTT-MinDNF) is NP-hard to approximate within any constant factor and cannot be approximated within $\log d$ factor unless $\mathsf{NP} \subseteq \mathsf{RTIME}(n^{polylog(n)})$, where d is the number of variables [AHPM04]. Using Gimpel's reduction from PTT-MinDNF to TT-MinDNF they also produced a simpler proof (than Masek's) for NP-hardness of TT-MinDNF.

On the approximation side the only known efficient approximating algorithm is the one resulting from using the greedy algorithm to solve the SET-COVER instance obtained in Quine-McCluskey algorithm [Chv79]. It gives $\ln 2^d = O(d)$ approximation factor.

In this paper we present the first result on hardness of approximating TT-MinDNF. More specifically, we prove the following theorem.

Theorem 4.1.1 *There exists a constant $\gamma > 0$ such that it is NP-hard to approximate TT-MinDNF to within a factor d^γ, where d is the number of variables of the TT-MinDNF instance.*

This result implies that the approximation factor achieved by the greedy algorithm is at most polynomially larger than the best possible. In addition we prove the first hardness of approximation result for a natural restriction of SET-COVER problem, in which the ground set is $\{0,1\}^d$ and all subsets are subcubes (see Section 4.2 for a formal definition).

The unpublished results of Allender *et al.* were recently substantially strengthened by the same authors and Paul McCabe [AHM$^+$06]. They prove NP-hardness of approximating

TT-MinDNF within any constant factor and also show how to get the d^γ factor under the assumption that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{polylog(n)})$. Their independent work is based on a similar approach.

Learning is another context where finding a small DNF formula consistent (or almost) with the given data is a fundamental problem. The problem was formulated by Leslie Valiant in his seminal paper introducing the PAC model of learning [Val84] and has been the subject of numerous subsequent works. A number of questions related to PAC learning of DNF expressions originated in his work. In particular, he asked whether DNF expressions are learnable from random examples with or without the use of the membership query (MQ) oracle [Val84, Val85]. Valiant's original definition required that the learning algorithm output a DNF expression but this restriction was later relaxed to any efficiently-computable hypothesis with the stricter version being referred to as *proper* learning. Hardness of proper learning DNF from random examples only in the distribution independent PAC learning was recently established by Alekhnovich *et al.* [ABF+04]. Feldman has extended their proof to learning with MQs [Fel06a] (both results are presented in Chapter 3).

We observe that hardness of TT-MinDNF implies hardness of *strongly proper* learning of DNF expressions with MQs even with respect to the uniform distribution, where strongly proper means that the size (number of terms) of a hypothesis has to be upper-bounded by the DNF-size of the target function. Our inapproximability result then translates to hardness even when the size of a hypothesis is $O(\log^\gamma(s/\epsilon))$ times larger than the size of the target. We note that, as proved by Jackson, unrestricted DNF expressions are learnable non-properly in this strong model [Jac97] (see also Chapter 5), and hence our result highlights the importance of knowledge representation in this model.

4.1.1 Relation to Other Work

Besides the results that we have already mentioned, one of the most significant results in DNF minimization is Umans' proof that finding a minimum DNF formula for a function given by a DNF formula (also called finding a minimum equivalent DNF and denoted MinEquDNF) is Σ_2^p-hard to approximate within N^γ for some constant $\gamma > 0$, where N is the size of the given DNF formula [Uma99]. Despite the same goal in both problems the difference in input makes the nature of the problem (and, eventually, the proof techniques) very different. In particular, the gaps differ exponentially in terms of the size of hard instances. Hardness results for some other variants of DNF minimization can be found in a survey by Umans *et al.* [UVSV06].

Hardness results for learning of DNF expressions with MQs are also known for the *exact* model of learning (which is weaker than PAC learning). The strongest results in this model are due to Hellerstein and Raghavan [HR02] and are based on information-theoretic hardness. For proper PAC learning without MQs a number of hardness results are known for several other representations [BR92, Gol78, KLPV87a, ABF$^+$04].

When learning with respect to the uniform distribution DNF expressions are known to be PAC learnable (non-properly) with MQs [Jac97] and PAC learnable properly in time $n^{O(\log(s/\epsilon))}$ [Ver90].

4.1.2 Outline and Organization

The proof of the TT-MinDNF hardness result has two key components. The first one is a reduction from a more general problem of covering a subset of the Boolean hypercube with a given set of subcubes (we denote it by PHC-COVER) to TT-MinDNF. PHC-COVER can be seen as a geometric version of the general SET-COVER problem. The second component of the proof is a reduction from a multi-prover proof system with certain simple properties to PHC-COVER. This reduction follows the key ideas of the inapproximability result for SET-COVER by Lund and Yannakakis [LY94] and its generalization by Bellare *et al.* [BGLR93]. Finally, a low-error PCP by Raz and Safra [RS97] is used to obtain a multi-prover proof system with the desired properties, yielding the inapproximability result for TT-MinDNF. The low-error PCP of Raz and Safra was used in a similar way to obtain hardness of approximating within $\Omega(\log n)$ for SET-COVER under the assumption that $\mathsf{P} \neq \mathsf{NP}$ [RS97].

Besides the main reduction in Section 4.4.1, we show a reduction from hypergraph vertex cover problem to PHC-COVER. The reduction is based on families of sets in which none of the sets is covered by k others. This reduction together with a recent result by Dinur *et al.* [DGKR05] implies the same inapproximability result for TT-MinDNF under a stronger assumption $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{\log(n)})$. This reduction is more direct and simple than both the reduction given in Section 4.4.1 and the reduction of Allender *et al.* [AHM$^+$06] (which gives the same result).

The rest of the chapter is organized as follows. In Section 4.3 we show that TT-MinDNF and two other covering problems on the hypercube can be reduced (in an approximation-preserving way) to PHC-COVER. Then, in Section 4.4, PHC-COVER is reduced to the hypergraph vertex cover and the low-error PCP by Raz and Safra [RS97] giving the desired hardness of approximation results. In Section 4.5 we prove the above-mentioned hardness result for proper learning with MQs.

4.2 Preliminaries

A Boolean partial function is a function $f : \{0,1\}^d \rightarrow \{0,1,*\}$. We say that a Boolean function g is consistent with a partial function f, if for every $a \in \{0,1\}^d$ such that $f(a) \neq *$, $g(a) = f(a)$. A subcube of a Boolean hypercube is a set $I_1 \times I_2 \times \cdots \times I_d$ where for each j, $I_j \subseteq \{0,1\}$. We identify each subcube with a term whose satisfying assignments are exactly the elements of the subcube.

The *size* of a DNF formula is the number of terms in it. The DNF-size of a function is the size of a minimum DNF formula equal to the function. Given the truth table of a function f the problem of finding the DNF-size of f is denoted TT-MinDNF. When f is a partial function the problem of finding the size of a minimum DNF consistent with f is referred to as PTT-MinDNF.

The problem of finding the size of a minimum cover of the d-dimensional Boolean hypercube with subcubes represented by the terms in $\mathcal{T} = \{T_i\}_{i=1}^m$ is referred to as HC-COVER. We also consider the following generalization of HC-COVER. Given a set of terms as above and a set of points $S \subseteq \{0,1\}^d$ find the size of a minimum cover of S by terms in $\mathcal{T}$. We refer to this generalized version as PHC-COVER. We say that PTT-MinDNF(f) $= C$ (HC-COVER($\mathcal{T}$) $= C$, or PHC-COVER($S, \mathcal{T}$) $= C$) if the size of a minimum DNF formula consistent with f (or, respectively, a cover for an instance $\mathcal{T}$ or $(S, \mathcal{T})$) equals C.

In all the above problems, we assume that the input is of size poly(2^d) (it cannot be larger as there are 3^d different terms). For PHC-COVER and HC-COVER the input can, in certain situations, be represented more concisely. However, for consistency with the definition of the usual set cover problem, we assume that all the 2^d points of the cube are given explicitly as part of the input. Hardness results for this setting imply hardness results for more concise representations of the same problem.

We use a dot '$\cdot$' to denote concatenation of bits and bit vectors. Unless defined otherwise we use a subscript to refer to an individual coordinate of a vector and use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. Let $\texttt{par}()$ denote the parity function defined for any bit vector. For any Boolean variable v and $b \in \{0,1\}$, let $\ell_v(b) = v$, if $b = 1$ and $\ell_v(b) = \bar{v}$, if $b = 0$. Similarly, for a vector of variables $w \in V^r$ and a vector $a \in \{0,1\}^r$, we define $\texttt{eq}(w, a) = \wedge_{i \leq r} \ell_{w_i}(a_i)$, or simply the term that checks if variables of w are set to a.

Our learning model is Valiant's well-known Probably Approximately Correct (PAC) learning model [Val84] defined in Section 2.2. A proper learning algorithm for a representation class $\mathcal{F}$ is referred to as *strongly* proper if, it outputs a hypothesis $h \in \mathcal{F}$ such that $\mathcal{F}\texttt{-size}(h) \leq \mathcal{F}\texttt{-size}(c)$, where c is the target concept.

4.3 Hypercube Reductions

Below we show that the covering problems defined in the previous section have similar approximation complexity by describing efficient reductions from PHC-COVER to PTT-MinDNF, from PTT-MinDNF to TT-MinDNF, and from TT-MinDNF to HC-COVER. Our reductions preserve the approximation ratio and increase the number of variables by a small constant factor.

4.3.1 From PHC-COVER to PTT-MinDNF

It can be easily seen that PTT-MinDNF is an instance of PHC-COVER. For the other direction our reduction converts an instance of PHC-COVER given by a set $S \subseteq \{0,1\}^d$ and a set of terms $\mathcal{T}$, to an instance of PTT-MinDNF given by a function f where each element of $\mathcal{T}$ corresponds to a *prime implicant* of f. A prime implicant of a function f is a term T such that T is consistent with f (that is it does not accept points where f equals 0) and is not covered properly by another term consistent with f. Any DNF formula consistent with f can always be easily converted to a DNF formula of the same size that includes only prime implicants of f. Therefore, any DNF formula for f produced by our reduction corresponds to a cover of S by terms from $\mathcal{T}$. We now provide the details of this mapping.

Theorem 4.3.1 *There exists a polynomial-time algorithm that given an instance $(S, \mathcal{T})$ of PHC-COVER over d variables produces the truth table of partial function f over $2d$ variables such that $(S, \mathcal{T})$ has a cover of size C if and only if there exists a C-term DNF formula consistent with f.*

Proof: For a point $x \in \{0,1\}^d$, let $p[x]$ denote a point in $\{0,1\}^{2d}$ equal to $x \cdot \bar{x}$ (that is, x on first d coordinates and the bit complement of x on coordinates from $d+1$ to $2d$). For a term T over d variables, let $p[T]$ denote a term over $2d$ variables in which all the positive literals are the same as in T while each negative literal $\bar{x}_i$ is replaced by literal x_{d+i}. Let $g(y) = \bigvee_{T \in \mathcal{T}} p[T](y)$. Then we map $(S, \mathcal{T})$ to the instance of PTT-MinDNF given by the following function:

$$f(y) = \begin{cases} 0 & \text{if } g(y) = 0 \\ 1 & \text{if } y = p[x] \text{ and } x \in S \\ * & \text{otherwise} \end{cases}$$

Let $\mathcal{S} \subseteq \mathcal{T}$ be a set of C terms such that $S \subseteq \cup_{T \in \mathcal{S}} T$. We claim that $h(y) = \bigvee_{T \in \mathcal{S}} p[T](y)$ is consistent with f. Let y be a point in $\{0,1\}^{2d}$. If $f(y) = 0$, then $g(y) = 0$ and so $h(y) = 0$. If $f(y) = 1$ then there exists x such that $y = p[x]$ and $x \in S$. Therefore, there exists $T \in \mathcal{S}$ such that $T(x) = 1$, which is equivalent to $p[T](p[x]) = 1$. In particular, $h(y) = 1$, which completes the proof of the claim.

For the other direction, let $h = \bigvee_{Z \in \mathcal{Z}} Z$ be a C-term DNF formula consistent with f. For a term $Z \in \mathcal{Z}$, let y be the point with the minimum number of 1's accepted by Z. By the consistency with f, we get that $f(y) \neq 0$ and hence $g(y) = 1$. Therefore let $m(Z)$ be a term of g that covers y and let $T_Z \in \mathcal{T}$ be some term for which $m(Z) = p[T_Z]$. We claim that $Z \subseteq m(Z)$. If for a point z, $Z(z) = 1$ then for every $i \leq 2d$, if $z_i = 0$ then $y_i = 0$. This is true since if $z_i = 0$ then Z does not include literal x_i and, therefore, by the minimality of y, $y_i = 0$. The term $m(Z) = p[T_Z]$ is monotone and, therefore, if it covers y then it covers z. This implies the claim that $Z \subseteq m(Z)$.

Define $\mathcal{T}' = \{T_Z \mid Z \in \mathcal{Z}\}$. If $x \in S$, then $f(p[x]) = 1$ and therefore, there exists $Z \in \mathcal{Z}$ such that $Z(p[x]) = 1$. This, in turn, implies that $T_Z(x) = 1$, that is, $\mathcal{T}'$ is a set of C subsets from $\mathcal{T}$ that covers S. $\square$

4.3.2 From PTT-MinDNF to TT-MinDNF

The next step is an approximation preserving reduction from a partially-specified truth table to a fully-specified one. A part of this reduction is based on Gimpel's reduction from partially to fully specified truth-table [Gim65].

Theorem 4.3.2 *There exists an algorithm that given the truth table of a partial function f on d variables and an integer $r \geq 1$ produces the truth table of partial function g over $d + r + 2$ variables such that there exists a C-term DNF consistent with f if and only if there exists $(2^{r-1}C + |f^{-1}(*)|)$-term DNF formula equal to g. The algorithm runs in time $2^{O(r+d)}$.*

Proof: The reduction has two components. The first component is Gimpel's reduction [Gim65]. It converts f to a fully-specified function that has a distinct prime implicant for each point x where f equals $*$ thereby forcing any consistent DNF to include a term for every $*$ of the original function. The addition of new terms does not preserve approximation factors and therefore the second component replicates f 2^{r-1} times to ensure that the size of the cover is still dominated by the original problem (for large enough r). For a vector in $\{0,1\}^{d+r+2}$, we refer to its first d coordinates as $x_1, \ldots, x_d$; its next r variables as $y_1, \ldots, y_r$; and its last two variables as z_1, z_2. We define Boolean function g over $\{0,1\}^{d+r+2}$ as follows:

$$g(xyz) = \begin{cases} \texttt{par}(y) & \text{if } f(x) = 1 \text{ and } z = 11 \\ 1 & \text{if } f(x) = * \text{ and} \\ & (z = \texttt{par}(x) \cdot \neg\texttt{par}(x) \text{ or } z = 11) \\ 0 & \text{otherwise} \end{cases}$$

Let $S = f^{-1}(*)$. We claim that there exists a C-term DNF consistent with f if and only if there exists a $(2^{r-1}C + |S|)$-term DNF equal to g. For the simpler direction, let $\mathcal{S}$ be a set of C terms such that $h(x) = \bigvee_{T \in \mathcal{S}} T(x)$ is consistent with f. For $b \in \{0,1\}$ we define $\texttt{sw-z}(b) = z_{2-b}$ (that is, $\texttt{sw-z}$ switches between z_1 and z_2 according to b). We claim that

$$g(xyz) \equiv (h(x) \wedge \texttt{par}(y) \wedge z_1 \wedge z_2) \bigvee ((f(x) = *) \wedge \texttt{sw-z}(\texttt{par}(x)))\,.$$

By definition, the expression $R(xyz) \equiv (f(x) = *) \wedge \texttt{sw-z}(\texttt{par}(x))$ equals to $g(xyz)$ on all points with $z \neq 11$. In addition, for $z = 11$, $R(xy \cdot 11) \equiv (f(x) = *)$. The expression $L(xyz) = h(x) \wedge \texttt{par}(y) \wedge z_1 \wedge z_2$ only covers points for which $z = 11$ and $L(xy \cdot 11) = h(x) \wedge \texttt{par}(y)$. Therefore, by the consistency of h with f, $L(xy \cdot 11)$ equals to $\texttt{par}(y)$ when $f(x) = 1$ and does not cover any points for which $f(x) = 0$. Altogether, $g(xyz) \equiv L(xyz) \vee R(xyz)$.

Expressions $L(xyz)$ and $R(xyz)$ are not in DNF. To convert them to DNF we note that

$$\texttt{par}(y) \equiv \bigvee_{a \in \{0,1\}^r, \texttt{par}(a)=1} \texttt{eq}(y, a)$$

and

$$(f(x) = *) \equiv \bigvee_{a \in S} \texttt{eq}(x, a)\ .$$

Therefore

$$g(xyz) \equiv \left(\bigvee_{T \in \mathcal{S}, a \in \{0,1\}^r, \texttt{par}(a)=1} T \wedge \texttt{eq}(y, a) \wedge z_1 \wedge z_2 \right) \bigvee \left(\bigvee_{a \in S} \texttt{eq}(x, a) \wedge \texttt{sw-z}(\texttt{par}(a)) \right),$$

that is, g has a DNF expression with $C2^{r-1} + |S|$ terms.

For the other direction, let $\mathcal{T}$ be a set of $C2^{r-1} + |S|$ terms such that $g(xyz) = \bigvee_{T \in \mathcal{T}} T(xyz)$. For each $a \in S$, let $\tau_a \in \mathcal{T}$ be a term that accepts point

$$p(a) = a \cdot 0^r \cdot \texttt{par}(a) \cdot \neg\texttt{par}(a)\ .$$

We first prove that τ_a contains all the literals of $\mathtt{eq}(x,a)$. If τ_a does not contain literal $\ell_{x_i}(a_i)$, then let a^i be the point a with the i-th bit negated. Clearly τ_a will also accept the point $a^i \cdot 0^r \cdot \mathtt{par}(a) \cdot \neg\mathtt{par}(a)$. But this contradicts the consistency with g, since $\mathtt{par}(a) = \neg\mathtt{par}(a^i)$. It follows that for each $a \in S$, there is a *distinct* term in $\mathcal{S}$ that can only accept points with x part in S. We denote this set of terms by $\mathcal{T}^*$.

Now let $D = \{p \mid p \in \{0,1\}^r, \mathtt{par}(p) = 1\}$, p be any point in D and a be any point such that $f(a) = 1$. Then, by definition of g, there exists a term $\tau_{p,a}$ that accepts the point $a \cdot p \cdot 11$. We claim that $\tau_{p,a}$ contains all the literals of $\mathtt{eq}(y,p)$. If $\tau_{p,a}$ does not contain literal $\ell_{y_i}(p_i)$, then let p^i be the point p with the i-th bit negated. Clearly $\tau_{p,a}$ will also accept the point $a \cdot p^i \cdot 11$. But this contradicts the consistency with g, since $g(a \cdot p^i \cdot 11) = \mathtt{par}(p^i) = 0$. Now let $\mathcal{T}_p = \{\tau_{p,a} \mid f(a) = 1\}$ and let $h_p(x) = \bigvee_{T \in \mathcal{T}_p} T(x \cdot p \cdot 11)$. We claim that $h_p(x)$ is consistent with f. This is true since if $f(a) = 1$ then $\tau_{p,a} \in \mathcal{T}_p$ and $\tau_{p,a}(a \cdot p \cdot 11) = 1$. If $f(a) = 0$ then $g(a \cdot p \cdot 11) = 0$ and since $\mathcal{T}_p \subseteq \mathcal{T}$ then no term in $\mathcal{T}_p$ can accept point $a \cdot p \cdot 11$. As we have shown, all the $\mathcal{T}_p$'s for $p \in D$ are disjoint and they are clearly disjoint from $\mathcal{T}^*$ (since $0^r \notin D$). Therefore $|\mathcal{T}| \geq |S| + \sum_{p \in D} |\mathcal{T}_p|$. As $|D| = 2^{r-1}$ we get that there exists p such that $|\mathcal{T}_p| \leq C$ and hence h_p is a C-term DNF formula consistent with f. □

By a suitable choice of r in Theorem 4.3.1 one easily obtains the following corollary (the proof is omitted for brevity):

Corollary 4.3.3 *If TT-MinDNF can be approximated within $h(d)$ in time $t(d)$ then PTT-MinDNF can be approximated within $h(2d + \log d) + 1$ in time $t(2d + \log d) + 2^{O(d)}$.*

4.3.3 From TT-MinDNF to HC-COVER

We now give a simple reduction proving that finding a minimum cover of the whole hypercube by terms from a restricted set $\mathcal{T}$ is not substantially easier than finding a minimum cover of a subset of the hypercube. Note that this reduction is not required as a step in our main result and is provided to extend our hardness of approximation results to HC-COVER.

Theorem 4.3.4 *There exists an algorithm that, given the truth table of a function f on d variables and an integer $r \geq 1$, produces a set of terms $\mathcal{T}$ over $d + r$ variables such that there exists a C-term DNF expression equal to f, if and only if, $\{0,1\}^{d+r}$ can be covered by $2^r C + |f^{-1}(0)|$ terms from $\mathcal{T}$. The algorithm runs in time $2^{O(r+d)}$.*

Proof: The idea of the proof is to create $\mathcal{T}$ that contains all the terms consistent with f and terms that cover $\neg f$. As in the proof of Theorem 4.3.2, we will replicate f many times to preserve the approximation ratio .

Our instance of the HC-COVER problem is over $d+r$ variables where we refer to the first d variables as $x_1, \dots x_d$ and to the next r variables as $y_1, \dots, y_r$. Let $\mathcal{T}^f$ be the set of all the terms consistent with f (we can assume that it includes only the prime implicants of f but this is not essential). For $p \in \{0,1\}^r$ let $\mathcal{T}^f_p = \{T \wedge \mathtt{eq}(y,p) \mid T \in \mathcal{T}^f\}$, let $S = f^{-1}(0)$ and $\mathcal{T}^{\neg f} = \{\mathtt{eq}(x,a) \mid a \in S\}$. Then we define

$$\mathcal{T} = \mathcal{T}^{\neg f} \bigcup \left(\bigcup_{p \in \{0,1\}^r} \mathcal{T}^f_p \right).$$

We claim that there exists a C-term DNF equal to f, if and only if, there exists a set $\mathcal{S} \subseteq \mathcal{T}$ of size $C2^r + |S|$ that is a cover of $\{0,1\}^{r+d}$. Let $\mathcal{S}^f$ be a set of C terms such that $\bigvee_{T \in \mathcal{S}^f} T = f$. Then it is clear that

$$\mathcal{S} = \mathcal{T}^{\neg f} \bigcup \left\{ T \wedge \mathtt{eq}(y,p) \mid p \in \{0,1\}^r,\ T \in \mathcal{S}^f \right\}$$

is a cover of $\{0,1\}^{r+d}$, has size $C2^r + |S|$, and includes only terms from $\mathcal{T}$.

For the other direction, let $\mathcal{S} \subseteq \mathcal{T}$ be a cover of $\{0,1\}^{d+r}$. We observe that the only way to cover $S \times \{0,1\}^r$ is by including all the terms of $\mathcal{T}^{\neg f}$ in $\mathcal{S}$. For each $p \in \{0,1\}^r$, let $\mathcal{S}_p = \mathcal{S} \cap \mathcal{T}^f_p$. Only terms in $\mathcal{S}_p$ cover the subset $f^{-1}(1) \times \{p\}$ and therefore $h_p(x) = \bigvee_{T \in \mathcal{S}_p} T(x \cdot p)$ equals exactly $f(x)$. All the $\mathcal{S}_p$'s are mutually disjoint and are disjoint from $\mathcal{T}^{\neg f}$. Therefore if $|\mathcal{S}| \le C2^r + |S|$, then $\mathtt{DNF\text{-}size}(f) \le C$. $\square$

As with Theorem 4.3.2, we obtain the following corollary.

Corollary 4.3.5 *If HC-COVER can be approximated within $h(d) = o(d)$ in time $t(d)$, then TT-MinDNF can be approximated within $h(2d+\log d)+1$ in time $t(2d+\log d)+2^{O(d)}$.*

We summarize the reductions in this section by the following equivalence theorem:

Theorem 4.3.6 *If there exists a constant $0 < \gamma \le 1$ such that there is no polynomial-time algorithm approximating PHC-COVER, to within a factor d^γ then there is no polynomial-time algorithm approximating TT-MinDNF, PTT-MinDNF and HC-COVER to within a factor $\Omega(d^\gamma)$.*

4.4 Hardness of Approximation

Below we prove hardness of approximating TT-MinDNF by presenting two reductions to PHC-COVER both showing hardness of approximating within a factor of d^γ for a constant $\gamma > 0$. The first one is a reduction from the problem of finding a vertex cover of a k-uniform hypergraph. It is simple (relative to the other reduction and the reduction by

Allender *et al.* [AHM+06]) but relies on a stronger assumption $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{\log(n)})$. The second one is a general reduction from one-round multi-prover proof systems. When used with a low-error PCP it gives NP-hardness of approximating within a factor of d^{γ}. In addition, it makes an explicit connection between γ and standard parameters of a proof system that might be useful in obtaining inapproximability factors with specific or optimal exponent γ.

4.4.1 Reduction from Hypergraph Vertex Cover to PHC-COVER

A k-uniform hypergraph $H = (V, E)$ consists of a set of vertices V and a collection E of k-element subsets of V called hyperedges. A *vertex cover* of H is a subset $S \subseteq V$ such that every hyperedge in E intersects S. The Ek-Vertex-Cover problem is the problem of finding a minimum size vertex cover on a k-uniform hypergraph. The problem is alternatively called the minimum hitting set problem with sets of size k and is equivalent to the set cover problem where each element of the universe occurs in exactly k sets.

The first explicit hardness result shown for Ek-Vertex-Cover was due to Trevisan who showed an inapproximability factor of $k^{1/19}$ [Tre01] (and a comparable result is implicit in Feige's proof of inapproximability of SET-COVER [Fei98]). As we aim at obtaining a large inapproximability factor for covering the hypercube, we are interested in results that hold for large values of k. The first result stating the range of k explicitly is due to Dinur *et al.* [DGKR05] who give the following theorem.

Theorem 4.4.1 *There exists some $c > 0$ such that unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\log\log(n)})$, there is no polynomial-time algorithm for approximating Ek-Vertex-Cover for $k \leq (\log M)^{1/c}$ to within a factor of $\lfloor k/2 \rfloor - 0.01$, where M is the number of hyperedges in the k-uniform hypergraph.*

Remark 4.4.2 *It can be easily seen from the proof of this theorem, that the number of vertices N is smaller than M and therefore the result can be stated with the number of vertices N in place of M.*

Union-free Families

A family of sets $\mathcal{F}$ is called k-union-free if $A_0 \not\subseteq A_1 \cup A_2 \cup \cdots \cup A_k$ for all distinct $A_0, A_1, \ldots, A_k \in \mathcal{F}$. They were introduced by Kautz and Singleton [KS64] (and then rediscovered by Erdös *et al.* [EFF85]). A family of sets $\mathcal{F}$ is a (s, ℓ)*-combinatorial design* if each set in $\mathcal{F}$ has size s and the intersection of any two sets in $\mathcal{F}$ has size at most ℓ. If $\ell < s/k$ then (s, ℓ)-combinatorial design is k-union-free. The first efficient construction of combinatorial designs was given by Nisan and Widgerson [NW94]. For our purposes,

k-union-free families can be obtained by derandomizing a straightforward randomized construction using the method of conditional probabilities (*cf.* [Vad04, Lecture 21]).

Theorem 4.4.3 *There exists a k-union-free family of sets over $[d]$ of size m for $d = O(k^2 \log m)$. Moreover, such $\mathcal{F}$ can be constructed in time poly(m, d).*

Simple Reduction to PHC-COVER

Below we present a reduction from SET-COVER with each point occurring in k sets to PHC-COVER.

Theorem 4.4.4 *There exists a polynomial-time algorithm that, given a k-uniform hypergraph $H = (V, E)$ with $|V| = N$, produces an instance $(S, \mathcal{T})$ of PHC-COVER over $d = O(k^2 \log N)$ variables such that H has a vertex cover of size C, if and only if, $(S, \mathcal{T})$ has a cover of size C. The algorithm runs in time $O(N2^d)$.*

Proof: We first transform the k-uniform hypergraph vertex cover problem to its dual set cover problem with each point occurring in k sets. That is, for $v \in V$, let $S_v = \{e \mid e \in E, v \in e\}$ and $\mathcal{S} = \{S_v \mid v \in V\}$. Then $(E, \mathcal{S})$ is the equivalent instance of SET-COVER. Let $\mathcal{F} = \{P_v\}_{v \in V}$ be a k-union-free family (with N elements indexed by nodes in V). By Theorem 4.4.3, such $\mathcal{F}$ exists and can be efficiently constructed for $d = O(k^2 \log N)$. For any set $P \subseteq [d]$, let $\chi(P)$ be a characteristic vector of P, that is vector with $\chi(P)_i = 1$ when $i \in P$ and $\chi(P)_i = 0$, otherwise. For each $e \in E$, let $x^e = \chi(\cup_{v \in e} P_v)$. We define $\mathcal{T} = \{\mathtt{eq}(x, \chi(P_v)) \mid v \in V\}$, and define $S = \{x^e \mid e \in E\}$.

To prove the correctness of this reduction all we need to show is that for each $e \in E$ and $v \in V$, $\mathtt{eq}(x, \chi(P_v))$ covers x^e, if and only if, $e \in S_v$ or, in other words, $v \in e$. If $v \in e$ then $P_v \subseteq \cup_{u \in e} P_u$ and, therefore $x^e_i = 1$ for all $i \in P_v$. This implies that $\mathtt{eq}(x, \chi(P_v)) = \wedge_{i \in P_v} x_i$ accepts x^e. On the other hand, if $v \notin e$ then for each $u \in e$, by the properties of $\mathcal{F}$, $P_v \not\subseteq \cup_{u \in e} P_u$. This implies that $\mathtt{eq}(x, \chi(P_v))$ will not accept x^e. □

Corollary 4.4.5 *There exists a constant $\gamma > 0$ such that, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\log(n)})$, there is no polynomial-time algorithm approximating PHC-COVER to within a factor d^γ, where d is the number of variables in the PHC-COVER instance.*

Proof: Given an instance of SAT on n variables, the reduction of Theorem 4.4.1 produces an instance of Ek-Vertex-Cover on $N = n^{O(\log \log n)}$ vertices for $k = (\log N)^{1/b}$, where $b = \max\{3, c\}$. The gap in vertex cover sizes between positive and negative instances is αk ($\alpha \approx 1/2$). Then reduction in Theorem 4.4.4 will produce an instance of PHC-COVER over $d = O(k^2 \log N) = O((\log N)^{1+2/b})$ with the same gap of αk, that in terms of d, is

$\Omega(d^{1/(b+2)}) > d^\gamma$ for any constant $\gamma < \frac{1}{2+b}$ (and large enough d). The running time of the reduction and the produced instance are both bounded by

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

(since $b \geq 3$). □

4.4.2 Reducing from Multi-prover Proof Systems

Below we prove hardness of approximating PHC-COVER by presenting a direct reduction from one-round multi-prover proof systems with certain properties to PHC-COVER. We then obtain the claimed result by coupling our reduction with the low-error PCP for NP due to Raz and Safra [RS97]. The reduction simulates the reduction from SET-COVER given by Bellare *et al.* [BGLR93] (which is a simple generalization of the reduction by Lund and Yannakakis [LY94]) on the Boolean hypercube. That is, the instance of PHC-COVER we create is the same as the instance of SET-COVER created in the reduction of Bellare *et al.* [BGLR93]. Therefore the analysis we give follows directly from the analysis of Lund and Yannakakis [LY94].

Following the definition by Bellare *et al.* [BGLR93] we distinguish five important parameters of one-round multi-prover proof systems and define the class $\mathrm{MIP}_1(\ell_r, p, \ell_a, \ell_q, \epsilon)$ as follows:

Definition 4.4.6 *$L \in MIP_1(\ell_r(n), p(n), \ell_a(n), \ell_q(n), \epsilon(n))$ if there exists a probabilistic polynomial-time verifier V, communicating with $p(n)$ provers such that for every $x \in \{0,1\}^n$ the verifier:*

- *tosses $\ell_r(n)$ random coins obtaining $r \in \{0,1\}^{\ell_r}$,*
- *computes $p(n)$ questions $q(r)_1, \ldots, q(r)_{p(n)}$ each of length at most $\ell_q(n)$,*
- *for each i, asks the i-th prover question $q(r)_i$ and gets $p(n)$ answers $a_1, \ldots, a_{p(n)}$ each of length at most $\ell_a(n)$,*
- *computes a predicate $V(x, r, a_1, \ldots, a_{p(n)})$ and accepts if and only if it is 1,*
- *has perfect completeness: if $x \in L$ then $\exists \bar{P} = P_1, \ldots, P_{p(n)}$ such that $\mathbf{Pr}_r[V\, accepts\ when\ interacting\ with\ \bar{P}] = 1$;*
- *has soundness error at most $\epsilon(n)$: if $x \notin L$ then $\forall \bar{P} = P_1, \ldots, P_{p(n)}$, $\mathbf{Pr}_r[V\, accepts\ when\ interacting\ with\ \bar{P}] \leq \epsilon(n)$.*

Our reduction will rely on three simple properties of V. The *functionality* property requires that for each $x \in \{0,1\}^n$ and each $a_1 \in \{0,1\}^{\ell_a}$ there is at most one vector $(a_2, a_3, \ldots, a_p)$ such that $V(x, r, a_1, a_2, \ldots, a_p) = 1$ for some $r \in \{0,1\}^{\ell_r}$. The second property, *uniformity*, requires that for each $i \in [p]$, queries of V to prover i are uniformly distributed over the set Q_i of all the possible queries to prover i. The last, *equality of question space sizes*, requires that $|Q_1| = |Q_2| = \cdots = |Q_p|$. Following Bellare *et al.* [BGLR93] we call V *canonical* if it has these three properties.

