Efficient learning algorithms with limited information

Anindya De

Efficient learning algorithms with limited information

by

Anindya De

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Fall 2013

Efficient learning algorithms with limited information

Abstract

Efficient learning algorithms with limited information

by

Anindya De

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Umesh V. Vazirani, Co-chair

Professor Luca Trevisan, Co-chair

The thesis explores efficient learning algorithms in settings which are more restrictive than the PAC model of learning [144] in one of the following two senses: (i) The learning algorithm has a very weak access to the unknown function, as in, it does not get labeled samples for the unknown function (ii) The error guarantee required from the hypothesis is more stringent than the PAC model.

Ever since its introduction, the PAC model of learning is considered as the standard model of learning. However, there are many situations encountered in practice in which the PAC model does not adequately model the learning problem in hand. To meet this limitation, several other models have been introduced for e.g. the *agnostic learning* model [65, 88], the *restricted focus of attention* model [12], the *positive examples only* model [35] amongst others. The last two models are the ones which are most relevant to the work in this thesis.

Beyond the motivation of modeling learning problems, another reason to consider these alternate models is because of the rich interplay between the questions arising here and other areas like computational complexity theory and social choice theory. Further, the study of these models lead to interesting questions in discrete Fourier analysis and probability theory. In fact, the connections forged between these learning questions and Fourier analysis and probability theory are key to the results in this thesis.

Part 1 of this thesis is motivated by an easy theorem of C.K. Chow [23] who showed that over the uniform distribution on the hypercube, a halfspace is uniquely identified (in the space of all Boolean functions) by its expectation and correlation with the input bits. Rephrased in the language of Fourier analysis, if $f : \{-1,1\}^n \to \{-1,1\}$ is a halfspace and $g : \{-1,1\}^n \to \{-1,1\}$ is any other Boolean function, such that the degree 0 and 1 Fourier coefficients of f and g are the same, then f and g are identical over the hypercube. The degree 0 and 1 Fourier coefficients of f are sometimes called the Chow parameters of f. Chow's result immediately gives rise to two questions: The first is whether this characterization is *robust* i.e. if the Chow parameters of f and g are close to each other and f is halfspace, then are f and g close to each other in Hamming distance? The second question is whether Chow's characterization is algorithmic. In

other words, is there an efficient algorithm which given the Chow parameters of a halfspace f, can reconstruct the halfspace f (exactly or approximately)? Chapter 2 answers both these questions in the affirmative and in fact shows a strong connection between the answers to the two questions.

In Chapter 3, we consider a different generalization of the theorem of Chow. It is easy to extend the theorem of Chow that show over any distribution D on the hypercube $\{-1,1\}^n$ (which has full support), a halfspace f is completely characterized within the space of all Boolean functions by its expectation and correlation with the input bits. Similar to Chapter 2, one can ask if this characterization can be made *algorithmic* and *robust*. While the question is interesting in its own right, for specific choices of distribution D, this problem has been investigated intensively in the social choice literature where it is known as Inverse power index problem. Informally speaking, the function f is viewed as a *voting function* and the correlation between f and an input bit is viewed as the *influence* of that input bit (The distribution D is dependent on the definition of influence that is used). Arguably the most popular choice for power indices in social choice literature is the Shapley-Shubik indices [130]. Despite much interest in the Inverse power index problem for Shapley-Shubik indices, prior to our work, there was no rigorous algorithm for this problem. In Chapter 3, we give the first polynomial time approximation scheme for the Inverse power index problem for Shapley-Shubik indices.

Part 2 of this thesis is motivated by the problem of learning distributions over the hypercube. While the problem of learning distributions has a rich history in statistics and computer science, there has not been much work in learning distributions over the hypercube which are interesting from the point of view of Boolean functions. While most natural learning problems that can be framed in this context can easily shown to be hard (under cryptographic assumption), a sweet spot comes from the following kind of learning problems: Let $\mathcal{C}$ be an *interesting* class of Boolean functions (say halfspaces). Given an unknown $f \in \mathcal{C}$, the distribution learning problem we consider is to learn the uniform distribution over $f^{-1}(1)$ provided we have access to random samples from this set. This question has interesting connections to questions in complexity theory like sampling NP solutions among others. In Chapter 3, we consider this problem for many different instantiations of the class $\mathcal{C}$. While we give efficient algorithms for classes like halfspaces and DNFs, we show that classes like CNFs and degree-d polynomial threshold functions which are learnable in the standard PAC model, are cryptographically hard to learn in this context.

To Maa and Baba who taught me the importance of hard work, honesty and perseverance.

And to Sachin Tendulkar for making the impossible look so routine.

Contents

Acknowledgments

Acknowledgments are often somewhat clichéd. And yet, as I start writing mine, I realize it is going to look no less clichéd. Probably the only reason this is so is because of how incredibly lucky I have been to have such an enormous number of incredibly nice and helpful people in my life and it is very important to me that I thank all these people.

I would like to begin by thanking my advisor Luca Trevisan for his help, guidance and the freedom he gave me to pursue my ideas throughout the last five years. During the beginning of my graduate career, when I was scared and had no idea of how to do research, Luca helped me take my first steps and when the time came that I do things on my own, Luca could take a step back as well. Besides his understanding of extractors, I hope his indefatigable spirit of pursuing a proof until he sees it as *series of logically inevitable steps* has rubbed off on me.

I would like to thank my *other* advisor, Umesh Vazirani for all his advice during these years and for convincing me to come to Berkeley in the first place. It took Umesh all of ten minutes during his first phone conversation with me to convince me to come to Berkeley and five years in hindsight, I am convinced that there is no better place to come for graduate school than Berkeley. While it will be naive to think that Umesh's uncanny ability to look at the bigger picture can be inculcated, I sincerely hope that I have taken a leaf off it.

To say that theory group at Berkeley is a great place to do theory is stating the obvious. What probably is not so obvious is how incredibly supportive the group is. Elchanan, thanks for inviting me to work with you and having the patience to continue working with me even when I was "only" a 100 times slower than you. Thanks for your cheerfulness when we discuss math (or anything else) and for always having some problem to talk about. Prasad and Satish, thanks for all the countless conversations we had about different problems and for the energy you show during these conversations. Alistair, Christos and Dick, thanks for always having the time to listen to any problems (academic or administrative) and the incredible hard work you put to solve these problems. I would like to acknowledge Prasad, Satish and Umesh for funding me during the last two years of my graduate school. I would also like to thank Lauren Williams and David Aldous for agreeing to be on my candidacy exam and dissertation committees respectively.

Not just that the theory students at Berkeley are incredibly smart, but the intense sense of camaraderie made Soda Hall a wonderful place to spend time. I am thankful to Grant for the time we spent thinking about monotone codes and telling me why owning a car is more important than a house; Lorenzo for all the conversations we had about India, Italy and cricket and for his friendship; Madhur for encouraging me about research and help me take my first steps when I was not so sure of things and also making me realize that cooked in the right way, tofu is not all that bad; Alexandre for all the time we spent gossiping and telling me about the only useful thing that can be purchased in Sau Paulo; Greg for feeding me the gossip which I could discuss with Alexandre and once, getting me a large sum of money from the department (even though till date, I am not quite sure of its source); James for listening to me while I griped about the world and made fun of Canada; Thomas for all the conversations and collaborating with me on "quantum extractors" knowing fully well that I have no idea about the first word and a vague one about the second. Isabelle and Raf, thanks for being incredible officemates and putting up with me while the stuff on my desk piled on; Anand, Piyush, Siu-Man, Siu-On and Tom for all the dinners we had or making fun of Siu-Man when we went for dinners and he started taking photographs of the food; Urmila for the times she drove me to Berkeley bowl and being a reliable (albeit creepy) source of gossip ; Andrew, Chris, George, Guoming, Henry, Ma'ayan, Nima, Omid, Antonio, Di, Woo, Jonah, Andrew, Paul, Alex, Aviad, Tselil, Rishi, Jarett, Jonah, Ben and other current and past Berkeley theory students, thanks for making the theory group here such a wonderful place. Also, thanks to Neha for bringing the delicious food in the theory lunches.

I would like to thank my co-authors Costis Daskalakis, Ilias Diakonikolas, Omid Etesami, Vitaly Feldman, Piyush Kurur, Ankur Moitra, Elchanan Mossel, Joe Neeman, Christopher Portmann, Renato Renner, Chandan Saha, Ramprasad Saptharishi, Rocco Servedio, Luca Trevisan, Madhur Tulsiani, Thomas Vidick and Tom Watson for all the things I learnt from them and for the patience they showed while I was learning these things.

I would like to thank Debkishore and Piyush for putting up with me as a housemate for all these years and for all the delightful times we shared. My social life at Berkeley (outside the EECS department) revolved around Ajith, Arka, Aastha, Debkishore, Devi, Himanshu, Piyush, Raj, Momo and Shaunak and I am thankful for all the birthday celebrations, dinners and above all sharing the experience of India's world cup win in 2011.

The material in this thesis is the result of my collaboration with Ilias Diakonikolas and Rocco Servedio. Ilias, thanks for initiating collaboration with me on linear threshold functions when I did not know that they are alternately called halfspaces. Thanks for all the fun times we spent collaborating, having dinners and above all for your friendship.

Rocco, I can go on and on about how much I have enjoy collaborating with you but you know all that. So, all I will say is that when I was very uncertain about my future in research and things looked very dark, thanks for making me believe in myself. Thanks for your friendship and the dinners I had with your family. And before I forget to say it, thanks for always letting me know, through your actions, that you have my back.

I would like to thank Cynthia Dwork for hosting me during the summer of 2010 and introducing me to the area of differential privacy. I spent two summers and countless visits at Columbia University and I would like to thank the faculty and the students in the theory group for the great time I had during these visits. I would especially like to thank Clement, Dimitris, Eva and Igor for the lunches, dinners and the fun conversations I had during the time I spent there. I am grateful to Oded Regev for hosting me during the summer of 2013 and to Oded and Jop for all the conversations about planted clique.

Looking back further, the time I spent during my undergraduate years at IIT Kanpur was undoubtedly the most amazing time of my life. I had a great time and made an amazing number of friends who continue to be my closest confidants. If I started naming all the friends I made and how much they have helped me, I very much doubt that I will ever be able to finish this thesis. So, let me just say that I am really really thankful for all you guys and all that you have done for me.

Much before I came to graduate school, some exceptional people inspired me to do research and theoretical computer science. I would like to thank S. A. Soman for showing faith in my abilities to do research when there was no just reason to do so and for introducing me to the term *NP-completeness*. Words are not enough to express what I owe to Manindra Agrawal and

Somenath Biswas but had it not been for them, I would have never come to graduate school. I am grateful to all the other teachers I had at IIT Kanpur especially Surender Baswana and Piyush Kurur for the mathematics I learnt from them. Thanks to Chandan, Piyush and Ramprasad for collaborating with me on my first theory paper and keeping me as a co-author even though it was very quickly evident that I had no idea of what was going on. Thanks to Ashish Goel for inviting me to spend a summer at Stanford and giving me the first taste of research in grad school.

I would like to thank Somnath for being a steadfast friend for the last sixteen years and reminding me how much fun growing up in Ranchi was and the fact that after all this separation in time and space, we are still on the same page. I am grateful to Soham for being a friend since the time I did not know what being a friend meant. Thanks for telling me about primality testing when I was in high school, and yes, when I grow up, I would like to be like you.

Mudra, thanks for the countless hours you spent listening to me on the phone while I was complaining about something or the other. I know I don't make it easy. Also, thanks for the time you feigned attention when I was talking about Malliavin calculus and the time you were genuinely interested when I was talking about Majority is Stablest.

Thanks to *Boudi* for the gifts I get from her on my birthday. Thanks to my niece Katyayani for her constant entertainment and I hope you're happy now that you have replaced me as the youngest in the family.

Manu, thanks for all the fights we had growing up and the time I tried to break your knee by banging my nose against it; thanks for all the cricket matches we played in our living room and ended up pretty much shattering every single thing made of glass; thanks for the time when we thought that we thought we have really old stamps because we confused the date of the people pictured on the stamps with how old the stamps were; thanks for all the old memories and the new ones we will continue to create.

In spite of my many tantrums, my mother took great pains in ensuring that I go to school. And my father, through his actions, taught me the importance of hard work and perseverance. They gave me a happy childhood and the foundations on which all that I have ever learnt is based. While no words can do justice, I hope these come close: with lots of fondness and love, this thesis is dedicated to Baba and Maa.

Chapter 1

Introduction

The task of learning is amongst the most fundamental tasks of human existence. The ability to learn, in other words, the ability to infer from observations is the most important reason for the success of human species. While learning or inference is performed routinely in all sciences, the problem of learning was first rigorously put under the scanner of theoretical computer science by the seminal work of Valiant [144].

In this work, Valiant defined the Probably Approximately Correct (PAC) model of learning. The model is parameterized by a domain X, a distribution D over X and a subset of functions $\mathcal{C}$ which map X to $\{0,1\}$. The class $\mathcal{C}$ is referred to as the *concept class*. The algorithmic problem is as follows : An adversary fixes an $f \in \mathcal{C}$ and the *learner* only access to samples of the form $(x, f(x))$ where $x \sim D$. The learner is said to be a (T, ϵ, δ) learner if after time T, the learner outputs a *hypothesis* h (of complexity bounded by T) such that over the randomness of samples, with probability $1-\delta$, the hypothesis h satisfies $\mathbf{P}r_{x \in D}[f(x) \neq h(x)] \leq \epsilon$.

From the perspective of computer science, probably the most natural domain to consider is $X = \{0,1\}^n$. In this setting, we would ideally like the learner to be efficient i.e. we would like the time complexity of the learner to be $T = poly(n, 1/\epsilon, \log(1/\delta))$. As a result, it makes sense to only consider classes $\mathcal{C}$ with limited expressive abilities like halfspaces, DNFs, decision trees etc. Even with this simple definition, there are a couple of classifications possible depending on the nature of the learner and the hypothesis. In particular, if the learner depends on the distribution D, it is said to be a *distribution dependent learner* whereas if the learner is independent of the distribution, then the learner is said to be a *distribution independent learner*. Likewise, if the hypothesis $h \in \mathcal{C}$, then the learner is said to be *proper*, else its said to be *improper*.

1.1 Deviations from the PAC model

Agnostic learning model

Soon after the introduction of the PAC model, it was realized that it does not adequately model all learning situations encountered in practice. A particularly well known variant of the model is the *agnostic learning model* the *agnostic learning* model introduced by Haussler [65] and Kearns *et. al.* [88]. In this model, similar to the PAC model, the learner gets access to random samples $(x, f(x))$ for an unknown $f \in \mathcal{C}$. However, an (adversarially chosen) η-fraction of samples are corrupted as in the value of $f(x)$ gets flipped (but x remains unchanged). The task of the learner is to produce a hypothesis h such that $\mathbf{P}r_{x \sim D}[h(x) \neq f(x)] \leq \epsilon + \eta$ for any $\epsilon > 0$. This model of learning proves to be much more challenging than the PAC learning model. An evidence of this is the fact that while halfspaces are learnable in the PAC model under any distribution (using linear programming), halfspaces are not known to be efficiently learnable in the agnostic learning model except for the uniform distribution where the running time is $n^{\tilde{O}(1/\epsilon^2)}$ [78]. In fact, under standard assumptions, this problem is known to be hard under arbitrary distributions [62]. Still, it remains of interest to get better algorithms for specific distributions of interest or under additional assumptions on the structure of the hidden halfspace.

k-Restricted Focus of Attention model of learning

Ben-David and Dichterman [12] introduced the "Restricted Focus of Attention" (RFA) learning framework to model the phenomenon (common in the real world) of a learner having incomplete access to examples. This model is parameterized by a number k. In this model, similar to the PAC models, random samples x are chosen from an underlying distribution D. However, instead of giving the learner access to $(x, f(x))$, the learner is shown $f(x)$ and k indices of x (of the learner's choice). To get an intuition as to why such a model might be helpful in real life applications, note that this models a learning problem where observing an attribute degenerates the sample and after k of the attributes are observed, the sample might get degenerated to the point such that observation of the $(k+1)^{th}$ attribute might be a very noisy process. This of course, can be the situation with experiments in natural sciences like biology and chemistry.

It is easy to observe (using a random sampling algorithm) that in the k-RFA model, the learner can compute $\mathbf{E}_{x \sim D}[f(x) \cdot \prod_{i \in S} x_i]$ for any $S \subseteq [n]$ of size bounded by k up to an error ϵ in time $poly(n \cdot k/\epsilon)$. In fact, the converse can also be shown i.e. we can simply assume that the learner has been given the value of $\mathbf{E}_{x \sim D}[f(x) \cdot \prod_{i \in S} x_i]$ for any $S \subseteq [n]$ of size bounded by k (up to polynomialy small error). Given this equivalent description of the k-RFA model of learning, it is clear that not every class is learnable in this model. This is because two very different functions can be identical with respect to the statistics on any k-sized set of coordinates. An easy example are two different parity functions $\chi_{S_1} : \{0,1\}^n \to \{0,1\}$ and $\chi_{S_2} : \{0,1\}^n \to \{0,1\}$ where χ_{S_1} computes the parity on the set $S_1 \subseteq [n]$ and χ_{S_2} computes the parity on set $S_2 \subseteq [n]$. If both $|S_1|, |S_2| > k$ and the underlying distribution D is uniform on $\{0,1\}^n$, then it is easy to see that $\mathbf{E}_{x \sim D}[\chi_{S_1}(x) \cdot \prod_{i \in S} x_i]$ is the same for any given S for both $f = \chi_{S_1}$ and $f = \chi_{S_2}$.

While not every function is learnable in this model (even with unlimited number of examples), it remains of interest to ask if there are specific classes of interest which might be learnable in this model. As we will see in Chapter 2, halfspaces are a class of functions which are learnable in the 1-RFA model. In contrast to the PAC model, where the sample complexity for learning a class can easily be deduced from its VC-dimension [17], getting non-trivial upper bounds on the sample complexity of learning in the k-RFA model is a formidable challenge. To give a flavor of this problem, in Chapter 2, we show that in the 1-RFA model, $poly(n) \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))}$ samples suffices to learn halfspaces (with an ϵ-accuracy) in the 1-RFA model. On the other hand, we only have a weak lower bound of $O(poly(n/\epsilon))$ on the sample complexity required to learn halfspaces in this model. To see an even more stark contrast, it can be shown that the class of polynomial threshold functions (PTFs) of degree d are learnable in the d-RFA model of learning but the only known upper bound on the sample complexity is $O(n^d \cdot 2^n)$. On the other hand, the only known lower bound on the sample complexity of learning is the trivial bound of $poly(n^d/\epsilon)$.

Learning from positive examples

Another interesting model of learning is when only kind of examples are available to the learning model. This model was considered explicitly in the paper [35] but was implicitly considered in many earlier works in practical machine learning. Besides the intrinsic appeal of the model, a large part of the motivation comes from the following kind of situation encountered in practice: Consider the problem of a child learning a language. In this setting, the learner (i.e. the child) has only positive samples available to him/her (the positive examples being the legitimate words). The learner explicitly does not explicitly get negative examples. Many other real-life situations have a similar flavor where the learner has access to just the positive examples.

As said earlier, previous papers like [35] have considered this model. In these papers, they show that with access to positive samples and unlabeled samples, one can simulate the execution of any SQ learner (the definition is given later). As most learning algorithm are in fact SQ learning algorithms, this implies that all concept classes which are learnable in the SQ model can be learnt using just positive example (we assume that the underlying distribution is something easy and hence the learner can itself sample unlabeled examples). While this seems to be a comprehensive answer, unfortunately, there is an important caveat in this solution. The caveat comes from the precise definition of error that is used. The paper [35] produces a hypothesis h such that $\mathbf{P}r_{x \sim D}[h(x) \neq f(x)] \leq \epsilon$ where D is the underlying distribution and f is the target function and the running time of their algorithm has a polynomial dependence on ϵ. Now, consider a situation in which $\mathbf{P}r_{x \sim D}[f(x) = 1]$ is negligible (say exponentially small). In these situations, it is conceivable that the hypothesis h can be identically zero. Now, while it is certainly true that h and f are equal with all but negligible probability, in a sense, this is a very uninteresting hypothesis as it is completely wrong on the *interesting* part of the distribution. Another way to quantify the issue is that $\mathbf{P}r_{x \sim D|f(x)=1}[f(x) \neq h(x)] = 1$ in this case. In other words, what we would like is that the hypothesis has an error which is small compared to the density of the positive part. Again, from a practical point of view, this is the guarantee that is desirable. For example, in the context of learning language, it is true that a random string is not a legitimate word and we would like the learner to have a small error with respect to the size of the lexicon as opposed to the size of all possible strings (of a certain length) in a given alphabet. To remedy this situation, the type of error guarantee that we would like the hypothesis to satisfy is that $\mathbf{P}r_{x \sim D|f(x)=1}[f(x) \neq h(x)] \leq \epsilon$.

While not exactly equivalent, another way to capture the kind of guarantee that we want is to consider the problem of learning the distribution over positive examples. In other words, since the learner gets $x \sim D$ conditioned on $f(x) = 1$, one can consider the task of learning the conditional distribution $\{x \sim D|f(x) = 1\}$. In other words, the learner needs to output a circuit $C : \{0,1\}^m \rightarrow Supp(D)$ such that if the distribution of the input is uniform on $\{0,1\}^m$, then the output C is ϵ-close to $\{x \sim D|f(x) = 1\}$ in total variation distance. To see the correspondence between the task of learning f such that $\mathbf{P}r_{x \sim D|f(x)=1}[f(x) \neq h(x)] \leq \epsilon$ and the task of learning the distribution $\{x \sim D|f(x) = 1\}$, note that the distribution $\{x \sim D|h(x) = 1\}$ is ϵ close in total variation distance to the distribution $\{x \sim D|f(x) = 1\}$. Of course, it might be the case that even though we learn h explicitly, sampling from $\{x \sim D|h(x) = 1\}$ is a hard problem. However, leaving aside this complication, one can see why learning a hypothesis h meeting the condition stated above is related to the problem of learning the distribution D.

1.2 Our results

We now discuss the various results which constitute this thesis. Chapter 2 contains results from [30] which is a joint work with Ilias Diakonikolas, Vitaly Feldman and Rocco Servedio. Chapter 3 contains results from [32] which is a joint work with Ilias Diakonikolas and Rocco Servedio. Chapter 4 contains results from as yet unpublished work [29] which is a joint work with Ilias Diakonikolas and Rocco Servedio.

Chow parameters problem

We begin by recalling that halfspaces are Boolean functions $f : \mathbb{R}^n \rightarrow \{-1,1\}$ of the form $f(x) = sign(w \cdot x - \theta)$ where $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$. Alternatively known as linear threshold functions, linear separators, Boolean perceptrons (of order 1) and weighted majority games, this class of functions has been extensively studied in computational complexity theory, learning theory and social choice theory amongst many other disciplines. As mentioned earlier, because linear programming is in $\mathbf{P}$, learning halfspaces in the PAC model is easy (in fact, the learner is distribution independent). However, learnability of halfspaces remains open in many other settings of interest. To define the model we are interested in, we begin by defining the *Chow parameters* of a function $f : \{-1,1\}^n \rightarrow \mathbb{R}$ as the $n+1$ values

$$\widehat{f}(0) = \mathbf{E}[f(x)], \quad \widehat{f}(i) = \mathbf{E}[f(x)x_i] \text{ for } i = 1, \ldots, n,$$

i.e. the $n+1$ degree-0 and degree-1 Fourier coefficients of f. (Here and throughout the chapter, all probabilities and expectations are with respect to the uniform distribution over $\{-1,1\}^n$ unless otherwise indicated.) It is easy to see that in general the Chow parameters of a Boolean function may provide very little information about f; for example, any parity function on at least two variables has all its Chow parameters equal to 0. However, in a surprising result, C.-K. Chow [23] showed that the Chow parameters of an LTF f *uniquely* specify f within the space of all Boolean functions mapping $\{-1,1\}^n \rightarrow \{-1,1\}$. Chow's proof (given in Section 2.2) is simple and elegant, but is completely non-constructive; it does not give any clues as to how one might use the Chow parameters to find f (or an LTF that is close to f). This naturally gives rise to the following algorithmic question, which we refer to as the "Chow Parameters Problem:"

The Chow Parameters Problem (rough statement): Given (exact or approximate) values for the Chow parameters of an unknown LTF f, output an (exact or approximate) representation of f as $\mathrm{sign}(v_1x_1 + \cdots + v_nx_n - \theta')$.

In Chapter 2, we give a partial resolution to this question. In particular, we prove the following theorem.

[Main theorem] There is an $\tilde{O}(n^2) \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))} \cdot \log(1/\delta)$-time algorithm $\mathcal{A}$ with the following property: Let $f : \{-1,1\}^n \to \{-1,1\}$ be an LTF and let $0 < \epsilon, \delta < 1/2$. If $\mathcal{A}$ is given as input ϵ, δ and (sufficiently precise estimates of) the Chow parameters of f, then $\mathcal{A}$ outputs integers $v_1, \ldots, v_n, \theta$ such that with probability at least $1 - \delta$, the linear threshold function $f^* = \mathrm{sign}(v_1x_1 + \cdots + v_nx_n - \theta)$ satisfies $\mathbf{P}r_x[f(x) \neq f^*(x)] \leq \epsilon$.

Thus we obtain an efficient randomized polynomial approximation scheme (ERPAS) with a *quasi-polynomial* dependence on $1/\epsilon$. For the subclass of LTFs with integer weights of magnitude at most $\mathrm{poly}(n)$, our algorithm runs in $\mathrm{poly}(n/\epsilon)$ time, i.e. it is a *fully* polynomial randomized approximation scheme (FPRAS).

An interesting feature of our algorithm is that it outputs an LTF with integer weights of magnitude at most $\sqrt{n} \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))}$. Hence, as a corollary of our approach, we obtain essentially optimal bounds on approximating arbitrary LTFs using LTFs with small integer weights. It has been known since the 1960s that every n-variable LTF f has an exact representation $\mathrm{sign}(w \cdot x - \theta)$ in which all the weights w_i are integers satisfying $|w_i| \leq 2^{O(n \log n)}$, and Håstad [64] has shown that there is an n-variable LTF f for which *any* integer-weight representation must have each $|w_i| \geq 2^{\Omega(n \log n)}$. However, by settling for an approximate representation (i.e. a representation $f' = \mathrm{sign}(w \cdot x - \theta)$ such that $\mathbf{P}r_x[f(x) \neq f'(x)] \leq \epsilon$), it is possible to get away with much smaller integer weights. As a consequence of our algorithm, we show that one can always construct such an approximate representation with integer weights of magnitude at most $\sqrt{n} \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))}$. In fact, we show in Chapter 2, this is nearly optimal as in there exists a LTF f such that any LTF g with integer weights bounded by $\sqrt{n} \cdot (1/\epsilon)^{O(\log\log(1/\epsilon))}$ must differ from f at ϵ fraction of the points.

Inverse Shapley value problem

A very important problem in Social choice theory is the scenario where n different parties are trying to arrive at an outcome by voting for or against a proposal. Translating this problem to the language of Boolean functions, we can assume that each party/voter casts a vote $+1$ or -1 indicating their choice. Further, there is a voting function $f : \{-1,1\}^n \to \{-1,1\}$ which aggregates the opinions of n voters to arrive at an outcome. The most popular kind of voting function (or scheme) used in practice is a weighted voting game in which the proposal passes if a weighted sum of yes-votes exceeds a predetermined threshold. In the language of computer science, these voting schemes are nothing but linear threshold functions (or halfspaces) over n Boolean variables (where the weights associated with different variables are all non-negative).

An important problem that arises in the design of good voting schemes is to quantify the influence of various voters. There have been several measures put forward by social choice theorists but the most popular one used in practice is the *Shapley-Shubik index* [130], which is also known as the index of *Shapley values* (we shall henceforth refer to it as such). The Shapley values (which we define shortly) have the property that given any weighted voting game, the Shapley values of the voters can be estimated to ϵ-accuracy in time $poly(n/\epsilon)$ using a standard random sampling approach. On the other hand, the inverse problem i.e. given a set of Shapley values, constructing a weighted voting game with (approximately) the same set of Shapley values, is significantly more challenging. In fact, despite much interest in the inverse problem from social choice theory and economics, prior to our work, there was no efficient algorithm for the inverse problem. Without further ado, we first define the Shapley values of a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$.

First, we will assume that the function f is monotone in all the coordinates. This can be done because from the point of view of social choice theory, this is the only interesting case. Further, we assume that f is not the constant function, and hence $f(-\mathbf{1}) = -1$ and $f(\mathbf{1}) = 1$ where $-\mathbf{1} = (-1,\ldots,-1)$ and $\mathbf{1} = (1,\ldots,1)$. For any permutation $\pi \in \mathbb{S}_n$, consider the following thought experiment : Flip the inputs from -1 to 1 in the order defined by π i.e. flip $x_{\pi(1)},\ldots,x_{\pi(n)}$ in order. Let $\pi(j)$ be the input such that when $x_{\pi(j)}$ flips from -1 to 1, the value of f changes from -1 to 1. The index $\pi(j)$ is said to be the pivotal input for the permutation π. The i^{th} Shapley index of f (denoted by $\tilde{f}(i)$) is defined to be

$$\tilde{f}(i) = \mathop{\mathbf{P}r}_{\pi\in\mathbb{S}_n}[x_i \text{ is pivotal for } \pi]$$

While unlike Chow parameters, the Shapley values of a weighted voting game do not uniquely identify it, it almost does the job in the sense that only one additional parameter (namely its expectation under the uniform distribution) uniquely identifies the function. Informally, the following algorithmic question is referred to as the inverse Shapley value problem.

> **The inverse Shapley value problem** (rough statement): Given (exact or approximate) values for the Shapley values of an unknown monotone LTF f, output a LTF g which has (approximately) the same Shapley values as f.

In Chapter 3, we give a partial resolution to this question. In particular, we prove the following theorem.

> **[Main theorem]** There is an $\tilde{O}(n^2)\cdot 2^{\mathrm{poly}(1/\epsilon)})\cdot\log(1/\delta)$-time algorithm $\mathcal{A}$ with the following property: Let $f : \{-1,1\}^n \to \{-1,1\}$ be a monotone LTF and let $0 < \epsilon, \delta < 1/2$. If $\mathcal{A}$ is given as input ϵ, δ and (sufficiently precise estimates of) the Shapley values of f, then $\mathcal{A}$ outputs integers $v_1,\ldots,v_n,\theta$ such that with probability at least $1-\delta$, the linear threshold function $f^* = \mathrm{sign}(v_1x_1+\cdots+v_nx_n-\theta)$ satisfies $\sum_{i=1}^n(\tilde{f}(i)-\widetilde{f^*(i)})^2 \le \epsilon$.

Thus we obtain an efficient randomized polynomial approximation scheme (ERPAS) with a *exponential* dependence on $1/\epsilon$. For the subclass of LTFs with integer weights of magnitude at most $\mathrm{poly}(n)$, our algorithm runs in $\mathrm{poly}(n/\epsilon)$ time, i.e. it is a *fully* polynomial randomized approximation scheme (FPRAS).

The machinery in this chapter bears some resemblance to the machinery in Chapter 2 though we prove new anti-concentration bounds for exchangeable distributions which are likely to be of further interest. Further, our techniques can potentially be extended to other inverse problems in social-choice theory.

Inverse approximate uniform generation

Over the past two decades, there has been substantial progress in the learning of Boolean functions. Compared to this, there has not been much progress in setting of learning distributions over $\{0,1\}^n$ since the work of Kearns *et al.* [87]. Part of the reason for this is that even very simple distributions over the hypercube are hard to learn (compared to the function learning version for the same class). In Chapter 4, we initiate the study of learning distributions on the hypercube which can be seen as the *inverse* of uniform generation of satisfying assignments of Boolean functions. We focus on specific classes of Boolean function which are interesting from learning theory point of view like halfspaces, CNFs, DNFs etc.

In such an inverse problem, the algorithm is given uniform random satisfying assignments of an unknown function f belonging to a class $\mathcal{C}$ of Boolean functions (such as linear threshold functions or polynomial-size DNF formulas), and the goal is to output a probability distribution D which is ϵ-close, in total variation distance, to the uniform distribution over $f^{-1}(1)$. Problems of this sort comprise a natural type of unsupervised learning problem in which the unknown distribution to be learned is the uniform distribution over satisfying assignments of an unknown function $f \in \mathcal{C}$.

Positive results: We prove a general positive result establishing sufficient conditions for efficient inverse approximate uniform generation for a class $\mathcal{C}$. We define a new type of algorithm called a *densifier* for $\mathcal{C}$, and show (roughly speaking) how to combine (i) a densifier, (ii) an approximate counting / uniform generation algorithm, and (iii) a Statistical Query learning algorithm, to obtain an inverse approximate uniform generation algorithm. We apply this general result to obtain a poly$(n, 1/\epsilon)$-time inverse approximate uniform generation algorithm for the class of n-variable linear threshold functions (halfspaces); and a quasipoly$(n, 1/\epsilon)$-time inverse approximate uniform generation algorithm for the class of $\mathrm{poly}(n)$-size DNF formulas.

Negative results: We prove a general negative result establishing that the existence of certain types of signature schemes in cryptography implies the hardness of certain inverse approximate uniform generation problems. We instantiate this negative result with known signature schemes from the cryptographic literature to prove (under a plausible cryptographic hardness assumption) that there are no subexponential-time inverse approximate uniform generation algorithms for 3-CNF formulas; for intersections of two halfspaces; for degree-2 polynomial threshold functions; and for monotone 2-CNF formulas.

Finally, we show that there is no general relationship between the complexity of the "forward" approximate uniform generation problem and the complexity of the inverse problem for a class $\mathcal{C}$ – it is possible for either one to be easy while the other is hard. In one direction, we show that the existence of certain types of Message Authentication Codes (MACs) in cryptography implies the hardness of certain corresponding inverse approximate uniform generation problems, and we combine this general result with recent MAC constructions from the cryptographic literature to show (under a plausible cryptographic hardness assumption) that there is a class $\mathcal{C}$ for which the "forward" approximate uniform generation problem is easy but the inverse approximate uniform generation problem is computationally hard. In the other direction, we also show (assuming the GRAPH ISOMORPHISM problem is computationally hard) that there is a problem for which inverse approximate uniform generation is easy but "forward" approximate uniform generation is computationally hard.

Chapter 2

Chow Parameters problem

We begin this chapter by recalling that a *linear threshold function*, or LTF, over $\{-1,1\}^n$ is a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ of the form

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{n} w_i x_i - \theta\right),$$

where $w_1, \dots, w_n, \theta \in \mathbb{R}$. The function $\operatorname{sign}(z)$ takes value 1 if $z \geq 0$ and takes value -1 if $z < 0$; the w_i's are the *weights* of f and θ is the *threshold.* Throughout this chapter we shall refer to them simply as LTFs.

The *Chow parameters* of a function $f : \{-1,1\}^n \to \mathbb{R}$ are the $n+1$ values

$$\widehat{f}(0) = \mathbf{E}[f(x)], \quad \widehat{f}(i) = \mathbf{E}[f(x)x_i] \text{ for } i = 1, \dots, n,$$

i.e. the $n+1$ degree-0 and degree-1 Fourier coefficients of f. (Here and throughout the chapter, all probabilities and expectations are with respect to the uniform distribution over $\{-1,1\}^n$ unless otherwise indicated.) It is easy to see that in general the Chow parameters of a Boolean function may provide very little information about f; for example, any parity function on at least two variables has all its Chow parameters equal to 0. However, in a surprising result, C.-K. Chow [23] showed that the Chow parameters of an LTF f *uniquely* specify f within the space of all Boolean functions mapping $\{-1,1\}^n \to \{-1,1\}$. Chow's proof (given in Section 2.2) is simple and elegant, but is completely non-constructive; it does not give any clues as to how one might use the Chow parameters to find f (or an LTF that is close to f). This naturally gives rise to the following algorithmic question, which we refer to as the "Chow Parameters Problem:"

> **The Chow Parameters Problem** (rough statement): Given (exact or approximate) values for the Chow parameters of an unknown LTF f, output an (exact or approximate) representation of f as $\operatorname{sign}(v_1x_1 + \cdots + v_nx_n - \theta')$.

Motivation and Prior Work. We briefly survey some previous research on the Chow Parameters problem (see Section 1.1 of [120] for a more detailed and extensive account). Motivated by applications in electrical engineering, the Chow Parameters Problem was intensively studied in the

1960s and early 1970s [45, 115, 150, 117]; several researchers suggested heuristics of various sorts [83, 152, 80, 36] which were experimentally analyzed in [151]. See [149] for a survey covering much of this early work and [10, 69] for some later work from this period.

Researchers in game theory and voting theory rediscovered Chow's theorem in the 1970s [99], and the theorem and related results have been the subject of study in those communities down to the present [123, 9, 42, 44, 136, 55, 103, 21, 53, 135, 6]. Since the Fourier coefficient $\widehat{f}(i)$ can be viewed as representing the "influence" of the i-th voter under voting scheme f (under the "Impartial Culture Assumption" in the theory of social choice, corresponding to the uniform distribution over inputs $x \in \{-1,1\}^n$), the Chow Parameters Problem corresponds to designing a set of weights for n voters so that each individual voter has a certain desired level of influence over the final outcome. This natural motivation has led practitioners to implement and empirically evaluate various heuristics for the Chow parameters problem, see [100, 102, 104, 89, 97, 98].

In the 1990s and 2000s several researchers in learning theory considered the Chow Parameters Problem. Birkendorf et al. [14] showed that the Chow Parameters Problem is equivalent to the problem of efficiently learning LTFs under the uniform distribution in the "1-Restricted Focus of Attention (1-RFA)" model of Ben-David and Dichterman [11] (we give more details on this learning model in Section 2.7). Birkendorf et al. showed that if f is an LTF with integer weights of magnitude at most $\mathrm{poly}(n)$, then estimates of the Chow parameters that are accurate to within an additive $\pm\epsilon/\mathrm{poly}(n)$ information-theoretically suffice to specify the halfspace f to within ϵ-accuracy. Other information-theoretic results of this flavor were given by [57, 129]. In complexity theory several generalizations of Chow's Theorem were given in [20, 126], and the Chow parameters play an important role in a recent study [22] of the approximation-resistance of linear threshold predicates in the area of hardness of approximation.

Despite this considerable interest in the Chow Parameters Problem from a range of different communities, the first provably effective and efficient algorithm for the Chow Parameters Problem was only obtained fairly recently. [120] gave a $\mathrm{poly}(n) \cdot 2^{2^{\tilde{O}(1/\epsilon^2)}}$-time algorithm which, given sufficiently accurate estimates of the Chow parameters of an unknown n-variable LTF f, outputs an LTF f' that has $\mathbf{P}r[f(x) \neq f'(x)] \leq \epsilon$.

Our results.

In this chapter we give a significantly improved algorithm for the Chow Parameters Problem, whose running time dependence on ϵ is almost doubly exponentially better than the [120] algorithm. Our main result is the following:

Theorem 1 (Main, informal statement)**.** *There is an $\tilde{O}(n^2) \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))} \cdot \log(1/\delta)$-time algorithm $\mathcal{A}$ with the following property: Let $f : \{-1,1\}^n \to \{-1,1\}$ be an LTF and let $0 < \epsilon, \delta < 1/2$. If $\mathcal{A}$ is given as input ϵ, δ and (sufficiently precise estimates of) the Chow parameters of f, then $\mathcal{A}$ outputs integers $v_1, \dots, v_n, \theta$ such that with probability at least $1-\delta$, the linear threshold function $f^* = \mathrm{sign}(v_1x_1 + \cdots + v_nx_n - \theta)$ satisfies $\mathbf{P}r_x[f(x) \neq f^*(x)] \leq \epsilon$.*

Thus we obtain an efficient randomized polynomial approximation scheme (ERPAS) with a *quasi-polynomial* dependence on $1/\epsilon$. We note that for the subclass of LTFs with integer weights of magnitude at most $\mathrm{poly}(n)$, our algorithm runs in $\mathrm{poly}(n/\epsilon)$ time, i.e. it is a *fully* polynomial randomized approximation scheme (FPRAS) (see Section 2.6 for a formal statement). Even for this restricted subclass of LTFs, the algorithm of [120] runs in time doubly exponential in $1/\epsilon$.

Our main result has a range of interesting implications in learning theory. First, it directly gives an efficient algorithm for learning LTFs in the uniform distribution 1-RFA model. Second, it yields a very fast agnostic-type algorithm for learning LTFs in the standard uniform distribution PAC model. Both these algorithms run in time quasi-polynomial in $1/\epsilon$. We elaborate on these learning applications in Section 2.7.

An interesting feature of our algorithm is that it outputs an LTF with integer weights of magnitude at most $\sqrt{n} \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))}$. Hence, as a corollary of our approach, we obtain essentially optimal bounds on approximating arbitrary LTFs using LTFs with small integer weights. It has been known since the 1960s that every n-variable LTF f has an exact representation $\mathrm{sign}(w \cdot x - \theta)$ in which all the weights w_i are integers satisfying $|w_i| \leq 2^{O(n \log n)}$, and Håstad [64] has shown that there is an n-variable LTF f for which *any* integer-weight representation must have each $|w_i| \geq 2^{\Omega(n \log n)}$. However, by settling for an approximate representation (i.e. a representation $f' = \mathrm{sign}(w \cdot x - \theta)$ such that $\mathbf{P}r_x[f(x) \neq f'(x)] \leq \epsilon$), it is possible to get away with much smaller integer weights. Servedio [129] showed that every LTF f can be ϵ-approximated using integer weights each at most $\sqrt{n} \cdot 2^{\tilde{O}(1/\epsilon^2)}$, and this bound was subsequently improved (as a function of ϵ) to $n^{3/2} \cdot 2^{\tilde{O}(1/\epsilon^{2/3})}$ in [38]. (We note that ideas and tools that were developed in work on low-weight approximators for LTFs have proved useful in a range of other contexts, including hardness of approximation [51], property testing [110], and explicit constructions of pseudorandom objects [40].)

Formally, our approach to proving Theorem 1 yields the following nearly-optimal weight bound on ϵ-approximators for LTFs:

Theorem 2 (Low-weight approximators for LTFs)**.** *Let $f : \{-1,1\}^n \to \{-1,1\}$ be any LTF. There is an LTF $f^* = \mathrm{sign}(v_1 x_1 + \cdots + v_n x_n - \theta)$ such that $\mathbf{P}r_x[f(x) \neq f^*(x)] \leq \epsilon$ and the weights v_i are integers that satisfy*

$$\sum_{i=1}^{n} v_i^2 = n \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))}.$$

The bound on the magnitude of the weights in the above theorem is optimal as a function of n and nearly optimal as a function of ϵ. Indeed, as shown in [64, 57], in general any ϵ-approximating LTF f^* for an arbitrary n-variable LTF f may need to have integer weights at least $\max\{\Omega(\sqrt{n}), (1/\epsilon)^{\Omega(\log\log(1/\epsilon))}\}$. Thus, Theorem 2 nearly closes what was previously an almost exponential gap between the known upper and lower bounds for this problem. Moreover, the proof of Theorem 2 is constructive (as opposed e.g. to the one in [38]), i.e. there is a randomized $\mathrm{poly}(n) \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))}$-time algorithm that constructs an ϵ-approximating LTF.

Techniques. We stress that not only are the quantitative results of Theorems 1 and 2 dramatically stronger than previous work, but the proofs are significantly more self-contained and elementary as well. The [120] algorithm relied heavily on several rather sophisticated results on spectral properties of linear threshold functions; moreover, its proof of correctness required a careful retracing of the (rather involved) analysis of a fairly complex property testing algorithm for linear threshold functions given in [110]. In contrast, our proof of Theorem 1 entirely bypasses these spectral results and does not rely on [110] in any way. Turning to low-weight approximators, the improvement from $2^{\tilde{O}(1/\epsilon^2)}$ in [129] to $2^{\tilde{O}(1/\epsilon^{2/3})}$ in [38] required a combination of rather delicate linear programming arguments and powerful results on the anti-concentration of sums of independent random variables due to Halász [63]. In contrast, our proof of Theorem 2 bypasses anti-concentration entirely and does not require any sophisticated linear programming arguments.

Two main ingredients underlie the proof of Theorem 1. The first is a new structural result relating the "Chow distance" and the ordinary (Hamming) distance between two functions f and g, where f is an LTF and g is an arbitrary bounded function. The second is a new and simple algorithm which, given (approximations to) the Chow parameters of an arbitrary Boolean function f, efficiently construct a "linear bounded function" (LBF) g – a certain type of bounded function – whose "Chow distance" from f is small. We describe each of these contributions in more detail below.

The main structural result.

In this subsection we first give the necessary definitions regarding Chow parameters and Chow distance, and then state Theorem 7, our main structural result.

Chow parameters and distance measures.

We formally define the Chow parameters of a function on $\{-1,1\}^n$:

Definition 3. *Given any function* $f : \{-1,1\}^n \to \mathbb{R}$, *its* Chow Parameters *are the rational numbers* $\widehat{f}(0), \widehat{f}(1), \ldots, \widehat{f}(n)$ *defined by* $\widehat{f}(0) = \mathbf{E}[f(x)]$, $\widehat{f}(i) = \mathbf{E}[f(x)x_i]$ *for* $1 \le i \le n$. *We say that the* Chow vector *of* f *is* $\vec{\chi}_f = (\widehat{f}(0), \widehat{f}(1), \ldots, \widehat{f}(n))$.

The Chow parameters naturally induce a distance measure between functions f and g:

Definition 4. *Let* $f, g : \{-1,1\}^n \to \mathbb{R}$. *We define the* Chow distance *between* f *and* g *to be* $d_{\text{Chow}}(f,g) \overset{def}{=} \|\vec{\chi}_f - \vec{\chi}_g\|_2$, *i.e. the Euclidean distance between the Chow vectors.*

This is in contrast with the familiar L_1-distance between functions:

Definition 5. *The* distance *between two functions* $f, g : \{-1,1\}^n \to \mathbb{R}$ *is defined as* $\operatorname{dist}(f,g) \overset{def}{=} \mathbf{E}[|f(x) - g(x)|]$. *If* $\operatorname{dist}(f,g) \le \epsilon$, *we say that* f *and* g *are* ϵ-close.

We note that if f, g are Boolean functions with range $\{-1,1\}$ then $\operatorname{dist}(f,g) = 2\,\mathbf{P}r[f(x) \neq g(x)]$ and thus dist is equivalent (up to a factor of 2) to the familiar Hamming distance.

The main structural result: small Chow-distance implies small distance.

The following fact can be proved easily using basic Fourier analysis (see Proposition 1.5 in [120]):

Fact 6. *Let $f, g : \{-1,1\}^n \to [-1,1]$. We have that $d_{\text{Chow}}(f,g) \le \sqrt{2 \cdot \mathrm{dist}(f,g)}$.*

Our main structural result, Theorem 7, is essentially a converse which bounds $\mathrm{dist}(f,g)$ in terms of $d_{\text{Chow}}(f,g)$ when f is an LTF and g is any bounded function:

Theorem 7 (Main Structural Result)**.** *Let $f : \{-1,1\}^n \to \{-1,1\}$ be an LTF and $g : \{-1,1\}^n \to [-1,1]$ be any bounded function. If $d_{\text{Chow}}(f,g) \le \epsilon$ then*

$$\mathrm{dist}(f,g) \le 2^{-\Omega\left(\sqrt[3]{\log(1/\epsilon)}\right)}.$$

Chow's theorem says that if f is an LTF and g is any bounded function then $d_{\text{Chow}}(f,g) = 0$ implies that $\mathrm{dist}(f,g) = 0$. In light of this, Theorem 7 may be viewed as a "robust" version of Chow's Theorem. Note that the assumption that g is bounded is necessary for the above statement, since the function $g(x) = \sum_{i=0}^{n} \widehat{f}(i) x_i$ (where $x_0 \equiv 1$) satisfies $d_{\text{Chow}}(f,g) = 0$, but has $\mathrm{dist}(f,g) = \Omega(1)$. Results of this sort but with weaker quantitative bounds were given earlier in [14, 57, 129, 120]; we discuss the relationship between Theorem 7 and some of this prior work below.

Discussion. Theorem 7 should be contrasted with Theorem 1.6 of [120], the main structural result of that paper. That theorem says that for $f : \{-1,1\}^n \to \{-1,1\}$ any LTF and $g : \{-1,1\}^n \to [-1,1]$ any bounded function[1], if $d_{\text{Chow}}(f,g) \le \epsilon$ then $\mathrm{dist}(f,g) \le \tilde{O}(1/\sqrt{\log(1/\epsilon)})$. Our new Theorem 7 provides a bound on $\mathrm{dist}(f,g)$ which is almost exponentially stronger than the [120] bound.

Theorem 7 should also be contrasted with Theorem 4 (the main result) of [57], which says that for f an n-variable LTF and g any Boolean function, if $d_{\text{Chow}}(f,g) \le (\epsilon/n)^{O(\log(n/\epsilon)\log(1/\epsilon))}$ then $\mathrm{dist}(f,g) \le \epsilon$. Phrased in this way, Theorem 7 says that for f an LTF and g any bounded function, if $d_{\text{Chow}}(f,g) \le \epsilon^{O(\log^2(1/\epsilon))}$ then $\mathrm{dist}(f,g) \le \epsilon$. So our main structural result may be viewed as an improvement of Goldberg's result that removes its dependence on n. Indeed, this is not a coincidence; Theorem 7 is proved by carefully extending and strengthening Goldberg's arguments using the "critical index" machinery developed in recent studies of structural properties of LTFs [129, 120, 40].

It is natural to wonder whether the conclusion of Theorem 7 can be strengthened to "$\mathrm{dist}(f,g) \le \epsilon^c$" where $c > 0$ is some absolute constant. We show that no such strengthening is possible, and in fact, no conclusion of the form "$\mathrm{dist}(f,g) \le 2^{-\gamma(\epsilon)}$" is possible for any function $\gamma(\epsilon) = \omega(\log(1/\epsilon)/\log\log(1/\epsilon))$; we prove this in Section 2.6.

[1]The theorem statement in [120] actually requires that g have range $\{-1,1\}$, but the proof is easily seen to extend to $g : \{-1,1\}^n \to [-1,1]$ as well.

The algorithmic component.

A straightforward inspection of the arguments in [120] shows that by using our new Theorem 7 in place of Theorem 1.6 of that paper throughout, the running time of the [120] algorithm can be improved to $\mathrm{poly}(n) \cdot 2^{(1/\epsilon)^{O(\log^2(1/\epsilon))}}$. This is already a significant improvement over the $\mathrm{poly}(n) \cdot 2^{2^{\tilde{O}(1/\epsilon^2)}}$ running time of [120], but is significantly worse than the $\mathrm{poly}(n) \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))}$ running time which is our ultimate goal.

The second key ingredient of our results is a new algorithm for constructing an LTF from the (approximate) Chow parameters of an LTF f. The previous approach to this problem [120] constructed an LTF with Chow parameters close to $\vec{\chi}_f$ directly and applied the structural result to the constructed LTF. Instead, our approach is based on the insight that it is substantially easier to find a bounded real-valued function g that is close to f in Chow distance. The structural result can then be applied to g to conclude that g is close to f in L_1-distance. The problem with this idea is, of course, that we need an LTF that is close to f and not a general bounded function. However, we show that it is possible to find g which is a "linear bounded function" (LBF), a type of bounded function closely related to LTFs. An LBF can then be easily converted to an LTF with only a small increase in distance from f. We now proceed to define the notion of an LBF and state our main algorithmic result formally. We first need to define the notion of a projection:

Definition 8. *For a real value a, we denote its projection to $[-1, 1]$ by $P_1(a)$. That is, $P_1(a) = a$ if $|a| \leq 1$ and $P_1(a) = \mathrm{sign}(a)$, otherwise.*

Definition 9. *A function $g : \{-1, 1\}^n \to [-1, 1]$ is referred to as a* linear bounded function *(LBF) if there exists a vector of real values $w = (w_0, w_1, \ldots, w_n)$ such that $g(x) = P_1(w_0 + \sum_{i=1}^n w_i x_i)$. The vector w is said to represent g.*

We are now ready to state our main algorithmic result:

Theorem 10 (Main Algorithmic Result)**.** *There exists a randomized algorithm* `ChowReconstruct` *that for every Boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$, given $\epsilon > 0, \delta > 0$ and a vector $\vec{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_n)$ such that $\|\vec{\chi}_f - \vec{\alpha}\| \leq \epsilon$, with probability at least $1 - \delta$, outputs an LBF g such that $\|\vec{\chi}_f - \vec{\chi}_g\| \leq 6\epsilon$. The algorithm runs in time $\tilde{O}(n^2 \epsilon^{-4} \log(1/\delta))$. Further, g is represented by a weight vector $\kappa v \in \mathbb{R}^{n+1}$, where $\kappa \in \mathbb{R}$ and v is an integer vector with $\|v\| = O(\sqrt{n}/\epsilon^3)$.*

We remark that the condition on the weight vector v given by Theorem 10 is the key for the proof of Theorem 2.

The way we use `ChowReconstruct` is to construct an LBF g whose Chow distance from f is small enough to ensure that $\mathrm{dist}(f, g)$ is at most ϵ. For general LTFs, this upper bound on $\mathrm{dist}(f, g)$ is given by Theorem 7; however in special cases other structural results may give even stronger bounds. In particular, a structural result of [14] gives that if f is an LTF with integer weights of magnitude bounded by $\mathrm{poly}(n)$, then as long as the Chow distance between f and g is $\epsilon/\mathrm{poly}(n)$, it must be the case that $\mathrm{dist}(f, g) \leq \epsilon$. Hence our algorithm performs extremely well for such LTFs f: given the (approximate) Chow parameters of an LTF f with $\mathrm{poly}(n)$ integer weights, it outputs an LBF g with $\mathrm{dist}(f, g) \leq \epsilon$. Given g, it is trivial to obtain a LTF f^* such that $\mathrm{dist}(f, f^*) \leq 2\epsilon$. Thus, for $\mathrm{poly}(n)$-weight LTFs, we obtain a FPRAS. (See Theorem 34 for a detailed statement of this result.

Discussion. It is interesting to note that the approach underlying Theorem 10 is much more efficient and significantly simpler than the algorithmic approach of [120]. The algorithm in [120] roughly works as follows: In the first step, it constructs a "small" set of candidate LTFs such that at least one of them is close to f, and in the second step it identifies such an LTF by searching over all such candidates. The first step proceeds by enumerating over "all" possible weights assigned to the "high influence" variables. This brute force search makes the [120] algorithm very inefficient. Moreover, its proof of correctness requires some sophisticated spectral results from [110], which make the approach rather complicated.

In this work, our algorithm is based on a boosting-based approach, which is novel in this context. Our approach is much more efficient than the brute force search of [120] and its analysis is much simpler, since it completely bypasses the spectral results of [110]. We also note that the algorithm of [120] crucially depends on the fact that the relation between Chow distance and distance has no dependence on n. (If this was not the case, the approach would not lead to a polynomial time algorithm.) Our boosting-based approach is quite robust, as it has no such limitation. This fact is crucial for us to obtain the aforementioned FPRAS for small-weight LTFs.

While we are not aware of any prior results similar to Theorem 10 being stated explicitly, we note that weaker forms of our theorem can be obtained from known results. In particular, Trevisan et. al. [139] describe an algorithm that given oracle access to a Boolean function f, $\epsilon' > 0$, and a set of functions $H = \{h_1, h_2, \ldots, h_k\}$, efficiently finds a bounded function g that for every $i \leq n$ satisfies $|\mathbf{E}[f \cdot h_i] - \mathbf{E}[g \cdot h_i]| \leq \epsilon'$. One can observe that if $H = \{1, x_1, \ldots, x_n\}$, then the function g returned by their algorithm is in fact an LBF and that the oracle access to f can be replaced with approximate values of $\mathbf{E}[f \cdot h_i]$ for every i. Hence, the algorithm in [139], applied to the set of functions $H = \{1, x_1, x_2, \ldots, x_n\}$, would find an LBF g which is close in Chow distance to f. A limitation of this algorithm is that, in order to obtain an LBF which is Δ-close in Chow distance to f, it requires that every Chow parameter of f be given to it with accuracy of $O(\Delta/\sqrt{n})$. In contrast, our algorithm only requires that the total distance of the given vector to $\vec{\chi}_f$ is at most $\Delta/6$. In addition, the bound on the integer weight approximation of LTFs that can be obtained from the algorithm in [139] is linear in $n^{3/2}$, whereas we obtain the optimal dependence of $\sqrt{n}$.

The algorithm in [139] is a simple adaptation of the hardcore set construction technique of Impagliazzo [71]. Our algorithm is also based on the ideas from [71] and, in addition, uses ideas from the distribution-specific boosting technique in [49].

Our algorithm can be seen as an instance of a more general approach to learning (or approximating) a function that is based on constructing a bounded function with the given Fourier coefficients. Another instance of this new approach is the recent algorithm for learning a certain class of polynomial threshold functions (which includes polynomial-size DNF formulae) from low-degree Fourier coefficients [50]. We note that the algorithm in [50] is based on an algorithm similar to ours. However, like the algorithm in [139], it requires that every low-degree Fourier coefficient be given to it with high accuracy. As a result it would be similarly less efficient in our application.

Organization. In Section 2.1 we record some mathematical preliminaries that will be used throughout the chapter. In Section 2.2 we present some observations regarding the complexity of solving the Chow parameters problem exactly and give an LP–based $2^{O(n)}$-time algorithm for it. Sections 2.3 and 2.4 contain the proof of our main structural result (Theorem 7). In Section 2.5 we present our main algorithmic ingredient (Theorem 10). Section 2.6 puts the pieces together and proves our main theorem (Theorem 1) and our other main result (Theorem 2), while Section 2.7 presents the consequences of our results to learning theory. Finally, in Section **??** we conclude the chapter and present a few interesting research directions.

2.1 Mathematical Preliminaries

Probabilistic Facts.