Similarly, we distinguish analogous parameters for a PCP system. We denote the class PCP$(\ell_r(n), p(n), \ell_a(n), \ell_q(n), \epsilon(n))$ to be the class of languages decidable by a PCP verifier V that uses $\ell_r(n)$ random bits, generates $p(n)$ questions of length $\ell_r(n)$, gets answers of length $\ell_a(n)$, has perfect completeness and soundness error $\epsilon(n)$.

Packing a Proof System into the Boolean Hypercube

The main tool for creating an approximation gap is a set system

$$\mathcal{B}_{m,l} = (B; C_1, C_2, \ldots, C_m)$$

where m, l are positive integers and for each $i \in [m]$, $C_i \subseteq B$. This set system has the property that if $I \subset [m]$ and $|I| \leq l$, then no union $\bigcup_{i \in I} D_i$ covers B, where D_i equals C_i or its complement.

Lemma 4.4.7 ([LY94]) *There exists $\mathcal{B}_{m,l} = (B; C_1, C_2, \ldots, C_m)$ for $|B| = O(2^{2l}m^2)$ and it can be constructed in time polynomial in $|B|$.*

The main construction of this section is given in the following lemma.

Lemma 4.4.8 *If $L \in MIP_1(\ell_r, p, \ell_a, \ell_q, \epsilon)$ with a canonical verifier V, then there exists an algorithm $\mathcal{A}$ that given x, produces an instance of PHC-COVER $(S_x, \mathcal{T}_x)$ over $d \leq \ell_r + p(\ell_q + 2^{\ell_a})$ variables such that*

- *if $x \in L$ then PHC-COVER$(S_x, \mathcal{T}_x) = p|Q_1|$, where Q_1 is the question space of the first prover.*
- *if $x \notin L$ then PHC-COVER$(S_x, \mathcal{T}_x) \geq \frac{1}{2}(2\epsilon)^{-1/p}|Q_1|$.*

Moreover, $\mathcal{A}$ runs in time polynomial in n and 2^d.

Proof: We start by describing a way to map an answer from a prover to a subset. In this mapping the first prover is treated differently from the rest because of the functionality property of V. As before, let $Q_i \subseteq \{0,1\}^{\ell_q}$ denote the set of questions that V asks prover i and let A_i be the answer space of prover i. Set $s_a = |A_2| + |A_3| + \ldots + |A_p|$ (note that $|A_1|$

is not included) and let $\mathcal{B}_{s_a,l} = (B; C_1, C_2, \ldots, C_{s_a})$ be a set system given by Lemma 4.4.7 for l to be specified later. We index the sets $C_1, C_2, \ldots, C_{s_a}$ by pairs (i, a_i) for $2 \le i \le p$ and $a_i \in A_i$. Let $A = \{(i, a_i) \mid i \in [p],\ a_i \in A_i\}$ be the set of all possible answers (answers from different provers correspond to different elements).

Now for each setting of a random string $r \in R = \{0,1\}^{\ell_r}$ and $(i, a_i) \in A$, we define a subset $C(r, i, a_i) \subseteq B$ as follows.

$$C(r,i,a_i) = \begin{cases} C_{i,a_i} & \text{if } i \ge 2 \\ B \setminus (C_{2,a_2} \cup \cdots \cup C_{p,a_p}) & \text{if } i = 1 \text{ and} \\ & \exists a_2, \ldots, a_p,\ V(x, r, a_1, a_2, \ldots, a_p) = 1 \\ \emptyset & \text{otherwise} \end{cases}$$

For $a_1 \in A_1$, $C(r, 1, a_1)$ is well-defined since V has the functionality property. The definition of $C(r, i, a_i)$ implies that if $V(x, r, a_1, a_2, \ldots, a_p) = 1$ then $\cup_{i \in [p]} C(r, i, a_i) = B$, that is, answers from provers that cause the verifier to accept correspond to "small" covers. Bellare *et al.* [BGLR93] define the following instance of SET-COVER. The ground set equals to $R \times B$ and for every $i \in [p],\ q_i \in Q_i,\ a_i \in A_i$ the set system includes a subset

$$Z(i, q_i, a_i) = \{(r, b) \mid q_i = q(r)_i \text{ and } b \in C(r, i, a_i)\}\ ,$$

where $q_i = q(r)_i$ means that V generates query q_i to prover i on input x and random string r. In other words, the set system is $\mathcal{Z}_x = \{Z(i, q_i, a_i) \mid i \in [p],\ q_i \in Q_i,\ a_i \in A_i\}$.

We now show that exactly the same set system can be created on a hypercube of dimension $d = \ell_r + p\ell_q + |A|$. We refer to the first ℓ_r variables of the Boolean cube $\{0,1\}^d$ as $y_{r,1}, \ldots, y_{r,\ell_r}$, the next $p\ell_q$ variables as $z_{i,j}$ for $i \in [p]$ and $j \in [\ell_q]$, and the last $|A|$ variables as $z_{(i,a_i)}$ for $(i, a_i) \in A$.

For every $r \in R$ and $b \in B$, let $z(r, b) \in \{0,1\}^A$ be a Boolean vector of length $|A|$ such that $z(r,b)_{(i,a_i)} = 1$ whenever $b \in C(r, i, a_i)$. Furthermore, let $[r, b] = r \cdot q(r)_1 \cdots q(r)_p \cdot z(r, b)$. Let $S_x = \{[r, b] \mid r \in R,\ b \in B\}$. We now proceed to define the terms. For $i \in [p]$, $q_i \in Q_i$, and $a_i \in A_i$, let $T(i, q_i, a_i)$ be the term that checks that variables of i-th question equal to q_i and that the variable corresponding to answer a_i from prover i is set to 1, or formally

$$T(i, q_i, a_i) = \mathtt{eq}(z_{i,1} \cdots z_{i,\ell_q}, q_i) \wedge z_{(i,a_i)}\ .$$

Let $\mathcal{T}_x = \{T(i, q_i, a_i) \mid i \in [p],\ q_i \in Q_i,\ a_i \in A_i\}$. It is easy to verify that the term $T(i, q_i, a_i)$ covers a point $[r, b]$ if and only if $q_i = q(r)_i$ and $b \in C(r, i, a_i)$. Therefore the set system $(S_x, \mathcal{T}_x)$ corresponds exactly to the SET-COVER instance of Bellare *et al.* [BGLR93], where $[r, b]$ corresponds to (r, b) and $T(i, q_i, a_i)$ corresponds to $Z(i, q_i, a_i)$.

The analysis of Lund and Yannakakis [LY94] can now be used to prove that for $x \in L$, PHC-COVER$(S_x, \mathcal{T}_x) \leq \sum_{i \in P} |Q_i| = p|Q_1|$ and for $x \notin L$, PHC-COVER$(S_x, \mathcal{T}_x) \geq (1 - \epsilon l^p) l \cdot |Q_1|$. Therefore by setting $l = (2\epsilon)^{-1/p}$ we will get the stated inapproximability gap of $(2\epsilon)^{-1/p}/(2p)$. For completeness we provide the details of this analysis using the notation of the above SET-COVER instance.

First, let $x \in L$ and let $\bar{P}$ be an honest deterministic prover. For $q_i \in Q_i$ denote by $P_i(q_i)$ the answer given by P_i to query q_i and set $\mathcal{S} = \{Z(i, q_i, P_i(q_i)) \mid i \in [p],\ q_i \in Q_i\}$. For every point (r, b) and $i \in [p]$, let $a'_i = P_i(q(r)_i)$. By perfect completeness of V, $V(x, r, \bar{a}') = 1$ and therefore $C(r, 1, a'_1) = B \setminus (C_{2,a'_2} \cup \cdots \cup C_{p,a'_p})$, i.e, $\cup_{i \in [p]} C(r, i, a'_i) = B$. This means that for some j, $b \in C(r, j, a'_j)$ and therefore $(r, b) \in Z(j, q(r)_j, a'_j)$. This means that $\mathcal{S} \subseteq \mathcal{Z}$ is a collection of size $\sum_{i \in [p]} |Q_i|$ that covers $R \times B$. By equality of question space sizes, $|\mathcal{S}| = \sum_{i \in [p]} |Q_i| = p|Q_1|$.

Let $x \notin L$ and $\mathcal{S} \subseteq \mathcal{Z}_x$ be a cover for $R \times B$. For a random string r and a set $C \subseteq B$, denote by (r, C) the set $\{(r, b) \mid b \in C\}$ and let $\mathcal{S}_r = \{Z(i, q(r)_i, a_i) \mid i \in [p],\ Z(i, q(r)_i, a_i) \in \mathcal{S}\}$, in other words $\mathcal{S}_r$ includes the terms from $\mathcal{S}$ that cover (r, B). We say that r is *good* if $|\mathcal{S}_r| \leq l$ and *bad* otherwise. Let δ be the fraction of good r's.

Claim 4.4.9 *There exists a prover $\bar{P}$ such that V will accept with probability δ/l^p.*

Proof: We define $\bar{P}$ with the following strategy: prover P_i on query q_i chooses a_i from the set $A^i_{q_i} = \{a \mid Z(i, q_i, a) \in \mathcal{S}\}$ randomly and uniformly (this set cannot be empty). Note that for every r, $\sum_i |A^i_{q(r)_i}| = |\mathcal{S}_r|$. If r is good, then $|\mathcal{S}_r| \leq l$ and hence there should exist $a'_1, \ldots, a'_p$ such that for every $i \leq p$, $a'_i \in A^i_{q(r)_i}$ and $V(x, r, a'_1, \ldots, a'_p) = 1$. To prove this, assume that for every $a_1 \in A^1_{q(r)_1}$, there exists $j(a_1)$ such that $V(x, r, a_1, \ldots, a_p) = 1$ but $a_{j(a_1)} \notin A^{j(a_1)}_{q(r)_{j(a_1)}}$. Then

$$Z(1, q(r)_1, a_1) \cap (r, B) = (r, C(1, \tau(1, a_1)) = (r, B \setminus \bigcup_{i \geq 2} C_{i,a_i}) \subseteq (r, \overline{C_{j(a_1), a_{j(a_1)}}}) \ .$$

This implies that

$$\left(\bigcup_{a_1 \in A^1_{q(r)_1}} \overline{C_{j(a_1), a_{j(a_1)}}} \right) \bigcup \left(\bigcup_{i \geq 2,\ a_i \in A^i_{q_i}} C_{i,a_i} \right) = B \ .$$

We obtained a union of at most l sets from $\mathcal{B}_{s_a,l}$ that does not include a set and its complement, and covers B. This contradicts the definition of $\mathcal{B}_{s_a,l}$, proving the existence of $a'_1, \ldots, a'_p$ as above. For good r's and each i, $|A^i_{q(r)_i}| \leq l$ and therefore, the probability that each P_i will answer with a'_i is at least l^{-p}. Hence this strategy has success probability at least δ/l^p. $\square$ (Claim 4.4.9)

Claim 4.4.10 $|\mathcal{S}| \geq (1-\delta)l|Q_1|$.

Proof: For each bad r, $|\mathcal{S}_r| \geq l$ and therefore $\sum_r |\mathcal{S}_r| = (1-\delta)2^{\ell_r}l$. On the other hand, each subset $Z(i, q_i, a_i) \in \mathcal{S}$ appears once in all the sets for which $q_i = q(r)_i$. The uniformity property of V implies that each query q_i is asked for exactly $2^{\ell_r}/|Q_i| = 2^{\ell_r}/|Q_1|$ different r's (the last equality follows from the equality of question space sizes property). This implies that

$$|\mathcal{S}| = \frac{|Q_i|}{2^{\ell_r}} \sum_r |\mathcal{S}_r| \geq \frac{|Q_i|}{2^{\ell_r}}(1-\delta)2^{\ell_r}l = (1-\delta)l|Q_1| \ .$$

□ (Claim 4.4.10)

By Claim 4.4.9 and soundness of V, we get that $\delta \leq \epsilon \cdot l^p$. By Claim 4.4.10, this implies that $|\mathcal{S}| \geq (1-\epsilon l^p)l|Q_1|$. □ (Lemma 4.4.8)

Obtaining Proof Systems with Canonical Verifiers

In this section, we show how to derive canonical multi-prover proofs systems from general PCPs for NP. The first step is obtaining a multi-prover system from a PCP. As shown by Bellare, Goldreich, and Safra, (their proof appears in Ta-Shma's paper [TS96]) the identity transformation of a PCP to an MIP (that is, just distributing p queries to p different provers) increases the soundness error of the proof system by a factor of at most p^p. That is,

Lemma 4.4.11 ([TS96])

$$PCP(\ell_r(n), p(n), \ell_a(n), \ell_q(n), \epsilon(n)) \subseteq MIP_1(\ell_r(n), p(n), \ell_a(n), \ell_q(n), p^p\epsilon(n)) \ .$$

The next step in our transformation is obtaining the functionality property.

Lemma 4.4.12 *If $L \in MIP_1(\ell_r, p, \ell_a, \ell_q, \epsilon)$ with a verifier V, then*

$$L \in MIP_1(\ell_r, p+1, p\ell_a, p\ell_q, \epsilon)$$

with a verifier V' that has the functionality property.

Proof: To get a verifier V' with the desired property, we add one more prover (which we place first in the enumeration). Given r, V' uses V to generate questions $q_1, \dots, q_p$, asks all the "old" provers their respective questions, and asks the new prover question $(q_1, q_2, \dots, q_p)$. Given answers $a_1, \dots, a_p$ from the "old" provers and an answer $(a'_1, \dots, a'_p)$ from the new prover, V' accepts if $a'_i = a_i$ for all $i \in [p]$, and $V(x, r, a_1, \dots, a_p) = 1$. We first observe that, by definition, V' has the functionality property. Next, we observe that V' interacts with the original p provers exactly as V does and accepts only when V does.

Therefore the soundness error of the new multi-prover system does not increase and, in particular, is at most ϵ. Perfect completeness is preserved since if the first prover answers his questions in the same way as the other p honest deterministic provers, then V' will accept whenever V accepts. Finally, the bounds on the length of queries and answers grow by a factor of at most p. □

Next we describe how to obtain the last two properties required to get a canonical verifier.

Lemma 4.4.13 *If $L \in MIP_1(\ell_r, p, \ell_a, \ell_q, \epsilon)$ with a verifier V, then*

$$L \in MIP_1((p+1)\ell_r, p, \ell_a, \ell_r + \ell_q, \epsilon)$$

with a verifier V' that has uniformity and "equality of answer space sizes" properties. Furthermore, if V has the functionality property then V' is canonical.

Proof: For each $q_i \in Q_i$, let R_{i,q_i} denote the set of random strings for which V generates question q_i for prover i. New verifier V' uses V to generate questions $q_1, q_2, \ldots, q_p$ and then asks questions $((q_1, j_1), (q_2, j_2), \ldots, (q_p, j_p))$ where j_i is an element of $[|R_{i,q_i}|]$ chosen randomly, uniformly, and independently of other choices. It is easy to see that after this modification the sets of possible questions are all of the same size 2^{ℓ_r} and the questions are distributed uniformly. These random bits can be disregarded by honest provers and therefore completeness is not changed. Clearly, randomly and independently chosen bits cannot help dishonest provers and therefore soundness error is still bounded by ϵ. Finally, the bound on questions size is at most $\ell_r + \ell_q$ and the number of random bits required is at most $(p+1)\ell_r$. The accepting predicate of V was not changed and thus functionality property is preserved in this transformation. □

We can now combine these transformations with the following theorem due to Raz and Safra [RS97],

Theorem 4.4.14 *For any $\beta \leq 1/4$, $\ell_q(n) \leq \log^{\beta} n$ there exist fixed positive constants $b_r, b_p, b_q, b_\epsilon$ such that $\mathsf{SAT} \in PCP(b_r \log n, b_p, \ell_a(n), b_q \log n, 2^{-b_\epsilon \ell_a(n)})$.*

obtaining the following result:

Lemma 4.4.15 *There exist fixed positive constants $c_r, c_p, c_q, c_\epsilon$ for which*

$$\mathsf{SAT} \in MIP_1(c_r \log n, c_p, \log\log n, c_q \log n, \log^{-c_\epsilon} n)$$

with a canonical verifier.

Proof: We start with the PCP from Theorem 4.4.14 and then apply Lemmas 4.4.11, 4.4.12, and 4.4.13 to get

$$\mathsf{SAT} \in \mathrm{MIP}_1((b_p+2)b_r \log n, b_p+1, b_p\ell_a(n), (b_pb_q+b_r)\log n, b_p^{b_p}2^{-b_\epsilon \ell_q(n)}) \ .$$

We now choose $\ell_a(n) = (\log\log n)/b_p$ and obtain the desired result for $c_r = (b_p+2)b_r$, $c_p = b_p+1$, $c_q = (b_pb_q+b_r)$, and any $c_\epsilon > b_\epsilon/b_p$ ("strictly greater" is to offset the constant factor $b_p^{b_p}$). □

We can now use the results from Sections 4.3 and 4.4.2 to summarize our inapproximabilily results for covering problems on the Boolean hypercube.

Theorem 4.4.16 (subsumes Th. 4.1.1) *There exists a constant $\gamma > 0$ such that, unless $\mathsf{P} = \mathsf{NP}$, there is no polynomial-time algorithm approximating TT-MinDNF, PTT-MinDNF, HC-COVER, and PHC-COVER, to within a factor $d^\gamma = \Omega(\log^\gamma(N))$, where d is the number of variables and N is the size of an instance.*

By using our reduction from MIP to PHC-COVER(Lemma 4.4.8) with the canonical-verifier MIP obtained from the PCP of Raz and Safra [RS97] (Lemma 4.4.15), we get an inapproximability gap of $(2\log n)^{c_\epsilon/c_p}/(2c_p)$ for $d \leq (c_r + c_pc_q + c_p)\log n$. This implies the claim for PHC-COVER. We use Theorem 4.3.6 to extend the result to TT-MinDNF, PTT-MinDNF and HC-COVER.

4.5 Hardness of Proper PAC+MQ Learning over the Uniform Distribution

We now show a simple application of the hardness of TT-MinDNF to the hardness of proper PAC learning of DNF expressions restricted to the uniform distribution over $\{0,1\}^n$. It is a very strong model in which, as proved by Jackson [Jac97], DNF expressions are learnable non-properly.

It has been observed by Allender *et al.* that TT-MinDNF naturally reduces to exact learning of DNF with MQs [AHPM04]. We further this observation by reducing TT-MinDNF to PAC+MQ learning of DNF over the uniform distribution. We denote the uniform distribution over $\{0,1\}^n$ by U.

Theorem 4.5.1 *There exists a constant $\gamma > 0$ such that, if there exists an algorithm $\mathcal{A}$ that for every Boolean function c and $\epsilon > 0$, $\mathcal{A}$, given access to EX(c,U) and MEM(c), runs in time poly$(n, s = \texttt{DNF-size}(c), 1/\epsilon)$ and, with probability at least $3/4$, outputs a DNF formula h of size at most $\log^\gamma(s/\epsilon)\cdot s$ that ϵ-approximates c with respect to U, then $\mathsf{NP} = \mathsf{RP}$.*

Proof: We reduce from TT-MinDNF and let γ be the constant from Theorem 4.1.1. Given the truth table of a function f over $d = \log n$ variables, we let the target concept be $c(x) = f(x_1 \cdots x_d)$. Clearly, $s = \texttt{DNF-size}(c) \leq 2^{\log n} = n$. The definition of $c(x)$ implies that $\mathrm{EX}(c, U)$ and $\mathrm{MEM}(c)$ can be efficiently simulated given the truth table of f. We then set $\epsilon = 1/(2n)$ and $\delta = 1/2$. A strongly proper algorithm on this input will (with probability at least 1/2) produce in time polynomial in $n = 2^d$ a DNF formula h of size $t \leq \log^{\gamma}(s/\epsilon) \cdot s$ that $\frac{1}{2n}$-approximates c. Now we choose a vector y of length $n - d$ randomly and uniformly and let h_y be the projection of h to first d variables with the last $n - d$ variables set to y. We claim that with probability at least 1/2, $f \equiv h_y$. To see this, note that

$$\frac{1}{2n} \geq \mathbf{Pr}_{x \in \{0,1\}^n}[c(x) \neq h(x)] = \mathbf{E}_{y \in \{0,1\}^{n-d}}\left[\mathbf{Pr}_{z \in \{0,1\}^d}[f(z) \neq h_y(z)]\right] ,$$

and thus for at least $\frac{1}{2}$ of y's, $\mathbf{Pr}_{z \in \{0,1\}^d}[f(z) \neq h_y(z)] = 0$, that is, $f(z) = h_y(z)$ for all z. For each y, the number of terms in h_y is at most t and therefore with probability at least 1/4, t approximates $\texttt{DNF-size}(f)$ within $\log^{\gamma}(s/\epsilon) = O(d^{\gamma})$. $\square$

Chapter 5

Attribute-Efficient and Non-adaptive Learning of DNF Expressions and Parities

In this chapter we study the problems of attribute-efficient PAC learning of parity functions and DNF expressions over $\{0,1\}^n$. We show that attribute-efficient learning of parities with respect to the uniform distribution is equivalent to decoding high-rate random linear codes from a low number of errors, a long-standing open problem in coding theory.

An algorithm is said to use membership queries (MQs) *non-adaptively* if the points at which the algorithm asks MQs do not depend on the target concept. We give the first non-adaptive and attribute-efficient algorithm for learning DNF with respect to the uniform distribution. Our algorithm runs in time $\tilde{O}(ns^4/\epsilon)$ and uses $\tilde{O}(s^4 \cdot \log^2 n/\epsilon)$ non-adaptive MQs where s is the number of terms in the shortest DNF representation of the target concept. The algorithm improves on the best previous algorithm for learning DNF (of Bshouty *et al.* [BJT04]) and can also be easily modified to tolerate random persistent classification noise in MQs.

5.1 Introduction

The problems of PAC learning parity functions and DNF expressions are among the most fundamental and well-studied problems in machine learning theory. Along with running time efficiency, an important consideration in the design of learning algorithms is their *attribute-efficiency*. A class $\mathcal{C}$ of Boolean functions is said to be *attribute-efficiently learnable* if there is an efficient algorithm which can learn any function $f \in \mathcal{C}$ using a number of examples which is polynomial in the "size" (description length) of the function f to be learned, rather than in n, the number of attributes in the domain over which learning takes place. Attribute-efficiency arises naturally from a ubiquitous practical scenario in which the total number of potentially influential attributes is much larger than the number of relevant attributes (i.e., the attributes on which the concept actually depends), whereas examples are either scarce or expensive to get.

Learning of DNF expressions and attribute-efficient learning of parities from random examples with respect to the uniform distribution are both long-standing challenges in learning theory. The lack of substantial progress on these questions has resulted in attempts to solve them in stronger learning models. The most well-studied such model is one in which a *membership query oracle* is given to the learner in addition to the example oracle. The learning algorithm may query this oracle for a value of the target function at any point of its choice. Jackson gave the first algorithm that learns DNF from membership queries (MQs) under the uniform distribution [Jac97] and later Bshouty, Jackson and Tamon gave a more efficient and attribute-efficient algorithm for learning DNF in the same setting [BJT99]. The first algorithm for attribute-efficient learning of parities using MQs is due to Blum, Hellerstein and Littlestone [BHL95], and their result was later refined by Uehara *et al.* [UTW97].

A restricted model of membership queries, which addresses some of the disadvantages of the MQ model, is the model in which MQs are asked non-adaptively. An algorithm is said to use MQs *non-adaptively* if the queries of the algorithm do not depend on the target concept (in our context we will often call it non-adaptive for brevity). In other words, the learning algorithm can be split into two stages. In the first stage, given the learning parameters, the algorithm generates a set S of queries for the membership oracle. In the second stage, given the answers to the queries in S, the algorithm produces a hypothesis (without further access to the oracle). An immediate advantage of this model (over the usual MQ model) is the fact that the queries to the membership oracle can be parallelized. This, for example, is crucial in DNA sequencing and other biological applications where tests are very time-consuming but can be parallelized (*cf.* [FKKM97, Dam98] and references therein). Another advantage of a non-adaptive learner is that the same set of points can be used to learn numerous concepts. This is conjectured to happen in the human brain where a single example can be used to learn several different concepts and hence systems that aim to reproduce the learning abilities of the human brain need to possess this property [Val94, Val00, Val06].

As it is detailed later, attribute-efficiency is easy to achieve using a simple technique that relies on adaptive MQs but there is no known general method to convert a learning algorithm to an attribute-efficient one using MQs non-adaptively. It is important to note that in the two practical applications mentioned above, attribute-efficiency is also a major concern. It is therefore natural to ask: which classes can be PAC learned attribute-efficiently by non-adaptive MQs? We refer to this model of learning as *ae.naMQ learning*. This question was first explicitly addressed by Damaschke [Dam98] who proved that any function of r variables is ae.naMQ learnable when it is represented by the truth table of the function (requiring $r \log n + 2^r$ bits). Later Hofmeister gave the first ae.naMQ algorithm for learning parities [Hof99] and Guijarro *et al.* gave an algorithm for learning functions of at most $\log n$ variables in the decision tree representation [GLR99]. But the question remains open for numerous other representations used in learning theory.

5.1.1 Previous Results

Attribute-efficient Learning of Parities. Blum *et al.* were the first to ask whether parities are learnable attribute-efficiently (in the related *on-line mistake-bound* model) [BHL95]. They also presented the first algorithm to learn parity functions attribute-efficiently using MQs. Their algorithm is based on the following approach: first all the relevant attributes are identified and then a simple (not attribute-efficient) algorithm restricted to the relevant variables is used to learn the concept. Since then other algorithms were proposed for attribute-efficient identification of relevant variables [BH98, GTT99]. All the algorithms are based on a binary search for a relevant variable given a positive and a negative example. Binary search and the fact that queries in the second stage depend on the variables identified in the first stage only allows for the construction of adaptive algorithms via this approach. Uehara *et al.* gave several algorithms for attribute-efficient learning of parities that again used adaptiveness in an essential way [UTW97].

Hofmeister gave the first ae.naMQ algorithm for learning parities based on BCH error-correcting codes. When learning the class of parities on at most k variables his algorithm has running time of $O(kn)$ and uses $O(k \log n)$ non-adaptive MQs. While the complexity of this algorithm is asymptotically optimal it is based on the relatively complex Berlekamp-Massey algorithm for creating and decoding BCH codes [Mas69].

Little previous work has been published on attribute-efficient learning of parities from random examples only. Indeed, the first non-trivial result in this direction has only recently been given by Klivans and Servedio [KS04b]. They prove that parity functions on at most k variables are learnable in polynomial time using $O(n^{1-\frac{1}{k}} \log n)$ examples.

Learning DNF. Efficient learning of unrestricted DNF formulae under the uniform distribution begins with a famous result by Jackson [Jac97]. The algorithm, while polynomial-time, is somewhat impractical due to the $\tilde{O}(ns^{10}/\epsilon^{12})$ bound on running time (where s is the number of terms in the target DNF). By substantially improving the key components of Jackson's algorithm, the works of Freund [Fre92], Bshouty *et al.* [BJT99], and Klivans and Servedio [KS03] resulted in an algorithm that learns DNF in time $\tilde{O}(ns^6/\epsilon^2)$ and uses

$\tilde{O}(ns^4/\epsilon^2)$ MQs[1]. This algorithm is non-adaptive, but is also not attribute-efficient. Using the algorithm for identification of relevant variables by Bshouty and Hellerstein mentioned above, Bshouty *et al.* gave an attribute-efficient version of their algorithm running in time $\tilde{O}(rs^6/\epsilon^2+n/\epsilon)$ and using $\tilde{O}(rs^4 \log n/\epsilon^2)$ adaptive MQs, where r is the number of relevant variables [BJT99].

Bshouty *et al.* give an algorithm for learning DNF expressions from examples generated by a random walk on the Boolean hypercube [BMOS03]. This model is more passive than non-adaptive MQs but the algorithm of Bshouty *et al.* is not attribute-efficient as it is an adaptation of the non-attribute-efficient algorithm Bshouty and Feldman [BF02]. In fact, it is information-theoretically impossible to learn anything non-trivial attribute-efficiently in this model.

5.1.2 Our Results

We give a simple and fast randomized algorithm for ae.naMQ learning of parities (Theorem 5.4.1) and provide a transformation that converts a non-adaptive parity learning algorithm into an algorithm for finding significant Fourier coefficients of a function while preserving attribute-efficiency and non-adaptiveness (Theorem 5.5.4). Using these components we give the first ae.naMQ algorithm for learning DNF expressions with respect to the uniform distribution (Theorem 5.6.8). It runs in time $\tilde{O}(ns^4/\epsilon)$ and uses $\tilde{O}(s^4 \log^2 n/\epsilon)$ MQs. The algorithm improves on the $\tilde{O}(ns^6/\epsilon^2)$-time and $\tilde{O}(ns^4/\epsilon^2)$-query algorithm of Bshouty *et al.* [BJT99]. In Theorem 5.7.3 we also show a simple and general modification that allows the above algorithm to efficiently handle random persistent classification noise in MQs (see Section 2.2.3 for the formal definition of the noise model). Earlier algorithms for learning DNFs that handled persistent classification noise were based on Jackson's DNF learning algorithm and therefore are substantially less efficient [JSS97, BF02].

Alongside our ae.naMQ algorithm for learning of parities we establish the equivalence between attribute-efficient learning of parities from random uniform examples and decoding high-rate random linear codes from a low number of errors, a long-standing open problem in coding theory widely believed to be intractable (Theorems 5.3.4 and 5.3.6). Thus we may consider this equivalence as evidence of the hardness of attribute-efficient learning of parities from random examples only. Previously hardness of attribute-efficient learning results were only known for specially designed concept classes [DGR99, Ser00].

The connection between attribute-efficient learning of parities by membership queries and linear codes was earlier observed by Hofmeister [Hof99]. His result allows to derive attribute-efficient parity learning algorithms from efficiently decodable linear codes with

[1]Bshouty *et al.* claimed sample complexity $\tilde{O}(ns^2/\epsilon^2)$ but this was in error as explained in Remark 5.6.3.

appropriate parameters. Our result can be seen as an adaptation of this connection to random and uniform examples. The restriction to the uniform distribution allows us to prove the connection in the other direction, giving the above-mentioned negative result for attribute-efficient learning of parities from random examples only.

5.1.3 Organization

In the next section we describe the models and tools that will be used in this work. In Section 5.3, we give the required background on binary linear codes and prove the equivalence between attribute-efficient learning of parities from random uniform examples and decoding high-rate random linear codes from a low number of errors. In Section 5.4, we show a simple algorithm for ae.naMQ learning of parities. Section 5.5 gives a way to convert a non-adaptive parity learning algorithm into an algorithm for finding significant Fourier coefficients of a function while preserving attribute-efficiency and non-adaptiveness, yielding an ae.naMQ algorithm for weakly learning DNF expressions. Then in Section 5.6 we describe our ae.naMQ algorithm for learning DNF expressions and in Section 5.7 we show how this algorithm can be modified to handle random persistent classification noise.

5.2 Preliminaries

For vectors $x, y \in \{0,1\}^n$ we denote by $x \oplus y$ the vector obtained by bitwise XOR of x and y; by $[k]$ the set $\{1, 2, \ldots, k\}$; by e_i a vector with 1 in i-th position and zeros in the rest; by x_i the i-th element of vector x. Dot product $x \cdot y$ of vectors $x, y \in \{0,1\}^n$ denotes $\sum_i x_i y_i \pmod 2$ or simply vector product xy^T over $\mathbf{GF}(2)$ (with vectors being row vectors by default). By $\texttt{weight}(x)$ we denote the Hamming weight of x and we define $\texttt{dist}(x, y) = \texttt{weight}(x \oplus y)$.

5.2.1 PAC Learning

We study learning of Boolean functions on the Boolean hypercube $\{0,1\}^n$. Our Boolean functions take values $+1$ (true) and -1 (false). Our main interest are the classes of parity functions and DNF expressions. Recall that a parity function $\chi_a(x)$ for a vector $a \in \{0,1\}^n$ is defined as $\chi_a(x) = (-1)^{a \cdot x}$. We refer to the vector associated with a parity function as its *index* and the Hamming weight of the vector as the length of the parity function. We denote the concept class of parity functions $\{\chi_a \mid a \in \{0,1\}^n\}$ by PAR and the class of all the parities of length at most k by PAR(k). We represent a parity function by listing all the variables on which it depends. This representation for a parity of length k requires $\theta(k \log n)$ bits.

For the standard DNF representation and any Boolean function f we denote by DNF-size(f) the number of terms in a DNF representation of f with the minimal number of terms. In context of learning DNF this parameter is always denoted s. Our learning model is Valiant's well-known PAC model [Val84] defined in Section 2.2. We are mostly interested in learning with respect to the uniform distribution, denoted by $\mathcal{U}$. When learning with respect to $\mathcal{U}$, EX(c,$\mathcal{U}$) can be trivially simulated using MEM(c) and therefore EX(c,$\mathcal{U}$) is not used at all. We also briefly review the concepts of weak learning, attribute efficiency and classification noise defined in Sections 2.2.2 and 2.2.3. We say that an algorithm *weakly* learns $\mathcal{F}$ if it produces a hypothesis h that $(\frac{1}{2} - \frac{1}{p(n,s)})$-approximates (or *weakly approximates*) c for some polynomial p. An algorithm $\mathcal{A}$ is said to be *attribute-efficient* if the number of examples (both random and received from the MQ oracle) it uses is polynomial in the size of the representation of the target concept and $1/\epsilon$. We say that a variable x_i is *relevant* for a function f if there exists $y \in \{0,1\}^n$ such that $f(y) \neq f(y \oplus e_i)$. The number of relevant variables of the target concept is denoted by parameter r.

We consider two standard models of noise in learning. The first one is the well-studied random classification noise model introduced by Angluin and Laird [AL88]. In this model for any $\eta \leq 1/2$ called the *noise rate* the regular example oracle EX($c, \mathcal{D}$) is replaced with the faulty oracle EX$^{\eta}$($c, \mathcal{D}$). On each call, EX$^{\eta}$($c, \mathcal{D}$), draws x according to $\mathcal{D}$, and returns $\langle x, c(x) \rangle$ with probability η and $\langle x, \neg c(x) \rangle$ with probability $1 - \eta$. When η approaches $1/2$ the result of the corrupted query approaches the result of the random coin flip, and therefore the running time of algorithms in this model is allowed to polynomially depend on $\frac{1}{1-2\eta}$.

This model of noise is not suitable for corrupting labels returned by MEM(c) since a learning algorithm can, with high probability, find the correct label at point x by asking the label of x polynomial (in $\frac{1}{1-2\eta}$) number of times and then returning the label that appeared in the majority of answers. An appropriate modification of the noise model is the introduction of *random persistent classification noise* by Goldman, Kearns and Shapire [GKS93]. In this model, as before, the answer to a query at each point x is flipped with probability $1 - \eta$. However, if the membership oracle was already queried about the value of f at some specific point x or x was already generated as a random example, the returned label has the same value as in the first occurrence (i.e., in such a case the noise persists and is not purely random). If the learner does not ask for the label of a point more than once then this noise can be treated as the usual independent random classification noise.

Fourier transform. The Fourier transform is a technique for learning with respect to the uniform distribution (primarily) based on the fact that the set of all parity functions

$\{\chi_a(x)\}_{a\in\{0,1\}^n}$ forms an orthonormal basis of the linear space of real-valued function over $\{0,1\}^n$. This fact implies that any real-valued function f over $\{0,1\}^n$ can be uniquely represented as a linear combination of parities, that is $f(x) = \sum_{a\in\{0,1\}^n} \hat{f}(a)\chi_a(x)$. The coefficient $\hat{f}(a)$ is called Fourier coefficient of f on a and equals $\mathbf{E}_{\mathcal{U}}[f(x)\chi_a(x)]$; a is called the *index* and $\texttt{weight}(a)$ the *degree* of $\hat{f}(a)$. Given the values of f on all the points of the hypercube $\{0,1\}^n$ one can compute the values of all the Fourier coefficients $\{\hat{f}(a)\}_{a\in\{0,1\}^n}$ using the Fast Fourier Transform (FFT) algorithm in time $O(n2^n)$ [CT65] (*cf.* [AHU74]). The same algorithm FFT also converts the set of all Fourier coefficients $\{\hat{f}(a)\}_{a\in\{0,1\}^n}$ into the values of the function f on all the points of the hypercube. This transformation is called inverse Fourier transform. For further details on the technique we refer the reader to the survey by Mansour [Man94].

Randomized functions. Besides deterministic functions on $\{0,1\}^n$ we will also deal with functions whose value on a point x is a real-valued random variable $\Psi(x)$ independent of $\Psi(y)$ for any $y \neq x$ and of any previous evaluations of $\Psi(x)$. To extend learning and Fourier definitions to this case we include the probability over the random variable Ψ in estimations of probability, expectation and variance. For example, we say that a randomized function Ψ ϵ-approximates f with respect to $\mathcal{D}$ if $\mathbf{Pr}_{\mathcal{D},\Psi}[f(x) = \Psi(x)] \geq 1-\epsilon$. Similarly, $\widehat{\Psi}(a) = \mathbf{E}_{\mathcal{U},\Psi}[\Psi(x)\chi_a(x)]$.

5.2.2 Learning by Non-adaptive Membership Queries

We say that an algorithm $\mathcal{A}$ uses MQs *non-adaptively* if it can be split into two stages. The first stage, given all the parameters of learning, (n, ϵ and a bound on the size of the target concept), generates a set of points $S \subseteq \{0,1\}^n$. The second stage, given the answers from MEM(c) on points in S, i.e. the set $\{(x, c(x)) \mid x \in S\}$, computes a hypothesis (or, in general, performs some computation). Neither of the stages has any other access to MEM(c). We note that in the general definition of PAC learning we did not assume that size of the target concept (or a bound on it) is given to the learning algorithm. When learning with adaptive queries a good bound can be found via the "guess-and-double" technique, but for non-adaptive algorithms we will assume that this bound is always given. To emphasize this we specify the parameters that have to be given to a non-adaptive algorithm in the name of the algorithm. Clearly the same "guess-and-double" technique can be used to produce a sequence of independent and non-adaptive executions of the learning algorithm.

The immediate consequence of non-adaptiveness is that in order to parallelize a non-adaptive learning algorithm only the usual computation has to be parallelized since all the MQs can be made in parallel. Non-adaptiveness is also useful when learning ℓ concepts from the same concept class in parallel. The fact that queries are independent of the target concept implies that same set of points can be used for learning different concepts. To achieve probability of success 1/2 in learning of all ℓ concepts we will have to learn with each concept with probability of success $1 - 1/(2\ell)$. This implies that the number of points needed for learning might grow by a factor of $\log \ell$ whereas in the general case ℓ times more examples might be required.

Results of Goldreich *et al.* imply that if one-way functions exist then the concept class of all polynomial circuits is not learnable even with respect to $\mathcal{U}$ and with access to a MQ oracle [GGM86, KV94a]. By modifying the values of each circuit to encode the circuit itself in a polynomial number of fixed points one can make this class learnable by non-adaptive MQs but not learnable from random and uniform examples only (the modification is very unlikely to be detected by random examples yet MQs to the fixed points will reveal the circuit). Similarly, by placing the encoding of the circuit in some location that is encoded in a fixed location, one can create a function class learnable by adaptive membership queries but not learnable by the non-adaptive ones (if one-way functions exist).

5.2.3 Learning by Non-adaptive Membership Queries

We say that an algorithm $\mathcal{A}$ uses MQs *non-adaptively* if it can be split into two stages. The first stage, given all the parameters of learning, (n, ϵ and a bound on the size of the target concept), generates a set of points $S \subseteq \{0,1\}^n$. The second stage, given the answers from MEM(c) on points in S, i.e. the set $\{(x, c(x)) \mid x \in S\}$, computes a hypothesis (or, in general, performs some computation). Neither of the stages has any other access to MEM(c). We note that in the general definition of PAC learning we did not assume that size of the target concept (or a bound on it) is given to the learning algorithm. When learning with adaptive queries a good bound can be found via the "guess-and-double" technique, but for non-adaptive algorithms we will assume that this bound is always given. To emphasize this we specify the parameters that have to be given to a non-adaptive algorithm in the name of the algorithm. Clearly the same "guess-and-double" technique can be used to produce a sequence of independent and non-adaptive executions of the learning algorithm.

The immediate consequence of non-adaptiveness is that in order to parallelize a non-adaptive learning algorithm only the usual computation has to be parallelized since all the MQs can be made in parallel. Non-adaptiveness is also useful when learning ℓ concepts from the same concept class in parallel. The fact that queries are independent of the target concept implies that same set of points can be used for learning different concepts. To achieve probability of success 1/2 in learning of all ℓ concepts we will have to learn with each concept with probability of success $1 - 1/(2\ell)$. This implies that the number of points needed for learning might grow by a factor of $\log \ell$ whereas in the general case ℓ

times more examples might be required.

Results of Goldreich *et al.* imply that if one-way functions exist then the concept class of all polynomial circuits is not learnable even with respect to $\mathcal{U}$ and with access to a MQ oracle [GGM86, KV94a]. By modifying the values of each circuit to encode the circuit itself in a polynomial number of fixed points one can make this class learnable by non-adaptive MQs but not learnable from random and uniform examples only (the modification is very unlikely to be detected by random examples yet MQs to the fixed points will reveal the circuit). Similarly, by placing the encoding of the circuit in some location that is encoded in a fixed location, one can create a function class learnable by adaptive membership queries but not learnable by the non-adaptive ones (if one-way functions exist). Further details of these simple separations are left to the reader.

5.3 Learning of Parities and Binary Linear Codes

In this section we show that attribute-efficient learning of parities with respect to the uniform distribution from random examples only is likely to be hard by proving that it is equivalent to an open problem in coding theory. Unlike in the rest of the paper in this section and the following section parity functions will be functions to $\{0,1\}$. To emphasize this we use $\dot{\chi}$ instead of χ.

5.3.1 Background on Linear Codes

We say that a code C is an $[m,n]$ code if C is a binary linear code of block length m and message length n. Any such code can be described by its $n \times m$ *generator matrix* G as follows: $C = \{xG \mid x \in \{0,1\}^n\}$. Equivalently, a code can be described by its *parity-check* matrix H of size $m \times (m-n)$ by $C = \{y \mid yH = 0^{m-n}\}$. It is well-known that $G{\cdot}H = 0^{n\times(m-n)}$ and decoding given a corrupted message y is equivalent to decoding given the *syndrome* of the corrupted message. The syndrome equals to yH and the decoding consists of finding a vector e of Hamming weight at most w such that $y \oplus e = xG$, where $w = \lfloor (d-1)/2 \rfloor$ and d is the *distance* of the code (*cf.* [vL98]). For a linear code C the distance equals to the Hamming weight of a non-zero vector with the smallest Hamming weight.

By saying that C is a *random* $[m,n]$ *code* we mean that C is defined by choosing randomly, uniformly, and independently n vectors in $\{0,1\}^m$ that form the basis of C. Alternatively, we can say that the generator matrix G of C was chosen randomly with each entry equal to 1 with probability 1/2 independently of others. We denote this distribution by $\mathcal{U}_{n\times m}$. Some authors restrict the random choice of G's to matrices of full rank n. As we will see, this definitions would only make our proofs simpler.

Binary linear codes generated randomly meet the Gilbert-Varshamov bound with high probability, that is, they achieve the best known *rate* (or n/m) versus distance trade-off (*cf.* [Sud02]). However decoding a random linear code or even determining its distance is a notorious open problem in coding theory. For example the McEliece cryptosystem is based, among other assumptions, on the hardness of this problem [McE78]. Besides that, while the average-case hardness of this problem is unknown, a number of worst-case problems related to decoding linear codes are NP-hard (*cf.* [Bar97, Var97, Sud02]).

A potentially simpler version of this problem in which the errors are assumed to be random and independent with some rate η (and not adversarial as in the usual definition) is equivalent to learning of parities with random classification noise of rate η, a long-standing open problem in learning theory. In fact, in Section 6.4.1 we prove that when learning parities from random and uniform examples, random classification noise of rate η is as hard as adversarial noise of rate η (up to a polynomial blowup in the running time). The only known non-trivial algorithm for learning parities with noise is a slightly subexponential algorithm by Blum *et al.* [BKW00]. In our discussion η is very low (e.g. $\frac{\log n}{n}$), yet even for this case no efficient noise-tolerant algorithms are known.

Correcting a random linear $[m, n]$ from up to w errors is defined as follows.

Definition 5.3.1 Input: *An $n \times m$ binary generator matrix G randomly chosen according to $\mathcal{U}_{n\times m}$ and $y \in \{0,1\}^m$.*
Output: *$x \in \{0,1\}^n$ such that $\mathtt{dist}(xG, y) \leq w$ if there exists one.*

A successful algorithm for this problem is an algorithm that would allow to correct up to w errors in a "good" fraction of randomly created linear codes. That is, with non-negligible probability over the choice of G, and for every y, the algorithm should produce the desired output. Note that the algorithm can only be successful when the code generated by G has distance at least $2w + 1$.

For simplicity, we will usually assume a constant probability of success but all the results can be translated to algorithms having the success probability lower-bounded by a polynomial (in m) fraction.

5.3.2 The Reduction

The equivalence of attribute-efficient learning of parities with respect to the uniform distribution and decoding of random linear codes relies on two simple lemmas. The first one, due to Hofmeister [Hof99], is that the syndrome decoding of a linear code implies attribute-efficient learning of parities. We include it with a proof for completeness.

Lemma 5.3.2 (Hofmeister) *Let H be a parity-check matrix of some $[m, n]$ w-error correcting code C. Let $\mathcal{A}$ be an algorithm that for any $y \in \{0,1\}^m$ such that $y = c \oplus e$ where*

$c \in C$ and $\texttt{weight}(e) \leq w$, given the syndrome yH, finds e. Then $\mathcal{A}$ learns $PAR(w)$ over $\{0,1\}^m$ given the values of an unknown parity on the columns of H.

Proof: The condition $y = c \oplus e$ for $c \in C$ implies that $yH = eH$. Therefore the syndrome yH is equal to the vector $eH = \dot{\chi}_e(H_1), \dot{\chi}_e(H_2), \ldots, \dot{\chi}_e(H_{m-n})$ where H_i is the i-th column of H. Therefore finding an error vector e of weight at most w using the syndrome yH is the same as finding a parity of length at most w given the values of the unknown parity on the columns of H. $\square$ This observation has lead Hofmeister to a simple ae.naMQ algorithm for learning parities that uses the columns of the parity check matrix of BCH code as MQs. We note that the converse of this lemma is only true if the learning algorithm is *proper*, that is, produces a parity function in PAR(w) as a hypothesis.

To obtain the claimed equivalence for the uniform distribution we first need to prove that generating a linear code by choosing a random and uniform parity check matrix (that is, from $\mathcal{U}_{n\times m-n}$) is equivalent to (or indistinguishable from) generating a linear code by choosing a random and uniform generator matrix (that is, from $\mathcal{U}_{n\times m}$).

Let $p(i,j)$ denote the probability that i vectors chosen randomly and uniformly from $\{0,1\}^j$ are linearly independent. Each $i \geq 1$ linearly independent vectors span subspace of size 2^i and therefore there are $2^j - 2^i$ vectors that are linearly independent of them. This implies that, $p(i+1,j) = p(i,j)(1-2^{-j+i})$. All vectors except for 0^j form a linearly independent set of size 1. Therefore $p(1,j) = (1-2^{-j})$. Hence

$$p(i,j) = (1-2^{-j}) \cdot (1-2^{-j+1}) \cdots (1-2^{-j+i-1}) \ .$$

Note that

$$p(i,j) \geq 1 - 2^{-j} - 2^{-j+1} - \cdots - 2^{-j+i-1} > 1 - 2^{-j+i} \tag{5.1}$$

and for $i = j$, $p(j,j) = \frac{1}{2}p(j,j-1) > \frac{1}{2}(1-\frac{1}{2}) = \frac{1}{4}$. This means that for any $i \leq j$, $p(i,j) > 1/4$.

Let $\mathcal{V}_{n\times m}$ denote the distribution on matrices of size $n \times m$ resulting from the following process. Choose randomly and uniformly a $m \times (m-n)$ matrix H of rank $m-n$ and then choose randomly and uniformly a matrix G of size $n \times m$ of rank n such that $GH = 0^{n\times(m-n)}$. To generate G's like this we find a basis $b_1, \ldots, b_n$ for the subspace of $\{0,1\}^m$ that is "orthogonal" to H in the standard (and efficient) way. Let G_0 denote the matrix whose rows are the vectors $b_1, \ldots, b_n$. It is easy to see that any matrix G of rank n such that $GH = 0^{n\times(m-n)}$, can be represented uniquely as $F \cdot G_0$ where F is a matrix of size $n \times n$ and full rank ($*$). Therefore we can generate G's as above by choosing randomly and uniformly a matrix F of rank n. If we choose a random matrix F according $\mathcal{U}_{n\times n}$, with probability at least $p(n,n) > 1/4$, it will have the full rank. We can repeatedly sample from $\mathcal{U}_{n\times n}$ to get a full-rank F with any desired probability. This implies that we can generate a matrix according to $\mathcal{V}_{n\times m}$ with probability $1-\delta$ in time $O(m^3 \log(1/\delta))$ (or less if a non-trivial matrix multiplication algorithm is used).

All we need to prove now is that $\mathcal{V}_{n\times m}$ is "close" to $\mathcal{U}_{n\times m}$. More specifically, the *statistical distance* between two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over X is defined to be $\Delta(\mathcal{D}_1, D_2) = \frac{1}{2}\sum_{x\in X} |\mathcal{D}_1(x) - \mathcal{D}_2(x)|$. It is well known and easy to see that for any event $E \subseteq X$, $|\mathbf{Pr}_{\mathcal{D}_1}[x\in E] - \mathbf{Pr}_{\mathcal{D}_2}[x \in E]| \leq \Delta(\mathcal{D}_1, D_2)$.

Lemma 5.3.3 *The distribution $\mathcal{V}_{n\times m}$ is uniform over matrices of size $n \times m$ and rank n. In particular, $\Delta(\mathcal{V}_{n\times m},\mathcal{U}_{n\times m}) \leq 2^{-m+n}$.*

Proof: Let G be any matrix of size $n\times m$ with linearly independent rows. Its probability under $\mathcal{U}_{n\times m}$ is $\mathcal{U}_{n\times m}(G) = 2^{-mn}$. When sampling with respect to $\mathcal{V}_{n\times m}$, G can be obtained only if all the columns of H are "orthogonal" to rows of G, that is belong to a linear subspace of $\{0,1\}^m$ of dimension $m-n$. The total number of H's like these of rank $m-n$ is $2^{(m-n)^2}p(m-n,m-n)$ (as follows from $(*)$) and the total number of matrices size $m\times(m-n)$ of rank $m-n$ is $2^{m(m-n)}p(m-n,m)$. Therefore the probability of getting each H like this is $2^{-n(m-n)}\frac{p(m-n,m-n)}{p(m-n,m)}$. Given H the total number of matrices of size $n\times m$ and rank n that are "orthogonal" to H is $p(n,n)2^{n^2}$ (as follows from $(*)$) and therefore G will be generated with probability $2^{-n^2}/p(n,n)$. Hence the total probability of G under $\mathcal{V}_{n\times m}$ is $\mathcal{V}_{n\times m}(G) = 2^{-mn}\frac{p(m-n,m-n)}{p(m-n,m)p(n,n)}$. For every $i<j$, $p(j-i,j)p(i,i) = p(j,j)$. Therefore $\mathcal{V}_{n\times m}(G) = 2^{-mn}/p(n,m)$. This implies that $\mathcal{V}_{n\times m}$ is uniform over matrices of size $n\times m$ and rank n. The statistical distance between $\mathcal{V}_{n\times m}$ and $\mathcal{U}_{n\times m}$ equals to

$$\frac{1}{2}\sum_{G\in\{0,1\}^{n\times m}} |\mathcal{V}_{n\times m}(G) - \mathcal{U}_{n\times m}(G)| \quad =$$

$$\frac{1}{2}\left[\sum_{\mathtt{rank}(G)<n} 2^{-mn} + \sum_{\mathtt{rank}(G)=n} 2^{-mn}\left(\frac{1}{p(n,m)}-1\right)\right] \quad = \quad 1-p(n,m)$$

According to equation (5.1), $1-p(n,m) < 1-(1-2^{-m+n}) = 2^{-m+n}$. $\square$ We can now prove that decoding of random linear codes implies attribute-efficient learning of parities from random examples only.

Theorem 5.3.4 *Assume that there exists an algorithm* `RandDec` *that corrects a random linear $[m,n]$ code from up to w errors with probability at least $1/2+\gamma$ for any constant γ. Then PAR(w) over $\{0,1\}^m$ is efficiently learnable from $m-n$ random examples.*

Proof: Let $\dot{\chi}_e \in \mathrm{PAR}(w)$ be the unknown parity function and $z_1, z_2, \ldots, z_{m-n}$ be random and uniform examples given by the example oracle. Let H be the $m\times(m-n)$ matrix whose column i is equal to z_i for each $i \leq m-n$. If H does not have rank $m-n$ we return χ_{0^m}. Otherwise let G be a random matrix such that $GH = 0^{n\times(m-n)}$ generated as in the description of $\mathcal{V}_{n\times m}$ for $\delta = \gamma/2$. The values of $\dot{\chi}_e$ on z_i's give us the vector eH. Let y be any solution to the linear equation $yH = eH$. Clearly $(y \oplus e)H = 0^{m-n}$ and therefore $y \oplus e$ equals to xG for some $x \in \{0,1\}^n$. This means that RandDec (if successful) will output x on input G and y. By the definition of x, $e = xG \oplus y$, giving us the desired parity function.

To analyze the success probability of the algorithm we observe that the procedure above generates G according to $\mathcal{V}_{n\times m}$ with probability at least $p(m-n,m)(1-\gamma/2) \geq 1-2^{-n}-\gamma/2$. According to Lemma 5.3.3, the statistical distance between the G generated as above and $\mathcal{U}_{n\times m}$ is at most 2^{-m+n}. RandDec is successful with probability $1/2+\gamma$ and therefore our algorithm will succeed with probability at least $1/2+\gamma-(\gamma/2+2^{-n}+2^{-m+n}) \geq 1/2$. $\square$ The transformation above produces an attribute-efficient algorithm only if $m-n$ is polynomial in w and $\log m$. According to the Gilbert-Varshamov bound, a random linear code will, with high probability, have distance $d = \Omega(\frac{m-n}{\log m})$. Therefore if the number of errors that RandDec can correct is at least $w = d^\alpha$ errors for some constant $\alpha > 0$ then the sample complexity of learning a parity of length at most w over m variables would equal $O(w^{1/\alpha}\log m)$. Therefore such an algorithm could be used to obtain an attribute-efficient algorithm for learning parities.

We have noted previously that using a parity learning algorithm to obtain a syndrome decoding algorithm requires the parity learning algorithm to be proper. When a distribution over examples is not restricted it is unknown whether proper learning of parities is harder than non-proper. Fortunately, when learning with respect to the uniform distribution any learning algorithm for parities can be converted to a proper and exact one (that is, with a hypothesis equal to the target function). We include a proof of this folklore fact for completeness.

Fact 5.3.5 *Let $\mathcal{A}$ be an algorithm that learns PAR(k) in time $t(n,k,\epsilon)$ and with sample complexity $s(n,k,\epsilon)$. Then there exists a probabilistic algorithm $\mathcal{A}'$ that learns PAR(k) properly and exactly in time $t(n,k,1/5)+\tilde{O}(nk)$ and using $s(n,k,1/5)$ samples.*

Proof: We assume for simplicity that if $\mathcal{A}$ is probabilistic then it succeeds with probability at least $3/4$. Let h be the output of $\mathcal{A}$ when running on an unknown parity $\dot{\chi}_e \in \mathrm{PAR}(k)$ with $\epsilon = 1/5$. Given h that is correct on $4/5$ of all the points we can use it simulate membership queries to $\dot{\chi}_e(x)$ as follows. Let $y \in \{0,1\}^n$ be any point and let x be a randomly and uniformly chosen point. Then $h(x) = \dot{\chi}_e(x)$ with probability at least $4/5$ and $h(x\oplus y) = \dot{\chi}_e(x\oplus y)$ with probability at least $4/5$. Therefore with probability at least $3/5$, $h(x) \oplus h(x\oplus y) = \dot{\chi}_e(x) \oplus \dot{\chi}_e(x\oplus y) = \dot{\chi}_e(y)$. We can increase the confidence in the label to $1-\delta$ by repeating this procedure for $O(\log(1/\delta))$ independent x's. Given these membership queries we can use a proper and exact MQ algorithm for learning PAR(k). A number of such algorithms are known running in time $\tilde{O}(nk)$ and using $O(k \log n)$ MQs (including `AEParityStat`(k) given in Theorem 5.4.1). In order to get correct answers to all the membership queries with probability at least 3/4 we need each of the MQs to be correct with probability $1-\delta$ for $\delta = \Omega(\frac{1}{k \log n})$. This means that making $O(k \log n)$ MQs will take $O(nk \log n \log(k \log n)) = \tilde{O}(nk)$ steps. Altogether we get algorithm $\mathcal{A}'$ that succeeds with probability at least 1/2 and has the claimed complexity bounds. □ We can now assume that algorithms for learning parity with respect to the uniform distribution are proper and exact (and in particular do not require parameter ϵ) and use this to obtain the other direction of the equivalence.

Theorem 5.3.6 *Assume that there exists an algorithm* $\texttt{AELearnPar}_{\mathcal{U}}(\texttt{k})$ *that efficiently learns PAR*(k) *over* $\{0,1\}^m$ *using at most* $q(m,k)$ *random examples. Then there exists an algorithm* `RandDec` *that corrects a random linear* $[m, m-q(m,k)]$ *code from up to* k *errors with probability at least* $1/2-\gamma$ *for any constant* $\gamma > 0$.

Proof: Let G and y be the input of `RandDec`, $n = m - q(m,k)$, x be the vector for which $y = xG \oplus e$ where $\texttt{weight}(e) \leq k$. If G is not of rank n we just return the vector 0^n. Otherwise let H be a random matrix such that $GH = 0^{n\times(m-n)}$ generated as rank $m-n$ we return χ_{0^m}. Otherwise let G be a random matrix such that $GH = 0^{n\times(m-n)}$ generated as in the description of $\mathcal{V}_{n\times m}$ for $\delta = \gamma/2$ (with the roles of G and H reversed).

The syndrome yH is equal to eH and gives the values of $\dot{\chi}_e$ on $q(m,k)$ columns of H. We feed these columns as random examples to $\texttt{AELearnPar}_{\mathcal{U}}(\texttt{k})$ and obtain $\dot{\chi}_e$ from it (if $\texttt{AELearnPar}_{\mathcal{U}}(\texttt{k})$ is successful). Given e we obtain x by solving the system of linear equations $xG = y \oplus e$. To analyze the success probability of the algorithm we observe that the procedure above generates H according to $\mathcal{V}_{m\times(m-n)}$ with probability at least $p(n,m)(1-\gamma/2) \geq 1 - 2^{-q(m,k)} - \gamma/2$. According to Lemma 5.3.3, the statistical distance between H's generated as above and $\mathcal{U}_{m\times(m-n)}$ is at most $2^{-m+(m-n)} = 2^{-n}$. Therefore $\texttt{AELearnPar}_{\mathcal{U}}(\texttt{k})$ will succeed with probability at least $1/2-2^{-n}$. This implies that `RandDec` will return the correct x with probability at least $1/2 - (2^{-m+q(m,k)} + 2^{-q(m,k)} + \gamma/2) \geq 1/2 - \gamma$. □

5.4 A Fast Randomized Algorithm for ae.naMQ Learning of Parities

We next present a simple randomized algorithm for ae.naMQ learning of parities. The only previously known ae.naMQ algorithm for learning parities is due to Hofmeister and is a deterministic algorithm based on constructing and decoding of BCH binary linear codes (see also Section 5.3.2)[Hof99]. The algorithm we present is substantially simpler and has essentially the same asymptotic complexity as Hofmeister's.

The basic idea of our algorithm is to use a distribution over $\{0,1\}^n$ for which each attribute is correlated with the parity function if and only if it is present in the parity.

Theorem 5.4.1 *For each $k \leq n$ there exists an algorithm* `AEParityStat`(k) *that ae.naMQ learns the class PAR(k) in time $O(nk \log n)$ and asks $O(k \log n)$ MQs.*

Proof: Let $\dot{\chi}_c$ be the target concept (such that $\texttt{weight}(c) \leq k$). We define $\mathcal{D}_{\frac{1}{t}}$ to be the product distribution such that for each i, $\mathbf{Pr}[x_i = 1] = \frac{1}{t}$. Let us draw a point x randomly according to distribution $\mathcal{D}_{\frac{1}{4k}}$. Then for each $i \leq n$

$$\begin{aligned}\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[x_i = 1 \text{ and } \dot{\chi}_c(x) = 1] &= \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1]\, \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[x_i = 1] \\ &= \frac{1}{4k}\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1] \ .\end{aligned}$$

Our second observation is that for any set of indices $B \subseteq [n]$ and the corresponding parity function $\dot{\chi}_b$,

$$\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_b(x) = 1] \leq 1 - \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\forall i \in B,\ x_i = 0] = 1 - (1 - \frac{1}{4k})^{|B|} \leq \frac{|B|}{4k} \ .$$

First examine the case that $c_i \neq 1$ and therefore does not influence $\dot{\chi}_c$. Then by the second observation,

$$\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1] = \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1] \leq \frac{k}{4k} \leq 1/4 \ .$$

Now assume that $c_i = 1$ and let $c' = c \oplus e_i$. Then $\dot{\chi}_{c'}(x)$ is independent of x_i and $\dot{\chi}_c(x) = 1$ if and only if $\dot{\chi}_{c'}(x) = 0$. Therefore

$$\begin{aligned}\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_c(x) = 1 \mid x_i = 1] &= \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_{c'}(x) = 0 \mid x_i = 1] \\ &= 1 - \mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[\dot{\chi}_{c'}(x) = 1] \geq 1 - \frac{k-1}{4k} > 3/4 \ .\end{aligned}$$

Hence estimation of $\mathbf{Pr}_{\mathcal{D}_{\frac{1}{4k}}}[x_i = 1 \text{ and } \dot{\chi}_c(x) = 1]$ within the half of the expectation can be used to find out whether $c_i = 1$. Lemma 2.3.1 for $\gamma = 1/2$ implies that by taking $O(k \log n)$ independent samples with respect to $\mathcal{D}_{\frac{1}{4k}}$ we will get that each estimate is correct with probability at least $1 - 1/(2n)$ and therefore we will discover c with probability at least $1 - n/(2n) = 1/2$. The running time of `AEParityStat`(k) is clearly $O(nk \log n)$. $\square$

5.5 Finding Fourier Coefficients and Weak DNF Learning

The original Jackson's algorithm for learning DNF expressions with respect to the uniform distribution is based on a procedure that weakly learns DNF with respect to the uniform distribution [Jac97]. The procedure for weak learning is essentially an algorithm that, given a Boolean function f finds a significant Fourier coefficient of f, if one exist. Jackson's algorithm is based on a technique by Goldreich and Levin for finding a significant Fourier coefficient [GL89] (also called the KM algorithm [KM93]). Bshouty, Jackson, and Tamon used a later algorithm by Levin [Lev93] to give a significantly faster weak learning algorithm [BJT99]. In this section we will briefly describe Levin's algorithm with improvements by Bshouty *et al.* Building on their ideas we then present an attribute-efficient and non-adaptive version of the improved Levin's algorithm. This algorithm will give us an ae.naMQ algorithm for weak learning of DNF expressions that will serve as the basis of our ae.naMQ algorithm for DNF learning.

A Fourier coefficient $\hat{\phi}(a)$ of a real-valued function ϕ over $\{0,1\}^n$ is said to be *θ-heavy* if $|\hat{\phi}(a)| \geq \theta$. For a Boolean f, $\mathbf{E}[f\chi_a] \geq \theta$ if and only if $\mathbf{Pr}[f = \chi_a] \geq 1/2 + \theta/2$. This means that $|\hat{f}(a)| \geq \theta$ is equivalent to either χ_a or $-\chi_a$ being a $(1/2-\theta/2)$-approximator of f. Therefore finding a significant Fourier coefficient of f is sometimes called weak parity learning [Jac97]. It can also be interpreted as a learning algorithm for parities in the agnostic learning framework of Haussler [Hau92] and Kearns *et al.* [KSS94] (*cf.* [FGKP06] for details).

Definition 5.5.1 *Let f be a Boolean function with at least one θ-heavy Fourier coefficient. Given $\theta > 0$ and access to MEM(f), the* weak parity learning *problem consists of finding a vector z such that $\hat{f}(z)$ is $\theta/2$-heavy.*

We will only consider algorithms for weak parity learning that are efficient, that is, produce the result in time polynomial in n, and θ^{-1}. In addition we are interested in weak parity learning algorithms that are attribute-efficient.