We require some basic probability results including the standard additive Hoeffding bound:

Theorem 11. *Let $X_1, \dots, X_n$ be independent random variables such that for each $j \in [n]$, X_j is supported on $[a_j, b_j]$ for some $a_j, b_j \in \mathbb{R}$, $a_j \le b_j$. Let $X = \sum_{j=1}^n X_j$. Then, for any $t > 0$, $\mathbf{P}r\left[|X - \mathbf{E}[X]| \ge t\right] \le 2\exp\left(-2t^2/\sum_{j=1}^n (b_j - a_j)^2\right)$.*

The Berry-Esséen theorem (see e.g. [52]) gives explicit error bounds for the Central Limit Theorem:

Theorem 12. *(Berry-Esséen) Let $X_1, \dots, X_n$ be independent random variables satisfying $\mathbf{E}[X_i] = 0$ for all $i \in [n]$, $\sqrt{\sum_i \mathbf{E}[X_i^2]} = \sigma$, and $\sum_i \mathbf{E}[|X_i|^3] = \rho_3$. Let $S = (X_1 + \cdots + X_n)/\sigma$ and let F denote the cumulative distribution function (cdf) of S. Then $\sup_x |F(x) - \Phi(x)| \le \rho_3/\sigma^3$ where Φ denotes the cdf of the standard gaussian random variable.*

For us, the most important consequence of the Berry-Esséen theorem is its application in proving anti-concentration for a weighted sum of Bernoulli random variables. To describe the application, we need to define the notion of regularity for a vector in $\mathbb{R}^n$.

Definition 13 (regularity). *Fix $\tau > 0$. We say that a vector $w = (w_1, \dots, w_n) \in \mathbb{R}^n$ is τ*-regular *if $\max_{i \in [n]} |w_i| \le \tau \|w\| = \tau\sqrt{w_1^2 + \cdots + w_n^2}$. A linear form $w \cdot x$ is said to be τ-regular if w is τ-regular, and similarly an LTF is said to be τ-regular if it is of the form $\mathrm{sign}(w \cdot x - \theta)$ where w is τ-regular.*

Regularity is a helpful notion because if w is τ-regular then the Berry-Esséen theorem (stated below) tells us that for uniform $x \in \{-1, 1\}^n$, the linear form $w \cdot x$ is "distributed like a Gaussian up to error τ." This can be useful for many reasons; in particular, it will let us exploit the strong anti-concentration properties of the Gaussian distribution. The next fact states this precisely.

Fact 14. *Let $w = (w_1, \dots, w_n)$ be a τ-regular vector in $\mathbb{R}^n$ and write σ to denote $\|w\|_2$. Then for any interval $[a, b] \subseteq \mathbb{R}$, we have $\left|\mathbf{P}r[\sum_{i=1}^n w_i x_i \in (a, b]] - \Phi([a/\sigma, b/\sigma])\right| \le 2\tau$, where*

$\Phi([c,d]) \overset{def}{=} \Phi(d) - \Phi(c)$. *In particular, it follows that*

$$\mathbf{P}r\left[\sum_{i=1}^{n} w_i x_i \in (a,b]\right] \leq |b-a|/\sigma + 2\tau.$$

Useful inequality.

We will need the following elementary inequality.

Fact 15. *For* $a, b \in (0,1)$, $(ab)^{\log(1/a)+\log(1/b)} \geq a^{2\log(1/a)} \cdot b^{2\log(1/b)}$.

Proof.

$$\begin{aligned} (ab)^{\log(1/a)+\log(1/b)} &= 2^{-\log^2(1/a)-\log^2(1/b)-2\log(1/a)\cdot\log(1/b)} \\ &\geq 2^{-2\log^2(1/a)-2\log^2(1/b)} \\ &= a^{2\log(1/a)} \cdot b^{2\log(1/b)}, \end{aligned}$$

where the inequality is the arithmetic-geometric mean inequality. □

Useful facts about affine spaces.

A subset $V \subseteq \mathbb{R}^n$ is said to be an *affine subspace* if it is closed under affine combinations of vectors in V. Equivalently, V is an affine subspace of $\mathbb{R}^n$ if $V = X + b$ where $b \in \mathbb{R}^n$ and X is a linear subspace of $\mathbb{R}^n$. The affine dimension of V is the same as the dimension of the linear subspace X. A hyperplane in $\mathbb{R}^n$ is an affine space of dimension $n-1$. Throughout the chapter we use bold capital letters such as $\mathbf{H}$ to denote hyperplanes.

In this chapter whenever we refer to a "subspace" we mean an affine subspace unless explicitly otherwise indicated. The dimension of an affine subspace V is denoted by $\dim(V)$. Similarly, for a set $S \subseteq \mathbb{R}^n$, we write $\mathrm{span}(S)$ to denote the affine span of S, i.e.

$$\mathrm{span}(S) = \{s + \sum_{i=1}^{m} w_i(x^i - y^i) \mid s, x^i, y^i \in S, w_i \in \mathbb{R}, m \in \mathbb{N}\}.$$

The following very useful fact about affine spaces was proved by Odlyzko[118].

Fact 16. *[118] Any affine subspace of* $\mathbb{R}^n$ *of dimension* d *contains at most* 2^d *elements of* $\{-1,1\}^n$.

2.2 On the Exact Chow Parameters Problem

In this section we make some observations regarding the complexity of the exact version of the Chow parameters problem and present a simple (albeit exponential time) algorithm for it, that beats brute-force search.

Proof of Chow's Theorem.

For completeness we state and prove Chow's theorem here:

Theorem 17 ([23]). *Let $f : \{-1,1\}^n \to \{-1,1\}$ be an LTF and let $g : \{-1,1\}^n \to [-1,1]$ be a bounded function such that $\widehat{g}(j) = \widehat{f}(j)$ for all $0 \le j \le n$. Then $g = f$.*

Proof. Write $f(x) = \mathrm{sign}(w_0 + w_1x_1 + \cdots + w_nx_n)$, where the weights are scaled so that $\sum_{j=0}^n w_j^2 = 1$. We may assume without loss of generality that $|w_0 + w_1x_1 + \cdots + w_nx_n| \neq 0$ for all x. (If this is not the case, first translate the separating hyperplane by slightly perturbing w_0 to make it hold; this can be done without changing f's value on any point of $\{-1,1\}^n$.) Now we have

$$\begin{aligned} 0 &= \sum_{j=0}^{n} w_j(\widehat{f}(j) - \widehat{g}(j)) \\ &= \mathbf{E}[(w_0 + w_1x_1 + \cdots + w_nx_n)(f(x) - g(x))] \\ &= \mathbf{E}[|f(x) - g(x)| \cdot |w_0 + w_1x_1 + \cdots + w_nx_n|]. \end{aligned}$$

The first equality is by the assumption that $\widehat{f}(j) = \widehat{g}(j)$ for all $0 \le j \le n$, the second equality is linearity of expectation (or Plancherel's identity), and the third equality uses the fact that

$$\mathrm{sign}(f(x) - g(x)) = f(x) = \mathrm{sign}(w_0 + w_1x_1 + \cdots + w_nx_n)$$

for any bounded function g with range $[-1,1]$. But since $|w_0 + w_1x_1 + \cdots + w_nx_n|$ is always strictly positive, we must have $\mathbf{P}r[f(x) \neq g(x)] = 0$ as claimed. □

An exact $2^{O(n)}$–time algorithm.

Let us start by pointing out that it is unlikely that the Chow Parameters problem can be solved exactly in polynomial time. Note that even checking the correctness of a candidate solution is $\sharp P$-complete, because computing $\widehat{f}(0)$ is equivalent to counting 0-1 knapsack solutions. This suggests (but does not logically imply) that the exact problem is intractable; characterizing its complexity is an interesting open problem (see Section **??**).

The naive brute-force approach (enumerate all possible n-variable LTFs, and for each one check whether it has the desired Chow parameters) requires $2^{\Theta(n^2)}$ time. The following proposition gives an improved (albeit exponential time) algorithm:

Proposition 18. *The Chow parameters problem can be solved exactly in time $2^{O(n)}$.*

Proof. Let α_i, $i = 0, 1, \dots, n$ be the target Chow parameters; we are given the promise that there exists an LTF $f : \{-1,1\}^n \to \{-1,1\}$ such that $\widehat{f}(i) = \alpha_i$ for all i. Our goal is to output (a weights-based representation of) the function f. Let $g : \{-1,1\}^n \to [-1,1]$ be a bounded function that has the same Chow parameters as f. We claim that there exists a linear program with 2^n variables and $O(2^n)$ constraints encoding the truth-table of g. Indeed, for every $x \in$

$\{-1,1\}^n$ we have a variable $g(x)$ and the constraints are as follows: For all $x \in \{-1,1\}^n$ we include the constraint $-1 \le g(x) \le 1$. We also include the $(n+1)$ constraints $\mathbf{E}_x[g(x)x_i] \equiv 2^{-n}\sum_{x\in\{-1,1\}^n} g(x)x_i = \alpha_i$, $i = 0,1,\ldots,n$ (where $x_0 \equiv 1$). Chow's theorem stated above implies that the aforementioned linear program has a *unique* feasible solution, corresponding to the truth table of the target LTF f. That is, the unique solution of the linear program will be integral and is identical to the target function. Since the size of the linear program is $2^{O(n)}$ and linear programming is in P, the truth table of f can thus be computed in time $2^{O(n)}$.

A weight-based representation of f as $\mathrm{sign}(w\cdot x - \theta)$ can then be obtained straightforwardly in time $2^{O(n)}$ by solving another linear program with variables (w,θ) and 2^n constraints, one for each $x \in \{-1,1\}^n$. □

We point out that our main algorithmic result also yields an algorithm for the exact Chow parameters problem that beats brute-force search, in particular it runs in time $2^{O(n\log n)}$. (See Theorem 34 and the remark following its statement.)

2.3 Proof overview of main structural result: Theorem 7

In this section we provide a detailed overview of the proof of Theorem 7, restated here for convenience:

Theorem 7 (Main Structural Result). *Let* $f : \{-1,1\}^n \to \{-1,1\}$ *be an LTF and* $g : \{-1,1\}^n \to [-1,1]$ *be any bounded function. If* $d_{\mathrm{Chow}}(f,g) \le \epsilon$ *then* $\mathrm{dist}(f,g) \le 2^{-\Omega\left(\sqrt[3]{\log(1/\epsilon)}\right)}$.

We give an informal overview of the main ideas of the proof of Theorem 7 in Section 2.3, and then proceed with a detailed outline of Theorem 7 in Section 2.3.

Informal overview of the proof.

We first note that throughout the informal explanation given in this subsection, for the sake of clarity we restrict our attention to the case in which $g : \{-1,1\}^n \to \{-1,1\}$ is a Boolean rather than a bounded function. In the actual proof we deal with bounded functions using a suitable weighting scheme for points of $\{-1,1\}^n$ (see the discussion before Fact 26 near the start of the proof of Theorem 7).

To better explain our approach, we begin with a few words about how Theorem 1.6 of [120] (the only previously known statement of this type that is "independent of n") is proved. The key to that theorem is a result on approximating LTFs using LTFs with "good anti-concentration"; more precisely, [120] shows that for any LTF f there is an LTF $f'(x) = \mathrm{sign}(v\cdot x - \nu)$, $\|v\| = 1$, that is extremely close to f (Hamming distance roughly $2^{-1/\epsilon}$) and which has "moderately good anti-concentration at radius ϵ," in the sense that $\mathbf{P}r[|v\cdot x - \nu| \le \epsilon] \le \tilde{O}(1/\sqrt{\log(1/\epsilon)})$. Given this, Theorem 1.6 of [120] is proved using a modification of the proof of the original Chow's Theorem. However, for this approach based on the original Chow proof to work, it is crucial that the Hamming distance between f and f' (namely $2^{-1/\epsilon}$) be very small compared to the anti-concentration radius (which is ϵ). Subject to this constraint it seems very difficult to give a significant quantitative improvement of the approximation result in a way that would improve the bound of Theorem 1.6 of [120].

Instead, we hew more closely to the approach used to prove Theorem 4 of [57]. This approach also involves a perturbation of the LTF f, but instead of measuring closeness in terms of Hamming distance, a more direct geometric view is taken. In the rest of this subsection we give a high-level explanation of Goldberg's proof and of how we modify it to obtain our improved bound.

The key to Goldberg's approach is a (perhaps surprising) statement about the geometry of hyperplanes as they relate to the Boolean hypercube. He establishes the following key geometric result (see Theorem 21 for a precise statement):

> If $\mathbf{H}$ is any n-dimensional hyperplane such that an α fraction of points in $\{-1, 1\}^n$ lie "very close" in Euclidean distance (essentially $1/\mathrm{quasipoly}(n/\alpha)$) to $\mathbf{H}$, then there is a hyperplane $\mathbf{H}'$ which actually *contains* all those $\alpha 2^n$ points of the hypercube.

With this geometric statement in hand, an iterative argument is used to show that if the Hamming distance between LTF f and Boolean function g is large, then the Euclidean distance between the centers of mass of (the positive examples for f on which f and g differ) and (the negative examples for f on which f and g differ) must be large; finally, this Euclidean distance between centers of mass corresponds closely to the Chow distance between f and g.

However, the $1/\mathrm{quasipoly}(n)$ closeness requirement in the key geometric statement means that Goldberg's Theorem 4 not only depends on n, but this dependence is superpolynomial. The heart of our improvement is to combine Goldberg's key geometric statement with ideas based on the "critical index" of LTFs to get a version of the statement which is completely independent of n. Roughly speaking, our analogue of Goldberg's key geometric statement is the following (a precise version is given as Lemma 22 below):

> If $\mathbf{H}$ is any n-dimensional hyperplane such that an α fraction of points in $\{-1, 1\}^n$ lie within Euclidean distance $\alpha^{O(\log(1/\alpha))}$ of $\mathbf{H}$, then there is a hyperplane $\mathbf{H}'$ which contains *all but a tiny fraction* of those $\alpha 2^n$ points of the hypercube.

Our statement is stronger than Goldberg's in that there is no dependence on n in the distance bound from $\mathbf{H}$, but weaker in that we do not guarantee $\mathbf{H}'$ passes through every point; it may miss a tiny fraction of points. We are able to handle the effect of missing points in the subsequent analysis. Armed with this improvement, a careful sharpening of Goldberg's iterative argument (to get rid of another dependence on n, unrelated to the tiny fraction of points missed by $\mathbf{H}'$) lets us prove Theorem 7.

Detailed outline of the proof.

As discussed in Section 2.3, the key to proving Theorem 7 is an improvement of Theorem 3 in [57].

Definition 19. *Given a hyperplane $\mathbf{H}$ in $\mathbb{R}^n$ and $\beta > 0$, the β-*neighborhood *of $\mathbf{H}$ is defined as the set of points in $\mathbb{R}^n$ at Euclidean distance at most β from $\mathbf{H}$.*

We recall the following fact which shows how to express the Euclidean distance of a point from a hyperplane using the standard representation of the hyperplane:

Fact 20. *Let $\mathbf{H} = \{x : w \cdot x - \theta = 0\}$ be a hyperplane in $\mathbb{R}^n$ where $\|w\| = 1$. Then for any $x \in \mathbb{R}^n$, the Euclidean distance $d(x, \mathbf{H})$ of x from $\mathbf{H}$ is $|w \cdot x - \theta|$.*

Theorem 21 (Theorem 3 in [57]). *Given any hyperplane in $\mathbb{R}^n$ whose β-neighborhood contains a subset S of vertices of $\{-1,1\}^n$, where $|S| = \alpha \cdot 2^n$, there exists a hyperplane which contains all elements of S provided that*

$$0 \le \beta \le \left((2/\alpha) \cdot n^{5+\lfloor \log(n/\alpha) \rfloor} \cdot (2 + \lfloor \log(n/\alpha) \rfloor)!\right)^{-1}.$$

Before stating our improved version of the above theorem, we define the set $U = \cup_{i=1}^n \mathbf{e}_i \cup \{\mathbf{0}\}$ where $\mathbf{0} \in \mathbb{R}^n$ is the all zeros vector and $\mathbf{e}_i \in \mathbb{R}^n$ is the unit vector in the i^{th} direction.

Our improved version of Theorem 21 is the following:

Lemma 22. *There exist a constant C_1 such that for every hyperplane $\mathbf{H}$ in $\mathbb{R}^n$ whose β-neighborhood contains a subset S of vertices of $\{-1,1\}^n$, where $|S| = \alpha \cdot 2^n$ and any $0 < \kappa < \alpha/2$, there exists a hyperplane $\mathbf{H}'$ in $\mathbb{R}^n$ that contains a subset $S^* \subseteq S$ of cardinality at least $(\alpha - \kappa) \cdot 2^n$ provided that*

$$0 < \beta \le \beta_0 \overset{def}{=} (\log(1/\kappa))^{-1/2} \cdot 2^{-\sqrt{\log\log(1/\kappa)}} \cdot \alpha^{C_1 \cdot \log(1/\alpha)}.$$

Moreover, the coefficient vector defining $\mathbf{H}'$ has at most

$$C_1 \cdot (1/\alpha^2) \cdot (\log\log(1/\kappa) + \log^2(1/\alpha))$$

nonzero coordinates. Further, for any $x \in U$, if x lies on $\mathbf{H}$ then x lies on $\mathbf{H}'$ as well.

Discussion. We note that while Lemma 22 may appear to be incomparable to Theorem 21 because it "loses" $\kappa 2^n$ points from the set S, in fact by taking $\kappa = 1/2^{n+1}$ it must be the case that our S^* is the same as S, and with this choice of κ, Lemma 22 gives a strict quantitative improvement of Theorem 21. (We stress that for our application, though, it will be crucial for us to use Lemma 22 by setting the κ parameter to depend only on α independent of n.) We further note that in any statement like Lemma 22 that does not "lose" any points from S, the bound on β must necessarily depend on n; we show this in Appendix A.1. Finally, the condition at the end of Lemma 22 (that if $x \in U$ lies on $\mathbf{H}$, then it lies on $\mathbf{H}'$ as well) allows us to obtain an analogous result in any affine subspace of $\mathbb{R}^n$ instead of $\mathbb{R}^n$. This is necessary for the iterative application of Lemma 22 in the proof of Theorem 7.

We give the detailed proof of Lemma 22 in Section 2.4. We now briefly sketch the main idea underlying the proof of the lemma. At a high level, the proof proceeds by reducing the number of variables from n down to $m = O\left((1/\alpha^2) \cdot \log(1/\beta)\right)$ followed by an application of Theorem 155, a generalization of Theorem 21 proved in Appendix A.2, in $\mathbb{R}^m$. (As we will see later, we use Theorem 155 instead of Theorem 21 because we need to ensure that points of U which lie on $\mathbf{H}$ continue to lie on $\mathbf{H}'$.) The reduction uses the notion of the τ-critical index applied to the vector w defining $\mathbf{H}$. (See Section 2.4 for the relevant definitions.)

The idea of the proof is that for coordinates i in the "tail" of w (intuitively, where $|w_i|$ is small) the value of x_i does not have much effect on $d(x, \mathbf{H})$, and consequently the condition of the lemma must hold true in a space of much lower dimension than n. To show that tail coordinates of x do not have much effect on $d(x, \mathbf{H})$, we do a case analysis based on the τ-critical index $c(w, \tau)$ of w to show that (in both cases) the 2-norm of the entire "tail" of w must be small. If $c(w, \tau)$ is large, then this fact follows easily by properties of the τ-critical index. On the other hand, if $c(w, \tau)$ is small we argue by contradiction as follows: By the definition of the τ-critical index and the Berry-Esséen theorem, the "tail" of w (approximately) behaves like a normal random variable with standard deviation equal to its 2-norm. Hence, if the 2-norm was large, the entire linear form $w \cdot x$ would have good anti-concentration, which would contradict the assumption of the lemma. Thus in both cases, we can essentially ignore the tail and make the effective number of variables be m which is independent of n.

As described earlier, we view the geometric Lemma 22 as the key to the proof of Theorem 7; however, to obtain Theorem 7 from Lemma 22 requires a delicate iterative argument, which we give in full in the following section. This argument is essentially a refined version of Theorem 4 of [57] with two main modifications: one is that we generalize the argument to allow g to be a bounded function rather than a Boolean function, and the other is that we get rid of various factors of $\sqrt{n}$ which arise in the [57] argument (and which would be prohibitively "expensive" for us). We give the detailed proof in Section 2.4.

2.4 Proof of Theorem 7

In this section we provide a detailed proof of our main structural result (Theorem 7).

Useful Technical Tools.

As described above, a key ingredient in the proof of Theorem 7 is the notion of the "critical index" of an LTF f. The critical index was implicitly introduced and used in [129] and was explicitly used in [38, 40, 120] and other works. To define the critical index we need to first recall the definition of "regularity" (see Definition 13). Intuitively, the critical index of w is the first index i such that from that point on, the vector $(w_i, w_{i+1}, \ldots, w_n)$ is regular. A precise definition follows:

Definition 23 (critical index). *Given a vector $w \in \mathbb{R}^n$ such that $|w_1| \geq \cdots \geq |w_n| > 0$, for $k \in [n]$ we denote by σ_k the quantity $\sqrt{\sum_{i=k}^n w_i^2}$. We define the τ-*critical index $c(w, \tau)$ of w *as the smallest index $i \in [n]$ for which $|w_i| \leq \tau \cdot \sigma_i$. If this inequality does not hold for any $i \in [n]$, we define $c(w, \tau) = \infty$.*

The following simple fact states that the "tail weight" of the vector w decreases exponentially prior to the critical index:

Fact 24. *For any vector $w = (w_1, \dots, w_n)$ such that $|w_1| \geq \cdots \geq |w_n| > 0$ and $1 \leq a \leq c(w, \tau)$, we have $\sigma_a < (1 - \tau^2)^{(a-1)/2} \cdot \sigma_1$.*

Proof. If $a < c(w, \tau)$, then by definition $|w_a| > \tau \cdot \sigma_a$. This implies that $\sigma_{a+1} < \sqrt{1 - \tau^2} \cdot \sigma_a$. Applying this inequality repeatedly, we get that $\sigma_a < (1 - \tau^2)^{(a-1)/2} \cdot \sigma_1$ for any $1 \leq a \leq c(w, \tau)$. $\square$

Proof of Lemma 22.

Let $\mathbf{H} = \{x \in \mathbb{R}^n \mid w \cdot x = \theta\}$ where we can assume (by rescaling) that $\|w\|_2 = 1$ and (by reordering the coordinates) that $|w_1| \geq |w_2| \geq \ldots \geq |w_n|$. Note that the Euclidean distance of any point $x \in \mathbb{R}^n$ from $\mathbf{H}$ is $|w \cdot x - \theta|$. Let us also define $V \overset{\text{def}}{=} \mathbf{H} \cap U$. Set $\tau \overset{\text{def}}{=} \alpha/4$ (for conceptual clarity we will continue to use "τ" for as long as possible in the arguments below). We note that we can assume that all weights are non-zero since we can project the problem to coordinates where $\mathbf{H}$ has non-zero weights. This does not affect distances or our bounds. We can therefore define the τ-critical index $c(w, \tau)$ of the vector $w \in \mathbb{R}^n$.

Fix the parameter $K_0 \overset{\text{def}}{=} C_2 \cdot (1/\tau^2) \cdot \log(1/\beta)$ for a constant C_2 to be chosen later and let $K_1 = \min\{c(w, \tau), K_0\}$. We partition $[n]$ into a set of "head" coordinates $H = [K_1]$ and a complementary set of "tail" coordinates $T = [n] \setminus H$. Writing w as (w_H, w_T) and likewise for x. (We can assume that $K_1 \leq n$ since otherwise the lemma follows immediately from Theorem 21.) We now prove by case analysis that $\|w_T\|_2$ must be small.

Claim 25. *We have $\|w_T\|_2 \leq 8\beta/\alpha$.*

Proof.
Case I: $c(w, \tau) > K_0$. In this case, $|H| = C_2 \cdot (1/\tau^2) \cdot \log(1/\beta)$ and it follows from Fact 24 that for large enough constant C_2, $\|w_T\| \leq \beta \leq 8\beta/\alpha$.
Case II: $c(w, \tau) \leq K_0$. In this case, $|H| = c(w, \tau)$. We use the fact that w_T is τ-regular to deduce that the norm of the tail must be small.

Suppose for the sake of contradiction that

$$\|w_T\|_2 > 2\beta/(\alpha - 3\tau) = 8\beta/\alpha.$$

By the Berry-Esséen theorem (Theorem 12, or more precisely Fact 14), for all $\delta > 0$ we have

$$\sup_{t \in \mathbb{R}} \underset{x_T}{\mathbf{P}r}\left[|w_T \cdot x_T - t| < \delta\right] \leq \frac{2\delta}{\|w_T\|} + 2\tau.$$

By setting $\delta \overset{\text{def}}{=} (\alpha - 3\tau)\|w_T\|/2 > \beta$ we get that

$$\sup_{t \in \mathbb{R}} \underset{x_T}{\mathbf{P}r}\left[|w_T \cdot x_T - t| < \delta\right] < \alpha,$$

and consequently

$$\begin{aligned}\mathop{\mathbf{P}r}_{x}[|w \cdot x - \theta| \leq \beta] &\leq \sup_{t \in \mathbb{R}} \mathop{\mathbf{P}r}_{x_T}[|w_T \cdot x_T - t| \leq \beta] \\ &\leq \sup_{t \in \mathbb{R}} \mathop{\mathbf{P}r}_{x_T}\left[|w_T \cdot x_T - t| < \delta\right] \\ &< \alpha\end{aligned}$$

which contradicts the existence of the set S in the statement of the lemma. □

By the Hoeffding bound, for $1 - \kappa$ fraction of $x \in \{-1, 1\}^n$ we have

$$|w_H \cdot x_H - \theta| \leq |w \cdot x - \theta| + |w_T \cdot x_T| \leq |w \cdot x - \theta| + \beta'$$

where $\beta' = C_3 \cdot (\beta/\alpha) \cdot \sqrt{\log(1/\kappa)}$ for a sufficiently large constant C_3.

By the assumption of the lemma, there exists a set $S \subseteq \{-1, 1\}^n$ of cardinality at least $\alpha \cdot 2^n$ such that for all $x \in S$ we have $|w \cdot x - \theta| \leq \beta$. A union bound and the above inequality imply that there exists a set $S^* \subseteq S$ of cardinality at least $(\alpha - \kappa) \cdot 2^n$ with the property that for all $x \in S^*$, we have

$$|w_H \cdot x_H - \theta| \leq \beta + \beta'.$$

Also, any $x \in U$ satisfies $\|x_T\| \leq 1$. Hence for any $x \in V$, we have that

$$\begin{aligned}|w_H \cdot x_H - \theta| &\leq |w \cdot x - \theta| + |w_T \cdot x_T| = |w_T \cdot x_T| \\ &\leq \|w_T\| \cdot \|x_T\| \leq 8\beta/\alpha \leq \beta'.\end{aligned}$$

Define the projection mapping $\phi_H : \mathbb{R}^n \to \mathbb{R}^{|H|}$ by $\phi_H : x \mapsto x_H$ and consider the image of S^*, i.e. $S' \stackrel{\text{def}}{=} \phi_H(S^*)$. It is clear that $|S'| \geq (\alpha - \kappa) \cdot 2^{|H|}$ and that for all $x_H \in S'$, we have

$$|w_H \cdot x_H - \theta| \leq \beta + \beta' \leq 2\beta'.$$

Similarly, if V' is the image of V under ϕ_H, then for every $x_H \in V'$ we have $|w_H \cdot x_H - \theta| \leq \beta'$. It is also clear that $\|w_T\| < 1/2$ and hence $\|w_H\| > 1/2$. Thus for every $x_H \in (S' \cup V')$ we have

$$\left| \frac{w_H \cdot x_H}{\|w_H\|} - \frac{\theta}{\|w_H\|} \right| \leq 4\beta'.$$

We now define the $|H|$-dimensional hyperplane $\mathbf{H}_H$ as $\mathbf{H}_H \stackrel{\text{def}}{=} \{x_H \in \mathbb{R}^{|H|} \mid w_H \cdot x_H = \theta\}$. As all points in $S' \cup V'$ are in the $4\beta'$-neighborhood of $\mathbf{H}_H$, we may now apply Theorem 155 for the hyperplane $\mathbf{H}_H$ over $\mathbb{R}^{|H|}$ to deduce the existence of an alternate hyperplane $\mathbf{H}'_H = \{x_H \in \mathbb{R}^{|H|} \mid v_H \cdot x_H = \nu\}$ that contains all points in $S' \cup V'$. The only condition we need to verify in order that Theorem 155 may be applied is that $4\beta'$ is upper bounded by

$$\beta_1 \stackrel{\text{def}}{=} \left(\frac{2}{\alpha - \kappa} \cdot |H|^{5 + \lfloor \log(|H|/(\alpha - \kappa)) \rfloor} \cdot (2 + \lfloor \log(|H|/(\alpha - \kappa)) \rfloor)! \right)^{-1}.$$

Recalling that, $|H| \leq K_0$ and $\kappa < \alpha/2$, we obtain that $\beta_1 \leq (\alpha/K_0)^{C_4 \log(K_0/\alpha)}$ for some large enough constant C_4. Using $K_0 = C_2 \cdot (4/\alpha)^2 \cdot \log(1/\beta)$ and $\beta' = C_3\beta\sqrt{\log(1/\kappa)}/\alpha$, we need to verify that

$$\beta \leq \beta_1\alpha/(4C_3 \cdot \sqrt{\log(1/\kappa)}) \leq \left(\alpha/(4C_3 \cdot \sqrt{\log(1/\kappa)})\right) \cdot \left(\frac{\alpha^3}{16 \cdot C_2 \cdot \log(1/\beta)}\right)^{C_4 \log\left(16C_2 \cdot \log(1/\beta)/\alpha^3\right)} .$$

Using Fact 15, we get that, for a sufficiently large constant C_5, it sufficient to ensure that

$$\beta \leq (\log(1/\kappa))^{-1/2} \cdot \alpha^{C_5 \log(1/\alpha)} \cdot \log(1/\beta)^{-C_5 \log\log(1/\beta)}.$$

For a sufficiently small β, $2^{-\sqrt{\log(1/\beta)}} \leq \log(1/\beta)^{-C_5 \log\log(1/\beta)}$ giving sufficient condition:

$$\beta \leq (\log(1/\kappa))^{-1/2} \cdot \alpha^{C_5 \log(1/\alpha)} \cdot 2^{-\sqrt{\log(1/\beta)}}. \tag{2.1}$$

Let

$$\beta_0 \stackrel{\text{def}}{=} (\log(1/\kappa))^{-1/2} \cdot 2^{-\sqrt{\log\log(1/\kappa)}} \cdot \alpha^{C_1 \log(1/\alpha)} ,$$

for C_1 to be chosen later. Then using concavity of the square root function we get

$$2^{-\sqrt{\log(1/\beta_0)}} \geq 2^{-\sqrt{C_1 \log^2(1/\alpha) + \log\log(1/\kappa)}} \geq 2^{-\sqrt{C_1}\log(1/\alpha)} \cdot 2^{-\sqrt{\log\log(1/\kappa)}}$$

and therefore for a sufficiently large constant C_1 it holds that

$$(\log(1/\kappa))^{-1/2} \cdot \alpha^{C_5 \log(1/\alpha)} \cdot 2^{-\sqrt{\log(1/\beta_0)}} \geq (\log(1/\kappa))^{-1/2} \cdot 2^{-\sqrt{\log\log(1/\kappa)}} \cdot \alpha^{C_5 \log(1/\alpha) + \sqrt{C_1}} \geq \beta_0.$$

Hence we obtained that condition (2.1) holds for $\beta = \beta_0$ and so also for any $\beta \leq \beta_0$. This implies the desired upper bound on $4\beta'$.

Thus, we get a new hyperplane $\mathbf{H}'_H = \{x_H \in \mathbb{R}^{|H|} \mid v_H \cdot x_H = \nu\}$ that contains all points in $S' \cup V'$. It is then clear that the n-dimensional hyperplane $\mathbf{H}' = \{x \in \mathbb{R}^n \mid v_H \cdot x_H = \nu\}$ contains all the points in $S^* = (\phi_H)^{-1}(S')$ and the points in V, and that the vector v_H defining $\mathbf{H}'$ has the claimed number of nonzero coordinates, concluding the proof of Lemma 22.

Proof of Theorem 7.

As mentioned earlier, our proof is essentially a refined version of the proof of Theorem 4 of [57]. The proof establishes the contrapositive of Theorem 7; it shows that if $\mathrm{dist}(f, g)$ is large then $d_{\mathrm{Chow}}(f, g)$ must also be large.

To aid the reader in understanding our proof, let us recall the high level structure of Goldberg's argument (which our argument follows quite closely). The first step in the argument is to show that the Chow distance $d_{\mathrm{Chow}}(f, g)$ corresponds to a Euclidean distance between two points μ_+ and μ_- in $\mathbb{R}^n$ which are the "centers of mass" of the "false positive" points V^0_+ and the "false negative" points V^0_- respectively (see Proposition 27). Hence, in order to show that the Chow distance is large, it is enough to show that μ_+ and μ_- are far apart, and to do this it is enough to lower bound

$(\mu_+ - \mu_-) \cdot \eta$ for any unit vector η. The proof attempts to do this in a sequence of stages; if any stage succeeds then we get the desired conclusion, and a dimension argument is used to show that after not too many stages, one of the stages must succeed.

In more detail, the analysis of the first stage works as follows: Fix a separating hyperplane $\mathbf{A}_0$ and consider the unit vector ℓ_0 which is normal to $\mathbf{A}_0$. If many points in $V^0 := V^0_+ \cup V^0_-$ lie far from $\mathbf{A}_0$ then it is not hard to lower bound $(\mu_+ - \mu_-) \cdot \eta$ (see Claim 30). On the other hand, if very few points in V^0 lie far from $\mathbf{A}_0$, then since $|V^0|$ is large (essentially of size at least $\epsilon 2^n$; recall that by assumption $\mathrm{dist}(f, g)$ is large) it must be the case that almost all the points in V^0 lie very close to the hyperplane $\mathbf{A}_0$. This means that we may apply the key geometric lemma, Lemma 22, to conclude that there is a hyperplane $\mathbf{A}_1$ which passes through almost all of the points in V_0.

In the next stage, essentially the same argument as above is carried out in the affine space spanned by the hyperplane $\mathbf{A}_1$. As above, it is argued that either a large set of points lies far from a separating hyperplane (in which case the Euclidean distance between μ_+ and μ_- can be lower bounded as above, see Claim 31), or else we can again apply Lemma 22 to conclude that there is a hyperplane $\mathbf{A}_2$ – which is an $(n-2)$-dimensional affine subspace of $\mathbb{R}^n$ – which passes through almost all of the points in V_0. Continuing this reasoning for $O(\log(1/\epsilon))$ stages, the argument gives that there is an $(n - O(\log(1/\epsilon)))$-dimensional affine subspace of $\mathbb{R}^n$ that contains $\Omega(\epsilon) \cdot 2^n$ points of V_0; but this contradicts a well-known upper bound on the number of points in $\{-1, 1\}^n$ that any affine subspace of $\mathbb{R}^n$ of a given dimension can contain (see Fact 16). This contradiction concludes the argument.

The arguments sketched above are those used by Goldberg in the proof of his Theorem 4, and indeed we follow the same high level steps in our proof; however there are two significant ways in which our proof differs from that of Goldberg. One of these ways is that we generalize Goldberg's arguments to allow g to be a bounded function rather than a Boolean function (this is why our detailed arguments given below use the weight function $\mathcal{W}(x)$). The second is that we carefully get rid of various factors of $\sqrt{n}$ which arise in the [57] argument (and which would be prohibitively "expensive" for us). Lemma 156 (see Appendix A.3) is useful for this purpose.

We are now ready to prove Theorem 7.

Proof of Theorem 7. Let $f : \{-1,1\}^n \to \{-1,1\}$ be an LTF and $g : \{-1,1\}^n \to [-1,1]$ be an arbitrary bounded function. Assuming that $\mathrm{dist}(f, g) = \epsilon$, we will prove that $d_{\mathrm{Chow}}(f, g) \geq \delta = \delta(\epsilon) = \epsilon^{O(\log^2(1/\epsilon))}$.

Let us define $V_+ = \{x \in \{-1,1\}^n \mid f(x) = 1, g(x) < 1\}$ and $V_- = \{x \in \{-1,1\}^n \mid f(x) = -1, g(x) > -1\}$. Also, for every point $x \in \{-1,1\}^n$, we associate a weight $\mathcal{W}(x) = |f(x) - g(x)|$ and for a set S, we define $\mathcal{W}(S) \stackrel{\text{def}}{=} \sum_{x \in S} \mathcal{W}(x)$.

It is clear that $V_+ \cup V_-$ is the disagreement region between f and g and that therefore $\mathcal{W}(V_+) + \mathcal{W}(V_-) = \epsilon \cdot 2^n$. We claim that without loss of generality we may assume that $(\epsilon - \delta) \cdot 2^{n-1} \leq \mathcal{W}(V_+), \mathcal{W}(V_-) \leq (\epsilon + \delta) \cdot 2^{n-1}$. Indeed, if this condition is not satisfied, we have that $|\widehat{f}(0) - \widehat{g}(0)| > \delta$ which gives the conclusion of the theorem.

We record the following straightforward fact which shall be used several times subsequently.

Fact 26. *For $\mathcal{W}$ as defined above, for all $X \subseteq \{-1,1\}^n$, $|X| \geq \mathcal{W}(X)/2$.*

We start by defining $V_+^0 = V_+$, $V_-^0 = V_-$ and $V^0 = V_+^0 \cup V_-^0$. The following simple proposition will be useful throughout the proof, since it characterizes the Chow distance between f and g (excluding the degree-0 coefficients) as the (normalized) Euclidean distance between two well-defined points in $\mathbb{R}^n$:

Proposition 27. *Let* $\mu_+ = \sum_{x \in V_+} \mathcal{W}(x) \cdot x$ *and* $\mu_- = \sum_{x \in V_-} \mathcal{W}(x) \cdot x$. *Then* $\sum_{i=1}^n (\widehat{f}(i) - \widehat{g}(i))^2 = 2^{-2n} \cdot \|\mu_+ - \mu_-\|^2$.

Proof. For $i \in [n]$ we have that $\widehat{f}(i) = \mathbf{E}[f(x)x_i]$ and hence $\widehat{f}(i) - \widehat{g}(i) = \mathbf{E}[(f(x) - g(x))x_i]$. Hence $2^n(\widehat{f}(i) - \widehat{g}(i)) = \sum_{x \in V_+} \mathcal{W}(x) \cdot x_i - \sum_{x \in V_-} \mathcal{W}(x) \cdot x_i = (\mu_+ - \mu_-) \cdot \mathbf{e}_i$ where $(\mu_+ - \mu_-) \cdot \mathbf{e}_i$ is the inner product of the vector $\mu_+ - \mu_-$ with the unit vector $\mathbf{e}_i$. Since $\mathbf{e}_1, \ldots, \mathbf{e}_n$ form a complete orthonormal basis for $\mathbb{R}^n$, it follows that

$$\|\mu_+ - \mu_-\|^2 = 2^{2n} \sum_{i \in [n]} (\widehat{f}(i) - \widehat{g}(i))^2$$

proving the claim. □

If $\eta \in \mathbb{R}^n$ has $\|\eta\| = 1$ then it is clear that $\|\mu_+ - \mu_-\| \geq (\mu_+ - \mu_-) \cdot \eta$. By Proposition 27, to lower bound the Chow distance $d_{\text{Chow}}(f, g)$, it suffices to establish a lower bound on $(\mu_+ - \mu_-) \cdot \eta$ for a unit vector η of our choice.

Before proceeding with the proof we fix some notation. For any line ℓ in $\mathbb{R}^n$ and point $x \in \mathbb{R}^n$, we let $\ell(x)$ denote the projection of the point x on the line ℓ. For a set $X \subseteq \mathbb{R}^n$ and a line ℓ in $\mathbb{R}^n$, $\ell(X) \overset{\text{def}}{=} \{\ell(x) : x \in X\}$. We use $\widehat{\ell}$ to denote the unit vector in the direction of ℓ (its orientation is irrelevant for us).

Definition 28. *For a function* $\mathcal{W} : \{-1, 1\}^n \to [0, \infty)$, *a set* $X \subseteq \{-1, 1\}^n$ *is said to be* (ϵ, ν)-balanced *if* $(\epsilon - \nu)2^{n-1} \leq \mathcal{W}(X) \leq (\epsilon + \nu)2^{n-1}$.

Whenever we say that a set X is (ϵ, ν)-balanced, the associated function $\mathcal{W}$ is implicitly assumed to be the one defined at the start of the proof of Theorem 7. Recall that as noted above, we may assume that the sets V_+ and V_- are balanced since otherwise the conclusion of the theorem follows easily.

The following technical proposition will be useful during the course of the proof; later we will apply it taking X_1 to be V_+^0 and X_2 to be V_-^0. Intuitively, it says that that if balanced sets X_1 and X_2 are (a) separated by a point q after projection onto a line ℓ, and (b) contain many points which (after projection onto ℓ) lie far from q, then the unit vector in the direction of ℓ "witnesses" the fact that the centers of mass of X_1 and X_2 are far from each other.

Proposition 29. *Let* $X_1, X_2 \subseteq \{-1, 1\}^n$ *be* (ϵ, ν)*-balanced sets where* $\nu \leq \epsilon/8$. *Let* ℓ *be a line in* $\mathbb{R}^n$ *and* $q \in \ell$ *be a point on* ℓ *such that the sets* $\ell(X_1)$ *and* $\ell(X_2)$ *lie on opposite sides of* q. *Suppose that* $S \overset{\text{def}}{=} \{x \mid x \in X_1 \cup X_2 \text{ and } \|\ell(x) - q\| \geq \beta\}$. *If* $\mathcal{W}(S) \geq \gamma 2^n$, *then for* $\mu_1 = \sum_{x \in X_1} \mathcal{W}(x) \cdot x$ *and* $\mu_2 = \sum_{x \in X_2} \mathcal{W}(x) \cdot x$, *we have*

$$|(\mu_1 - \mu_2) \cdot \widehat{\ell}| \geq (\beta\gamma - \nu\sqrt{2\ln(16/\epsilon)})2^n.$$

In particular, for $\nu\sqrt{2\ln(16/\epsilon)} \leq \beta\gamma/2$, we have $|(\mu_1 - \mu_2)\cdot\widehat{\ell}| \geq (\beta\gamma/2)2^n$.

Proof. We may assume that the projection $\ell(x)$ of any point $x \in X_1$ on ℓ is of the form $q + \lambda_x\widehat{\ell}$ where $\lambda_x > 0$, and that the projection $\ell(x)$ of any point $x \in X_2$ on ℓ is of the form $q - \lambda_x\widehat{\ell}$ where $\lambda_x > 0$. We can thus write

$$\begin{aligned}(\mu_1 - \mu_2)\cdot\widehat{\ell} &= \sum_{x\in X_1}\mathcal{W}(x)(q\cdot\widehat{\ell}+\lambda_x) - \sum_{x\in X_2}\mathcal{W}(x)(q\cdot\widehat{\ell}-\lambda_x)\\ &= (\mathcal{W}(X_1)-\mathcal{W}(X_2))\,q\cdot\widehat{\ell} + \sum_{x\in X_1\cup X_2}\mathcal{W}(x)\cdot\lambda_x.\end{aligned}$$

By the triangle inequality we have

$$\left|(\mu_1-\mu_2)\cdot\widehat{\ell}\right| \geq \sum_{x\in X_1\cup X_2}\mathcal{W}(x)\cdot\lambda_x - |q\cdot\widehat{\ell}|\,|(\mathcal{W}(X_1)-\mathcal{W}(X_2))|$$

so it suffices to bound each term separately. For the first term we can write

$$\sum_{x\in X_1\cup X_2}\mathcal{W}(x)\cdot\lambda_x \geq \sum_{x\in S}\mathcal{W}(x)\cdot\lambda_x \geq \beta\gamma 2^n.$$

To bound the second term, we first recall that (by assumption) $|\mathcal{W}(X_1)-\mathcal{W}(X_2)| \leq \nu 2^n$. Also, we claim that $|q\cdot\widehat{\ell}| < \sqrt{2\ln(16/\epsilon)}$. This is because otherwise the Hoeffding bound implies that the function defined by $g(x) = \text{sign}(x\cdot\widehat{\ell} - q\cdot\widehat{\ell})$ will be $\epsilon/8$ close to a constant function on $\{-1,1\}^n$. In particular, at least one of $|X_1|, |X_2|$ must be at most $(\epsilon/8)2^n$. However, by Fact 26, for $i = 1, 2$ we have that $|X_i| \geq \mathcal{W}(X_i)/2 \geq (\epsilon/4 - \nu/4)2^n > (\epsilon/8)2^n$ resulting in a contradiction. Hence it must be the case that $|q\cdot\widehat{\ell}| < \sqrt{2\ln(16/\epsilon)}$. This implies that $|(\mu_1-\mu_2)\cdot\widehat{\ell}| \geq (\beta\gamma - \nu\sqrt{2\ln(16/\epsilon)})2^n$ and the proposition is proved. $\square$

We consider a separating hyperplane $\mathbf{A}_0$ for f and assume (without loss of generality) that $\mathbf{A}_0$ does not contain any points of the unit hypercube $\{-1,1\}^n$. Let $\mathbf{A}_0 = \{x \in \mathbb{R}^n \mid w\cdot x = \theta\}$, where $\|w\| = 1$, $\theta \in \mathbb{R}$ and $f(x) = \text{sign}(w\cdot x - \theta)$.

Consider a line ℓ_0 normal to $\mathbf{A}_0$, so w is the unit vector defining the direction of ℓ_0 that points to the halfspace $f^{-1}(1)$. As stated before, the exact orientation of ℓ_0 is irrelevant to us and the choice of orientation here is arbitrary. Let $q_0 \in \mathbb{R}^n$ be the intersection point of ℓ_0 and $\mathbf{A}_0$. Then we can write the line ℓ_0 as $\ell_0 = \{p \in \mathbb{R}^n \mid p = q_0 + \lambda w, \lambda \in \mathbb{R}\}$.

Define $\beta \stackrel{\text{def}}{=} \epsilon^{C_2\cdot\log(1/\epsilon)}$ for a constant C_2 to be chosen later and consider the set of points

$$S_0 = \{x : x \in V^0 \mid \|\ell_0(x) - q_0\| \geq \beta\}.$$

The following claim states that if $\mathcal{W}(S_0)$ is not very small, we get the desired lower bound on the Chow distance. It follows from the geometric characterization of Chow distance, Proposition 27, and Proposition 29.

Claim 30. *Suppose that $\mathcal{W}(S_0) \geq \gamma_0 \cdot 2^n$ where $\gamma_0 \overset{def}{=} \beta^{4\log(1/\epsilon)-2} \cdot \epsilon$. Then $d_{\text{Chow}}(f,g) \geq \delta$, where $\delta \overset{def}{=} \beta^{4\log(1/\epsilon)}$.*

Proof. To prove the desired lower bound, we will apply Proposition 27. Consider projecting every point in V^0 on the line ℓ_0. Observe that the projections of V^0_+ are separated from the projections of V^0_- by the point q_0. Also, we recall that the sets V^0_+ and V^0_- are (ϵ, δ) balanced. Thus, for $\mu_+ = \sum_{x \in V^0_+} \mathcal{W}(x) \cdot x$ and $\mu_- = \sum_{x \in V^0_-} \mathcal{W}(x) \cdot x$, we can apply Proposition 29 to get that $|(\mu_+ - \mu_-) \cdot w| \geq (\beta\gamma_0 - \delta\sqrt{2\ln(16/\epsilon)})2^n \geq \delta 2^n$. This implies that $\|\mu_+ - \mu_-\|^2 \geq \delta^2 2^{2n}$ and using Proposition 27, this proves that $d_{\text{Chow}}(f,g) \geq \delta$. □

If the condition of Claim 30 is not satisfied, then we have that $\mathcal{W}(V^0 \setminus S_0) \geq (\epsilon - \gamma_0)2^n$. By Fact 26, we have $|V^0 \setminus S_0| \geq (\epsilon - \gamma_0)2^{n-1}$. We now apply Lemma 22 to obtain another hyperplane $\mathbf{A}_1$ which passes through all but $\kappa_1 \cdot 2^n$ points ($\kappa_1 \overset{\text{def}}{=} \gamma_0/2$) in $V^0 \setminus S_0$. We note that, for a sufficiently large constant C_2, the condition of the lemma is satisfied, as $\log(1/\kappa_1) = \text{poly}(\log(1/\epsilon))$ and $|V^0 \setminus S_0| > (\epsilon/4) \cdot 2^n$.

From this point onwards, our proof uses a sequence of $\lfloor \log(1/\epsilon) \rfloor$ cases, each of which follows along essentially the same lines as the "zeroth" case analyzed above. To this end, we define $\gamma_j = \beta^{4\log(1/\epsilon)-2(j+1)} \cdot \epsilon$. At the beginning of case j, we will have an affine space A_j of dimension $n-j$ such that $\mathcal{W}(V^0 \cap A_j) \geq (\epsilon - 2(\sum_{\ell=0}^{j-1} \gamma_\ell))2^n$. We note that this is indeed satisfied at the beginning of case 1. To see this, recall that $\mathcal{W}(V^0 \setminus S_0) > (\epsilon - \gamma_0)2^n$. Also, we have that

$$\begin{aligned} \mathcal{W}((V^0 \setminus S_0) \setminus (V^0 \cap \mathbf{A}_1)) &\leq 2|(V^0 \setminus S_0) \setminus (V^0 \cap \mathbf{A}_1)| \\ &\leq 2\kappa_1 2^n = \gamma_0 2^n. \end{aligned}$$

These together imply that $\mathcal{W}(V^0 \cap \mathbf{A}_1) \geq (\epsilon - 2\gamma_0)2^n$ confirming the hypothesis for $j = 1$.

We next define $V^j = V^0 \cap A_j$, $V^j_+ = V^j \cap V_+$ and $V^j_- = V^j \cap V_-$. Let $A'_{j+1} = A_j \cap \mathbf{A}_0$. Note that $A_j \nsubseteq \mathbf{A}_0$. This is because A_j contains points from $\{-1,1\}^n$ as opposed to $\mathbf{A}_0$ which does not. Also, A_j is not contained in a hyperplane parallel to $\mathbf{A}_0$ because A_j contains points of the unit hypercube lying on either side of $\mathbf{A}_0$. Hence it must be the case that $\dim(A'_{j+1}) = n - (j+1)$. Let ℓ_j be a line orthogonal to A'_{j+1} which is parallel to A_j. Again, we observe that the direction of ℓ_j is unique.

Our aim is essentially to establish that the conditions of Proposition 29 hold so that we may apply it to the line ℓ_j and thus obtain an analogue of Claim 30 (recall that Proposition 29 played a key role in the proof of Claim 30). Towards that end, we observe that all points in A'_{j+1} project to the same point in ℓ_j, which we call q_j. Let us define $\Lambda^j_+ = \ell_j(V^j_+)$ and $\Lambda^j_- = \ell_j(V^j_-)$. We observe that the sets Λ^j_+ and Λ^j_- are separated by q_j. Next, we define S_j as :

$$S_j = \{x \in V^j \mid \|\ell_j(x) - q_j\|_2 \geq \beta\}.$$

The next claim is analogous to Claim 30. It says that if $\mathcal{W}(S_j)$ is not too small, then we get the desired lower bound on the Chow distance. The underlying ideas are the same as Claim 30 but the proof is slightly more technical; we postpone it to Appendix A.3.

Claim 31. *For $j \le \log(8/\epsilon)$, suppose that $\mathcal{W}(S_j) \ge \gamma_j \cdot 2^n$ where γ_j is as defined above. Then $d_{\text{Chow}}(f, g) \ge \delta$.*

If the hypothesis of Claim 31 fails, then we construct an affine space A_{j+1} of dimension $n-j-1$ such that $\mathcal{W}(V^0 \cap A_{j+1}) \ge (\epsilon - 2\sum_{\ell=0}^{j} \gamma_\ell)2^n$ as described next. We recall that $U = \cup_{i=1}^n \mathbf{e}_i \cup \mathbf{0}$. It is obvious there is some subset $Y_j \subseteq U$ such that $|Y_j| = j$ and $\text{span}(A_j \cup Y_j) = \mathbb{R}^n$. Now, let us define $\mathbf{H}'_j \overset{\text{def}}{=} \text{span}(Y_j \cup A'_{j+1})$. Clearly, $\mathbf{H}'_j$ is a hyperplane and every point $x \in (V^0 \cap A_j) \setminus S_j$ is at a distance at most β from $\mathbf{H}'_j$. This is because every $x \in (V^0 \cap A_j) \setminus S_j$ is at a distance at most β from A'_{j+1} and $A'_{j+1} \subset \mathbf{H}'_j$. Also, note that all $x \in Y_j$ lie on $\mathbf{H}'_j$.

Note that $\mathcal{W}((V^0 \cap A_j) \setminus S_j) \ge (\epsilon - 2\sum_{\ell=0}^{j-1} \gamma_\ell - \gamma_j)2^n$. As prior calculation has shown, for $j \le \log(8/\epsilon)$ we have $\mathcal{W}((V^0 \cap A_j) \setminus S_j) \ge (\epsilon - 2\sum_{\ell=0}^{j-1} \gamma_\ell - \gamma_j)2^n \ge (\epsilon/2)2^n$. Using Fact 26, we get that $|(V^0 \cap A_j) \setminus S_j| \ge (\epsilon/4)2^n$. Thus, putting $\kappa_j = \gamma_j/2$ and applying Lemma 22, we get a new hyperplane $\mathbf{H}_j$ such that $|((V^0 \cap A_j) \setminus S_j) \setminus (\mathbf{H}_j \cap V^0)| \le (\gamma_j/2) \cdot 2^n$. Using that the range of $\mathcal{W}$ is bounded by 2, we get $\mathcal{W}(((V^0 \cap A_j) \setminus S_j) \setminus (\mathbf{H}_j \cap V^0)) \le \gamma_j \cdot 2^n$. Thus, we get that $\mathcal{W}(\mathbf{H}_j \cap V^0 \cap A_j) \ge (\epsilon - 2\sum_{\ell=0}^{j} \gamma_\ell)2^n$. Also, $Y_j \subset \mathbf{H}_j$.

Let us now define $A_{j+1} = A_j \cap \mathbf{H}_j$. It is clear that $\mathcal{W}(A_{j+1} \cap V^0) \ge (\epsilon - 2\sum_{\ell=0}^{j} \gamma_\ell)2^n$. Also, $\dim(A_{j+1}) < \dim(A_j)$. To see this, assume for contradiction that $\dim(A_j) = \dim(A_{j+1})$. This means that $A_j \subseteq \mathbf{H}_j$. Also, $Y_j \subset \mathbf{H}_j$. This means that $\text{span}(A_j \cup Y_j) \subset \mathbf{H}_j$. But $\text{span}(A_j \cup Y_j) = \mathbb{R}^n$ which cannot be contained in $\mathbf{H}_j$. Thus we have that $\dim(A_{j+1}) = \dim(A_j) - 1$.

Now we observe that taking $j = \lfloor \log(8/\epsilon) \rfloor$, we have a subspace A_j of dimension $n - j$ which has $\mathcal{W}(A_j \cap V^0) \ge (\epsilon - 2\sum_{\ell=0}^{j-1} \gamma_\ell)2^n > (\epsilon/2)2^n$. By Fact 26, we have that $|A_j \cap V^0| \ge (\epsilon/4)2^n$. However, by Fact 16, a subspace of dimension $n - j$ can contain at most 2^{n-j} points of $\{-1, 1\}^n$. Since $j = \lfloor \log(8/\epsilon) \rfloor$, this leads to a contradiction. That implies that the number of cases must be strictly less than $\lfloor \log(8/\epsilon) \rfloor$. In particular, for some $j < \lfloor \log(8/\epsilon) \rfloor$, it must be the case that $|S_j| \ge \gamma_j 2^n$. For this j, by Claim 31, we get a lower bound of δ on $d_{\text{Chow}}(f, g)$. This concludes the proof of Theorem 7. □

2.5 The Algorithm and its Analysis

Algorithm and Proof Overview.

In this section we give a proof overview of Theorem 10, restated below for convenience. We give the formal details of the proof in the following subsection.

Theorem 10 (Main Algorithmic Result). *There exists a randomized algorithm* `ChowReconstruct` *that for every Boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$, given $\epsilon > 0, \delta > 0$ and a vector $\vec{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_n)$ such that $\|\vec{\chi}_f - \vec{\alpha}\| \le \epsilon$, with probability at least $1 - \delta$, outputs an LBF g such that $\|\vec{\chi}_f - \vec{\chi}_g\| \le 6\epsilon$. The algorithm runs in time $\tilde{O}(n^2\epsilon^{-4}) \cdot \log(1/\delta)$. Further, g is represented by a weight vector $\kappa v \in \mathbb{R}^{n+1}$, where $\kappa \in \mathbb{R}$ and v is an integer vector with $\|v\| = O(\sqrt{n}/\epsilon^3)$.*

We now provide an intuitive overview of the algorithm and its analysis. Our algorithm is motivated by the following intuitive reasoning: since the function $\alpha_0 + \sum_{i \in [n]} \alpha_i \cdot x_i$ has the desired Chow parameters, why not just use it to define an LBF g_1 as $P_1(\alpha_0 + \sum_{i\in[n]} \alpha_i \cdot x_i)$? The answer, of course, is that as a result of applying the projection operator, the Chow parameters of g_1 can become quite different from the desired vector $\vec{\alpha}$. Nevertheless, it seems quite plausible to expect that g_1 will be better than a random guess.

Given the Chow parameters of g_1 we can try to correct them by adding the difference between $\vec{\alpha}$ and $\vec{\chi}_{g_1}$ to the vector that represents g_1. Again, intuitively we are adding a real-valued function $h_1 = \alpha_0 - \widehat{g}_1(0) + \sum_{i\in[n]}(\alpha_i - \widehat{g}_1(i)) \cdot x_i$ that has the Chow parameters that we would like to add to the Chow parameters of g_1. And, again, the projection operation is likely to ruin our intention, but we could still hope that we got closer to the vector $\vec{\alpha}$, and that by repeating this operation we will converge to an LBF with Chow parameters close to $\vec{\alpha}$.

While this might appear too naive, this is almost exactly what we do in `ChowReconstruct`. The main difference between this naive proposal and our actual algorithm is that at step t we actually add only half the difference between $\vec{\alpha}$ and the Chow vector of the current hypothesis $\vec{\chi}_{g_t}$. This is necessary in our proof to offset the fact that $\vec{\alpha}$ is only an approximation to $\vec{\chi}_f$ and the fact that we can only approximate the Chow parameters of g_t. An additional minor modification is required to ensure that the final weight vector is a multiple of an integer weight vector of length $O(\sqrt{n}/\epsilon^3)$.