Definition 5.5.2 Attribute-efficient weak parity algorithm *is an algorithm that given k, θ, and MEM(f) for f that has a θ-heavy Fourier coefficient of degree at most k efficiently solves weak parity learning problem and asks polynomial in $k, \log n$, and θ^{-1} number of MQs.*

We follow the presentation of Levin's weak parity algorithm given by Bshouty *et al.* and refer the reader to their paper for detailed proofs of all the statements and smaller remarks (we use the same definitions and notation to simplify the reference). Levin's algorithm is based on estimating a Fourier coefficient $\hat{f}(a)$ by sampling f on randomly-chosen pairwise independent points. More specifically, the following pairwise independent distribution is generated. For a fixed m, a random m-by-n 0-1 matrix R is chosen and the set $Y = \{pR \mid p \in \{0,1\}^m \setminus \{0^m\}\}$ is formed. For different vectors p_1 and p_2 in $\{0,1\}^m \setminus \{0^m\}$, p_1R and p_2R are pairwise independent. The variance σ^2 of a Boolean function is upper-bounded by 1 and thus Bienaymé-Chebyshev's inequality (Lemma 2.3.2) implies that

$$\mathbf{Pr}_R\left[\left|\frac{\sum_{x\in Y} f(x)\chi_a(x)}{2^m-1} - \hat{f}(a)\right| \geq \gamma\right] \leq \frac{1}{(2^m-1)\gamma^2} \tag{5.2}$$

Therefore using a sample for $m = \log(16\rho^{-1}\theta^{-2}+1)$, $\sum_{x\in Y} f(x)\chi_a(x)$ will, with probability at least $1-\rho$, approximate $\hat{f}(a)$ within $\theta/4$.

On the other hand, $\sum_{x\in Y} f(x)\chi_a(x)$ is a summation over all (but one[2]) elements of a linear subspace of $\{0,1\}^n$ and therefore can be seen as a Fourier coefficient of f restricted to subspace Y. That is, if we define $f_R(p) = f(pR)$ then, by definition of Fourier transform, for every $z \in \{0,1\}^m$

$$\widehat{f_R}(z) = 2^{-m} \sum_{p\in\{0,1\}^m} f_R(p)\chi_z(p) \ .$$

This together with equality $\chi_a(pR) = \chi_{aR^T}(p)$ implies that $\hat{f}(a)$ is approximated by $\widehat{f_R}(aR^T)$ (with probability at least $1-\rho$).

All the coefficients $\widehat{f_R}(z)$ can be computed exactly in time $O(m2^m)$ via the FFT algorithm giving estimations to all the Fourier coefficients of f.

Another key element of the weak parity algorithm is the following equation.

Lemma 5.5.3 ([BJT99]) *For $c \in \{0,1\}^n$ let $f_c(x) = f(x\oplus c)$. Then $\widehat{f_c}(a) = \hat{f}(a)\chi_a(c)$.*

Proof:

$$\widehat{f_c}(a) = 2^{-n}\sum_{x\in\{0,1\}^n} f(x\oplus c)\chi_a(x) = 2^{-n}\sum_{x\in\{0,1\}^n} f(x)\chi_a(x\oplus c) = \hat{f}(a)\chi_a(c) \ . \tag{5.3}$$

□

Assuming that $\hat{f}(a) \geq \theta$ estimation of $\hat{f}(a)$ within $\theta/4$ (when successful) has the same sign as $\hat{f}(a)$. Similarly we can obtain the sign of $\widehat{f_c}(a)$. By Lemma 5.5.3, the sign of the product $\hat{f}(a)\widehat{f_c}(a)$ is equal to $\chi_a(c)$. This gives a way to make MQs for χ_a using the values $\widehat{f_{c,R}}(aR^T)$ for a random R. Levin and Bshouty *et al.* implicitly used this technique

[2]The value at 0^m does not influence the estimation substantially and therefore can be offset by slightly increasing the size of sample space Y [BJT99].

with a basic membership query algorithm for learning parities. The speed-up in Levin's algorithm is achieved by making each MQ to many χ_a's in parallel. Therefore only a non-adaptive membership query algorithm for learning parities can be used. In our next theorem we give an interpretation of improved Levin's algorithm that makes the use of a non-adaptive membership query algorithm explicit.

Theorem 5.5.4 *Let $\mathcal{B}(k)$ be an ae.naMQ algorithm for learning parities that runs in time $t(n,k)$ and uses $q(n,k)$ MQs. There exists an attribute-efficient and non-adaptive algorithm* `AEBoundedSieve`-$\mathcal{B}(\theta,k)$ *that, with probability at least $1-\delta$, solves the weak parity learning problem.* `AEBoundedSieve`-$\mathcal{B}(\theta,k)$ *runs in time $\tilde{O}\left(\theta^{-2}t(n,k)\cdot q(n,k)\log(1/\delta)\right)$ and asks $\tilde{O}\left(\theta^{-2}q^2(n,k)\log(1/\delta)\right)$ MQs.*

Proof: We assume for simplicity that $\mathcal{B}(k)$ succeeds with probability at least $3/4$. Besides that according to Fact 5.3.5, we can assume that $\mathcal{B}(k)$ is a proper algorithm.

Let S be the set of MQs for an execution of $\mathcal{B}(k)$. Choose randomly an m-by-n matrix R for $m = \log(16\theta^{-2}\cdot 4\cdot(q(n,k)+1)+1)$ and compute the Fourier transforms of $f_R = f_{0^n,R}$ and $f_{y,R}$ for each $y\in S$ via the FFT algorithm. Then, for each $z\in\{0,1\}^m$, we run $\mathcal{B}(k)$ with the answer to MQ $y\in S$ equal to $\texttt{sign}(\widehat{f_R}(z)\widehat{f_{y,R}}(z))$. If the output of $\mathcal{B}(k)$ is a parity function χ_a of length at most k then we test that $(i): |\widehat{f_R}(z)|\geq 3\theta/4$ and $(ii): aR^T = z$. If both conditions are satisfied we add a to the set of hypotheses H.

By equation (5.2), for a such that $|\hat{f}(a)|\geq\theta$ and $\texttt{weight}(a)\leq k$, with probability at least $1-\frac{1}{4(q(n,k)+1)}$, each of the estimations $\widehat{f_{y,R}}(aR^T)$ for $y\in S\cup\{0^n\}$ will be within $\theta/4$ of $\widehat{f_y}(a)$. In particular, with probability at least $3/4$, for all $y\in S\cup\{0^n\}$, $\texttt{sign}(\widehat{f_y}(a)) = \texttt{sign}(\widehat{f_{y,R}}(aR^T))$. If all the signs are correct then by Lemma 5.5.3, $\texttt{sign}(\widehat{f_R}(z)\widehat{f_{y,R}}(z)) = \chi_a(y)$ and as a result $\mathcal{B}(k)$ will succeed with probability at least $3/4$. Therefore a will satisfy both conditions (i) and (ii) and will be added as a possible hypothesis with probability at least $1/2$. Note that $\mathcal{B}(k)$ is executed on up to 2^m possible hypotheses while using the same set of queries S. This is only possible for a non-adaptive algorithm $\mathcal{B}(k)$.

On the other hand, for any fixed b such that $|\hat{f}(b)|<\theta/2$, if $bR^T = z$ (condition (ii)) then with probability at least $1-\frac{1}{4(q(n,k)+1)}\geq 7/8$, $\widehat{f_R}(z)$ approximates $\hat{f}(b)$ within $\theta/4$. This implies that $|\widehat{f_R}(z)|<3\theta/4$ and therefore condition (i) will be failed with probability at least $7/8$. This implies that b can be added to the set of hypotheses with probability at most $1/8$.

Now we use a simple method of Bshouty *et al.* to remove all "bad" (not $\theta/2$-heavy) hypotheses from the set of hypotheses without removing the "good" ones (θ-heavy) [BJT99]. We repeat the described algorithm ℓ times for independent choices of R and S generating ℓ sets of hypotheses (each of size at most 2^m). This procedure generates at most $\ell 2^m$ hypotheses. According to Chernoff's bound (Lemma 2.3.1) each "good" hypothesis appears in at least $1/3$ of all the sets with probability at least $1 - 2^{-\alpha\ell}$ and each fixed "bad" hypothesis appears in at least $1/3$ of all the sets with probability at most $2^{-\alpha\ell}$, for a fixed constant α (since $1/8 < 1/3 < 1/2$). Note that we need to fix a "bad" hypothesis to apply this argument. A hypothesis can be fixed as soon as it has appeared in a set of hypotheses. We then exclude the first set in which a hypothesis has appeared when counting the fraction of sets in which the hypothesis has appeared (Chernoff bound is now on $\ell - 1$ trials but this is insubstantial). By setting $\ell = (m + \log m + 2\log(1/\delta) + 3)/\alpha$ we will get that $\ell 2^m 2^{-\alpha\ell-1} \leq \delta/2$. Therefore the probability that a "bad" hypothesis will appear in $1/3$ of the sets is at most $\delta/2$. Similarly all "good" hypotheses will appear in $1/3$ of the sets with probability at least $1-\delta/2$. Thus by picking any a that appears in at least $1/3$ of all the sets we will find a $\theta/2$-heavy coefficient with probability at least $1-\delta$.

Computing each of the Fourier transforms takes $O(m2^m) = \tilde{O}(\theta^{-2}\cdot q(n,k))$ time. They are performed for each of $q(n,k)$ MQs of $\mathcal{B}$ and this is repeated $\ell = O(m+\log(1/\delta))$ times giving the total bound of $\tilde{O}(\theta^{-2}q^2(n,k)\log(1/\delta))$. For each of the 2^m values of z we run $\mathcal{B}(k)$ and tests (i) and (ii). This takes $O(2^m(t(n,k)+mn)) = \tilde{O}(\theta^{-2}t(n,k)\cdot q(n,k))$ time and is repeated $\ell = O(m + \log(1/\delta))$ times. Therefore the total running time is $\tilde{O}(\theta^{-2}\cdot t(n,k)\cdot q(n,k)\log(1/\delta))$. Similarly we observe that each of the estimations via FFT uses 2^m examples and $\ell\cdot(q(n,k)+1)$ such estimations are done. This implies that the sample complexity of the algorithm is $\tilde{O}\left(\theta^{-2}q^2(n,k)\log(1/\delta)\right)$. It can also be easily seen that all MQs are non-adaptive. □

Another way to see Theorem 5.5.4 is as a way to convert an ae.naMQ algorithm for learning of parities to an ae.naMQ algorithm for agnostic learning of parities.

By plugging `AEParityStat`(k) algorithm (Theorem 5.4.1) into Theorem 5.5.4 we obtain our weak parity learning algorithm.

Corollary 5.5.5 *There exists an attribute-efficient and non-adaptive weak parity learning algorithm* `AEBoundedSieve`$(\theta,\mathtt{k})$ *that succeeds with probability at least* $1-\delta$*, runs in time* $\tilde{O}\left(nk^2\theta^{-2}\log(1/\delta)\right)$*, and asks* $\tilde{O}\left(k^2\log^2 n\cdot\theta^{-2}\log(1/\delta)\right)$ *MQs.*

Jackson has proved that for every distribution $\mathcal{D}$, every DNF formula f has a parity function that weakly approximates f with respect to $\mathcal{D}$ [Jac97]. A refined version of this claim by Bshouty and Feldman shows that f has a short parity that weakly approximates f if the distribution is not too far from the uniform [BF02]. More formally, for a real-valued function ϕ we define $L_\infty(\phi) = \max_x\{|\phi(x)|\}$ and we view a distribution $\mathcal{D}$ as a function over $\{0,1\}^n$ that for a point x gives its probability weight under $\mathcal{D}$.

Lemma 5.5.6 ([BF02]) *For any Boolean function f of DNF-size s and a distribution $\mathcal{D}$*

over $\{0,1\}^n$ there exists a parity function χ_a such that

$$|\mathbf{E}_{\mathcal{D}}[f\chi_a]| \geq \frac{1}{2s+1} \text{ and } \texttt{weight}(a) \leq \log\left((2s+1)L_\infty(2^n\mathcal{D})\right) .$$

By combining this fact with Corollary 5.5.5 we get an algorithm for weakly learning DNF.

Theorem 5.5.7 *There exist an algorithm* $\texttt{WeakDNF}_{\mathcal{U}}(s)$ *that for a Boolean function f of DNF-size s given n, s, and access to MEM(f), with probability at least $1/2$, finds a $(\frac{1}{2} - \Omega(\frac{1}{s}))$-approximator to f with respect to $\mathcal{U}$. Furthermore,* $\texttt{WeakDNF}_{\mathcal{U}}(s)$ *runs in time $\tilde{O}\left(ns^2\right)$ and asks $\tilde{O}\left(s^2\log^2 n\right)$ non-adaptive MQs.*

Proof: Lemma 5.5.6 implies that there exists a parity χ_a on at most $\log(2s+1)$ variables such that $|\mathbf{E}_{\mathcal{U}}[f\chi_a]| = |\hat{f}(a)| \geq \frac{1}{2s+1}$. This means that f has a $\frac{1}{2s+1}$-heavy Fourier coefficient of degree at most $\log(2s+1)$. Using Corollary 5.5.5 for $\delta = 1/2$, we can find a $\frac{1}{2(2s+1)}$-heavy Fourier coefficient $\hat{f}(a')$ in time $\tilde{O}\left(ns^2\right)$ and using $\tilde{O}\left(s^2\log^2 n\right)$ non-adaptive MQs. The parity $\chi_{a'}$ or its negation $(\frac{1}{2} - \frac{1}{4(2s+1)})$-approximates f. □

The algorithm for weakly learning DNFs by Bshouty *et al.* requires $\tilde{O}\left(ns^2\right)$ MQs and runs in time[3] $\tilde{O}\left(ns^2\right)$ [BJT99].

5.6 Learning DNF Expressions

In this section we show an ae.naMQ algorithm for learning DNF expressions. Following Jackson's approach we first show how to generalize our weak DNF learning algorithm to other distributions [Jac97]. We then use Freund's boosting algorithm to obtain a strong DNF learning algorithm [Fre92]. Besides achieving attribute-efficiency and non-adaptiveness we show a way to speed up the boosting process by exploiting several properties of our `WeakDNF` algorithm.

5.6.1 Weak DNF Learning with Respect to Any Distribution

The first step in Jackson's approach is to generalize a weak parity algorithm to work for any real-valued function. We follow this approach and give a generalization of our $\texttt{AEBoundedSieve}(\theta,\texttt{k})$ algorithm (Corollary 5.5.5) to any real-valued and also randomized functions.

Lemma 5.6.1 *There exists an algorithm* $\texttt{AEBoundedSieveRV}(\theta,\texttt{k},\texttt{V})$ *that for any real-valued randomized function Ψ with a θ-heavy Fourier coefficient of degree at most k, given*

[3]The running time bound is based on use of a membership query oracle, that given any two vectors $x, y \in \{0,1\}^n$, passed to it "by reference", returns $f(x \oplus y)$ in $O(1)$ time.

k, θ, $V \geq \mathbf{Var}_{\mathcal{U},\Psi}(\Psi(x))$, and an oracle access to Ψ, finds, with probability at least $1-\delta$, a $\theta/2$-heavy Fourier coefficient of Ψ of degree at most k. The algorithm runs in time $\tilde{O}\left(nk^2\theta^{-2}V\log(1/\delta)\right)$ and asks $\tilde{O}\left(k^2\log^2 n \cdot \theta^{-2}V\log(1/\delta)\right)$ non-adaptive MQs.

Proof: By revisiting the proof of Theorem 5.5.4, we can see that the only place where we used the fact that f is Boolean and deterministic is when relying on equation (5.2) in which the variance of the random variable $f(x) \in \{-1,+1\}$ was upper-bounded by 1. In this bound $f(x)$ is already treated as a random variable on pairwise independent x's. For any point x, $\Psi(x)$ is independent of any other evaluations of Ψ and therefore evaluations of Ψ on pairwise independent points are pairwise independent. This implies that in order to estimate $\widehat{\Psi}(a)$ within $\theta/4$ we only need to account for the fact that the variance of $\Psi(x)$ is not necessarily bounded by 1. This can be done by using $\mathbf{Var}(\Psi) \leq V$ times more samples, that is, we set $m = \log(16V\theta^{-2}\cdot 4\cdot(q(n,k)+1)+1)$. It is now straightforward to verify that the rest of the proof of Theorem 5.5.4 is unchanged. The increase in the required sample size increases the running time and the sample complexity of the algorithm by a factor $\tilde{O}(V)$ giving us the claimed bounds. □

As in Jackson's work we use the generalized weak parity algorithm to obtain an algorithm that weakly learns DNF expressions with respect to any distribution. The algorithm is efficient only when the distribution function is "close" to the uniform and requires access to the value of the distribution function at any point x.

Theorem 5.6.2 *There exist an algorithm* `WeakDNF(s,B)` *that for a Boolean function f of DNF-size s and any distribution $\mathcal{D}$, given $n, s, B \geq L_\infty(2^n\mathcal{D}(x))$, access to MEM($f$), and an oracle access to $\mathcal{D}$, with probability at least $1-\delta$, finds a $(\frac{1}{2}-\Omega(\frac{1}{s}))$-approximator to f with respect to $\mathcal{D}$. Furthermore,* `WeakDNF(s,B)`

- *runs in time $\tilde{O}(ns^2B\log(1/\delta))$;*
- *asks $\tilde{O}(s^2\log^2 n\cdot B\log(1/\delta))$ non-adaptive MQs;*
- *returns a parity function of length at most $O(\log(sB))$ or its negation.*

Proof: Lemma 5.5.6 states that there exists a vector a of Hamming weight bounded by $O(\log(sL_\infty(2^n\mathcal{D})))$ such that $|\mathbf{E}_{\mathcal{D}}[f(x)\chi_a(x)]| = \Omega(1/s)$. But

$$\mathbf{E}_{\mathcal{D}}[f(x)\chi_a(x)] = \sum_x [f(x)\mathcal{D}(x)\chi_a(x)] = \mathbf{E}[f(x)2^n\mathcal{D}(x)\chi_a(x)] = \widehat{\psi}(a)\ , \tag{5.4}$$

where $\psi(x) = f(x)2^n\mathcal{D}(x)$. This means that $\psi(x)$ has a $\Omega(1/s)$-heavy Fourier coefficient of degree bounded by $O(\log(sL_\infty(2^n\mathcal{D}))) = O(\log(sB))$. We can apply `AEBoundedSieveRV` on $\psi(x)$ to find its $\Omega(1/s)$-heavy Fourier coefficient of degree $O(\log(sB))$. All we need to do this is to provide a bound V on the variance of $f(x)2^n\mathcal{D}(x)$.

$$\begin{aligned}\mathbf{Var}(f(x)2^n\mathcal{D}(x)) &= \mathbf{E}[(f(x)2^n\mathcal{D}(x))^2] - \mathbf{E}^2[f(x)2^n\mathcal{D}(x)] \\ &\leq L_\infty(2^n\mathcal{D}(x))\mathbf{E}[2^n\mathcal{D}(x)] - \mathbf{E}^2[f(x)2^n\mathcal{D}(x)] \leq L_\infty(2^n\mathcal{D}(x))\mathbf{E}[2^n\mathcal{D}(x)] \\ &= L_\infty(2^n\mathcal{D}(x)) \leq B \qquad (5.5)\end{aligned}$$

This bound on variance relies essentially on the fact that $\mathcal{D}(x)$ is a distribution function [4] and therefore $\mathbf{E}[2^n\mathcal{D}(x)] = \mathbf{E}_\mathcal{D}[1] = 1$. This improves on $L^2_\infty(2^n\mathcal{D}(x))$ bound for an unrestricted function $\mathcal{D}(x)$ that was used in analysis of previous weak DNF learning algorithms [Jac97, BJT99].

We can now run $\texttt{AEBoundedSieveRV}(\theta,\texttt{k},\texttt{V})$ for $\theta = \Omega(1/s)$, $k = O(\log(sB))$, $V = B$, and a simulated oracle access to $\psi = f2^n\mathcal{D}$ to obtain a' such that $|\widehat{\psi}(a')| = \Omega(1/s)$ and $\texttt{weight}(a') = O(\log(sB))$. By equation (5.4), we get that $|\mathbf{E}_\mathcal{D}[f(x)\chi_{a'}(x)]| = \Omega(1/s)$ and therefore $\chi_{a'}(x)$ or its negation $(\frac{1}{2} - \Omega(\frac{1}{s}))$-approximates f with respect to $\mathcal{D}$. The claimed complexity bounds can be obtained by using Lemma 5.6.1 for θ, k and V as above. $\square$

5.6.2 Background on Boosting a Weak DNF Learner

Jackson's DNF learning algorithm was obtained by converting his weak learning algorithm to a strong one via a *boosting* algorithm [Jac97]. *Boosting* is a general technique for improving the accuracy of a learning algorithm. It was introduced by Schapire who gave the first efficient boosting algorithm [Sch90]. Let $\mathcal{C}$ be a concept class and let $\texttt{WL}_\gamma$ be a weak learning algorithm for $\mathcal{C}$ that for any distribution $\mathcal{D}$, produces a $(1/2-\gamma)$-approximating hypothesis. Known boosting algorithms have the following structure.

- At stage zero $\texttt{WL}_\gamma$ is run on $\mathcal{D}_0 = \mathcal{D}$ to obtain h_0.
- At stage i a distribution $\mathcal{D}_i$ is constructed using $\mathcal{D}$ and previous weak hypotheses $h_0, \ldots, h_{i-1}$. The distribution $\mathcal{D}_i$ usually favors the points on which the previous weak hypotheses do poorly. Then random examples from $\mathcal{D}_i$ are simulated to run $\texttt{WL}_\gamma$ with respect to $\mathcal{D}_i$ and obtain h_i.
- After repeating this for a number of times an ϵ-approximating hypothesis h is created using all the generated weak hypotheses.

Jackson's use of Freund's boosting algorithm slightly deviates from this scheme as it provides the weak learner with the oracle that returns the density of the distribution

[4] Actual $\mathcal{D}(x)$ given to a weak learner will be equal to $c\mathcal{D}'(x)$ where $\mathcal{D}'(x)$ is a distribution and c is a constant in $[2/3, 4/3]$ [BJT99]. This modifies the bound above by a small constant factor.

function $\mathcal{D}_i$ at any desired point instead of simulating random examples with respect to $\mathcal{D}_i$. The WeakDNF algorithm also requires oracle access to $\mathcal{D}_i(x)$ and therefore we will use a boosting algorithm in the same way. The running time of Jackson's (and our) algorithm for weak learning of DNF expression depends polynomially on $L_\infty(2^n\mathcal{D})$ [Jac97] and therefore it can only be boosted by 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 Freund's boost-by-majority algorithm [Fre95] is used to produce distribution functions bounded by $O(\epsilon^{-2})$. More recently, Klivans and Servedio have observed [KS03] that a later Freund's B-Comb boosting algorithm [Fre92] produces distribution functions bounded by $\tilde{O}(1/\epsilon)$, thereby improving the dependence of running time and sample complexity on ϵ. This improvement together with improved weak DNF learning algorithm due to Bshouty *et al.* [BJT99] gives DNF learning algorithm that runs in $\tilde{O}(ns^6/\epsilon^2)$ time and has sample complexity of $\tilde{O}(ns^4/\epsilon^2)$.

Remark 5.6.3 *Bshouty* et al. *claimed sample complexity of $\tilde{O}(ns^2/\epsilon^2)$ based on erroneous assumption that sample points for weak DNF learning can be reused across boosting stages. A distribution function $\mathcal{D}_i$ in i-th stage depends on hypotheses produced in previous stages. The hypotheses depend on random sample points and therefore in i-th stage the same set of sample points cannot be considered as chosen randomly and independently of $\mathcal{D}_i$ [Jac04]. This implies that new and independent points have to be sampled for each boosting stage and increases the sample complexity of the algorithm by Bshouty* et al. *by a factor of $O(s^2)$.*

As in the work of Klivans and Servedio [KS03], we use Freund's B-Comb boosting algorithm [Fre92] to boost the accuracy of our weak DNF learning algorithm. We will now briefly describe the B-Comb boosting algorithm (see also the work of Klivans and Servedio for a detailed discussion on application of B-Comb to learning DNF expressions [KS03]).

Freund's B-Comb Boosting Algorithm

B-Comb boosting algorithm is based on a combination of two other boosting algorithms. The first one in an earlier F1 algorithm due to Freund and is used to boost from accuracy $\frac{1}{2} - \gamma$ to accuracy 1/4 [Fre95]. Its output is the function equal to the majority vote of the weak hypotheses that it received. This algorithm is used as a weak learner by the second boosting algorithm B-Filt. At stage k B-Filt sets h_ℓ to be either the output of a weak learner or a random coin flip (that is a randomized function equal to either 1 or -1, each with probability 1/2). Accordingly the distribution function generated at stage i depends on random coin flips and the final hypothesis is a majority vote over hypotheses from the weak learner and random coin flips. As it is done by Freund, we analyze the algorithm for a fixed setting of these coin flip hypotheses [Fre92]. Freund's analysis shows that with overwhelming probability over the coin flips the randomized hypothesis produced by the boosting algorithm ϵ-approximates the target function.

Each of the executions of `F1` has $O(\gamma^{-2})$ stages and `B-Filt` has $O(\log(1/\epsilon))$ stages. We denote the distribution function generated at stage i of `F1` during stage ℓ of `B-Filt` as $\mathcal{D}^{\texttt{Comb}}_{\ell,i}$. In both boosting algorithms $\mathcal{D}_i(x) = \beta(i, N(x))\mathcal{D}/\alpha$, where $N(x)$ is the number of previous hypotheses that are correct on x, β is a fixed function from a pair of integers to the interval $[0,1]$ computable in polynomial (in the length of its input) time, and α is the normalization factor equal to $\mathbf{E}_{\mathcal{D}}[\beta(i, N(x))]$. We can therefore say that

$$\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x) = \beta(\ell, N_{\texttt{Filt}}(x)) \cdot \beta(i, N_{\texttt{F1}}(x))\mathcal{D}(x)/(\alpha_\ell \alpha_{\ell,i}) \ , \tag{5.6}$$

where $N_{\texttt{Filt}}(x)$ and $N_{\texttt{F1}}(x)$ count the correct hypotheses so far for `B-Filt` and `F1` respectively. The normalization factor α_ℓ equals $\mathbf{E}_{\mathcal{D}}[\beta(\ell, N_{\texttt{Filt}}(x))]$ and

$$\alpha_{\ell,i} = \mathbf{E}_{\mathcal{D}}[\beta(\ell, N_{\texttt{Filt}}(x)) \cdot \beta(i, N_{\texttt{F1}}(x))\mathcal{D}(x)/\alpha_\ell] \ .$$

The analysis by Freund implies that for every ℓ and i,

$$L_\infty(2^n \mathcal{D}^{\texttt{Comb}}_{\ell,i}) \leq 1/(\alpha_\ell \alpha_{\ell,i}) = \tilde{O}(1/\epsilon) \ . \tag{5.7}$$

In Figure 5.6.2 we include the pseudocode of `B-Comb` algorithm simplified and adapted to our setting.

5.6.3 Optimized Boosting

We now use Freund's `B-Comb` boosting algorithm [Fre92] to boost the accuracy of our weak DNF learning algorithm. Unlike in the previous work, we will exploit several properties of `WeakDNF` to achieve faster execution of each boosting stage. Specifically, we note that evaluation of the distribution function $\mathcal{D}_i(x)$ at boosting stage i involves evaluation of $i-1$ previous hypotheses on x and therefore, in a general case, for a sample of size q will require $\Omega(i \cdot q)$ steps, making the last stages of boosting noticeably slower. Our goal is to show that for our `WeakDNF` algorithm and the `B-Comb` boosting algorithm the evaluation of $\mathcal{D}_i(x)$ for the whole sample needed by `WeakDNF` can be made more efficiently.

The idea of the speed-up is to use equation (5.6) together with the facts that weak hypotheses are parities and MQs of `WeakDNF` come from a "small" number of low-dimension linear subspaces. Let g be a function that is equal to a linear combination of short parity functions. We start by showing a very efficient way to compute the values of g on a

```
B-Comb($\epsilon, \delta, \mathcal{D}, \mathtt{WL}_\gamma$)
 1. $k \leftarrow c_0 \log(1/\epsilon)$
 2. $\Theta \leftarrow c_1\epsilon/\log(1/\epsilon)$
 3. $h_0 \leftarrow$ F1$(1/4, \delta/(2k+1), \mathcal{D}, \mathtt{WL}_\gamma)$
 4. for $\ell \leftarrow 1$ to $k$
 5.     $N(x) \equiv |\{h_j \mid 0 \le j \le \ell - 1 \text{ and } h_j(x) = f(x)\}|$
 6.     $\alpha'_\ell \leftarrow$ EstExpRel$(\beta(\ell, N(x)), \mathcal{D}, 1/3, \delta/(2k+1))$
 7.     if $\alpha'_\ell \ge \Theta$ then
 8.         $\mathcal{D}'_\ell \equiv \beta(\ell, N(x))/\alpha'_\ell$
 9.         $h_\ell \leftarrow$ F1$(1/4, \delta/(2k+1), \mathcal{D}'_\ell, \mathtt{WL}_\gamma)$
10.     else
11.         $h_\ell \leftarrow$ Random$(1/2)$
12. end for
13. return Majority$(h_0, h_1, \ldots, h_k)$

F1($\epsilon, \delta, \mathcal{D}, \mathtt{WL}_\gamma$)
 1. $k \leftarrow c_2/\gamma^2$
 2. $\Theta \leftarrow c_3\epsilon^2$
 3. $h_0 \leftarrow \mathtt{WL}_\gamma(\mathcal{D}, \delta/(2k+1))$
 4. for $i \leftarrow 1$ to $k$
 5.     $N(x) \equiv |\{h_j \mid 0 \le j \le i - 1 \text{ and } h_j(x) = f(x)\}|$
 6.     $\alpha'_i \leftarrow$ EstExpRel$(\beta(i, N(x)), \mathcal{D}, 1/3, \delta/(2k+1))$
 7.     if $\alpha'_i \ge \Theta$ then
 8.         $\mathcal{D}'_i \equiv \beta(i, N(x))/\alpha'_i$
 9.         $h_i \leftarrow \mathtt{WL}_\gamma(\mathcal{D}'_i, \delta/(2k+1))$
10.     else
11.         $k \leftarrow i - 1$
12.         break for
13. end for
14. return Majority$(h_0, h_1, \ldots, h_k)$
```

Figure 5.1: Pseudocode of `B-Comb` boosting algorithm. The first part is `B-Filt` with `F1` used as a weak learner. $\mathtt{WL}_\gamma$ is a weak learning algorithm that has accuracy $\frac{1}{2} - \gamma$ and takes an oracle for a distribution $\mathcal{D}$ and confidence δ as parameters. `EstExpRel`$(R, \mathcal{D}, \lambda, \delta)$ produces estimates of the expectation of a random variable R with respect to a distribution $\mathcal{D}$ within relative accuracy λ and confidence δ (that is the estimate $v' \in [(1-\lambda)v, (1+\lambda)v]$, where v is the true expectation). Various unspecified constants are denoted by $c_0, c_1, \ldots$ The membership query oracle for the target function f is available to all procedures.

linear subspace of $\{0,1\}^n$. We will assume that vectors of Hamming weight at most w are represented by the list of indices where the vector is equal to 1 (as we did for parities). One can easily see that adding such vectors or multiplying them by any vector takes $O(w \log n)$ time.

Lemma 5.6.4 *Let $\{c_1, c_2, \ldots, c_i\}$ be a set of vectors in $\{0,1\}^n$ of Hamming weight at most w; $\bar{\alpha} \in \mathbb{R}^i$ be a real-valued vector, and R be a m-by-n 0-1 matrix. Then the set of pairs*

$$S = \{\langle p, \sum_{j \le i} \alpha_j \chi_{c_j}(pR) \rangle \mid p \in \{0,1\}^m\}$$

can be computed in time $\tilde{O}(i \cdot w \log n + 2^m)$.

Proof: We define $g(x) = \sum_{j \le i} \alpha_j \chi_{c_j}(x)$ and for $p \in \{0,1\}^m$ we define $g_R(p) = g(pR)$ (as in Sect. 5.5). Our goal is to find the values of function g_R on all the points of $\{0,1\}^m$. The function g is given as a linear combination of parities, or in other words, we are given its Fourier transform. Given the Fourier transform of g we can derive the Fourier transform of g_R from the following equation:

$$g_R(p) = \sum_{j \le i} \alpha_j \chi_{c_j}(pR) = \sum_{j \le i} \alpha_j \chi_{c_j R^T}(p) = \sum_{z \in \{0,1\}^m} \left[\Big(\sum_{j \le i;\ c_j R^T = z} \alpha_j \Big) \chi_z(p) \right] .$$

Hence $\widehat{g_R}(z) = \sum_{j \le i;\ c_j R^T = z} \alpha_j$. Given the Fourier transform of g_R we can use the FFT algorithm to perform the inverse Fourier transform of g_R giving us the desired values of $g_R(p)$ on all the points of $\{0,1\}^m$. This task can be performed in $O(m 2^m)$ steps. To compute the Fourier transform of g_R we need to compute $c_j R^T$ for each $j \le i$ and sum the ones that correspond to the same z. Given that each c_j is of Hamming weight w, $c_j R^T$ can be computed in $O(wm \log n)$ steps (note that we do not read the entire matrix R). Therefore the computation of the Fourier transform and the inversion using the FFT algorithm will take $O(m(iw \log n + 2^m)) = \tilde{O}(i \cdot w \log n + 2^m)$ steps. $\square$ Note that a straightforward computation would take $\Omega(iw 2^m \log n)$ steps. We apply Lemma 5.6.4 to speed up the evaluation of $\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x)$ on points at which `WeakDNF` asks non-adaptive MQs (here again we will rely on the non-adaptiveness of the weak learning algorithm). The speed-up is based on the following observations.

1. `WeakDNF` is based on estimating Fourier coefficients on a "small" number of linear subspaces of $\{0,1\}^n$ (as in equation (5.2)).

2. `WeakDNF` produces a short parity function (or its negation) as the hypothesis.

3. In computation of $\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x)$ the only information that is needed about the previous hypotheses is $N_{\texttt{Filt}}(x)$ and $N_{\texttt{F1}}(x)$, that is the number of hypotheses so far that are correct on the given point. The number of correct hypotheses is determined by $f(x)$ and the sum (in particular, a linear combination) of the values of the hypotheses on x.

Now we prove these observations formally and show a more efficient way to compute $\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x)$ given oracle access to $N_{\texttt{Filt}}(x)$, in other words, we show a more efficient way to compute $N_{\texttt{F1}}(x)$.

Lemma 5.6.5 *Let $\{b_1\chi_{c_1}, b_2\chi_{c_2}, \ldots, b_i\chi_{c_i}\}$ be the hypotheses returned by* `WeakDNF`(s,B) *in i first stages of* `F1` *boosting algorithm during stage ℓ of* `B-Filt`*, where $b_j \in \{-1,+1\}$ is the sign of χ_{c_j} (indicating whether or not it is negated). Let W be the set of queries for the $(i+1)$-th execution of* `WeakDNF`(s,B) *with confidence parameter δ and $B \geq L_\infty(2^n\mathcal{D}^{\texttt{Comb}}_{\ell,i})$. Then, given MEM$(f)$ and an oracle access to $N_{\texttt{Filt}}(x)$, the set of pairs*

$$S = \{\langle x, \lambda\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x)\rangle \mid x \in W\}$$

for some constant $\lambda \in [2/3, 4/3]$, can be computed, with probability at least $1-\delta$, in time $\tilde{O}((i + s^2B)\log^2 n \log(1/\delta))$.

Proof: We start by proving our first observation. By revisiting the proof of Theorem 5.5.4 we can see that our weak parity algorithm asks queries on $Y = \{pR \mid p \in \{0,1\}^m\}$ for a randomly chosen R and then for each query z of a ae.naMQ parity algorithm it asks queries on points of the set $Y_z = \{z \oplus y \mid y \in Y\}$. The set Y_z is a subset of the linear subspace of dimension $m+1$ spanned by the rows of R and vector y. These queries are then repeated $O(m+\log(1/\delta))$ times to single out "good" Fourier coefficients. Therefore by substituting the parameters of `WeakDNF`(s,B) into the proofs of Lemma 5.6.1 and Theorem 5.5.4, we can see that W can be decomposed into $\tilde{O}(\log^2(sB)\log n\log(1/\delta))$ linear subspaces of dimension $m = \log T$ for $T = \tilde{O}(s^2B\log n)$.

Our second observation is given by Theorem 5.6.2 and states that for each $j \leq i$, χ_{c_j} is a parity on at most $\log sB$ variables.

Our next observation is that the number of hypotheses from $\{b_1\chi_{c_1}, b_2\chi_{c_2}, \ldots, b_i\chi_{c_i}\}$ that agree with f on x equals to

$$N_{\texttt{F1}}(x) = \frac{f(x)\left(\sum_{j\leq i} b_j\chi_{c_j}(x)\right)}{2} + \frac{i}{2}\ ,$$

that is, given $\sum_{j\leq i} b_j\chi_{c_j}(x)$ and $f(x)$, $N_{\texttt{F1}}(x)$ can be computed in $O(1)$ steps. According to Lemma 5.6.4, we can compute $\sum_{j\leq i} b_j\chi_{c_j}(x)$ on a linear subspace of dimension m in time $\tilde{O}(iw \log n + 2^m)$. Together with the first observation this implies that computing $N_{\mathtt{F1}}(x)$ for all points in W can be done in time

$$\tilde{O}(\log^2 (sB) \log n \log (1/\delta))\tilde{O}(i \cdot \log (sB) \log n + s^2 B \log n) = \tilde{O}((i + s^2 B) \log^2 n \log (1/\delta)) \ .$$

Equation (5.6) implies that for every point x, given $N_{\mathtt{F1}}(x)$, oracle access to $N_{\mathtt{Filt}}(x)$ and $\alpha_\ell \alpha_{\ell,i}$ we obtain $\mathcal{D}^{\mathtt{Comb}}_{\ell,i}(x)$. The normalization factor $\alpha_{\ell,i}$ is estimated with relative accuracy $1/3$ and therefore instead of the true $\mathcal{D}^{\mathtt{Comb}}_{\ell,i}(x)$ we will obtain $\lambda \mathcal{D}^{\mathtt{Comb}}_{\ell,i}(x)$ for some constant $\lambda \in [2/3, 4/3]$. □ Lemma 5.6.5 assumes oracle access to $N_{\mathtt{Filt}}(x)$. In the next lemma we show that this oracle can be simulated efficiently.

Lemma 5.6.6 *Let $\{h_0, h_1, \ldots, h_{\ell-1}\}$ be the set of hypotheses obtained by* `B-Comb` *in ℓ first stages of boosting. Let W be the set of queries for the $(i+1)$-th execution of* `WeakDNF`(s, B) *with confidence parameter δ and $B \geq L_\infty(2^n \mathcal{D}^{\mathtt{Comb}}_{\ell,i})$. Then, given MEM($f$), the set of pairs $S = \{\langle x, N_{\mathtt{Filt}}(x)\rangle \mid x \in W\}$ can be computed, with probability at least $1 - \delta$, in time $\tilde{O}(\ell s^2 B \cdot \log^2 n \log (1/\delta))$.*

Proof: For each $j \leq \ell - 1$, h_j is an output of `F1` or a random coin flip hypothesis. `WeakDNF`(s, B) returns $(\frac{1}{2} - \Omega(\frac{1}{s}))$-approximate hypotheses and therefore each hypothesis generated by `F1` is a majority vote of $O(\gamma^- 2) = O(s^2)$ short parities (or their negations). A majority vote of these parities and their negations is simply the sign of their sum, and in particular is determined by a linear combination of parity functions. Hence, as in Lemma 5.6.5, $h_j(x)$ for all points in W can be computed $\tilde{O}((s^2 + s^2 B) \log^2 n \log (1/\delta))$ time. Therefore for any stage ℓ, $h_0, h_1, \ldots, h_{\ell-1}$ can be computed on points in W in $\tilde{O}(\ell s^2 B \log^2 n \log (1/\delta))$ steps giving the required oracle $N_{\mathtt{Filt}}(x)$. □

Remark 5.6.7 *In this simulation of* `B-Comb` *we ignored the running time and sample complexity of procedure* `EstExpRel` *that is used to evaluate the normalization factors. The factor $\alpha_\ell = \mathbf{E}[\beta(\ell, N_{\mathtt{Filt}}(x))]$ needs to be estimated within relative accuracy $1/3$ and its value is only used when the estimate $\alpha'_\ell \geq \Theta = c_1\epsilon / \log (1/\epsilon)$ for some constant c_1 since otherwise* `B-Comb` *uses a random coin flip hypothesis (see line 7 of the pseudocode). This implies that the estimate is only used when $\alpha_\ell \geq 3\Theta/4$. The Chernoff bound (Lemma 2.3.1) implies that if $\alpha_\ell \geq 3\Theta/4$ then using $M = O(\frac{log(1/\epsilon) \log (1/\delta)}{\epsilon})$ random uniform samples will be sufficient to estimate α_ℓ within relative accuracy $1/3$ with confidence $1 - \delta$. If $\alpha_\ell < 3\Theta/4$ then with probability $1 - \delta$ the obtained estimate α'_ℓ will be less than Θ and therefore will not be used. Evaluating $N_{\mathtt{Filt}}(x)$ on each of these points will take $O(\ell n s^2)$ steps and therefore each of these estimation will run in time $\tilde{O}(\ell n s^2 \log (1/\delta)/\epsilon)$.*

At each stage of the `F1` *boosting algorithm we need to estimate*

$$\alpha_{\ell,i} = \mathbf{E}[\beta(\ell, N_{\texttt{Filt}}(x)) \cdot \beta(i, N_{\texttt{F1}}(x))/\alpha'_\ell]$$

to within relative accuracy $1/3$ *and its value is only used when the estimate* $\alpha'_{\ell,i} \geq c$ *for some constant c. Therefore it is sufficient to estimate* $\alpha_{\ell,i}$ *to within constant additive accuracy. With probability at least* $1-\delta$ *this can be achieved by using a sample of* $O(\log(1/\delta))$ *random uniform points. Estimating both* $N_{\texttt{Filt}}(x)$ *and* $N_{\texttt{F1}}(x)$ *on each point takes* $O(\ell n s^2)$ *steps and therefore each of these estimations runs in time* $O(\ell n s^2 \log(1/\delta))$*.*

We are now ready to describe the resulting ae.naMQ algorithm for learning DNF expressions.

Theorem 5.6.8 *There exists an algorithm* `AENALearnDNF`(s) *that for any Boolean function* f *of DNF-size* s*, given* n, s, ϵ*, and access to MEM*(f)*, with probability at least* $1/2$*, finds an* ϵ*-approximator to* f *with respect to* $\mathcal{U}$*. Furthermore,* `AENALearnDNF`(s) *runs in time* $\tilde{O}\left(ns^4/\epsilon\right)$ *and asks* $\tilde{O}\left(s^4 \log^2 n/\epsilon\right)$ *non-adaptive MQs.*

Proof: As we know from the description of `B-Filt`, it has $O(\log(1/\epsilon))$ stages and for each ℓ and i, $L_\infty(2^n \mathcal{D}^{\texttt{Comb}}_{\ell,i}) = \tilde{O}(1/\epsilon)$. Therefore the running time of each execution of `WeakDNF`(s, B) is $\tilde{O}(ns^2/\epsilon)$. In particular, for every boosting stage of `F1`, it dominates the running time of computing the distribution function $\mathcal{D}^{\texttt{Comb}}_{\ell,i}$ (Lemmas 5.6.5 and 5.6.6) and estimations of α_ℓ and $\alpha_{\ell,i}$ (Remark 5.6.7). There are total $O(s^2 \log(1/\epsilon))$ executions of `WeakDNF` and therefore the total running time of `AENALearnDNF`(s) is $\tilde{O}\left(ns^4/\epsilon\right)$ and the total number of non-adaptive MQs used is $\tilde{O}\left(s^4 \log^2 n/\epsilon\right)$. □

The improvements to the algorithm by Bshouty *et al.* are summarized below [BJT99].

- The use of attribute-efficient weak learning improves the total sample complexity from $\tilde{O}\left(ns^4/\epsilon^2\right)$ to $\tilde{O}\left(s^4 \log^2 n/\epsilon^2\right)$ and the same running time is achieved without assumptions on the MQ oracle (see Theorem 5.5.7).
- Faster computation of distribution functions used in boosting improves the total running time from $\tilde{O}\left(ns^6/\epsilon^2\right)$ to $\tilde{O}\left(ns^4/\epsilon^2\right)$ (see Lemmas 5.6.4, 5.6.5 and 5.6.6).
- Tighter estimation of variance improves the dependence of running time and sample complexity on ϵ from $1/\epsilon^2$ to $1/\epsilon$ (equation (5.5)).

Remark 5.6.9 *While the analysis of the speedup was done for Freund's* `B-Comb` *booster the same idea works for any other booster in which estimation of new weight function is based on a linear combination of previous hypotheses. In particular, for the other known boosting algorithms that produce smooth distributions:* `SmoothBoost` *by Servedio [Ser03] and* `AdaFlat` *by Gavinsky [Gav03].*

5.7 Handling Noise

Now we would like to show that our DNF learning algorithm can be modified to tolerate random persistent classification noise in MQs. To simplify the proof we first show that we can assume that we are dealing with random and independent classification noise.

Lemma 5.7.1 *The probability that* `AENALearnDNF`(s) *asks an MQ for the same point more than once is upper bounded by* $P \cdot 2^{-n/\log Q}$ *where* P *and* Q *are polynomial in* n, s *and* $1/\epsilon$.

Proof: We start by observing that in the algorithm `AENALearnDNF`(s) all the points that are given to the MQ oracle are chosen uniformly and the points that are used in different executions of `WeakDNF` are independent. As can be seen from the proof of Theorem 5.5.4, the generated points are of the form $pR \oplus y$, where R is a randomly and uniformly chosen matrix, y is chosen randomly according to $\mathcal{D}_{\frac{1}{4k}}$ (defined in Theorem 5.4.1) or equal to 0^n, and $p \in \{0,1\}^m$. Points generated for two randomly chosen R_1 and R_2 are independent of each other and uniformly distributed. Let $y_0 = 0^n$, q be the number of samples taken from $\mathcal{D}_{\frac{1}{4k}}$, and $y_1, y_2, \ldots, y_q$ denote the samples.

For some randomly chosen R, let $x_1 = p_1 R \oplus y_i$ and $x_2 = p_2 R \oplus y_j$ be two different sample points. For two different sample points either $i \neq j$ or $p_1 \neq p_2$. If $i \neq j$ then either $i \neq 0$ or $j \neq 0$. Without loss of generality we assume that $i \neq 0$. Then

$$\mathbf{Pr}_{y_i \sim \mathcal{D}_{\frac{1}{4k}}}[p_1 R \oplus y_i = p_2 R \oplus y_j] = \mathbf{Pr}_{y_i \sim \mathcal{D}_{\frac{1}{4k}}}[y_i = p_1 R \oplus p_2 R \oplus y_j] \leq (1 - \frac{1}{4k})^n \leq e^{-n/(4k)} \ .$$

If $p_1 \neq p_2$ then $\mathbf{Pr}_{R \sim \mathcal{U}_{m \times n}}[(p_1 \oplus p_2)R = y_i \oplus y_j] = 2^{-n}$. This implies that for any two MQs made by `AENALearnDNF`(s), probability that they are equal is at most $e^{-n/(4k)}$. As it can be seen from the analysis of `AENALearnDNF`(s), $k = O(\log(s/\epsilon))$ and the total number of MQs used is polynomial in n, s and $1/\epsilon$. □

If an algorithm does not ask a MQ for the same point again then persistent classification noise can be treated as random and independent.

5.7.1 Boosting Weak Parity Learning Algorithm in the Presence of Noise

The main part of the modification is to show an algorithm that can locate heavy Fourier coefficients of any randomized function can be used to learn DNFs in the presence of noise. Our method can be applied in more general setting. In particular, it could be used to prove that Jackson's original algorithm is resistant to persistent noise in MQs and was recently used to produce a noise tolerant DNF learning algorithm by Feldman *et al.* [FGKP06]. Previous methods to produce noise-tolerant DNF learning algorithms gave statistical query analogues of Jackson's algorithm and then simulated statistical queries[5] in the presence of noise [JSS97, BF02]. Our approach is more direct and the resulting algorithm is substantially more efficient than the previous ones.

The goal of a weak DNF learning algorithm at stage i of boosting is to find a parity correlated with the function $2^n \mathcal{D}_i(x) f(x)$ given an oracle access to values of $\mathcal{D}_i(x)$ and the oracle for f with noise of rate $\eta < 1/2$ instead of MEM(f). Handling the noisy case is further complicated by the fact that the computation of $\mathcal{D}_i(x)$ by the boosting algorithm uses the value $f(x)$ (in particular, `B-Comb` and `B-Filt` need the value of $f(x)$ to compute $N(x)$) which is not available in the noisy case. To make this dependence explicit we define $\mathcal{D}_i(x, b)$ (for $b \in \{-1, +1\}$) to be the value of $\mathcal{D}_i$ on x when the boosting algorithm is supplied with the value b in place of $f(x)$ to compute $D_i(x)$ (in particular, $\mathcal{D}_i(x) = \mathcal{D}_i(x, f(x))$). We will now show a general method to compute a Fourier coefficient of a function that depends on $f(x)$ given a noisy oracle for f.

Lemma 5.7.2 *Let $g(x, b)$ be any real-valued function over $\{0,1\}^n \times \{-1, +1\}$ and let Φ^η denote a randomized function such that for every x, $\Phi^\eta(x) = f(x)$ with probability $1-\eta$ and $\Phi^\eta(x) = -f(x)$ with probability η. Then for each $a \in \{0,1\}^n$, $[g(\widehat{x, f(x)})](a) = \widehat{\Psi_{g,\eta}}(a)$, where $\Psi_{g,\eta}$ is a randomized function defined as*

$$\Psi_{g,\eta}(x) = \frac{1}{2}\left(\frac{1}{1-2\eta}(g(x,1) - g(x,-1)) \cdot \Phi^\eta(x) + g(x,1) + g(x,-1)\right) .$$

Proof: We use the following observation due to Bshouty and Feldman [BF02]. For any real-valued function $\psi(x, b)$

$$\psi(x, f(x)) = \psi(x,-1)\frac{1-f(x)}{2} + \psi(x,1)\frac{1+f(x)}{2} =$$

$$\frac{1}{2}((\psi(x,1) - \psi(x,-1))f(x) + \psi(x,1) + \psi(x,-1)) .$$

Then

$$\mathbf{E}_{x,\Phi^\eta(x)}[\frac{1}{2}(\psi(x,1) - \psi(x,-1)) \cdot \Phi^\eta(x)] = (1-2\eta)\mathbf{E}_x[\frac{1}{2}(\psi(x,1) - \psi(x,-1))f(x)] ,$$

[5]They used stronger versions of statistical queries than those introduced by Kearns [Kea98].

and therefore we can offset the effect of noise in $g(x, f(x))$ as follows.

$$
\begin{aligned}
[\widehat{g(x, f(x))}](a) &= \mathbf{E}[g(x, f(x))\chi_a(x)] \\
&= \frac{1}{2}\left(\mathbf{E}_x[(g(x,1) - g(x,-1))\chi_a(x)f(x)] + \mathbf{E}_x[(g(x,1) + g(x,-1))\chi_a(x)]\right) \\
&= \frac{1}{2}\left(\frac{1}{1-2\eta}\mathbf{E}_{x,\Phi^\eta(x)}[(g(x,1) - g(x,-1))\chi_a(x)\cdot\Phi^\eta(x)] + \mathbf{E}_x[(g(x,1) + g(x,-1))\chi_a(x)]\right) \\
&= \mathbf{E}_{x,\Phi^\eta(x)}\left[\frac{1}{2}\left(\frac{1}{1-2\eta}(g(x,1) - g(x,-1))\cdot\Phi^\eta(x) + g(x,1) + g(x,-1)\right)\chi_a(x)\right] = \widehat{\Psi}(a)
\end{aligned}
$$

□ An oracle for $\Phi^\eta(x)$ is exactly the membership query oracle for $f(x)$ with noise of rate η that is given to us (by Lemma 5.7.1 we can ignore the persistency of noise). Therefore Lemma 5.7.2 gives a way to find heavy Fourier coefficients using an oracle for $\Phi^\eta(x)$ instead of the membership query oracle for $f(x)$. We apply it to `WeakDNF` and obtain our noise-tolerant ae.naMQ DNF learning algorithm.

Theorem 5.7.3 *There exists an algorithm* `AENALearnDNF`(s, η) *that for any Boolean function f of DNF-size s, given n, s, η, ϵ, and access to MEM(f) corrupted by random persistent classification noise of rate η, with probability at least $1/2$, finds an ϵ-approximator to f with respect to $\mathcal{U}$. Furthermore,* `AENALearnDNF`(s, η) *runs in time $\tilde{O}\left(ns^4/(\epsilon(1-2\eta)^2)\right)$ and asks $\tilde{O}\left(s^4\log^2 n/(\epsilon(1-2\eta)^2)\right)$ non-adaptive MQs.*

Proof: Section 5.6.3 gives a way to efficiently compute $\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x)$ given the label $f(x)$. This computation defines the oracle for $\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x, b)$ where b is the supposed label of $f(x)$. Let $g(x, b) = b \cdot 2^n\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x, b)$ and let $\Psi_{g,\eta}(x)$ be defined as in Lemma 5.7.2. Given the oracle for $\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x, b)$ and oracle access to $\Phi^\eta(x)$ we use `AEBoundedSieveRV`$(\theta,\mathtt{k},\mathtt{V})$ on $\Psi_{g,\eta}(x)$ in the same way it was used on $\psi(x)$ by `WeakDNF`(s, B) (see the proof of Theorem 5.6.2). By Lemma 5.7.2, $\Psi_{g,\eta}(x)$ has the same Fourier coefficients as $f(x)2^n\mathcal{D}^{\texttt{Comb}}_{\ell,i}(x, f(x))$. Therefore this modified weak learning algorithm will produce an equivalent hypothesis. We can deal with the noise while estimating the normalization factor $\alpha_{\ell,i}$ in exactly the same way.

Furthermore, the definition of $\Psi_{g,\eta}$ and equation (5.5) imply that

$$L_\infty(\Psi_{g,\eta}) \le \frac{2}{1-2\eta}L_\infty(2^n\mathcal{D}^{\texttt{Comb}}_{\ell,i}) \text{ and } \mathbf{Var}(\Psi_{g,\eta}) \le \frac{4}{(1-2\eta)^2}L_\infty(2^n\mathcal{D}^{\texttt{Comb}}_{\ell,i}) \ .$$

By substituting these bounds into Theorem 5.6.2 we obtain that the running time and the sample complexity of each execution of the modified weak learner will grow by $(1-2\eta)^2$. They also imply that `WeakDNF`(s, B) will produce parities on $\log\left(s^2/(\epsilon(1-2\eta))\right)$ variables (this change is absorbed by $\tilde{O}$ notation). $\square$

Chapter 6

Learning Parities with Noise

In this chapter we study learning of parities under the uniform distribution with random classification noise, also called the noisy parity problem. We reduce a number of basic problems regarding learning under the uniform distribution to learning of noisy parities, thus establishing the central role of this problem in learning under the uniform distribution. In particular, we show that under the uniform distribution, agnostic learning of parities, PAC learning of DNF expressions, and learning of k-juntas all reduce to learning parities with random classification noise. The first reduction also yields the first nontrivial algorithm for agnostic learning of parities under the uniform distribution.

6.1 Introduction

A parity function is the XOR of some set of variables $T \subseteq [n]$, where $[n]$ denotes the set $\{1, 2, \ldots, n\}$. This is one of most fundamental concept classes in learning theory. In particular, the powerful Fourier transform learning technique is based on representing functions as linear combination of parities. In the absence of noise, one can identify the set T by running Gaussian elimination on the given examples. The presence of noise in the labels, however, leads to a number of challenging and important problems. We address learning of parities in the presence of two types of noise.

Random classification noise is a relatively benign model of noise in which the label of each point is flipped randomly 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 $\frac{1}{1-2\eta}$ (see Section 2.2.3 for the details on the model).

The other model we consider is the *agnostic* learning model of Haussler [Hau92] and Kearns *et al.* [KSS94]. In this model, informally, nothing is known about the process that generated the examples and the learning algorithm is required to do nearly as well as is possible using hypotheses from a given class. This corresponds to a common empirical approach when few or no assumptions are made on the data and a fixed space of hypotheses is searched to find the "best" approximation of the unknown function (see Section 2.2.4 for more details on the model). Designing algorithms that learn in this model is notoriously hard and very few positive results are known [KSS94, LBW95, GKS01, KKMS05].

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

6.1.1 Overview and Previous Work

We start by summarizing the main known results about the problems of learning parities with random classification noise and learning parities in the agnostic model.

- **Adversarial Noise:** Without any restrictions on the distribution of examples the problem of (proper) agnostic learning parities is known to be NP-hard. This follows easily from NP-hardness of maximum-likelihood decoding of linear codes proved by Berlekamp *et al.* [BMvT78] (a significantly stronger version of this result follows from a celebrated result of Håstad [Has01]). We are unaware of non-trivial algorithms for this problem under any fixed distribution, prior to our work. The problem of learning parities with adversarial noise under the uniform distribution is equivalent to finding a significant Fourier coefficient of a Boolean function and related to the problem of decoding Hadamard codes. If the learner can ask *membership queries* (or queries that allow the learner to get the value of function f at any point), a celebrated result of Goldreich and Levin gives a polynomial time algorithm for this problem [GL89]. Later algorithms were given by Kushilevitz and Mansour [KM93], Levin [Lev93], Bshouty *et al.* [BJT04], and Feldman (see Section 5.5).

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

In this work, we focus on learning parities under the uniform distribution. We reduce a number of fundamental open problems on 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. In particular, our reduction and the result of Blum *et al.* imply the first non-trivial algorithm for learning parities with adversarial noise under the uniform distribution [BKW03].

Theorem 6.1.1 *For any constant $\eta < 1/2$, parities are learnable under the uniform distribution with adversarial noise of rate η in time $O(2^{\frac{n}{\log n}})$.*

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

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

Learning DNF formulae

Learning of DNF expressions from random examples is a famous open problem originating from Valiant's seminal paper on PAC learning [Val84]. In this problem we are given access to examples of a Boolean function f on points 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. The best known algorithm for learning DNF in this model was given by Klivans and Servedio [KS04a] and runs in time $2^{\tilde{O}(n^{1/3})}$.

For learning DNF under the uniform distribution a simple quasi-polynomial algorithm was given by Verbeurgt [Ver90]. His algorithm essentially collects all the terms of size $\log(s/\epsilon) + O(1)$ that are consistent with the target function, i.e. do not accept negative points and runs in time $O(n^{\log(s/\epsilon)})$. We are unaware of an algorithm improving on this approach. Jackson [Jac97] proved that DNFs are learnable under the uniform distribution if the learning algorithm is allowed to ask membership queries. This breakthrough and influential result gives essentially the only known approach to learning of unrestricted DNFs in polynomial time.

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

Theorem 6.1.2 *Let $\mathcal{A}$ be an algorithm that learns parities of k variables over $\{0,1\}^n$ for every noise rate $\eta < 1/2$ in time $T(n, k, \frac{1}{1-2\eta})$ using at most $S(n, k, \frac{1}{1-2\eta})$ examples. Then there exists an algorithm that learns DNF expressions of size s in time $\tilde{O}(\frac{s^4}{\epsilon^2} \cdot T(n, \log B, B) \cdot S(n, \log B, B)^2)$, where $B = \tilde{O}(s/\epsilon)$.*

Learning k-juntas

A Boolean function is a k-junta if it depends only on k variables out of n. Learning of k-juntas was proposed by Blum and Langley [BL97, Blu94], as a clean formulation of the problem of efficient learning in the presence of irrelevant features. Moreover, for $k = O(\log n)$, a k-junta is a special case of a polynomial-size decision tree or a DNF expression. Thus, learning juntas is a first step toward learning polynomial-size decision trees and DNFs under the uniform distribution. A brute force approach to this problem would be to take $O(k \log n)$ samples, and then run through all n^k subsets of possible relevant variables. The first non-trivial algorithm was given only recently by Mossel *et al.* [MOS04], and runs in time roughly $O(n^{0.7k})$. Their algorithm relies on new analysis of the Fourier transform of juntas. However, even the question of whether one can learn k-juntas in polynomial time for $k = \omega(1)$ still remains open (*cf.* [Blu03a]).

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

Theorem 6.1.3 *Let $\mathcal{A}$ be an algorithm that learns parities of k variables on $\{0,1\}^n$ for every noise rate $\eta < 1/2$ in time $T(n, k, \frac{1}{1-2\eta})$. Then there exists an algorithm that learns k-juntas in time $O(2^{2k} k \cdot T(n, k, 2^{k-1}))$.*

This reduction also applies to learning k-juntas with random noise. A parity of k variables is a special case of a k-junta. Thus we can reduce the noisy junta problem to a special case, at the cost of an increase in the noise level. By suitable modifications, the reduction from DNF can also be made resilient to random noise.

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

6.1.2 Organization

The rest of this chapter is organized as follows: we present our main technical Lemma regarding finding significant Fourier coefficients in Section 6.3. We derive the consequences of this Lemma in Sections 6.4.

6.2 Preliminaries

We start by briefly reviewing the relevant learning models. See Sections 2.2 for more details on the models.

6.2.1 Learning Models

The learning models discussed in this work are based on Valiant's well-known PAC model [Val84]. In this model, for a concept c and distribution $\mathcal{D}$ over X, an *example oracle* $\mathrm{EX}(c,\mathcal{D})$ is an oracle that upon request returns an example $\langle x, c(x)\rangle$ where x is chosen randomly with respect to $\mathcal{D}$. For $\epsilon \geq 0$ we say that function g ϵ-approximates a function f with respect to distribution $\mathcal{D}$ if $\mathbf{Pr}_{\mathcal{D}}[f(x) = g(x)] \geq 1-\epsilon$. For a concept class $\mathcal{C}$, we say that an algorithm $\mathcal{A}$ PAC learns $\mathcal{C}$, if for every $\epsilon > 0$, $c \in \mathcal{C}$, and distribution $\mathcal{D}$ over X, $\mathcal{A}$ given access to $\mathrm{EX}(c,\mathcal{D})$ outputs, with probability at least 1/2, a hypothesis h that ϵ-approximates c. The learning algorithm is *efficient* if it runs in time polynomial in $1/\epsilon$, and the *size* s of the learning problem where the size of the learning problem is equal to the length of an input to c plus the description length of c in the representation associated with $\mathcal{C}$. An algorithm is said to *weakly* learn $\mathcal{C}$ if it produces a hypothesis h that $(\frac{1}{2} - \frac{1}{p(s)})$-approximates (or *weakly approximates*) c for some polynomial p.

The *random classification noise* model introduced by Angluin and Laird formalizes the simplest type of white label noise [AL88]. In this model for any $\eta \leq 1/2$ called the *noise rate* the regular example oracle $\mathrm{EX}(c,\mathcal{D})$ is replaced with the noisy oracle $\mathrm{EX}^{\eta}(c,\mathcal{D})$. On each call, $\mathrm{EX}^{\eta}(c,\mathcal{D})$, draws x according to $\mathcal{D}$, and returns $\langle x, c(x)\rangle$ with probability

$1-\eta$ and $\langle x, \neg c(x)\rangle$ with probability η. When η approaches $1/2$ the label of the corrupted example approaches the result of a random coin flip, and therefore the running time of algorithms in this model is allowed to polynomially depend on $\frac{1}{1-2\eta}$.

The *agnostic* PAC learning model was introduced by Haussler [Hau92] and Kearns *et al.* [KSS94] in order to relax the assumption that examples are labeled by a concept from a specific concept class. In this model no assumptions are made on the function that labels the examples. In other words, the learning algorithm has no prior beliefs about the target concept (and hence the name of the model). The goal of the agnostic learning algorithm for a concept class $\mathcal{C}$ is to produce a hypothesis $h \in \mathcal{C}$ whose error on the target concept is close to the best possible by a concept from $\mathcal{C}$.

Formally, for two Boolean functions f and h and a distribution $\mathcal{D}$ over the domain, we define $\Delta_{\mathcal{D}}(f,h) = \mathbf{Pr}_{\mathcal{D}}[f \neq h]$. Similarly, for a concept class $\mathcal{C}$ and a function f, define $\Delta_{\mathcal{D}}(f,\mathcal{C}) = \inf_{h\in\mathcal{C}}\{\Delta_{\mathcal{D}}(f,h)\}$. Kearns *et al.* define the agnostic PAC learning model as follows [KSS94].

Definition 6.2.1 *An algorithm $\mathcal{A}$* agnostically *(PAC) learns a concept class $\mathcal{C}$ if for every $\epsilon > 0$, a Boolean function f and distribution $\mathcal{D}$ over X, $\mathcal{A}$, given access to $EX(f,\mathcal{D})$, outputs, with probability at least $1/2$, a hypothesis $h \in \mathcal{C}$ such that $\Delta_{\mathcal{D}}(f,h) \leq \Delta_{\mathcal{D}}(f,\mathcal{C})+\epsilon$. As before, the learning algorithm is* efficient *if it runs in time polynomial in s and $1/\epsilon$.*

One can also consider a more general agnostic learning in which the examples are drawn from an arbitrary distribution over $X \times \{0,1\}$ (and not necessarily consistent with a function). It is easy to verify that our positive result applies to this setting as well (to see this, note that oracles defined in Definition 6.3.1 can simulate this generalized scenario).

The agnostic learning model can also be thought of as a model of adversarial noise. By definition, a Boolean function f differs from some function in $c \in \mathcal{C}$ on $\Delta_{\mathcal{D}}(f,\mathcal{C})$ fraction of the domain (the fraction is measured relative to distribution $\mathcal{D}$). Therefore f can be thought of as c corrupted by noise of rate $\Delta_{\mathcal{D}}(f,\mathcal{C})$. Unlike in the random classification noise model the points on which a concept can be corrupted are unrestricted and therefore we refer to it as *adversarial classification noise*. Note that an agnostic learning algorithm will not necessarily find a hypothesis that approximates c – any other function in $\mathcal{C}$ that differs from f on at most $\Delta_{\mathcal{D}}(f,\mathcal{C}) + \epsilon$ fraction of the domain is acceptable. This way to view the agnostic learning is convenient when the performance of a learning algorithm depends on the rate of disagreement (that is the noise rate).