The proof of correctness of this algorithm proceeds roughly as follows. If the difference vector is sufficiently large (namely, more than a small multiple of the difference between $\vec{\chi}_f$ and $\vec{\alpha}$) then the linear function h_t defined by this vector can be easily shown to be correlated with $f - g_t$, namely $\mathbf{E}[(f - g_t)h_t] \geq c\|\vec{\chi}_{g_t} - \vec{\alpha}\|^2$ for a constant $c > 0$. As was shown in [139] and [49] this condition for a Boolean h_t can be used to decrease a simple potential function measuring $\mathbf{E}[(f - g_t)^2]$, the l_2^2 distance of the current hypothesis to f. One issue that arises is this: while the l_2^2 distance is only reduced if h_t is added to g_t, in order to ensure that g_{t+1} is an LBF, we need to add the vector of difference (used to define h_t) to the weight vector representing g_t. To overcome this problem the proof in [139] uses an additional point-wise counting argument from [71]. This counting argument can be adapted to the real valued h_t, but the resulting argument becomes quite cumbersome. Instead, we augment the potential function in a way that captures the additional counting argument from [71] and easily generalizes to the real-valued case.

Proof of Theorem 10.

We begin by describing the `ChowReconstruct` algorithm. The algorithm builds g through the following iterative process. Let $g'_0 \equiv 0$ and let $g_0 = P_1(g'_0)$. Given g_t, the algorithm approximates each Chow parameter of g_t to accuracy $\epsilon/(4\sqrt{n+1})$; let $(\beta_0, \beta_1, \ldots, \beta_n)$ denote the results. For each $0 \leq i \leq n$, define $\tilde{g}_t(i)$ to be the closest value to β_i that ensures that $\alpha_i - \tilde{g}_t(i)$ is an integer multiple of $\epsilon/(2\sqrt{n+1})$. Let $\tilde{\chi}_{g_t} = (\tilde{g}_t(0), \ldots, \tilde{g}_t(n))$ denote the resulting vector of coefficients. Note that

$$\|\tilde{\chi}_{g_t} - \vec{\chi}_{g_t}\| \leq \sqrt{\sum_{i=0}^{n}(\epsilon/(2\sqrt{n+1}))^2} = \epsilon/2.$$

Define $\rho = \|\vec{\alpha} - \tilde{\chi}_{g_t}\|$. If $\rho \leq 4\epsilon$ then the algorithm stop s and output s g_t. By the triangle inequality,

$$\begin{aligned} \|\vec{\chi}_f - \vec{\chi}_{g_t}\| &\leq \|\vec{\chi}_f - \vec{\alpha}\| + \|\vec{\alpha} - \tilde{\chi}_{g_t}\| + \|\tilde{\chi}_{g_t} - \vec{\chi}_{g_t}\| \\ &\leq \epsilon(1 + 4 + 1/2) < 6\epsilon, \end{aligned}$$

so g_t satisfies the claimed condition.

Otherwise (if $\rho > 4\epsilon$), let $g'_{t+1} = g'_t + h_t/2$ and $g_{t+1} = P_1(g'_{t+1})$ where h_t is defined by

$$h_t \triangleq \sum_{i=0}^{n} (\alpha_i - \tilde{g}_t(i))x_i.$$

Note that this is equivalent to adding the vector $(\vec{\alpha} - \tilde{\chi}_{g_t})/2$ to the degree 0 and 1 Fourier coefficients of g'_t (which are also the components of the vector representing g_t). This concludes the description of the `ChowReconstruct` algorithm.

To prove the convergence of this process we define a potential function at step t as

$$\begin{aligned} E(t) &= \mathbf{E}[(f - g_t)^2] + 2\,\mathbf{E}[(f - g_t)(g_t - g'_t)] \\ &= \mathbf{E}[(f - g_t)(f - 2g'_t + g_t)]. \end{aligned}$$

The key claim in the proof of Theorem 10 is the following:

Claim 32. *We have $E(t + 1) - E(t) \leq -2\epsilon^2$.*

Proof. To prove Claim 32 we first prove that

$$\mathbf{E}[(f - g_t)h_t] \geq \rho\left(\rho - \frac{3}{2}\epsilon\right). \tag{2.2}$$

To see this, observe that by the Cauchy-Schwarz inequality, we have

$$\begin{aligned} \mathbf{E}[(f - g_t)h_t] &= \sum_{i=0}^{n} (\widehat{f}(i) - \widehat{g_t}(i))(\alpha_i - \tilde{g}_t(i)) \\ &= \sum_{i=0}^{n} \Big[(\widehat{f}(i) - \alpha_i)(\alpha_i - \tilde{g}_t(i)) + \\ &\qquad (\tilde{g}_t(i) - \widehat{g_t}(i))(\alpha_i - \tilde{g}_t(i)) + (\alpha_i - \tilde{g}_t(i))^2\Big] \\ &\geq -\rho\epsilon - \rho\epsilon/2 + \rho^2 \geq \rho^2 - \frac{3}{2}\rho\epsilon. \end{aligned}$$

In addition, by Parseval's identity,

$$\mathbf{E}[h_t^2] = \sum_{i=0}^{n} (\alpha_i - \tilde{g}_t(i))^2 = \rho^2\ . \tag{2.3}$$

Now,

$$\begin{aligned} E(t+1) - E(t) &= \mathbf{E}[(f-g_{t+1})(f-2g'_{t+1}+g_{t+1})] - \mathbf{E}[(f-g_t)(f-2g'_t+g_t)] \\ &= \mathbf{E}\left[(f-g_t)(2g'_t-2g'_{t+1}) + (g_{t+1}-g_t)(2g'_{t+1}-g_t-g_{t+1})\right] \\ &= -\mathbf{E}[(f-g_t)h_t] + \mathbf{E}\left[(g_{t+1}-g_t)(2g'_{t+1}-g_t-g_{t+1})\right]. \end{aligned} \tag{2.4}$$

To upper-bound the expression $\mathbf{E}\left[(g_{t+1}-g_t)(2g'_{t+1}-g_t-g_{t+1})\right]$ we prove that for every point $x \in \{-1,1\}^n$,

$$(g_{t+1}(x)-g_t(x))(2g'_{t+1}(x)-g_t(x)-g_{t+1}(x)) \le h_t(x)^2/2. \tag{2.5}$$

We first observe that

$$|g_{t+1}(x)-g_t(x)| = |P_1(g'_t(x)+h_t(x)/2) - P_1(g'_t(x))| \le |h_t(x)/2|,$$

where the equality is by definition of g_{t+1} and g_t and the inequality holds because a projection operation does not increase the distance. Now the triangle inequality gives

$$|2g'_{t+1}(x)-g_t(x)-g_{t+1}(x)| \le |g'_{t+1}(x)-g_t(x)| + |g'_{t+1}(x)-g_{t+1}(x)|.$$

We shall argue that either each of the two summands above on the right-hand size is at most $|h_t(x)/2|$, or else the left-hand side of (2.5) is zero.

For the first summand, we have that $|g'_{t+1}(x)-g_t(x)| = |h_t(x)/2+g'_t(x)-g_t(x)|$. This can be larger than $|h_t(x)/2|$ in only two ways: the first of these is that $g'_t(x)-g_t(x) \neq 0$ and $g'_t(x)-g_t(x)$ has the same sign as $h_t(x)$. By the definition of P_1, this implies that $g_t(x) = \text{sign}(g'_t(x))$ and $\text{sign}(h_t(x)) = \text{sign}(g'_t(x)-g_t(x)) = g_t(x)$. However, in this case $|g'_{t+1}(x)| \ge |g'_t(x)| > 1$ and $\text{sign}(g'_{t+1}(x)) = \text{sign}(g'_t(x)) = g_t(x)$. As a result $g_{t+1}(x) = g_t(x)$ and $(g_{t+1}(x)-g_t(x))(2g'_{t+1}(x)-g_t(x)-g_{t+1}(x)) = 0$.

The second way in which it is possible to have $|h_t(x)/2+g'_t(x)-g_t(x)| > |h_t(x)/2|$ is if $g'_t(x)-g_t(x) \neq 0$, $g'_t(x)-g_t(x)$ has the opposite sign from $h_t(x)/2$, and $|g'_t(x)-g_t(x)| > 2|h_t(x)/2|$. In this case we have that $|g'_{t+1}(x)| > 1$ and $g_{t+1}(x) = \text{sign}(g'_{t+1}(x)) = \text{sign}(g'_t(x)) = g_t(x)$, so $(g_{t+1}(x)-g_t(x))(2g'_{t+1}(x)-g_t(x)-g_{t+1}(x)) = 0$ as above.

Similarly, for the second summand, $|g'_{t+1}(x)-g_{t+1}(x)| > |h_t(x)/2|$ implies that $g_{t+1}(x) = \text{sign}(g'_{t+1}(x))$ and $|g'_{t+1}(x)| \ge |h_t(x)/2|+1$. This implies that $|g'_t(x)| \ge |g'_{t+1}(x)| - |h_t(x)/2| > 1$ and $g_t(x) = \text{sign}(g'_t(x)) = \text{sign}(g'_{t+1}(x)) = g_{t+1}(x)$, which means $(g_{t+1}(x)-g_t(x))(2g'_{t+1}(x)-g_t(x)-g_{t+1}(x)) = 0$.

Altogether we obtain that

$$\begin{aligned} (g_{t+1}(x)-g_t(x))(2g'_{t+1}(x)-g_t(x)-g_{t+1}(x)) &\le \max\{0, |h_t(x)/2|(|h_t(x)/2|+|h_t(x)/2|)\} \\ &= h_t(x)^2/2, \end{aligned}$$

establishing (2.5) as desired. This pointwise inequality implies that

$$\mathbf{E}\left[(g_{t+1}-g_t)(2g'_{t+1}-g_t-g_{t+1})\right] \le \mathbf{E}[h_t^2]/2 = \rho^2/2, \tag{2.6}$$

where we used (2.3) for the equality. By substituting equations (2.2) and (2.6) into equation (2.4), we obtain the claimed decrease in the potential function,

$$E(t+1) - E(t) \le -\rho^2 + \frac{3}{2}\rho\epsilon + \rho^2/2 = -(\rho - 3\epsilon)\rho/2 \le -2\epsilon^2,$$

and Claim 32 is proved. □

We now observe that

$$E(t) = \mathbf{E}[(f-g_t)^2] + 2\,\mathbf{E}[(f-g_t)(g_t - g'_t)] \ge 0 \tag{2.7}$$

for all t. This follows from noting that for every x and $f(x) \in \{-1,1\}$, if $g_t(x) - g'_t(x)$ is non-zero then, by the definition of P_1, $g_t(x) = \mathrm{sign}(g'_t(x))$ and $\mathrm{sign}(g_t(x) - g'_t(x)) = -g_t(x)$. In this case, either $f(x) - g_t(x) = 0$ or else $\mathrm{sign}(f(x) - g_t(x)) = -g_t(x)$ and hence $(f(x) - g_t(x))(g_t(x) - g'_t(x)) \ge 0$. Therefore

$$\mathbf{E}[(f-g_t)(g_t - g'_t)] \ge 0$$

(and, naturally, $\mathbf{E}[(f-g_t)^2] \ge 0$).

It is easy to see that $E(0) = 1$ and consequently (2.7) and Claim 32) imply that the process will stop after at most $1/(2\epsilon^2)$ steps.

We now establish the claimed weight bound on the LBF output by the algorithm and the bound on the running time. Let T denote the number of iterations of the algorithm. By our construction, the function $g_T = P_1(\sum_{t<T} h_t/2)$ is an LBF represented by weight vector $\vec{w}$ such that $w_i = \sum_{j<T}(\alpha_i - \tilde{g}_j(i))/2$. Our rounding of the estimates of Chow parameters of g_t ensures that each $(\alpha_i - \tilde{g}_j(i))/2$ is an integer multiple of $\kappa = \epsilon/(2\sqrt{n+1})$. Hence g_T can be represented by a vector $\vec{w} = \kappa\vec{v}$, where vector $\vec{v}$ has only integer components. At every step j,

$$\sqrt{\sum_{i=0}^{n}(\alpha_i - \tilde{g}_j(i))^2} \le 2 + \epsilon + \epsilon/2 = O(1).$$

Therefore, by the triangle inequality, $\|\vec{w}\| = O(\epsilon^{-2})$ and hence $\|\vec{v}\| = \|\vec{w}\|/\kappa = O(\sqrt{n}/\epsilon^3)$.

The running time of the algorithm is essentially determined by finding $\tilde{\chi}_{g_t}$ in each step t. Finding $\tilde{\chi}_{g_t}$ requires estimating each $\widehat{g_t}(i) = \mathbf{E}[g_t(x) \cdot x_i]$ to accuracy $\epsilon/(4\sqrt{n+1})$. Chernoff bounds imply that, by using the empirical mean of $g_t(x) \cdot x_i$ on $O((n/\epsilon^2) \cdot \log(n/(\epsilon\delta)))$ random points as our estimate of $\widehat{g_t}(i)$, we can ensure that, with probability at least $1-\delta$, the estimates are within $\epsilon/(4\sqrt{n+1})$ of the true values for all $n+1$ Chow parameters of g_t for every $t \le T = O(\epsilon^{-2})$.

Evaluating g_t on any point $x \in \{-1,1\}^n$ takes $O(n \cdot \log(n/\epsilon))$ time and we need to evaluate it on $O((n/\epsilon^2) \cdot \log(n/(\epsilon\delta)))$ points in each of $O(\epsilon^{-2})$ steps. This gives us the claimed total running time bound, and the proof of Theorem 10 is complete.

2.6 The Main Results

Proofs of Theorems 1 and 2.

In this subsection we put the pieces together and prove our main results. We start by giving a formal statement of Theorem 1:

Theorem 33 (Main). *There is a function $\kappa(\epsilon) \overset{def}{=} 2^{-O(\log^3(1/\epsilon))}$ such that the following holds: Let $f : \{-1,1\}^n \to \{-1,1\}$ be an LTF and let $0 < \epsilon, \delta < 1/2$. Write $\vec{\chi}_f$ for the Chow vector of f and assume that $\vec{\alpha} \in \mathbb{R}^{n+1}$ is a vector satisfying $\|\vec{\alpha} - \vec{\chi}_f\| \le \kappa(\epsilon)$. Then, there is an algorithm $\mathcal{A}$ with the following property: Given as input $\vec{\alpha}$, ϵ and δ, algorithm $\mathcal{A}$ performs $\tilde{O}(n^2 \cdot \mathrm{poly}(1/\kappa(\epsilon))) \cdot \log(1/\delta)$ bit operations and outputs the (weights-based) representation of an LTF f^* which with probability at least $1 - \delta$ satisfies $\mathrm{dist}(f, f^*) \le \epsilon$.*

Proof of Theorem 33. Suppose that we are given a vector $\vec{\alpha} \in \mathbb{R}^{n+1}$ that satisfies $\|\vec{\alpha} - \vec{\chi}_f\| \le \kappa(\epsilon)$, where f is the unknown LTF to be learned. To construct the desired f^*, we run algorithm `ChowReconstruct` (from Theorem 10) on input $\vec{\alpha}$ with its "ϵ" parameter set to $\kappa(\epsilon)$. The algorithm runs in time $\tilde{O}(n^2 \cdot \mathrm{poly}(1/\kappa(\epsilon))) \cdot \log(1/\delta)$ and outputs an LBF g such that with probability at least $1-\delta$ we have $d_{\mathrm{Chow}}(f,g) \le 6\kappa(\epsilon)$. Applying Theorem 7 we get that with probability at least $1 - \delta$ we have $\mathrm{dist}(f,g) \le \epsilon/2$. (We can set the constants appropriately in the definition of the function $\kappa(\epsilon)$ above, so that the conclusion of applying Theorem 7 is "$\mathrm{dist}(f,g) \le \epsilon/2$".) Writing the LBF g as $g(x) = P_1(v_0 + \sum_{i=1}^n v_i x_i)$, we now claim that $f^*(x) = \mathrm{sign}(v_0 + \sum_{i=1}^n v_i x_i)$ has $\mathrm{dist}(f, f^*) \le \epsilon$. This is simply because for each input $x \in \{-1,1\}^n$, the contribution that x makes to to $\mathrm{dist}(f, f^*)$ is at most twice the contribution x makes to $\mathrm{dist}(f,g)$. This completes the proof of Theorem 33. □

As a simple corollary, we obtain Theorem 2.

Proof of Theorem 2. Let $f : \{-1,1\}^n \to \{-1,1\}$ be an arbitrary LTF. We apply Theorem 33 above, for $\delta = 1/3$, and consider the LTF f^* produced by the above proof. Note that the weights v_i defining f^* are identical to the weights of the LBF g output by the algorithm `ChowReconstruct`. It follows from Theorem 10 that these weights are integers that satisfy $\sum_{i=1}^n v_i^2 = O(n \cdot \kappa(\epsilon)^{-6})$, and the proof is complete. □

As pointed out in Section 2 our algorithm runs in $\mathrm{poly}(n/\epsilon)$ time for LTFs whose integer weight is at most $\mathrm{poly}(n)$. Formally, we have:

Theorem 34. *Let $f = \mathrm{sign}(\sum_{i=1}^n w_i x_i - \theta)$ be an LTF with integer weights w_i such that $W \overset{def}{=} \sum_{i=1}^n |w_i|$. Fix $0 < \epsilon, \delta < 1/2$. Write $\vec{\chi}_f$ for the Chow vector of f and assume that $\vec{\alpha} \in \mathbb{R}^{n+1}$ is a vector satisfying $\|\vec{\alpha} - \vec{\chi}_f\| \le \epsilon/(12W)$. Then, there is an algorithm $\mathcal{A}'$ with the following property: Given as input $\vec{\alpha}$, W', ϵ and δ, where $W' \ge W$ and $W' = \mathrm{poly}(W)$, algorithm $\mathcal{A}'$ performs $\mathrm{poly}(nW/\epsilon) \cdot \log(1/\delta)$ bit operations and outputs the (weights-based) representation of an LTF f^* which with probability at least $1 - \delta$ satisfies $\mathrm{dist}(f, f^*) \le \epsilon$.*

Before we proceed with the proof, we remark that the above theorem implies an algorithm for the exact problem with running time $2^{O(n \log n)}$. This follows by applying the theorem for $\epsilon = 2^{-n-1}$ recalling that any LTF has an exact integer-weight representation with $W = 2^{O(n \log n)}$.

Proof. As stated before, both the algorithm and proof of the above theorem are essentially identical to the ones in Theorem 33. The details follow.

Given a vector $\vec{\alpha} \in \mathbb{R}^{n+1}$ satisfying $\|\vec{\alpha} - \vec{\chi}_f\| \leq \epsilon/(12W)$, where f is the unknown LTF, we run algorithm `ChowReconstruct` on input $\vec{\alpha}$ with its "ϵ" parameter set to $\epsilon/(12W')$. The algorithm runs in time $\mathrm{poly}(nW'/\epsilon) \cdot \log(1/\delta)$, which is $\mathrm{poly}(nW/\epsilon) \cdot \log(1/\delta)$ by our assumption on W, and outputs an LBF g such that with probability at least $1 - \delta$, $d_{\mathrm{Chow}}(f, g) \leq 6\epsilon/(12W') \leq \epsilon/(2W)$. At this point, we need to apply the following simple structural result of [14]:

Fact 35. *Let $f = \mathrm{sign}(\sum_{i=1}^n w_i x_i - \theta)$ be an LTF with integer weights w_i, where $W \overset{def}{=} \sum_{i=1}^n |w_i|$, and let $g : \{-1,1\}^n \to [-1,1]$ be an arbitrary bounded function. Fix $0 < \epsilon < 1/2$. If $d_{\mathrm{Chow}}(f, g) \leq \epsilon/W$, then $\mathrm{dist}(f, g) \leq \epsilon$.*

The above fact implies that, with probability at least $1 - \delta$, the LBF g output by the algorithm satisfies $\mathrm{dist}(f, g) \leq \epsilon/2$. If $g(x) = P_1(v_0 + \sum_{i=1}^n v_i x_i)$, then as in the proof of Theorem 33 we have that the LTF $f^*(x) = \mathrm{sign}(v_0 + \sum_{i=1}^n v_i x_i)$ has $\mathrm{dist}(f, f^*) \leq \epsilon$. This completes the proof. $\square$

Near-optimality of Theorem 7.

Theorem 7 says that if f is an LTF and $g : \{-1,1\}^n \to [-1,1]$ satisfy $d_{\mathrm{Chow}}(f, g) \leq \epsilon$ then $\mathrm{dist}(f, g) \leq 2^{-\Omega(\sqrt[3]{\log(1/\epsilon)})}$. It is natural to wonder whether the conclusion can be strengthened to "$\mathrm{dist}(f, g) \leq \epsilon^c$" where $c > 0$ is some absolute constant. Here we observe that no conclusion of the form "$\mathrm{dist}(f, g) \leq 2^{-\gamma(1/\epsilon)}$" is possible for any function $\gamma(1/\epsilon) = \omega(\log(1/\epsilon)/\log\log(1/\epsilon))$.

To see this, fix γ to be any function such that

$$\gamma(1/\epsilon) = \omega(\log(1/\epsilon)/\log\log(1/\epsilon)).$$

If there were a stronger version of Theorem 7 in which the conclusion is "then $\mathrm{dist}(f, g) \leq 2^{-\gamma(1/\epsilon)}$," the arguments of Section 2.6 would give that for any LTF f, there is an LTF $f' = \mathrm{sign}(v \cdot x - \nu)$ such that $\mathbf{P}r[f(x) \neq f'(x)] \leq \epsilon$, where each $v_i \in \mathbb{Z}$ satisfies $|v_i| \leq \mathrm{poly}(n) \cdot (1/\epsilon)^{o(\log\log(1/\epsilon))}$. Taking $\epsilon = 1/2^{n+1}$, this tells us that f' must agree with f on every point in $\{-1,1\}^n$, and each integer weight in the representation $\mathrm{sign}(v \cdot x - \nu)$ is at most $2^{o(n \log n)}$. But choosing f to be Håstad's LTF from [64], this is a contradiction, since any integer representation of that LTF must have every $|v_i| \geq 2^{\Omega(n \log n)}$.

2.7 Applications to learning theory

In this section we show that our approach yields a range of interesting algorithmic applications in learning theory.

Learning threshold functions in the 1-RFA model.

Ben-David and Dichterman [11] introduced the "Restricted Focus of Attention" (RFA) learning framework to model the phenomenon (common in the real world) of a learner having incomplete access to examples. We focus here on the uniform-distribution "1-RFA" model. In this setting each time the learner is to receive a labeled example, it first specifies an index $i \in [n]$; then an n-bit string x is drawn from the uniform distribution over $\{-1, 1\}^n$ and the learner is given $(x_i, f(x))$. So for each labeled example, the learner is only shown the i-th bit of the example along with the label.

Birkendorf et al. [14] asked whether LTFs can be learned in the uniform distribution 1-RFA model, and showed that a sample of $O(n \cdot W^2 \cdot \log(\frac{n}{\delta})/\epsilon^2)$ many examples is information-theoretically sufficient for learning an unknown threshold function with integer weights w_i that satisfy $\sum_i |w_i| \leq W$. The results of Goldberg [57] and Servedio [129] show that samples of size $(n/\epsilon)^{O(\log(n/\epsilon)\log(1/\epsilon))}$ and $\mathrm{poly}(n) \cdot 2^{\tilde{O}(1/\epsilon^2)}$ respectively are information-theoretically sufficient for learning an arbitrary LTF to accuracy ϵ, but none of these earlier results gave a computationally efficient algorithm. [120] gave the first algorithm for this problem; as a consequence of their result for the Chow Parameters Problem, they gave an algorithm which learns LTFs to accuracy ϵ and confidence $1 - \delta$ in the uniform distribution 1-RFA model, running in $2^{2^{\tilde{O}(1/\epsilon^2)}} \cdot n^2 \cdot \log n \cdot \log(\frac{n}{\delta})$ bit operations. As a direct consequence of Theorem 1, we obtain a much more time efficient learning algorithm for this learning task.

Theorem 36. *There is an algorithm which performs $\tilde{O}(n^2) \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))} \cdot \log(\frac{1}{\delta})$ bit-operations and properly learns LTFs to accuracy ϵ and confidence $1 - \delta$ in the uniform distribution 1-RFA model.*

Agnostic-type learning.

In this section we show that a variant of our main algorithm gives a very fast "agnostic-type" algorithm for learning LTFs under the uniform distribution.

Let us briefly review the uniform distribution agnostic learning model [86] in our context. Let $f : \{-1, 1\}^n \to \{-1, 1\}$ be an arbitrary boolean function. We write $\mathsf{opt} = \mathrm{dist}(f, \mathcal{H}) \stackrel{\text{def}}{=} \min_{h \in \mathcal{H}} \mathbf{Pr}_x[h(x) \neq f(x)]$, where $\mathcal{H}$ denotes the class of LTFs. A uniform distribution agnostic learning algorithm is given uniform random examples labeled according to an arbitrary f and outputs a hypothesis h satisfying $\mathrm{dist}(h, f) \leq \mathsf{opt} + \epsilon$.

The only efficient algorithm for learning LTFs in this model [77] is non-proper and runs in time $n^{\mathrm{poly}(1/\epsilon)}$. This motivates the design of more efficient algorithms with potentially relaxed guarantees. [120] give an "agnostic-type" algorithm, that guarantees $\mathrm{dist}(h, f) \leq \mathsf{opt}^{\Omega(1)} + \epsilon$ and runs in time $\mathrm{poly}(n) \cdot 2^{\mathrm{poly}(1/\epsilon)}$. In contrast, we give an algorithm that is significantly more efficient, but has a relaxed error guarantee.

Theorem 37. *There is an algorithm $\mathcal{B}$ with the following performance guarantee: Let f be any Boolean function and let* $\mathsf{opt} = \mathrm{dist}(f, \mathcal{H})$. *Given $0 < \epsilon, \delta < 1/2$ and access to independent* *uniform examples* $(x, f(x))$*, algorithm* $\mathcal{B}$ *outputs the (weights-based) representation of an LTF* f^* *which with probability* $1-\delta$ *satisfies* $\mathrm{dist}(f^*, f) \le 2^{-\Omega(\sqrt[3]{\log(1/\mathsf{opt})})} + \epsilon$*. The algorithm performs* $\tilde{O}(n^2) \cdot (1/\epsilon)^{O(\log^2(1/\epsilon))} \cdot \log(1/\delta)$ *bit operations.*

Proof. We describe the algorithm $\mathcal{B}$ in tandem with a proof of correctness. We start by estimating each Chow parameter of f (using the random labeled examples) to accuracy $O(\kappa(\epsilon)/\sqrt{n})$; we thus compute a vector $\vec{\alpha} \in \mathbb{R}^{n+1}$ that satisfies $\|\vec{\alpha} - \vec{\chi}_f\| \le \kappa(\epsilon)$. We then run algorithm `ChowReconstruct` (from Theorem 10) on input $\vec{\alpha}$, with its "ϵ" parameter set to $\kappa(\epsilon)$. The algorithm runs in time $\mathrm{poly}(1/\kappa(\epsilon)) \cdot \tilde{O}(n^2) \cdot \log(1/\delta)$ and outputs an LBF g such that with probability at least $1-\delta$ we have $d_{\mathrm{Chow}}(f, g) \le 6\kappa(\epsilon)$. By assumption, there exists an LTF h^* such that $\mathrm{dist}(h^*, f) \le \mathsf{opt}$. By Fact 6 we get $d_{\mathrm{Chow}}(h^*, f) \le \sqrt{2\mathsf{opt}}$. An application of the triangle inequality now gives $d_{\mathrm{Chow}}(g, h^*) \le \sqrt{2\mathsf{opt}} + 6\kappa(\epsilon)$. By Theorem 7, we thus obtain $\mathrm{dist}(g, h^*) \le 2^{-\Omega(\sqrt[3]{\log(1/\mathsf{opt})})} + \epsilon/2$. Writing the LBF g as $g(x) = P_1(v_0 + \sum_{i=1}^n v_i x_i)$, as above we have that $f^*(x) = \mathrm{sign}(v_0 + \sum_{i=1}^n v_i x_i)$ has $\mathrm{dist}(f, f^*) \le 2^{-\Omega(\sqrt[3]{\log(1/\mathsf{opt})})} + \epsilon$. It is easy to see that the running time is dominated by the execution of `ChowReconstruct`, and the proof of Theorem 37 is complete. □

Chapter 3

Inverse Shapley value problem

In this chapter we consider the common scenario in which each of n voters must cast a binary vote for or against some proposal. What is the best way to design such a voting scheme? Throughout the chapter we consider only *weighted voting schemes,* in which the proposal passes if a weighted sum of yes-votes exceeds a predetermined threshold. Weighted voting schemes are predominant in voting theory and have been extensively studied for many years, see [46, 153] and references therein. In computer science language, we are dealing with *linear threshold functions* (henceforth abbreviated as *LTFs*) over n Boolean variables.

If it is desired that each of the n voters should have the same "amount of power" over the outcome, then a simple majority vote is the obvious solution. However, in many scenarios it may be the case that we would like to assign different levels of voting power to the n voters – perhaps they are shareholders who own different amounts of stock in a corporation, or representatives of differently sized populations. In such a setting it is much less obvious how to design the right voting scheme; indeed, it is far from obvious how to correctly quantify the notion of the "amount of power" that a voter has under a given fixed voting scheme. As a simple example, consider an election with three voters who have voting weights 49, 49 and 2, in which a total of 51 votes are required for the proposition to pass. While the disparity between voting weights may at first suggest that the two voters with 49 votes each have most of the "power," any coalition of two voters is sufficient to pass the proposition and any single voter is insufficient, so the voting power of all three voters is in fact equal.

Many different *power indices* (methods of measuring the voting power of individuals under a given voting scheme) have been proposed over the course of decades. These include the Banzhaf index [9], the Deegan-Packel index [34], the Holler index [68], and others (see the extensive survey of de Keijzer [90]). Perhaps the best known, and certainly the oldest, of these indices is the *Shapley-Shubik index* [130], which is also known as the index of *Shapley values* (we shall henceforth refer to it as such). Informally, the Shapley value of a voter i among the n voters is the fraction of all $n!$ orderings of the voters in which she "casts the pivotal vote" (see Definition 38 in Section 3.1 for a precise definition, and [125] for much more on Shapley values). We shall work with the Shapley values throughout this chapter.

Given a particular weighted voting scheme (i.e., an n-variable linear threshold function), standard sampling-based approaches can be used to efficiently obtain highly accurate estimates of the n Shapley values (see also the works of [101, 8]). However, the *inverse* problem is much more challenging: given a vector of n desired values for the Shapley values, how can one design a weighted voting scheme that (approximately) achieves these Shapley values? This problem, which we refer to as the *Inverse Shapley Value Problem*, is quite natural and has received considerable attention; various heuristics and exponential-time algorithms have been proposed [6, 47, 91, 96], but prior to our work no provably correct and efficient algorithms were known.

Our Results. We give the first efficient algorithm with provable performance guarantees for the Inverse Shapley Value Problem. Our results apply to "reasonable" voting schemes; roughly, we say that a weighted voting scheme is "reasonable" if fixing a tiny fraction of the voting weight does not already determine the outcome, i.e., if the threshold of the linear threshold function is not too extreme. (See Definition 39 in Section 3.1 for a precise definition.) This seems to be a plausible property for natural voting schemes. Roughly speaking, we show that if there is any reasonable weighted voting scheme that approximately achieves the desired input vector of Shapley values, then our algorithm finds such a weighted voting scheme. Our algorithm runs in fixed polynomial time in n, the number of voters, for any constant error parameter $\epsilon > 0$. In a bit more detail, our first main theorem, stated informally, is as follows (see Section 3.5 for Theorem 63 which gives a precise theorem statement):

Main Theorem (arbitrary weights, informal statement). *There is a poly(n)-time algorithm with the following properties: The algorithm is given any constant accuracy parameter $\epsilon > 0$ and any vector of n real values $a(1), \dots, a(n)$. The algorithm has the following performance guarantee: if there is any monotone increasing reasonable LTF $f(x)$ whose Shapley values are very close to the given values $a(1), \dots, a(n)$, then with very high probability the algorithm outputs $v \in \mathbb{R}^n$, $\theta \in \mathbb{R}$ such that the linear threshold function $h(x) = \mathrm{sign}(v \cdot x - \theta)$ has Shapley values ϵ-close to those of f.*

Our second main theorem gives an even stronger guarantee if there is a weighted voting scheme with small weights (at most $\mathrm{poly}(n)$) whose Shapley values are close to the desired values. For this problem we give an algorithm which achieves $1/\mathrm{poly}(n)$ accuracy in $\mathrm{poly}(n)$ time. An informal statement of this result is (see Section 3.5 for Theorem 64 which gives a precise theorem statement):

Main Theorem (bounded weights, informal statement). *There is a poly(n, W)-time algorithm with the following properties: The algorithm is given a weight bound W and any vector of n real values $a(1), \dots, a(n)$. The algorithm has the following performance guarantee: if there is any monotone increasing reasonable LTF $f(x) = \mathrm{sign}(w \cdot x - \theta)$ whose Shapley values are very close to the given values $a(1), \dots, a(n)$ and where each w_i is an integer of magnitude at most W, then with very high probability the algorithm outputs $v \in \mathbb{R}^n$, $\theta \in \mathbb{R}$ such that the linear threshold function $h(x) = \mathrm{sign}(v \cdot x - \theta)$ has Shapley values $n^{-1/8}$-close to those of f.*

Discussion and Our Approach. At a high level, the Inverse Shapley Value Problem that we consider is similar to the "Chow Parameters Problem" that has been the subject of several recent papers [58, 119, 31]. The Chow parameters are another name for the n Banzhaf indices; the Chow

Parameters Problem is to output a linear threshold function which approximately matches a given input vector of Chow parameters. (To align with the terminology of the current paper, the "Chow Parameters Problem" might perhaps better be described as the "Inverse Banzhaf Problem.")

Let us briefly describe the approaches in [119] and [31] at a high level for the purpose of establishing a clear comparison with this chapter. Each of the papers [119, 31] combines structural results on linear threshold functions with an algorithmic component. The structural results in [119] deal with anti-concentration of affine forms $w \cdot x - \theta$ where $x \in \{-1, 1\}^n$ is uniformly distributed over the Boolean hypercube, while the algorithmic ingredient of [119] is a rather straightforward brute-force search. In contrast, the key structural results of [31] are geometric statements about how n-dimensional hyperplanes interact with the Boolean hypercube, which are combined with linear-algebraic (rather than anti-concentration) arguments. The algorithmic ingredient of [31] is more sophisticated, employing a boosting-based approach inspired by the work of [138, 70].

Our approach combines aspects of both the [119] and [31] approaches. Very roughly speaking, we establish new structural results which show that linear threshold functions have good anti-concentration (similar to [119]), and use a boosting-based approach derived from [138] as the algorithmic component (similar to [31]). However, this high-level description glosses over many "Shapley-specific" issues and complications that do not arise in these earlier works; below we describe two of the main challenges that arise, and sketch how we meet them in this work.

First challenge: establishing anti-concentration with respect to non-standard distributions. The Chow parameters (i.e., Banzhaf indices) have a natural definition in terms of the uniform distribution over the Boolean hypercube $\{-1, 1\}^n$. Being able to use the uniform distribution with its many nice properties (such as complete independence among all coordinates) is very useful in proving the required anti-concentration results that are at the heart of [119]. In contrast, it is not *a priori* clear what is (or even whether there exists) the "right" distribution over $\{-1, 1\}^n$ corresponding to the Shapley values. Here we derive such a distribution μ over $\{-1, 1\}^n$, but it is much less well-behaved than the uniform distribution (it is supported on a proper subset of $\{-1, 1\}^n$, and it is not even pairwise independent). Nevertheless, we are able to establish anti-concentration results for affine forms $w \cdot x - \theta$ corresponding to linear threshold functions under the distribution μ as required for our results. This is done by showing that any *reasonable* linear threshold function can be expressed with "nice" weights (see Theorem 40 of Section 3.1), and establishing anti-concentration for any "nice" weight vector by carefully combining anti-concentration bounds for p-biased distributions across a continuous family of different choices of p (see Section 3.3 for details).

Second challenge: using anti-concentration to solve the Inverse Shapley problem. The main algorithmic ingredient that we use is a procedure from [138]. Given a vector of values $(\mathbf{E}[f(x)x_i])$ for $i = \{1, \ldots, n\}$ (correlations between the unknown linear threshold function f and the individual input variables), it efficiently constructs a bounded function $g : \{-1, 1\}^n \to [-1, 1]$ which closely matches these correlations, i.e., $\mathbf{E}[f(x)x_i] \approx \mathbf{E}[g(x)x_i]$ for all i. Such a procedure is very useful for the Chow parameters problem, because the Chow parameters correspond precisely to the values $\mathbf{E}[f(x)x_i]$ – i.e., the degree-1 Fourier coefficients of f – with respect to the uniform distribution. (This correspondence is at the heart of Chow's original proof [24] showing that the exact values of the Chow parameters suffice to information-theoretically specify any linear threshold function; anti-concentration is used in [119] to extend Chow's original arguments about degree-1 Fourier coefficients to the setting of approximate reconstruction.)

For the inverse Shapley problem, there is no obvious correspondence between the correlations of individual input variables and the Shapley values. Moreover, without a notion of "degree-1 Fourier coefficients" for the Shapley setting, it is not clear why anti-concentration statements with respect to μ should be useful for approximate reconstruction. We deal with both these issues by developing a notion of the *degree-*1 *Fourier coefficients of* f *with respect to distribution* μ and relating these coefficients to the Shapley values [1]. (We actually require two related notions: one is the "coordinate correlation coefficient" $\mathbf{E}_{x \sim \mu}[f(x)x_i]$, which is necessary for the algorithmic [138] ingredient, and one is the "Fourier coefficient" $\hat{f}(i) = \mathbf{E}_{x \sim \mu}[f(x)L_i]$, which is necessary for Lemma 52, see below.) We define both notions and establish the necessary relations between them in Section 3.2.

Armed with the notion of the degree-1 Fourier coefficients under distribution μ, we prove a key result (Lemma 52) saying that if the LTF f is anti-concentrated under distribution μ, then any bounded function g which closely matches the degree-1 Fourier coefficients of f must be close to f in ℓ_1 distance with respect to μ. (This is why anti-concentration with respect to μ is useful for us.) From this point, exploiting properties of the [138] algorithm, we can pass from g to an LTF whose Shapley values closely match those of f.

Organization. Useful preliminaries are given in Section 3.1, including the crucial fact (Theorem 40) that all "reasonable" linear threshold functions have weight representations with "nice" weights. In Section 3.2 we define the distribution μ and the notions of Fourier coefficients and "coordinate correlation coefficients," and the relations between them, that we will need. At the end of that section we prove a crucial lemma, Lemma 52, which says that anti-concentration of affine forms and closeness in Fourier coefficients together suffice to establish closeness in ℓ_1 distance. Section 3.3 proves that "nice" affine forms have the required anti-concentration, and Section 3.4 describes the algorithmic tool from [138] that lets us establish closeness of coordinate correlation coefficients. Section 3.5 puts the pieces together to prove our main theorems. Finally, in Section **??** we conclude the chapter and present a few open problems.

3.1 Preliminaries

Notation and terminology. For $n \in \mathbb{Z}_+$, we denote by $[n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$. For $i, j \in \mathbb{Z}_+$, $i \leq j$, we denote $[i, j] \stackrel{\text{def}}{=} \{i, i+1, \ldots, j\}$.

Given a vector $w = (w_1, \ldots, w_n) \in \mathbb{R}^n$ we write $\|w\|_1$ to denote $\sum_{i=1}^n |w_i|$. A *linear threshold function*, or LTF, is a function $f : \{-1, 1\}^n \to \{-1, 1\}$ which is such that $f(x) = \text{sign}(w \cdot x - \theta)$ for some $w \in \mathbb{R}^n, \theta \in \mathbb{R}$.

[1]We note that Owen [121] has given a characterization of the Shapley values as a weighted average of p-biased influences (see also [79]). However, this is not as useful for us as our characterization in terms of "μ-distribution" Fourier coefficients, because we need to ultimately relate the Shapley values to anti-concentration with respect to μ.

Our arguments will also use a variant of linear threshold functions which we call *linear bounded functions* (LBFs). The projection function $P_1 : \mathbb{R} \to [-1, 1]$ is defined by $P_1(t) = t$ for $|t| \leq 1$ and $P_1(t) = \text{sign}(t)$ otherwise. An LBF $g : \{-1, 1\}^n \to [-1, 1]$ is a function $g(x) = P_1(w \cdot x - \theta)$.

Shapley values. Here and throughout the chapter we write $\mathbb{S}_n$ to denote the symmetric group of all $n!$ permutations over $[n]$. Given a permutation $\pi \in \mathbb{S}_n$ and an index $i \in [n]$, we write $x(\pi, i)$ to denote the string in $\{-1, 1\}^n$ that has a 1 in coordinate j if and only if $\pi(j) < \pi(i)$, and we write $x^+(\pi, i)$ to denote the string obtained from $x(\pi, i)$ by flipping coordinate i from -1 to 1. With this notation in place we can define the generalized Shapley indices of a Boolean function as follows:

Definition 38. (Generalized Shapley values) *Given* $f : \{-1, 1\}^n \to \{-1, 1\}$, *the* i-th generalized Shapley value of f *is the value*

$$\tilde{f}(i) \overset{def}{=} \mathbf{E}_{\pi \sim_R \mathbb{S}_n}[f(x^+(\pi, i)) - f(x(\pi, i))] \tag{3.1}$$

(where "$\pi \sim_R \mathbb{S}_n$" means that π is selected uniformly at random from $\mathbb{S}_n$).

A function $f : \{-1, 1\}^n \to \{-1, 1\}$ is said to be *monotone increasing* if for all $i \in [n]$, whenever two input strings $x, y \in \{-1, 1\}^n$ differ precisely in coordinate i and have $x_i = -1$, $y_i = 1$, it is the case that $f(x) \leq f(y)$. It is easy to check that for monotone functions our definition of generalized Shapley values agrees with the usual notion of Shapley values (which are typically defined only for monotone functions) up to a multiplicative factor of 2; in the rest of the chapter we omit "generalized" and refer to these values simply as the Shapley values of f.

We will use the following notion of the "distance" between the vectors of Shapley values for two functions $f, g : \{-1, 1\}^n \to [-1, 1]$:

$$d_{\text{Shapley}}(f, g) \overset{\text{def}}{=} \sqrt{\sum_{i=1}^{n} (\tilde{f}(i) - \tilde{g}(i))^2},$$

i.e., the Shapley distance $d_{\text{Shapley}}(f, g)$ is simply the Euclidean distance between the two n-dimensional vectors of Shapley values. Given a vector $a = (a(1), \ldots, a(n)) \in \mathbb{R}^n$ we will also use $d_{\text{Shapley}}(a, f)$ to denote $\sqrt{\sum_{i=1}^{n} (\tilde{f}(i) - a(i))^2}$.

The linear threshold functions that we consider. Our algorithmic results hold for linear threshold functions which are not too "extreme" (in the sense of having a very skewed threshold). We will use the following definition:

Definition 39. **(η-reasonable LTF)** *Let* $f : \{-1, 1\}^n \to \{-1, 1\}$, $f(x) = \text{sign}(w \cdot x - \theta)$ *be an LTF. For* $0 < \eta < 1$ *we say that* f *is* η-reasonable *if* $\theta \in [-(1 - \eta)\|w\|_1, (1 - \eta)\|w\|_1]$.

All our results will deal with η-reasonable LTFs; throughout the chapter η should be thought of as a small fixed absolute constant (such as $1/1000$). LTFs that are not η-reasonable do not seem to correspond to very interesting voting schemes since typically they will be very close to constant functions. (For example, even at $\eta = 0.99$, if the LTF $f(x) = \text{sign}(x_1 + \cdots + x_n - \theta)$ has a threshold $\theta > 0$ which makes it not an η-reasonable LTF, then f agrees with the constant function -1 on all but a $2^{-\Omega(n)}$ fraction of inputs in $\{-1,1\}^n$.)

Turning from the threshold to the weights, some of the proofs in our chapter will require us to work with LTFs that have "nice" weights in a certain technical sense. Prior work [129, 120] has shown that for any LTF, there is a weight vector realizing that LTF that has essentially the properties we need; however, since the exact technical condition that we require is not guaranteed by any of the previous works, we give a full proof that any LTF has a representation of the desired form. The following theorem is proved in Appendix B.1:

Theorem 40. *Let $f : \{-1,1\}^n \to \{-1,1\}$ be an η-reasonable LTF and $k \in [2,n]$. There exists a representation of f as $f(x) = \mathrm{sign}(v_0 + \sum_{i=1}^n v_i x_i)$ such that (after reordering coordinates so that condition (i) below holds) we have: (i) $|v_i| \geq |v_{i+1}|$, $i \in [n-1]$; (ii) $|v_0| \leq (1-\eta)\sum_{i=1}^n |v_i|$; and (iii) for all $i \in [0,k-1]$ we have $|v_i| \leq (2/\eta) \cdot \sqrt{n} \cdot k^{\frac{k}{2}} \cdot \sigma_k$, where $\sigma_k \overset{def}{=} \sqrt{\sum_{j \geq k} v_j^2}$.*

Tools from probability. We will use the following standard tail bound:

Theorem 41. **(Chernoff Bounds)** *Let X be a random variable taking values in $[-a,a]$ and let $X_1,\dots,X_t$ be i.i.d. samples drawn from X. Let $\overline{X} = \sum_{i=1}^t X_i/t$. Then for any $\gamma > 0$, we have*

$$\mathbf{P}r\left[\left|\overline{X} - \mathbf{E}[X]\right| \geq \gamma\right] \leq 2\exp(-\gamma^2 t/(2a^2)).$$

We will also use the Littlewood-Offord inequality for p-biased distributions over $\{-1,1\}^n$. One way to prove this is by using the LYM inequality (which can be found e.g. as Theorem 8.6 of [76]); for an explicit reference and proof of the following statement see e.g. [1].

Theorem 42. *Fix $\delta \in (0,1)$ and let D_δ denote the δ-*biased distribution over *$\{-1,1\}^n$ (under which each coordinate is set to 1 independently with probability δ.) Fix $w \in \mathbb{R}^n$ and define $S = \{i : |w_i| \geq \epsilon\}$. If $|S| \geq K$, then for all $\theta \in \mathbb{R}$ we have $\mathbf{P}r_{x\sim D_\delta}[|w \cdot x - \theta| < \epsilon] \leq \frac{1}{\sqrt{K\delta(1-\delta)}}$.*

Basic Facts about function spaces. We will use the following basic facts:

Fact 43. *The $n+1$ functions $1, x_1, \dots, x_n$ are linearly independent and form a basis for the subspace $V = \{f : \{-1,1\}^n \to \mathbb{R}$ and f is linear $\}$.*

Fact 44. *Fix any $\Omega \subseteq \{-1,1\}^n$ and let μ be a probability distribution over Ω such that $\mu(x) > 0$ for all $x \in \Omega$. We define $\langle f,g\rangle_\mu \overset{def}{=} \mathbf{E}_{\omega\sim\mu}[f(\omega)g(\omega)]$ for $f,g : \Omega \to \mathbb{R}$. Suppose that $f_1,\dots,f_m : \Omega \to \mathbb{R}$ is an orthonormal set of functions, i.e., $\langle f_i, f_j\rangle_\mu = \delta_{ij}$ for all $i,j \in [m]$. Then we have $\langle f,f\rangle_\mu^2 \geq \sum_{i=1}^m \langle f, f_i\rangle_\mu^2$. As a corollary, if $f,h : \Omega \to \{-1,1\}$ then we have $\sqrt{\sum_{i=1}^m \langle f-h, f_i\rangle_\mu^2} \leq 2\sqrt{\mathbf{P}r_{x\sim\mu}[f(x) \neq h(x)]}$.*

3.2 Analytic Reformulation of Shapley values

The definition of Shapley values given in Definition 38 is somewhat cumbersome to work with. In this section we derive alternate characterizations of Shapley values in terms of "Fourier coefficients" and "coordinate correlation coefficients" and establish various technical results relating Shapley values and these coefficients; these technical results will be crucially used in the proof of our main theorems.

There is a particular distribution μ that plays a central role in our reformulations. We start by defining this distribution μ and introducing some relevant notation, and then give our results.

The distribution μ. Let us define $\Lambda(n) \overset{\text{def}}{=} \sum_{0<k<n} \frac{1}{k} + \frac{1}{n-k}$; clearly we have $\Lambda(n) = \Theta(\log n)$, and more precisely we have $\Lambda(n) \le 2\log n$. We also define $Q(n,k)$ as $Q(n,k) \overset{\text{def}}{=} \frac{1}{k} + \frac{1}{n-k}$ for $0 < k < n$, so we have $\Lambda(n) = \sum_{k=1}^{n-1} Q(n,k)$.

For $x \in \{-1,1\}^n$ we write $\mathrm{wt}(x)$ to denote the number of 1's in x. We define the set B_n to be $B_n \overset{\text{def}}{=} \{x \in \{-1,1\}^n : 0 < \mathrm{wt}(x) < n\}$, i.e., $B_n = \{-1,1\}^n \setminus \{\mathbf{1}, -\mathbf{1}\}$.

The distribution μ is supported on B_n and is defined as follows: to make a draw from μ, sample $k \in \{1,\dots,n-1\}$ with probability $Q(n,k)/\Lambda(n)$. Choose $x \in \{-1,1\}^n$ uniformly at random from the k-th "weight level" of $\{-1,1\}^n$, i.e., from $\{-1,1\}^n_{=k} \overset{\text{def}}{=} \{x \in \{-1,1\}^n : \mathrm{wt}(x) = k\}$.

Useful notation. For $i = 0,\dots,n$ we define the "coordinate correlation coefficients" of a function $f : \{-1,1\}^n \to \mathbb{R}$ (with respect to μ) as:

$$f^*(i) \overset{\text{def}}{=} \mathbf{E}_{x\sim\mu}[f(x)\cdot x_i] \tag{3.2}$$

(here and throughout the chapter x_0 denotes the constant 1).

Later in this section we will define an orthonormal set of linear functions $L_0, L_1, \dots, L_n : \{-1,1\}^n \to \mathbb{R}$. We define the "Fourier coefficients" of f (with respect to μ) as:

$$\hat{f}(i) \overset{\text{def}}{=} \mathbf{E}_{x\sim\mu}[f(x)\cdot L_i(x)]. \tag{3.3}$$

An alternative expression for the Shapley values. We start by expressing the Shapley values in terms of the coordinate correlation coefficients:

Lemma 45. *Given $f : \{-1,1\}^n \to [-1,1]$, for each $i = 1,\dots,n$ we have*

$$\tilde{f}(i) = \frac{f(\mathbf{1}) - f(-\mathbf{1})}{n} + \frac{\Lambda(n)}{2}\cdot\left(f^*(i) - \frac{1}{n}\sum_{j=1}^n f^*(j)\right),$$

or equivalently,

$$f^*(i) = \frac{2}{\Lambda(n)}\cdot\left(\tilde{f}(i) - \frac{f(\mathbf{1}) - f(-\mathbf{1})}{n}\right) + \frac{1}{n}\sum_{j=1}^n f^*(j).$$

Proof. Recall that $\tilde{f}(i)$ can be expressed as follows:

$$\tilde{f}(i) = \mathbf{E}_{\pi\sim_R \mathbb{S}_n}[f(x^+(\pi,i)) - f(x(\pi,i))]. \tag{3.4}$$

Since the i-th coordinate of $x^+(\pi,i)$ is 1 and the i-th coordinate of $x(\pi,i)$ is -1, we see that $\tilde{f}(i)$ is a weighted sum of $\{f(x)x_i\}_{x\in\{-1,1\}^n}$. We now compute the weights associated with any such $x \in \{-1,1\}^n$.

- Let x be a string that has $\mathrm{wt}(x)$ coordinates that are 1 and has $x_i = 1$. Then the total number of permutations $\pi \in \mathbb{S}_n$ such that $x^+(\pi,i) = x$ is $(\mathrm{wt}(x)-1)!(n-\mathrm{wt}(x))!$. Consequently the weight associated with $f(x)x_i$ for such an x is $(\mathrm{wt}(x)-1)!\cdot(n-\mathrm{wt}(x))!/n!$.
- Now let x be a string that has $\mathrm{wt}(x)$ coordinates that are 1 and has $x_i = -1$. Then the total number of permutations $\pi \in \mathbb{S}_n$ such that $x(\pi,i) = x$ is $\mathrm{wt}(x)!(n-\mathrm{wt}(x)-1)!$. Consequently the weight associated with $f(x)x_i$ for such an x is $\mathrm{wt}(x)!\cdot(n-\mathrm{wt}(x)-1)!/n!$.

Thus we may rewrite Equation (3.4) as

$$\begin{aligned}\tilde{f}(i) &= \sum_{x:\{-1,1\}^n:x_i=1} \frac{(\mathrm{wt}(x)-1)!(n-\mathrm{wt}(x))!}{n!} f(x)\cdot x_i + \\ & \sum_{x:\{-1,1\}^n:x_i=-1} \frac{\mathrm{wt}(x)!(n-\mathrm{wt}(x)-1)!}{n!} f(x)\cdot x_i.\end{aligned}$$

Let us now define $\nu(f) \stackrel{\text{def}}{=} (f(\mathbf{1}) - f(-\mathbf{1}))/n$. Using the fact that $x_i^2 = 1$, it is easy to see that one gets

$$\begin{aligned}2\tilde{f}(i) &= 2\nu(f) + \\ & 2\left(\sum_{x\in B_n} f(x)\cdot\frac{(\mathrm{wt}(x)-1)!(n-\mathrm{wt}(x)-1)!}{n!}\cdot((n/2-\mathrm{wt}(x))+(nx_i)/2)\right) \\ &= 2\nu(f) + \sum_{x\in B_n}\left(f(x)\cdot\frac{(\mathrm{wt}(x)-1)!(n-\mathrm{wt}(x)-1)!}{(n-1)!}\cdot x_i + \right. \\ & \left. f(x)\cdot\frac{(\mathrm{wt}(x)-1)!(n-\mathrm{wt}(x)-1)!}{n!}\cdot(n-2\mathrm{wt}(x))\right) \\ &= 2\nu(f) + \sum_{x\in B_n}\left(f(x)\cdot\frac{n}{\mathrm{wt}(x)(n-\mathrm{wt}(x))\binom{n}{\mathrm{wt}(x)}}\cdot x_i + \right. \\ & \left. f(x)\cdot\frac{1}{\mathrm{wt}(x)(n-\mathrm{wt}(x))\binom{n}{\mathrm{wt}(x)}}\cdot(n-2\mathrm{wt}(x))\right). \end{aligned} \tag{3.5}$$

We next observe that $n - 2\mathrm{wt}(x) = -(\sum_{j\in[n]} x_j)$. Next, let us define $P(n,k)$ (for $k \in [1, n-1]$) as follows :

$$P(n,k) \stackrel{\text{def}}{=} \frac{Q(n,k)}{\binom{n}{k}} = \frac{\frac{1}{k}+\frac{1}{n-k}}{\binom{n}{k}}.$$

So we may rewrite Equation (3.5) in terms of $P(n, \mathrm{wt}(x))$ as

$$2\tilde{f}(i) = 2\nu(f) + \sum_{x \in B_n} [f(x) \cdot x_i \cdot P(n, \mathrm{wt}(x))] - \sum_{x \in B_n} \left[f(x) \cdot P(n, \mathrm{wt}(x)) \cdot (\sum_{i=1}^{n} x_i)/n \right].$$

We have

$$\sum_{x \in B_n} P(n, \mathrm{wt}(x)) = \sum_{k=1}^{n-1} \sum_{x \in \{-1,1\}^n_{=k}} P(n, \mathrm{wt}(x)) = \sum_{k=1}^{n-1} \binom{n}{k} \cdot P(n,k) = \sum_{k=1}^{n-1} Q(n,k) = \Lambda(n),$$

and consequently we get

$$2\tilde{f}(i) = 2\nu(f) + \Lambda(n) \cdot \left(\mathop{\mathbf{E}}_{x \sim \mu} [f(x) \cdot x_i] - \mathop{\mathbf{E}}_{x \sim \mu} \left[f(x) \cdot (\sum_{i=1}^{n} x_i)/n \right] \right),$$

finishing the proof. □

Construction of a Fourier basis for distribution μ. For all $x \in B_n$ we have that $\mu(x) > 0$, and consequently by Fact 43 we know that the functions $1, x_1, \ldots, x_{n+1}$ form a basis for the subspace of linear functions from $B_n \to \mathbb{R}$. By Gram-Schmidt orthogonalization, we can obtain an orthonormal basis $L_0, \ldots, L_n$ for this subspace, i.e., a set of linear functions such that $\langle L_i, L_i \rangle_\mu = 1$ for all i and $\langle L_i, L_j \rangle_\mu = 0$ for all $i \neq j$.

We now give explicit expressions for these basis functions. We start by defining $L_0 : B_n \to \mathbb{R}$ as $L_0 : x \mapsto 1$. Next, by symmetry, we can express each L_i as

$$L_i(x) = \alpha(x_1 + \ldots + x_n) + \beta x_i.$$

Using the orthonormality properties it is straightforward to solve for α and β. The following Lemma gives the values of α and β:

Lemma 46. *For the choices*

$$\alpha \overset{\mathit{def}}{=} \frac{1}{n} \cdot \left(\sqrt{\frac{\Lambda(n)}{n\Lambda(n) - 4(n-1)}} - \frac{\sqrt{\Lambda(n)}}{2} \right), \quad \beta \overset{\mathit{def}}{=} \frac{\sqrt{\Lambda(n)}}{2},$$

the set $\{L_i\}_{i=0}^n$ is an orthonormal set of linear functions under the distribution μ.

We note for later reference that $\alpha = -\Theta\left(\frac{\sqrt{\log n}}{n}\right)$ and $\beta = \Theta(\sqrt{\log n})$.

We start with the following proposition which gives an explicit expression for $\mathbf{E}_{x \sim \mu}[x_i x_j]$ when $i \neq j$; we will use it in the proof of Lemma 46.