6.2.2 Fourier Transform

Our main reduction uses Fourier-analytic techniques which were first introduced to computational learning theory by Linial *et al.* [LMN93]. In this context we view Boolean functions as functions $f : \{0,1\}^n \to \{-1,1\}$. All probabilities and expectations are taken with respect to the uniform distribution unless specifically stated otherwise. For a Boolean vector $a \in \{0,1\}^n$ let $\chi_a(x) = (-1)^{a \cdot x}$, where '$\cdot$' denotes an inner product modulo 2, and let $\texttt{weight}(a)$ denote the Hamming weight of a.

We define an inner product of two real-valued functions over $\{0,1\}^n$ to be $\langle f, g \rangle = \mathbf{E}_x[f(x)g(x)]$. The technique is based on the fact that the set of all parity functions $\{\chi_a(x)\}_{a\in\{0,1\}^n}$ forms an orthonormal basis of the linear space of real-valued functions over $\{0,1\}^n$ with the above inner product. This fact implies that any real-valued function f over $\{0,1\}^n$ can be uniquely represented as a linear combination of parities, that is $f(x) = \sum_{a\in\{0,1\}^n} \hat{f}(a)\chi_a(x)$. The coefficient $\hat{f}(a)$ is called Fourier coefficient of f on a and equals $\mathbf{E}_x[f(x)\chi_a(x)]$; a is called the *index* and $\texttt{weight}(a)$ the *degree* of $\hat{f}(a)$. We say that a Fourier coefficient $\hat{f}(a)$ is θ-heavy if $|\hat{f}(a)| \geq \theta$. Let $L_2(f) = \mathbf{E}_x[(f(x))^2]^{1/2}$. Parseval's identity states that

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

6.3 Reduction To Learning Parities with Random Noise

In this section, we describe our reductions from learning of parities with adversarial noise to learning of parities with random noise. We will also show applications of this reduction to learning of DNF and juntas. We start by describing the main technical 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 [Jac97], we call such an algorithm a *weak parity algorithm*.

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

6.3.1 Finding Heavy Fourier Coefficients

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

Definition 6.3.1 *For any function $f : \{0,1\}^n \to [-1,1]$ a probabilistic oracle $\mathbb{O}(f)$ is the oracle that produces samples $\langle x, b\rangle$, where x is chosen randomly and uniformly from $\{0,1\}^n$ and $b \in \{-1,+1\}$ is a random variable with expectation $f(x)$.*

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

Claim 6.3.2 *The statistical distance between the outputs of $\mathbb{O}(f)$ and $\mathbb{O}(g)$ is upper-bounded by $L_2(f-g)$.*

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

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

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

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

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

where the product Aa is performed mod 2.

Lemma 6.3.3 *For the function f_A defined above:*

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

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

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

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

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

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

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

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

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

We will use Lemma 6.3.3 to project f in a way that separates one of its significant Fourier coefficients from the rest. We will do this by choosing A to be a random $m \times n$ matrix for appropriate choice of m.

Lemma 6.3.4 *Let $f : \{0,1\}^n \to [-1,1]$ be any function, and let $s \neq 0^n$ be any vector. Choose A randomly and uniformly from $\{0,1\}^{m \times n}$. With probability at least $2^{-(m+1)}$, the following conditions hold:*

$$\widehat{f_A}(s) = \hat{f}(s) \tag{6.1}$$

$$\sum_{a \in \{0,1\}^n \setminus \{s\}} \widehat{f_A}^2(a) \le L_2^2(f) 2^{-m+1} \tag{6.2}$$

Proof: Event (6.1) holds if $As = 1^m$, which happens with probability 2^{-m}.

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

$$\begin{aligned}
\mathbf{Pr}_v[v \cdot a = 1 \mid v \cdot s = 1] &= 1/2 \\
\text{Therefore,} \quad \mathbf{Pr}_A[Aa = 1^m \mid As = 1^m] &= 2^{-m}
\end{aligned}$$

Whereas for $a = 0^n$, $\mathbf{Pr}_A[Aa = 1^m] = 0$. Hence

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

By Markov's inequality,

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

Thus conditioned on event (6.1), event (6.2) happens with probability at least 1/2. So both events happen with probability at least $2^{-(m+1)}$. □

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

Theorem 6.3.5 *Let $\mathcal{A}$ be an algorithm that learns parities of k variables over $\{0,1\}^n$ for every noise rate $\eta < 1/2$ in time $T(n,k,\frac{1}{1-2\eta})$ using at most $S(n,k,\frac{1}{1-2\eta})$ examples. Then there exists an algorithm* `WP-R` *that for every function $f : \{0,1\}^n \to [-1,1]$ that has a θ-heavy Fourier coefficient s of degree at most k, given access to $\mathbb{O}(f)$, with probability at least $1/2$, finds s. Furthermore,* `WP-R` *runs in time $O(T(n,k,1/\theta)\cdot S^2(n,k,1/\theta))$ and uses $O(S^3(n,k,1/\theta))$ random examples.*

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

1. Let $m = \lceil 2\log S\rceil + 3$. Let $A \in \{0,1\}^{m\times n}$ be a randomly chosen matrix and $\mathbb{O}(f_A)$ be the oracle for A-projection of f. Run the algorithm $\mathcal{A}$ on $\mathbb{O}(f_A)$.
2. If $\mathcal{A}$ stops in $T(n,k,1/\theta)$ steps and outputs r with $\texttt{weight}(r) \leq k$, check that r is at least $\theta/2$-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 $\chi_s(x)$.

By Lemma 6.3.4, in Step 1, with probability at least 2^{-m-1} , we create a function f_A such that $|\widehat{f_A}(s)| \geq \theta$ and

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

By Claim 6.3.2, the statistical distance between the oracle $\mathbb{O}(f_A)$ and oracle $\mathbb{O}(\widehat{f_A}(s)\chi_s(x))$ is bounded by

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

hence this distance is small. Since $\mathcal{A}$ uses at most S samples, with probability at least $\frac{1}{2}$, it will not notice the difference between the two oracles. But $\mathbb{O}(\widehat{f_A}(s)\chi_s(x))$ is exactly the noisy parity χ_s with noise rate $1/2 - \widehat{f_A}/2$. If $\widehat{f_A} \geq \theta$ we will get a parity with $\eta \leq 1/2 - \theta/2 < 1/2$ and otherwise we will get a negation of χ_s with $\eta \leq 1/2 - \theta/2$. Hence we get $(1-2\eta)^{-1} \leq 1/\theta$, so the algorithm $\mathcal{A}$ will learn the parity s when executed either with the oracle $\mathbb{O}(f_A)$ or its negation. We can check that the coefficient produced by $\mathcal{A}$ is indeed heavy using Chernoff bounds (see Lemma 2.3.1), and repeat until we succeed. Using $O(2^m) = O(S^2)$ repetitions, we will get a $\theta/2$-heavy Fourier coefficient of degree k with probability at least $1/2$. A-projection always clears the coefficient $\hat{f}(0^n)$ and therefore we need to check whether this coefficient is θ-heavy separately. $\square$

Remark 6.3.6 *A function f can have at most $L_2^2(f)/\theta^2$ θ-heavy Fourier coefficients. Therefore by repeating* `WP-R` *$O((L_2^2(f)/\theta^2) \cdot \log(L_2(f)/\theta)) = \tilde{O}(L_2^2(f)/\theta^2)$ times we can, with high probability, obtain all the θ-heavy Fourier coefficients of f as it is required in some applications of this algorithm.*

6.4 Applications

In this section we will demonstrate a number of applications of Theorem 6.3.5.

6.4.1 Learning of Parities with Adversarial Noise

A weak parity algorithm is in its essence an algorithm for learning of parities with adversarial noise. In particular, Theorem 6.3.5 gives the following reduction from adversarial to random noise.

Theorem 6.4.1 *The problem of learning parities with adversarial noise of rate $\eta < \frac{1}{2}$ reduces to learning parities with random noise of rate η.*

Proof: Let f be a parity χ_s corrupted by noise of rate η. Then $\hat{f}(s) = \mathbf{E}[f\chi_s] \geq (1-\eta) + (-1)\eta = 1 - 2\eta$. Now apply the reduction from Theorem 6.3.5 setting $k = n$. We get an oracle for the function $\hat{f}(s)\chi_s(x)$, which is $\chi_s(x)$ with random noise of level η. $\square$

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

Lemma 6.4.2 ([BKW03]) *Parity functions on $\{0,1\}^n$ can be learned in time and sample complexity $2^{O(\frac{n}{\log n})}$ in the presence of random noise of rate η for any constant $\eta < \frac{1}{2}$.*

This algorithm together with Theorem 6.4.1 gives Theorem 6.1.1.

One can also interpret Theorem 6.4.1 in terms of coding theory problems. Learning a parity function with noise is equivalent to decoding a random linear code from the same type of noise (see Section 5.3 for more details on this equivalence). More formally, we say that a code C is an $[m, n]$ code if C is a binary linear code of block length m and message length n. Any such code can be described by its $n \times m$ *generator matrix* G as follows: $C = \{xG \mid x \in \{0,1\}^n\}$. A *random linear* $[m, n]$ *code* C is produced by choosing randomly and uniformly a generator matrix G for C (that is, each element of G equals to the outcome of an unbiased coin flip). It is now easy to verify that Theorem 6.3.5 implies the following result.

Theorem 6.4.3 *Assume that there exists an algorithm* `RandCodeRandError` *that corrects a random linear* $[m, n]$ *code from random errors of rate* η *with probability at least* $1/2$ *(over the choice of the code, errors, and the random bits of the algorithm) in time* $T(m, n)$*. Then there exists an algorithm* `RandCodeAdvError` *that corrects a random linear* $[M, n]$ *code from up to* $\eta \cdot M$ *errors with probability at least* $1/2$ *(over the choice of the code and the random bits of the algorithm) in time* $O(m^2 \cdot T(m, n))$ *for* $M = O(m^3)$*.*

The sample bounds in Theorem 6.3.5 correspond to the block length of linear codes. Note that for $\eta \geq 1/4$, there might be more than one codeword within the relative distance η. In this case, by repetitively using `RandCodeAdvError` as in Remark 6.3.6, we can list-decode the random code.

6.4.2 Learning DNF Expressions

Jackson's celebrated result gives a way to use a weak parity algorithm and Freund's boosting algorithm [Fre95] to build an algorithm for learning DNF expressions with respect to the uniform distribution [Jac97]. His approach can be adapted to our setting. We give an outline of the algorithm and omit the now-standard analysis.

We view a probability distribution $\mathcal{D}$ 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 6.4.4 ([BF02]) *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* $|\mathbf{E}_{\mathcal{D}}[f\chi_a]| \geq \frac{1}{2s+1}$ *and*

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

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 6.3.5, 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 6.4.5 *Let $\mathcal{A}$ be an algorithm that learns parities of k variables over $\{0,1\}^n$ for every noise rate $\eta < 1/2$ in time $T(n,k,\frac{1}{1-2\eta})$ using at most $S(n,k,\frac{1}{1-2\eta})$ examples. Then there exists an algorithm* `WP-R'` *that for every real-valued function ϕ that has a θ-heavy Fourier coefficient s of degree at most k, given access to random uniform examples of ϕ, finds s in time $O(T(n,k,L_\infty(\phi)/\theta)\cdot S(n,k,L_\infty(\phi)/\theta)^2)$ with probability at least* $1/2$.

The running time of `WP-R'` depends on $L_\infty(2^n\mathcal{D})$ (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 [Fre95] 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 $\tilde{O}(\epsilon)$. By using `WP-R'` with this boosting algorithm in the same way as in Jackson's DNF learning algorithm, we obtain Theorem 6.1.2.

6.4.3 Learning Juntas

For the class of k-juntas, we can get a simpler reduction with better parameters for noise. Since there are at most 2^k non-zero coefficients and each of them is at least 2^{-k+1}-heavy, for a suitable choice of m, the projection step is likely to isolate just one of them. This leaves us with an oracle $\mathbb{O}(\hat{f}(s)\chi_s)$. Since $|\hat{f}(s)| \geq 2^{-k+1}$, the noise parameter is bounded by $\eta < 1/2 - 2^{-k}$. Using Remark 6.3.6, we will obtain the complete Fourier spectrum of f by repeating the algorithm $O(k2^{2k})$ times. The proof of Theorem 6.1.3 follows from these observations. Instead of repeating `WP-R` one can also use a simple recursive procedure of Mossel *et al.* [MOS04, Sec 3.1] that requires only k invocations of `WP-R`.

6.4.4 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 6.4.6 *Let $\mathcal{A}$ be an algorithm that learns parities of k variables on $\{0,1\}^n$ for every noise rate $\eta < 1/2$ in randomized time $T(n, k, \frac{1}{1-2\eta})$. Then there exists an algorithm that learns k-juntas with random noise of rate η' in time $O(k2^{2k} \cdot T(n, k, \frac{2^{k-1}}{1-2\eta'}))$.*

A noisy parity of k variables is a special case of a k-junta. Thus we have reduced the noisy junta problem to a special case viz. noisy parity, at the cost of an increase in the noise level.

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 (learnability of DNF from noisy membership queries was established by Jackson *et al.* [JSS97]). Nevertheless, as we have shown in Section 5.7.1, 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 6.4.7 *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 $\tilde{O}(\frac{s^4}{\epsilon^2} \cdot T(n, \log B, \frac{B}{1-2\eta'}) \cdot S(n, \log B, \frac{B}{1-2\eta'})^2)$ where $B = \tilde{O}(s/\epsilon)$.*

Chapter 7

Hardness of Proper Agnostic Learning of Monomials

In this chapter we study the learnability of monomials (or conjunctions) in the agnostic learning framework of Haussler [Hau92] and Kearns *et al.* [KSS94]. We show that for any constant ϵ, finding a monomial that agrees with an unknown function on $1/2+\epsilon$ fraction of examples is NP-hard even when there exists a monomial that agrees with the unknown function on $1-\epsilon$ fraction of examples. This implies that even weak agnostic learning of monomials is NP-hard, resolving an open question of Blum [Blu98]. This hardness result is optimal and significantly improves on a number of previous results for this problem.

7.1 Introduction

We study the computational complexity of approximation problems arising in *agnostic learning* of monomials. The agnostic framework [Hau92, KSS94] is a useful variant of Valiant's PAC learning model in which, informally, nothing is known about the target function and a learning algorithm is required to do nearly as well as is possible using hypotheses from a given class. Haussler's work [Hau92] implies that learnability in this model is, in a sense, equivalent to the ability to come up with a member of the hypothesis class that has close-to-the-optimal agreement rate with the given examples.

For a number of concept classes it is known that finding a hypothesis with the best agreement rate is NP-hard [AL88, HvHS95, KL93]. However, for most practical purposes a hypothesis with agreement rate close to the maximum would be sufficient. This reduces agnostic learning of a function class to a natural combinatorial approximation problem or, more precisely, to the following two problems: approximating the maximum agreement rate and the minimum disagreement rate. We address the approximation complexity of these problems for the class of monomials (also referred to as terms). The class of monomials is one of the simplest and most well-studied function classes easily learnable in a variety of settings. Angluin and Laird proved that finding a monotone monomial with the maximum agreement rate (this problem is denoted MMon-MA) is NP-hard [AL88]. This was extended to general monomials by Kearns and Li [KL93] (the problem is denoted Mon-MA). Ben-David *et al.* gave the first inapproximability result for this problem, proving that the maximum agreement rate is NP-hard to approximate within a factor of $\frac{770}{767} - \epsilon$ for any constant $\epsilon > 0$ [BDEL03]. This result was more recently improved by Bshouty and Burroughs to the inapproximability factor of $\frac{59}{58} - \epsilon$ [BB06].

The problem of approximating the minimum disagreement with a monomial (denoted Mon-MD) was first considered by Kearns *et al.* who give an approximation preserving reduction from the SET-COVER problem to Mon-MD (similar result was also obtained by Hoffgen *et al.* [HvHS95]). This reduction together with the hardness of approximation results for SET-COVER due to Lund and Yannakakis [LY94] (see also [RS97]) implies that Mon-MD is NP-hard to approximate within a factor of $c \log n$ for some constant c.

On the positive side, the only non-trivial approximation algorithm is due to Bshouty and Burroughs and achieves $2 - \frac{\log n}{n}$-approximation for the agreement rate [BB06]. Note that factor 2 can always be achieved by either constant 0 or constant 1 function.

In this work, we give the following inapproximability results for Mon-MA.

Theorem 7.1.1 *For every constant $\epsilon > 0$, Mon-MA is NP-hard to approximate within a factor of $2 - \epsilon$.*

Then, under a slightly stronger assumption, we show that the second order term is small.

Theorem 7.1.2 *For any constant $\lambda > 0$, there is no polynomial-time algorithm that approximates Mon-MA within a factor of $2 - 2^{-\log^{1-\lambda} n}$, unless $\mathsf{NP} \subseteq \mathsf{RTIME}(n^{poly \log(n)})$.*

Theorem 7.1.2 also implies strong hardness results for Mon-MD.

Corollary 7.1.3 *For any constant $\lambda > 0$, there is no polynomial time algorithm that approximates Mon-MD within a factor of $2^{\log^{1-\lambda} n}$, unless $\mathsf{NP} \subseteq \mathsf{RTIME}(n^{poly \log(n)})$.*

In practical terms, these results imply that even very low (sub-constant) amounts of adversarial noise in the examples make finding a term with agreement rate larger (even by very small amount) than 1/2, NP-hard, in other words even *weak agnostic learning* of monomials is NP-hard. This resolves an open problem due to Blum [Blu98, Blu03b].

All of our results hold for the MMon-MA problem as well. A natural equivalent formulation of the MMon-MA problem is maximizing the number of satisfied *monotone disjunction constraints*, that is, equations of the form $t(x) = b$, where $t(x)$ is a disjunction of (unnegated) variables and $b \in \{0,1\}$. We denote this problem by MAX-B-MSAT where B is the bound on the number of variables in each disjunction (see Definition 7.2.5 for more details). A corollary of our hardness result for MMon-MA is the following theorem

Theorem 7.1.4 *For any constant ϵ, there exists a constant B such that* MAX-B-MSAT *is* NP*-hard to approximate within* $2-\epsilon$*.*

This result gives a form of the PCP theorem with imperfect completeness.

Finally, we show that Theorems 7.1.1 and 7.1.2 can be easily used to obtain hardness of agnostic learning results for classes richer than monomials, thereby improving on several known results and establishing hardness of agreement max/minimization for new function classes.

It is important to note that our results do not rule out agnostic learning of monomials when the disagreement rate is very low (i.e. $2^{-\log^{1-o(1)} n}$), weak agnostic learning with agreement lower than $1/2 + 2^{-\log^{1-o(1)} n}$, or non-proper agnostic learning of monomials.

Our proof technique is based on using Feige's multi-prover proof system for 3SAT-5 (3SAT with each variable occurring in exactly 5 clauses) together with set systems possessing a number of specially-designed properties. The set systems are then constructed by a simple probabilistic algorithm. As in previous approaches, our inapproximability results are eventually based on the PCP theorem. However, previous results reduced the problem to an intermediate problem (such as MAX-CUT, MAX-E2-SAT, or SET-COVER) thereby substantially losing the generality of the constraints. We believe that key ideas of our technique might be useful in dealing with other constraint satisfaction problems involving constraints that are conjunctions or disjunctions of Boolean variables.

7.1.1 Related Work

Besides the results for monomials mentioned earlier, hardness of agnostic learning results are known for a number of other classes. Optimal hardness results are known for the class of parities. Håstad proved that approximating agreements with parities within a factor of $2-\epsilon$ is NP-hard for any constant ϵ. Amaldi and Kann [AK95a], Ben-David *et al.* [BDEL03], and Bshouty and Burroughs [BB06] prove hardness of approximating agreements with halfspaces (factors $\frac{262}{261}$, $\frac{418}{415}$, and $\frac{85}{84}$, respectively). Similar inapproximability results are also known for 2-term DNF, decision lists and balls [BDEL03, BB06]. Following this work, Feldman *et al.* proved that approximating agreements with halfspaces over $\mathbb{R}^n$ within a factor of $2-\epsilon$ is NP-hard for any constant ϵ [FGKP06], and, independently, Guruswami and Raghavendra established the same result for halfspaces over the Boolean hypercube [GR06].

Arora *et al.* give strong inapproximability results for minimizing disagreements with halfspaces (factor $2^{\log^{0.5-\epsilon} n}$) and with parities[1] (factor $2^{\log^{1-\epsilon} n}$) under the assumption that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{poly \log n})$. Bshouty and Burroughs prove inapproximability of minimizing disagreements with k-term multivariate polynomials (factor $\ln n$) and a number of other classes [BB02].

For an extension of the agnostic framework where a learner can output a hypothesis from a richer class of functions (see also Section 7.2.1) the first non-trivial algorithm for learning monomials was recently given by Kalai *et al.* [KKMS05]. Their algorithm learns monomials agnostically in time $2^{\tilde{O}(\sqrt{n})}$. They also gave a breakthrough result for agnostic learning of halfspaces by showing a simple algorithm that for any constant $\epsilon > 0$ agnostically learns halfspaces with respect to the uniform distribution up to ϵ accuracy (both their algorithms output thresholds of parities as hypotheses).

We also note that minimum disagreement cannot be approximated for classes that are known to be not *properly* learnable, i.e., when a hypothesis has to use the same representation as the class being learned (since in the usual PAC model it is assumed that there exists a hypothesis with zero disagreement rate). In particular, the minimum disagreement with various classes of DNF formulae, intersections of halfspaces, decision trees, and juntas cannot be approximated [PV88, ABF$^+$04].

7.2 Preliminaries and Notation

For a vector v, we denote its ith element by v_i (unless explicitly defined otherwise). For a positive integer m we denote $[m] = \{1, 2, \ldots, m\}$.

The domain of all discussed Boolean functions is the Boolean hypercube $\{0,1\}^n$. The ith literal is a function over $\{0,1\}^n$ equal to the i-th coordinate of a point and denoted x_i, or its negation, denoted $\bar{x}_i$. A *monomial* is a conjunction of literals and/or constants (0 and 1). It is also commonly referred to as a conjunction. A monotone monomial is a monomial that includes only positive literals and constants. We denote the concept class of all monomials by Mon and the class of all monotone monomials by MMon. A DNF formula is a disjunction of terms and a k-term DNF formula is a disjunction of k terms. A *halfspace* or a *threshold* function is a function equal to $\sum_{i \in [n]} w_i x_i \geq \theta$ (as a Boolean expression), where $w_1, \ldots, w_k, \theta$ are integers.

[1]This problem is more commonly known as "finding the nearest codeword".

7.2.1 The Problem

Our model of learning is the agnostic PAC learning model [Hau92, KSS94] described in Section 2.2.4. Here we briefly review some of the relevant definitions.

For two Boolean functions f and h and a distribution $\mathcal{D}$ over $\{0,1\}^n$ we define $\Delta_{\mathcal{D}}(f,h) = \mathbf{Pr}_{\mathcal{D}}[f \neq h]$. Similarly, for a class of Boolean functions $\mathcal{C}$ and a function f define $\Delta_{\mathcal{D}}(f,\mathcal{C}) = \min_{h\in\mathcal{C}}\{\Delta_{\mathcal{D}}(f,h)\}$.

Definition 7.2.1 *An algorithm $\mathcal{A}$* agnostically *(PAC) learns a concept class $\mathcal{C}$ if for every $\epsilon > 0$, a Boolean function f and distribution $\mathcal{D}$ over X, $\mathcal{A}$, given access to $EX(f,\mathcal{D})$, outputs, with probability at least $1/2$, a hypothesis $h \in \mathcal{C}$ such that $\Delta_{\mathcal{D}}(f,h) \leq \Delta_{\mathcal{D}}(f,\mathcal{C})+\epsilon$. As usual, the learning algorithm is* efficient *if it runs in time polynomial in n and $1/\epsilon$.*

One can also consider a more general agnostic learning in which the examples are drawn from an arbitrary distribution over $X \times \{0,1\}$ (and not necessarily consistent with a function). In Remark 7.2.4 we prove that learning in this more general setting is not harder than in the agnostic model defined above.

In this work we will deal with samples of fixed size instead of random examples generated with respect to some distribution. One can easily see that these settings are essentially equivalent. In one direction, given an agnostic learning algorithm and a sample S we can just run the algorithm on examples chosen randomly and uniformly from S, thereby obtaining a hypothesis with the disagreement rate on S equal to the error guaranteed by the agnostic learning algorithm. For the other direction, one can use uniform convergence results for agnostic learning given by Haussler [Hau92] (based on the earlier work in statistical learning theory). They state that for every $c \in \mathcal{C}$ and sample S of size poly(VC-dim$(\mathcal{C}), \epsilon$) randomly drawn with respect to a distribution $\mathcal{D}$, with high probability the true error of c will be within ϵ of the disagreement rate of c on S. Monomials over $\{0,1\}^n$ have VC dimension n and therefore we can, without loss of generality, restrict our attention to algorithms that operate on samples of fixed size.

Besides algorithms with this strong agnostic guarantee it is natural and potentially useful to consider algorithms that output hypotheses with weaker yet non-trivial guarantees (e.g. having error of at most twice the optimum or within an additive constant of the optimum). According to the uniform convergence results, these weaker forms of agnostic learning are equivalent to approximating the maximum agreement rate or the minimum disagreement rate of a function in $\mathcal{C}$ with a given sample. We now proceed to define the problems more formally.

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

Definition 7.2.2 *For a class of functions $\mathcal{C}$ and domain X, we define the* Maximum Agreement *problem* $\mathcal{C}$-MA *as follows: The input is a set of examples $S \subseteq X \times \{0, 1\}$. The problem is to find a function $h \in \mathcal{C}$ such that $\texttt{AgreeR}(h, S) = \texttt{AgreeR}(\mathcal{C}, S)$.*

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

An extension of the original agnostic learning framework is the model in which a hypothesis may come from a richer class $\mathcal{H}$. The corresponding combinatorial problems were introduced by Bshouty and Burroughs and are denoted $\mathcal{C}/\mathcal{H}$-MA and $\mathcal{C}/\mathcal{H}$-MD [BB06]. Note that an approximation algorithm for these problems can return a value larger than $\texttt{AgreeR}(\mathcal{C}, S)$ and therefore cannot be used to approximate the value $\texttt{AgreeR}(\mathcal{C}, S)$.

Remark 7.2.3 *An α-approximation algorithm for $\mathcal{C}'$-MA(MD) where $\mathcal{C} \subseteq \mathcal{C}' \subseteq \mathcal{H}$ is an α-approximation algorithm for $\mathcal{C}/\mathcal{H}$-MA(MD).*

Remark 7.2.4 *A more general agnostic learning in which random examples are not necessarily consistent with a function corresponds to agreement maximization and disagreement minimization over samples that might contain contradicting examples (i.e. $\langle x, 0\rangle$ and $\langle x, 1\rangle$). We remark that such contradicting examples do not make the problems of agreement maximization and disagreement minimization harder. To see this, let S be a sample and let S' be S with all contradicting pairs of examples removed (that is for each example $\langle x, 0\rangle$ we remove it together with one example $\langle x, 1\rangle$). Every function has the same agreement rate of $1/2$ with examples in $S \setminus S'$. Therefore for any concept class $\mathcal{C}$, $\texttt{AgreeR}(\mathcal{C}, S) = \gamma \cdot \texttt{AgreeR}(\mathcal{C}, S') + (1-\gamma)/2$, where $\gamma = |S'|/|S|$. Therefore agreement maximization (or disagreement minimization) over S is equivalent to agreement maximization (or disagreement minimization) over S'. The same also holds for approximate agreement maximization and disagreement minimization. This is true since for every function h and $\alpha \geq 1$, if $\alpha \cdot \texttt{AgreeR}(h, S') \geq \texttt{AgreeR}(\mathcal{C}, S')$ then*

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

(and similarly for approximate disagreement minimization). Therefore if an agreement maximization or disagreement minimization problem is hard for samples with contradictions then its hard for samples without contradictions and hence the corresponding version of agnostic learning is hard.

7.2.2 Agreement with Monomials and Set Covers

For simplicity we first consider the MMon-MA problem. The standard reduction of the general to the monotone case [KLPV87a] implies that this problem is at least as hard to approximate as Mon-MA. We will later observe that our proof will hold for the unrestricted case as well. We start by giving two equivalent ways to formulate MMon-MA.

Definition 7.2.5 *The Maximum Monotone Disjunction Constraints problem* MAX-MSAT *is defined as follows: The input is a set C of* monotone disjunction constraints*, that is, equations of the form $d(x) = b$ where, $d(x)$ is a monotone disjunction and $b \in \{0,1\}$. The output is a point $z \in \{0,1\}^n$ that maximizes the number of satisfied equations in C. For an integer function B,* MAX-B-MSAT *is the same problem with each disjunction containing at most B variables.*

Lemma 7.2.6 MMon-*MA is equivalent to* MAX-MSAT.

Proof: For a vector $v \in \{0,1\}^n$, let d_v denote the monotone disjunction equal to $\vee_{v_i=1} x_i$. Furthermore for a Boolean vector $v \in \{0,1\}^n$, we denote by $\bar{v}$ the bitwise negation of v. We claim that a monotone disjunction constraint $d_v = b$ is equivalent to example $\langle \bar{v}, \bar{b} \rangle$ in an instance of MMon-MA. To show this, we prove that a point $z \in \{0,1\}^n$ satisfies $d_v = b$ if and only if monomial $c_z = \wedge_{z_i=1} x_i$ is consistent with example $\langle \bar{v}, \bar{b} \rangle$. This is true since z satisfies $d_v = 0$ if and only if for all $i \in [n]$, $z_i = 1$ implies $v_i = 0$. This is equivalent to saying that $\wedge_{z_i=1} \bar{v}_i = 1$ or c_z is consistent with example $\langle \bar{v}, 1 \rangle$. Similarly, z satisfies $d_v = 1$ if and only if c_z is consistent with example $\langle \bar{v}, 0 \rangle$. □

Another equivalent way to formulate MMon-MA is the following.

Definition 7.2.7 *We define the* Balanced Set Cover *problem or BAL-SET-COVER as follows.*
Input: $\mathcal{S} = (S^+, S^-, \{S_i^+\}_{i \in [n]}, \{S_i^-\}_{i \in [n]})$ *where* $S_1^+, \ldots, S_n^+ \subseteq S^+$ *and* $S_1^-, \ldots, S_n^- \subseteq S^-$.
Output: *A set of indices I that maximizes the sum of two values,* $\texttt{Agr}^-(\mathcal{S}, I) = |\bigcup_{i \in I} S_i^-|$ *and* $\texttt{Agr}^+(\mathcal{S}, I) = |S^+| - |\bigcup_{i \in I} S_i^+|$. *We denote this sum by* $\texttt{Agr}(\mathcal{S}, I) = \texttt{Agr}^-(\mathcal{S}, I) + \texttt{Agr}^+(\mathcal{S}, I)$ *and denote the maximum value of agreement by* $\texttt{MMaxAgr}(\mathcal{S})$.

Lemma 7.2.8 MMon-*MA is equivalent to BAL-SET-COVER.*

Proof: Let S be a set of examples. We define an instance $\mathcal{S} = (S^+, S^-, \{S_i^+\}_{i \in [n]}, \{S_i^-\}_{i \in [n]})$ of BAL-SET-COVER as follows. Let $S^- = \{x \mid \langle x, 0 \rangle \in S\}$ and $S^+ = \{x \mid \langle x, 1) \rangle \in S\}$ (note that this definition is consistent with notation in Section 7.2.1). Now let $S_i^- = \{x \mid x \in S^- \text{ and } x_i = 0\}$ and $S_i^+ = \{x \mid x \in S^+ \text{ and } x_i = 0\}$.

Then for any set of indices $I \subseteq [n]$, the monotone monomial $t_I = \wedge_{i\in I} x_i$ is consistent with all the examples in S^- that have a zero in at least one of the coordinates with indices in I, that is, with examples in $\bigcup_{i\in I} S_i^-$. It is also consistent with all the examples in S^+ that do not have zeros in coordinates with indices in I, that is, $S^+ \setminus \bigcup_{i\in I} S_i^+$. Therefore the number of examples with which t_I agrees is exactly $\mathtt{Agr}(\mathcal{S}, I)$.