Proposition 47. *For all $1 \leq i < j \leq n$ we have $\mathbf{E}_{x \sim \mu}[x_i x_j] = 1 - \frac{4}{\Lambda(n)}$.*

Proof. For brevity let us write $A_k = \{-1,1\}^n_{=k}$, i.e., $A_k = \{x \in \{-1,1\}^n : \mathrm{wt}(x) = k\}$, the k-th "slice" of the hypercube. Since μ is supported on $B_n = \cup_{k=1}^{n-1} A_k$, we have

$$\mathbf{E}_{x\sim\mu}[x_i x_j] = \sum_{0<k<n} \mathop{\mathbf{E}}_{x\sim\mu}[x_i x_j \mid x \in A_k] \cdot \mathop{\mathbf{P}r}_{x\sim\mu}[x \in A_k].$$

If $k = 1$ or $n-1$, it is clear that

$$\mathbf{E}_{x\sim\mu}[x_i x_j \mid x \in A_k] = 1 - \frac{2}{n} - \frac{2}{n} = 1 - \frac{4}{n},$$

and when $2 \le k \le n-2$, we have

$$\mathbf{E}_{x\sim\mu}[x_i x_j \mid x \in A_k] = \frac{1}{\binom{n}{k}} \cdot \left(2\binom{n-2}{k-2} + 2\binom{n-2}{k} - \binom{n}{k} \right).$$

Recall that $\Lambda(n) = \sum_{0<k<n} \frac{1}{k} + \frac{1}{n-k}$ and $Q(n,k) = \frac{1}{k} + \frac{1}{n-k}$ for $0 < k < n$. This means that we have

$$\mathop{\mathbf{P}r}_{x\sim\mu}[x \in A_k] = Q(n,k)/\Lambda(n).$$

Thus we may write $\mathbf{E}_{x\sim\mu}[x_i x_j]$ as

$$\begin{aligned}\mathbf{E}_{x\sim\mu}[x_i x_j] \quad &= \quad \sum_{2\le k\le n-2} \frac{Q(n,k)}{\Lambda(n)} \cdot \mathbf{E}_{x\sim\mu}[x_i x_j \mid x \in A_k] + \\ &\qquad \sum_{k\in\{1,n-1\}} \frac{Q(n,k)}{\Lambda(n)} \cdot \mathbf{E}_{x\sim\mu}[x_i x_j \mid x \in A_k].\end{aligned}$$

For the latter sum, we have

$$\sum_{k\in\{1,n-1\}} \frac{Q(n,k)}{\Lambda(n)} \cdot \mathbf{E}_{x\sim\mu}[x_i x_j \mid x \in A_k] = \frac{1}{\Lambda(n)} \left(1 - \frac{4}{n}\right) \cdot \frac{2n}{n-1}.$$

For the former, we can write

$$\begin{aligned}&\sum_{k=2}^{n-2} \frac{Q(n,k)}{\Lambda(n)} \cdot \mathbf{E}_{x\sim\mu}[x_i x_j \mid x \in A_k] \\ = \quad &\sum_{k=2}^{n-2} \frac{1}{\Lambda(n)} \frac{(k-1)!(n-k-1)!}{(n-1)!} \cdot \left(2\binom{n-2}{k-2} + 2\binom{n-2}{k} - \binom{n}{k} \right) \\ = \quad &\sum_{k=2}^{n-2} \frac{1}{\Lambda(n)} \cdot \left(\frac{2(k-1)}{(n-1)(n-k)} + \frac{2(n-k-1)}{(n-1)k} - \frac{n}{k(n-k)} \right) \\ = \quad &\sum_{k=2}^{n-2} \frac{1}{\Lambda(n)} \cdot \left(\frac{2}{n-k} - \frac{2}{n-1} + \frac{2}{k} - \frac{2}{n-1} - \frac{1}{k} - \frac{1}{n-k} \right) \\ = \quad &\sum_{k=2}^{n-2} \frac{1}{\Lambda(n)} \cdot \left(\frac{1}{n-k} + \frac{1}{k} - \frac{4}{n-1} \right).\end{aligned}$$

Thus, we get that overall $\mathbf{E}_{x\sim\mu}[x_i x_j]$ equals

$$\begin{aligned}
& \frac{1}{\Lambda(n)}\left(1-\frac{4}{n}\right)\cdot\frac{2n}{n-1}+\sum_{k=2}^{n-2}\frac{1}{\Lambda(n)}\cdot\left(\frac{1}{n-k}+\frac{1}{k}-\frac{4}{n-1}\right) \\
= \; & \frac{1}{\Lambda(n)}\left(2+\frac{2}{n-1}-\frac{8}{n-1}\right)+\frac{1}{\Lambda(n)}\left(\sum_{k=2}^{n-2}\frac{1}{k}+\frac{1}{n-k}\right)-\frac{4}{\Lambda(n)}+\frac{8}{\Lambda(n)(n-1)} \\
= \; & \frac{1}{\Lambda(n)}\left(\sum_{k=1}^{n-1}Q(n,k)\right)-\frac{4}{\Lambda(n)}=1-\frac{4}{\Lambda(n)},
\end{aligned}$$

as was to be shown. □

Proof of Lemma 46. We begin by observing that

$$\mathbf{E}_{x\sim\mu}[L_i(x)L_0(x)]=\mathbf{E}_{x\sim\mu}[L_i(x)]=\mathbf{E}_{x\sim\mu}[\alpha(x_1+\ldots+x_n)+\beta x_i]=0$$

since $\mathbf{E}_{x\sim\mu}[x_i]=0$. Next, we solve for α and β using the orthonormality conditions on the set $\{L_i\}_{i=1}^n$. As $\mathbf{E}_{x\sim\mu}[L_i(x)L_j(x)]=0$ and $\mathbf{E}_{x\sim\mu}[L_i(x)L_i(x)]=1$, we get that $\mathbf{E}_{x\sim\mu}[L_i(x)(L_i(x)-L_j(x))]=1$. This gives

$$\begin{aligned}
\mathbf{E}_{x\sim\mu}[L_i(x)\cdot(L_i(x)-L_j(x))] &= \mathbf{E}_{x\sim\mu}[L_i(x)\cdot\beta(x_i-x_j)] \\
&= \mathbf{E}_{x\sim\mu}[\beta((\alpha+\beta)x_i+\alpha x_j)\cdot(x_i-x_j)] \\
&= \alpha\beta+\beta^2-\alpha\beta-\beta^2\mathbf{E}_{x\sim\mu}[x_j x_i] \\
&= \beta^2(1-\mathbf{E}_{x\sim\mu}[x_i x_j])=4\beta^2/\Lambda(n)=1,
\end{aligned}$$

where the penultimate equation above uses Proposition 47. Thus, we have shown that $\beta=\frac{\sqrt{\Lambda(n)}}{2}$. To solve for α, we note that

$$\sum_{i=1}^n L_i(x)=(\alpha n+\beta)(x_1+\ldots+x_n).$$

However, since the set $\{L_i\}_{i=1}^n$ is orthonormal with respect to the distribution μ, we get that

$$\mathbf{E}_{x\sim\mu}[(L_1(x)+\ldots+L_n(x))(L_1(x)+\ldots+L_n(x))]=n$$

and consequently

$$(\alpha n+\beta)^2\,\mathbf{E}_{x\sim\mu}[(x_1+\ldots+x_n)(x_1+\ldots+x_n)]=n$$

Now, using Proposition 47, we get

$$\begin{aligned}
\mathbf{E}_{x\sim\mu}[(x_1+\ldots+x_n)(x_1+\ldots+x_n)] &= \sum_{i=1}^n\mathbf{E}_{x\sim\mu}[x_i^2]+\sum_{i\neq j}\mathbf{E}_{x\sim\mu}[x_i x_j] \\
&= n+n(n-1)\cdot\left(1-\frac{4}{\Lambda(n)}\right)
\end{aligned}$$

Thus, we get that

$$(\alpha n + \beta)^2 \cdot \left(n + n(n-1) \cdot \left(1 - \frac{4}{\Lambda(n)}\right)\right) = n.$$

Simplifying further,

$$(\alpha n + \beta) = \sqrt{\frac{\Lambda(n)}{n\Lambda(n) - 4(n-1)}}$$

and thus

$$\alpha = \frac{1}{n} \cdot \left(\sqrt{\frac{\Lambda(n)}{n\Lambda(n) - 4(n-1)}} - \frac{\sqrt{\Lambda(n)}}{2}\right)$$

as was to be shown. □

Relating the Shapley values to the Fourier coefficients. The next lemma gives a useful expression for $\hat{f}(i)$ in terms of $\tilde{f}(i)$:

Lemma 48. *Let $f : \{-1,1\}^n \to [-1,1]$ be any bounded function. Then for each $i = 1, \ldots, n$ we have*

$$\hat{f}(i) = \frac{2\beta}{\Lambda(n)} \cdot \left(\tilde{f}(i) - \frac{f(\mathbf{1}) - f(-\mathbf{1})}{n}\right) + \frac{1}{n} \cdot \sum_{j=1}^{n} \hat{f}(j).$$

Proof. Lemma 46 gives us that $L_i(x) = \alpha(x_1 + \ldots + x_n) + \beta x_i$, and thus we have

$$\begin{aligned}\hat{f}(i) \equiv \mathbf{E}_{x\sim\mu}[f(x) \cdot L_i(x)] &= \alpha \left(\sum_{j=1}^{n} \mathbf{E}_{x\sim\mu}[f(x) \cdot x_j]\right) + \beta \mathbf{E}_{x\sim\mu}[f(x) \cdot x_i] \\ &= \alpha \sum_{j=1}^{n} f^*(j) + \beta f^*(i). \qquad (3.6)\end{aligned}$$

Summing this for $i = 1$ to n, we get that

$$\sum_{j=1}^{n} \hat{f}(j) = (\alpha n + \beta) \sum_{j=1}^{n} f^*(j). \qquad (3.7)$$

Plugging this into (3.6), we get that

$$f^*(i) = \frac{1}{\beta} \cdot \left(\hat{f}(i) - \frac{\alpha}{\alpha n + \beta} \cdot \sum_{j=1}^{n} \hat{f}(j)\right) \qquad (3.8)$$

Now recall that from Lemma 45, we have

$$\begin{aligned}\tilde{f}(i) &= \nu(f) + \frac{\Lambda(n)}{2} \cdot \left(\mathop{\mathbf{E}}_{x\sim\mu}[f(x) \cdot x_i] - \mathop{\mathbf{E}}_{x\sim\mu}\left[f(x) \cdot (\sum_{i=1}^{n} x_i)/n\right]\right) \\ &= \nu(f) + \frac{\Lambda(n)}{2} \cdot \left(f^*(i) - \frac{\sum_{j=1}^{n} f^*(j)}{n}\right)\end{aligned}$$

where $\nu(f) = (f(\mathbf{1}) - f(-\mathbf{1}))/n$. Hence, combining the above with (3.7) and (3.8), we get

$$\frac{1}{\beta} \cdot \left(\hat{f}(i) - \frac{\alpha}{\alpha n + \beta} \cdot \sum_{j=1}^{n} \hat{f}(j) \right) = \frac{2}{\Lambda(n)} \cdot (\tilde{f}(i) - \nu(f)) + \frac{1}{n(\alpha n + \beta)} \cdot \sum_{j=1}^{n} \hat{f}(j).$$

From this, it follows that

$$\frac{1}{\beta} \cdot \hat{f}(i) = \frac{2}{\Lambda(n)} \cdot (\tilde{f}(i) - \nu(f)) + \frac{1}{\alpha n + \beta} \cdot \left(\frac{1}{n} + \frac{\alpha}{\beta} \right) \cdot \sum_{j=1}^{n} \hat{f}(j),$$

and hence

$$\hat{f}(i) = \frac{2\beta}{\Lambda(n)} \cdot (\tilde{f}(i) - \nu(f)) + \frac{1}{n} \cdot \sum_{j=1}^{n} \hat{f}(j)$$

as desired. □

Bounding Shapley distance in terms of Fourier distance. Recall that the Shapley distance $d_{\text{Shapley}}(f, g)$ between $f, g : \{-1, 1\}^n \to [-1, 1]$ is defined as $d_{\text{Shapley}}(f, g) \stackrel{\text{def}}{=} \sqrt{\sum_{i=1}^{n} (\tilde{f}(i) - \tilde{g}(i))^2}$. We define the *Fourier distance* between f and g as $d_{\text{Fourier}}(f, g) \stackrel{\text{def}}{=} \sqrt{\sum_{i=0}^{n} (\hat{f}(i) - \hat{g}(i))^2}$.

Our next lemma shows that if the Fourier distance between f and g is small then so is the Shapley distance.

Lemma 49. *Let $f, g : \{-1, 1\}^n \to [-1, 1]$. Then,*

$$d_{\text{Shapley}}(f, g) \leq \frac{4}{\sqrt{n}} + \frac{\Lambda(n)}{2\beta} \cdot d_{\text{Fourier}}(f, g).$$

Proof. Let $\nu(f) = (f(\mathbf{1}) - f(-\mathbf{1}))/n$ and $\nu(g) = (g(\mathbf{1}) - g(-\mathbf{1}))/n$. From Lemma 48, we have that for all $1 \leq i \leq n$,

$$\frac{\Lambda(n)}{2\beta} \cdot \left(\hat{f}(i) - \frac{\sum_{j=1}^{n} \hat{f}(j)}{n} \right) + \nu(f) = \tilde{f}(i).$$

Using a similar relation for g, we get that for every $1 \leq i \leq n$,

$$\frac{\Lambda(n)}{2\beta} \cdot \left(\hat{f}(i) - \frac{\sum_{j=1}^{n} \hat{f}(j)}{n} - \hat{g}(i) + \frac{\sum_{j=1}^{n} \hat{g}(j)}{n} \right) + \nu(f) - \nu(g) = \tilde{f}(i) - \tilde{g}(i).$$

We next define the following vectors: let $v \in \mathbb{R}^n$ be defined by $v_i = \tilde{f}(i) - \tilde{g}(i)$, $i \in [n]$ (so our goal is to bound $\|v\|_2$). Let $u \in \mathbb{R}^n$ be defined by $u_i = \nu(f) - \nu(g)$, $i \in [n]$. Finally, let $w \in \mathbb{R}^n$ be defined by

$$w_i = \left(\hat{f}(i) - \frac{\sum_{j=1}^{n} \hat{f}(j)}{n} - \hat{g}(i) + \frac{\sum_{j=1}^{n} \hat{g}(j)}{n} \right), \quad i \in [n].$$

With these definitions the vectors u, v and w satisfy $\frac{\Lambda(n)}{2\beta} \cdot w + u = v$, and hence we have

$$\|v\|_2 \le \|u\|_2 + \frac{\Lambda(n)}{2\beta} \cdot \|w\|_2.$$

Since the range of f and g is $[-1, 1]$, we immediately have that

$$\|u\|_2 = \left(\frac{f(\mathbf{1}) - g(\mathbf{1}) - f(-\mathbf{1}) + g(-\mathbf{1})}{n}\right) \cdot \sqrt{n} \le \frac{4}{\sqrt{n}},$$

so all that remains is to bound $\|w\|_2$ from above. To do this, let us define another vector $w' \in \mathbb{R}^n$ by $w'_i = \hat{f}(i) - \hat{g}(i)$. Let $\mathbf{e} \in \mathbb{R}^n$ denote the unit vector $\mathbf{e} = (1/\sqrt{n}, \dots, 1/\sqrt{n})$. Letting $w'_{\mathbf{e}}$ denote the projection of w along $\mathbf{e}$, it is easy to see that

$$w'_{\mathbf{e}} = \left(\frac{\sum_{j=1}^{n}(\hat{f}(j) - \hat{g}(j))}{n}, \dots, \frac{\sum_{j=1}^{n}(\hat{f}(j) - \hat{g}(j))}{n}\right).$$

This means that $w = w' - w'_{\mathbf{e}}$ and that w is the projection of w' in the space orthogonal to $\mathbf{e}$. Consequently we have $\|w\|_2 \le \|w'\|_2$, and hence

$$\|v\|_2 \le \frac{4}{\sqrt{n}} + \frac{\Lambda(n)}{2\beta}\|w'\|_2$$

as was to be shown. □

Bounding Fourier distance by "correlation distance." The following lemma will be useful for us since it lets us bound from above Fourier distance in terms of the distance between vectors of correlations with individual variables:

Lemma 50. *Let $f, g : \{-1, 1\}^n \to \mathbb{R}$. Then we have*

$$d_{\text{Fourier}}(f, g) \le O(\sqrt{\log n}) \cdot \sqrt{\sum_{i=0}^{n}(f^*(i) - g^*(i))^2}.$$

Proof. We first observe that $\hat{f}(0) = f^*(0)$ and $\hat{g}(0) = g^*(0)$, so $(\hat{f}(0) - \hat{g}(0))^2 = (f^*(0) - g^*(0))^2$. Consequently it suffices to prove that

$$\sqrt{\sum_{i=1}^{n}(\hat{f}(i) - \hat{g}(i))^2} \le O(\sqrt{\log n}) \cdot \sqrt{\sum_{i=1}^{n}(f^*(i) - g^*(i))^2},$$

which is what we show below.

From (3.6), we get

$$\hat{f}(i) = \alpha \sum_{j=1}^{n} f^*(j) + \beta f^*(i) \quad \text{and} \quad \hat{g}(i) = \alpha \sum_{j=1}^{n} g^*(j) + \beta g^*(i).$$

and thus we have

$$(\hat{f}(i) - \hat{g}(i)) = \alpha \left(\sum_{j=1}^{n} f^*(j) - \sum_{j=1}^{n} g^*(j) \right) + \beta(f^*(i) - g^*(i)).$$

Now consider vectors $u, v, w \in \mathbb{R}^n$ where for $i \in [n]$,

$$u_i = (\hat{f}(i) - \hat{g}(i)), \quad v_i = \left(\sum_{j=1}^{n} f^*(j) - \sum_{j=1}^{n} g^*(j) \right), \quad \text{and} \quad w_i = (f^*(i) - g^*(i))$$

By combining the triangle inequality and Cauchy-Schwarz, we have

$$\|u\|_2^2 \leq 2(\alpha^2 \|v\|_2^2 + \beta^2 \|w\|_2^2),$$

and moreover

$$\|v\|_2^2 = n \left(\sum_{j=1}^{n} f^*(j) - \sum_{j=1}^{n} g^*(j) \right)^2 \leq n^2 \left(\sum_{j=1}^{n} (f^*(j) - g^*(j))^2 \right) = n^2 \|w\|_2^2.$$

Hence, we obtain

$$\|u\|_2^2 \leq 2(\alpha^2 n^2 + \beta^2) \|w\|_2^2$$

Recalling that $\alpha^2 n^2 = \Theta(\log n)$ and $\beta^2 = \Theta(\log n)$, we conclude that

$$d_{\text{Fourier}}(f, g) = \sqrt{\sum_{i=1}^{n} (\hat{f}(i) - \hat{g}(i))^2} \leq O(\sqrt{\log n}) \cdot \sqrt{\sum_{i=1}^{n} (f^*(i) - g^*(i))^2}$$

which completes the proof. □

From Fourier closeness to ℓ_1-closeness. An important technical ingredient in our work is the notion of an affine form $\ell(x)$ having "good anti-concentration" under distribution μ; we now give a precise definition to capture this.

Definition 51 (Anti-concentration). *Fix $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$, and let the affine form $\ell(x)$ be $\ell(x) \overset{def}{=} w \cdot x - \theta$. We say that $\ell(x)$ is (δ, κ)*-anti-concentrated under μ *if* $\mathbf{Pr}_{x \sim \mu}[|\ell(x)| \leq \delta] \leq \kappa$.

The next lemma plays a crucial role in our results. It essentially shows that for $f = \text{sign}(w \cdot x - \theta)$, if the affine form $\ell(x) = w \cdot x - \theta$ is anti-concentrated, then *any* bounded function $g : \{-1, 1\}^n \to [-1, 1]$ that has $d_{\text{Fourier}}(f, g)$ small must in fact be close to f in ℓ_1 distance under μ.

Lemma 52. *Let $f : \{-1, 1\}^n \to \{-1, 1\}$, $f = \text{sign}(w \cdot x - \theta)$ be such that $w \cdot x - \theta$ is (δ, κ)-anti-concentrated under μ (for some $\kappa \leq 1/2$), where $|\theta| \leq \|w\|_1$. Let $g : \{-1, 1\}^n \to [-1, 1]$ be such that $d_{\text{Fourier}}(f, g) \leq \rho$. Then we have*

$$\mathbf{E}_{x \sim \mu}[|f(x) - g(x)|] \leq (4\|w\|_1 \sqrt{\rho})/\delta + 2\kappa.$$

Proof. Let us rewrite $\ell(x) \stackrel{\text{def}}{=} w \cdot x - \theta$ as a linear combination of the orthonormal basis elements $L_0, L_1, \dots, L_n$ (w.r.t. μ), i.e.,

$$\ell(x) = \hat{\ell}(\emptyset)L_0 + \sum_{i=1}^{n} \hat{\ell}(i)L_i.$$

Recalling the definitions of L_i for $i = 1, \dots, n$ and the fact that $L_0 = 1$, we get $\hat{\ell}(\emptyset) = -\theta$.

We first establish an upper bound on $\theta^2 + \sum_{j=1}^{n} \hat{\ell}(j)^2$ as follows :

$$\begin{aligned} \theta^2 + \sum_{j=1}^{n} \hat{\ell}(j)^2 = \mathbf{E}_{x\sim\mu}[(w \cdot x - \theta)^2] &\le 2\mathbf{E}_{x\sim\mu}[(w \cdot x)^2] + 2\theta^2 \\ &\le 2\|w\|_1^2 + 2\|w\|_1^2 = 4\|w\|_1^2. \end{aligned}$$

The first equality above uses the fact that the L_i's are orthonormal under μ, while the first inequality uses $(a+b)^2 \le 2(a^2+b^2)$ for $a, b \in \mathbb{R}$. The second inequality uses the assumed bound on $|\theta|$ and the fact that $|w \cdot x|$ is always at most $\|w\|_1$.

Next, Plancherel's identity (linearity of expectation) gives us that

$$\begin{aligned} \mathbf{E}_{x\sim\mu}[(f(x) - g(x)) \cdot (w \cdot x - \theta)] &= \theta(\hat{g}(0) - \hat{f}(0)) + \sum_{j=1}^{n} \hat{\ell}(i)(\hat{f}(i) - \hat{g}(i)) \\ &\le \sqrt{\sum_{j=0}^{n} (\hat{f}(j) - \hat{g}(j))^2} \cdot \sqrt{\theta^2 + \sum_{j=1}^{n} \hat{\ell}(i)^2} \\ &\le 2\|w\|_1\sqrt{\rho} \qquad (3.9) \end{aligned}$$

where the first inequality is Cauchy-Schwarz and the second follows by the conditions of the lemma.

Now note that since $f = \operatorname{sign}(w \cdot x - \theta)$, for all $x \in \{-1,1\}^n$ we have

$$(f(x) - g(x)) \cdot (w \cdot x - \theta) = |f(x) - g(x)| \cdot |w \cdot x - \theta|$$

Let E denote the event that $|w \cdot x - \theta| > \delta$. Using the fact that the affine form $w \cdot x - \theta$ is (δ, κ)-anti-concentrated, we get that $\mathbf{P}r[E] \ge 1 - \kappa$, and hence

$$\begin{aligned} \mathbf{E}_{x\sim\mu}[(f(x) - g(x)) \cdot (w \cdot x - \theta)] &\ge \mathbf{E}_{x\sim\mu}[(f(x) - g(x)) \cdot (w \cdot x - \theta) \mid E]\,\mathbf{P}r[E] \\ &\ge \delta(1-\kappa)\mathbf{E}_{x\sim\mu}[|f(x) - g(x)| \mid E]. \end{aligned}$$

Recalling that $\kappa \le 1/2$, this together with (3.9) implies that

$$\mathbf{E}_{x\sim\mu}[|f(x) - g(x)| \mid E] \le \frac{4\|w\|_1\sqrt{\rho}}{\delta},$$

which in turn implies (since $|f(x) - g(x)| \le 2$ for all $x \in \{-1,1\}^n$) that

$$\mathbf{E}_{x\sim\mu}[|f(x) - g(x)|] \le \frac{4\|w\|_1\sqrt{\rho}}{\delta} + 2\kappa$$

as was to be shown. $\square$

3.3 A Useful Anti-concentration Result

In this section we prove an anti-concentration result for monotone increasing η-reasonable affine forms (see Definition 39) under the distribution μ. Note that even if k is a constant the result gives an anti-concentration probability of $O(1/\log n)$; this will be crucial in the proof of our first main result in Section 3.5.

Theorem 53. *Let $L(x) = w_0 + \sum_{i=1}^{n} w_i x_i$ be a monotone increasing η-reasonable affine form, so $w_i \geq 0$ for $i \in [n]$ and $|w_0| \leq (1-\eta)\sum_{i=1}^{n}|w_i|$. Let $k \in [n], 0 < \zeta < 1/2$, $k \geq 2/\eta$ and $r \in \mathbb{R}_+$ be such that $|S| \geq k$, where $S := \{i \in [n] : |w_i| \geq r\}$. Then*

$$\mathop{\mathbf{P}r}_{x\sim\mu}\left[|L(x)| < r\right] = O\left(\frac{1}{\log n}\cdot\frac{1}{k^{1/3-\zeta}}\cdot\left(\frac{1}{\zeta}+\frac{1}{\eta}\right)\right).$$

This theorem essentially says that under the distribution μ, the random variable $L(x)$ falls in the interval $[-r, r]$ with only a very small probability. Such theorems are known in the literature as "anti-concentration" results, but almost all such results are for the uniform distribution or for other product distributions, and indeed the proofs of such results typically crucially use the fact that the distributions are product distributions.

In our setting, the distribution μ is not even a pairwise independent distribution, so standard approaches for proving anti-concentration cannot be directly applied. Instead, we exploit the fact that μ is a *symmetric* distribution; a distribution is symmetric if the probability mass it assigns to an n-bit string $x \in \{-1,1\}^n$ depends only on the number of 1's of x (and not on their location within the string). This enables us to perform a somewhat delicate reduction to known anti-concentration results for biased product distributions. Our proof adopts a point of view which is inspired by the combinatorial proof of the basic Littlewood-Offord theorem (under the uniform distribution on the hypercube) due to Benjamini et. al. [13]. The detailed proof is given in the following subsection.

Proof of Theorem 53.

Recall that $\{-1,1\}^n_{=i}$ denotes the i-th "weight level" of the hypercube, i.e., $\{x \in \{-1,1\}^n : \mathrm{wt}(x) = i\}$. We view a random draw $x \sim \mu$ as being done according to a two-stage process:

1. Draw $i \in [n-1]$ with probability $q(n,i) \stackrel{\text{def}}{=} Q(n,i)/\Lambda(n)$. (Note that this is the probability μ assigns to $\{-1,1\}^n_{=i}$.)
2. Independently pick a uniformly random permutation $\pi : [n] \to [n]$, i.e., $\pi \sim_R \mathbb{S}_n$. The string x is defined to have $x_{\pi(1)} = \ldots = x_{\pi(i)} = 1$ and $x_{\pi(i+1)} = \ldots = x_{\pi(n)} = -1$.

It is easy to see that the above description of μ is equivalent to its original definition. Another crucial observation is that any symmetric distribution can be sampled in the same way, with $q(n,k)$ being the only quantity dependent on the particular distribution. We next define a (r,i)-balanced permutation.

Definition 54 ((r,i)-balanced permutation)**.** *A permutation $\pi : [n] \to [n]$ is called (r,i)-balanced if $|w_0 + \sum_{j=1}^{i} w_{\pi(j)} - \sum_{j=i+1}^{n} w_{\pi(j)}| \le r$.*

For $i \in [n-1]$, let us denote by $p(r,i)$ the fraction of all $n!$ permutations that are (r,i) balanced. That is,

$$p(r,i) = \mathop{\mathbf{Pr}}_{\pi \sim_R \mathbb{S}_n}\left[|w_0 + \sum_{j=1}^{i} w_{\pi(j)} - \sum_{j=i+1}^{n} w_{\pi(j)}| \le r \right].$$

At this point, as done in [13], we use the above two-stage process defining μ to express the desired "small ball" probability in a more convenient way. Conditioning on the event that the i-th layer is selected in the first stage, the probability that $|L(x)| < r$ is $p(r,i)$. By the law of total probability we can write:

$$\mathop{\mathbf{Pr}}_{x \sim \mu}\left[|L(x)| < r\right] = \sum_{i=1}^{n-1} p(r,i) q(n,i).$$

We again observe that $p(r,i)$ is only dependent on the affine form $L(x)$ and does not depend on the particular symmetric distribution; $q(n,i)$ is the only part dependent on the distribution. The high-level idea of bounding the quantity $\sum_{i=1}^{n-1} p(r,i)q(n,i)$ is as follows: For i which are "close to 1 or $n-1$", we use Markov's inequality to argue that the corresponding $p(r,i)$'s are suitably small; this allows us to bound the contribution of these indices to the sum, using the fact that each $q(n,i)$ is small. For the remaining i's, we use the fact that the p_i's are identical for all symmetric distributions. This allows us to perform a subtle "reduction" to known anti-concentration results for biased product distributions.

We start with the following simple claim, a consequence of Markov's inequality, that shows that if one of i or $n-i$ is reasonably small, the probability $p(r,i)$ is quite small.

Claim 55. *For all $i \in [n-1]$ we have*

$$p(r,i) \le (4/\eta) \cdot \min\{i, n-i\}/n.$$

Proof. For $i \in [n-1]$, let $\mathcal{E}_i = \{\pi \in \mathbb{S}_n : |w_0 + \sum_{j=1}^{i} w_{\pi(j)} - \sum_{j=i+1}^{n} w_{\pi(j)}| \le r\}$. By definition we have that $p(r,i) = \mathbf{Pr}_{\pi \sim_R \mathbb{S}_n}[\mathcal{E}_i]$.

Let $i \le n/2$. If the event $\mathcal{E}_i$ occurs, we certainly have that $w_0 + \sum_{j=1}^{i} w_{\pi(j)} - \sum_{j=i+1}^{n} w_{\pi(j)} \ge -r$ which yields that

$$\sum_{j=1}^{i} w_{\pi(j)} \ge (1/2)(\sum_{i=1}^{n} w_i - r - w_0).$$

That is,

$$p(r,i) \le \mathop{\mathbf{Pr}}_{\pi \sim_R \mathbb{S}_n}\left[\sum_{j=1}^{i} w_{\pi(j)} \ge (1/2)(\sum_{i=1}^{n} w_i - r - w_0) \right].$$

Consider the random variable $X = \sum_{j=1}^{i} w_{\pi(j)}$ and denote $\alpha \stackrel{\text{def}}{=} (1/2)(\sum_{i=1}^{n} w_i - r - w_0)$. We will bound from above the probability

$$\mathop{\mathbf{Pr}}_{\pi \sim_R \mathbb{S}_n}\left[X \ge \alpha\right].$$

Since π is chosen uniformly from $\mathbb{S}_n$, we have that $\mathbf{E}_{\pi \sim_R \mathbb{S}_n}[w_{\pi(j)}] = (1/n) \cdot \sum_{i=1}^{n} w_i$, hence

$$\mathbf{E}_{\pi \sim_R \mathbb{S}_n}[X] = (i/n) \cdot \sum_{i=1}^{n} w_i.$$

Recalling that $|w_0| \leq (1-\eta) \cdot \sum_{i=1}^{n} w_i$ and noting that $\sum_{i=1}^{n} w_i \geq \sum_{i \in S} w_i \geq kr \geq (2/\eta) \cdot r$, we get

$$\alpha \geq (\eta/4) \cdot \sum_{i=1}^{n} w_i.$$

Therefore, noting that $X > 0$, by Markov's inequality, we obtain that

$$\mathop{\mathbf{P}r}_{\pi \sim_R \mathbb{S}_n} [X \geq \alpha] \leq \frac{\mathbf{E}_{\pi \sim_R \mathbb{S}_n}[X]}{\alpha} \leq (4/\eta) \cdot (i/n)$$

as was to be proven.

If $i \geq n/2$, we proceed analogously. If $\mathcal{E}_i$ occurs, we have $w_0 + \sum_{j=1}^{i} w_{\pi(j)} - \sum_{j=i+1}^{n} w_{\pi(j)} \leq r$ which yields that

$$\sum_{j=i+1}^{n} w_{\pi(j)} \geq (1/2)(\sum_{i=1}^{n} w_i + w_0 - r).$$

We then repeat the exact same Markov type argument for the random variable $\sum_{j=i+1}^{n} w_{\pi(j)}$. This completes the proof of the claim. $\square$

Of course, the above lemma is only useful when either i or $n-i$ is relatively small. Fix $i_0 < n/2$ (to be chosen later). Note that, for all $i \leq n/2$, it holds $q(n,i) \leq \frac{2}{i \cdot \Lambda(n)}$. By Claim 55 we thus get that

$$\sum_{i=1}^{i_0} p(r,i)q(n,i) \leq \sum_{i=1}^{i_0} \frac{2}{i \cdot \Lambda(n)} \cdot \frac{4}{\eta} \cdot \frac{i}{n} \leq \frac{8 i_0}{\eta \cdot n \cdot \Lambda(n)}. \quad (3.10)$$

By symmetry, we get

$$\sum_{i=n-i_0}^{n-1} p(r,i)q(n,i) \leq \frac{8 i_0}{\eta \cdot n \cdot \Lambda(n)}. \quad (3.11)$$

We proceed to bound from above the term $\sum_{i=i_0+1}^{n-i_0-1} p(r,i)q(n,i)$. To this end, we exploit the fact, mentioned earlier, that the $p(r,i)$'s depend only on the affine form and not on the particular symmetric distribution over weight levels. We use a subtle argument to essentially reduce anti-concentration statements about μ to known anti-concentration results.

For $\delta \in (0,1)$ let D_δ be the δ-biased distribution over $\{-1,1\}^n$; that is the product distribution in which each coordinate is 1 with probability δ and -1 with probability $1-\delta$. Denote by $g(\delta, i)$ the probability that D_δ assigns to $\{-1,1\}^n_{=i}$, i.e., $g(\delta,i) = \binom{n}{i}\delta^i(1-\delta)^{n-i}$. Theorem 5 now yields

$$\mathop{\mathbf{P}r}_{x \sim D_\delta} [|L(x)| < r] \leq \frac{1}{\sqrt{k\delta(1-\delta)}}.$$

Using symmetry, we view a random draw $x \sim D_\delta$ as a two-stage procedure, exactly as in μ, the only difference being that in the first stage we pick the i-th weight level of the hypercube, $i \in [0, n]$, with probability $g(\delta, i)$. We can therefore write

$$\mathop{\mathbf{P}r}_{x \sim D_\delta} [|L(x)| < r] = \sum_{i=0}^{n} g(\delta, i) p(r, i)$$

and thus conclude that

$$\sum_{i=i_0+1}^{n-i_0-1} g(\delta, i) p(r, i) \leq \sum_{i=0}^{n} g(\delta, i) p(r, i) \leq \frac{1}{\sqrt{k\delta(1-\delta)}}. \tag{3.12}$$

We now state and prove the following crucial lemma. The idea of the lemma is to bound from above the sum $\sum_{i=i_0+1}^{n-i_0-1} p(r, i) q(n, i)$ by suitably averaging over anti-concentration bounds obtained from the δ-biased product distributions:

Lemma 56. *Let $F : [0, 1] \to \mathbb{R}_+$ be such that $q(n, i) \leq \int_{\delta=0}^{1} F(\delta) g(\delta, i) d\delta$ for all $i \in [i_0 + 1, n - i_0 - 1]$. Then,*

$$\sum_{i=i_0+1}^{n-i_0-1} p(r, i) q(n, i) \leq \frac{1}{\sqrt{k}} \cdot \int_{\delta=0}^{1} \frac{F(\delta)}{\sqrt{\delta(1-\delta)}} d\delta.$$

Proof. We have the following sequence of inequalities

$$\begin{aligned} \sum_{i=i_0+1}^{n-i_0-1} p(r, i) q(n, i) &\leq \sum_{i=i_0+1}^{n-1-i_0} \left(\int_{\delta=0}^{1} F(\delta) g(\delta, i) d\delta \right) \cdot p(r, i) \\ &= \int_{\delta=0}^{1} F(\delta) \left(\sum_{i=i_0+1}^{n-i_0-1} g(\delta, i) p(r, i) \right) d\delta \\ &\leq \frac{1}{\sqrt{k}} \cdot \int_{\delta=0}^{1} \frac{F(\delta)}{\sqrt{\delta(1-\delta)}} d\delta \end{aligned}$$

where the first line follows from the assumption of the lemma, the second uses linearity and the third uses (3.12). □

We thus need to choose appropriately a function F satisfying the lemma statement which can give a non-trivial bound on the desired sum. Fix $\zeta > 0$, and define $F(\delta)$ as

$$F(\delta) \overset{\text{def}}{=} \frac{1024}{\Lambda(n)} \cdot \frac{(n+1)^{1/2+\zeta}}{i_0^{1/2+\zeta}} \left(\frac{1}{\delta^{1/2-\zeta}} + \frac{1}{(1-\delta)^{1/2-\zeta}} \right).$$

The following claim (proved in Section 3.3) says that this choice of $F(\delta)$ satisfies the conditions of Lemma 56:

Claim 57. *For the above choice of $F(\delta)$ and $i_0 \leq i \leq n - i_0$, $q(n, i) \leq \int_{\delta=0}^{1} F(\delta) g(\delta, i) d\delta$.*

Now, applying Lemma 56, for this choice of $F(\delta)$, we get that

$$\begin{aligned}&\sum_{i=i_0+1}^{n-i_0-1} p(r,i)q(n,i)\\ \leq\ & \frac{1}{\sqrt{k}}\cdot\frac{1024}{\Lambda(n)}\cdot\frac{(n+1)^{1/2+\zeta}}{i_0^{1/2+\zeta}}\int_{\delta=0}^{1}\left(\frac{1}{\delta^{1/2-\zeta}}+\frac{1}{(1-\delta)^{1/2-\zeta}}\right)\frac{1}{\sqrt{\delta(1-\delta)}}d\delta.\\ =\ & O\left(\frac{1}{\zeta}\cdot\frac{1}{\sqrt{k}}\cdot\frac{1}{\Lambda(n)}\cdot\frac{(n+1)^{1/2+\zeta}}{i_0^{1/2+\zeta}}\right).\end{aligned}$$

We choose (with foresight) $i_0 = \lceil \frac{n}{k^{1/3}} \rceil$. Then the above expression simplifies to

$$\sum_{i=i_0+1}^{n-i_0-1} p(r,i)q(n,i) = O\left(\frac{1}{\zeta}\cdot\frac{1}{\Lambda(n)}\cdot\frac{1}{k^{1/3-\zeta}}\right)$$

Now plugging $i_0 = \lceil \frac{n}{k^{1/3}} \rceil$ in (3.10) and (3.11), we get

$$\sum_{i\leq i_0 \vee i\geq n-i_0} p(r,i)q(n,i) = O\left(\frac{1}{\eta\Lambda(n)}\cdot\frac{1}{k^{1/3}}\right)$$

Combining these equations, we get the final result, and Theorem 53 is proved. □

Proof of Claim 57

We will need the following basic facts :

Fact 58. *For $x, y \in \mathbb{R}_+$ let $\Gamma : \mathbb{R}_+ \to \mathbb{R}$ be the usual "Gamma" function, so that*

$$\int_{\delta=0}^{1} \delta^x(1-\delta)^y d\delta = \frac{\Gamma(x+1)\cdot\Gamma(y+1)}{\Gamma(x+y+2)}$$

Recall that for $z \in \mathbb{Z}_+$, $\Gamma(z) = (z-1)!$.

Fact 59. *(Stirling's approximation) For $z \in \mathbb{R}_+$, we have $\Gamma(z) = \sqrt{\frac{2\pi}{z}}\cdot\left(\frac{z}{e}\right)^z\cdot\left(1+O\left(\frac{1}{z}\right)\right)$. In particular, there is an absolute constant $c_0 > 0$ such that for $z \geq c_0$*

$$\frac{1}{2}\cdot\sqrt{\frac{2\pi}{z}}\cdot\left(\frac{z}{e}\right)^z \leq \Gamma(z) \leq 2\cdot\sqrt{\frac{2\pi}{z}}\cdot\left(\frac{z}{e}\right)^z.$$

Fact 60. *For $x \in \mathbb{R}$ and $x \geq 2$, we have $\left(1-\frac{1}{x}\right)^x \geq \frac{1}{4}$.*

We can now proceed with the proof of Claim 57. We consider the case when $i_0 \le i \le n/2$. (The proof of the complementary case ($n - i_0 - 1 \ge i > n/2$) is essentially identical.) We have the following chain of inequalities:

$$\begin{aligned}
&\int_{\delta=0}^{1} F(\delta) g(\delta, i) d\delta \\
= \; & \frac{1024}{\Lambda(n)} \cdot \frac{(n+1)^{1/2+\zeta}}{i_0^{1/2+\zeta}} \cdot \binom{n}{i} \cdot \int_{\delta=0}^{1} \delta^{i} (1-\delta)^{n-i} \cdot \left(\frac{1}{\delta^{1/2-\zeta}} + \frac{1}{(1-\delta)^{1/2-\zeta}} \right) d\delta \\
\ge \; & \frac{1024}{\Lambda(n)} \cdot \frac{(n+1)^{1/2+\zeta}}{i_0^{1/2+\zeta}} \cdot \binom{n}{i} \cdot \int_{\delta=0}^{1} \delta^{i-1/2+\zeta} (1-\delta)^{n-i} \, d\delta \\
= \; & \frac{1024}{\Lambda(n)} \cdot \frac{(n+1)^{1/2+\zeta}}{i_0^{1/2+\zeta}} \cdot \binom{n}{i} \cdot \frac{\Gamma(n-i+1) \cdot \Gamma(i+1/2+\zeta)}{\Gamma(n+3/2+\zeta)} \quad \text{(using Fact 58)} \\
= \; & \frac{1024}{\Lambda(n)} \cdot \frac{(n+1)^{1/2+\zeta}}{i_0^{1/2+\zeta}} \cdot \frac{\Gamma(n+1)}{\Gamma(i+1) \cdot \Gamma(n-i+1)} \cdot \frac{\Gamma(n-i+1) \cdot \Gamma(i+1/2+\zeta)}{\Gamma(n+3/2+\zeta)} \\
= \; & \frac{1024}{\Lambda(n)} \cdot \frac{(n+1)^{1/2+\zeta}}{i_0^{1/2+\zeta}} \cdot \frac{\Gamma(n+1) \cdot \Gamma(i+1/2+\zeta)}{\Gamma(i+1) \cdot \Gamma(n+3/2+\zeta)}
\end{aligned}$$

We now proceed to bound from below the right hand side of the last inequality. Towards that, using Fact 59 and assuming n and i are large enough, we have

$$\begin{aligned}
&\frac{\Gamma(n+1) \cdot \Gamma(i+1/2+\zeta)}{\Gamma(i+1) \cdot \Gamma(n+3/2+\zeta)} \\
\ge \; & \frac{1}{16} \cdot \frac{(n+1)^{n+1/2}}{(i+1)^{i+1/2}} \cdot \frac{(i+1/2+\zeta)^{i+\zeta}}{(n+3/2+\zeta)^{n+\zeta+1}} \\
\ge \; & \frac{1}{16} \cdot \frac{1}{n+2} \cdot \frac{(n+1)^{n+1/2}}{(i+1)^{i+1/2}} \cdot \frac{(i+1/2+\zeta)^{i+\zeta}}{(n+3/2+\zeta)^{n+\zeta}} \\
\ge \; & \frac{1}{16} \cdot \frac{1}{n+2} \cdot \frac{(n+1)^{n+\zeta}}{(n+3/2+\zeta)^{n+\zeta}} \cdot \frac{(i+1/2+\zeta)^{i+\zeta}}{(i+1)^{i+\zeta}} \cdot \frac{(n+1)^{1/2-\zeta}}{(i+1)^{1/2-\zeta}} \\
\ge \; & \frac{1}{256} \cdot \frac{1}{n+2} \cdot \frac{(n+1)^{1/2-\zeta}}{(i+1)^{1/2-\zeta}} \ge \frac{1}{512} \cdot \frac{1}{(n+1)^{1/2+\zeta}} \cdot \frac{1}{(i+1)^{1/2-\zeta}}
\end{aligned}$$

Plugging this back, we get

$$\begin{aligned}
\int_{\delta=0}^{1} F(\delta) g(\delta, i) d\delta \; & \ge \; \frac{1024}{\Lambda(n)} \cdot \frac{(n+1)^{1/2+\zeta}}{i_0^{1/2+\zeta}} \cdot \frac{\Gamma(n+1) \cdot \Gamma(i+1/2+\zeta)}{\Gamma(i+1) \cdot \Gamma(n+3/2+\zeta)} \\
& \ge \; \frac{1024}{\Lambda(n)} \cdot \frac{(n+1)^{1/2+\zeta}}{i_0^{1/2+\zeta}} \cdot \frac{1}{512} \cdot \frac{1}{(n+1)^{1/2+\zeta}} \cdot \frac{1}{(i+1)^{1/2-\zeta}} \\
& = \; \frac{2}{\Lambda(n)} \cdot \frac{1}{i_0^{1/2+\zeta}} \cdot \frac{1}{(i+1)^{1/2-\zeta}} \ge \frac{2}{\Lambda(n)} \cdot \frac{1}{i} \ge q(n, i)
\end{aligned}$$

which concludes the proof of the claim. □

3.4 A Useful Algorithmic Tool

In this section we describe a useful algorithmic tool arising from recent work in computational complexity theory. The main result we will need is the following theorem of [138] (the ideas go back to [70] and were used in a different form in [31]):

Theorem 61. *([138]) Let X be a finite domain, μ be a samplable probability distribution over X, $f : X \to [-1,1]$ be a bounded function, and $\mathcal{L}$ be a finite family of Boolean functions $\ell : X \to \{-1,1\}$. There is an algorithm* Boosting-TTV *with the following properties: Suppose* Boosting-TTV *is given as input a list $(a_\ell)_{\ell\in\mathcal{L}}$ of real values and a parameter $\xi > 0$ such that $|\mathbf{E}_{x\sim\mu}[f(x)\ell(x)] - a_\ell| \le \xi/16$ for every $\ell \in \mathcal{L}$. Then* Boosting-TTV *outputs a function $h : X \to [-1,1]$ with the following properties:*

(i) $|\mathbf{E}_{x\sim\mu}[\ell(x)h(x) - \ell(x)f(x)]| \le \xi$ for every $\ell \in \mathcal{L}$;

(ii) $h(x)$ is of the form $h(x) = P_1(\frac{\xi}{2} \cdot \sum_{\ell\in\mathcal{L}} w_\ell \ell(x))$ where the w_ℓ's are integers whose absolute values sum to $O(1/\xi^2)$.

The algorithm runs for $O(1/\xi^2)$ iterations, where in each iteration it estimates $\mathbf{E}_{x\sim\mu}[h'(x)\ell(x)]$ to within additive accuracy $\pm\xi/16$. Here each h' is a function of the form $h'(x) = P_1(\frac{\xi}{2} \cdot \sum_{\ell\in\mathcal{L}} v_\ell \ell(x))$, where the v_ℓ's are integers whose absolute values sum to $O(1/\xi^2)$.

We note that Theorem 61 is not explicitly stated in the above form in [138]; in particular, neither the time complexity of the algorithm nor the fact that it suffices for the algorithm to be given "noisy" estimates a_ℓ of the values $\mathbf{E}_{x\sim\mu}[f(x)\ell(x)]$ is explicitly stated in [138]. So for the sake of completeness, in the following we state the algorithm in full (see Figure 3.4) and sketch a proof of correctness of this algorithm using results that are explicitly proved in [138].

Proof of Theorem 61. It is clear from the description of the algorithm that (if and) when the algorithm Boosting-TTV terminates, the output h satisfies property (i) and has the form $h(x) = P_1(\frac{\xi}{2} \cdot \sum_{\ell\in\mathcal{L}} w_\ell \ell(x))$ where each w_ℓ is an integer. It remains to bound the number of iterations (which gives a bound on the sum of magnitudes of w_ℓ's) and indeed to show that the algorithm terminates at all.

Towards this, we recall Claim 3.4 in [138] states the following:

Claim 62. *For all $x \in supp(\mu)$ and all $t \ge 1$, we have $\sum_{j=1}^{t} f_j(x) \cdot (f(x) - h_{j-1}(x)) \le (4/\gamma) + (\gamma t)/2$.*

We now show how this immediately gives Theorem 61. Fix any $j \ge 0$, and suppose without loss of generality that $a_\ell - a_{\ell,j} > \gamma$. We have that

$$|\mathbf{E}_{x\sim\mu}[f_{j+1}(x)h_j(x)] - a_{\ell,j}| \le \xi/16 \quad \text{and hence} \quad \mathbf{E}_{x\sim\mu}[f_{j+1}(x)h_j(x)] \le a_{\ell,j} + \xi/16,$$

and similarly

$$|\mathbf{E}_{x\sim\mu}[f_{j+1}(x)f(x)] - a_\ell| \le \xi/16 \quad \text{and hence} \quad \mathbf{E}_{x\sim\mu}[f_{j+1}(x)f(x)] \ge a_\ell - \xi/16.$$

Combining these inequalities with $a_\ell - a_{\ell,j} > \gamma = \xi/2$, we conclude that

$$\mathbf{E}_{x\sim\mu}[f_{j+1}(x)(f(x) - h_j(x))] \ge 3\xi/8.$$

Putting this together with Claim 62, we get that

$$\frac{3\xi t}{8} \le \sum_{j=1}^{t} \mathbf{E}_{x\sim\mu}[f_j(x)(f(x) - h_{j-1}(x))] \le \frac{4}{\gamma} + \frac{\gamma t}{2}.$$

Since $\gamma = \xi/2$, this means that if the algorithm runs for t time steps, then $8/\xi \ge (\xi t)/8$, which implies that $t \le 64/\xi^2$. This concludes the proof. □

Boosting-TTV

Parameters:

ξ	:=	positive real number
μ	:=	samplable distribution over finite domain X
$\mathcal{L}$	:=	finite list of functions such that all $\ell \in \mathcal{L}$ maps X to $\{-1, 1\}$.
$(a_\ell)_{\ell\in\mathcal{L}}$	:=	list of real numbers with the promise that some $f : X \to [-1, 1]$ has $|\mathbf{E}_{x\sim\mu}[f(x)\ell(x)] - a_\ell| \le \xi/16$ for all $\ell \in \mathcal{L}$.

Output:
An LBF $h(x) \equiv P_1(\sum_{\ell\in\mathcal{L}} w_\ell \ell(x))$, where $w_\ell \in \mathbb{Z}$, such that $\mathbf{E}_{x\sim\mu}[h(x)\ell(x)] - f(x)\ell(x)| \le \xi$ for all $\ell \in \mathcal{L}$.

Algorithm:

1. Let $\mathcal{L}^0 \overset{\text{def}}{=} \{\ell : \ell \in \mathcal{L} \text{ or } -\ell \in \mathcal{L}\}$. Fix $\gamma \overset{\text{def}}{=} \xi/2$.
2. Let $h_0 \overset{\text{def}}{=} 0$. Set $t = 0$.
3. For each $\ell \in \mathcal{L}$, find $a_{\ell,t} \in \mathbb{R}$ such that $|\mathbf{E}_{x\sim\mu}[h_t(x)\ell(x)] - a_{\ell,t}| \le \xi/16$.
4. If $|a_\ell - a_{\ell,t}| \le \gamma$ for all $\ell \in \mathcal{L}$, then stop and output h_t. Otherwise, fix ℓ to be any element of $\mathcal{L}$ such that $|a_\ell - a_{\ell,t}| > \gamma$.
    - If $a_\ell - a_{\ell,t} > \gamma$ then set $f_{t+1} \overset{\text{def}}{=} \ell$ else set $f_{t+1} \overset{\text{def}}{=} -\gamma$. Note that $f_{t+1} \in \mathcal{L}^0$.
    - Define h_{t+1} as $h_{t+1}(x) \overset{\text{def}}{=} P_1(\gamma(\sum_{j=1}^{t+1} f_j(x)))$.
5. Set $t = t + 1$ and go to Step 3.

3.5 Our Main Results

In this section we combine ingredients from the previous subsections and prove our main results, Theorems 63 and 64.

Our first main result gives an algorithm that works if *any* monotone increasing η-reasonable LTF has approximately the right Shapley values:

Theorem 63. *There is an algorithm* IS *(for* Inverse-Shapley*) with the following properties.* IS *is given as input an accuracy parameter $\epsilon > 0$, a confidence parameter $\delta > 0$, and n real values $a(1), \dots, a(n)$; its output is a pair $v \in \mathbb{R}^n, \theta \in \mathbb{R}$. Its running time is* $\mathrm{poly}(n, 2^{\mathrm{poly}(1/\epsilon)}, \log(1/\delta))$. *The performance guarantees of* IS *are the following:*

1. *Suppose there is a monotone increasing η-reasonable LTF $f(x)$ such that $d_{\mathrm{Shapley}}(a, f) \le 1/\mathrm{poly}(n, 2^{\mathrm{poly}(1/\epsilon)})$. Then with probability $1 - \delta$ algorithm* IS *outputs $v \in \mathbb{R}^n$, $\theta \in \mathbb{R}$ which are such that the LTF $h(x) = \mathrm{sign}(v \cdot x - \theta)$ has $d_{\mathrm{Shapley}}(f, h) \le \epsilon$.*

2. *For any input vector $(a(1), \dots, a(n))$, the probability that* IS *outputs $v \in \mathbb{R}^n, \theta \in \mathbb{R}$ such that the LTF $h(x) = \mathrm{sign}(v \cdot x - \theta)$ has $d_{\mathrm{Shapley}}(f, h) > \epsilon$ is at most δ.*

Proof. We first note that we may assume $\epsilon > n^{-c}$ for a constant $c > 0$ of our choosing, for if $\epsilon \le n^{-c}$ then the claimed running time is $2^{\Omega(n^2 \log n)}$. In this much time we can easily enumerate all LTFs over n variables (by trying all weight vectors with integer weights at most n^n; this suffices by [116]) and compute their Shapley values exactly, and thus solve the problem. So for the rest of the proof we assume that $\epsilon > n^{-c}$.

It will be obvious from the description of IS that property (2) above is satisfied, so the main job is to establish (1). Before giving the formal proof we first describe an algorithm and analysis achieving (1) for an idealized version of the problem. We then describe the actual algorithm and its analysis (which build on the idealized version).

Recall that the algorithm is given as input ϵ, δ and $a(1), \dots, a(n)$ that satisfy $d_{\mathrm{Shapley}}(a, f) \le 1/\mathrm{poly}(n, 2^{\mathrm{poly}(1/\epsilon)})$ for some monotone increasing η-reasonable LTF f. The idealized version of the problem is the following: we assume that the algorithm is also given the two real values $f^*(0)$, $\sum_{i=1}^n f^*(i)/n$. It is also helpful to note that since f is monotone and η-reasonable (and hence is not a constant function), it must be the case that $f(\mathbf{1}) = 1$ and $f(-\mathbf{1}) = -1$.

The algorithm for this idealized version is as follows: first, using Lemma 45, the values $\tilde{f}(i)$, $i = 1, \dots, n$ are converted into values $a^*(i)$ which are approximations for the values $f^*(i)$. Each $a^*(i)$ satisfies $|a^*(i) - f^*(i)| \le 1/\mathrm{poly}(n, 2^{O(\mathrm{poly}(1/\epsilon))})$. The algorithm sets $a^*(0)$ to $f^*(0)$. Next, the algorithm runs Boosting-TTV with the following input: the family $\mathcal{L}$ of Boolean functions is $\{1, x_1, \dots, x_n\}$; the values $a^*(0), \dots, a^*(n)$ comprise the list of real values; μ is the distribution; and the parameter ξ is set to $1/\mathrm{poly}(n, 2^{\mathrm{poly}(1/\epsilon)})$. (We note that each execution of Step 3 of Boosting-TTV, namely finding values that closely estimate $\mathbf{E}_{x \sim \mu}[h_t(x) x_i]$ as required, is easily achieved using a standard sampling scheme; for completeness in Appendix B.2 we describe a procedure Estimate-Correlation that can be used to do all the required estimations with overall failure probability at most δ.) Boosting-TTV outputs an LBF $h(x) = P_1(v \cdot x - \theta)$; the output of our overall algorithm is the LTF $h'(x) = \mathrm{sign}(v \cdot x - \theta)$.

Let us analyze this algorithm for the idealized scenario. By Theorem 61, the output function h that is produced by Boosting-TTV is an LBF $h(x) = P_1(v \cdot x - \theta)$ that satisfies the equation $\sqrt{\sum_{j=0}^{n}(h^*(j) - f^*(j))^2} = 1/\mathrm{poly}(n, 2^{\mathrm{poly}(1/\epsilon)})$. Given this, Lemma 50 implies that $d_{\mathrm{Fourier}}(f, h) \leq \rho \stackrel{\mathrm{def}}{=} 1/\mathrm{poly}(n, 2^{\mathrm{poly}(1/\epsilon)})$.