For the other direction, let $\mathcal{S} = (S^+, S^-, \{S_i^+\}_{i\in[n]}, \{S_i^-\}_{i\in[n]})$ be an instance of BAL-SET-COVER. For a point $s \in S^+$, let $y^+(s) \in \{0,1\}^n$ the point such that for all $i \in [n]$, $y^+(s)_i = 0$ if and only if $s \in S_i^+$. For a point $s \in S^-$, we define a point $y^-(s) \in \{0,1\}^n$ analogously. Now let

$$Y_{\mathcal{S}} = \{\langle y^+(s), 1\rangle \mid s \in S^+\} \cup \{\langle y^-(s), 0\rangle \mid s \in S^-\} \ .$$

It is easy to see that this mapping is exactly the inverse of the mapping given in the first direction. This implies that for any monotone monomial $t_I = \wedge_{i\in I} x_i$, $|Y_{\mathcal{S}}| \cdot \mathtt{AgreeR}(t_I, Y_{\mathcal{S}}) = \mathtt{Agr}(\mathcal{S}, I)$. □

Remark 7.2.9 *Lemmas 7.2.6 and 7.2.8 imply that BAL-SET-COVER is equivalent to* MAX-MSAT*. In addition, we claim that if in a given instance $\mathcal{S}$ of BAL-SET-COVER, every point $s \in S^+ \cup S^-$ belongs to almost t subsets, then every clause of the corresponding instance of* MAX-MSAT *has at most t variables (i.e. is an instance of* MAX-t-MSAT*).*

Proof: Every point $s \in S^-$ corresponds to example $\langle y^-(s), 0\rangle$ in the equivalent instance of MMon-MA. Recall that for all $i \in [n]$, $y^-(s)_i = 0$ if and only if $s \in S_i^-$. Now, according to the transformation given in the proof of Lemma 7.2.6, example $\langle \bar{v}, \bar{b}\rangle$ is equivalent to clause constraint $d_v = b$. This implies that point s corresponds to constraint $\vee_{y^-(s)_i=0} x_i = 1$. By the definition of $y^-(s)$, this is equivalent to $\vee_{s\in S_i^-} x_i = 1$. Therefore the clause that corresponds to point s has at most t variables. By the same argument, this also holds for points in S^+. □

In the rest of the discussion it will be more convenient to work with instances of the Balanced Set Cover problem instead of instances of MMon-MA. It is also possible to formulate Mon-MA in a similar fashion. We need to specify an additional bit for each variable that tells whether this variable is negated in the monomial or not (when it is present). Therefore the formulation uses the same input and the following output.
*Output(*Mon-*MA):* A set of indices I and a vector $v \in \{0,1\}^n$ that maximizes the value

$$\mathtt{Agr}(\mathcal{S}, I, v) = |\bigcup_{i\in I} Z_i^-| + |S^+| - |\bigcup_{i\in I} Z_i^+|,$$

where $Z_i^{+/-} = S_i^{+/-}$ if $v_i = 0$ and $Z_i^{+/-} = S^{+/-} \setminus S_i^{+/-}$ if $v_i = 1$. We denote the maximum value of agreement with a general monomial by $\texttt{MaxAgr}(\mathcal{S})$.

7.3 Hardness of Approximating Mon-MA and Mon-MD

It is easy to see that BAL-SET-COVER is similar to the SET-COVER problem. Indeed, our hardness of approximation result will employ some of the ideas from Feige's hardness of approximation result for SET-COVER [Fei98].

7.3.1 Feige's Multi-Prover Proof System

Feige's reduction to the SET-COVER problem is based on a multi-prover proof system for MAX-3SAT-5 [Fei98]. MAX-3SAT-5 is the problem of maximizing the number of satisfied clauses in a given 3CNF-5 formula, that is a CNF formula with each clause containing exactly 3 variables and each variable appearing in exactly 5 clauses. Using the PCP theorem [ALM$^+$98, AS98], Feige shows that MAX-3SAT-5 is NP-hard to approximate. Namely,

Lemma 7.3.1 ([ALM$^+$98, AS98, Fei98]) *There exists a constant $\tau > 0$ for which it is NP-hard to distinguish between satisfiable 3CNF-5 formulas and those in which at most $(1 - \tau)$ fraction of the clauses can be satisfied simultaneously.*

The basis of Feige's proof system is a two-prover protocol for MAX-3SAT-5 in which the verifier chooses a random clause and a random variable in that clause. It then asks for the values of all the variables in the clause from the first prover and for the value of the chosen variable from the second prover. The verifier accepts if the values given by the first prover satisfy the clause and the values of the chosen variable returned by both provers are consistent. It is easy to see, that if the input formula ϕ is satisfiable then there exist provers that will cause the verifier to accept. On the other hand, if at most $1 - \tau$ fraction of ϕ's clauses are (simultaneously) satisfiable, then for every prover, the verifier will accept with probability at most $1 - \tau/3$. The soundness of this proof system can be amplified by performing the test ℓ times and using Raz' parallel repetition theorem [Raz98]. In Feige's proof system the ℓ challenges are distributed to k provers with each prover getting $\ell/2$ clause questions and $\ell/2$ variable questions. This is done using an asymptotically-good code with k codewords of length ℓ and Hamming weight $\ell/2$. Specifically, let $C = \{z^{(1)}, z^{(2)}, \ldots, z^{(k)}\}$ be an asymptotically-good code of length ℓ and Hamming weight $\ell/2$. The prover i gets the j-th clause query for every j such that $z_j^{(i)} = 1$ and gets the j-th variable query for every j such that $z_j^{(i)} = 0$. The verifier accepts if all provers gave answers that satisfy all the clause questions and at least two provers gave consistent answers. Here the fact that C is an asymptotically good code implies that for any $i_1 \neq i_2$, the set $\Delta(i_1, i_2) = \{j \mid z_j^{(i_1)} \neq z_j^{(i_2)}\}$ has size at least $\alpha \cdot \ell$ for some constant α. This implies that in order to produce consistent answers, provers i_1 and i_2 have to answer consistently in at least $|\Delta(i_1, i_2)| \geq \alpha \cdot \ell$ challenges. Simple analysis given by Feige shows that if at most $1 - \tau$ clauses of ϕ can be satisfied then this cannot happen with probability larger than $2^{-c_\tau \ell}$ (for some fixed constant c_τ). In addition, when ϕ is satisfiable there exist provers that always answer consistently.

In this protocol choosing ℓ 3SAT-5 challenges requires $\ell \log(5n)$ random bits. There are $\frac{5}{3}n$ different clauses and an answer to a clause question consists of 3 bits. The answer to a variable question is just a single bit. Therefore the query to each prover has length $\frac{\ell}{2}(\log n + \log(\frac{5}{3}n)) = \ell \log(\sqrt{\frac{5}{3}}n)$ and a response to such a query is of length $\frac{\ell}{2}(3+1) = 2\ell$.

We now summarize the relevant properties of Feige's proof system. For integer k and ℓ such that $\ell \geq c_\ell \log k$ for some fixed constant c_ℓ, Feige's k-prover proof system for MAX-3SAT-5 has the following properties:

1. Given a 3CNF-5 formula ϕ over n variables, verifier V tosses a random string r of length $\ell \log(5n)$ and generates k queries $q_1(r), \ldots q_k(r)$ of length $\ell \log(\sqrt{\frac{5}{3}}n)$.

2. Given answers $a_1, \ldots a_k$ of length 2ℓ from the provers, V computes $V_1(r, a_1), \ldots, V_k(r, a_k) \in [2^\ell]$ for fixed functions $V_1, \ldots, V_k$. These are the functions that choose a single bit from an answer to each clause question. The bit that is chosen depends only on r.

3. V accepts if there exist $i \neq j$ such that $V_i(r, a_i) = V_j(r, a_j)$ and for all i, a_i satisfies the clause questions in query $q_i(r)$.

4. Completeness: if ϕ is satisfiable, then there exist a k-prover $\bar{P}$ for which, with probability 1, $V_1(r, a_1) = V_2(r, a_2) = \cdots = V_k(r, a_k)$ and for all i, a_i satisfies the clause questions in query $q_i(r)$ (note that this is stronger than the acceptance predicate above).

5. Soundness: if at most $1 - \tau$ clauses of ϕ can be satisfied simultaneously, then for any $\bar{P}$, V accepts with probability at most $k^2 2^{-c_\tau \cdot \ell}$ for some fixed constant c_τ.

7.3.2 Balanced Set Partitions

As in Feige's proof, the second part of our reduction is a set system with certain properties tailored to be used with the equality predicate in Feige's proof system. Our set system consists of two main parts. The first part is sets grouped by partitions in a way that sets in the same partition are disjoint (and hence correlated) and sets from different partitions are uncorrelated. The second part of our set system is a collection of uncorrelated smaller sets. Formally, a *balanced set partition* $\mathcal{B}(m, L, M, k, \gamma)$ has the following properties:

1. There is a ground set B of m points.
2. There is a collection of L distinct partitions $p_1, \ldots, p_L$.
3. For $i \in [L]$, partition p_i is a collection of k disjoint sets $B_{i,1}, \ldots, B_{i,k} \subseteq B$ whose union is B.
4. There is a collection of M sets $C_1, \ldots, C_M \subseteq B$.
5. Let $\rho_{s,t} = 1 - (1 - \frac{1}{k^2})^s (1 - \frac{1}{k})^t$. For any $I \subseteq [M]$ and $J \subseteq [L] \times [k]$ with all elements having different first coordinate (corresponding to partition number), it holds
$$\left| \frac{\left| \left(\bigcup_{i \in I} C_i\right) \cup \left(\bigcup_{(i,j) \in J} B_{i,j}\right) \right|}{m} - \rho_{|I|,|J|} \right| \leq \gamma \ .$$

Property 5 of a balanced set partition implies that each set $B_{i,j}$ has size approximately m/k and each set C_i has size approximately m/k^2. In addition, this property implies that any union of s different C_i's and t $B_{i,j}$'s from distinct partitions has size approximately equal to $\rho_{s,t} m$. Here $\rho_{s,t} m$ is the expected size of this union if all elements of C_i's and $B_{i,j}$'s were chosen randomly and independently.

To see why balanced set partitions are useful in proving hardness of approximating BAL-SET-COVER, consider an instance $\mathcal{S}$ of BAL-SET-COVER defined as follows. For $\mathcal{B}(m, L, M, k, \gamma)$ as above, let $S^+ = S^- = B$, $S^-_{j,i} = B_{j,i}$, and $S^+_{j,i} = B_{j,1}$ (note that for all i, $S^+_{j,i} = S^+_{j,1}$). Now for any $j \in [L]$, let $I_j = \{(j, i) \mid i \in [k]\}$ be index set of all sets in partition j. Then
$$\mathtt{Agr}(\mathcal{S}, I_j) = \left| \bigcup_{i \in [k]} B_{j,i} \right| + |B| - |B_{j,1}| \ .$$

According to property 3 of $\mathcal{S}$, $\bigcup_{i \in [k]} B_{j,i} = B$ while according to property 5, $|B_{j,1}| \leq (\frac{1}{k} + \gamma)m$. This implies that $\mathtt{Agr}(\mathcal{S}, I_j) \geq (2 - \frac{1}{k} - \gamma)m$. On the other hand, for any index set I that does not include sets from the same partition, we have that
$$\mathtt{Agr}(\mathcal{S}, I) = \left| \bigcup_{(j,i) \in I} B_{j,i} \right| + |B| - \left| \bigcup_{(j,i) \in I} B_{j,1} \right| \ .$$

By property 5 of balanced set partitions, $|(\bigcup_{(j,i) \in I} B_{j,i})| \leq (\rho_{0,|I|} + \gamma)m$, and similarly, $|(\bigcup_{(j,i) \in I} B_{j,1})| \geq (\rho_{0,|I|} - \gamma)m$. Therefore $\mathtt{Agr}(\mathcal{S}, I) \leq (1 + 2\gamma)m$. For sufficiently large k

and sufficiently small γ, this creates a multiplicative gap of $2 - \epsilon$ between the two cases. To relate this to Feige's proof system we design a reduction in which consistent answers from provers corresponds to choosing sets from the same partition, while inconsistent answers correspond to choosing sets from distinct partitions. Then properties 3,4 and 5 of Feige's proof system would imply a multiplicative gap of $2 - \epsilon$ for the instances of BAL-SET-COVER that will be created by the reduction.

In our reduction to each set $S^+_{j,i}$ and $S^-_{j,i}$ we add a single set C_k for some $k \in [M]$. Adding these smaller sets does not substantially influence the agreement rate if the size of I is relatively small. But if the size of I is large then these small sets will cover both S^+ and S^- resulting in $\mathtt{Agr}(\mathcal{S}, I) \leq (1 + 2\gamma)m$. This property of index sets will be required to convert an index set with high agreement rate to a good provers' strategy. In this sense, the addition of smaller sets is analogous to the use of the random skew in Håstad's long code test [Has01].

7.3.3 Creating Balanced Set Partitions

In this section, we show a straightforward randomized algorithm that produces balanced set partitions.

Theorem 7.3.2 *There exists a randomized algorithm that on input k, L, M, γ produces a balanced set partition $\mathcal{B}(m, L, M, k, \gamma)$ for $m = \tilde{O}(k^2\gamma^{-2}\log(M + L))$ in time $O((M + L)m)$.*

Proof: First we create the sets $B_{j,i}$. To create each partition $j \in [L]$, we roll m k-sided dice and denote the outcomes by $d_1, \ldots, d_m$. Set $B_{j,i} = \{r \mid d_r = i\}$. This clearly defines a collection of disjoint sets whose union is $[m]$. To create M sets $C_1, \ldots, C_M$, for each $i \in [M]$ and each $r \in [m]$, we include r in C_i with probability $\frac{1}{k^2}$.

Now let $I \subseteq [M]$ and $J \subseteq [L] \times [k]$ be a set of indices with different first coordinate (corresponding to sets from different partitions) and let $U = \left(\bigcup_{i \in I} C_i\right) \bigcup \left(\bigcup_{(i,j) \in J} B_{i,j}\right)$. Elements of these sets are chosen independently and therefore for each $r \in [m]$,

$$\mathbf{Pr}[r \in U] = 1 - (1 - \frac{1}{k^2})^{|I|}(1 - \frac{1}{k})^{|J|} = \rho_{|I|,|J|}$$

independently of other elements of $[m]$. Using Chernoff bounds, we get that for any $\delta > 0$,

$$\mathbf{Pr}\left[\left|\frac{|U|}{m} - \rho_{|I|,|J|}\right| > \delta\right] \leq 2e^{-2m\delta^2} \ ,$$

which is exactly the property 5 of balanced set partitions (for $\delta = \gamma$). Our next step is to ensure that property 5 holds for all possible index sets I and J. This can be done by first observing that it is enough to ensure that this condition holds for $\delta = \gamma/2$, $|I| \leq k^2 \ln \frac{1}{\delta}$ and $|J| \leq k \ln \frac{1}{\delta}$. This is true since for $|I| \geq k^2 \ln \frac{1}{\delta}$ and every t, $\rho_{|I|,t} \geq 1 - \delta$. Therefore $|U|/m - \rho_{|I|,t} \leq 1 - \rho_{|I|,t} \leq \delta < \gamma$. For the other side of the bound on the size of the union, let I' be a subset of I of size $k^2 \ln \frac{1}{\delta}$ and U' be the union of sets with indices in I' and J. It then follows that

$$\rho_{|I|,t} - \frac{|U|}{m} \leq 1 - \frac{|U'|}{m} \leq 1 - (\rho_{k^2 \ln \frac{1}{\delta},t} - \delta) \leq 1 - (1-\delta) + \delta = \gamma \ .$$

The second condition, $|J| \leq k \ln \frac{1}{\delta}$, is obtained analogously.

There are at most M^s different index sets $I \subseteq [M]$ of size at most s and at most $(kL)^t$ different index sets J of size at most t. Therefore, the probability that property 5 does not hold is at most

$$((kL)^{k \ln \frac{1}{\delta}} + M^{k^2 \ln \frac{1}{\delta}}) \cdot 2e^{-2m\delta^2} \ .$$

For

$$m \geq 2k^2\gamma^{-2} \cdot \ln(kL + M) \cdot \ln \frac{2}{\gamma} + 2,$$

this probability is less than 1/2. □

We can now proceed to the reduction itself.

7.3.4 Main Reduction

Below we describe our main transformation from Feige's proof system to BAL-SET-COVER. To avoid confusion we denote the number of variables in a given 3CNF-5 formula by d and use n to denote the number of sets in the produced BAL-SET-COVER instance (that corresponds to the number of variables in the equivalent instance of MMon-MA).

Theorem 7.3.3 *For every $\epsilon > 0$ (not necessarily constant), there exists an algorithm $\mathcal{A}$ that given a 3CNF-5 formula ϕ over d variables, produces an instance $\mathcal{S}(\phi)$ of BAL-SET-COVER on base sets S^+ and S^- of size T such that*

1. *$\mathcal{A}$ runs in time $d^{O(\ell)}$ plus the time to create a balanced set partition $\mathcal{B}(m, 2^\ell, 4^\ell, \frac{4}{\epsilon}, \frac{\epsilon}{8})$, where $\ell = c_1 \log \frac{1}{\epsilon}$ for some constant c_1, and $m = \tilde{O}(\epsilon^{-4})$ is the size of the ground set of the balanced set partition.*

2. *$|S^+| = |S^-| = T = (5d)^\ell m$.*

3. *$n \leq \frac{4}{\epsilon}(6d)^\ell$.*

4. *If ϕ is satisfiable, then $\texttt{MMaxAgr}(\mathcal{S}(\phi)) \geq (2-\epsilon)T$.*

5. *If at most $1-\tau$ clauses of ϕ can be satisfied simultaneously, then* $|\texttt{MMaxAgr}(\mathcal{S}(\phi)) - T| \leq \epsilon \cdot T$.

Proof: Let $k = \frac{4}{\epsilon}$, $\gamma = \epsilon/8$, and V be Feige's verifier for MAX-3SAT-5. Given ϕ, we construct an instance $\mathcal{S}(\phi)$ of BAL-SET-COVER as follows. Let R denote the set of all possible random strings used by V, let Q_i denote the set of all possible queries to prover i and let $A_i(\phi, q) \subseteq \{0,1\}^{2\ell}$ denote the set of answers of prover i to query q in which all clause questions are satisfied. Let $L = 2^{\ell}$, $M = 2^{2\ell}$, and $\mathcal{B}(m, L, M, k, \gamma)$ be a balanced set partition. We set $S^+ = S^- = R \times B$, and for every $r \in R$ and $B' \subseteq B$, let (r, B') denote the set $\{(r, b) \mid b \in B'\}$. We now proceed to define the sets in $\mathcal{S}(\phi)$. Let $\mathcal{I} = \{(q, a, i) \mid i \in [k], q \in Q_i, a \in A_i(\phi, q)\}$. For $(q, a, i) \in \mathcal{I}$, we define

$$S^-_{(q,a,i)} = \bigcup_{q_i(r)=q} (r, B_{V_i(r,a),i} \cup C_a) \text{ and}$$

$$S^+_{(q,a,i)} = \bigcup_{q_i(r)=q} (r, B_{V_i(r,a),1} \cup C_a) \ .$$

For $r \in R$, let $\mathcal{S}(\phi)_r$ denote the instance of BAL-SET-COVER obtained by restricting $\mathcal{S}(\phi)$ to points with the first coordinate equal to r. We also denote the restrictions of S^- and S^+ to points with the first coordinate equal to r by S^-_r and S^+_r. It is easy to see that for every $I \subseteq \mathcal{I}$, $\texttt{Agr}(\mathcal{S}(\phi), I) = \sum_{r \in R} \texttt{Agr}(\mathcal{S}(\phi)_r, I)$.

Intuitively, sets $S^-_{(q,a,i)}$ (or $S^+_{(q,a,i)}$) correspond to prover i responding a when presented with query q. We can also immediately observe that answers from different provers that are mapped to the same value (and hence cause the verifier to accept) correspond to sets in S^- that are almost disjoint and strongly overlapping sets in S^+ (following the idea outlined in Section 7.3.2). To formalize this intuition, we prove the following claim.

Claim 7.3.4 *If ϕ is satisfiable, then* $\texttt{MMaxAgr}(\mathcal{S}(\phi)) \geq (2-\epsilon)T$ *for* $T = m|R|$.

Proof: Let $\bar{P}$ be the k-prover that always answers correctly and consistently and let $P_i(q)$ denote the answer of the ith prover to question q. By the correctness of $\bar{P}$, $P_i(q) \in A_i(\phi, q)$, and therefore we can define

$$I = \{(q, P_i(q), i) \mid i \in [k],\ q \in Q_i\} \ .$$

For each $r \in R$, the prover $\bar{P}$ satisfies

$$V_1(r, P_1(q_1(r))) = V_2(r, P_2(q_2(r))) = \cdots = V_k(r, P_k(q_k(r))) = c(r) \ .$$

Therefore,

$$S_r^- \cap \bigcup_{i\in[k]} S^-_{(q_i(r),P_i(q_i(r)),i)} \supseteq \bigcup_{i\in[k]} (r, B_{c(r),i}) = (r,B) = S_r^- \ .$$

This means that sets with indices in I cover all the points in $S^- = R\times B$, or $\mathtt{Agr}^-(\mathcal{S}(\phi), I) = m|R| = T$. On the other hand for each $r \in R$,

$$\begin{aligned} S_r^+ \cap \bigcup_{i\in[k]} S^+_{(q_i(r),P_i(q_i(r)),i)} &= \bigcup_{i\in[k]} (r, B_{c(r),1} \cup C_{P_i(q_i(r))}) \\ &= (r, B_{c(r),1}) \cup (r, \bigcup_{i\in[k]} C_{P_i(q_i(r))}) \ . \end{aligned}$$

This implies that for each r, only $(r, B_{c(r),1} \cup \bigcup_{i\in[k]} C_{P_i(q_i(r))})$ is covered in $S_r^+ = (r,B)$. By property 5 of balanced set partitions, the size of this set is at most

$$(1-(1-\frac{1}{k})(1-\frac{1}{k^2})^k + \gamma)m \le (1-(1-\frac{1}{k})^2+\gamma)m \le (\frac{2}{k}+\gamma)m < \epsilon m \ .$$

This means that $\mathtt{Agr}^+(\mathcal{S}(\phi), I) \le \epsilon m|R| = \epsilon T$. Altogether,

$$\mathtt{Agr}(\mathcal{S}(\phi), I) \ge (1+1-\epsilon)m|R| = (2-\epsilon)T \ .$$

□ (Claim 7.3.4)

We now deal with the case when at most $1-\tau$ clauses of ϕ can be satisfied simultaneously. Our goal is to show that, for every $I \subseteq \mathcal{I}$, if $\mathtt{Agr}(\mathcal{S}(\phi), I)$ is "significantly" larger than $(1+\epsilon)T$ (or smaller than $(1-\epsilon)T$) then I can be used to design k provers that violate the soundness guarantee of the verifier V. We now briefly outline the idea of the proof. If I has agreement larger than $(1+\epsilon)|B|$ with $\mathcal{S}(\phi)_r$ then it must include sets from the same partition (see Section 7.3.2 for an outline of the proof of this). Sets in the same partition correspond to consistent answers and therefore can be used by provers to "fool" V. In order to enable the provers to find consistent answers, we need to make sure that the set of all answers corresponding to a certain query to a prover is not "too" large. This is ensured by the inclusion of C_a's in $S^{+/-}_{(q,a,i)}$'s (and this is exactly the reason why these sets are needed). If "too" many different answers correspond to a certain query then the included C_a's will cover almost all of (r,B). In this situation the agreement of I with $\mathcal{S}(\phi)_r$ cannot be larger than $(1+\epsilon)|B|$ (or smaller than $(1-\epsilon)|B|$) and hence the number of possible answers is not "too" large. We now give the formal analysis of this case.

We say that r is *good* if $|\mathtt{Agr}(\mathcal{S}(\phi)_r, I) - m| > \frac{\epsilon}{2}m$, and let δ_I denote the fraction of good r's. Then

$$\mathtt{Agr}(\mathcal{S}(\phi), I) \leq \delta_I \cdot 2T + (1-\delta_I)(1+\epsilon/2)T \leq (1+\epsilon/2+2\delta_I)T \text{ , and}$$

$$\mathtt{Agr}(\mathcal{S}(\phi), I) \geq (1-\delta_I)(1-\epsilon/2)T \geq (1-\epsilon/2-\delta_I)T \ .$$

Hence,

$$|\mathtt{Agr}(\mathcal{S}(\phi), I) - T| \leq (\epsilon/2 + 2\delta_I)T. \tag{7.1}$$

Claim 7.3.5 *If at most $1-\tau$ clauses of ϕ can be satisfied simultaneously then for any set of indices $I \subseteq \mathcal{I}$, there exists a prover $\bar{P}$ that will make the verifier V accept with probability at least $\delta_I(k^2 \ln \frac{8}{\epsilon})^{-2}$.*

Proof: We define $\bar{P}$ with the following randomized strategy. Let q be a question to prover i. Let P_i be the prover that presented with q, answers with a random element from $A_i^q = \{a \mid (q,a,i) \in I\}$. We will show that properties of $\mathcal{B}$ imply that, for any good r, there exist i', j', $a_{i'} \in A_{i'}^{q_{i'}(r)}$ and $a_{j'} \in A_{j'}^{q_{j'}(r)}$ such that $V_{i'}(r, a_{i'}) = V_{j'}(r, a_{j'})$. That is, for a random string r, V accepts given answers $a_{i'}$ and $a_{j'}$ from provers i' and j'. In addition, we will show that for a good $r \in R$ and all $i \in [k]$, $|A_i^{q_i(r)}| \leq k^2 \ln \frac{8}{\epsilon}$. This would imply that, with probability at least $(k^2 \ln \frac{8}{\epsilon})^{-2}$, $P_{i'}$ will choose $a_{i'}$ and $P_{j'}$ will choose $a_{j'}$ causing V to accept. As this happens for all good r's, the success probability of $\bar{P}$ is at least $\delta_I(k^2 \ln \frac{8}{\epsilon})^{-2}$ (as claimed by the lemma).

We now prove that both properties hold for any good random string r. First denote $V_i^r = \{V_i(r,a) \mid a \in A_i^{q_i(r)}\}$. This is the set of all values computed by V when used with our prover P_i on a random string r (see property 2 of Feige's proof system). By the definition of $\mathcal{S}(\phi)$, this is also the set of all partition indices of sets in I that cover $S_r^{-/+}$. Therefore,

$$\begin{aligned}\mathtt{Agr}^-(\mathcal{S}(\phi)_r, I) &= \left| S_r^- \cap \left(\bigcup_{(q,a,i)\in I} S_{(q,a,i)}^- \right) \right| \\ &= \left| \left(\bigcup_{i\in[k],\ j\in V_i^r} B_{j,i} \right) \bigcup \left(\bigcup_{i\in[k],\ a\in A_i^{q_i(r)}} C_a \right) \right| \ .\end{aligned} \tag{7.2}$$

Now, if for all $i \neq j$, $V_i^r \cap V_j^r = \emptyset$, then all elements in sets $V_1^r, \ldots, V_k^r$ are distinct and therefore, by property 5 of balanced set partitions,

$$\left|\frac{\mathtt{Agr}^-(\mathcal{S}(\phi)_r, I)}{m} - 1 + (1-\frac{1}{k^2})^s(1-\frac{1}{k})^t\right| \le \gamma \ , \tag{7.3}$$

where $s = \left|\bigcup_{i\in[k]} A_i^{q_i(r)}\right|$ and $t = \sum_{i\in[k]} |V_i^r|$. Similarly,

$$\begin{aligned}\mathtt{Agr}^+(\mathcal{S}(\phi)_r, I) &= m - \left|S_r^+ \cap \left(\bigcup_{(q,a,i)\in I} S^+_{(q,a,i)}\right)\right| \\ &= m - \left|\left(\bigcup_{i\in[k],\ j\in V_i^r} B_{j,1}\right) \bigcup \left(\bigcup_{i\in[k],\ a\in A_i^{q_i(r)}} C_a\right)\right| \ .\end{aligned} \tag{7.4}$$

Again, if for all $i \ne j$, $V_i^r \cap V_j^r = \emptyset$, then by property 5 of balanced set partitions,

$$\left|\frac{1}{m}\left|\left(\bigcup_{i\in[k],\ j\in V_i^r} B_{j,1}\right) \bigcup \left(\bigcup_{i\in[k],\ a\in A_i^{q_i(r)}} C_a\right)\right| - 1 + (1-\frac{1}{k^2})^s(1-\frac{1}{k})^t\right| \le \gamma \ .$$

Hence

$$\left|\frac{\mathtt{Agr}^+(\mathcal{S}(\phi)_r, I)}{m} - (1-\frac{1}{k^2})^s(1-\frac{1}{k})^t\right| \le \gamma \ . \tag{7.5}$$

This implies (by combining equations (7.3) and (7.5)) that $|\mathtt{Agr}(\mathcal{S}(\phi)_r, I) - m| \le 2\gamma m < \frac{\epsilon}{2}m$, and therefore r is not good. In particular, for any good r, there exist i' and j' such that $V_{i'}^r \cap V_{j'}^r \ne \emptyset$. By the definition of V_i^r, this implies that there exist $a_{i'} \in A_{i'}^{q_{i'}(r)}$ and $a_{j'} \in A_{j'}^{q_{j'}(r)}$ such that $V_{i'}(r, a_{i'}) = V_{j'}(r, a_{j'})$. This gives the first property of good r's.

Now assume that for a good r and some $i' \in [k]$, $|A_{i'}^{q_{i'}(r)}| \ge k^2 \ln\frac{8}{\epsilon}$. Then $s = \left|\bigcup_{i\in[k]} A_i^{q_i(r)}\right| \ge k^2 \ln\frac{8}{\epsilon}$ and in particular, $(1-\frac{1}{k^2})^s < \frac{\epsilon}{8}$. This means that

$$\left|\bigcup_{i\in[k],\ a\in A_i^{q_i(r)}} C_a\right| \ge (1-\frac{\epsilon}{8}-\gamma)m \ .$$

Equations (7.2) and (7.4) imply that $\mathtt{Agr}^-(\mathcal{S}(\phi)_r, I) \ge (1-\frac{\epsilon}{8}-\gamma)m$ and $\mathtt{Agr}^+(\mathcal{S}(\phi)_r, I) \le (\frac{\epsilon}{8}+\gamma)m$. Altogether, this would again imply that $|\mathtt{Agr}(\mathcal{S}(\phi)_r, I) - m| \le (\frac{\epsilon}{4}+2\gamma)m = \frac{\epsilon}{2}m$, contradicting the assumption that r is good. This gives the second property of good r's and hence finishes the proof of the claim. □(Claim 7.3.5)

Using the bound on the soundness of V, Claim 7.3.5 implies that $\delta_I(k^2 \ln \frac{8}{\epsilon})^{-2} \leq k^2 2^{-c_\tau \cdot \ell}$, or $\delta_I \leq (k^3 \ln \frac{8}{\epsilon})^2 2^{-c_\tau \cdot \ell}$. Thus for

$$\ell = \frac{1}{c_\tau} \log\left(\frac{4}{\epsilon}(k^3 \ln \frac{8}{\epsilon})^2\right) \leq c_1 \log \frac{1}{\epsilon}$$

we get $\delta_I \leq \frac{\epsilon}{4}$. We set c_1 to be at least as large as c_ℓ (constant defined in Section 7.3.1). For $\delta_I \leq \frac{\epsilon}{4}$ equation (7.1) gives $|\mathtt{Agr}(\mathcal{S}(\phi), I) - T| \leq \epsilon T$. The total number of sets used in the reduction $n = |\mathcal{I}| = k \cdot |Q| \cdot |A|$, where $|Q|$ is the number of different queries that a prover can get and $|A|$ is the total number of answers that a prover can return (both $|A|$ and $|Q|$ are equal for all the provers). Therefore, by the properties 1 and 2 of Feige's proof system,

$$n = \frac{4}{\epsilon} \cdot 2^{2\ell} \cdot 2^{\ell \log(\sqrt{\frac{5}{3}} \cdot d)} = \frac{4}{\epsilon} \cdot (4\sqrt{\frac{5}{3}} \cdot d)^\ell < \frac{4}{\epsilon} \cdot (6d)^\ell \ .$$

Finally, according to Theorem 7.3.2, $m = \tilde{O}(\epsilon^{-4})$. Construction of $\mathcal{S}(\phi)$ takes time polynomial (in fact even linear) in its size. There are $2n$ subsets in $\mathcal{S}(\phi)$, each of size at most $m \cdot |Q|$. Therefore, given the balanced set partition $\mathcal{B}$, the reduction takes $d^{O(\ell)}$ time. $\square$ (Theorem 7.3.3)

BAL-SET-COVER is equivalent to MMon-MA and therefore this reduction is sufficient for obtaining hardness of approximation for MMon-MA. However, it does not immediately imply hardness of approximation for Mon-MA. Mon-MA corresponds to a generalized version of BAL-SET-COVER described in Section 7.2.2. There, in addition to a set of indices I, a candidate solution includes a vector $v \in \{0,1\}^n$. The agreement of an instance $\mathcal{S}(\phi)$ with (I, v) is defined to be

$$\mathtt{Agr}(\mathcal{S}(\phi), I, v) = |\bigcup_{i \in I} Z_i^-| + |S^+| - |\bigcup_{i \in I} Z_i^+|,$$

where $Z_i^{+/-} = S_i^{+/-}$ if $v_i = 0$ and $Z_i^{+/-} = S^{+/-} \setminus S_i^{+/-}$ if $v_i = 1$. Note that if ϕ is satisfiable then $\mathtt{MaxAgr}(\mathcal{S}(\phi)) \geq \mathtt{MMaxAgr}(\mathcal{S}(\phi)) \geq (2-\epsilon)T$. Therefore to extend our reduction to Mon-MA, it is sufficient to show the following claim.

Claim 7.3.6 *In the conditions of Theorem 7.3.3 and for an unsatisfiable ϕ, $|\mathtt{MaxAgr}(\mathcal{S}(\phi)) - T| \leq \epsilon \cdot T$.*

Proof: Assume that the claim does not hold and let I and v be such that $|\mathtt{Agr}(\mathcal{S}(\phi), I, v) - T| > \epsilon \cdot T$. Clearly, $v \neq 0^n$ (since 0^n is exactly the case handled by Theorem 7.3.3). Let $(q, a, i) \in I$ be the index for which $v_{(q,a,i)} = 1$. The set $Z^-_{(q,a,i)} = S^- \setminus S^-_{(q,a,i)}$ has size

$T - |S^-_{(q,a,i)}|$. But

$$|S^-_{(q,a,i)}| = \left|\bigcup_{q_i(r)=q} (r, B_{V_i(r,a),i} \cup C_a)\right| < |R||B_{V_i(r,a),i} \cup C_a)| \leq |R|(\rho_{1,1} + \gamma)$$
$$\leq |R|(\frac{1}{k} + \frac{1}{k^2} + \gamma) \leq |R|\epsilon/2$$

and therefore $|Z^-_{(q,a,i)}| > (1 - \epsilon/2)T$. Similarly, $|Z^+_{(q,a,i)}| > (1 - \epsilon/2)T$. This implies that $(1 - \epsilon)T \leq \mathtt{Agr}(\mathcal{S}(\phi), I, v) \leq (1 + \epsilon)T$. Hence we have arrived at a contradiction. □

In order to use this reduction to prove hardness of MAX-B-MSAT we need to show an additional property of the main reduction.

Lemma 7.3.7 *For every instance $\mathcal{S}(\phi)$ of BAL-SET-COVER generated by algorithm $\mathcal{A}$, each point $(r, b) \in R \times B$ appears in at most poly$(1/\epsilon)$ subsets in $\mathcal{S}(\phi)$.*

Proof: According to the definition of $S^-_{(q,a,i)}$, a point $(r, b) \in S^-_{(q,a,i)}$ only if $q = q_i(r)$. Therefore, there are at most $k \cdot |A_i(\phi, q)| \leq k \cdot 2^{2\ell} = O(\epsilon^{-2c_1-1})$ sets containing (r, b). The same applies to any $(r, b) \in S^+$. □

7.4 Results and Applications

We are now ready to use the reduction from Section 7.3.4 with balanced set partitions from Section 7.3.3 to prove our main theorems.

Theorem 7.4.1 (same as 7.1.1) *For every constant $\epsilon' > 0$, MMon/Mon-MA is NP-hard to approximate within a factor of $2 - \epsilon'$.*

Proof: We use Theorem 7.3.3 for $\epsilon = \epsilon'/3$. For a satisfiable formula ϕ, the reduction produces an instance $\mathcal{S}(\phi)$ of BAL-SET-COVER such that $\mathtt{MMaxAgr}(\mathcal{S}(\phi)) \geq (2 - \epsilon) \cdot T = (2 - \epsilon'/3) \cdot T$. If at most $1 - \tau$ clauses of ϕ can be satisfied simultaneously, then $\mathtt{MMaxAgr}(\mathcal{S}(\phi)) \leq (1 + \epsilon) \cdot T = (1 + \epsilon'/3) \cdot T$. The multiplicative gap between these two cases is $\frac{2-\epsilon'/3}{1+\epsilon'/3} > 2 - \epsilon'$ and therefore a $(2 - \epsilon')$-approximating algorithm for Mon-MA or MMon-MA can distinguish between them. By Lemma 7.3.1, this is NP-hard. To finish the proof we also need to verify that our reduction is efficient. In this reduction $k = \frac{4}{\epsilon}$, $\gamma = \frac{\epsilon}{8}$, and $\ell = c_1 \log(1/\epsilon)$ are constants and therefore $\mathcal{B}(m, 2^\ell, 4^\ell, \frac{4}{\epsilon}, \frac{\epsilon}{8})$ can be constructed in constant randomized time. The reduction creates an instance of BAL-SET-COVER of size polynomial in d and runs in time $d^{O(\ell)} = \text{poly}(d)$. By derandomizing the construction of $\mathcal{B}$ in the trivial way, we get a deterministic polynomial-time reduction. □

Furthermore, Remark 7.2.9 and Lemma 7.3.7 imply that for any constant ϵ, there exists a constant B such that MAX-B-MSAT is NP-hard to approximate within $2 - \epsilon$, proving Theorem 7.1.4.

Theorem 7.1.1 can be easily extended to sub-constant ϵ.

Theorem 7.4.2 (same as 7.1.2) *For any constant $\lambda > 0$, there is no polynomial-time algorithm that approximates* MMon/Mon-MA *within a factor of $2 - 2^{-\log^{1-\lambda} n}$, unless* $\mathsf{NP} \subseteq \mathsf{RTIME}(n^{poly\log(n)})$.

Proof: We repeat the proof of Theorem 7.4.1 with $\epsilon' = 2^{-\log^r d}$ for some r to be specified later. Then $k = 12 \cdot 2^{\log^r d}$, $\gamma = 2^{-\log^r d}/24$ and $\ell = c_1 \cdot \log^r d$. Therefore, according to Theorem 7.3.2, $\mathcal{B}(m, 2^\ell, 4^\ell, \frac{12}{\epsilon'}, \frac{\epsilon'}{24})$ can be constructed in polynomial in $2^{\log^r d}$ randomized time and $m = 2^{c_2 \log^r d}$. The rest of the reduction takes time $d^{O(\ell)} = 2^{O(\log^{r+1} d)}$ and creates an instance of BAL-SET-COVER over $n = d^{c_3 \log^r d} = 2^{c_3 \log^{r+1} d}$ variables. Therefore, for $r = \frac{1}{\lambda}$,

$$\log^{1-\lambda} n \le c_4 \log^{(r+1)\frac{r-1}{r}} d < \log^r d$$

for large enough d. Therefore the reduction implies hardness of approximating MMon/Mon-MA within a factor $2 - \epsilon' = 2 - 2^{-\log^r d} > 2 - 2^{-\log^{1-\lambda} n}$. □

It is easy to see that the gap in the agreement rate between $1 - \epsilon$ and $1/2 + \epsilon$ implies a gap in the disagreement rate of $\frac{1/2-\epsilon}{\epsilon} > \frac{1}{3\epsilon}$ (for small enough ϵ). That is, Theorem 7.4.2 gives the following multiplicative gap for approximating Mon-MD.

Corollary 7.4.3 (same as 7.1.3) *For any constant $\lambda > 0$, there is no polynomial time algorithm that approximates* MMon/Mon-MD *within a factor of $2^{\log^{1-\lambda} n}$, unless* $\mathsf{NP} \subseteq \mathsf{RTIME}(n^{poly\log(n)})$.

A simple application of these results is hardness of approximate agreement maximization with function classes richer than monomials. More specifically, let $\mathcal{C}$ be a class that includes monotone monomials. Assume that for every $f \in \mathcal{C}$ such that f has high agreement with the sample, one can extract a monomial with "relatively" high agreement. Then we could approximate the agreement or the disagreement rate with monomials, contradicting Theorems 7.1.1 and 7.1.2. A simple and, in fact, the most general class with this property, is the class of thresholds of monomials with low integer weights. Let $\mathtt{TH}_W(\mathcal{C})$ denote the class of all functions equal to $\frac{1}{2} + \frac{1}{2}\mathtt{sign}(\sum_{i \le k} w_i(2f_i - 1))$, where $k, w_1, \ldots, w_k$ are integer, $\sum_{i \le k} |w_i| \le W$, and $f_1, \ldots, f_k \in \mathcal{C}$ (this definition of a threshold function is simply $\mathtt{sign}(\sum_{i \le k} w_i f_i)$ when f_i and the resulting function are in the range $\{-1, +1\}$). The following lemma is a straightforward generalization of a simple lemma due to Hajnal *et al.* [HMP+93] (the original version is for $\delta = 0$).

Lemma 7.4.4 *Let $\mathcal{C}$ be a class of functions and let $f = \sum_{i\in[k]} h_i \in \mathtt{TH}_W(\mathcal{C})$. If for some function g and distribution $\mathcal{D}$, $\mathbf{Pr}_{\mathcal{D}}[f = g] \geq 1 - \delta$, then there exists $j \in [k]$ such that $|\mathbf{Pr}_{\mathcal{D}}[h_j = g] - 1/2| \geq \frac{1-\delta(W+1)}{2W}$.*

Proof: Let $\mathcal{D}'$ be the distribution $\mathcal{D}$ conditioned on $f(x) = g(x)$. By the definition of $\mathcal{D}'$, $\mathbf{Pr}_{\mathcal{D}'}[f = g] = 1$. We can therefore apply the original lemma ($\delta = 0$) and get that there exists $h_j \in \mathcal{C}$ such that $|\mathbf{Pr}_{\mathcal{D}'}[h_j = g] - 1/2| \geq \frac{1}{2W}$. If $\mathbf{Pr}_{\mathcal{D}'}[h_j = g] \geq 1/2 + \frac{1}{2W}$ then

$$\mathbf{Pr}_{\mathcal{D}}[h_j = g] \geq (1-\delta)(\frac{1}{2} + \frac{1}{2W}) = \frac{1}{2} + \frac{1-\delta(W+1)}{2W} \ .$$

Similarly, if $\mathbf{Pr}_{\mathcal{D}'}[h_j = g] \leq 1/2 - \frac{1}{2W}$ then $\mathbf{Pr}_{\mathcal{D}}[h_j = g] \leq \frac{1}{2} - \frac{1-\delta(W+1)}{2W}$. Combining these inequalities gives the claim. □

Lemma 7.4.4 implies the following results.

Corollary 7.4.5 *For any constant $\lambda > 0$ and $t = 2^{\log^{1-\lambda} n}$, there is no polynomial-time algorithm that approximates $\mathsf{MMon}/\mathtt{TH}_t(\mathsf{Mon})$-MD within a factor of t, unless $\mathsf{NP} \subseteq \mathsf{RTIME}(n^{poly\log(n)})$.*

Proof: We use Theorem 7.3.3 for $\epsilon = \frac{1}{t^2}$. Let S be the set of examples in the instance of MMon-MA equivalent to the instance $\mathcal{S}(\phi)$ of BAL-SET-COVER. If there exists a monomial with the disagreement rate with S of at most $\epsilon/2 = \frac{1}{2t^2}$ then a t-approximation algorithm for $\mathsf{MMon}/\mathtt{TH}_t(\mathsf{Mon})$-MD will produce a function $f \in \mathtt{TH}_t(\mathsf{Mon})$ with the disagreement rate of at most $\frac{1}{2t}$ with S. By Lemma 7.4.4, there exist a monomial h of f such that

$$|\mathtt{AgreeR}(h, S) - \frac{1}{2}| \geq \frac{1 - \frac{1}{2t}(t+1)}{2t} = \frac{1}{4t} - \frac{1}{4t^2} > \frac{1}{2t^2} \ .$$

To apply the lemma we take $\mathcal{D}$ to be uniform over S. Therefore the approximation algorithm can distinguish between this case and the case when no monomial has agreement rate within $\epsilon/2 = \frac{1}{2t^2}$ of $1/2$.

The parameters of the reduction are analogous to those in the proof of Theorem 7.4.2 and hence the resulting assumption is the same. □

Corollary 7.4.6 *For every integer constant $W \geq 1$ and constant $\epsilon' > 0$, $\mathsf{MMon}/\mathtt{TH}_W(\mathsf{Mon})$-MA is NP-hard to approximate within a factor of $1 + \frac{1}{W} - \epsilon'$.*

Proof: The reduction in Theorem 7.1.1 proves hardness of distinguishing instances of MMon-MA with the maximum agreement rate ρ being $\geq 1 - \frac{\epsilon}{2}$ and instances for which $|\rho - 1/2| \leq \frac{\epsilon}{2}$. If there exists an algorithm that, given sample S such that $\mathtt{AgreeR}(\mathsf{MMon}, S) \geq 1 - \frac{\epsilon}{2}$, can produce a function $f \in \mathrm{TH}_W(\mathsf{Mon})$ such that f agrees with at least $\frac{W}{W+1} + \epsilon$

fraction of examples then, by Lemma 7.4.4 (for $\mathcal{D}$ being the uniform distribution over S and $\delta = 1 - (\frac{W}{W+1} + \epsilon))$, one of the monomials of f has agreement rate ρ that satisfies

$$|\rho - \frac{1}{2}| \geq \frac{1-\delta(W+1)}{2W} \geq \frac{1-(\frac{1}{W+1}-\epsilon)(W+1)}{2W} = \frac{\epsilon(W+1)}{2W} > \frac{\epsilon}{2} .$$

Producing f that agrees with at least $\frac{W}{W+1}+\epsilon$ fraction of examples can be done by a $\frac{1-\epsilon/2}{\frac{W}{W+1}+\epsilon}$-approximation algorithm for $\mathsf{MMon}/\mathtt{TH}_W(\mathsf{Mon})$-MA. Therefore, $\mathsf{MMon}/\mathtt{TH}_W(\mathsf{Mon})$-MA cannot be approximated within

$$\frac{1-\epsilon/2}{\frac{W}{W+1}+\epsilon} = 1 + \frac{1-\frac{3}{2}\epsilon(W+1)}{W+\epsilon(W+1)} \geq 1 + \frac{1}{W} - \epsilon'$$

for $\epsilon = \frac{\epsilon'}{3(W+1)}$. □

A k-term DNF $t_1 \vee t_2 \vee \cdots \vee t_k$ can be expressed as $\frac{1}{2}+\frac{1}{2}\mathtt{sign}(\sum_{i\in[k]}(-1)t'_i + (k-1))$, where $t'_i = 2t_i - 1$ (here $\mathtt{sign}(0)$ is defined to be 1). Therefore k-term DNF$\subset \mathtt{TH}_{2k-1}(\mathsf{Mon})$. This implies that Corollary 7.4.6 improves the best known inapproximability factor for (2-term DNF)-MA from $\frac{59}{58} - \epsilon$ [BB06] to $4/3 - \epsilon$ and gives the first result on hardness of agreement maximization with thresholds of any constant number of terms.

Chapter 8

Conclusions and Further Directions

In this thesis we presented positive and negative results on a number of central problems in learning theory, most notably the problem of learning DNF expressions. We saw some powerful applications of the recent advanced techniques for proving inapproximability results and new techniques for manipulating the Fourier transform of real-valued functions. We believe that our contribution as well as many other recent developments in learning theory provide grounds for optimism about the prospects for resolving of many other fundamental questions.

In this chapter we highlight some of the major open questions related to this research and a number of directions in which our work might be extended.

Proper learning

In Chapter 3 we have conclusively answered the questions of proper learning of DNF expressions and intersections (or unions) of thresholds in the PAC+MQ model. In Chapter 4 we also made some progress toward proving hardness of proper PAC+MQ learning of DNF when the distribution is uniform. It is easy to see that, in the uniform case finding a DNF hypothesis $\log(s/\epsilon)$ times larger than the target can be achieved in time $n^{\log(s/\epsilon)}$, and therefore, is unlikely to be NP-hard. This means that substantial improvements of the result will have to be based on different (probably stronger) assumptions. In fact, we believe that it is likely that DNF *are* learnable properly in this model. In this situation our hardness result would be close to optimal. A small improvement of our result could be achieved by closing the gap between $O(d)$ and d^{γ} for approximating TT-MinDNF, a problem of independent interest.

An important way in which this line of research might be extended is to consider algorithms that use richer representations for their hypotheses. The closest natural candidates are Boolean circuits of depth 3 (or any constant depth) and polynomial threshold functions (or linear thresholds of monomials). Among the algorithms that use these representations as their hypotheses are Bshouty's algorithm for learning decision trees using membership queries [Bsh95] and Jackson's algorithm for learning DNF [Jac97] (the original version outputs a majority of parities but the algorithm can be modified easily to produce a majority of monomials). Clearly, the ultimate goal would be to prove hardness results without any restrictions on the representation of hypotheses (other than it being a circuit of polynomial size).

Learning with respect to the uniform distribution

In Chapter 5 we gave an algorithm for learning DNF that is relatively fast and has several other useful properties (attribute-efficiency, non-adaptiveness, and noise-tolerance). Quite surprisingly, our algorithm and all other known efficient algorithms for learning unrestricted DNF with respect to the uniform distribution [Jac97, JSS97, BJT04, BF02, BMOS03] use the same approach (due to Jackson): learn weakly by finding a correlated parity and then boost the accuracy. It seems that this approach has a significant overhead in terms of the running time complexity, and the running time of our algorithm appears to be close to the limits of this technique. Therefore it would be interesting to explore other approaches to learning DNF in this setting.

An important open question is whether DNF expressions are learnable without membership queries. As we noted in the introduction, even the potentially simpler question: "Are $\omega(1)$-juntas efficiently learnable with respect to the uniform distribution?" is still unresolved[1]. The results of Chapter 6 imply that progress on this question could be achieved by showing an algorithm for learning parities of $\omega(1)$ variables with noise of rate $1/2-o(1)$. 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 parities [BKW03].

Agnostic Learning

The results of Chapter 7 have essentially ruled out any interesting form of proper agnostic learning of monomials, adding to the present strong evidence of the hardness of this problem. Similar negative results were recently shown for halfspaces [FGKP06, GR06]. While the reduction in Chapter 6 does show that agnostic learning of parities is not

[1]Avrim Blum has declared a reward of \$1000 for the "head" of this problem [Blu03a].

harder than learning parities with random noise, the problem is still apparently very hard. Many fundamental problems in this model remain open; including the representation-independent learnability of monomials, parities, and thresholds. From a more theoretical perspective, it is very interesting to understand the limits of learnability in this natural model. Besides that, as demonstrated by this thesis, the study of this model has strong connections to significant problems in other areas of the theory of computation.

The agnostic learning model, when seen as a noise model, allows the labels of examples to be corrupted by an adversary with unlimited computational power, which is overly pessimistic for most real-life learning scenarios. The only other well-studied model of noise is the random classification noise model, which is often too benign. Therefore, from a more practical perspective, it is important to define and investigate models that place more realistic assumptions on the nature of noise. The difficulty of this task lies primarily in finding models that are both natural and general enough to be applicable to a variety of contexts.

Bibliography

[ABF$^+$04] M. Alekhnovich, M. Braverman, V. Feldman, A. Klivans, and T. Pitassi. Learnability and automizability. In *Proceeding of FOCS*, pages 621–630, 2004.

[AGGR05] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery*, 11(1):5–33, July 2005.

[AHM$^+$06] E. Allender, L. Hellerstein, P. McCabe, T. Pitassi, and M.Saks. Minimizing DNF formulas and AC^0 circuits given a truth table. In *Proceedings of IEEE Conference on Computational Complexity*, pages 237–251, 2006.

[AHPM04] E. Allender, L. Hellerstein, T. Pitassi, and M.Saks. On the complexity of finding minimal representations of boolean functions. 2004. Unpublished.

[AHU74] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley, 1974.

[AK95a] 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.

[AK95b] D. Angluin and M. Kharitonov. When won't membership queries help? *Journal of Computer and System Sciences*, 50(2):336–355, 1995.

[AL88] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2:343–370, 1988.

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

[Ang87] D. Angluin. Learning Regular Sets from Queries and Counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs : A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[Bar97] A. Barg. Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(046), 1997.

[BB02] N. Bshouty and L. Burroughs. Bounds for the minimum disagreement problem with applications to learning theory. In *Proceedings of COLT*, pages 271–286, 2002.

[BB06] N. Bshouty and L. Burroughs. Maximizing agreements and coagnostic learning. *Theoretical Computer Science*, 350(1):24–39, 2006.

[BDEL03] S. Ben-David, N. Eiron, and P. M. Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003.

[BEHW87] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

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

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximations. In *Proceedings of STOC*, pages 294–304, 1993.

[BH98] N. Bshouty and L. Hellerstein. Attribute efficient learning with queries. *Journal of Computer and System Sciences*, 56:310–319, 1998.

[BHL95] A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. *Journal of Computer and System Sciences*, 50:32–40, 1995.

[BJT99] N. Bshouty, J. Jackson, and C. Tamon. More efficient PAC learning of DNF with membership queries under the uniform distribution. In *Proceedings of COLT*, pages 286–295, 1999.

[BJT04] N. Bshouty, J. Jackson, and C. Tamon. More efficient PAC-learning of DNF with membership queries under the uniform distribution. *Journal of Computer and System Sciences*, 68(1):205–234, 2004.

[BKW00] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *Proceedings of STOC*, pages 435–440. ACM Press, 2000.

[BKW03] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal 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.

[Blu94] A. Blum. Relevant examples and relevant features: Thoughts from computational learning theory, 1994. In AAAI Fall Symposium on 'Relevance'.

[Blu98] A. Blum. Lecture notes for 15-854 machine learning theory. Available at `http://www.cs.cmu.edu/~avrim/ML98/index.html`, 1998.

[Blu03a] A. Blum. Open problem: Learning a function of r relevant variables. In *Proceeding of COLT*, pages 731–733, 2003.

[Blu03b] A. Blum. Tutorial on Machine Learning Theory given at FOCS '03, 2003. Available at `http://www.cs.cmu.edu/~avrim/Talks/FOCS03/`.

[BMOS03] N. Bshouty, E. Mossel, R. O'Donnell, and R. Servedio. Learning DNF from random walks. In *Proceedings of FOCS*, pages 189–199, 2003.

[BMvT78] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24, 1978.

[BR92] A. L. Blum and R. L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5(1):117–127, 1992.

[BR95] A. Blum and S. Rudich. Fast learning of k-term DNF formulas with queries. *Journal of Computer and System Sciences*, 51(3):367–373, 1995.

[Bsh95] N. Bshouty. Exact learning via the monotone theory. *Information and Computation*, 123(1):146–153, 1995.

[Bsh96] N. Bshouty. A subexponential exact learning algorithm for DNF using equivalence queries. *Information Processing Letters*, 59:37–39, 1996.

[BT96] N. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996.

[Che52] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist.*, 23:493–507, 1952.

[Chv79] V. Chvàtal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[CM94] O. Coudert and J. Madre. METAPRIME, an Interactive Fault-Tree Analyser. *IEEE Transactions on Reliability*, 43(1):121–127, 1994.

[CS01] O. Coudert and T. Sasao. Two-level logic minimization. In R. Brayton, S. Hassoun, and T. Sasao, editors, *Logic Synthesis and Verification*, pages 1–29. Kluwer, 2001.

[CT65] J. Cooley and J. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Computat.*, 19:297–301, 1965.

[Czo99] S. Czort. The complexity of minimizing disjunctive normal form formulas. Master's thesis, University of Aarhus, 1999.

[Dam98] P. Damaschke. Adaptive versus nonadaptive attribute-efficient learning. In *Proceedings of STOC*, pages 590–596, 1998.

[DGKR05] I. Dinur, V. Guruswami, S. Khot, and O. Regev. A new multilayered pcp and the hardness of hypergraph vertex cover. *SIAM Journal of Computing*, 34(5):1129–1146, 2005.

[DGR99] S. Decatur, O. Goldreich, and D. Ron. Computational sample complexity. *SIAM Journal on Computing*, 29(3):854–879, 1999.

[DHW91] N. Littlestone D. Haussler, M.Kearns and M. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, 1991.

[DRS02] I. Dinur, O. Regev, and C. Smyth. The hardness of 3-uniform hypergraph coloring. In *Proceedings of FOCS*, pages 33–42, 2002.

[EFF85] P. Erdös, P. Frankl, and Z. Furedi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51:79–89, 1985.

[Fei95] U. Feige. Randomized graph products, chromatic numbers, and the Lovász θ-function. In *STOC*, pages 635–640, 1995.

[Fei98] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[Fel05] V. Feldman. On attribute efficient and non-adaptive learning of parities and DNF expressions. In *Proceedings of COLT*, pages 576–590, 2005.

[Fel06a] V. Feldman. Hardness of approximate two-level logic minimization and PAC learning with membership queries. In *Proceedings of STOC*, pages 363–372, 2006.

[Fel06b] V. Feldman. Optimal hardness results for maximizing agreements with monomials. In *Proceedings of Conference on Computational Complexity (CCC)*, pages 226–236, 2006.

[FGKP06] V. Feldman, P. Gopalan, S. Khot, and A. Ponuswami. New Results for Learning Noisy Parities and Halfspaces. In *Proceedings of FOCS*, pages 563–574, 2006.

[FK96] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Proceedings of Conference on Computational Complexity (CCC-96)*, pages 278–289, May 24–27 1996.

[FKKM97] M. Farach, S. Kannan, E. Knill, and S. Muthukrishnan. Group testing problems in experimental molecular biology. In *Proceedings of Sequences '97*, 1997.

[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.

[Fre95] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[Gav03] D. Gavinsky. Optimally-smooth adaptive boosting and application to agnostic learning. *Journal of Machine Learning Research*, 4:101–117, 2003.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GHS00] V. Guruswami, J. Hastad, and M. Sudan. Hardness of approximate hypergraph coloring. In *Proceedings of FOCS*, pages 149–158, 2000.

[Gim65] J.F. Gimpel. A method for producing a boolean function having an arbitrary prescribed prime implicant table. *IEEE Transactions on Computers*, 14:485–488, 1965.

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

[GKS93] S. Goldman, M. Kearns, and R. Schapire. Exact identification of read-once formulas using fixed points of amplification functions. *SIAM Journal on Computing*, 22(4):705–726, 1993.

[GKS01] S. A. Goldman, S. Kwek, and S. D. Scott. Agnostic learning of geometric patterns. *Journal of Computer and System Sciences*, 62(1):123–151, 2001.

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

[GLR99] D. Guijarro, V. Lavin, and V. Raghavan. Exact learning when irrelevant variables abound. In *Proceedings of EuroCOLT '99*, pages 91–100, 1999.

[Gol78] E. A. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[GR06] V. Guruswami and P. Raghavendra. Hardness of Learning Halfspaces with Noise. In *Proceedings of FOCS*, pages 543–552, 2006.

[GTT99] D. Guijarro, J. Tarui, and T. Tsukiji. Finding relevant variables in PAC model with membership queries. *Lecture Notes in Artificial Intelligence*, 1720:313 – 322, 1999.

[Han92] T. Hancock. *The complexity of learning formulas and decision trees that have restricted reads.* PhD thesis, Harvard University, 1992.

[Has01] J. Hastad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[Hau90] D. Haussler. Probably approximately correct learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1101–1108, 1990.

[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.

[Hel01] L. Hellerstein. Learning DNF formulas: a selected survey, 2001. Avalilable at `http://cis.poly.edu/~hstein/lics.ps`.

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

[Hof99] T. Hofmeister. An application of codes to attribute-efficient learning. In *Proceedings of EuroCOLT*, pages 101–110, 1999.

[HR02] L. Hellerstein and V. Raghavan. Exact learning of DNF formulas using DNF hypotheses. In *Proceedings of STOC*, pages 465–473, 2002.

[HvHS95] K. Hoffgen, K. van Horn, and H. U. Simon. Robust trainability of single neurons. *Journal of Computer and System Sciences*, 50(1):114–125, 1995.

[Jac95] J. Jackson. *The Harmonic sieve: a novel application of Fourier analysis to machine learning theory and practice.* PhD thesis, Carnegie Mellon University, August 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.

[Jac04] J. Jackson. Personal communication, 2004.

[JKS02] J. Jackson, A. Klivans, and R. Servedio. Learnability beyond AC^0. In *Proceedings of STOC*, pages 776–784, 2002.

[JSS97] J. Jackson, E. Shamir, and C. Shwartzman. Learning with queries corrupted by classification noise. In *Proceedings of the Fifth Israel Symposium on the Theory of Computing Systems*, pages 45–53, 1997.

[Kea98] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.

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

[Kho02] S. Khot. Hardness results for coloring 3-Colorable 3-Uniform hypergraphs. In *Proceedings of FOCS*, pages 23–32, 2002.

[KKMS05] A. Tauman Kalai, A. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. In *Proceedings of FOCS*, 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.

[KLPV87a] M. Kearns, M. Li, L. Pitt, and L. Valiant. On the learnability of Boolean formulae. In *Proceedings of STOC*, pages 285–295, 1987.

[KLPV87b] M. Kearns, M. Li, L. Pitt, and L. Valiant. Recent results on boolean concept learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 337–352, 1987.

[KM93] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[KS64] W. Kautz and R. Singleton. Nonrandom binary superimposed codes. *IEEE Trans. Inform. Theory*, 10:363–377, 1964.

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

[KS04a] A. Klivans and R. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *Journal of Computer and System Sciences*, 68(2):303–318, 2004.

[KS04b] A. Klivans and R. Servedio. Toward attribute efficient learning of decision lists and parities. In *Proceedings of COLT*, pages 234–248, 2004.

[KSS94] M. Kearns, R. Schapire, and L. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

[KV94a] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

[KV94b] M. Kearns and U. Vazirani. *An introduction to computational learning theory.* MIT Press, Cambridge, MA, 1994.

[LBW95] W. S. Lee, P. L. Bartlett, and R. C. Williamson. On efficient agnostic learning of linear combinations of basis functions. In *Proceedings of COLT*, pages 369–376, 1995.

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

[Lit88] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[Liu02] H. Liu. Routing table compaction in ternary cam. *IEEE Micro*, 22(1):58–64, 2002.

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

[LV89] N. Linial and U. Vazirani. Graph products and chromatic numbers. In *Proceeding of FOCS*, pages 124–128, 1989.

[LY94] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

[Man94] Y. Mansour. Learning boolean functions via the fourier transform. In V. P. Roychodhury, K. Y. Siu, and A. Orlitsky, editors, *Theoretical Advances in Neural Computation and Learning*, pages 391–424. Kluwer, 1994.

[Mas69] J. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Inform. Theory*, 15:122–127, 1969.

[Mas79] W. Masek. Some NP-complete set covering problems. unpublished, 1979.

[McC56] E. L. Jr. McCluskey. Minimization of Boolean Functions. *Bell Sys. Tech. Jour.*, 35:1417–1444, 1956.

[McE78] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN progress report*, 42-44, 1978.

[MOS04] E. Mossel, R. O'Donnell, and R. Servedio. Learning functions of k relevant variables. *Journal of Computer and System Sciences*, 69(3):421–434, 2004.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[NJS98] R. Nock, P. Jappy, and J. Sallantin. Generalized graph colorability and compressibility of boolean formulae. In *Proceedings of the 9th International Symposium on Algorithms and Computation (ISAAC)*, 1998.

[NR97] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudorandom functions. In *Proceedings of FOCS*, pages 458–467, 1997.

[NW94] N. Nisan and A. Wigderson. Hardness versus randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[Pap94] C. H. Papadimitriou. *Computational Complexity.* Addison Wesley Pub. Co., 1994.

[Pau74] W. J. Paul. Boolesche minimalpolynome und überdeckungsprobleme. *Acta Informatica*, 4:321–336, 1974.

[PB90] L. Pitt and R. Board. On the necessity of Occam algorithms. In *Proceddings of STOC*, pages 54–63, 1990.

[PV88] L. Pitt and L. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[Qui52] W.V. Quine. The problem of simplifying truth functions. *Americal Mathematical Monthly*, 59:521–531, 1952.

[Qui56] W.V. Quine. A way to simplify truth functions. *Americal Mathematical Monthly*, 62:627–631, 1956.

[Raz98] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

[RS97] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of STOC*, pages 475–484, 1997.

[Sch90] R. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[Ser00] R. Servedio. Computational sample complexity and attribute-efficient learning. *Journal of Computer and System Sciences*, 60(1):161–178, 2000.

[Ser01] R. Servedio. On learning monotone DNF under product distributions. In *Proceedings of COLT*, pages 473–489, 2001.

[Ser03] R. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

[SM00] Y. Sakai and A. Maruoka. Learning monotone log-term DNF formulas under the uniform distribution. *Theory of Computing Systems*, 33:17–33, 2000.

[Sud02] M. Sudan. Essential coding theory (lecture notes). Available at `http://theory.lcs.mit.edu/~madhu/FT02/`, 2002.

[Tre01] L. Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of STOC*, pages 453–461, 2001.

[TS96] A. Ta-Shma. A Note on PCP vs. MIP. *Information Processing Letters*, 58(3):135–140, 1996.

[Uma99] C. Umans. Hardness of approximating σ_2^p minimization problems. In *Proceedings of FOCS*, pages 465–474, 1999.

[UTW97] R. Uehara, K. Tsuchida, and I. Wegener. Optimal attribute-efficient learning of disjunction, parity, and threshold functions. In *Proceedings of EuroCOLT '97*, pages 171–184, 1997.

[UVSV06] C. Umans, T. Villa, and A. L. Sangiovanni-Vincentelli. Complexity of two-level logic minimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(7):1230–1246, 2006.

[Vad04] S. Vadhan. Lecture notes on pseudorandomness. Available at `http://www.courses.fas.harvard.edu/~cs225/`, 2004.

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

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

[Val94] L. G. Valiant. *Circuits of the Mind*. Oxford University Press, 1994.

[Val00] L. G. Valiant. A neuroidal architecture for cognitive computation. *Journal of the ACM*, 47(5):854–882, 2000.

[Val06] L. G. Valiant. Knowledge infusion. In *Proceedings of AAAI*, 2006.

[Var97] A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *Proceedings of STOC*, pages 92–109, 1997.

[Ver90] K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 314–326, 1990.

[vL98] J.H. van Lint. *Introduction to Coding Theory*. Springer, Berlin, 1998.