At this point, we have established that h is a bounded function that satisfies the constraint $d_{\mathrm{Fourier}}(f, h) \leq 1/\mathrm{poly}(n, 2^{\mathrm{poly}(1/\epsilon)})$. We would like to apply Lemma 52 and thereby assert that the ℓ_1 distance between f and h (with respect to μ) is small. To see that we can do this, we first note that since f is a monotone increasing η-reasonable LTF, by Theorem 40 it has a representation as $f(x) = \mathrm{sign}(w \cdot x + w_0)$ whose weights satisfy the properties claimed in that theorem; in particular, for any choice of $\zeta > 0$, after rescaling all the weights, the largest-magnitude weight has magnitude 1, and the $k \stackrel{\mathrm{def}}{=} \Theta_{\zeta,\eta}(1/\epsilon^{6+2\zeta})$ largest-magnitude weights each have magnitude at least $r \stackrel{\mathrm{def}}{=} 1/(n \cdot k^{O(k)})$. (Note that since $\epsilon \geq n^{-c}$ we indeed have $k \leq n$ as required.) Given this, Theorem 53 implies that the affine form $L(x) = w \cdot x + w_0$ satisfies

$$\mathop{\mathbf{Pr}}_{x \sim \mu}[|L(x)| < r] \leq \kappa \stackrel{\mathrm{def}}{=} \epsilon^2/(512 \log(n)), \tag{3.13}$$

i.e., it is (r, κ)-anticoncentrated with $\kappa = \epsilon^2/(512\log(n))$. Thus we may indeed apply Lemma 52, and it gives us that

$$\mathbf{E}_{x\sim\mu}[|f(x) - h(x)|] \leq \frac{4\|w\|_1\sqrt{\rho}}{r} + 2\kappa \leq \epsilon^2/(128 \log n). \tag{3.14}$$

Now let $h' : \{-1, 1\}^n \to \{-1, 1\}$ be the LTF defined as $h'(x) = \mathrm{sign}(v \cdot x - \theta)$ (recall that h is the LBF $P_1(v \cdot x - \theta)$). Since f is a $\{-1, 1\}$-valued function, it is clear that for every input x in the support of μ, the contribution of x to $\mathbf{Pr}_{x\sim\mu}[f(x) \neq h'(x)]$ is at most twice its contribution to $\mathbf{E}_{x\sim\mu}[|f(x) - h(x)|]$. Thus we have that $\mathbf{Pr}_{x\sim\mu}[f(x) \neq h'(x)] \leq \epsilon^2/(64 \log n)$. We may now apply Fact 44 to obtain that $d_{\mathrm{Fourier}}(f, h') \leq \epsilon/(4\sqrt{\log n})$. Finally, Lemma 49 gives that

$$d_{\mathrm{Shapley}}(f, h') \leq 4/\sqrt{n} + \sqrt{\Lambda(n)} \cdot \epsilon/(4\sqrt{\log n}) < \epsilon/2.$$

So indeed the LTF $h'(x) = \mathrm{sign}(v \cdot x - \theta)$ satisfies $d_{\mathrm{Shapley}}(f, h') \leq \epsilon/2$ as desired.

Now we turn from the idealized scenario to actually prove Theorem 63, where we are not given the values of $f^*(0)$ and $\sum_{i=1}^{n} f^*(i)/n$. To get around this, we note that $f^*(0), \sum_{i=1}^{n} f^*(i)/n \in [-1, 1]$. So the idea is that we will run the idealized algorithm repeatedly, trying "all" possibilities (up to some prescribed granularity) for $f^*(0)$ and for $\sum_{i=1}^{n} f^*(i)/n$. At the end of each such run we have a "candidate" LTF h'; we use a simple procedure Shapley-Estimate (see Appendix B.2) to estimate $d_{\mathrm{Shapley}}(f, h')$ to within additive accuracy $\pm\epsilon/10$, and we output any h' whose estimated value of $d_{\mathrm{Shapley}}(f, h')$ is at most $8\epsilon/10$.

We may run the idealized algorithm $\mathrm{poly}(n, 2^{\mathrm{poly}(1/\epsilon)})$ times without changing its overall running time (up to polynomial factors). Thus we can try a net of possible guesses for $f^*(0)$ and

$\sum_{i=1}^{n} f^*(i)/n$ which is such that one guess will be within $\pm 1/\text{poly}(n, 2^{\text{poly}(1/\epsilon)})$ of the the correct values for both parameters. It is straightforward to verify that the analysis of the idealized scenario given above is sufficiently robust that when these "good" guesses are encountered, the algorithm will with high probability generate an LTF h' that has $d_{\text{Shapley}}(f, h') \leq 6\epsilon/10$. A straightforward analysis of running time and failure probability shows that properties (1) and (2) are achieved as desired, and Theorem 63 is proved. □

For any monotone η-reasonable target LTF f, Theorem 63 constructs an output LTF whose Shapley distance from f is at most ϵ, but the running time is exponential in $\text{poly}(1/\epsilon)$. We now show that if the target monotone η-reasonable LTF f has integer weights that are at most W, then we can construct an output LTF h with $d_{\text{Shapley}}(f, h) \leq n^{-1/8}$ running in time $\text{poly}(n, W)$; this is a far faster running time than provided by Theorem 63 for such small ϵ. (The "1/8" is chosen for convenience; it will be clear from the proof that any constant strictly less than 1/6 would suffice.)

Theorem 64. *There is an algorithm* ISBW *(for* Inverse-Shapley *with* Bounded Weights*) with the following properties.* ISBW *is given as input a weight bound* $W \in \mathbb{Z}_+$*, a confidence parameter* $\delta > 0$*, and* n *real values* $a(1), \ldots, a(n)$*; its output is a pair* $v \in \mathbb{R}^n, \theta \in \mathbb{R}$*. Its running time is* $\text{poly}(n, W, \log(1/\delta))$*. The performance guarantees of* ISBW *are the following:*

1. *Suppose there is a monotone increasing* η*-reasonable LTF* $f(x) = \text{sign}(u \cdot x - \theta)$*, where each* u_i *is an integer with* $|u_i| \leq W$*, such that* $d_{\text{Shapley}}(a, f) \leq 1/\text{poly}(n, W)$*. Then with probability* $1 - \delta$ *algorithm* ISBW *outputs* $v \in \mathbb{R}^n$*,* $\theta \in \mathbb{R}$ *which are such that the LTF* $h(x) = \text{sign}(v \cdot x - \theta)$ *has* $d_{\text{Shapley}}(f, h) \leq n^{-1/8}$*.*

2. *For any input vector* $(a(1), \ldots, a(n))$*, the probability that* IS *outputs* v, θ *such that the LTF* $h(x) = \text{sign}(v \cdot x - \theta)$ *has* $d_{\text{Shapley}}(f, h) > n^{-1/8}$ *is at most* δ*.*

Proof. Let $f(x) = \text{sign}(u \cdot x - \theta)$ be as described in the theorem statement. We may assume that each $|u_i| \geq 1$ (by scaling all the u_i's and θ by $2n$ and then replacing any zero-weight u_i with 1). Next we observe that for such an affine form $u \cdot x - \theta$, Theorem 53 immediately yields the following corollary:

Corollary 65. *Let* $L(x) = \sum_{i=1}^{n} u_i x_i - \theta$ *be a monotone increasing* η*-reasonable affine form. Suppose that* $u_i \geq r$ *for all* $i = 1, \ldots, n$*. Then for any* $\zeta > 0$*, we have*

$$\mathop{\mathbf{Pr}}_{x\sim\mu} [|L(x)| < r] = O\left(\frac{1}{\log n} \cdot \frac{1}{n^{1/3-\zeta}} \cdot \left(\frac{1}{\zeta} + \frac{1}{\eta}\right)\right).$$

With this anti-concentration statement in hand, the proof of Theorem 64 closely follows the proof of Theorem 63. The algorithm runs Boosting-TTV with $\mathcal{L}$, $a^*(i)$ and μ as before but now with ξ set to $1/\text{poly}(n, W)$. The LBF h that Boosting-TTV outputs satisfies $d_{\text{Fourier}}(f, h) \leq \rho \overset{\text{def}}{=} 1/\text{poly}(n, W)$. We apply Corollary 65 to the affine form $L(x) \overset{\text{def}}{=} \frac{u}{\|u\|_1} \cdot x - \frac{\theta}{\|u\|_1}$ and get that for $r = 1/\text{poly}(n, W)$, we have

$$\mathop{\mathbf{Pr}}_{x\sim\mu}[|L(x)| < r] \leq \kappa \overset{\text{def}}{=} \epsilon^2/(1024 \log n) \tag{3.15}$$

where now $\epsilon \stackrel{\text{def}}{=} n^{-1/8}$, in place of Equation (3.13). Applying Lemma 52 we get that

$$\mathbf{E}_{x\sim\mu}[|f(x) - h(x)|] \leq \frac{4\|w\|_1\sqrt{\rho}}{r} + 4\kappa \leq \epsilon^2/(128\log n)$$

analogous to (3.14). The rest of the analysis goes through exactly as before, and we get that the LTF $h'(x) = \mathrm{sign}(v \cdot x - \theta)$ satisfies $d_{\mathrm{Shapley}}(f, h') \leq \epsilon/2$ as desired. The rest of the argument is unchanged so we do not repeat it. □

Chapter 4

Inverse approximate uniform generation

The generation of (approximately) uniform random combinatorial objects has been an important research topic in theoretical computer science for several decades. In complexity theory, well-known results have related approximate uniform generation to other fundamental topics such as approximate counting and the power of nondeterminism [74, 72, 132, 133, 134]. On the algorithms side, celebrated algorithms have been given for a wide range of approximate uniform generation problems such as perfect matchings [75], graph colorings (see e.g. [73, 147, 66]), satisfying assignments of DNF formulas [82, 72, 81], of linear threshold functions (i.e., knapsack instances) [112, 43] and more.

Before describing the inverse problems that we consider, let us briefly recall the usual framework of approximate uniform generation. An approximate uniform generation problem is defined by a class $\mathcal{C}$ of combinatorial objects and a polynomial-time relation $R(x, y)$ over $\mathcal{C} \times \{0,1\}^*$. An input instance of the problem is an object $x \in \mathcal{C}$, and the problem, roughly speaking, is to output an approximately uniformly random element y from the set $R_x := \{y : R(x, y) \text{ holds}\}$. Thus an algorithm A (which must be randomized) for the problem must have the property that for all $x \in \mathcal{C}$, the output distribution of $A(x)$ puts approximately equal weight on every element of R_x. For example, taking the class of combinatorial objects to be $\{\text{all } n \times n \text{ bipartite graphs}\}$ and the polynomial-time relation R over (G, M) pairs to be "M is a perfect matching in G," the resulting approximate uniform generation problem is to generate an (approximately) uniform perfect matching in a given bipartite graph; a $\mathrm{poly}(n, \log(1/\epsilon))$-time algorithm was given in [75]. As another example, taking the combinatorial object to be a linear threshold function (LTF) $f(x) = \mathrm{sign}(w \cdot x - \theta)$ mapping $\{-1,1\}^n \to \{-1,1\}$ (represented as a vector $(w_1, \dots, w_n, \theta)$) and the polynomial-time relation R over (f, x) to be "x is a satisfying assignment for f," we arrive at the problem of generating approximately uniform satisfying assignments for an LTF (equivalently, feasible solutions to zero-one knapsack). A polynomial-time algorithm was given by [112] and a faster algorithm was subsequently proposed by [43].

The focus of this chapter is on *inverse* problems in approximate uniform generation. In such problems, instead of having to *output* (near-)uniform elements of R_x, the *input* is a sample of elements drawn uniformly from R_x, and the problem (roughly speaking) is to "reverse engineer" the sample and output a distribution which is close to the uniform distribution over R_x. More precisely, following the above framework, a problem of this sort is again defined by a class $\mathcal{C}$ of combinatorial objects and a polynomial-time relation R. However, now an input instance of the problem is a sample $\{y_1, \ldots, y_m\}$ of strings drawn uniformly at random from the set $R_x := \{y : R(x, y) \text{ holds}\}$, where now $x \in \mathcal{C}$ is *unknown*. The goal is to output an *ϵ-sampler* for R_x, i.e., a randomized algorithm (which takes no input) whose output distribution is ϵ-close in total variation distance to the uniform distribution over R_x. Revisiting the first example from the previous paragraph, for the inverse problem the input would be a sample of uniformly random perfect matchings of an unknown bipartite graph G, and the problem is to output a sampler for the uniform distribution over all perfect matchings of G. For the inverse problem corresponding to the second example, the input is a sample of uniform random satisfying assignments of an unknown LTF over the Boolean hypercube, and the desired output is a sampler that generates approximately uniform random satisfying assignments of the LTF.

Discussion. Before proceeding we briefly consider some possible alternate definitions of inverse approximate uniform generation, and argue that our definition is the "right" one (we give a precise statement of our definition in Section 4.1, see Definition 76).

One stronger possible notion of inverse approximate uniform generation would be that the output distribution should be supported on R_x and put nearly the same weight on every element of R_x, instead of just being ϵ-close to uniform over R_x. However a moment's thought suggests that this notion is too strong, since it is impossible to efficiently achieve this strong guarantee even in simple settings. (Consider, for example, the problem of inverse approximate uniform generation of satisfying assignments for an unknown LTF. Given access to uniform satisfying assignments of an LTF f, it is impossible to efficiently determine whether f is (say) the majority function or an LTF that differs from majority on precisely one point in $\{-1, 1\}^n$, and thus it is impossible to meet this strong guarantee.)

Another possible definition of inverse approximate uniform generation would be to require that the algorithm output an ϵ-approximation of the unknown object x instead of an ϵ-sampler for R_x. Such a proposed definition, though, leads immediately to the question of how one should measure the distance between a candidate object x' and the true "target" object x. The most obvious choice would seem to be the total variation distance between $\mathcal{U}_{R_x}$ (the uniform distribution over R_x) and $\mathcal{U}_{R_{x'}}$; but given this distance measure, it seems most natural to require that the algorithm actually output an ϵ-approximate sampler for R_x.

Inverse approximate uniform generation via reconstruction and sampling. While our ultimate goal, as described above, is to obtain algorithms that output a sampler, algorithms that attempt to reconstruct the unknown object x will also play an important role for us. Given $\mathcal{C}, R$ as above, we say that an (ϵ, δ)-*reconstruction algorithm* is an algorithm $A_{\text{reconstruct}}$ that works as follows: for any $x \in \mathcal{C}$, if $A_{\text{reconstruct}}$ is given as input a sample of $m = m(\epsilon, \delta)$ i.i.d. draws from the uniform distribution over R_x, then with probability $1 - \delta$ the output of $A_{\text{reconstruct}}$ is an object $\tilde{x} \in \tilde{C}$ such that the variation distance $d_{\text{TV}}(\mathcal{U}_{R_x}, \mathcal{U}_{R_{\tilde{x}}})$ is at most ϵ. (Note that the class $\tilde{\mathcal{C}}$ need not coincide with the original class $\tilde{\mathcal{C}}$, so $\tilde{x}$ need not necessarily belong to $\mathcal{C}$.) With this notion in hand, an intuitively appealing schema for algorithms that solve inverse approximate uniform generation problems is to proceed in the following two stages:

1. **(Reconstruct the unknown object):** Run a reconstruction algorithm $A_{\mathrm{reconstruct}}$ with accuracy and confidence parameters $\epsilon/2, \delta/2$ to obtain $\tilde{x} \in \tilde{\mathcal{C}}$;

2. **(Sample from the reconstructed object):** Let A_{sample} be an algorithm which solves the approximate uniform generation problem $(\tilde{\mathcal{C}}, R)$ to accuracy $\epsilon/2$ with confidence $1 - \delta/2$. The desired sampler is the algorithm A_{sample} with its input set to $\tilde{x}$.

We refer to this as the *standard approach* for solving inverse approximate uniform generation problems. Most of our positive results for inverse approximate uniform generation can be viewed as following this approach, but we will see an interesting exception in Section 4.6, where we give an efficient algorithm for an inverse approximate uniform generation problem which does not follow the standard approach.

Relation between inverse approximate uniform generation and other problems.

Most of our results will deal with uniform generation problems in which the class $\mathcal{C}$ of combinatorial objects is a class of syntactically defined Boolean functions over $\{-1, 1\}^n$ (such as the class of all LTFs, all $\mathrm{poly}(n)$-term DNF formulas, all 3-CNFs, etc.) and the polynomial-time relation $R(f, y)$ for $f \in \mathcal{C}$ is "y is a satisfying assignment for f." In such cases our inverse approximate uniform generation problem can be naturally recast in the language of learning theory as an unsupervised learning problem (learning a probability distribution from a known class of possible target distributions): we are given access to samples from $\mathcal{U}_{f^{-1}(1)}$, the uniform distribution over satisfying assignments of $f \in \mathcal{C}$, and the task of the learner is to construct a hypothesis distribution D such that $d_{\mathrm{TV}}(\mathcal{U}_{f^{-1}(1)}, D) \leq \epsilon$ with high probability. We are not aware of prior work in unsupervised learning that focuses specifically on distribution learning problems of this sort (where the target distribution is uniform over the set of satisfying assignments of an unknown member of a known class of Boolean functions).

Our framework also has some similarities to "uniform-distribution learning from positive examples only," since in both settings the input to the algorithm is a sample of points drawn uniformly at random from $f^{-1}(1)$, but there are several differences as well. One difference is that in uniform-distribution learning from positive examples the goal is to output a hypothesis function h, whereas here our goal is to output a hypothesis *distribution* (note that outputting a function h essentially corresponds to the reconstruction problem described above). A more significant difference is that the success criterion for our framework is significantly more demanding than for uniform-distribution learning. In uniform-distribution learning of a Boolean function f over the hypercube $\{-1, 1\}^n$, the hypothesis h must satisfy $\mathbf{P}r[h(x) \neq f(x)] \leq \epsilon$, where the probability is uniform over all 2^n points in $\{-1, 1\}^n$. Thus, for a given setting of the error parameter ϵ, in uniform-distribution learning the constant -1 function is an acceptable hypothesis for any function f that has $|f^{-1}(1)| \leq \epsilon 2^n$. In contrast, in our inverse approximate uniform generation framework we measure error by the total variation distance between $\mathcal{U}_{f^{-1}(1)}$ and the hypothesis distribution D, so no such "easy way out" is possible when $|f^{-1}(1)|$ is small; indeed the hardest instances of inverse approximate uniform generation problems are often those for which $f^{-1}(1)$ is a very small fraction of $\{-1,1\}^n$. Essentially we require a hypothesis with small *multiplicative* error relative to $|f^{-1}(1)|/2^n$ rather than the additive-error criterion that is standard in uniform-distribution learning. We are not aware of prior work on learning Boolean functions in which such a "multiplicative-error" criterion has been employed.

We summarize the above discussion with the following observation, which essentially says that reconstruction algorithms directly yield uniform-distribution learning algorithms:

Observation 66. *Let $\mathcal{C}$ be a class of Boolean functions $\{-1,1\}^n \to \{-1,1\}$ and let $R(f,y)$ be the relation "y is a satisfying assignment for f." Suppose there exists a $t(n,\epsilon,\delta)$-time (ϵ,δ)-reconstruction algorithm for $\mathcal{C}$ that outputs elements of $\tilde{\mathcal{C}}$. Then there is an $(O(\log(1/\delta)/\epsilon^2) + O(t(n,\epsilon,\delta/3)\cdot\log(1/\delta)/\epsilon))$-time uniform-distribution learning algorithm that outputs hypotheses in $\tilde{\mathcal{C}}$ (i.e., given access to uniform random labeled examples $(x,f(x))$ for any $f\in\mathcal{C}$, the algorithm with probability $1-\delta$ outputs a hypothesis $h\in\tilde{C}$ such that $\mathbf{P}r[h(x)\neq f(x)]\leq\epsilon$).*

Proof. The learning algorithm draws an initial set of $O(\log(1/\delta)/\epsilon^2)$ uniform labeled examples to estimate $|f^{-1}(1)|/2^n$ to within an additive $\pm(\epsilon/4)$ with confidence $1-\delta/3$. If the estimate is less than $3\epsilon/4$ the algorithm outputs the constant -1 hypothesis. Otherwise, by drawing $O(t(n,\epsilon,\delta/3)\cdot\log(1/\delta)/\epsilon))$ uniform labeled examples, with failure probability at most $\delta/3$ it can obtain $t(n,\epsilon,\delta/3)$ positive examples (i.e., points that are uniformly distributed over $f^{-1}(1)$). Finally the learning algorithm can use these points to run the reconstruction algorithm with parameters $\epsilon,\delta/3$ to obtain a hypothesis $h\in\tilde{\mathcal{C}}$ that has $d_{\mathrm{TV}}(\mathcal{U}_{f^{-1}(1)},\mathcal{U}_{h^{-1}(1)})\leq\epsilon$ with failure probability at most $\delta/3$. Such a hypothesis h is easily seen to satisfy $\mathbf{P}r[h(x)\neq f(x)]\leq\epsilon$. □

As described in the following subsection, in this chapter we prove negative results for the inverse approximate uniform generation problem for classes such as 3CNF-formulas, monotone 2-CNF formulas, and degree-2 polynomial threshold functions. Since efficient uniform-distribution learning algorithms are known for these classes, these results show that the inverse approximate uniform generation problem is indeed harder than standard uniform-distribution learning for some natural and interesting classes of functions.

The problem of inverse approximate uniform generation is also somewhat reminiscent of the problem of reconstructing Markov Random Fields (MRFs) from random samples [19, 28, 113]. Much progress has been made on this problem over the past decade, especially when the hidden graph is a tree. However, there does not seem to be a concrete connection between this problem and the problems we study. One reason for this seems to be that in MRF reconstruction, the task is to reconstruct the *model* and not just the distribution; because of this, various conditions need to be imposed in order to guarantee the uniqueness of the underlying model given random samples from the distribution. In contrast, in our setting the explicit goal is to construct a high-accuracy distribution, and it may indeed be the case that there is no unique underlying model (i.e., Boolean function f) given the samples received from the distribution.

Our results.

We give a wide range of both positive and negative results for inverse approximate uniform generation problems. As noted above, most of our results deal with uniform generation of satisfying assignments, i.e., $\mathcal{C}$ is a class of Boolean functions over $\{-1,1\}^n$ and for $f \in \mathcal{C}$ the relation $R(f,y)$ is "y is a satisfying assignment for f." All the results, both positive and negative, that we present below are for problems of this sort unless indicated otherwise.

Positive results: A general approach and its applications. We begin by presenting a general approach for obtaining inverse approximate uniform generation algorithms. This technique combines approximate uniform generation and counting algorithms and Statistical Query (SQ) learning algorithms with a new type of algorithm called a "densifier," which we introduce and define in Section 4.2. Very roughly speaking, the densifier lets us prune the entire space $\{-1,1\}^n$ to a set S which (essentially) contains all of $f^{-1}(1)$ and is not too much larger than $f^{-1}(1)$ (so $f^{-1}(1)$ is "dense" in S). By generating approximately uniform elements of S it is possible to run an SQ learning algorithm and obtain a high-accuracy hypothesis which can be used, in conjunction with the approximate uniform generator, to obtain a sampler for a distribution which is close to the uniform distribution over $f^{-1}(1)$. (The approximate counting algorithm is needed for technical reasons which we explain in Section 4.2.) In Section 4.2 we describe this technique in detail and prove a general result establishing its effectiveness.

In Sections 4.3 and 4.4 we give two main applications of this general technique to specific classes of functions. The first of these is the class **LTF** of all LTFs over $\{-1,1\}^n$. Our main technical contribution here is to construct a densifier for LTFs; we do this by carefully combining known efficient online learning algorithms for LTFs (based on interior-point methods for linear programming) [108] with known algorithms for approximate uniform generation and counting of satisfying assignments of LTFs [112, 43]. Given this densifier, our general approach yields the desired inverse approximate uniform generator for LTFs:

Theorem 67. (Informal statement) *There is a* $\mathrm{poly}(n, 1/\epsilon)$*-time algorithm for the inverse problem of approximately uniformly generating satisfying assignments for LTFs.*

Our second main positive result for a specific class, in Section 4.4, is for the well-studied class $\mathbf{DNF}_{n,s}$ of all size-s DNF formulas over n Boolean variables. Here our main technical contribution is to give a densifier which runs in time $n^{O(\log(s/\epsilon))}$ and outputs a DNF formula. A challenge here is that known SQ algorithms for learning DNF formulas require time exponential in $n^{1/3}$. To get around this, we view the densifier's output DNF as an OR over $n^{O(\log(s/\epsilon))}$ "metavariables" (corresponding to all possible conjunctions that could be present in the DNF output by the densifier), and we show that it is possible to apply known *malicious noise tolerant* SQ algorithms for learning *sparse disjunctions* as the SQ-learning component of our general approach. Since efficient approximate uniform generation and approximate counting algorithms are known [72, 82] for DNF formulas, with the above densifier and SQ learner we can carry out our general technique, and we thereby obtain our second main positive result for a specific function class:

Theorem 68. (Informal statement) *There is a $n^{O(\log(s/\epsilon))}$-time algorithm for the inverse problem of approximately uniformly generating satisfying assignments for s-term DNF formulas.*

Negative results based on cryptography. In light of the "standard approach," it is clear that in order for an inverse approximate uniform generation problem $(\mathcal{C}, R)$ to be computationally hard, it must be the case that either stage (1) (reconstructing the unknown object) or stage (2) (sampling from the reconstructed object) is hard. (If both stages have efficient algorithms $A_{\text{reconstruct}}$ and A_{sample} respectively, then there is an efficient algorithm for the whole inverse approximate uniform generation problem that combines these algorithms according to the standard approach.) Our first approach to obtaining negative results can be used to obtain hardness results for problems for which stage (2), near-uniform sampling, is computationally hard. The approach is based on signature schemes from public-key cryptography; roughly speaking, the general result which we prove is the following (we note that the statement given below is a simplification of our actual result which omits several technical conditions; see Theorem 118 of Section 4.5 for a precise statement):

Theorem 69. (Informal statement) *Let $\mathcal{C}$ be a class of functions such that there is a parsimonious reduction from* CIRCUIT-SAT *to $\mathcal{C}$-*SAT*. Then known constructions of secure signature schemes imply that there is no subexponential-time algorithm for the inverse problem of approximately uniformly generating satisfying assignments to functions in $\mathcal{C}$.*

The secure signature schemes we require are known to exist under many different assumptions. These include the RSA$'$ assumption of Micali, Rabin and Vadhan [111], the DH-DDH assumption of Lysyanskaya [107], and the strong RSA assumption of Cramer and Shoup [25]. The same assumptions can also be used in Corollary 70.

This theorem yields a wide range of hardness results for specific classes that show that our positive results (for LTFs and DNF) lie quite close to the boundary of what classes have efficient inverse approximate uniform generation algorithms. We prove:

Corollary 70. (Informal statement) *Under known constructions of secure signature schemes, there is no subexponential-time algorithm for the inverse approximate uniform generation problem for either of the following classes of functions: (i) 3-CNF formulas; (ii) intersections of two halfspaces.*

We show that our signature-scheme-based hardness approach can be extended to settings where there is no parsimonious reduction as described above. Using "blow-up"-type constructions of the sort used to prove hardness of approximate counting, we prove the following:

Theorem 71. (Informal statement) *Under the same assumptions as Corollary 70, there is no subexponential-time algorithm for the inverse approximate uniform generation problem for either of the following classes: (i) monotone 2-CNF; (ii) degree-2 polynomial threshold functions.*

It is instructive to compare the above hardness results with the problem of uniform generation of NP-witnesses. In particular, while it is obvious that no efficient randomized algorithm can produce even a single satisfying assignment of a given 3-SAT instance (assuming $\mathbf{NP} \not\subseteq \mathbf{BPP}$), the seminal results of Jerrum *et al.* [72] showed that given access to an NP-oracle, it is possible to generate approximately uniform satisfying assignments for a given 3-SAT instance. It is interesting to ask whether one requires the full power of adaptive access to NP-oracles for this task, or whether a weaker form of "advice" suffices. Our hardness results can be understood in this context as giving evidence that receiving polynomially many random satisfying assignments of a 3-SAT instance does not help in further uniform generation of satisfying assignments.[1]

Our signature-scheme based approach cannot give hardness results for problems that have polynomial-time algorithms for the "forward" problem of sampling approximately uniform satisfying assignments. Our second approach to proving computational hardness can (at least sometimes) surmount this barrier. The approach is based on Message Authentication Codes in cryptography; the following is an informal statement of our general result along these lines (as before the following statement ignores some technical conditions; see Theorem 138 for a precise statement):

Theorem 72. (Informal statement) *There are known constructions of MACs with the following property: Let $\mathcal{C}$ be a class of circuits such that the verification algorithm of the MAC can be implemented in $\mathcal{C}$. Then there is no subexponential-time inverse approximate uniform generation algorithm for $\mathcal{C}$.*

We instantiate this general result with a specific construction of a MAC that is a slight variant of a construction due to Pietrzak [124]. The computational assumption in this construction is a slight variant of the standard Learning Parity with Noise (LPN) assumption. This specific construction yields a class $\mathcal{C}$ for which the "forward" approximate uniform generation problem is computationally easy, but (under a plausible computational hardness assumption) the inverse approximate uniform generation problem is computationally hard.

The above construction based on MACs shows that there are problems $(\mathcal{C}, R)$ for which the inverse approximate uniform generation problem is computationally hard although the "forward" approximate uniform generation problem is easy. As our last result, we exhibit a group-theoretic problem (based on graph automorphisms) for which the reverse situation holds: under a plausible hardness assumption the *forward* approximate uniform generation problem is computationally hard, but we give an efficient algorithm for the *inverse* approximate uniform generation problem (which does not follow our general technique or the "standard approach").

Related work. Concurrent (but independent) to our work, Anderson, Goyal and Rademacher [3] considered the following problem : Given access to random samples drawn uniformly from an unknown simplex $\mathcal{X}$ (i.e. an intersection of $n+1$ halfspaces) over $\mathbb{R}^n$, estimate the simplex. More precisely, their task is to output $n+1$ halfspaces $H_1, \ldots, H_n$ such that if $\mathcal{X}' = H_1 \cap \ldots \cap H_n$, then $d_{TV}(\mathcal{U}_X, \mathcal{U}_{X'}) \leq \epsilon$. Anderson *et al.* give a $\mathrm{poly}(n)$-time algorithm for this problem. Combining this with an efficient algorithm for sampling from convex bodies [105], we get a $\mathrm{poly}(n)$-time algorithm to output a sampler for the distribution $\mathcal{U}_{X'}$.

[1]There is a small caveat here in that we are not given the 3-SAT formula *per se* but rather access to random satisfying assignments of the formula. However, there is a simple elimination based algorithm to reconstruct a high-accuracy approximation for a 3-SAT formula if we have access to random satisfying assignments for the formula.

Note that this is the same as inverse approximate uniform generation for the class of intersection of $n+1$ halfspaces, but with one crucial difference: the underlying measure is the Lebesgue measure on $\mathbb{R}^n$ as opposed to the uniform measure on $\{0,1\}^n$ (as it is in our case). The distinction between these two measures is indeed a very significant one; recall that, as we mentioned in Corollary 70, an analogous result is impossible (under a cryptographic hardness assumption) even for an intersection of *two* halfspaces for the uniform measure on $\{0,1\}^n$. Perhaps not too surprisingly, the techniques of Anderson *et al.* are rather disjoint from the techniques in this chapter; in particular, they rely on ideas from algorithmic convex geometry, especially ideas related to Independent Component Analysis (ICA) [56].

Structure of this chapter. After the preliminaries in Section 4.1, we present in Section 4.2 our general upper bound technique. In Sections 4.3 and 4.4 we apply this technique to obtain efficient inverse approximate uniform generation algorithms for LTFs and DNFs respectively. Section 4.5 contains our hardness results. In Section 4.6 we give an example of a problem for which approximate uniform generation is hard, while the inverse problem is easy. Finally, in Section **??** we conclude the chapter suggesting further directions for future work.

4.1 Preliminaries and Useful Tools

Notation and Definitions.

For $n \in \mathbb{Z}_+$, we will denote by $[n]$ the set $\{1,\ldots,n\}$. For a distribution D over a finite set $\mathcal{W}$ we denote by $D(x)$, $x \in \mathcal{W}$, the probability mass that D assigns to point x, so $D(x) \geq 0$ and $\sum_{x\in\mathcal{W}} D(x) = 1$. For $S \subseteq \mathcal{W}$, we write $D(S)$ to denote $\sum_{x\in S} D(x)$. For a finite set X we write $x \in_U X$ to indicate that x is chosen uniformly at random from X. For a random variable x, we will write $x \sim D$ to denote that x follows distribution D. Let D, D' be distributions over $\mathcal{W}$. The *total variation distance* between D and D' is $d_{\mathrm{TV}}(D,D') \stackrel{\text{def}}{=} \max_{S\subseteq\mathcal{W}} |D(S) - D'(S)| = (1/2)\cdot\|D-D'\|_1$, where $\|D-D'\|_1 = \sum_{x\in\mathcal{W}} |D(x) - D'(x)|$ is the L_1–distance between D and D'.

We will denote by $\mathcal{C}_n$, or simply $\mathcal{C}$, a Boolean concept class, i.e., a class of functions mapping $\{-1,1\}^n$ to $\{-1,1\}$. We usually consider syntactically defined classes of functions such as the class of all n-variable linear threshold functions or the class of all n-variable s-term DNF formulas. We stress that throughout this chapter a class $\mathcal{C}$ is viewed as a *representation class*. Thus we will say that an algorithm "takes as input a function $f \in \mathcal{C}$" to mean that the input of the algorithm is a *representation* of $f \in \mathcal{C}$.

We will use the notation $\mathcal{U}_n$ (or simply $\mathcal{U}$, when the dimension n is clear from the context) for the uniform distribution over $\{-1,1\}^n$. Let $f : \{-1,1\}^n \to \{-1,1\}$. We will denote by $\mathcal{U}_{f^{-1}(1)}$ the uniform distribution over satisfying assignments of f. Let D be a distribution over $\{-1,1\}^n$ with $0 < D(f^{-1}(1)) < 1$. We write $D_{f,+}$ to denote the conditional distribution D restricted to $f^{-1}(1)$; so for $x \in f^{-1}(1)$ we have $D_{f,+}(x) = D(x)/D(f^{-1}(1))$. Observe that, with this notation, we have that $\mathcal{U}_{f^{-1}(1)} \equiv \mathcal{U}_{f,+}$.

We proceed to define the notions of approximate counting and approximate uniform generation for a class of Boolean functions:

Definition 73 (approximate counting)**.** *Let $\mathcal{C}$ be a class of n-variable Boolean functions. A randomized algorithm $\mathcal{A}^{\mathcal{C}}_{\text{count}}$ is an* efficient approximate counting algorithm for class $\mathcal{C}$, *if for any $\epsilon, \delta > 0$ and any $f \in \mathcal{C}$, on input ϵ, δ and $f \in \mathcal{C}$, it runs in time* $\text{poly}(n, 1/\epsilon, \log(1/\delta))$ *and with probability $1 - \delta$ outputs a value $\widehat{p}$ such that*

$$\frac{1}{(1+\epsilon)} \cdot \mathop{\mathbf{Pr}}_{x \sim \mathcal{U}}[f(x) = 1] \leq \widehat{p} \leq (1+\epsilon) \cdot \mathop{\mathbf{Pr}}_{x \sim \mathcal{U}}[f(x) = 1].$$

Definition 74 (approximate uniform generation)**.** *Let $\mathcal{C}$ be a class of n-variable Boolean functions. A randomized algorithm $\mathcal{A}^{\mathcal{C}}_{\text{gen}}$ is an* efficient approximate uniform generation algorithm for class $\mathcal{C}$, *if for any $\epsilon > 0$ and any $f \in \mathcal{C}$, there is a distribution $D = D_{f,\epsilon}$ supported on $f^{-1}(1)$ with*

$$\frac{1}{1+\epsilon} \cdot \frac{1}{|f^{-1}(1)|} \leq D(x) \leq (1+\epsilon) \cdot \frac{1}{|f^{-1}(1)|}$$

for each $x \in f^{-1}(1)$, such that for any $\delta > 0$, on input ϵ, δ and $f \in \mathcal{C}$, algorithm $A^{\mathcal{C}}_{\text{gen}}(\epsilon, \delta, f)$ runs in time $\text{poly}(n, 1/\epsilon, \log(1/\delta))$ *and either outputs a point $x \in f^{-1}(1)$ that is distributed precisely according to $D = D_{f,\epsilon}$, or outputs $\perp$. Moreover the probability that it outputs $\perp$ is at most δ.*

An approximate uniform generation algorithm is said to be *fully polynomial* if its running time dependence on ϵ is $\text{poly}(\log(1/\epsilon))$.

Before we define our inverse approximate uniform generation problem, we need the notion of a *sampler* for a distribution:

Definition 75. *Let D be a distribution over $\{-1, 1\}^n$. A* sampler for D *is a circuit C with $m =$* $\text{poly}(n)$ *input bits $z \in \{-1, 1\}^m$ and n output bits $x \in \{-1, 1\}^n$ which is such that when $z \sim \mathcal{U}_m$ then $x \sim D$. For $\epsilon > 0$, an* ϵ-sampler for D *is a sampler for some distribution D' which has $d_{\text{TV}}(D', D) \leq \epsilon$.*

For clarity we sometimes write "C is a 0-sampler for D" to emphasize the fact that the outputs of $C(z)$ are distributed *exactly* according to distribution D. We are now ready to formally define the notion of an inverse approximate uniform generation algorithm:

Definition 76 (inverse approximate uniform generation)**.** *Let $\mathcal{C}$ be a class of n-variable Boolean functions. A randomized algorithm $\mathcal{A}^{\mathcal{C}}_{\text{inv}}$ is an* inverse approximate uniform generation algorithm for class $\mathcal{C}$, *if for any $\epsilon, \delta > 0$ and any $f \in \mathcal{C}$, on input ϵ, δ and sample access to $\mathcal{U}_{f^{-1}(1)}$, with probability $1 - \delta$ algorithm $A^{\mathcal{C}}_{\text{inv}}$ outputs an ϵ-sampler C_f for $\mathcal{U}_{f^{-1}(1)}$.*

Hypothesis Testing.

Our general approach works by generating a collection of hypothesis distributions, one of which is close to the target distribution $\mathcal{U}_{f^{-1}(1)}$. Thus, we need a way to select a high-accuracy hypothesis distribution from a pool of candidate distributions which contains at least one high-accuracy hypothesis. This problem has been well studied, see e.g. Chapter 7 of [37]. We use the following result which is an extension of Lemma C.1 of [26].

Proposition 77. *Let D be a distribution over a finite set $\mathcal{W}$ and $\mathcal{D}_\epsilon = \{D_j\}_{j=1}^N$ be a collection of N distributions over $\mathcal{W}$ with the property that there exists $i \in [N]$ such that $d_{\mathrm{TV}}(D, D_i) \le \epsilon$. There is an algorithm $\mathcal{T}^D$, which is given access to:*

(i) samplers for D and D_k, for all $k \in [N]$,

(ii) a $(1+\beta)$–approximate evaluation oracle $\mathrm{EVAL}_{D_k}(\beta)$, for all $k \in [N]$, which, on input $w \in \mathcal{W}$, deterministically outputs a value $\widetilde{D}_k^\beta(w)$, such that $D_k(w)/(1+\beta) \le \widetilde{D}_k^\beta(w) \le (1+\beta)D_k(w)$, where $\beta > 0$ is any parameter satisfying $(1+\beta)^2 \le 1 + \epsilon/8$,

an accuracy parameter ϵ and a confidence parameter δ, and has the following behavior: It makes

$$m = O\left((1/\epsilon^2) \cdot (\log N + \log(1/\delta))\right)$$

draws from D and from each D_k, $k \in [N]$, and $O(m)$ calls to each oracle $\mathrm{EVAL}_{D_k}(\beta)$, $k \in [N]$, performs $O(mN^2)$ arithmetic operations, and with probability $1-\delta$ outputs an index $i^\star \in [N]$ that satisfies $d_{\mathrm{TV}}(D, D_{i^\star}) \le 6\epsilon$.

Before we proceed with the proof, we note that there are certain crucial differences between the current setting and the setting of [26, 27] (as well as other related works that use versions of Proposition 77). In particular, in our setting, the set $\mathcal{W}$ is of size 2^n, which was not the case in [26, 27]. Hence, we cannot assume the distributions D_i are given explicitly in the input. Thus Proposition 77 carefully specifies what kind of access to these distributions is required. Proposition 77 is an extension of similar results in the previous works; while the idea of the proof is essentially the same, the details are more involved. We postpone the proof to Appendix C.1.

Remark 78. As stated Proposition 77 assumes that algorithm $\mathcal{T}^D$ has access to samplers for all the distributions D_k, so each call to such a sampler is guaranteed to output an element distributed according to D_k. Let $D_k^\perp$ be a distribution over $\mathcal{W} \cup \{\perp\}$ which is such that (i) $D_k^\perp(\perp) \le 1/2$, and (ii) the conditional distribution $(D_k^\perp)_\mathcal{W}$ of $D_k^\perp$ conditioned on not outputting $\perp$ is precisely D_k. It is easy to see that the proof of Proposition 77 extends to a setting in which $\mathcal{T}^D$ has access to samplers for $D_k^\perp$ rather than samplers for D_k; each time a sample from D_k is required the algorithm can simply invoke the sampler for $D_k^\perp$ repeatedly until an element other than $\perp$ is obtained. (The low-probability event that many repetitions are ever needed can be "folded into" the failure probability δ.)

4.2 A general technique for inverse approximate uniform generation

In this section we present a general technique for solving inverse approximate uniform generation problems. Our main positive results follow this conceptual framework. At the heart of our approach is a new type of algorithm which we call a *densifier* for a concept class $\mathcal{C}$. Roughly speaking, this is an algorithm which, given uniform random positive examples of an unknown $f \in \mathcal{C}$, constructs a set S which (essentially) contains all of $f^{-1}(1)$ and which is such that $f^{-1}(1)$ is "dense" in S. Our main result in this section, Theorem 81, states (roughly speaking) that the existence of (i) a computationally efficient densifier, (ii) an efficient approximate uniform generation algorithm, (iii) an efficient approximate counting algorithm, and (iv) an efficient *statistical query (SQ)* learning algorithm, together suffice to yield an efficient algorithm for our inverse approximate uniform generation problem.

We have already defined approximate uniform generation and approximate counting algorithms, so we need to define SQ learning algorithms and densifiers. The *statistical query* (SQ) learning model is a natural restriction of the PAC learning model in which a learning algorithm is allowed to obtain estimates of statistical properties of the examples but cannot directly access the examples themselves. Let D be a distribution over $\{-1,1\}^n$. In the SQ model [85], the learning algorithm has access to a *statistical query oracle*, $\mathrm{STAT}(f, D)$, to which it can make a query of the form (χ, τ), where $\chi : \{-1,1\}^n \times \{-1,1\} \to [-1,1]$ is the *query function* and $\tau > 0$ is the *tolerance*. The oracle responds with a value v such that $|\mathbf{E}_{x\sim D}\left[\chi\left(x, f(x)\right)\right] - v| \leq \tau$, where $f \in \mathcal{C}$ is the target concept. The goal of the algorithm is to output a hypothesis $h : \{-1,1\}^n \to \{-1,1\}$ such that $\mathbf{P}r_{x\sim D}[h(x) \neq f(x)] \leq \epsilon$. The following is a precise definition:

Definition 79. *Let $\mathcal{C}$ be a class of n-variable Boolean functions and D be a distribution over $\{-1,1\}^n$. An* SQ learning algorithm for $\mathcal{C}$ under D *is a randomized algorithm $\mathcal{A}_{\mathrm{SQ}}^{\mathcal{C}}$ that for every $\epsilon, \delta > 0$, every target concept $f \in \mathcal{C}$, on input ϵ, δ and with access to oracle $\mathrm{STAT}(f, D)$ and to independent samples drawn from D, outputs with probability $1-\delta$ a hypothesis $h : \{-1,1\}^n \to \{-1,1\}$ such that $\mathbf{P}r_{x\sim D}[h(x) \neq f(x)] \leq \epsilon$. Let $t_1(n, 1/\epsilon, 1/\delta)$ be the running time of $\mathcal{A}_{\mathrm{SQ}}^{\mathcal{C}}$ (assuming each oracle query is answered in unit time), $t_2(n)$ be the maximum running time to evaluate any query provided to $\mathrm{STAT}(f, D)$ and $\tau(n, 1/\epsilon)$ be the minimum value of the tolerance parameter ever provided to $\mathrm{STAT}(f, D)$ in the course of $\mathcal{A}_{\mathrm{SQ}}^{\mathcal{C}}$'s execution. We say that $\mathcal{A}_{\mathrm{SQ}}^{\mathcal{C}}$ is* efficient *(and that $\mathcal{C}$ is* efficiently SQ *learnable with respect to distribution D), if $t_1(n, 1/\epsilon, 1/\delta)$ is polynomial in n, $1/\epsilon$ and $1/\delta$, $t_2(n)$ is polynomial in n and $\tau(n, 1/\epsilon)$ is lower bounded by an inverse polynomial in n and $1/\epsilon$. We call an* SQ *learning algorithm $\mathcal{A}_{\mathrm{SQ}}^{\mathcal{C}}$ for $\mathcal{C}$* distribution independent *if $\mathcal{A}_{\mathrm{SQ}}^{\mathcal{C}}$ succeeds for any distribution D. If $\mathcal{C}$ has an efficient distribution independent* SQ *learning algorithm we say that $\mathcal{C}$ is* efficiently SQ learnable (distribution independently).

We sometimes write an "(ϵ, δ)–SQ learning algorithm" to explicitly state the accuracy parameter ϵ and confidence parameter Throughout this chapter, we will only deal with distribution independent SQ learning algorithms.

To state our main result, we introduce the notion of a *densifier* for a class $\mathcal{C}$ of Boolean functions. Intuitively, a densifier is an algorithm which is given access to samples from $\mathcal{U}_{f^{-1}(1)}$ (where f is an unknown element of $\mathcal{C}$) and outputs a subset $S \subseteq \{-1,1\}^n$ which is such that (i) S contains "almost all" of $f^{-1}(1)$, but (ii) S is "much smaller" than $\{-1,1\}^n$ – in particular it is small enough that $f^{-1}(1) \cap S$ is (at least moderately) "dense" in S.

Definition 80. *Fix a function $\gamma(n, 1/\epsilon, 1/\delta)$ taking values in $(0,1]$ and a class $\mathcal{C}$ of n-variable Boolean functions. An algorithm $\mathcal{A}_{\text{den}}^{(\mathcal{C},\mathcal{C}')}$ is said to be* a γ-densifier for function class $\mathcal{C}$ using class $\mathcal{C}'$ *if it has the following behavior: For every $\epsilon, \delta > 0$, every $1/2^n \leq \widehat{p} \leq 1$, and every $f \in \mathcal{C}$, given as input $\epsilon, \delta, \widehat{p}$ and a set of independent samples from $\mathcal{U}_{f^{-1}(1)}$, the following holds: Let $p \overset{def}{=} \mathbf{Pr}_{x \sim \mathcal{U}_n}[f(x) = 1]$. If $p \leq \widehat{p} < (1+\epsilon)p$, then with probability at least $1-\delta$, algorithm $\mathcal{A}_{\text{den}}^{(\mathcal{C},\mathcal{C}')}$ outputs a function $g \in \mathcal{C}'$ such that:*

(a) $\mathbf{Pr}_{x \sim \mathcal{U}_{f^{-1}(1)}}[g(x) = 1] \geq 1 - \epsilon$.

(b) $\mathbf{Pr}_{x \sim \mathcal{U}_{g^{-1}(1)}}[f(x) = 1] \geq \gamma(n, 1/\epsilon, 1/\delta)$.

We will sometimes write an "$(\epsilon, \gamma, \delta)$–densifier" to explicitly state the parameters in the definition.

Our main conceptual approach is summarized in the following theorem:

Theorem 81 (General Upper Bound). *Let $\mathcal{C}, \mathcal{C}'$ be classes of n-variable Boolean functions. Suppose that*

- *$\mathcal{A}_{\text{den}}^{(\mathcal{C},\mathcal{C}')}$ is an $(\epsilon, \gamma, \delta)$-densifier for $\mathcal{C}$ using $\mathcal{C}'$ running in time $T_{\text{den}}(n, 1/\epsilon, 1/\delta)$.*
- *$\mathcal{A}_{\text{gen}}^{\mathcal{C}'}$ is an (ϵ, δ)-approximate uniform generation algorithm for $\mathcal{C}'$ running in time $T_{\text{gen}}(n, 1/\epsilon, 1/\delta)$.*
- *$\mathcal{A}_{\text{count}}^{\mathcal{C}'}$ is an (ϵ, δ)-approximate counting algorithm for $\mathcal{C}'$ running in time $T_{\text{count}}(n, 1/\epsilon, 1/\delta)$.*
- *$\mathcal{A}_{\text{SQ}}^{\mathcal{C}}$ is an (ϵ, δ)-SQ learning algorithm for $\mathcal{C}$ such that: $\mathcal{A}_{\text{SQ}}^{\mathcal{C}}$ runs in time $t_1(n, 1/\epsilon, 1/\delta)$, $t_2(n)$ is the maximum time needed to evaluate any query provided to $\mathrm{STAT}(f, D)$, and $\tau(n, 1/\epsilon)$ is the minimum value of the tolerance parameter ever provided to $\mathrm{STAT}(f, D)$ in the course of $\mathcal{A}_{\text{SQ}}^{\mathcal{C}}$'s execution.*

Then there exists an inverse approximate uniform generation algorithm $\mathcal{A}_{\text{inv}}^{\mathcal{C}}$ for $\mathcal{C}$. The running time of $\mathcal{A}_{\text{inv}}^{\mathcal{C}}$ is polynomial in $T_{\text{den}}(n, 1/\epsilon, 1/\delta)$, $1/\gamma$, $T_{\text{gen}}(n, 1/\epsilon, 1/\delta)$, $T_{\text{count}}\ (n, 1/\epsilon, 1/\delta)$, $t_1(n, 1/\epsilon, 1/\delta)$, $t_2(n)$ and $1/\tau(n, 1/\epsilon)$ provided the dependence of $T_{\text{den}}(\cdot)$, $T_{\text{gen}}(\cdot)$, $T_{\text{count}}(\cdot)$, $t_1(\cdot)$, $t_2(\cdot)$ and $\tau(\cdot)$ is polynomial in their input parameters. [2]

Sketch of the algorithm. The inverse approximate uniform generation algorithm $\mathcal{A}_{\text{inv}}^{\mathcal{C}}$ for $\mathcal{C}$ works in three main conceptual steps. Let $f \in \mathcal{C}$ be the unknown target function and recall that our algorithm $\mathcal{A}_{\text{inv}}^{\mathcal{C}}$ is given access to samples from $\mathcal{U}_{f^{-1}(1)}$.

[2]It is straightforward to derive an explicit running time bound for $\mathcal{A}_{\text{inv}}^{\mathcal{C}}$ in terms of the above functions from our analysis, but the resulting expression is extremely long and rather uninformative so we do not provide it.

(1) In the first step, $\mathcal{A}_{\text{inv}}^{\mathcal{C}}$ runs the densifier $\mathcal{A}_{\text{den}}^{(\mathcal{C},\mathcal{C}')}$ on a set of samples from $\mathcal{U}_{f^{-1}(1)}$. Let $g \in \mathcal{C}'$ be the output function of $\mathcal{A}_{\text{den}}^{(\mathcal{C},\mathcal{C}')}$.

Note that by setting the input to the approximate uniform generation algorithm $\mathcal{A}_{\text{gen}}^{\mathcal{C}'}$ to g, we obtain an approximate sampler C_g for $\mathcal{U}_{g^{-1}(1)}$. The output distribution D' of this sampler, is by definition supported on $g^{-1}(1)$ and is close to $D = \mathcal{U}_{g^{-1}(1)}$ in total variation distance.

(2) The second step is to run the SQ-algorithm $\mathcal{A}_{\text{SQ}}^{\mathcal{C}}$ to learn the function $f \in C$ under the distribution D. Let h be the hypothesis constructed by $\mathcal{A}_{\text{SQ}}^{\mathcal{C}}$.

(3) In the third and final step, the algorithm simply samples from C_g until it obtains an example x that has $h(x) = 1$, and outputs this x.

Remark 82. The reader may have noticed that the above sketch does not seem to use the approximate counting algorithm $\mathcal{A}_{\text{count}}^{\mathcal{C}'}$; we will revisit this point below.

Remark 83. The connection between the above algorithm sketch and the "standard approach" discussed in the Introduction is as follows: The function $g \wedge h$ essentially corresponds to the reconstructed object $\tilde{x}$ of the "standard approach." The process of sampling from C_g and doing rejection sampling until an input that satisfies h is obtained, essentially corresponds to the A_{sample} procedure of the "standard approach."

Intuition, motivation and discussion.

To motivate the high-level idea behind our algorithm, consider a setting in which $f^{-1}(1)$ is only a tiny fraction (say $1/2^{\Theta(n)}$) of $\{-1,1\}^n$. It is intuitively clear that we would like to use some kind of a learning algorithm in order to come up with a good approximation of $f^{-1}(1)$, but we need this approximation to be accurate at the "scale" of $f^{-1}(1)$ itself rather than at the scale of all of $\{-1,1\}^n$, so we need some way to ensure that the learning algorithm's hypothesis is accurate at this small scale. By using a densifier to construct g such that $g^{-1}(1)$ is not too much larger than $f^{-1}(1)$, we can use the distribution $D = \mathcal{U}_{g^{-1}(1)}$ to run a learning algorithm and obtain a good approximation of $f^{-1}(1)$ at the desired scale. (Since D and D' are close in variation distance, this implies we also learn f with respect to D'.)

To motivate our use of an SQ learning algorithm rather than a standard PAC learning algorithm, observe that there seems to be no way to obtain correctly labeled examples distributed according to D. However, we show that it is possible to accurately simulate statistical queries under D having access only to random positive examples from $f^{-1}(1)$ and to unlabeled examples drawn from D (subject to additional technical caveats discussed below). We discuss the issue of how it is possible to successfully use an SQ learner in our setting in more detail below.

Discussion and implementation issues. While the three main conceptual steps (1)-(3) of our algorithm may (hopefully) seem quite intuitive in light of the preceding motivation, a few issues immediately arise in thinking about how to implement these steps. The first one concerns running the SQ-algorithm $\mathcal{A}_{\text{SQ}}^{\mathcal{C}}$ in Step 2 to learn f under distribution D (recall that $D = \mathcal{U}_{g^{-1}(1)}$ and is close to D'). Our algorithm $\mathcal{A}_{\text{inv}}^{\mathcal{C}}$ needs to be able to efficiently simulate $\mathcal{A}_{\text{SQ}}^{\mathcal{C}}$ given its available information. While it would be easy to do so given access to random labeled examples $(x, f(x))$, where $x \sim D$, such information is not available in our setting. To overcome this obstacle, we show (see Proposition 85) that for *any* samplable distribution D, we can efficiently simulate a statistical query algorithm under D using samples from $D_{f,+}$. This does not quite solve the problem, since we only have samples from $\mathcal{U}_{f^{-1}(1)}$. However, we show (see Claim 88) that for our setting, i.e., for $D = \mathcal{U}_{g^{-1}(1)}$, we can simulate a sample from $D_{f,+}$ by a simple rejection sampling procedure using samples from $\mathcal{U}_{f^{-1}(1)}$ and query access to g.

Some more issues remain to be handled. First, the simulation of the statistical query algorithm sketched in the previous paragraph only works under the assumption that we are given a sufficiently accurate approximation $\widetilde{b_f}$ of the probability $\mathbf{P}r_{x\sim D}[f(x) = 1]$. (Intuitively, our approximation should be smaller than the smallest tolerance τ provided to the statistical query oracle by the algorithm $\mathcal{A}_{\text{SQ}}^{\mathcal{C}}$.) Second, by Definition 80, the densifier only succeeds under the assumption that it is given in its input an $(1 + \epsilon)$-multiplicative approximation $\widehat{p}$ to $p = \mathbf{P}r_{x\in\mathcal{U}_n}[f(x) = 1]$.

We handle these issues as follows: First, we show (see Claim 89) that, given an accurate estimate $\widehat{p}$ and a "dense" function $g \in \mathcal{C}'$, we can use the approximate counting algorithm $\mathcal{A}_{\text{count}}^{\mathcal{C}'}$ to efficiently compute an accurate estimate $\widetilde{b_f}$. (This is one reason why Theorem 81 requires an approximate counting algorithm for $\mathcal{C}'$.) To deal with the fact that we do not a priori have an accurate estimate $\widehat{p}$, we run our sketched algorithm for all possible values of $\mathbf{P}r_{x\sim\mathcal{U}_n}[f(x) = 1]$ in an appropriate multiplicative "grid" of size $N = O(n/\epsilon)$, covering all possible values from $1/2^n$ to 1. We thus obtain a set $\mathcal{D}$ of N candidate distributions one of which is guaranteed to be close to the true distribution $\mathcal{U}_{f^{-1}(1)}$ in variation distance. At this point, we would like to apply our hypothesis testing machinery (Proposition 77) to find such a distribution. However, in order to use Proposition 77, in addition to sample access to the candidate distributions (and the distribution being learned), we also require a *multiplicatively accurate* approximate evaluation oracle to evaluate the probability mass of any point under the candidate distributions. We show (see Lemma 169) that this is possible in our generic setting, using properties of the densifier and the approximate counting algorithm $\mathcal{A}_{\text{count}}^{\mathcal{C}'}$ for $\mathcal{C}'$.

Now we are ready to begin the detailed proof of Theorem 81. The reader who wishes to proceed directly to the applications of Theorem 81 may skip to Section 4.3.

Simulating statistical query algorithms.

Our algorithm $\mathcal{A}_{\text{inv}}^{\mathcal{C}}$ will need to simulate a statistical query algorithm for $\mathcal{C}$, with respect to a specific distribution D. Note, however that $\mathcal{A}_{\text{inv}}$ only has access to uniform positive examples of $f \in \mathcal{C}$, i.e., samples from $\mathcal{U}_{f^{-1}(1)}$. Hence we need to show that a statistical query algorithm can be efficiently simulated in such a setting. To do this it suffices to show that one can efficiently provide valid responses to queries to the statistical query oracle $\text{STAT}(f, D)$, i.e., that one can simulate the oracle. Assuming this can be done, the simulation algorithm $\mathcal{A}_{\text{SQ}-\text{SIM}}$ is very simple: Run the statistical query algorithm $\mathcal{A}_{\text{SQ}}$, and whenever it makes a query to $\text{STAT}(f, D)$, simulate it. To this end, in the following lemma we describe a procedure that simulates an SQ oracle.

(Our approach here is similar to that of earlier simulation procedures that have been given in the literature, see e.g. Denis *et al.* [35].)

Lemma 84. *Let $\mathcal{C}$ be a concept class over $\{-1,1\}^n$, $f \in \mathcal{C}$, and D be a samplable distribution over $\{-1,1\}^n$. There exists an algorithm* $\texttt{Simulate-STAT}_f^D$ *with the following properties: It is given access to independent samples from $D_{f,+}$, and takes as input a number $\widetilde{b_f} \in [0,1]$, a $t(n)$-time computable query function $\chi : \{-1,1\}^n \times \{-1,1\} \to [-1,1]$, a tolerance τ and a confidence δ. It has the following behavior: it uses $m = O\left((1/\tau^2)\log(1/\delta)\right)$ samples from D and $D_{f,+}$, runs in time $O\left(m \cdot t(n)\right)$, and if $|\widetilde{b_f} - \mathbf{Pr}_{x\sim D}[f(x)=1]| \le \tau'$, then with probability $1-\delta$ it outputs a number v such that*

$$|\mathbf{E}_{x\sim D}\left[\chi\left(x, f(x)\right)\right] - v| \le \tau + \tau'. \tag{4.1}$$

The proof of Lemma 84 is deferred to Appendix C.2. Given the above lemma, we can state and prove our general result for simulating SQ algorithms:

Proposition 85. *Let $\mathcal{C}$ be a concept class and D be a samplable distribution over $\{-1,1\}^n$. Suppose there exists an SQ-learning algorithm $\mathcal{A}_{\mathrm{SQ}}$ for $\mathcal{C}$ under D with the following performance: $\mathcal{A}_{\mathrm{SQ}}$ runs in time $T_1 = t_1(n, 1/\epsilon, 1/\delta)$, each query provided to $\mathrm{STAT}(f,D)$ can be evaluated in time $T_2 = t_2(n)$, and the minimum value of the tolerance provided to $\mathrm{STAT}(f,D)$ in the course of its execution is $\tau = \tau(n, 1/\epsilon)$. Then, there exists an algorithm $\mathcal{A}_{\mathrm{SQ-SIM}}$ that is given access to*

(i) independent samples from $D_{f,+}$; and

(ii) a number $\widetilde{b_f} \in [0,1]$,

and efficiently simulates the behavior of $\mathcal{A}_{\mathrm{SQ}}$. In particular, $\mathcal{A}_{\mathrm{SQ-SIM}}$ has the following performance guarantee: on input an accuracy ϵ and a confidence δ, it uses $m = O\left((1/\tau^2)\cdot\log(T_1/\delta)\cdot T_1\right)$ samples from D and $D_{f,+}$, runs in time $T_{\mathrm{SQ-SIM}} = O\left(mT_2\right)$, and if $|\widetilde{b_f} - \mathbf{Pr}_{x\sim D}[f(x)=1]| \le \tau/2$ then with probability $1-\delta$ it outputs a hypothesis $h : \{-1,1\}^n \to \{-1,1\}$ such that $\mathbf{Pr}_{x\sim D}[h(x) \ne f(x)] \le \epsilon$.

The proof of Proposition 85 is deferred to Appendix C.2. Proposition 85 tells us we can efficiently simulate a statistical query algorithm for a concept class $\mathcal{C}$ under a samplable distribution D if we have access to samples drawn from $D_{f,+}$ (and a very accurate estimate of $\mathbf{Pr}_{x\sim D}[f(x)=1]$). In our setting, we have that $D = \mathcal{U}_{g^{-1}(1)}$ where $g \in \mathcal{C}'$ is the function that is output by $\mathcal{A}_{\mathrm{den}}^{(\mathcal{C},\mathcal{C}')}$. So, the two issues we must handle are (i) obtaining samples from D, and (ii) obtaining samples from $D_{f,+}$.

For (i), we note that, even though we do not have access to samples drawn *exactly* from D, it suffices for our purposes to use a τ'-sampler for D for a sufficiently small τ'. To see this we use the following fact:

Fact 86. *Let D, D' be distributions over $\{-1,1\}^n$ with $d_{\mathrm{TV}}(D,D') \le \tau'$. Then for any bounded function $\phi : \{-1,1\}^n \to [-1,1]$ we have that $|\mathbf{E}_{x\sim D}[\phi(x)] - \mathbf{E}_{x\sim D'}[\phi(x)]| \le 2\tau'$.*

The proof of Fact 86 is deferred to Appendix C.2. The above fact implies that the statement of Proposition 85 continuous to hold with the same parameters if instead of a 0-sampler for D we have access to a τ'-sampler for D, for $\tau' = \tau/8$. The only difference is that in Step 1 of the subroutine `Simulate-STAT`$_f^D$ we empirically estimate the expectation $\mathbf{E}_{x\sim D'}[\chi(x,-1)]$ up to an additive $\pm\tau/4$. By Fact 86, this will be a $\pm(\tau/4 + 2\tau') = \pm\tau/2$ accurate estimate for the $\mathbf{E}_{x\sim D}[\chi(x,-1)]$. That is, we have:

Corollary 87. *The statement of Proposition 85 continues to hold with the same parameters if instead of a* 0*-sampler for* D *we have access to a* $\tau' = \tau/8$*-sampler for* D*.*

For (ii), even though we do not have access to the distribution $D = \mathcal{U}_{g^{-1}(1)}$ directly, we note below that we can efficiently sample from $D_{f,+}$ using samples from $\mathcal{U}_{f^{-1}(1)}$ together with evaluations of g (recall again that g is provided as the output of the densifier).

Claim 88. *Let* $g : \{-1,1\}^n \to \{-1,1\}$ *be a* $t_g(n)$ *time computable function such that it satisfies* $\mathbf{P}r_{x\sim\mathcal{U}_{f^{-1}(1)}}[g(x) = 1] \geq \epsilon'$*. There is an efficient subroutine that is given* ϵ' *and a circuit to compute* g *as input, uses* $m = O((1/\epsilon')\log(1/\delta))$ *samples from* $\mathcal{U}_{f^{-1}(1)}$*, runs in time* $O(m \cdot t_g(n))$*, and with probability* $1-\delta$ *outputs a sample* x *such that* $x \sim D_{f,+}$*, where* $D = \mathcal{U}_{g^{-1}(1)}$*.*

As before, we defer the proof of Claim 88 to Appendix C.2.

Getting a good estimate $\widetilde{b_f}$ of $\mathbf{P}r_{x\sim D}[f(x) = 1]$. The simulations presented above require an additively accurate estimate $\widetilde{b_f}$ of $\mathbf{P}r_{x\sim D}[f(x) = 1]$. We now show that in our context, such an estimate can be easily obtained if we have access to a good estimate $\widehat{p}$ of $p = \mathbf{P}r_{x\in\mathcal{U}_n}[f(x) = 1]$, using the fact that we have an efficient approximate counting algorithm for $\mathcal{C}'$ and that $D \equiv \mathcal{U}_{g^{-1}(1)}$ where $g \in \mathcal{C}'$.

Claim 89. *Let* $g : \{-1,1\}^n \to \{-1,1\}$*,* $g \in \mathcal{C}'$ *be a* $t_g(n)$ *time computable function, satisfying* $\mathbf{P}r_{x\sim\mathcal{U}_{g^{-1}(1)}}[f(x) = 1] \geq \gamma'$ *and* $\mathbf{P}r_{x\sim\mathcal{U}_{f^{-1}(1)}}[g(x) = 1] \geq 1-\epsilon'$*. Let* $\mathcal{A}^{\mathcal{C}'}_{\text{count}}$ *be an* (ϵ,δ)*-approximate counting algorithm for* $\mathcal{C}'$ *running in time* $T_{\text{count}}(n, 1/\epsilon, 1/\delta)$*. There is a procedure* `Estimate-Bias` *with the following behavior:* `Estimate-Bias` *takes as input a value* $0 < \widehat{p} \leq 1$*, a parameter* $\tau' > 0$*, a confidence parameter* δ'*, and a representation of* $g \in \mathcal{C}'$*.* `Estimate-Bias` *runs in time* $O(t_g \cdot T_{\text{count}}(n, 2/\tau', 1/\delta'))$ *and satisfies the following: if* $p \overset{def}{=} \mathbf{P}r_{x\sim\mathcal{U}_n}[f(x) = 1] < \widehat{p} \leq (1+\epsilon')p$*, then with probability* $1-\delta'$ `Estimate-Bias` *outputs a value* $\widetilde{b_f}$ *such that* $|\widetilde{b_f} - \mathbf{P}r_{x\sim D}[f(x) = 1]| \leq \tau'$*.*

The proof of Claim 89 is deferred to Appendix C.2.

An algorithm that succeeds given the (approximate) bias of f.

In this section, we present an algorithm $\mathcal{A}'^{\mathcal{C}}_{\text{inv}}(\epsilon,\delta,\widehat{p})$ which, in addition to samples from $\mathcal{U}_{f^{-1}(1)}$, takes as input parameters $\epsilon,\delta,\widehat{p}$. The algorithm succeeds in outputting a hypothesis distribution D_f

satisfying $d_{\rm TV}(D_f, \mathcal{U}_{f^{-1}(1)}) \leq \epsilon$ if the input parameter $\widehat{p}$ is a multiplicatively accurate approximation to $\mathbf{Pr}_{x\sim\mathcal{U}_n}[f(x)=1]$. The algorithm follows the three high-level steps previously outlined and uses the subroutines of the previous subsection to simulate the statistical query algorithm.

```
Algorithm $\mathcal{A}'^{\mathcal{C}}_{\rm inv}(\mathcal{U}_{f^{-1}(1)}, \epsilon, \delta, \widehat{p})$:
Input: Independent samples from $\mathcal{U}_{f^{-1}(1)}$, accuracy and confidence parameters $\epsilon, \delta$, and a value $1/2^n < \widehat{p} \leq 1$.
Output: If $\mathbf{Pr}_{x\sim\mathcal{U}_n}[f(x)=1] \leq \widehat{p} < (1+\epsilon)\mathbf{Pr}_{x\sim\mathcal{U}_n}[f(x)=1]$, with probability $1-\delta$ outputs an $\epsilon$-sampler $C_f$ for $\mathcal{U}_{f^{-1}(1)}$.

1. Fix $\epsilon_1 \overset{\rm def}{=} \epsilon/6$ and $\gamma \overset{\rm def}{=} \gamma(n, 1/\epsilon_1, 3/\delta)$. Run the $\gamma$-densifier $\mathcal{A}^{(\mathcal{C},\mathcal{C}')}_{\rm den}(\epsilon_1, \delta/3, \widehat{p})$ using random samples from $\mathcal{U}_{f^{-1}(1)}$. Let $g \in \mathcal{C}'$ be its output.
2.  a) Fix $\epsilon_2 \overset{\rm def}{=} \epsilon\gamma/7$, $\tau_2 \overset{\rm def}{=} \tau(n, 1/\epsilon_2)$ and $m \overset{\rm def}{=} \Theta\left((1/\tau_2^2)\cdot\log(T_1/\delta)\cdot T_1\right)$, where $T_1 = t_1(n, 1/\epsilon_2, 12/\delta)$.
    b) Run the generator $\mathcal{A}^{\mathcal{C}'}_{\rm gen}(g, \tau_2/8, \delta/(12m))$ $m$ times and let $S_D \subseteq \{-1,1\}^n$ be the multiset of samples obtained.
    c) Run Simulate-sample$^{D_{f,+}}(\mathcal{U}_{f^{-1}(1)}, g, \gamma, \delta/(12m))$ $m$ times and let $S_{D_{f,+}} \subseteq \{-1,1\}^n$ be the multiset of samples obtained.
    d) Run Estimate-Bias with parameters $\widehat{p}$, $\tau' = \tau_2/2$, $\delta' = \delta/12$, using the representation for $g \in \mathcal{C}'$, and let $\widetilde{b_f}$ be the value it returns.
    e) Run $\mathcal{A}_{\rm SQ-SIM}(S_D, S_{D_{f,+}}, \epsilon_2, \widetilde{b_f}, \delta/12)$. Let $h : \{-1,1\}^n \to \{-1,1\}$ be the output hypothesis.
3. Output the sampler $C_f$ which works as follows:
    For $i = 1$ to $t = \Theta\left((1/\gamma)\log(1/(\delta\epsilon)\right)$ do:
        a) Set $\epsilon_3 \overset{\rm def}{=} \epsilon\gamma/48000$.
        b) Run the generator $\mathcal{A}^{\mathcal{C}'}_{\rm gen}(g, \epsilon_3, \delta\epsilon/(12t))$ and let $x^{(i)}$ be its output.
        c) If $h(x^{(i)}) = 1$, output $x^{(i)}$.
    If no $x^{(i)}$ with $h(x^{(i)}) = 1$ has been obtained, output the default element $\perp$.
   Let $\hat{D}$ denote the distribution over $\{-1,1\}^n \cup \{\perp\}$ for which $C_f$ is a 0-sampler, and let $\hat{D}'$ denote the conditional distribution of $\hat{D}$ restricted to $\{-1,1\}^n$ (i.e., excluding $\perp$).
```

We note that by inspection of the code for C_f, we have that the distribution $\hat{D}'$ is identical to $(D_{g,\epsilon_3})_{h^{-1}(1)}$, where D_{g,ϵ_3} is the distribution corresponding to the output of the approximate uniform generator when called on function g and error parameter ϵ_3 (see Definition 74) and $(D_{g,\epsilon_3})_{h^{-1}(1)}$ is D_{g,ϵ_3} conditioned on $h^{-1}(1)$.

We have the following:

Theorem 90. *Let* $p \overset{def}{=} \mathbf{Pr}_{x\in\mathcal{U}_n}[f(x) = 1]$. *Algorithm* $\mathcal{A}'^{\mathcal{C}}_{\text{inv}}(\epsilon, \delta, \widehat{p})$ *has the following behavior: If* $p \le \widehat{p} < (1+\epsilon)p$, *then with probability* $1-\delta$ *the following both hold:*

(i) the output C_f *is a sampler for a distribution* $\hat{D}$ *such that* $d_{\text{TV}}(\hat{D}, \mathcal{U}_{f^{-1}(1)}) \le \epsilon$*; and*

(ii) the functions h, g *satisfy* $|h^{-1}(1) \cap g^{-1}(1)|/|g^{-1}(1)| \ge \gamma/2$.

The running time of $\mathcal{A}'^{\mathcal{C}}_{\text{inv}}$ *is polynomial in* $T_{\text{den}}(n, 1/\epsilon, 1/\delta)$, $T_{\text{gen}}(n, 1/\epsilon, 1/\delta)$, $T_{\text{count}}(n, 1/\epsilon, 1/\delta)$, $t_1(n, 1/\epsilon, 1/\delta)$, $t_2(n)$, $1/\tau(n, 1/\epsilon)$, *and* $1/\gamma(n, 1/\epsilon, 1/\delta)$.

Proof. We give an intuitive explanation of the pseudocode in tandem with a proof of correctness. We argue that Steps 1-3 of the algorithm implement the corresponding steps of our high-level description and that the algorithm succeeds with confidence probability $1-\delta$.

We assume throughout the argument that indeed $\widehat{p}$ lies in $[p, (1+\epsilon)p)$. Given this, by Definition 80 with probability $1-\delta/3$ the function g satisfies properties (a) and (b) of Definition 80, i.e., $\mathbf{Pr}_{x\sim\mathcal{U}_{f^{-1}(1)}}[g(x) = 1] \ge 1-\epsilon_1$ and $\mathbf{Pr}_{x\sim\mathcal{U}_{g^{-1}(1)}}[f(x) = 1] \ge \gamma$. We condition on this event (which we denote E_1) going forth.

We now argue that Step 2 simulates the SQ learning algorithm $\mathcal{A}^{\mathcal{C}}_{\text{SQ}}$ to learn the function $f \in \mathcal{C}$ under distribution $D \equiv \mathcal{U}_{g^{-1}(1)}$ to accuracy ϵ_2 with confidence $1-\delta/3$. Note that the goal of Step (b) is to obtain m samples from a distribution D'' (the distribution "$D_{g,\tau_2/8}$" of Definition 74) such that $d_{\text{TV}}(D'', D) \le \tau_2/8$. To achieve this, we call the approximate uniform generator for g a total of m times with failure probability $\delta/(12m)$ for each call (i.e., each call returns $\perp$ with probability at most $\delta/(12m)$). By a union bound, with failure probability at most $\delta/12$, all calls to the generator are successful and we obtain a set S_D of m independent samples from D''. Similarly, the goal of Step (c) is to obtain m samples from $D_{f,+}$ and to achieve it we call the subroutine $\texttt{Simulate-sample}^{D_{f,+}}$ a total of m times with failure probability $\delta/(12m)$ each. By Claim 88 and a union bound, with failure probability at most $\delta/12$, this step is successful, i.e., it gives a set $S_{D_{f,+}}$ of m independent samples from $D_{f,+}$. The goal of Step (d) is to obtain a value $\widetilde{b_f}$ satisfying $|\widetilde{b_f} - \mathbf{Pr}_{x\sim D}[f(x) = 1]| \le \tau_2/2$; by Claim 89, with failure probability at most $\delta/12$ the value $\widetilde{b_f}$ obtained in this step is as desired. Finally, Step (e) applies the simulation algorithm $\mathcal{A}_{\text{SQ}-\text{SIM}}$ using the samples S_D and $S_{D_{f,+}}$ and the estimate $\widetilde{b_f}$ of $\mathbf{Pr}_{x\sim D}[f(x) = 1]$ obtained in the previous steps. Conditioning on Steps (b), (c) and (d) being successful Corollary 87 implies that Step (e) is successful with probability $1-\delta/12$, i.e., it outputs a hypothesis h that satisfies $\mathbf{Pr}_{x\sim D}[f(x) \ne h(x)] \le \epsilon_2$. A union bound over Steps (c), (d) and (e) completes the analysis of Step 2. For future reference, we let E_2 denote the event that the hypothesis h constructed in Step 2(e) has $\mathbf{Pr}_{x\sim D}[f(x) \ne h(x)] \le \epsilon_2$ (so we have that E_2 holds with probability at least $1-\delta/3$; we additionally condition on this event going forth). We observe that since (as we have just shown)

$\mathbf{Pr}_{x\sim\mathcal{U}_{g^{-1}(1)}}[f(x)\neq h(x)]\leq\epsilon_2$ and $\mathbf{Pr}_{x\sim\mathcal{U}_{g^{-1}(1)}}[f(x)=1]\geq\gamma$, we have $\mathbf{Pr}_{x\sim\mathcal{U}_{g^{-1}(1)}}[h(x)=1]\geq\gamma-\epsilon_2\geq\gamma/2$, which gives item (ii) of the theorem; so it remains to establish item (i) and the claimed running time bound.

To establish (i), we need to prove that the output distribution $\hat{D}$ of the sampler C_f is ϵ-close in total variation distance to $\mathcal{U}_{f^{-1}(1)}$. This sampler attempts to draws t samples from a distribution D' such that $d_{\mathrm{TV}}(D',D)\leq\epsilon_3$ (this is the distribution "D_{g,ϵ_3}" in the notation of Definition 74) and it outputs one of these samples that satisfies h (unless none of these samples satisfies h, in which case it outputs a default element $\perp$). The desired variation distance bound follows from the next lemma for our choice of parameters:

Lemma 91. *Let $\hat{D}$ be the output distribution of $\mathcal{A}'^{\mathcal{C}}_{\mathrm{inv}}(\mathcal{U}_{f^{-1}(1)},\epsilon,\delta,\widehat{p})$. If $\mathbf{Pr}_{x\sim\mathcal{U}_n}[f(x)=1]\leq\widehat{p}\leq(1+\epsilon)\mathbf{Pr}_{x\sim\mathcal{U}_n}[f(x)=1]$, then conditioned on Events E_1 and E_2, we have*

$$\begin{aligned} d_{\mathrm{TV}}(\hat{D},\mathcal{U}_{f^{-1}(1)}) &\leq \frac{\epsilon}{6}+\frac{\epsilon}{6}+\frac{4\epsilon_3}{\gamma}+\epsilon_1+\frac{\epsilon_2}{2\gamma}+\frac{\epsilon_2}{\gamma-\epsilon_2}\\ &\leq \frac{\epsilon}{6}+\frac{\epsilon}{6}+\frac{\epsilon}{12000}+\frac{\epsilon}{6}+\frac{\epsilon}{14}+\frac{\epsilon}{6}<\epsilon. \end{aligned}$$

Proof. Consider the distribution $D'=D_{g,\epsilon_3}$ (see Definition 74) produced by the approximate uniform generator in Step 3 of the algorithm. Let $D'|_{h^{-1}(1)}$ denote distribution D' restricted to $h^{-1}(1)$. Let S denote the set $g^{-1}(1)\cap h^{-1}(1)$. The lemma is an immediate consequence of Claims 92, 94, 95 and 96 below using the triangle inequality (everything below is conditioned on E_1 and E_2). □

Claim 92. $d_{\mathrm{TV}}(\hat{D},\hat{D}')\leq\epsilon/6$.

Proof. Recall that $\hat{D}'$ is simply $\hat{D}$ conditioned on not outputting $\perp$.

We first claim that with probability at least $1-\delta\epsilon/12$ all t points drawn in Step 3 of the code for C_f are distributed according to the distribution $D'=D_{g,\epsilon_3}$ over $g^{-1}(1)$. Each of the t calls to the approximate uniform generator has failure probability $\delta\epsilon/(12t)$ (of outputting $\perp$ rather than a point distributed according to D') so by a union bound no calls fail with probability at least $1-\delta\epsilon/12$, and thus with probability at least $1-\delta\epsilon/12$ indeed all t samples are independently drawn from such a distribution D'.

Conditioned on this, we claim that a satisfying assignment for h is obtained within the t samples with probability at least $1-\delta\epsilon/12$. This can be shown as follows:

Claim 93. *Let $h:\{-1,1\}^n\to\{-1,1\}$ be the hypothesis output by $\mathcal{A}^{\mathcal{C}}_{\mathrm{SQ-SIM}}$. We have*

$$\mathop{\mathbf{Pr}}_{x\sim D'}[h(x)=1]\geq\gamma/4.$$

Proof. First recall that, by property (b) in the definition of the densifier (Definition 80), we have $\mathbf{Pr}_{x\sim D}[f(x)=1]\geq\gamma$. Since $d_{\mathrm{TV}}(D',D)\leq\epsilon_3$, by definition we get

$$\mathop{\mathbf{Pr}}_{x\sim D'}[f(x)=1]\geq\mathop{\mathbf{Pr}}_{x\sim D}[f(x)=1]-\epsilon_3\geq\gamma-\epsilon_3\geq3\gamma/4.$$

Now by the guarantee of Step 2 we have that $\mathbf{P}r_{x\sim D}[f(x) \neq h(x)] \leq \epsilon_2$. Combined with the fact that $d_{\mathrm{TV}}(D', D) \leq \epsilon_3$, this implies that

$$\mathop{\mathbf{P}r}_{x\sim D'}[f(x) \neq h(x)] \leq \epsilon_2 + \epsilon_3 \leq \gamma/2.$$

Therefore, we conclude that

$$\mathop{\mathbf{P}r}_{x\sim D'}[h(x) = 1] \geq \mathop{\mathbf{P}r}_{x\sim D'}[f(x) = 1] - \mathop{\mathbf{P}r}_{x\sim D'}[f(x) \neq h(x)] \geq 3\gamma/4 - \gamma/2 \geq \gamma/4$$

as desired. □

Hence, for an appropriate constant in the big-Theta specifying t, with probability at least $1 - \delta\epsilon/12 > 1 - \delta/12$ some $x^{(i)}$ is a satisfying assignment of h. that with probability at least $1 - \delta\epsilon/12$ some $x^{(i)}$, $i \in [t]$, has $h(x) = 1$. Thus with overall failure probability at most $\delta\epsilon/6$ a draw from $\hat{D}'$ is not $\perp$, and consequently we have $d_{\mathrm{TV}}(\hat{D}, \hat{D}') \leq \delta\epsilon/6 \leq \epsilon/6$. □

Claim 94. $d_{\mathrm{TV}}(\hat{D}', D'|_{h^{-1}(1)}) \leq \epsilon/6$.

Proof. The probability that any of the t points $x^{(1)}, \ldots, x^{(t)}$ is not drawn from D' is at most $t \cdot \delta\epsilon/(12t) < \epsilon/12$. Assuming that this does not happen, the probability that no $x^{(i)}$ lies in $h^{-1}(1)$ is at most $(1 - \gamma/4)^t < \delta\epsilon/12 < \epsilon/12$ by Claim 93. Assuming this does not happen, the output of a draw from $\hat{D}$ is distributed identically according to $D'|_{h^{-1}(1)}$. Consequently we have that $d_{\mathrm{TV}}(\hat{D}, D'|_{h^{-1}(1)}) \leq \epsilon/6$ as claimed. □

Claim 95. $d_{\mathrm{TV}}(D'|_{h^{-1}(1)}, \mathcal{U}_S) \leq 4\epsilon_3/\gamma$.

Proof. The definition of an approximate uniform generator gives us that $d_{\mathrm{TV}}(D', \mathcal{U}_{g^{-1}(1)}) \leq \epsilon_3$, and Claim 93 gives that $\mathbf{P}r_{x\sim D'}[h(x) = 1] \geq \gamma/4$. We now recall the fact that for any two distributions D_1, D_2 and any event E, writing $D_i|_E$ to denote distribution D_i conditioned on event E, we have

$$d_{\mathrm{TV}}(D_1|_E, D_2|_E) \leq \frac{d_{\mathrm{TV}}(D_1, D_2)}{D_1(E)}.$$

The claim follows since $\mathcal{U}_{g^{-1}(1)}|_{h^{-1}(1)}$ is equivalent to $\mathcal{U}_S$. □

Claim 96. $d_{\mathrm{TV}}(\mathcal{U}_S, \mathcal{U}_{f^{-1}(1)}) \leq \epsilon_1 + \frac{\epsilon_2}{2\gamma} + \frac{\epsilon_2}{\gamma - \epsilon_2}$.

Proof. The proof requires a careful combination of the properties of the function g constructed by the densifier and the guarantee of the SQ algorithm. Recall that $S = g^{-1}(1) \cap h^{-1}(1)$. We consider the set $S' = g^{-1}(1) \cap f^{-1}(1)$. By the triangle inequality, we can bound the desired variation distance as follows:

$$d_{\mathrm{TV}}(\mathcal{U}_S, \mathcal{U}_{f^{-1}(1)}) \leq d_{\mathrm{TV}}(\mathcal{U}_{f^{-1}(1)}, \mathcal{U}_{S'}) + d_{\mathrm{TV}}(\mathcal{U}_{S'}, \mathcal{U}_S). \quad (4.2)$$

We will bound from above each term of the RHS in turn. To proceed we need an expression for the total variation distance between the uniform distribution on two finite sets. The following fact is obtained by straightforward calculation:

Fact 97. *Let A, B be subsets of a finite set $\mathcal{W}$ and $\mathcal{U}_A$, $\mathcal{U}_B$ be the uniform distributions on A, B respectively. Then,*

$$d_{\mathrm{TV}}(\mathcal{U}_A, \mathcal{U}_B) = (1/2) \cdot \frac{|A \cap \overline{B}|}{|A|} + (1/2) \cdot \frac{|B \cap \overline{A}|}{|B|} + (1/2) \cdot |A \cap B| \cdot \left| \frac{1}{|A|} - \frac{1}{|B|} \right|. \quad (4.3)$$

To bound the first term of the RHS of (4.2) we apply the above fact for $A = f^{-1}(1)$ and $B = S'$. Note that in this case $B \subseteq A$, hence the second term of (4.3) is zero. Regarding the first term, note that

$$\frac{|A \cap \overline{B}|}{|A|} = \frac{|f^{-1}(1) \cap \overline{g^{-1}(1)}|}{|f^{-1}(1)|} \leq \epsilon_1,$$

where the inequality follows from Property (a) of the densifier definition. Similarly, for the third term we can write

$$|A \cap B| \cdot \left| \frac{1}{|A|} - \frac{1}{|B|} \right| = |B| \cdot \left| \frac{1}{|A|} - \frac{1}{|B|} \right| = 1 - \frac{|B|}{|A|} = 1 - \frac{|f^{-1}(1) \cap g^{-1}(1)|}{|f^{-1}(1)|} \leq \epsilon_1,$$

where the inequality also follows from Property (a) of the densifier definition. We therefore conclude that $d_{\mathrm{TV}}(\mathcal{U}_{f^{-1}(1)}, \mathcal{U}_{S'}) \leq \epsilon_1$.

We now proceed to bound the second term of the RHS of (4.2) by applying Fact 97 for $A = S'$ and $B = S$. It turns out that bounding the individual terms of (4.3) is trickier in this case. For the first term we have:

$$\frac{|A \cap \overline{B}|}{|A|} = \frac{|f^{-1}(1) \cap g^{-1}(1) \cap \overline{h^{-1}(1)}|}{|f^{-1}(1) \cap g^{-1}(1)|} = \frac{|f^{-1}(1) \cap g^{-1}(1) \cap \overline{h^{-1}(1)}|}{|g^{-1}(1)|} \cdot \frac{|g^{-1}(1)|}{|f^{-1}(1) \cap g^{-1}(1)|} \leq \frac{\epsilon_2}{\gamma},$$

where the last inequality follows from the guarantee of the SQ learning algorithm and Property (b) of the densifier definition. For the second term we have

$$\frac{|B \cap \overline{A}|}{|B|} = \frac{|\overline{f^{-1}(1)} \cap g^{-1}(1) \cap h^{-1}(1)|}{|g^{-1}(1) \cap h^{-1}(1)|}.$$

To analyze this term we recall that by the guarantee of the SQ algorithm it follows that the numerator satisfies

$$|\overline{f^{-1}(1)} \cap g^{-1}(1) \cap h^{-1}(1)| \leq \epsilon_2 \cdot |g^{-1}(1)|.$$

From the same guarantee we also get

$$|f^{-1}(1) \cap g^{-1}(1) \cap \overline{h^{-1}(1)}| \leq \epsilon_2 \cdot |g^{-1}(1)|.$$

Now, Property (b) of the densifier definition gives $|f^{-1}(1) \cap g^{-1}(1)| \geq \gamma \cdot |g^{-1}(1)|$. Combing these two inequalities implies that

$$|g^{-1}(1) \cap h^{-1}(1)| \geq |f^{-1}(1) \cap g^{-1}(1) \cap h^{-1}(1)| \geq (\gamma - \epsilon_2) \cdot |g^{-1}(1)|.$$

In conclusion, the second term is upper bounded by $(1/2) \cdot \frac{\epsilon_2}{\gamma - \epsilon_2}$.

For the third term, we can write

$$|A \cap B| \cdot \left| \frac{1}{|A|} - \frac{1}{|B|} \right| = |f^{-1}(1) \cap g^{-1}(1) \cap h^{-1}(1)| \cdot \left| \frac{1}{|f^{-1}(1) \cap g^{-1}(1)|} - \frac{1}{|g^{-1}(1) \cap h^{-1}(1)|} \right|.$$

To analyze these term we relate the cardinalities of these sets. In particular, we can write

$$\begin{aligned} |f^{-1}(1) \cap g^{-1}(1)| &= |f^{-1}(1) \cap g^{-1}(1) \cap h^{-1}(1)| + |f^{-1}(1) \cap g^{-1}(1) \cap \overline{h^{-1}(1)}| \\ &\leq |f^{-1}(1) \cap g^{-1}(1) \cap h^{-1}(1)| + \epsilon_2 \cdot |g^{-1}(1)| \\ &\leq |f^{-1}(1) \cap g^{-1}(1) \cap h^{-1}(1)| + \frac{\epsilon_2}{\gamma} \cdot |f^{-1}(1) \cap g^{-1}(1)| \end{aligned}$$

where the last inequlity is Property (b) of the densifier defintion. Therefore, we obtain

$$(1 - \frac{\epsilon_2}{\gamma}) \cdot |f^{-1}(1) \cap g^{-1}(1)| \leq |f^{-1}(1) \cap g^{-1}(1) \cap h^{-1}(1)| \leq |f^{-1}(1) \cap g^{-1}(1)|.$$

Similarly, we have

$$\begin{aligned} |g^{-1}(1) \cap h^{-1}(1)| &= |f^{-1}(1) \cap g^{-1}(1) \cap h^{-1}(1)| + |\overline{f^{-1}(1)} \cap g^{-1}(1) \cap h^{-1}(1)| \\ &\leq |f^{-1}(1) \cap g^{-1}(1) \cap h^{-1}(1)| + \epsilon_2 \cdot |g^{-1}(1)| \\ &\leq |f^{-1}(1) \cap g^{-1}(1) \cap h^{-1}(1)| + \frac{\epsilon_2}{\gamma - \epsilon_2} \cdot |g^{-1}(1) \cap h^{-1}(1)| \end{aligned}$$

and therefore

$$(1 - \frac{\epsilon_2}{\gamma - \epsilon_2}) \cdot |g^{-1}(1) \cap h^{-1}(1)| \leq |f^{-1}(1) \cap g^{-1}(1) \cap h^{-1}(1)| \leq |g^{-1}(1) \cap h^{-1}(1)|.$$

The above imply that the third term is bounded by $(1/2) \cdot \frac{\epsilon_2}{\gamma - \epsilon_2}$. This completes the proof of the claim. □

With Lemma 91 established, to finish the proof of Theorem 90 it remains only to establish the claimed running time bound. This follows from a straightforward (but somewhat tedious) verification, using the running time bounds established in Lemma 84, Proposition 85, Corollary 87, Claim 88 and Claim 89. □

Getting from $\mathcal{A}'^{\mathcal{C}}_{\text{inv}}$ to $\mathcal{A}^{\mathcal{C}}_{\text{inv}}$: An approximate evaluation oracle.

Recall that the algorithm $\mathcal{A}'^{\mathcal{C}}_{\text{inv}}$ from the previous subsection is only guaranteed (with high probability) to output a sampler for a hypothesis distribution $\hat{D}$ that is statistically close to the target distribution $\mathcal{U}_{f^{-1}(1)}$ if it is given an input parameter $\widehat{p}$ satisfying $p \leq \widehat{p} < (1+\epsilon)p$, where $p \stackrel{\text{def}}{=} \mathbf{Pr}_{x \in \mathcal{U}_n}[f(x) = 1]$. Given this, a natural idea is to run $\mathcal{A}'^{\mathcal{C}}_{\text{inv}}$ a total of $k = O(n/\epsilon)$ times, using "guesses" for $\widehat{p}$ that increase multiplicatively as powers of $1 + \epsilon$, starting at $1/2^n$ (the smallest possible value) and going up to 1. This yields hypothesis distributions $\hat{D}_1, \ldots, \hat{D}_k$ where $\hat{D}_i$ is the distribution obtained by setting $\widehat{p}$ to $\widehat{p}_i \stackrel{\text{def}}{=} (1+\epsilon)^{i-1}/2^n$. With such distributions in hand, an obvious approach is to use the "hypothesis testing" machinery of Section 4.1 to identify a high-accuracy $\hat{D}_i$ from this collection. This is indeed the path we follow, but some care is needed to make the approach go through. The details are given in Section C.2.

4.3 Linear Threshold Functions

In this section we apply our general framework from Section 4.2 to prove Theorem 67, i.e., obtain a polynomial time algorithm for the problem of inverse approximate uniform generation for the class $\mathcal{C} = \mathbf{LTF}_n$ of n-variable linear threshold functions over $\{-1, 1\}^n$. More formally, we prove:

Theorem 98. *There is an algorithm $\mathcal{A}_{\text{inv}}^{\mathbf{LTF}}$ which is a* $\text{poly}\,(n, 1/\epsilon, 1/\delta)$*-time inverse approximate uniform generation algorithm for the class* $\mathbf{LTF}_n$.

The above theorem will follow as an application of Theorem 81 for $\mathcal{C}' = \mathcal{C} = \mathbf{LTF}_n$. The literature provides us with three of the four ingredients that our general approach requires for LTFs – approximate uniform generation, approximate counting, and Statistical Query learning – and our main technical contribution is giving the fourth necessary ingredient, a densifier. We start by recalling the three known ingredients in the following subsection.

Tools from the literature.

We first record two efficient algorithms for approximate uniform generation and approximate counting for $\mathbf{LTF}_n$, due to Dyer [43]:

Theorem 99. *(approximate uniform generation for* $\mathbf{LTF}_n$*, [43]) There is an algorithm $\mathcal{A}_{\text{gen}}^{\mathbf{LTF}}$ that on input (a weights–based representation of) an arbitrary $h \in \mathbf{LTF}_n$ and a confidence parameter $\delta > 0$, runs in time* $\text{poly}(n, \log(1/\delta))$ *and with probability $1 - \delta$ outputs a point x such that $x \sim \mathcal{U}_{h^{-1}(1)}$.*

We note that the above algorithm gives us a somewhat stronger guarantee than that in Definition 74. Indeed, the algorithm $\mathcal{A}_{\text{gen}}^{\mathbf{LTF}}$ with high probability outputs a point $x \in \{-1, 1\}^n$ whose distribution is *exactly* $\mathcal{U}_{h^{-1}(1)}$ (as opposed to a point whose distribution is *close* to $\mathcal{U}_{h^{-1}(1)}$).

Theorem 100. *(approximate counting for* $\mathbf{LTF}_n$*, [43]) There is an algorithm $\mathcal{A}_{\text{count}}^{\mathbf{LTF}}$ that on input (a weights–based representation of) an arbitrary $h \in \mathbf{LTF}_n$, an accuracy parameter $\epsilon > 0$ and a confidence parameter $\delta > 0$, runs in time* $\text{poly}(n, 1/\epsilon, \log(1/\delta))$ *and outputs $\widehat{p} \in [0, 1]$ that with probability $1 - \delta$ satisfies* $\widehat{p} \in [1 - \epsilon, 1 + \epsilon] \cdot \mathbf{Pr}_{x \sim \mathcal{U}_n}[h(x) = 1]$.

We also need an efficient SQ learning algorithm for halfpaces. This is provided to us by a result of Blum et. al. [15]:

Theorem 101. *(*SQ *learning algorithm for* $\mathbf{LTF}_n$*, [15]) There is a distribution-independent* SQ *learning algorithm $\mathcal{A}_{\text{SQ}}^{\mathbf{LTF}}$ for $\mathbf{LTF}_n$ that has running time $t_1 = \text{poly}(n, 1/\epsilon, \log(1/\delta))$, uses at most $t_2 = \text{poly}(n)$ time to evaluate each query, and requires tolerance of its queries no smaller than $\tau = 1/\text{poly}(n, 1/\epsilon)$.*

A densifier for $\mathbf{LTF}_n$.

The last ingredient we need in order to apply our Theorem 81 is a computationally efficient densifer for $\mathbf{LTF}_n$. This is the main technical contribution of this section and is summarized in the following theorem:

Theorem 102. *(efficient proper densifier for $\mathbf{LTF}_n$) Set $\gamma(\epsilon, \delta, n) \overset{def}{=} \Theta\left(\delta/(n^2 \log n)\right)$. There is an $(\epsilon, \gamma, \delta)$–densifier $\mathcal{A}_{\text{den}}^{\mathbf{LTF}}$ for $\mathbf{LTF}_n$ that, for any input parameters $0 < \epsilon, \delta$, $1/2^n \leq \widehat{p} \leq 1$, outputs a function $g \in \mathbf{LTF}_n$ and runs in time* $\text{poly}(n, 1/\epsilon, \log(1/\delta))$.

Discussion and intuition. Before we prove Theorem 102, we provide some intuition. Let $f \in \mathbf{LTF}_n$ be the unknown LTF and suppose that we would like to design an $(\epsilon, \gamma, \delta)$–densifier $\mathcal{A}_{\text{den}}^{\mathbf{LTF}}$ for f. That is, given sample access to $\mathcal{U}_{f^{-1}(1)}$, and a number $\widehat{p}$ satisfying $p \leq \widehat{p} < (1+\epsilon)p$, where $p = \mathbf{P}r_{x \in \mathcal{U}_n}[f(x) = 1]$, we would like to efficiently compute (a weights–based representation for) an LTF $g : \{-1,1\}^n \to \{-1,1\}$ such that the following conditions are satisfied:

(a) $\mathbf{P}r_{x \sim \mathcal{U}_{f^{-1}(1)}}[g(x) = 1] \geq 1 - \epsilon$, and

(b) $\mathbf{P}r_{x \sim \mathcal{U}_n}[g(x) = 1] \leq (1/\gamma) \cdot \mathbf{P}r_{x \sim \mathcal{U}_n}[f = 1]$.

(While condition (b) above appears slightly different than property (b) in our Definition 80, because of property (a), the two statements are essentially equivalent up to a factor of $1/(1-\epsilon)$ in the value of γ.)

We start by noting that it is easy to handle the case that $\widehat{p}$ is large. In particular, observe that if $\widehat{p} \geq 2\gamma$ then $p = \mathbf{P}r_{x \sim \mathcal{U}_n}[f(x) = 1] \geq \widehat{p}/(1+\epsilon) \geq \widehat{p}/2 \geq \gamma$, and we can just output $g \equiv 1$ since it clearly satisfies both properties of the definition. For the following intuitive discussion we will henceforth assume that $\widehat{p} \leq 2\gamma$.

Recall that our desired function g is an LTF, i.e., $g(x) = \text{sign}(v \cdot x - t)$, for some $(v,t) \in \mathbb{R}^{n+1}$. Recall also that our densifier has sample access to $\mathcal{U}_{f^{-1}(1)}$, so it can obtain random positive examples of f, each of which gives a linear constraint over the v, t variables. Hence a natural first approach is to attempt to construct an appropriate linear program over these variables whose feasible solutions satisfy conditions (a) and (b) above. We begin by analyzing this approach; while it turns out to not quite work, it will gives us valuable intuition for our actual algorithm, which is presented further below.

Note that following this approach, condition (a) is relatively easy to satisfy. Indeed, consider any $\epsilon > 0$ and suppose we want to construct an LTF $g = \text{sign}(v \cdot x - t)$ such that $\mathbf{P}r_{x \sim \mathcal{U}_{f^{-1}(1)}}[g(x) = 1] \geq 1 - \epsilon$. This can be done as follows: draw a set S_+ of $N_+ = \Theta\left((1/\epsilon) \cdot (n^2 + \log(1/\delta))\right)$ samples from $\mathcal{U}_{f^{-1}(1)}$ and consider a linear program $\mathcal{LP}_+$ with variables $(w, \theta) \in \mathbb{R}^{n+1}$ that enforces all these examples to be positive. That is, for each $x \in S_+$, we will have an inequality $w \cdot x \geq \theta$. It is clear that $\mathcal{LP}_+$ is feasible (any weights–based representation for f is a feasible solution) and that it can be solved in $\text{poly}(n, 1/\epsilon, \log(1/\delta))$ time, since it is defined by $O(N_+)$ many linear constraints and the coefficients of the constraint matrix are in $\{\pm 1\}$. The following simple claim shows that with high probability any feasible solution of $\mathcal{LP}_+$ satisfies condition (a):

Claim 103. *With probability at least* $1-\delta$ *over the sample* S_+*,* any $g \in \mathbf{LTF}_n$ *consistent with* S_+ *satisfies condition (a).*

Proof. Consider an LTF g and suppose that it does not satisfy condition (a), i.e., $\mathbf{P}r_{x \sim \mathcal{U}_n}[g(x) = -1 | f(x) = 1] > \epsilon$. Since each sample $x \in S_+$ is uniformly distributed in $f^{-1}(1)$, the probability it does not "hit" the set $g^{-1}(-1) \cap f^{-1}(1)$ is at most $1-\epsilon$. The probability that *no* sample in S_+ hits $g^{-1}(-1) \cap f^{-1}(1)$ is thus at most $(1-\epsilon)^{N_+} \leq \delta/2^{n^2}$. Recalling that there exist at most 2^{n^2} distinct LTFs over $\{-1,1\}^n$ [114], it follows by a union bound that the probability there exists an LTF that does not satisfy condition (a) is at most δ as desired. □

The above claim directly implies that with high probability *any* feasible solution (w^*, θ^*) to $\mathcal{LP}_+$ is such that $g^*(x) = \text{sign}(w^* \cdot x - \theta^*)$ satisfies condition (a). Of course, an arbitrary feasible solution to $\mathcal{LP}_+$ is by no means guaranteed to satisfy condition (b). (Note for example that the constant 1 function is certainly feasible for $\mathcal{LP}_+$.) Hence, a natural idea is to include additional constraints in our linear program so that condition (b) is also satisfied.

Along these lines, consider the following procedure: Draw a set $S_-$of $N_- = \lfloor \delta/\widehat{p} \rfloor$ uniform unlabeled samples from $\{-1,1\}^n$ and label them negative. That is, for each sample $x \in S_-$, we add the constraint $w \cdot x < \theta$ to our linear program. Let $\mathcal{LP}$ be the linear program that contains all the constraints defined by $S_+ \cup S_-$. It is not hard to prove that with probability at least $1-2\delta$ over the sample S_-, we have that $S_- \subseteq f^{-1}(-1)$ and hence (any weight based representation of) f is a feasible solution to $\mathcal{LP}$. In fact, it is possible to show that *if* γ *is sufficiently small* — roughly, $\gamma \leq \delta/\left(4(n^2 + \log(1/\delta))\right)$ is what is required — then with high probability each solution to $\mathcal{LP}$ also satisfies condition (b). The catch, of course, is that the above procedure is not computationally efficient because N_- may be very large – if $\widehat{p}$ is very small, then it is infeasible even to write down the linear program $\mathcal{LP}$.

Algorithm Description. The above discussion motivates our actual densifier algorithm as follows: The problem with the above described naive approach is that it generates (the potentially very large set) S_- all at once at the beginning of the algorithm. Note that having a large set S_- is not necessarily in and of itself a problem, since one could potentially use the ellipsoid method to solve $\mathcal{LP}$ if one could obtain an efficient separation oracle. Thus intuitively, if one had an online algorithm which would generate S_- *on the fly*, then one could potentially get a feasible solution to $\mathcal{LP}$ in polynomial time. This serves as the intuition behind our actual algorithm.

More concretely, our densifier $\mathcal{A}_{\text{den}}^{\mathbf{LTF}}$ will invoke a computationally efficient *online* learning algorithm for LTFs. In particular, $\mathcal{A}_{\text{den}}^{\mathbf{LTF}}$ will run the online learner $\mathcal{A}_{\text{MT}}^{\mathbf{LTF}}$ for a sequence of stages and in each stage it will provide as counterexamples to $\mathcal{A}_{\text{MT}}^{\mathbf{LTF}}$, random labeled examples from a carefully chosen distribution. These examples will be positive for the online learner's current hypothesis, but negative for f (with high probability). Since $\mathcal{A}_{\text{MT}}^{\mathbf{LTF}}$ makes a small number of mistakes in the worst-case, this process is guaranteed to terminate after a small number of stages (since in each stage we *force* the online learner to make a mistake).

We now provide the details. We start by recalling the notion of *online learning* for a class $\mathcal{C}$ of Boolean functions. In the online model, learning proceeds in a sequence of stages. In each stage the learning algorithm is given an unlabeled example $x \in \{-1,1\}^n$ and is asked to predict the value $f(x)$, where $f \in \mathcal{C}$ is the unknown target concept. After the learning algorithm makes its prediction, it is given the correct value of $f(x)$. The goal of the learner is to identify f while minimizing the total number of mistakes. We say that an online algorithm learns class $\mathcal{C}$ with mistake bound M if it makes at most M mistakes on *any* sequence of examples consistent with some $f \in \mathcal{C}$. Our densifier makes essential use of a computationally efficient online learning algorithm for the class of linear threshold functions by Maass and Turan [108]:

Theorem 104. *([108], Theorem 3.3) There exists a* $\mathrm{poly}(n)$ *time deterministic online learning algorithm* $\mathcal{A}_{\mathrm{MT}}^{\mathbf{LTF}}$ *for the class* $\mathbf{LTF}_n$ *with mistake bound* $M(n) \overset{def}{=} \Theta(n^2 \log n)$. *In particular, at every stage of its execution, the current hypothesis maintained by* $\mathcal{A}_{\mathrm{MT}}^{\mathbf{LTF}}$ *is a (weights–based representation of an) LTF that is consistent with all labeled examples received so far.*

We note that the above algorithm works by reducing the problem of online learning for LTFs to a convex optimization problem. Hence, one can use any efficient convex optimization algorithm to do online learning for LTFs, e.g. the ellipsoid method [92, 61]. The mistake bound in the above theorem follows by plugging in the algorithm of Vaidya [141, 142].

We now proceed with a more detailed description of our densifier followed by pseudocode and a proof of correctness. As previously mentioned, the basic idea is to execute the online learner to learn f while cleverly providing counterexamples to it in each stage of its execution. Our algorithm starts by sampling N_+ samples from $\mathcal{U}_{f^{-1}(1)}$ and making sure that these are classified correctly by the online learner. This step guarantees that our final solution will satisfy condition (a) of the densifier. Let $h \in \mathbf{LTF}_n$ be the current hypothesis at the end of this process. If h satisfies condition (b) (we can efficiently decide this by using our approximate counter for $\mathbf{LTF}_n$), we output h and terminate the algorithm. Otherwise, we use our approximate uniform generator to construct a uniform satisfying assignment $x \in \mathcal{U}_{h^{-1}(1)}$ and we label it negative, i.e., we give the labeled example $(x, -1)$ as a counterexample to the online learner. Since h does not satisfy condition (b), i.e., it has “many” satisfying assignments, it follows that with high probability (roughly, at least $1-\gamma$) over the choice of $x \in \mathcal{U}_{h^{-1}(1)}$, the point x output by the generator will indeed be negative for f. We continue this process for a number of stages. If all counterexamples thus generated are indeed consistent with f (this happens with probability roughly $1 - \gamma \cdot M$, where $M = M(n) = \Theta(n^2 \log n)$ is an upper bound on the number of stages), after at most M stages we have either found a hypothesis h satisfying condition (b) or the online learner terminates. In the latter case, the current hypothesis of the online learner is identical to f, as follows from Theorem 104. (Note that the above argument puts an upper bound of $O(\delta/M)$ on the value of γ.) Detailed pseudocode follows:

Algorithm $\mathcal{A}_{\mathrm{den}}^{\mathbf{LTF}}(\mathcal{U}_{f^{-1}(1)}, \epsilon, \delta, \widehat{p})$:
Input: Independent samples from $\mathcal{U}_{f^{-1}(1)}$, parameters $\epsilon, \delta > 0$, and a value $1/2^n \leq \widehat{p} \leq 1$.
Output: If $p \leq \widehat{p} \leq (1+\epsilon)p$, with probability $1-\delta$ outputs a function $g \in \mathbf{LTF}_n$ satisfying conditions (a) and (b).

```
1. Draw a set $S_+$ of $N_+ = \Theta\left((1/\epsilon) \cdot (n^2 + \log(1/\delta))\right)$ examples from $\mathcal{U}_{f^{-1}(1)}$.

2. Initialize $i = 0$ and set $M \stackrel{\text{def}}{=} \Theta(n^2 \log n)$.
   While $(i \le M)$ do the following:

   a) Execute the $i$-th stage of $A^{\mathbf{LTF}}_{\mathrm{MT}}$ and let $h^{(i)} \in \mathbf{LTF}_n$ be its current hypothesis.
   b) If there exists $x \in S_+$ with $h^{(i)}(x) = -1$ do the following:
      • Give the labeled example $(x, 1)$ as a counterexample to $A^{\mathbf{LTF}}_{\mathrm{MT}}$.
      • Set $i = i + 1$ and go to Step 2.
   c) Run $\mathcal{A}^{\mathbf{LTF}}_{\mathrm{count}}(h^{(i)}, \epsilon, \delta/(4M))$ and let $\widehat{p_i}$ be its output.
   d) Set $\gamma \stackrel{\text{def}}{=} \delta/(16M)$. If $\widehat{p_i} \le \widehat{p}/\big(\gamma \cdot (1+\epsilon)^2\big)$ then output $h^{(i)}$;
   e) otherwise, do the following:
      • Run $\mathcal{A}^{\mathbf{LTF}}_{\mathrm{gen}}(h^{(i)}, \delta/(4M))$ and let $x^{(i)}$ be its output.
      • Give the point $(x^{(i)}, -1)$ as a counterexample to $A^{\mathbf{LTF}}_{\mathrm{MT}}$.
      • Set $i = i + 1$ and go to Step 2.

3. Output the current hypothesis $h^{(i)}$ of $A^{\mathbf{LTF}}_{\mathrm{MT}}$.
```

Theorem 105. *Algorithm $\mathcal{A}^{\mathbf{LTF}}_{\mathrm{den}}(\mathcal{U}_{f^{-1}(1)}, \epsilon, \delta, \widehat{p})$ runs in time* $\mathrm{poly}\,(n, 1/\epsilon, \log(1/\delta))$. *If $p \le \widehat{p} < (1+\epsilon)p$ then with probability $1-\delta$ it outputs a vector (w, θ) such that $g(x) = \mathrm{sign}(w \cdot x - \theta)$ satisfies conditions (a) and (b) at the start of Section 4.3.*

Proof. First note that by Claim 103, with probability at least $1 - \delta/4$ over S_+ any LTF consistent with S_+ will satisfy condition (a). We will condition on this event and also on the event that each call to the approximate counting algorithm and to the approximate uniform generator is successful. Since Step 2 involves at most M iterations, by a union bound, with probability at least $1-\delta/4$ all calls to $\mathcal{A}^{\mathbf{LTF}}_{\mathrm{count}}$ will be successful, i.e., for all i we will have that $p_i/(1+\epsilon) \le \widehat{p_i} \le (1+\epsilon) \cdot p_i$, where $p_i = \mathbf{P}r_{x \in \mathcal{U}_n}[h^{(i)}(x) = 1]$. Similarly, with failure probability at most $\delta/4$, all points $x^{(i)}$ constructed by $\mathcal{A}^{\mathbf{LTF}}_{\mathrm{gen}}$ will be uniformly random over $(h^{(i)})^{-1}(1)$. Hence, with failure probability at most $3\delta/4$ all three conditions will be satisfied.

Conditioning on the above events, if the algorithm outputs a hypothesis $h^{(i)}$ in Step 2(d), this hypothesis will certainly satisfy condition (b), since $p_i \le (1+\epsilon)\widehat{p_i} \le \widehat{p}/\big(\gamma \cdot (1+\epsilon)\big) \le p/\gamma$. In this case, the algorithm succeeds with probability at least $1 - 3\delta/4$. It remains to show that if the algorithm returns a hypothesis in Step 3, it will be successful with probability at least $1-\delta$. To see this, observe that if no execution of Step 2(e) generates a point $x^{(i)}$ with $f(x^{(i)}) = 1$, all the counterexamples given to $A^{\mathbf{LTF}}_{\mathrm{MT}}$ are consistent with f. Therefore, by Theorem 104, the hypothesis of Step 3 will be identical to f, which trivially satisfies both conditions.

We claim that with overall probability at least $1 - \delta/4$ all executions of Step 2(e) generate points $x^{(i)}$ with $f(x^{(i)}) = -1$. Indeed, fix an execution of Step 2(e). Since $\widehat{p_i} > \widehat{p}/\left((1+\epsilon)^2 \cdot \gamma\right)$, it follows that $p \leq (4\gamma)p_i$. Hence, with probability at least $1 - 4\gamma$ a uniform point $x^{(i)} \sim \mathcal{U}_{(h^i)^{-1}(1)}$ is a negative example for f, i.e., $x^{(i)} \in f^{-1}(-1)$. By a union bound over all stages, our claim holds except with failure probability $4\gamma \cdot M = \delta/4$, as desired. This completes the proof of correctness.

It remains to analyze the running time. Note that Step 2 is repeated at most $M = O(n^2 \log n)$ times. Each iteration involves (i) one round of the online learner $\mathcal{A}^{\mathbf{LTF}}_{\mathrm{MT}}$ (this takes $\mathrm{poly}(n)$ time by Theorem 104), (ii) one call of $\mathcal{A}^{\mathbf{LTF}}_{\mathrm{count}}$ (this takes $\mathrm{poly}(n, 1/\epsilon, \log(1/\delta))$ time by Theorem 100), and (iii) one call to $\mathcal{A}^{\mathbf{LTF}}_{\mathrm{gen}}$ (this takes $\mathrm{poly}(n, 1/\epsilon, \log(1/\delta))$ time by Theorem 99). This completes the proof of Theorem 105. □

Discussion. As mentioned before, the algorithm $\mathcal{A}^{\mathbf{LTF}}_{\mathrm{den}}$ is the most important technical contribution of this section and hence it is instructive to understand, at a high level, the ingredients which are combined to construct a densifier. Let $\mathcal{C}$ be a class of Boolean functions and $\mathcal{C}_n$ consist of functions in $\mathcal{C}$ over n variables. While we do not formalize it here, one can modify the proof of Theorem 105, *mutatis mutandis*, to show that $\mathcal{C}_n$ has a $(\epsilon, \gamma, \delta)$-densifier $\mathcal{A}^{\mathcal{C}}_{\mathrm{den}}$ with running time $T(n)$ (where ϵ, γ and δ are $1/\mathrm{poly}(n)$ and $T(n)$ is $\mathrm{poly}(n)$) provided that the following conditions hold:

(i) $\mathcal{C}_n$ has an (ϵ, δ)-approximate counting algorithm and an (ϵ, δ)-approximate uniform generation algorithm both of which run in time $\mathrm{poly}(n, 1/\epsilon, 1/\delta)$;

(ii) There is an online learning algorithm $\mathcal{A}^{\mathcal{C}}_{OL}$ for $\mathcal{C}$ with a $\mathrm{poly}(n)$ running time and $\mathrm{poly}(n)$ mistake bound.

It will be interesting to see if there are other interesting classes of functions for which this framework gives an "automatic" construction of densifiers.

4.4 DNFs

In this section we apply our general positive result, Theorem 81, to give a quasipolynomial-time algorithm for the inverse approximate uniform generation problem for s-term DNF formulas. Let $\mathbf{DNF}_{n,s}$ denote the class of all s-term DNF formulas over n Boolean variables (which for convenience we think of as $0/1$ variables). Our main result of this section is the following:

Theorem 106. *There is an algorithm $\mathcal{A}^{\mathbf{DNF}_{n,s}}_{\mathrm{inv}}$ which is an inverse approximate uniform generation algorithm for the class $\mathbf{DNF}_{n,s}$. Given input parameters ϵ, δ the algorithm runs in time* $\mathrm{poly}\left(n^{\log(s/\epsilon)}, \log(1/\delta)\right)$.

We note that even in the standard uniform distribution learning model the fastest known running time for learning s-term DNF formulas to accuracy ϵ is $\mathrm{poly}(n^{\log(s/\epsilon)}, \log(1/\delta))$ [146, 143]. Thus it seems likely that obtaining a $\mathrm{poly}(n, s, 1/\epsilon)$-time algorithm would require a significant breakthrough in computational learning theory.

For our application of Theorem 81 for DNFs we shall have $\mathcal{C} = \mathbf{DNF}_{n,s}$ and $\mathcal{C}' = \mathbf{DNF}_{n,t}$ for some t which we shall specify later. As in the case of LTFs, the literature provides us with three of the four ingredients that our general approach requires for DNF — approximate uniform generation, approximate counting, and Statistical Query learning (more on this below) — and our main technical contribution is giving the fourth necessary ingredient, a densifier. Before presenting and analyzing our densifier algorithm we recall the other three ingredients.

Tools from the literature.

Karp, Luby and Madras [81] have given approximate uniform generation and approximate counting algorithms for DNF formulas. (We note that [72] give an efficient algorithm that with high probability outputs an *exactly* uniform satisfying assignment for DNFs.)

Theorem 107. *(Approximate uniform generation for DNFs, [81]) There is an approximate uniform generation algorithm* $\mathcal{A}_{\text{gen}}^{\mathbf{DNF}_{n,t}}$ *for the class* $\mathbf{DNF}_{n,t}$ *that runs in time* $\text{poly}(n, t, 1/\epsilon, \log(1/\delta))$.

Theorem 108. *(Approximate counting for DNFs, [81]) There is an approximate counting algorithm* $\mathcal{A}_{\text{gen}}^{\mathbf{DNF}_{n,t}}$ *for the class* $\mathbf{DNF}_{n,t}$ *that runs in time* $\text{poly}(n, t, 1/\epsilon, \log(1/\delta))$.

The fastest known algorithm in the literature for SQ learning s-term DNF formulas under arbitrary distributions runs in time $n^{O(n^{1/3}\log s)} \cdot \text{poly}(1/\epsilon)$ [95], which is much more than our desired running time bound. However, we will see that we are able to use known *malicious noise tolerant* SQ learning algorithms for learning *sparse disjunctions* over N Boolean variables rather than DNF formulas. In more detail, our densifier will provide us with a set of $N = n^{O(\log(s/\epsilon))}$ many conjunctions which is such that the target function f is very close to a disjunction (which we call f') over an unknown subset of at most s of these N conjunctions. Thus intuitively any learning algorithm for disjunctions, run over the "feature space" of conjunctions provided by the densifier, would succeed if the target function were f', but the target function is actually f (which is not necessarily exactly a disjunction over these N variables). Fortunately, known results on the malicious noise tolerance of specific SQ learning algorithms imply that it is in fact possible to use these SQ algorithms to learn f to high accuracy, as we now explain.

We now state the precise SQ learning result that we will use. The following theorem is a direct consequence of, e.g., Theorems 5 and 6 of [33] or alteratively of Theorems 5 and 6 of [5]:

Theorem 109. *(Malicious noise tolerant* SQ *algorithm for learning sparse disjunctions) Let* $\mathcal{C}_{\mathbf{DISJ},k}$ *be the class of all disjunctions of length at most k over N Boolean variables* $x_1, \dots, x_N$. *There is a distribution-independent* SQ *learning algorithm* $\mathcal{A}_{\text{SQ}}^{\mathbf{DISJ}}$ *for* $\mathcal{C}_{\mathbf{DISJ},k}$ *that has running time* $t_1 = \text{poly}(N, 1/\epsilon, \log(1/\delta))$, *uses at most* $t_2 = \text{poly}(N)$ *time to evaluate each query, and requires tolerance of its queries no smaller than* $\tau = 1/\text{poly}(k, 1/\epsilon)$. *The algorithm outputs a hypothesis which is a disjunction over* $x_1, \dots, x_N$.

Moreover, there is a fixed polynomial $\ell(\cdot)$ *such that algorithm* $\mathcal{A}_{\text{SQ}}^{\mathbf{DISJ}}$ *has the following property: Fix a distribution D over* $\{0,1\}^N$. *Let f be an N-variable Boolean function which is such that* $\mathbf{P}r_{x\sim D}[f'(x) \neq f(x)] \leq \kappa$, *where* $f' \in \mathcal{C}_{\mathbf{DISJ},k}$ *is some k-variable disjunction and*

$\kappa \leq \ell(\epsilon/k) < \epsilon/2$. Then if $\mathcal{A}_{\text{SQ}}^{\mathbf{DISJ}}$ is run with a $\text{STAT}(f, D)$ oracle, with probability $1-\delta$ it outputs a hypothesis h such that $\mathbf{P}r_{x\sim D}[h(x) \neq f'(x)] \leq \epsilon/2$, and hence $\mathbf{P}r_{x\sim D}[h(x \neq f(x)] \leq \epsilon$.

(We note in passing that at the heart of Theorem 109 is an *attribute-efficient* SQ algorithm for learning sparse disjunctions. Very roughly speaking, an attribute efficient SQ learning algorithm is one which can learn a target function over N variables, which actually depends only on an unknown subset of $k \ll N$ of the variables, using statistical queries for which the minimum value of the tolerance τ is "large." The intuition behind Theorem 109 is that since the distance between f and f' is much less than τ, the effect of using a $\text{STAT}(f, \mathcal{D})$ oracle rather than a $\text{STAT}(f', \mathcal{D})$ oracle is negligible, and hence the SQ algorithm will succeed whether it is run with f or f' as the target function.)

A densifier for $\text{DNF}_{n,s}$ and the proof of Theorem 106.

In this subsection we state our main theorem regarding the existence of densifiers for DNF formulas, Theorem 110, and show how Theorem 106 follows from this theorem.

Theorem 110. *Let $\gamma(n, s, 1/\epsilon, 1/\delta) = 1/(4n^{2\log(2s/\ell(\epsilon/s))}\log(s/\delta))$. Algorithm $\mathcal{A}_{\text{den}}^{\mathbf{DNF}_{n,s}}(\mathcal{U}_{f^{-1}(1)}, \epsilon, \delta, \widehat{p})$ outputs a collection $\mathcal{S}$ of conjunctions $C_1, \ldots, C_{|\mathcal{S}|}$ and has the following performance guarantee: If $p \overset{\text{def}}{=} \mathbf{P}r_{x\sim\mathcal{U}_n}[f(x) = 1] \leq \widehat{p} < (1+\epsilon)p$, then with probability at least $1-\delta$, the function $g(x) \overset{\text{def}}{=} \vee_{i\in[|\mathcal{S}|]} C_i$ satisfies the following:*

1. *$\mathbf{P}r_{x\sim\mathcal{U}_{f^{-1}(1)}}[g(x) = 1] \geq 1-\epsilon$;*
2. *$\mathbf{P}r_{x\sim\mathcal{U}_{g^{-1}(1)}}[f(x) = 1] \geq \gamma(n, s, 1/\epsilon, 1/\delta)$.*
3. *There is a DNF $f' = C_{i_1} \vee \cdots \vee C_{i_{s'}}$, which is a disjunction of $s' \leq s$ of the conjunctions $C_1, \ldots, C_{|\mathcal{S}|}$, such that $\mathbf{P}r_{x\sim\mathcal{U}_{g^{-1}(1)}}[f'(x) \neq f(x)] \leq \ell(\epsilon/s)$, where $\ell(\cdot)$ is the polynomial from Theorem 109.*

The size of $\mathcal{S}$ and the running time of $\mathcal{A}_{\text{den}}^{\mathbf{DNF}_{n,s}}(\mathcal{U}_{f^{-1}(1)}, \epsilon, \delta, \widehat{p})$ is $\text{poly}(n^{\log(s/\epsilon)}, \log(1/\delta))$.

With a slight abuse of terminology we may rephrase the above theorem as saying that $\mathcal{A}_{\text{den}}^{\mathbf{DNF}_{n,s}}$ is a $(\epsilon, \gamma, \delta)$-densifier for function class $\mathcal{C} = \mathbf{DNF}_{n,s}$ using class $\mathcal{C}' = \mathbf{DNF}_{n,t}$ where $t = n^{O(\log(s/\epsilon))}$. We defer the description of Algorithm $\mathcal{A}_{\text{den}}^{\mathbf{DNF}_{n,s}}$ and the proof of Theorem 110 to the next subsection.

Proof of Theorem 106. The proof is essentially just an application of Theorem 81. The only twist is the use of a SQ disjunction learning algorithm rather than a DNF learning algorithm, but the special properties of Algorithm $\mathcal{A}_{\text{SQ}}^{\mathbf{DISJ}}$ let this go through without a problem.

In more detail, in Step 2(e) of Algorithm $\mathcal{A}_{\text{inv}}'^{\mathcal{C}}$ (see Section 4.2), in the execution of Algorithm $\mathcal{A}_{\text{SQ-SIM}}$, the SQ algorithm that is simulated is the algorithm $\mathcal{A}_{\text{SQ}}^{\mathbf{DISJ}}$ run over the feature space $\mathcal{S}$ of all conjunctions that are output by Algorithm $\mathcal{A}_{\text{den}}^{\mathbf{DNF}_{n,s}}$ in Step 1 of Algorithm $\mathcal{A}_{\text{inv}}'^{\mathcal{C}}$

(i.e., these conjunctions play the role of variables $x_1, \dots, x_N$ for the SQ learning algorithm). Property (3) of Theorem 110 and Theorem 109 together imply that the algorithm $\mathcal{A}_{\text{SQ}}^{\mathbf{DISJ}}$, run on a $\text{STAT}(f, \mathcal{U}_{g^{-1}(1)})$ oracle with parameters ϵ, δ, would with probability $1-\delta$ output a hypothesis h' satisfying $\mathbf{Pr}_{x \sim \mathcal{U}_{g^{-1}(1)}}[h'(x) \neq f(x)] \leq \epsilon$. Hence the hypothesis h that is output by $\mathcal{A}_{\text{SQ}-\text{SIM}}$ in Step 2(e) of Algorithm $\mathcal{A}'^{\mathcal{C}}_{\text{inv}}$ fulfills the necessary accuracy (with respect to f under $D = \mathcal{U}_{g^{-1}(1)}$) and confidence requirements, and the overall algorithm $A^{\mathcal{C}}_{\text{inv}}$ succeeds as described in Theorem 81.

Finally, combining the running time bounds of $\mathcal{A}_{\text{den}}^{\mathbf{DNF}_{n,s}}$ and $\mathcal{A}_{\text{SQ}}^{\mathbf{DISJ}}$ with the time bounds of the other procedures described earlier, one can straightforwardly verify that the running time of the overall algorithm $\mathcal{A}^{\mathcal{C}}_{\text{inv}}$ is $\text{poly}(n^{\log(s/\epsilon)}, \log(1/\delta))$. □

Construction of a densifier for $\text{DNF}_{n,s}$ and proof of Theorem 110.

Let $f = T_1 \vee \dots \vee T_s$ be the target s-term DNF formula, where $T_1, \dots, T_s$ are the terms (conjunctions). The high-level idea of our densifier is quite simple: If T_i is a term which is "reasonably likely" to be satisfied by a uniform draw of x from $f^{-1}(1)$, then T_i is at least "mildly likely" to be satisfied by $r = 2\log n$ consecutive independent draws of x from $f^{-1}(1)$. Such a sequence of draws $x^1, \dots, x^r$ will with high probability *uniquely identify* T_i. By repeating this process sufficiently many times, with high probability we will obtain a pool $C_1, \dots, C_{|\mathcal{S}|}$ of conjunctions which contains all of the terms T_i that are reasonably likely to be satisfied by a uniform draw of x from $f^{-1}(1)$. Theorem 110 follows straightforwardly from this.

We give detailed pseudocode for our densifier algorithm below:

```
Algorithm $\mathcal{A}_{\text{den}}^{\mathbf{DNF}_{n,s}}(\mathcal{U}_{f^{-1}(1)}, \epsilon, \delta, \widehat{p})$:
Input: Independent samples from $\mathcal{U}_{f^{-1}(1)}$, parameters $\epsilon, \delta > 0$, and a value $1/2^n < \widehat{p} \leq 1$.
Output: If $p \leq \widehat{p} \leq (1+\epsilon)p$, with probability $1-\delta$ outputs a set $\mathcal{S}$ of conjunctions $C_1, \dots, C_{|\mathcal{S}|}$
as described in Theorem 110

  1. Initialize set $\mathcal{S}$ to $\emptyset$. Let $\ell(\cdot)$ be the polynomial from Theorem 109.
  2. For $i = 1$ to $M = 2n^{2\log(2s/\ell(\epsilon/s))}\log(s/\delta)$, repeat the following:
      a) Draw $r = 2\log n$ satisfying assignments $x^1, \dots, x^r$ from $\mathcal{U}_{f^{-1}(1)}$.
      b) Let $C_i$ be the AND of all literals that take the same value in all $r$ strings $x^1, \dots, x^r$
         (note $C_i$ may be the empty conjunction). We say $C_i$ is a candidate term.
      c) If the candidate term $C_i$ satisfies $\mathbf{Pr}_{x \sim \mathcal{U}_n}[C_i(x) = 1] \leq \widehat{p}$ then add $C_i$ to the set $\mathcal{S}$.
  3. Output $\mathcal{S}$.
```

The following crucial claim makes the intuition presented at the start of this subsection precise:

Claim 111. *Suppose T_j is a term in f such that $\mathbf{Pr}_{x\sim\mathcal{U}_{f^{-1}(1)}}[T_j(x) = 1] \geq \ell(\epsilon/s)/(2s)$. Then with probability at least $1-\delta/s$, term T_j is a candidate term at some iteration of Step 2 of Algorithm $\mathcal{A}_{\text{den}}^{\mathbf{DNF}_{n,s}}(\mathcal{U}_{f^{-1}(1)}, \epsilon, \delta, \widehat{p})$.*

Proof. Fix a given iteration i of the loop in Step 2. With probability at least

$$(\ell(\epsilon/s)/(2s))^{2\log n} = (1/n)^{2\log(2s/\ell(\epsilon/s))},$$

all $2\log n$ points $x^1, \ldots, x^{2\log n}$ satisfy T_j; let us call this event E, and condition on E taking place. We claim that conditioned on E, the points $x^1, \ldots, x^{2\log n}$ are independent uniform samples drawn from $T_j^{-1}(1)$. (To see this, observe that each x^i is an independent sample chosen uniformly at random from $f^{-1}(1) \cap T_j^{-1}$; but $f^{-1}(1) \cap T_j^{-1}(1)$ is identical to $T_j^{-1}(1)$.) Given that $x^1, \ldots, x^{2\log n}$ are independent uniform samples drawn from $T_j^{-1}(1)$, the probability that any literal which is *not* present in T_j is contained in C_i (i.e., is satisfied by all $2\log n$ points) is at most $2n/n^2 \leq 1/2$. So with overall probability at least $\frac{1}{2n^{2\log(2s/\ell(\epsilon/s))}}$, the term T_j is a candidate term at iteration i. Consequently T_j is a candidate term at some iteration with probability at least $1-\delta/s$, by the choice of $M = 2n^{2\log(2s/\ell(\epsilon/s))}\log(s/\delta)$. □

Now we are ready to prove Theorem 110:

Proof of Theorem 110. The claimed running time bound of $\mathcal{A}_{\text{den}}^{\mathbf{DNF}_{n,s}}$ is easily verified, so it remains only to establish (1)-(3). Fix $\widehat{p}$ such that $p \leq \widehat{p} < (1+\epsilon)p$ where $p = \mathbf{Pr}_{x\sim\mathcal{U}_n}[f(x) = 1]$.

Consider any fixed term T_j of f such that $\mathbf{Pr}_{x\sim\mathcal{U}_{f^{-1}(1)}}[T_j(x) = 1] \geq \ell(\epsilon/s)/(2s)$. By Claim 111 we have that with probability at least $1-\delta/s$, term T_j is a candidate term at some iteration of Step 2 of the algorithm. We claim that in step (c) of this iteration the term T_j will in fact be added to $\mathcal{S}$. This is because by assumption we have

$$\mathop{\mathbf{Pr}}_{x\sim\mathcal{U}_n}[T_j(x) = 1] \leq \mathop{\mathbf{Pr}}_{x\sim\mathcal{U}_n}[f(x) = 1] = p \leq \widehat{p}.$$

So by a union bound, with probability at least $1-\delta$ every term T_j in f such that $\mathbf{Pr}_{x\sim\mathcal{U}_{f^{-1}(1)}}[T_j(x) = 1] \geq \ell(\epsilon/s)/(2s)$ is added to $\mathcal{S}$.

Let L be the set of those terms T_j in f that have $\mathbf{Pr}_{x\sim\mathcal{U}_{f^{-1}(1)}}[T_j(x) = 1] \geq \ell(\epsilon/s)/(2s)$. Let f' be the DNF obtained by taking the OR of all terms in L. By a union bound over the (at most s) terms that are in f but not in f', we have $\mathbf{Pr}_{x\sim\mathcal{U}_{f^{-1}(1)}}[f'(x) = 1] \geq 1 - \ell(\epsilon/s)/2$. Since g (as defined in Theorem 110 has $g(x) = 1$ whenever $f'(x) = 1$, it follows that $\mathbf{Pr}_{x\sim\mathcal{U}_{f^{-1}(1)}}[g(x) = 1] \geq 1 - \ell(\epsilon/s)/2 \geq 1-\epsilon$, giving item (1) of the theorem.

For item (2), since $f(x) = 1$ whenever $f'(x) = 1$, we have $\mathbf{Pr}_{x\sim\mathcal{U}_{g^{-1}(1)}}[f(x) = 1] \geq \mathbf{Pr}_{x\sim\mathcal{U}_{g^{-1}(1)}}[f'(x) = 1]$. Every x such that $f'(x) = 1$ also has $g(x) = 1$ so to lower bound $\mathbf{Pr}_{x\sim\mathcal{U}_{g^{-1}(1)}}[f'(x) = 1]$ it is enough to upper bound the number of points in $g^{-1}(1)$ and lower bound the number of points in $f'^{-1}(1)$. Since each C_i that is added to $\mathcal{S}$ is satisfied by at most $\widehat{p}2^n \leq (1+\epsilon)p2^n$ points, we have that $|g^{-1}(1)| \leq (1+\epsilon)pM2^n$. Since at least $1-\epsilon$ of the points that satisfy f also satisfy f', we have that $|f'^{-1}(1)| \geq p(1-\epsilon)2^n$. Thus we have $\mathbf{Pr}_{x\sim\mathcal{U}_{g^{-1}(1)}}[f'(x) = 1] \geq p(1-\epsilon)/((1+\epsilon)pM) = \frac{1-\epsilon}{1+\epsilon} \cdot \frac{1}{M} > \frac{1}{2M}$, giving (2).

Finally, for (3) we have that $f(x) \neq f'(x)$ only on those inputs that have $f(x) = 1$ but $f'(x) = 0$ (because some term outside of L is satisfied by x and no term in L is satisfied by x). Even if all such inputs x lie in $g^{-1}(1)$ (the worst case), there can be at most $(\ell(\epsilon/s)/2)p2^n$ such inputs, and we know that $|g^{-1}(1)| \geq |f^{-1}(1)| \geq p(1-\epsilon)2^n$. So we have $\mathbf{P}r_{x \sim \mathcal{U}_{g^{-1}(1)}}[f(x) \neq f'(x)] \leq \frac{\ell(\epsilon/s)/2}{1-\epsilon} \leq \ell(\epsilon/s)$, and we have (3) as desired. $\square$

Inverse approximate uniform generation for k-DNFs.

We briefly note that our general approach immediately yields an efficient inverse approximate uniform generation algorithm for the class of k-DNFs for any constant k. Let k-$\mathbf{DNF}$ denote the class of all k-DNFs over n Boolean variables, i.e., DNF formulas in which each term (conjunction) has at most k literals.

Theorem 112. *There is an algorithm $\mathcal{A}_{\text{inv}}^{k\text{-}\mathbf{DNF}}$ which is an inverse approximate uniform generation algorithm for the class k-$\mathbf{DNF}$. Given input parameters ϵ, δ the algorithm runs in time* $\text{poly}\left(n^k, 1/\epsilon, \log(1/\delta)\right)$.

For any k-DNF f it is easy to see that $\mathbf{P}r_{x \sim \mathcal{U}_n}[f(x) = 1] \geq 1/2^k$, and consequently the constant 1 function is a γ-densifier for k-$\mathbf{DNF}$ with $\gamma = 1/2^k$. Theorem 112 then follows immediately from Theorem 81, using the algorithms for approximate uniform generation and counting of DNF formulas mentioned above [81] together with well-known algorithms for SQ learning k-DNF formulas in $\text{poly}(n^k, 1/\epsilon, \log(1/\delta))$ time [85].

4.5 Negative results for inverse approximate uniform generation

In this section, we will prove hardness results for inverse approximate uniform generation problems for specific classes $\mathcal{C}$ of Boolean functions. As is standard in computational learning theory, our hardness results are based on cryptographic hardness assumptions. The hardness assumptions we use are well studied assumptions in cryptography such as the strong RSA assumption, Decisional Diffie Hellman problem, and hardness of learning parity with noise.

As was alluded to in the introduction, in light of the standard approach, there are two potential barriers to obtaining inverse approximate uniform generation algorithms for a class $\mathcal{C}$ of functions. The first is that "reconstructing" the object from class $\mathcal{C}$ may be hard, and the second is that sampling approximately uniform random satisfying assignments from the reconstructed object may be hard. While any hard inverse approximate uniform generation problem must be hard because of one of these two potential barriers, we emphasize here that even if one of the two steps in the standard approach is shown to be hard, this does not constitute a proof of hardness of the overall inverse approximate uniform generation problem, as there is may exist some efficient algorithm for the class $\mathcal{C}$ which departs from the standard approach. Indeed, we will give such an example in Section 4.6, where we give an efficient algorithm for a specific inverse approximate uniform generation problem that does not follow the standard approach. (In fact, for that problem, the second step of the standard approach is provably no easier than the well-known graph automorphism problem, which has withstood several decades of effort towards even getting a sub-exponential time algorithm.)

Our hardness results come in two flavors. Our first hardness results, based on signature schemes, are for problems where it is provably hard (of course under a computational hardness assumption) to sample approximately uniform satisfying assignments. In contrast, our hardness results of the second flavor are based on Message Authentication Codes (MACs). We give such a result for a specific class $\mathcal{C}$ which has the property that it is actually easy to sample uniform satisfying assignments for functions in $\mathcal{C}$; hence, in an informal sense, it is the first step in the standard approach that is algorithmically hard for this problem. The following subsections describe all of our hardness results in detail.

Hardness results based on signature schemes.

In this subsection we prove a general theorem, Theorem 118, which relates the hardness of inverse approximate uniform generation to the existence of certain secure signature schemes in cryptography. Roughly speaking, Theorem 118 says that if secure signature schemes exist, then the inverse approximate uniform generation problem is computationally hard for any class $\mathcal{C}$ which is Levin-reducible from CIRCUIT-SAT. We will use this general result to establish hardness of inverse approximate uniform generation for several natural classes of functions, including 3-CNF formulas, intersections of two halfspaces, and degree-2 polynomial threshold functions (PTFs).

We begin by recalling the definition of public key signature schemes. For an extensive treatment of signature schemes, see [59]. For simplicity, and since it suffices for our purposes, we only consider schemes with deterministic verification algorithms.

Definition 113. *A signature scheme is a triple (G, S, V) of polynomial-time algorithms with the following properties :*

- **(Key generation algorithm)** *G is a randomized algorithm which on input 1^n produces a pair (pk, sk) (note that the sizes of both pk and sk are polynomial in n).*
- **(Signing algorithm)** *S is a randomized algorithm which takes as input a message m from the message space $\mathcal{M}$, a secret key sk and randomness $r \in \{0,1\}^n$, and outputs a signature $\sigma = S(m, sk, r)$.*
- **(Verification algorithm)** *V is a deterministic algorithm such that $V(m, pk, \sigma) = 1$ for every $\sigma = S(m, sk, r)$.*

We will require signature schemes with some special properties which we now define, first fixing some notation. Let (G, S, V) be a signature scheme. For a message space $\mathcal{M}$ and pair (pk, sk) of public and secret keys, we define the set $\mathcal{R}_{1,sk}$ of "valid" signed messages as the set of all possible signed messages $(m, \sigma = S(m, sk, r))$ as m ranges over all of $\mathcal{M}$ and r ranges over all of $\{0,1\}^n$. Similarly, we define the set $\mathcal{R}_{2,pk}$ of "potential" signed messages as $\mathcal{R}_{2,pk} = \{(m,\sigma) : V(m,pk,\sigma) = 1\}$. Likewise, we define the set of valid signatures for message m, denoted $\mathcal{R}_{1,sk}(m)$, as the set of all possible pairs $(m, \sigma = S(m,sk,r))$ as r ranges over all of $\{0,1\}^n$, and we define the set of potential signatures for message m as $\mathcal{R}_{2,pk}(m) = \{(m,\sigma) : V(m,pk,\sigma) = 1\}$.

Definition 114. *Let (G,S,V) be a signature scheme and $\mathcal{M}$ be a message space. A pair (pk,sk) of public and secret keys is said to be (δ,η)-*special *if the following properties hold :*

- *Let $\mathcal{R}_{1,sk}$ be the set of valid signed messages and $\mathcal{R}_{2,pk}$ be the set of potential signed messages. Then $\frac{|\mathcal{R}_{1,sk}|}{|\mathcal{R}_{2,pk}|} \geq 1-\eta$.*
- *For any fixed pair $(m,\sigma) \in \mathbb{R}_{1,sk}(m)$, we have $\mathbf{Pr}_{r\in\{0,1\}^n}[\sigma = S(m,sk,r)] = \frac{1}{|\mathcal{R}_{1,sk}(m)|}$.*
- *Define two distributions D and D' over pairs (m,σ) as follows : D is obtained by choosing $m \in_U \mathcal{M}$ and choosing $\sigma \in_U \mathcal{R}_{1,sk}(m)$. D' is the distribution defined to be uniform over the set $\mathcal{R}_{1,sk}$. Then $d_{TV}(D,D') \leq \delta$.*

From now on, in the interest of brevity, $\mathcal{M}$ will denote the "obvious" message space $\mathcal{M}$ associated with a signature scheme unless mentioned otherwise. Similarly, the randomness r for the signing algorithm S will always assumed to be $r \in_U \{0,1\}^n$.

We next recall the standard notion of existential unforgeability under RMA (Random Message Attack):

Definition 115. *A signature scheme (G,S,V) is said to be (t,ϵ)-*RMA secure *if the following holds: Let $(pk,sk) \leftarrow G(1^n)$. Let $(m_1,\dots,m_t)$ be chosen uniformly at random from $\mathcal{M}$. Let $\sigma_i \leftarrow S(m_i,sk,r)$. Then, for any probabilistic algorithm A running in time t,*

$$\mathop{\mathbf{Pr}}_{(pk,sk),(m_1,\dots,m_t),(\sigma_1,\dots,\sigma_t)}[A(pk,m_1,\dots,m_t,\sigma_1,\dots,\sigma_t) = (m',\sigma')] \leq \epsilon$$

where $V(m',pk,\sigma') = 1$ and $m' \neq m_i$ for all $i = 1,\dots,t$.

Next we need to formally define the notion of hardness of inverse approximate uniform generation:

Definition 116. *Let $\mathcal{C}$ be a class of n-variable Boolean functions. $\mathcal{C}$ is said to be $(t(n),\epsilon,\delta)$-*hard for inverse approximate uniform generation *if there is no algorithm A running in time $t(n)$ which is an (ϵ,δ)-inverse approximate uniform generation algorithm for $\mathcal{C}$.*

Finally, we will also need the definition of an invertible Levin reduction:

Definition 117. *A binary relation R is said to reduce to another binary relation R' by a* time-t invertible Levin reduction *if there are three algorithms α, β and γ, each running in time $t(n)$ on instances of length n, with the following property:*

- *For every $(x, y) \in R$, it holds that $(\alpha(x), \beta(x, y)) \in R'$;*
- *For every $(\alpha(x), z) \in R'$, it holds that $(x, \gamma(\alpha(x), z)) \in R$.*

Furthermore, the functions β and γ are injective maps with the property that $\gamma(\alpha(x), \beta(x, y)) = y$.

Note that for any class of functions $\mathcal{C}$, we can define the binary relation $R_{\mathcal{C}}$ as follows : $(f, x) \in R_{\mathcal{C}}$ if and only if $f(x) = 1$ and $f \in \mathcal{C}$. In this section, whenever we say that there is an invertible Levin reduction from class $\mathcal{C}_1$ to class $\mathcal{C}_2$, we mean that there is an invertible Levin reduction between the corresponding binary relations $R_{\mathcal{C}_1}$ and $R_{\mathcal{C}_2}$.

A general hardness result based on signature schemes.

We now state and prove our main theorem relating signature schemes to hardness of inverse approximate uniform generation:

Theorem 118. *Let (G, S, V) be a (t, ϵ)-RMA secure signature scheme. Suppose that with probability at least 99/100 a random pair $(pk, sk) \leftarrow G(1^n)$ is (δ, η)-special. Let $\mathcal{C}$ be a class of n-variable Boolean functions such that there is a Levin reduction from CIRCUIT-SAT to $\mathcal{C}$ running in time $t'(n)$. Let κ_1 and κ_2 be such that $\kappa_1 \leq 1 - 2 \cdot (2\eta + \delta + t'(n)/|\mathcal{M}|)$, $\kappa_2 \leq 1 - 2t'(n) \cdot (\eta + \delta)$ and $\epsilon \leq (1 - \kappa_1)(1 - \kappa_2)/4$. If $t_1(\cdot)$ is a time function such that $2t_1(t'(n)) \leq t(n)$, then $\mathcal{C}$ is $(t_1(n), \kappa_1, \kappa_2)$-hard for inverse approximate uniform generation.*

The high-level idea of the proof is simple: Suppose there were an efficient algorithm for the inverse approximate uniform generation problem for $\mathcal{C}$. Because of the invertible Levin reduction from CIRCUIT-SAT to $\mathcal{C}$, there is a signature scheme for which the verification algorithm (using any given public key) corresponds to a function in $\mathcal{C}$. The signed messages $(m_1, \sigma_1), \ldots, (m_t, \sigma_t)$ correspond to points from $\mathcal{U}_{f^{-1}(1)}$ where $f \in \mathcal{C}$. Now the existence of an efficient algorithm for the inverse approximate uniform generation problem for $\mathcal{C}$ (i.e. an algorithm which, given points from $\mathcal{U}_{f^{-1}(1)}$, can generate more such points) translates into an algorithm which, given a sample of signed messages, can generate a new signed message. But this violates the existential unforgeability under RMA of the signature scheme.

We now proceed to the formal proof.

Proof. Assume towards a contradiction that there is an algorithm A for inverse approximate uniform generation A_{inv} which runs in time t_1 such that with probability $1-\kappa_2$, the output distribution is κ_1-close to the target distribution. If we can show that for any (δ, η)-special key pair (pk, sk) the resulting signature scheme is not (t, ϵ) secure, then this will result in a contradiction. We will now use algorithm A to construct an adversary which breaks the signature scheme for (δ, η)-special key pairs (pk, sk).

Towards this, fix a (δ, η)-special key pair (pk, sk) and consider the function $V_{pk} : \mathcal{M} \times \{0, 1\}^* \rightarrow \{0, 1\}$ defined as $V_{pk}(m, \sigma) = V(m, pk, \sigma)$. Clearly, V_{pk} is an instance of CIRCUIT-SAT (i.e. V_{pk} is computed by a satisfiable polynomial-size Boolean circuit). Since there is an invertible Levin reduction from CIRCUIT-SAT to $\mathcal{C}$, given pk, the adversary in time $t'(n)$ can compute $\Phi_{pk} \in \mathcal{C}$ with the following properties (let β and γ be the corresponding algorithms in the definition of the Levin reduction):

- For every (m, σ) such that $V_{pk}(m, \sigma) = 1$, $\Phi_{pk}(\beta(V_{pk}, (m, \sigma))) = 1$.
- For every x such that $\Phi_{pk}(x) = 1$, $V_{pk}(\gamma(\Phi_{pk}, x)) = 1$.

Recall that the adversary receives signatures $(m_1, \sigma_1), \ldots, (m_{t'(n)}, \sigma_{t'(n)})$. Let $x_i = \beta(V_{pk}, (m_i, \sigma_i))$. Let D_x be the distribution of $(x_1, \ldots, x_{t'(n)})$. We next make the following claim.

Claim 119. *Let $y_1, \ldots, y_{t'}$ be drawn uniformly at random from $\Phi_{pk}^{-1}(1)$ and let D_y be the corresponding distribution of $(y_1, \ldots, y_t)$. Then, D_y and D_x are $t'(n) \cdot (2\eta + \delta)$-close in statistical distance.*

Proof. Note that D_y and D_x are $t'(n)$-way product distributions. If $D_x^{(1)}$ and $D_y^{(1)}$ are the corresponding marginals on the first coordinate, then $t'(n) \cdot d_{TV}(D_x^{(1)}, D_y^{(1)}) \leq d_{TV}(D_x, D_y)$. Thus, it suffices to upper bound $d_{TV}(D_x^{(1)}, D_y^{(1)})$, which we now do.

$$d_{TV}(D_x^{(1)}, D_y^{(1)}) \leq \sum_{z \in supp(D_y^{(1)}) \setminus supp(D_x^{(1)})} \left|D_x^{(1)}(z) - D_y^{(1)}(z)\right| + \sum_{z \in supp(D_x^{(1)})} \left|D_x^{(1)}(z) - D_y^{(1)}(z)\right|.$$

By definition of (pk, sk) being (δ, η)-special, we get that

$$\sum_{z \in supp(D_y^{(1)}) \setminus supp(D_x^{(1)})} |D_x^{(1)}(z) - D_y^{(1)}(z)| \leq \eta.$$

To bound the next sum, let $\tau = \mathbf{P}r[D_y^{(1)} \in supp(D_x^{(1)})]$. Note that $\tau \geq 1 - \eta$. We have

$$\begin{aligned}\sum_{z \in supp(D_x^{(1)})} \left|D_x^{(1)}(z) - D_y^{(1)}(z)\right| &\leq \sum_{z \in supp(D_x^{(1)})} \left|\tau D_x^{(1)}(z) - D_y^{(1)}(z)\right| + (1-\tau) \sum_{z \in supp(D_x^{(1)})} D_x^{(1)}(z) \\ &\leq \eta + \tau \cdot \sum_{z \in supp(D_x^{(1)})} \left|D_x^{(1)}(z) - \frac{D_y^{(1)}(z)}{\tau}\right|.\end{aligned}$$

We observe that $\frac{D_y^{(1)}(z)}{\tau}$ restricted to $supp(D_x^{(1)})$ is simply the uniform distribution over the image of the set $\mathcal{R}_{1,sk}$ and hence is the same as applying the map β on the distribution D'. Likewise $D_x^{(1)}$ is the same as applying the map β on D (mentioned in Definition 114). Hence, we have that

$$d_{TV}(D_x^{(1)}, D_y^{(1)}) \leq 2\eta + d_{TV}(D, D') \leq 2\eta + \delta.$$

$\square$

Now, observe that the instances x_i are each of length at most $t'(n)$. Since the distributions D_x and D_y are $t'(n)\cdot(2\eta+\delta)$ close, hence our adversary can run A_{inv} in time $t(n)$ on the examples $x_1, \ldots, x_{t'(n)}$ and succeed with probability $1-\kappa_2 - t'(n)\cdot(2\eta+\delta) \geq (1-\kappa_2)/2$ in producing a sampler whose output distribution is κ_1-close to $\mathcal{U}_{\Phi_{pk}^{-1}(1)}$. Call this output distribution Z. Let $\beta(D)$ denote the distribution obtained by applying the map β on D. The proof of Claim 119 shows that $\beta(D)$ is $(2\eta+\delta)$-close to the distribution $\mathcal{U}_{\Phi_{pk}^{-1}(1)}$. Thus, with probability $(1-\kappa_2)/2$, Z is $(\kappa_1 + (2\eta+\delta))$-close to the distribution $\beta(D)$. By definition of D, we have

$$\mathop{\mathbf{Pr}}_{(m,\sigma)\in D}[\forall i \in [t'], m_i \neq m] \geq 1 - \frac{t'}{|\mathcal{M}|}.$$

Thus, with probability $\frac{1-\kappa_2}{2}$,

$$\mathop{\mathbf{Pr}}_{z\in Z}[z = g(m,\sigma) \text{ and } \forall i \in [t'], m_i \neq m] \geq 1 - \kappa_1 - (2\eta+\delta) - \frac{t'}{|\mathcal{M}|} \geq \frac{1-\kappa_1}{2}$$

Thus, with overall probability $(1-\kappa_1)(1-\kappa_2)/4 \geq \epsilon$, the adversary succeeds in producing $z = g(m,\sigma)$ such that $\forall i \in [t'], m_i \neq m$. Applying the map γ on (Φ_{pk}, z), the adversary gets the pair (m,σ). Also, note that the total running time of the adversary is $t_1(t'(n)) + t'(n) \leq 2t_1(t'(n)) \leq t(n)$ which contradicts the (t,ϵ)-RMA security of the signature scheme. □

A specific hardness assumption.

At this point, at the cost of sacrificing some generality, we consider a particular instantiation of a signature scheme from the literature which meets our requirements. While similar signature schemes can be constructed under many different cryptographic assumptions in the literature, we forsake such generality to keep the discussion from getting too cumbersome.

To state our cryptographic assumption, we need the following notation:

- PRIMES$_k$ is the set of k-bit prime numbers.
- RSA$_k$ is the set of all products of two primes of length $\lfloor (k-1)/2 \rfloor$.

The following cryptographic assumption (a slight variant of the standard RSA assumption) appears in [111].

Assumption 1. ***The RSA$'$ $s(k)$ assumption:*** *Fix any $m \in$ RSA$_k$ and let $x \in_U \mathbb{Z}_m^*$ and $p \in_U$ PRIMES$_{k+1}$. Let A be any probabilistic algorithm running in time $s(k)$. Then,*

$$\mathop{\mathbf{Pr}}_{(x,p)}[A(m,x,p) = y \text{ and } y^p = x \ (mod\ m)] \leq \frac{1}{s(k)}.$$

As mentioned in [111], given the present state of computational number theory, it is plausible to conjecture the RSA$'$ $s(k)$ assumption for $s(k) = 2^{k^\delta}$ for some absolute constant $\delta > 0$. For the sake of conciseness, for the rest of this section we write “Assumption 1 holds true” to mean that

Assumption 1 holds true with $s(k) = 2^{n^\delta}$ for some fixed constant $\delta > 0$. (We note, though, that all our hardness results go through giving superpolynomial hardness using only $s(k) = k^{\omega(1)}$.)

Micali *et al.* [111] give a construction of a "unique signature scheme" using Assumption 1:

Theorem 120. *If Assumption 1 holds true, then there is a* $(t = 2^{n^\delta}, \epsilon = 1/t)$*-RMA secure signature scheme* (G, S, V) *with the following property : For any message* $m \in \mathcal{M}$*, there do not exist* $\sigma_1 \neq \sigma_2$ *such that* $V(m, \sigma_1) = V(m, \sigma_2) = 1$*. In this scheme the signing algorithm* S *is deterministic and the message space* $\mathcal{M}$ *is of size* 2^{n^δ}*.*

The above theorem says that under the RSA$'$ $s(k)$ assumption, there is a deterministic signature scheme such that there is only one signature σ_m for every message m, and for every message m the only accepting input for V is (m, σ_m). As a consequence, the signature scheme in Theorem 120 has the property that every (pk, sk) pair that can be generated by G is $(0, 0)$-special.

Remark 121. It is important to note here that constructions of $(0, 0)$ special signature schemes are abundant in the literature. A partial list follows : Lysyanskaya [107] constructed a deterministic $(0, 0)$ special signature scheme using a strong version of the Diffie–Hellman assumption. Hohenberger and Waters [67] constructed a scheme with a similar guarantee using a variant of the Diffie–Hellman assumption on bilinear groups. In fact, going back much further, Cramer and Shoup [25, 54] show that using the Strong RSA assumption, one can get a $(0, 0)$ special signature scheme (which however is not deterministic). We remark that the scheme as stated in [25] is not $(0, 0)$ special in any obvious sense, but the more efficient version in [54] can be easily verified to be $(0, 0)$ special. Throughout this section, for the sake of simplicity, we use the signature scheme in Theorem 120.

Instantiating Theorem 118 with the signature scheme from Theorem 120, we obtain the following corollary:

Corollary 122. *Suppose that Assumption 1 holds true. Then the following holds : Let* $\mathcal{C}$ *be a function class such that there is a polynomial time (*n^k*-time) invertible Levin reduction from CIRCUIT-SAT to* $\mathcal{C}$*. Then* $\mathcal{C}$ *is* $(2^{n^c}, 1 - 2^{-n^c}, 1 - 2^{-n^c})$*-hard for inverse approximate uniform generation for some constant* $c > 0$ *(depending only on the "*δ*" in Assumption 1 and on* k*).*

Inverse approximate uniform generation hardness results for specific function classes whose satisfiability problem is NP-complete.

In this subsection we use Corollary 122 to prove hardness results for inverse approximate uniform generation for specific function classes $\mathcal{C}$ for which there are invertible Levin reductions from CIRCUIT-SAT to $\mathcal{C}$.

Recall that a 3-CNF formula is a conjunction of clauses (disjunctions) of length 3. The following fact can be easily verified by inspecting the standard reduction from CIRCUIT-SAT to 3-CNF-SAT.

Fact 123. *There is a polynomial time invertible Levin reduction from CIRCUIT-SAT to 3-CNF-SAT.*

As a corollary, we have the following result.

Corollary 124. *If Assumption 1 holds true, then there exists an absolute constant $c > 0$ such that the class 3-CNF is $(2^{n^c}, 1 - 2^{-n^c}, 1 - 2^{-n^c})$-hard for inverse approximate uniform generation.*

Corollary 124 is interesting in light of the well known fact that the class of all 3-CNF formulas is efficiently PAC learnable from uniform random examples (in fact under any distribution).

We next observe that the problem of inverse approximate uniform generation remains hard even for 3-CNF formulas in which each variable occurs a bounded number of times. To prove this we will use the fact that polynomial time invertible Levin reductions compose:

Fact 125. *If there is a polynomial time invertible Levin reduction from CIRCUIT-SAT to $\mathcal{C}$ and a polynomial time Levin reduction from $\mathcal{C}$ to $\mathcal{C}_1$, then there is a polynomial time invertible Levin reduction from CIRCUIT-SAT to $\mathcal{C}_1$.*

The following theorem says that the inverse approximate uniform generation problem remains hard for the class of all 3-CNF formulas in which each variable occurs at most 4 times (hereafter denoted 3,4-CNF).

Theorem 126. *If Assumption 1 holds true, then there exists an absolute constant $c > 0$ such that 3,4-CNF-SAT is $(2^{n^c}, 1 - 2^{-n^c}, 1 - 2^{-n^c})$-hard for inverse approximate uniform generation.*

Proof. Tovey [137] shows that there is a polynomial time invertible Levin reduction from 3-CNF-SAT to 3,4-CNF-SAT. Using Fact 125, we have a polynomial time Levin reduction from CIRCUIT-SAT to 3,4-CNF-SAT. Now the result follows from Corollary 122 □

The next theorem shows that the class of all intersections of two halfspaces over n Boolean variables is hard for inverse approximate uniform generation.

Theorem 127. *If Assumption 1 holds true, then there exists an absolute constant $c > 0$ such that $\mathcal{C} = \{$all intersections of two halfspaces over n Boolean variables$\}$ is $(2^{n^c}, 1-2^{-n^c}, 1-2^{-n^c})$-hard for inverse approximate uniform generation.*

Proof. We recall that the SUBSET-SUM problem is defined as follows : An instance Φ is defined by positive integers $w_1, \ldots, w_n, s > 0$. A satisfying assignment for this instance is given by $x \in \{0,1\}^n$ such that $\sum_{i=1}^n w_i x_i = s$. It is well known that the SUBSET-SUM problem is NP-complete and it is folklore that there is a invertible Levin reduction from 3-SAT to SUBSET-SUM. However, since it is somewhat difficult to find this reduction explicitly in the literature, we outline such a reduction.

To describe the reduction, we first define 1-in-3-SAT. An instance Ψ of 1-in-3-SAT is defined over Boolean variables $x_1, \ldots, x_n$ with the following constraints : The i^{th} constraint is defined by a subset of at most three literals over $x_1, \ldots, x_n$. An assignment to $x_1, \ldots, x_n$ satisfies Ψ if and only if for every constraint there is exactly one literal which is set to true. Schaefer [127] showed that 3-SAT reduces to 1-in-3-SAT in polynomial time, and the reduction can be easily verified to be an invertible Levin reduction. Now the standard textbook reduction from 3-SAT to SUBSET-SUM (which can be found e.g. in [122]) applied to instances of 1-in-3-SAT, can be easily seen to be a polynomial time invertible Levin reduction. By Fact 125, we thus have a polynomial time invertible Levin reduction from 3-CNF-SAT to SUBSET-SUM.

With this reduction in hand, it remains only to observe that that any instance of SUBSET-SUM is also an instance of "intersection of two halfspaces," simply because $\sum_{i=1}^{n} w_i x_i = s$ if and only if $s \leq \sum_{i=1}^{n} w_i \cdot x_i \leq s$. Thus, there is a polynomial time invertible Levin reduction from 3-CNF-SAT to the class of all intersections of two halfspaces. This finishes the proof. □

A hardness result where the satisfiability problem is in P.

So far all of our hardness results have been for classes $\mathcal{C}$ of NP-complete languages. As Theorem 118 requires a reduction from CIRCUIT-SAT to $\mathcal{C}$, this theorem cannot be directly used to prove hardness for classes $\mathcal{C}$ which are not NP-hard. We next give an extension of Theorem 118 which can apply to classes $\mathcal{C}$ for which the satisfiability problem is in P. Using this result we will show hardness of inverse approximate uniform generation for MONOTONE-2-CNF-SAT. (Recall that a monotone 2-CNF formula is a conjunction of clauses of the form $x_i \vee x_j$, with no negations; such a formula is trivially satisfiable by the all-true assignment.)

We begin by defining by a notion of invertible one-many reductions that we will need.

Definition 128. *CIRCUIT-SAT is said to have an η-*almost invertible one-many reduction to a function class $\mathcal{C}$ *if the following conditions hold:*

- *There is a polynomial time computable function f such that given an instance Φ of CIRCUIT-SAT (i.e. Φ is a satisfiable circuit), $\Psi = f(\Phi)$ is an instance of $\mathcal{C}$ (i.e. $\Psi \in \mathcal{C}$ and Ψ is satisfiable).*
- *Fix any instance Φ of CIRCUIT-SAT and let $\mathcal{A} = \Psi^{-1}(1)$ denote the set of satisfying assignments of Ψ. Then $\mathcal{A}$ can be partitioned into sets $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $|\mathcal{A}_2|/|\mathcal{A}| \leq \eta$ and there is an efficiently computable function $g : \mathcal{A}_1 \to \Phi^{-1}(1)$ such that $g(x)$ is a satisfying assignment of Φ for every $x \in \mathcal{A}_1$.*
- *For every y which is a satisfying assignment of Φ, the number of pre-images of y under g is exactly the same, and the uniform distribution over $g^{-1}(y)$ is polynomial time samplable.*

We next state the following simple claim which will be helpful later.

Claim 129. *Suppose there is an η-almost invertible one-many reduction from CIRCUIT-SAT to $\mathcal{C}$. Let f and g be the functions from Definition 128. Let Φ be an instance of CIRCUIT-SAT and let $\Psi = f(\Phi)$ be the corresponding instance of $\mathcal{C}$. Define distributions D_1 and D_2 as follows :*

- *A draw from D_1 is obtained by choosing y uniformly at random from $\Phi^{-1}(1)$ and then outputting z uniformly at random from $g^{-1}(y)$.*

- *A draw from D_2 is obtained by choosing z' uniformly at random from $\Psi^{-1}(1)$.*

Then we have $d_{TV}(D_1, D_2) \leq \eta$.

Proof. This is an immediate consequence of the fact that D_1 is uniform over the set $\mathcal{A}_1$ while D_2 is uniform over the set $\mathcal{A}$ (from Definition 128). □

We next have the following extension of Corollary 122.

Theorem 130. *Suppose that Assumption 1 holds true. Then if $\mathcal{C}$ is a function class such that there is an η-almost invertible one-many reduction (for $\eta = 2^{-\Omega(n)}$) from CIRCUIT-SAT to $\mathcal{C}$, then $\mathcal{C}$ is $(2^{n^c}, 1 - 2^{-n^c}, 1 - 2^{-n^c})$-hard for inverse approximate uniform generation for some absolute constant $c > 0$.*

Proof. The proof is similar to the proof of Corollary 122. Assume towards a contradiction that there is an algorithm for inverse approximation uniform generation A_{inv} for $\mathcal{C}$ which runs in time t_1 such that with probability $1 - \kappa_2$, the output distribution is κ_1-close to the target distribution. (We will set t_1, κ_1 and κ_2 later to 2^{n^c}, $1 - 2^{-n^c}$ and $1 - 2^{-n^c}$ respectively.)

Let (G, S, V) be the RMA-secure signature scheme constructed in Theorem 120. Note that (G, S, V) is a (T, ϵ)-RMA secure signature scheme where $T = 2^{n^\delta}$, $\epsilon = 1/T$ and $|\mathcal{M}| = 2^{n^\mu}$ for constant $\delta, \mu > 0$. Let (pk, sk) be a choice of key pair. We will us A_{inv} to contradict the security of (G, S, V). Towards this, consider the function $V_{pk} : \mathcal{M} \times \{0,1\}^* \to \{0,1\}$ defined as $V_{pk}(m, \sigma) = V(m, pk, \sigma)$. Clearly, V_{pk} is an instance of CIRCUIT-SAT. Consider the η-invertible one-many reduction from CIRCUIT-SAT to $\mathcal{C}$. Let α and β have the same meaning as in Definition 128. Let $\Psi = \alpha(V_{pk})$ and let $\mathcal{A}$, $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same meaning as in Definition 128. The adversary receives message-signature pairs $(m_1, \sigma_1) \ldots (m_{t_1}, \sigma_{t_1})$ where $m_1, \ldots, m_{t_1}$ are chosen independently at random from $\mathcal{M}$. For any i, (m_i, σ_i) is a satisfying assignment of V_{pk}. By definition, in time $t_2 = t_1 \cdot \text{poly}(n)$, the adversary can sample $(z_1, \ldots, z_{t_1})$ such that $z_1, \ldots, z_{t_1}$ are independent and $z_i \sim \mathcal{U}_{\beta^{-1}(m_i, \sigma_i)}$. Note that this means that each z_i is an independent sample from $\mathcal{A}_1$ and $|z_i| = \text{poly}(n)$. Note that $(z_1, \ldots, z_{t_1})$ is a t_1-fold product distribution such that if D' denotes the distribution of z_i, then by Claim 129, $d_{TV}(D', \mathcal{U}_{\Psi^{-1}(1)}) \leq \eta$. Hence, if D is the distribution of $(z_1, \ldots, z_{t_1})$, then $d_{TV}(D, \mathcal{U}^t_{\Psi^{-1}(1)}) \leq t_1 \eta$.

Hence, the adversary can now run A_{rec} on the samples $z_1, \ldots, z_{t_1}$ and as long as $1 - \kappa_2 - t_1 \eta \geq (1 - \kappa_2)/2$, succeeds in producing a sampler with probability $(1 - \kappa_2)/2$ whose output distribution (call it Z) is κ_1 close to the distribution $\mathcal{U}_{\Psi^{-1}(1)}$. Note that as $\eta = 2^{-\Omega(n)}$, for any $c > 0$, $t_1 = 2^{n^c}$ and $\kappa_2 = 1 - 2^{-n^c}$ satisfies this condition. Hence, we get that $d_{TV}(Z, D') \leq \kappa_1 + \eta$. Now, observe that

$$\mathop{\mathbf{P}r}_{\rho \in D'}[\beta(\rho) = (m, \sigma) \text{ and } m \neq m_i] = 1 - \frac{t_1}{|\mathcal{M}|}.$$

The above uses the fact that every element in the range of β has the same number of pre-images. This of course implies that

$$\mathop{\mathbf{P}r}_{\rho \in Z}[\beta(\rho) = (m, \sigma) \text{ and } m \neq m_i] \geq 1 - \frac{t_1}{|\mathcal{M}|} - (\kappa_1 + \eta).$$

Again as long as $\kappa_1 \leq 1 - 2(\eta + t_1/|\mathcal{M}|)$, the adversary succeeds in getting a valid message signature pair (m, σ) with $m \neq m_i$ for any $1 \leq i \leq t_1$ with probability $(1-\kappa_1)/2$. Again, we can ensure $\kappa_1 \leq 1 - 2(\eta + t_1/|\mathcal{M}|)$ by choosing c sufficiently small compared to μ. The total probability of success is $(1-\kappa_1)(1-\kappa_2)/4$ and the total running time is $t_1(\text{poly}(n)) + \text{poly}(n)$. Again if c is sufficiently small compared to μ and δ, then the total running time is at most $t_1(\text{poly}(n)) + \text{poly}(n) < T$ and the success probability is at least $(1-\kappa_1)(1-\kappa_2)/4 > \epsilon$, resulting in a contradiction. □

We now demonstrate a polynomial time η-invertible one-many reduction from CIRCUIT-SAT to MONOTONE-2-CNF-SAT for $\eta = 2^{-\Omega(n)}$. The reduction uses the "blow-up" idea used to prove hardness of approximate counting for MONOTONE-2-CNF-SAT in [72]. We will closely follow the instantiation of this technique in [148].

Lemma 131. *There is a polynomial time η-almost invertible one-many reduction from CIRCUIT-SAT to MONOTONE-2-CNF-SAT where $\eta = 2^{-\Omega(n)}$.*

Proof. We begin by noting the following simple fact.

Fact 132. *If there is a polynomial time invertible Levin reduction from CIRCUIT-SAT to a class $\mathcal{C}_1$ and an η-almost invertible one-many reduction from $\mathcal{C}_1$ to $\mathcal{C}_2$, then there is a polynomial time η-almost invertible one-many reduction from CIRCUIT-SAT to $\mathcal{C}_2$.*

Since there is an invertible Levin reduction from CIRCUIT-SAT to 3-CNF-SAT, by virtue of Fact 132, it suffices to demonstrate a polynomial time η-almost invertible one-many reduction from 3-CNF-SAT to MONOTONE-2-CNF-SAT. To do this, we first construct an instance of VERTEX-COVER from the 3-CNF-SAT instance. Let $\Phi = \bigwedge_{i=1}^{m} \Phi_i$ be the instance of 3-CNF-SAT. Construct an instance of VERTEX-COVER by introducing seven vertices for each clause Φ_i (one corresponding to every satisfying assignment of Φ_i). Now, put an edge between any two vertices of this graph if the corresponding assignments to the variables of Φ conflict on some variable. We call this graph G. We observe the following properties of this graph :

- G has exactly $7m$ vertices.
- Every vertex cover of G has size at least $6m$.
- There is an efficiently computable and invertible injection ℓ between the satisfying assignments of Φ and the vertex covers of G of size $6m$. To get the vertex cover corresponding to a satisfying assignment, for every clause Φ_i, include the six vertices in the vertex cover which conflict with the satisfying assignment.

We next do the blow-up construction. We create a new graph G' by replacing every vertex of G with a cloud of $10m$ vertices, and for every edge in G we create a complete bipartite graph between the corresponding clouds in G'. Clearly, the size of the graph G' is polynomial in the size of the 3-CNF-SAT formula. We define a map g_1 between vertex covers of G' and vertex covers of G as follows : Let S' be a vertex cover of G'. We define the set $S = g_1(S')$ in the following way. For every vertex v in the graph G, if all the vertices in the corresponding cloud in G' are in S', then include $v \in S$, else do not include v in S. It is easy to observe that g_1 maps vertex covers of G' to vertex covers of G. It is also easy to observe that a vertex cover of G of size s has $(2^{10m}-1)^{7m-s}$ pre-images under g_1.

Now, observe that we can construct a MONOTONE-2-CNF-SAT formula Ψ which has a variable corresponding to every vertex in G' and every subset S' of G' corresponds to a truth assignment $y_{S'}$ to Ψ such that $\Psi(y_{S'}) = 1$ if and only if S' is a vertex cover of G'. Because of this correspondence, we can construct a map g_1' which maps satisfying assignments of Ψ to vertex covers of G. Further, a vertex cover of size s in graph G has $(2^{10m}-1)^{7m-s}$ pre-images under g_1'. Since the total number of vertex covers of G of size s is at most $\binom{7m}{s}$, the total number of satisfying assignments of Ψ which map to vertex covers of G of size more than $6m$ can be bounded by :

$$\sum_{s=6m+1}^{7m} \binom{7m}{s} \cdot (2^{10m}-1)^{7m-s} \leq m \cdot \binom{7m}{6m+1} \cdot (2^{10m}-1)^{m-1} \leq (2^{10m}-1)^m \cdot \frac{2^{7m}}{2^{10m}-1}$$

On the other hand, since Φ has at least one satisfying assignment, hence G has at least one vertex cover of size $6m$ and hence the total number of satisfying assignments of Ψ which map to vertex covers of G of size $6m$ is at least $(2^{10m}-1)^m$. Thus, if we let $\mathcal{A}$ denote the set of satisfying assignments of Ψ and $\mathcal{A}_1$ be the set of satisfying assignment of Ψ which map to vertex covers of G of size exactly $6m$ (under g_1), then $|\mathcal{A}_1|/|\mathcal{A}| \geq 1 - 2^{-\Omega(n)}$. Next, notice that we can define the map g mapping $\mathcal{A}_1$ to the satisfying assignments of Φ in the following manner : $g(x) = \ell^{-1}(g_1(x))$. It is easy to see that this map satisfies all the requirements of the map g from Definition 128 which concludes the proof. □

Combining Lemma 131 with Theorem 130, we have the following corollary.

Corollary 133. *If Assumption 1 holds true, then MONOTONE-2-CNF-SAT is* $(2^{n^c}, 1-2^{-n^c}, 1-2^{-n^c})$ *hard for inverse approximate uniform generation for some absolute constant* $c > 0$.

As a consequence of the above result, we also get hardness for inverse approximate uniform generation of degree-2 *polynomial threshold functions (PTFs)*; these are functions of the form $\text{sign}(q(x))$ where $q(x)$ is a degree-2 multilinear polynomial over $\{0,1\}^n$.

Corollary 134. *If Assumption 1 holds true, then the class of all* n*-variable degree-2 polynomial threshold functions is* $(2^{n^c}, 1-2^{-n^c}, 1-2^{-n^c})$ *hard for inverse approximate uniform generation for some absolute constant* $c > 0$.

Proof. This follows immediately from the fact that every monotone 2-CNF formula can be expressed as a degree-2 PTF. To see this, note that if $\Phi = \bigwedge_{i=1}^{m}(x_{i1} \vee x_{i2})$ where each x_{ij} is a 0/1 variable, then $\Phi(x)$ is true if and only if $\sum_{i=1}^{m} x_{i1} + x_{i2} - x_{i1} \cdot x_{i2} \geq m$. This finishes the proof. □

Hardness results based on Message Authentication Codes.

All of the previous hardness results intuitively correspond to the case when the second step of our "standard approach" is algorithmically hard. Indeed, consider a class $\mathcal{C}$ of functions that has an efficient approximate uniform generation algorithm. Unless $P \neq NP$ there cannot be any Karp reduction from CIRCUIT-SAT to $\mathcal{C}$ (this would contradict the NP-completeness of CIRCUIT-SAT) and hence Theorem 118 is not applicable in this setting. In fact, even for $\eta = 1 - 1/\mathrm{poly}(n)$ there cannot be any η-almost invertible one-many reduction from CIRCUIT-SAT to $\mathcal{C}$ unless $P \neq NP$. This makes Theorem 130 inapplicable in this setting. Thus, to prove hardness results for classes that have efficient approximate uniform generation algorithms, we need some other approach.

In this section we show that Message Authentication Codes (MAC) can be used to establish hardness of inverse approximate uniform generation for such classes. We begin by defining MACs. (We remark that we use a restricted definition which is sufficient for us; for the most general definition, see [59].)

Definition 135. *A* Message Authentication Code (MAC) *is a triple* (G, T, V) *of polynomial-time algorithms with the following properties :*

- **(Key generation algorithm)** $G(\cdot)$ *is a randomized algorithm which on input* 1^n *produces a secret key* sk*;*
- **(Tagging algorithm)** T *is a randomized algorithm which takes as input message* m*, secret key* sk *and randomness* r *and outputs* $\sigma \leftarrow T(m, sk, r)$*;*
- **(Verification algorithm)** V *is a deterministic algorithm which takes as input message* m*, secret key* sk *and* σ*. If* $\sigma = T(m, sk, r)$ *for some* r *then* $V(m, sk, \sigma) = 1$.

For the purposes of our hardness results we require MACs with some special properties. While our hardness results can be derived from slightly more general MACs than those we specify below, we forsake some generality for the sake of clarity. For a MAC (G, T, V) and a choice of secret key sk, we say σ is a *valid tag* for message m if there exists r such that $\sigma = T(m, sk, r)$. Likewise, we say that σ is a *potential tag* for message m if $V(m, sk, \sigma) = 1$.

Definition 136. *A Message Authentication Code* (G, T, V) *over a message space* $\mathcal{M}$ *is said to be* special *if the following conditions hold : For any secret key* sk*,*

- *For every message* $m \in \mathcal{M}$*, the set of valid tags is identical to the set of potential tags..*
- *For every two messages* $m_1 \neq m_2$ *and every* σ_1, σ_2 *such that* σ_i *is a valid tag for* m_i*, we have* $\mathbf{P}r_r[T(m_1, sk, r) = \sigma_1] = \mathbf{P}r_r[T(m_2, sk, r) = \sigma_2]$.. *In particular, the cardinality of the set of valid tags for* m *is the same for all* m.

We next define the standard notion of security under Random Message attacks for MACs. As before, from now onwards, we will assume implicitly that $\mathcal{M}$ is the message space.

Definition 137. *A special MAC (G, T, V) is said to be (t, ϵ)-*RMA secure *if the following holds : Let $sk \leftarrow G(1^n)$. Let $(m_1, \dots, m_t)$ be chosen uniformly at random from $\mathcal{M}$. Let $\sigma_i \leftarrow T(m_i, sk, r)$. Then for any probabilistic algorithm A running in time t,*

$$\mathop{\mathbf{Pr}}_{sk,(m_1,\dots,m_t),(\sigma_1,\dots,\sigma_t)}[A(m_1, \dots, m_t, \sigma_1, \dots, \sigma_t) = (m', \sigma')] \leq \epsilon$$

where $V(m', sk, \sigma') = 1$ and $m' \neq m_i$ for all $i = 1, \dots, t$.

It is known how to construct MACs meeting the requirements in Definition 137 under standard cryptographic assumptions (see [59]).

A general hardness result based on Message Authentication Codes.

The next theorem shows that special MACs yield hardness results for inverse approximate uniform generation.

Theorem 138. *Let $c > 0$ and (G, T, V) be a (t, ϵ)-RMA secure special MAC for some $t = 2^{n^c}$ and $\epsilon = 1/t$ with a message space $\mathcal{M}$ of size $2^{\Omega(n)}$. Let V_{sk} denote the function $V_{sk} : (m, \sigma) \mapsto V(m, sk, \sigma)$. If $V_{sk} \in \mathcal{C}$ for every s_k, then there exists $\delta > 0$ such that $\mathcal{C}$ is (t_1, κ, η)-hard for inverse approximate uniform generation for $t_1 = 2^{n^\delta}$ and $\kappa = \eta = 1 - 2^{-n^\delta}$.*

Proof. Towards a contradiction, let us assume that there is an algorithm A_{inv} for inverse approximate uniform generation of $\mathcal{C}$ which runs in time t_1 and with probability $1 - \eta$ outputs a sampler whose statistical distance is at most κ from the target distribution. (We will set t_1, κ and η later in the proof.) We will use A_{inv} to contradict the security of the MAC. Let sk be a secret key chosen according to $G(1^n)$. Now, the adversary receives message-tag pairs $(m_1, \sigma_1), \dots, (m_{t_1}, \sigma_{t_1})$ where $m_1, \dots, m_{t_1}$ are chosen independently at random from $\mathcal{M}$. Because the MAC is special, for each i we have that σ_i is a uniformly random valid tag for the message m_i. Hence each (m_i, σ_i) is an independent and uniformly random satisfying assignment of V_{sk}.

We can thus run A_{inv} on the samples $(m_1, \sigma_1), \dots, (m_{t_1}, \sigma_{t_1})$ with its accuracy parameter set to κ and its confidence parameter set to $1 - \eta$. Taking $\kappa = \eta = 1 - 2^{-n^\delta}$, we can choose δ small enough compared to c, and with $t_1 = 2^{n^\delta}$ we get that the total running time of A_{inv} is at most $2^{n^c}/2$. By the definition of inverse approximate uniform generation, with probability at least $1 - \eta = 2^{-n^\delta}$ the algorithm A_{inv} outputs a sampler for a distribution Z that is $\kappa = (1 - 2^{-n^\delta})$-close to the uniform distribution over the satisfying assignments of V_{sk}. Now, observe that

$$\mathop{\mathbf{Pr}}_{(m,\sigma)\sim\mathcal{U}_{V_{sk}^{-1}(1)}}[m_i \neq m \text{ for all } i \in [t_1]] \geq 1 - \frac{t_1}{|\mathcal{M}|}.$$

Thus,

$$\mathop{\mathbf{Pr}}_{z\sim Z}[z = (m, \sigma) \text{ and } m_i \neq m \text{ for all } i \in [t_1]] \geq (1 - \kappa) - \frac{t_1}{|\mathcal{M}|}.$$

This means that with probability $(1 - \eta) \cdot ((1 - \kappa) - \frac{t_1}{|\mathcal{M}|})$, the adversary can output a forgery. It is clear that for a suitable choice of δ relative to c, recalling that $\kappa = \eta = 1 - 2^{-n^\delta}$, the probability of outputting a forgery is greater than 2^{-n^c}, which contradicts the security of the MAC. □

Unlike signature schemes, which permitted intricate reductions (cf. Theorem 118), in the case of MACs we get a hardness result for complexity class $\mathcal{C}$ only if V_{sk} itself belongs to $\mathcal{C}$. While special MACs are known to exist assuming the existence of one-way functions [59], the constructions are rather involved and rely on constructions of pseudorandom functions (PRFs) as an intermediate step. As a result, the verification algorithm V also involves computing PRFs; this means that using these standard constructions, one can only get hardness results for a class $\mathcal{C}$ if PRFs can be computed in $\mathcal{C}$. As a result, the class $\mathcal{C}$ tends to be fairly complex, making the corresponding hardness result for inverse approximate uniform generation for $\mathcal{C}$ somewhat uninteresting.

One way to bypass this is to use construction of MACs which do not involve use of PRFs as an intermediate step. In recent years there has been significant progress in this area [93, 41]. While both these papers describe several MACs which do not require PRFs, the one most relevant for us is the MAC construction of [93] based on the hardness of the "Learning Parity with Noise" (LPN) problem.

Some specific hardness assumptions, and a corresponding specific hardness result.

We first state a "decision" version of LPN. To do this, we need the following notation:

- Let Ber_τ denote the following distribution over $GF(2)$: If $x \leftarrow Ber_\tau$, then $\mathbf{P}r[x = 1] = \tau$.
- For $x \in GF(2)^n$, we use $\Lambda(x, \tau, \cdot)$ to denote the distribution $(r, x \cdot r \oplus e)$ over $GF(2)^n \times GF(2)$ where $r \sim GF(2)^n$ and $e \sim Ber_\tau$ and $x \cdot r = \oplus_i x_i r_i \pmod 2$.

Assumption 2. *Let $\tau \in (0, 1/2)$ and let $\mathcal{O}_{x,\tau}$ be an oracle which, each time it is invoked, returns an independent uniformly random sample from $\Lambda(x, \tau, \cdot)$. The LPN assumption states that for any* $\mathrm{poly}(n)$*-time algorithm $\mathcal{A}$,*

$$\left| \left[\mathop{\mathbf{P}r}_{x \in GF(2)^n} [\mathcal{A}^{\mathcal{O}_{x,\tau}} = 1 \right] - \left[\mathop{\mathbf{P}r}_{x \in GF(2)^n} [\mathcal{A}^{\mathcal{O}_{x,1/2}} = 1 \right] \right| \leq \epsilon$$

for some ϵ which is negligible in n.

LPN is a well-studied problem; despite intensive research effort, the fastest known algorithm for this problem takes time $2^{O(n/\log n)}$ [16]. For our applications, we will need a variant of the above LPN assumption. To define the assumption, let $\Lambda(x, \ell, \tau, \cdot)$ denote the distribution over $(A, A \cdot x \oplus e)$ where A is uniformly random in $GF(2)^{\ell \times n}$ and e is uniformly random over the set $\{z \in GF(2)^\ell : wt(z) \leq \lceil \tau \ell \rceil\}$. The vector e is usually referred to as the *noise vector.*

Assumption 3. *Let $\tau \in (0, 1/2)$, $\ell = c \cdot n$ for some $0 < c < 1/2$ and let $\mathcal{O}_{x,\ell,\tau}$ be an oracle which returns a uniformly random sample from $\Lambda(x, \ell, \tau, \cdot)$. Then the* (t, ϵ) exact LPN assumption *states that for any algorithm $\mathcal{A}$ running in time t,*

$$\left| \mathop{\mathbf{P}r}_{x \in GF(2)^n} \left[\mathcal{A}^{\mathcal{O}_{x,\ell,\tau}} = 1\right] - \mathop{\mathbf{P}r}_{x \in GF(2)^n} \left[\mathcal{A}^{\mathcal{O}_{x,\ell,1/2}} = 1\right] \right| \leq \epsilon$$

For the sake of brevity, we henceforth refer to this assumption by saying "the exact (n, ℓ, τ) LPN problem is (t, ϵ)-hard."

The above conjecture seems to be very closely related to Assumption 2, but it is not known whether Assumption 2 formally reduces to Assumption 3. Assumption 3 has previously been suggested in the cryptographic literature [84] in the context of getting perfect completeness in LPN-based protocols. We note that Arora and Ge [4] have investigated the complexity of this problem and gave an algorithm which runs in time $n^{O(\ell)}$. We believe that the proximity of Assumption 3 to the well-studied Assumption 2, as well as the failure to find algorithms for Assumption 3, make it a plausible conjecture. For the rest of this section we use Assumption 3 with $t = 2^{n^\beta}$ and $\epsilon = 2^{-n^\beta}$ for some fixed $\beta > 0$.

We next define a seemingly stronger variant of Assumption 3 which we call *subset exact LPN*. This requires the following definitions: For $x, v \in GF(2)^n$, $\ell, d \leq n$ and $\tau \in (0, 1/2)$, we define the distribution $\Lambda^a(x, v, \ell, \tau, \cdot)$ as follows :

$$\Lambda^a(x, v, \ell, \tau, \cdot) = \begin{cases} \Lambda(x \cdot v, \ell, 1/2, \cdot) & \text{if } wt(v) < d \\ \Lambda(x \cdot v, \ell, \tau, \cdot) & \text{if } wt(v) \geq d \end{cases}$$

where $x \cdot v \in GF(2)^n$ is defined by $(x \cdot v)_i = x_i \cdot v_i$. In other words, if $wt(v) \geq d$, then the distribution $\Lambda^a(x, v, \ell, \tau)$ projects x into the non-zero coordinates of v and then outputs samples corresponding to exact LPN for the projected vector. We define the oracle $\mathcal{O}^a_{x,\ell,d,\tau}(\cdot)$ which takes an input $v \in GF(2)^n$ and outputs a random sample from $\Lambda^a(x, v, \ell, \tau, \cdot)$. The subset exact LPN assumption states the following:

Assumption 4. *Let $\tau \in (0, 1/2)$, $\ell = c \cdot n$ and $d = c' \cdot n$ for some $0 < c, c' < 1/2$. The (t, ϵ)-*subset exact LPN assumption *says that for any algorithm $\mathcal{A}$ running in time t,*

$$\left| \underset{x \in GF(2)^n}{\mathbf{Pr}} \left[\mathcal{A}^{\mathcal{O}^a_{x,\ell,d,\tau}} = 1 \right] - \underset{x \in GF(2)^n}{\mathbf{Pr}} \left[\mathcal{A}^{\mathcal{O}^a_{x,\ell,d,1/2}} = 1 \right] \right| \leq \epsilon.$$

For the sake of brevity, we henceforth refer to this assumption by saying "the subset exact (n, ℓ, d, τ) LPN problem is (t, ϵ)-hard."

Assumption 4 is very similar to the *subset LPN assumption* used in [93] and previously considered in [124]. The subset LPN assumption is the same as Assumption 4 but with $\ell = 1$ and the coordinates of the noise vector e being drawn independently from Ber_τ. Pietrzak [124] showed that the subset LPN assumption is implied by the standard LPN assumption (Assumption 2) with a minor change in the security parameters. Along the same lines, the next lemma shows that Assumption 3 implies Assumption 4 with a minor change in parameters. The proof is identical to the proof of Theorem 1 in [124] and hence we do not repeat it here.

Lemma 139. *If the exact (n, ℓ, τ) LPN problem is (t, ϵ) hard, then for any $g \in \mathbb{N}$, the subset exact $(n', \ell, n + g, \tau)$ LPN problem is (t', ϵ') hard for $n' \geq n + g$, $t' = t/2$ and $\epsilon' = \epsilon + \frac{2t}{2^{g+1}}$.*

Proof. The proof of this lemma follows verbatim from the proof of Theorem 1 in [124]. The key observation is that the reduction from subset LPN to LPN in Theorem 1 in [124] is independent of the noise distribution. □

From Lemma 139, we get that Assumption 3 implies Assumption 4. In particular, we can set $\ell = n/5$ and $g = n/10$, $n' \geq 11n/10$. Then we get that if the exact (n, ℓ, τ) problem is $(2^{n^{\beta}}, 2^{-n^{\beta}})$ hard for some $\beta > 0$, then the subset exact $(n', \ell, 11n/10, \tau)$ is also $(2^{n^{\beta'}}, 2^{-n^{\beta'}})$ hard for some other $\beta' > 0$. For the rest of this section, we set the value of ℓ and g as above and we assume that the subset exact $(n', \ell, 11n/10, \tau)$ is $(2^{n^{\beta'}}, 2^{-n^{\beta'}})$ hard for some $\beta' > 0$.

Now we are ready to define the following Message Authentication Code (MAC) (G, S, V), which we refer to as LPN-MAC:

- The key generation algorithm G chooses a random matrix $X \in GF(2)^{\lambda \times n}$ and a string $x' \in GF(2)^{\lambda}$, where $\lambda = 2n$.
- The tagging algorithm samples $R \in GF(2)^{\ell \times \lambda}$ and $e \in GF(2)^{\ell}$ where e is a randomly chosen vector in $GF(2)^{\ell}$ with at most $\lceil \tau \ell \rceil$ ones. The algorithm outputs $(R, R^T \cdot (X \cdot m + x') + e)$.
- The verification algorithm, given tag (R, Z) for message m, computes $y = Z + R^T \cdot (X \cdot m + x')$ and accepts if and only if the total number of ones in y is at most $\lceil \tau \ell \rceil$.

Note that all arithmetic operations in the description of the above MAC are done over $GF(2)$. The following theorem shows that under suitable assumptions the above MAC is special and secure as desired:

Theorem 140. *Assuming that the exact (n, ℓ, τ) problem is (t, ϵ) hard for $t = 2^{n^{\beta}}$ and $\epsilon = 2^{-n^{\beta}}$ for $\beta > 0$, LPN-MAC described above is a (t', ϵ')-RMA-secure special MAC for $t' = 2^{n^{\beta'}}$ and $\epsilon' = 2^{-n^{\beta'}}$ for some $\beta' > 0$.*

Proof. First, it is trivial to observe that the MAC described above is a special MAC. Thus, we are only left with the task of proving the security of this construction. In [93] (Theorem 5), the authors show that the above MAC is secure with the above parameters under Assumption 2 provided the vector e in the description of LPN-MAC is drawn from a distribution where every coordinate of e is an independent draw from Ber_{τ}. (We note that the MAC of Theorem 5 in [93] is described in a slightly different way, but Dodis *et al.* [41] show that the above MAC and the MAC of Theorem 5 in [93] are exactly the same). Follow the same proof verbatim except whenever [93] use the subset LPN assumption, we use the subset exact LPN assumption (i.e. Assumption 4), we obtain a proof of Theorem 140. □

A problem for which inverse approximate uniform generation is hard but approximate uniform generation is easy.

Given Theorem 138, in order to come up with a problem where inverse approximate uniform generation is hard but approximate uniform generation is easy, it remains only to show that the verification algorithm for LPN-MAC can be implemented in a class of functions for which approximate uniform generation is easy. Towards this, we have the following definition.

Definition 141. *BILINEAR-MAJORITY$_{\ell,n,\lambda,\tau}$ is a class of Boolean functions such that every $f \in$BILINEAR-MAJORITY$_{\ell,n,\lambda,\tau}$, $f : GF(2)^{\ell\times\lambda} \times GF(2)^{\ell} \times GF(2)^{n} \to \{0,1\}$ is parameterized by subsets $S_1,\dots,S_\lambda \subseteq [n]$ and $x^0 \in GF(2)^{\lambda}$ and is defined as follows : On input $(R,Z,m) \in GF(2)^{\ell\times\lambda} \times GF(2)^{\ell} \times GF(2)^{n}$, define*

$$y_i = Z_i + \sum_{j=1}^{\lambda} R_{ij} \cdot (\sum_{\ell \in S_j} m_\ell + x_j^0)$$

where all the additions and multiplications are in $GF(2)$. Then $f(R,Z,m) = 1$ if and only if at most $\lceil \tau\ell \rceil$ coordinates $y_1,\dots,y_\ell$ are 1.

Claim 142. *For the LPN-MAC with parameters ℓ, n, λ and τ described earlier, the verification algorithm V can be implemented in the class BILINEAR-MAJORITY$_{\ell,n,\lambda,\tau}$.*

Proof. Consider the LPN-MAC with parameters ℓ, n, λ and τ and secret key X and x'. Now define a function f in BILINEAR-MAJORITY$_{\ell,n,\lambda,\tau}$ where $x^0 = x'$ and the subset $S_j = \{i : X_{ji} = 1\}$. It is easy to check that the corresponding $f(R,Z,m) = 1$ if and only if (R,Z) is a valid tag for message m. $\square$

The next and final claim says that there is an efficient approximate uniform generation algorithm for BILINEAR-MAJORITY$_{\ell,n,\lambda,\tau}$:

Claim 143. *There is an algorithm which given any $f \in$BILINEAR-MAJORITY$_{\ell,n,\lambda,\tau}$ (with parameters $S_1,\dots,S_\lambda \subseteq [n]$ and $x^0 \in GF(2)^{\lambda}$) and an input parameter $\delta > 0$, runs in time $\mathrm{poly}(n,\ell,\lambda,\log(1/\delta))$ and outputs a distribution which is δ-close to being uniform on $f^{-1}(1)$.*

Proof. The crucial observation is that for any (R,m), the set $\mathcal{A}_{R,m} = \{z : f(R,Z,m) = 1\}$ has cardinality independent of R and m. This is because after we fix R and m, if we define $b_i = \sum_{j=1}^{\lambda} R_{ij} \cdot (\sum_{\ell\in S_j} m_\ell + x_j^0)$, then $y_i = Z_i + b_i$. Thus, for every fixing of R and m, since b_i is fixed, the set of those Z such that the number of y_i's which are 1 is bounded by $\tau\ell$ is independent of R and m. This implies that the following sampling algorithm returns a uniformly random element of $f^{-1}(1)$:

- Randomly sample R and m. Compute b_i as defined earlier.
- Let $a = \lceil \tau\ell \rceil$ and consider the halfspace $g(y) = sign(a - \sum_{i=1}^{\ell} y_i)$. Now, we use Theorem 99 to sample uniformly at random from $g^{-1}(1)$ and hence draw a uniformly random y from the set $\{y \in \{0,1\}^{\ell} : \sum_{i=1}^{\ell} y_i \le a\}$.

- We set $Z_i = y_i + b_i$. Output (R, Z, m).

The guarantee on the running time of the procedure follows simply by using the running time of Theorem 99. Similarly, the statistical distance of the output from the uniform distribution on $f^{-1}(1)$ is at most δ. □

4.6 Efficient inverse approximate uniform generation when approximate uniform generation is infeasible

In Section 4.3 we gave an efficient algorithm for the inverse approximate uniform generation problem for halfspaces, and in Section 4.4 we gave a quasi-polynomial time algorithm for the inverse approximate uniform generation problem for DNFs. Since both these algorithms follow the standard approach, both crucially use efficient algorithms for the corresponding uniform generation problems [81, 112]. In this context, it is natural to ask the following question: *Is inverse approximate uniform generation easy only if the corresponding approximate uniform generation problem is easy?*

In this section we show that the answer to this question is “no” (for at least two reasons). First, we point out that a negative answer follows easily from the well-known fact that it is computationally hard to “detect unique solutions.” In more detail, we recall the definition of the UNIQUE-SAT problem. UNIQUE-SAT is a promise problem where given a CNF Φ, the task is to distinguish between the following two cases:

- Φ has no satisfying assignment; versus
- Φ has *exactly* one satisfying assignment.

In a famous result, Valiant and Vazirani [145] showed the following.

Theorem 144. *[145] There is a randomized polynomial time reduction from CNF-SAT to UNIQUE-SAT.*

Let $\mathcal{C}$ denote the class of all n-variable CNF formulas that have exactly one satisfying assignment. As an immediate corollary of Theorem [145] we have the following:

Corollary 145. *There is a constant $c > 0$ such that unless SAT $\in$ BPTIME$(t(n))$, there is no approximate uniform generation algorithm for $\mathcal{C}$ which runs in time BPTIME$(t(n^c))$ even for variation distance $\epsilon = 1/2$.*

On the other hand, it is clear that there is a linear time algorithm for the inverse approximate uniform generation problem for the class $\mathcal{C}$: simply draw a single example x and output the trivial distribution supported on that one example.

The above simple argument shows that there indeed exist classes $\mathcal{C}$ where inverse uniform generation is “easy” but approximate uniform generation is “hard”, but this example is somewhat unsatisfying, as the algorithm for inverse approximate uniform generation is trivial. It is natural to ask the following meta-question: is there a class of functions $\mathcal{C}$ such that approximation uniform generation is hard, but inverse approximate generation is easy because of a polynomial-time algorithm that "uses its samples in a non-trivial way?" In the rest of this section we give an example of such a problem.

Efficient inverse approximate uniform generation for graph automorphism. The following problem is more naturally defined in terms of a relation over combinatorial objects rather than in terms of a function and its satisfying assignments. Let us define $\mathcal{G}_n$ to be the set of all (simple undirected) graphs over vertex set $[n]$ and $\mathbb{S}_n$ to be the symmetric group over $[n]$. We define the relation $R_{\text{aut}}(G, \sigma)$ over $\mathcal{G}_n \times \mathbb{S}_n$ as follows: $R_{\text{aut}}(G, \sigma)$ holds if and only if σ is an automorphism for the graph G. (Recall that "σ is an automorphism for graph G" means that (x, y) is an edge in G if and only if $(\sigma(x), \sigma(y))$ is also an edge in G.) The inverse approximate uniform generation problem for the relation R_{aut} is then as follows: There is an unknown n-vertex graph G. The algorithm receives uniformly random samples from the set $\text{Aut}(G) := \{\sigma \in \mathbb{S}_n \ : \ R_{\text{aut}}(G, \sigma) \text{ holds } \}$. On input ϵ, δ, with probability $1 - \delta$ the algorithm must output a sampler whose output distribution is ϵ-close to the uniform distribution over $\text{Aut}(G)$.

It is easy to see that $\text{Aut}(G)$ is a subgroup of $\mathbb{S}_n$, and hence the identity permutation e_n must belong to $\text{Aut}(G)$. To understand the complexity of this problem we recall the graph isomorphism problem:

Definition 146. *GRAPH-ISOMORPHISM is defined as follows : The input is a pair of graphs $G_1, G_2 \in \mathcal{G}_n$ and the goal is to determine whether they are isomorphic.*

While it is known that GRAPH-ISOMORPHISM is unlikely to be NP-complete [128, 18], even after several decades of effort the fastest known algorithm for GRAPH-ISOMORPHISM has a running time of $2^{\tilde{O}(\sqrt{n})}$ [7]. This gives strong empirical evidence that GRAPH-ISOMORPHISM is a computationally hard problem. The following claim establishes that approximate uniform generation for R_{aut} is as hard as GRAPH-ISOMORPHISM:

Claim 147. *If there is a $t(n)$-time algorithm for approximate uniform generation for the relation R_{aut} (with error $1/3$), then there is a $O(t(2n))$-time randomized algorithm for GRAPH-ISOMORPHISM.*

Proof. Let A be the hypothesized $t(n)$-time algorithm, so A, run on input $(G, 1/2)$ where G is an n-node graph, returns an element $\sigma \in \text{Aut}(G)$ drawn from a distribution D that has $d_{\text{TV}}(D, \mathcal{U}_{\text{Aut}(G)}) \leq 1/3$. We now describe an algorithm for GRAPH-ISOMORPHISM (using A).

Let the two input graphs be H_1 and H_2 of size n. Define $G = H_1 \cup H_2$ (of size $2n$). Next, define $\text{Aut}(G)_s = \{\sigma : \sigma \in \text{Aut}(G) \text{ and } \sigma \text{ moves a vertex of } H_1 \text{ to a vertex of } H_2\}$. We observe that given H_1, H_2 and a permutation σ on the set $[2n]$, it is easy to check membership in $\text{Aut}(G)_s$. Further, using Lagrange's theorem, it can also be shown that if $|\text{Aut}(G)_s| > 1$, then $|\text{Aut}(G)_s|/|\text{Aut}(G)| \geq 1/2$.

To decide if given graphs H_1 and H_2 are isomorphic, we run A on $G = H_1 \cup H_2$. If H_1 and H_2 are not isomorphic, then $\text{Aut}(G)_s$ is empty and hence the output of A does not belong to $\text{Aut}(G)_s$

(which can be easily checked). On the other hand, if H_1 and H_2 are isomorphic, with probability at least $1/6$, the output belongs to $\mathrm{Aut}(G)_s$ (and this can again be efficiently checked). We repeat this algorithm several times to boost its completeness. $\square$

While approximate uniform generation for R_{aut} is hard, the next theorem shows that the *inverse* approximate uniform generation problem for R_{aut} is in fact easy:

Theorem 148. *There is a randomized algorithm $A^{\mathrm{aut}}_{\mathrm{inv}}$ with the following property: The algorithm takes as input $\epsilon, \delta > 0$. Given access to uniform random samples from $\mathrm{Aut}(G)$ (where G is an unknown n-node graph), $A^{\mathrm{aut}}_{\mathrm{inv}}$ runs in time $\mathrm{poly}(n, \log(1/\epsilon), \log(1/\delta))$ and with probability $1-\delta$ outputs a sampler C_{aut} with the following property : The running time of C_{aut} is $O(n \log n + \log(1/\epsilon))$ and the output distribution of C_{aut} is ϵ-close to the uniform distribution over $\mathrm{Aut}(G)$.*

Proof. The central tool in the proof is the following theorem of Alon and Roichman [2]:

Theorem 149. *[2] Let H be any group and let $h_1, \ldots, h_k$ be chosen uniformly at random from H. Consider the set $S = \cup_{i=1}^{k}\{h_i, h_i^{-1}\}$. Then, for $k = O(\log |H| + \log(1/\delta))$, with probability at least $1-\delta$ the Cayley graph (H, S) has its second largest eigenvalue at most $1/2$.*

We now describe our algorithm $A^{\mathrm{aut}}_{\mathrm{inv}}$. On input ϵ, δ it draws $k = O(n \log n + \log(1/\delta))$ permutations $g_1, \ldots, g_k$ from $\mathrm{Aut}(G)$. It computes $g_1^{-1}, \ldots, g_k^{-1}$ and sets $S = \cup_{i=1}^{k}\{g_i, g_i^{-1}\}$. The sampler C_{aut} is defined as follows: It uses its input random bits to perform a random walk on the Cayley graph $(\mathrm{Aut}(G), S)$, starting at e_n, for $T = O(n \log n + \log(1/\epsilon))$ steps; it outputs the element of H which it reaches at the end of the walk. (Note that in order to perform this random walk it is not necessary to have $\mathrm{Aut}(G)$ explicitly – it suffices to explicitly have the set S.)

The analysis is simple: we first observe that every graph G has an automorphism group of size $|\mathrm{Aut}(G)| \leq n!$. Theorem 149 then guarantees that with probability at least $1-\delta$ the Cayley graph $(\mathrm{Aut}(G), S)$ has its second eigenvalue bounded by $1/2$. Assuming that the second eigenvalue is indeed at most $1/2$, standard results in the theory of random walks on graphs imply that the distribution of the location reached at the end of the walk has variation distance at most ϵ from the uniform distribution over $\mathrm{Aut}(G)$. This concludes the proof. $\square$

Another interesting direction to pursue is to study inverse approximate uniform generation for combinatorial problems like matching and coloring as opposed to the "Boolean function satisfying assignment"–type problems that have been the main focus of this chapter. We note that preliminary arguments suggest that there is a simple efficient algorithm for inverse approximate uniform generation of perfect matchings in bipartite graphs. Similarly, preliminary arguments suggest that for the range of parameters for which the "forward" approximate uniform generation problem for colorings is known to be easy (namely, the number q of allowable colors satisfies $q > 11\Delta/6$ where Δ is the degree [147]), the inverse approximate uniform generation problem also admits an efficient algorithm. These preliminary results give rise to the question of whether there are similar *combinatorial* problems for which the complexity of the "forward" approximate uniform generation problem is not known and yet we can determine the complexity of inverse approximate uniform generation (like the group theoretic setting of Section 4.6).

Finally, for many combinatorial problems, the approximate uniform generation algorithm is to run a Markov chain on the state space. In the regimes where the uniform generation problem is hard, the Markov chain does not mix rapidly which is in turn equivalent to the existence of sparse cuts in the state space. However, an intriguing possibility arises here: If one can show that the state space can be partitioned into a small number of components such that each component has no sparse cuts, then given access to a small number of random samples from the state space (with at least one such example belonging to each component), one may be able to easily perform approximate uniform generation. Since the inverse approximate uniform generation algorithms that we consider have access to random samples, this opens the possibility of efficient approximate uniform generation algorithms in such cases. To conclude, we give an example of a natural combinatorial problem (from statistical physics) where it seems that this is essentially the situation (although we do not have a formal proof). This is the 2-D Ising model, for which the natural Glauber dynamics is known to have exponential mixing time beyond the critical temperature [109]. On the other hand, it was recently shown that even beyond the critical temperature, if one fixes the boundary to have the same spin (all positive or all negative) then the mixing time comes down from exponential to quasipolynomial [106]. While we do not know of a formal reduction, the fact that fixing the boundary to the same spin brings down the mixing time of the Glauber dynamics from exponential to quasipolynomial is "morally equivalent" to the existence of only a single sparse cut in the state space of the graph [131]. Finding other such natural examples is an intriguing goal.

Chapter 5

Future work

In this chapter, we provide some future research directions related to material presented in this thesis.

5.1 Research directions for the Chow parameters problem

In Chapter 2, we achieved a running time of $\tilde{O}(n^2 \cdot \epsilon^{\text{poly}\log(1/\epsilon)})$ for the Chow parameters problem. An obvious open problem is to improve the dependence on the error parameter ϵ in the running time and to get it down to say $\tilde{O}(n^2 \cdot \epsilon^{\text{poly}\log\log(1/\epsilon)})$ or even $\text{poly}(n/\epsilon)$.

It would also be interesting to characterize the complexity of the *exact* problem (i.e., that of finding the linear threshold given the exact value of Chow parameters, or deciding that no such game exists). We conjecture that the exact problem is intractable, namely $\sharp P$-hard.

Furthermore, in Chapter 2, we saw that the Chow parameters algorithm gives an agnostic-type learning algorithm for halfspaces. Is it possible to use the ideas here to push this bound further and get a better agnostic learning algorithm for halfspaces?

5.2 Research directions for the Inverse Shapley value problem

In Chapter 3, we achieved a running time of $\tilde{O}(n^2 \cdot 2^{\text{poly}(1/\epsilon)})$ for the Inverse Shapley value problem. An obvious open problem is to improve the dependence on the error parameter ϵ in the running time. Our algorithm is an Efficient Polynomial Time Approximation Scheme (EPTAS). Is there a Fully Polynomial Time Approximation Scheme (FPTAS), i.e., an algorithm with running time $\text{poly}(n, 1/\epsilon)$?

It would also be interesting to characterize the complexity of the *exact* problem (i.e., that of designing a weighted voting game that *exactly* achieves a given set of Shapley values, or deciding that no such game exists). We conjecture that the exact problem is intractable, namely $\sharp P$-hard.

Besides this, one can consider getting similar efficient algorithms for the inverse problems corresponding to other power indices like the Deegan-Packel index, Holler index amongst others.

5.3 Research directions for Inverse approximate uniform generation

In Chapter 4, we considered inverse problems in approximate uniform generation for a range of interesting and well-studied classes of functions including LTFs, DNFs, CNFs, polynomial threshold functions, and more. While our findings have determined the computational complexity of inverse approximate uniform generation for these classes, several interesting questions and directions remain to be pursued. We outline some of these directions below.

One natural goal is to extend our results (both positive and negative) to a wider range of function classes; we list several specific classes that seem particularly worthy of investigation. The first of these is the class of intersections of two monotone LTFs. We note that Morris and Sinclair [112] gave efficient approximate uniform generation / counting algorithms for intersections of two monotone LTFs, but on the other hand, no distribution independent PAC or SQ learning algorithm is known for this class (although quasipoly(n)-time algorithms are known if both LTFs have integer weights that are at most $\mathrm{poly}(n)$ [94]). The second class is that of poly(n)-size decision trees. Our DNF result gives a quasipoly(n/ϵ)-time inverse approximate uniform generation algorithm for this class; can this be improved to $\mathrm{poly}(n, 1/\epsilon)$? We note that in order to obtain such a result one would presumably have to bypass the "standard approach," since decision trees are not known to be PAC learnable faster than quasipoly(n/ϵ)-time under the uniform distribution on $\{-1, 1\}^n$. (We further note that while [48] gives a reduction from learning the uniform distribution over satisfying assignments of a decision tree to the problem of PAC learning decision trees under the uniform distribution, this reduction relies crucially on the assumption — implicit in the [48] framework — that the probability mass function of the hypothesis distribution can be efficiently evaluated on any input $x \in \{-1, 1\}^n$. In our framework this assumption need not hold so the [48] reduction does not apply.) Still other natural classes to investigate are context free languages (for which quasi-polynomial time uniform generation algorithms are known [60]) and various classes of branching programs. It may also be of interest to consider similar problems when the underlying measure is (say) Gaussian or log-concave.

Bibliography

[1] M. Aizenman et al. “On Bernoulli decompositions for random variables, concentration bounds, and spectral localization”. In: *Probability Theory and Related Fields* 143.1-2 (2009), pp. 219–238.

[2] N. Alon and Y. Roichman. “Random Cayley Graphs and Expanders”. In: *Random Structures and Algorithms* 5 (1994), pp. 271–284.

[3] J. Anderson, N. Goyal, and L. Rademacher. “Efficient Learning of Simplices”. In: *Conference on Learning Theory*. 2012.

[4] Sanjeev Arora and Rong Ge. “New Algorithms for Learning in Presence of Errors”. In: *ICALP 2011*. 2011, pp. 403–415.

[5] J. Aslam and S. Decatur. “Specification and simulation of statistical query algorithms for efficiency and noise tolerance”. In: *Journal of Computer and System Sciences* 56 (1998), pp. 191–208.

[6] H. Aziz, M. Paterson, and D. Leech. “Efficient algorithm for designing weighted voting games”. In: *IEEE Intl. Multitopic Conf.* 2007, pp. 1–6.

[7] László Babai. “Moderately Exponential Bound for Graph Isomorphism”. In: *Proceedings of the 1981 International FCT-Conference on Fundamentals of Computation Theory*. 1981, pp. 34–50.

[8] Y. Bachrach et al. “Approximating power indices: theoretical and empirical analysis”. In: *Autonomous Agents and Multi-Agent Systems* 20.2 (2010), pp. 105–122.

[9] J. Banzhaf. “Weighted Voting Doesn’t Work: A Mathematical Analysis”. In: *Rutgers Law Review* 19 (1965), pp. 317–343.

[10] C. R. Baugh. “Chow Parameters in Pseudothreshold Logic”. In: *SWAT (FOCS)*. 1973, pp. 49–55.

[11] S. Ben-David and E. Dichterman. “Learning with restricted focus of attention”. In: *Journal of Computer and System Sciences* 56.3 (1998), pp. 277–298.

[12] Shai Ben-David and Eli Dichterman. “Learning with Restricted Focus of Attention”. In: *J. Comput. Syst. Sci.* 56.3 (1998), pp. 277–298.

[13] I. Benjamini, G. Kalai, and O. Schramm. “Noise sensitivity of Boolean functions and applications to percolation”. In: *Inst. Hautes Études Sci. Publ. Math.* 90 (1999), pp. 5–43.

[14] A. Birkendorf et al. “On restricted-focus-of-attention learnability of Boolean functions”. In: *Machine Learning* 30 (1998), pp. 89–123.

[15] A. Blum et al. “A polynomial time algorithm for learning noisy linear threshold functions”. In: *Algorithmica* 22.1/2 (1997), pp. 35–52.

[16] Avrim Blum, Adam Kalai, and Hal Wasserman. “Noise-tolerant learning, the parity problem, and the statistical query model”. In: *Journal of the ACM* 50.4 (July 2003), pp. 506–519.

[17] Anselm Blumer et al. “Learnability and the Vapnik-Chervonenkis dimension”. In: *Journal of the ACM* 36.4 (1989), pp. 929–965.

[18] R. Boppana, J. Håstad, and S. Zachos. “Does co-NP have short interactive proofs?” In: *Information Processing Letters* 25 (1987), pp. 127–132.

[19] G. Bresler, E. Mossel, and A. Sly. “Reconstruction of Markov Random Fields from Samples: Some Observations and Algorithms”. In: *APPROX-RANDOM*. 2008, pp. 343–356.

[20] J. Bruck. “Harmonic analysis of polynomial threshold functions”. In: *SIAM Journal on Discrete Mathematics* 3.2 (1990), pp. 168–177.

[21] F. Carreras. “On the design of voting games”. In: *Mathematical Methods of Operations Research* 59.3 (2004), pp. 503–515.

[22] M. Cheraghchi et al. “Approximating Linear Threshold Predicates”. In: *13th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems — APPROX 2010*. 2010, pp. 110–123.

[23] C. K. Chow. “On the characterization of threshold functions”. In: *SWCT (FOCS)*. 1961, pp. 34–38.

[24] C.K. Chow. “On the characterization of threshold functions”. In: *Proc. 2nd FOCS*. 1961, pp. 34–38.

[25] R. Cramer and V. Shoup. “Signature schemes based on the strong RSA assumption”. In: *ACM Trans. Inf. Syst. Secur.* 3.3 (2000), pp. 161–185.

[26] C. Daskalakis, I. Diakonikolas, and R.A. Servedio. “Learning k-modal distributions via testing”. In: *SODA*. 2012, pp. 1371–1385.

[27] C. Daskalakis, I. Diakonikolas, and R.A. Servedio. “Learning Poisson Binomial Distributions”. In: *STOC*. 2012, pp. 709–728.

[28] C. Daskalakis, E. Mossel, and S. Roch. “Optimal phylogenetic reconstruction”. In: *STOC*. 2006, pp. 159–168.

[29] A. De, I. Diakonikolas, and R. Servedio. “Inverse problems in approximate uniform generation”. Available at http://arxiv.org/pdf/1211.1722v1.pdf. 2012.

[30] A. De et al. “Near-optimal solutions for the Chow Parameters Problem and low-weight approximation of halfspaces”. To appear in *STOC 2012*. 2012.

[31] A. De et al. “Near-optimal solutions for the Chow Parameters Problem and low-weight approximation of halfspaces”. In: *Proc. 44th ACM Symposium on Theory of Computing (STOC)*. 2012, pp. 729–746.

[32] Anindya De, Ilias Diakonikolas, and Rocco A. Servedio. “The Inverse Shapley Value Problem”. In: *ICALP (1)*. 2012, pp. 266–277.

[33] S. Decatur. “Statistical queries and faulty PAC oracles”. In: *Proceedings of the Sixth Workshop on Computational Learning Theory*. Santa Cruz, California, 1993, pp. 262–268.

[34] J. Deegan and E. Packel. “A New Index of Power for Simple n-Person Games”. In: *International Journal of Game Theory* 7 (1978), pp. 113–123.

[35] François Denis, Rémi Gilleron, and Fabien Letouzey. “Learning from positive and unlabeled examples”. In: *Theor. Comput. Sci.* 348.1 (2005), pp. 70–83.

[36] M. Dertouzos. *Threshold Logic: A Synthesis Approach*. Cambridge, MA: MIT Press, 1965.

[37] L. Devroye and G. Lugosi. *Combinatorial methods in density estimation*. Springer: Springer Series in Statistics, 2001.

[38] I. Diakonikolas and R. Servedio. “Improved approximation of linear threshold functions”. In: *Proc. 24th Annual IEEE Conference on Computational Complexity (CCC)*. 2009, pp. 161–172.

[39] I. Diakonikolas and R. Servedio. “Improved approximation of linear threshold functions”. In: *Proc. 24th CCC*. 2009, pp. 161–172.

[40] I. Diakonikolas et al. “Bounded independence fools halfspaces”. In: *SIAM J. on Comput.* 39.8 (2010), pp. 3441–3462.

[41] Yevgeniy Dodis et al. “Message Authentication, Revisited”. In: *ECRYPT12*. 2012, pp. 355–374.

[42] P. Dubey and L.S. Shapley. “Mathematical properties of the Banzhaf power index”. In: *Mathematics of Operations Research* 4 (1979), pp. 99–131.

[43] M. Dyer. “Approximate Counting by Dynamic Programming”. In: *STOC*. 2003, pp. 693–699.

[44] E. Einy and E. Lehrer. “Regular simple games”. In: *International Journal of Game Theory* 18 (1989), pp. 195–207.

[45] C.C. Elgot. “Truth functions realizable by single threshold organs”. In: *Proceedings of the Symposium on Switching Circuit Theory and Logical Design (FOCS)*. 1960, pp. 225–245.

[46] E. Elkind et al. “Computational complexity of weighted voting games”. In: *AAAI*. 2007, pp. 718–723.

[47] S. Fatima, M. Wooldridge, and N. Jennings. “An Anytime Approximation Method for the Inverse Shapley Value Problem”. In: *AAMAS’08*. 2008, pp. 935–942.

[48] Jon Feldman, Ryan O’Donnell, and Rocco A. Servedio. “Learning Mixtures of Product Distributions over Discrete Domains”. In: *SIAM J. Comput.* 37.5 (2008), pp. 1536–1564.

[49] V. Feldman. “Distribution-Specific Agnostic Boosting”. In: *Proceedings of Innovations in Computer Science*. 2010, pp. 241–250.

[50] V. Feldman. “Learning DNF Expressions from Fourier Spectrum”. In: *Proceedings of Conference on Learning Theory*. 2012.

[51] Vitaly Feldman et al. “Agnostic Learning of Monomials by Halfspaces Is Hard”. In: *FOCS*. 2009, pp. 385–394.

[52] W. Feller. *An introduction to probability theory and its applications*. John Wiley & Sons, 1968.

[53] D. Felsenthal and M. Machover. “A priori voting power: what is it all about?” In: *Political Studies Review* 2.1 (2004), pp. 1–23.

[54] M. Fischlin. “The Cramer-Shoup Strong-RSA Signature Scheme Revisited”. In: *Public Key Cryptography - PKC 2003*. 2003, pp. 116–129.

[55] J. Freixas. “Different ways to represent weighted majority games”. In: *Top (Journal of the Spanish Society of Statistics and Operations Research)* 5.2 (1997), pp. 201–212.

[56] A. Frieze, M. Jerrum, and R. Kannan. “Learning Linear Transformations”. In: *focs1996*. 1996, pp. 359–368.

[57] P. Goldberg. “A Bound on the Precision Required to Estimate a Boolean Perceptron from its Average Satisfying Assignment”. In: *SIAM Journal on Discrete Mathematics* 20 (2006), pp. 328–343.

[58] P. Goldberg. “A Bound on the Precision Required to Estimate a Boolean Perceptron from its Average Satisfying Assignment”. In: *SIDMA* 20 (2006), pp. 328–343.

[59] Oded Goldreich. *Foundations of Cryptography-volume 2*. Cambridge: Cambridge University Press, 2004. ISBN: 0521791723.

[60] V. Gore et al. “A quasi-polynomial-time algorithm for sampling words from a context-free language”. In: *Inf. Comput.* 134.1 (1997), pp. 59–74.

[61] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Vol. 2. Springer, 1988.

[62] Venkat Guruswami and Prasad Raghavendra. “Hardness of Learning Halfspaces with Noise”. In: *SIAM J. Comput.* 39.2 (2009), pp. 742–765.

[63] G. Halász. “Estimates for the concentration function of combinatorial number theory and probability”. In: *Period. Math. Hungar.* 8.3 (1977), pp. 197–211.

[64] J. Håstad. “On the size of weights for threshold gates”. In: *SIAM Journal on Discrete Mathematics* 7.3 (1994), pp. 484–492.

[65] David Haussler. “Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications”. In: *Information and Computation* 100.1 (1992), pp. 78–150.

[66] Thomas P. Hayes and Eric Vigoda. “A Non-Markovian Coupling for Randomly Sampling Colorings”. In: *FOCS*. 2003, pp. 618–627.

[67] S. Hohenberger and B. Waters. “Constructing Verifiable Random Functions with Large Input Spaces”. In: *EUROCRYPT*. 2010, pp. 656–672.

[68] M.J. Holler. “Forming coalitions and measuring voting power”. In: *Political studies* 30 (1982), pp. 262–271.

[69] S.L. Hurst. “The application of Chow Parameters and Rademacher-Walsh matrices in the synthesis of binary functions”. In: *The Computer Journal* 16 (2 1973), pp. 165–173.

[70] R. Impagliazzo. “Hard-core distributions for somewhat hard problems”. In: *Proc. 36th FOCS*. Milwaukee, Wisconsin, 1995, pp. 538–545.

[71] Russell Impagliazzo. “Hard-Core Distributions for Somewhat Hard Problems”. In: *Proc. 36th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, 1995, pp. 538–545.

[72] M. Jerrum, L. G. Valiant, and V. V. Vazirani. “Random Generation of Combinatorial Structures from a Uniform Distribution”. In: *Theor. Comput. Sci.* 43 (1986), pp. 169–188.

[73] Mark Jerrum. “A Very Simple Algorithm for Estimating the Number of k-Colorings of a Low-Degree Graph”. In: *Random Struct. Algorithms* 7.2 (1995), pp. 157–166.

[74] Mark Jerrum and Alistair Sinclair. “Approximating the Permanent”. In: *SIAM J. Comput.* 18.6 (1989), pp. 1149–1178.

[75] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. “A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries”. In: *J. ACM* 51.4 (2004), pp. 671–697.

[76] S. Jukna. *Extremal combinatorics with applications in computer science*. Springer, 2001.

[77] A. Kalai et al. “Agnostically Learning Halfspaces”. In: *Proceedings of the 46th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2005, pp. 11–20.

[78] Adam Kalai et al. “Agnostically Learning Halfspaces”. In: *SIAM Journal on Computing* 37.6 (2008). Special issue for FOCS 2005., pp. 1777–1805.

[79] G. Kalai and S. Safra. “Threshold phenomena and influence”. In: *Computational Complexity and Statistical Physics*. Oxford University Press, 2006, pp. 25–60.

[80] K.R. Kaplan and R.O. Winder. “Chebyshev approximation and threshold functions”. In: *IEEE Trans. Electronic Computers* EC-14 (1965), pp. 315–325.

[81] R. M. Karp, M. Luby, and N. Madras. “Monte-Carlo Approximation Algorithms for Enumeration Problems”. In: *Journal of Algorithms* 10.3 (1989), pp. 429–448.

[82] R.M. Karp and M. Luby. “Monte-Carlo algorithms for enumeration and reliability problems”. In: *FOCS*. 1983, pp. 56–64.

[83] P. Kaszerman. “A geometric test-synthesis procedure for a threshold device”. In: *Information and Control* 6.4 (1963), pp. 381–398.

[84] Jonathan Katz, Ji Sun Shin, and Adam Smith. “Parallel and Concurrent Security of the HB and HB^+ Protocols”. In: *Journal of Cryptology* 23.3 (2010), pp. 402–421.

[85] M. Kearns. “Efficient noise-tolerant Learning from statistical queries”. In: *Journal of the ACM* 45.6 (1998), pp. 983–1006.

[86] M. Kearns, R. Schapire, and L. Sellie. “Toward Efficient Agnostic Learning”. In: *Machine Learning* 17.2/3 (1994), pp. 115–141.

[87] M. Kearns et al. “On the learnability of discrete distributions”. In: *Proceedings of the 26th Symposium on Theory of Computing*. 1994, pp. 273–282.

[88] Michael J. Kearns, Robert E. Schapire, and Linda Sellie. “Toward Efficient Agnostic Learning”. In: *Machine Learning* 17.2-3 (1994), pp. 115–141.

[89] B. de Keijzer, T. Klos, and Y. Zhang. “Enumeration and exact design of weighted voting games”. In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems : volume 1 - Volume 1*. AAMAS ’10. 2010, pp. 391–398.

[90] Bart de Keijzer. “A Survey on the Computation of Power Indices”. 2008.

[91] Bart de Keijzer, Tomas Klos, and Yingqian Zhang. “Enumeration and exact design of weighted voting games”. In: *AAMAS*. 2010, pp. 391–398.

[92] L.G. Khachiyan. “Polynomial algorithms in linear programming”. In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53 –72.

[93] Eike Kiltz et al. “Efficient Authentication from Hard Learning Problems”. In: *ECRYPT11*. 2011, pp. 7–26.

[94] A. Klivans, R. O’Donnell, and R. Servedio. “Learning intersections and thresholds of halfspaces”. In: *Journal of Computer & System Sciences* 68.4 (2004), pp. 808–840.

[95] A. Klivans and R. Servedio. “Learning DNF in time $2^{\tilde{O}(n^{1/3})}$”. In: *Journal of Computer & System Sciences* 68.2 (2004), pp. 303–318.

[96] S. Kurz. “On the inverse power index problem”. In: *Optimization* (2011).

[97] S. Kurz. “On the inverse power index problem”. In: *Optimization* 61.8 (2012), pp. 989–1011.

[98] S. Kurz and S. Napel. “Heuristic and exact solutions to the inverse power index problem for small voting bodies”. Available as arxiv report http://arxiv.org/abs/1202.6245. 2012.

[99] E. Lapidot. “The counting vector of a simple game”. In: *Proceedings of the AMS* 31 (1972), pp. 228–231.

[100] A. Laruelle and M. Widgren. “Is the allocation of voting power among EU states fair?” In: *Public Choice* 94 (1998), pp. 317–339.

[101] D. Leech. “Computing Power Indices for Large Voting Games”. In: *Management Science* 49.6 (2003).

[102] D. Leech. “Designing the voting system for the EU Council of Ministers”. In: *Public Choice* 113 (2002), pp. 437–464.

[103] D. Leech. “Power Indices as an Aid to Institutional Design: the Generalised Apportionment Problem”. In: *Yearbook on New Political Economy*. Ed. by M. Holler et al. 2003.

[104] D. Leech. “Voting power in the governance of the International Monetary Fund”. In: *Annals of Operations Research* 109 (2002), 375??97.

[105] L. Lovász and S. Vempala. “Logconcave Functions: Geometry and Efficient Sampling Algorithms”. In: *Prooceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. 2003, pp. 650–659.

[106] E. Lubetzky et al. “Quasi-polynomial mixing of the 2D stochastic Ising model with plus boundary up to criticality”. In: *Journal of the European Mathematical Society* (2013).

[107] Anna Lysyanskaya. “Unique Signatures and Verifiable Random Functions from the DH-DDH Separation”. In: *Advances in Cryptology — (CRYPTO 2002)*. Vol. 2442. 2002, pp. 597–612.

[108] W. Maass and G. Turan. “How fast can a threshold gate learn?” In: *Computational Learning Theory and Natural Learning Systems*. 1994, pp. 381–414.

[109] F. Martinelli. “Lectures on Glauber dynamics for discrete spin models”. In: *Lectures on Probability Theory and Statistics*. Vol. 1717. Lecture Notes in Mathematics. Springer, 1998, pp. 93–191.

[110] K. Matulef et al. “Testing Halfspaces”. In: *SIAM J. on Comput.* 39.5 (2010), pp. 2004–2047.

[111] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. “Verifiable Random Functions”. In: *Proc. 40th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1999, pp. 120–130.

[112] Ben Morris and Alistair Sinclair. “Random Walks on Truncated Cubes and Sampling 0-1 Knapsack Solutions”. In: *SIAM J. Comput.* 34.1 (2004), pp. 195–226.

[113] E. Mossel. “Distorted Metrics on Trees and Phylogenetic Forests”. In: *IEEE/ACM Trans. Comput. Biology Bioinform.* 4.1 (2007).

[114] S. Muroga. *Threshold logic and its applications*. New York: Wiley-Interscience, 1971.

[115] S. Muroga, I. Toda, and M. Kondo. “Majority decision functions of up to six variables”. In: *Math. Comput.* 16 (1962), pp. 459–472.

[116] S. Muroga, I. Toda, and S. Takasu. “Theory of majority switching elements”. In: *J. Franklin Institute* 271 (1961), pp. 376–418.

[117] S. Muroga, T. Tsuboi, and C.R. Baugh. *Enumeration of Threshold Functions of Eight Variables*. Tech. rep. 245. Univ. of Illinois, Urbana, 1967.

[118] A. M. Odlyzko. “On subspaces spanned by random selections of ± 1 vectors”. In: *J. Comb. Theory, Ser. A* 47.1 (1988), pp. 124–133.

[119] R. O’Donnell and R. Servedio. “The Chow Parameters Problem”. In: *Proc. 40th STOC*. 2008, pp. 517–526.

[120] R. O'Donnell and R. Servedio. "The Chow Parameters Problem". In: *SIAM J. on Comput.* 40.1 (2011), pp. 165–199.

[121] G. Owen. "Multilinear extensions of games". In: *Management Science* 18.5 (1972). Part 2, Game theory and Gaming, pp. 64–79.

[122] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[123] L.S. Penrose. "The Elementary Statistics of Majority Voting". In: *Journal of the Royal Statistical Society* 109.1 (1946), pp. 53–57.

[124] Krzysztof Pietrzak. "Subspace LWE". In: *Theory of Cryptography Conference*. 2012, pp. 548–563.

[125] A.E. Roth, ed. *The Shapley value*. University of Cambridge Press, 1988.

[126] V.P. Roychowdhury et al. "Vector analysis of threshold functions". In: *Information and Computation* 120.1 (1995), pp. 22–31.

[127] Thomas J. Schaefer. "The Complexity of Satisfiability Problems". In: *STOC*. 1978, pp. 216–226.

[128] Uwe Schöning. "Graph Isomorphism is in the Low Hierarchy". In: *J. Comp. Sys. Sci.* 37.3 (1988), pp. 312–323.

[129] R. Servedio. "Every linear threshold function has a low-weight approximator". In: *Comput. Complexity* 16.2 (2007), pp. 180–209.

[130] L. Shapley and M. Shubik. "A Method for Evaluating the Distribution of Power in a Committee System". In: *American Political Science Review* 48 (1954), pp. 787–792.

[131] A. Sinclair. " " Personal communication. 2012.

[132] Alistair Sinclair and Mark Jerrum. "Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains". In: *Inf. Comput.* 82.1 (1989), pp. 93–133.

[133] M. Sipser. "A complexity-theoretic approach to randomness". In: *STOC*. 1983, pp. 330–335.

[134] L. Stockmeyer. "The complexity of approximate counting". In: *STOC*. 1983, pp. 118–126.

[135] K. Takamiya and A. Tanaka. *Computational complexity in the design of voting games*. Tech. rep. 653. The Institute of Social and Economic Research, Osaka University, 2006.

[136] A. Taylor and W. Zwicker. "A Characterization of Weighted Voting". In: *Proceedings of the AMS* 115.4 (1992), pp. 1089–1094.

[137] Craig A. Tovey. "A simplified NP-complete satisfiability problem". In: *Discrete Applied Mathematics* 8.1 (1984), pp. 85–89.

[138] L. Trevisan, M. Tulsiani, and S. Vadhan. *Regularity, Boosting and Efficiently Simulating every High Entropy Distribution*. Tech. rep. 103. Conference version in Proc. CCC 2009. ECCC, 2008.

[139] Luca Trevisan, Madhur Tulsiani, and Salil P. Vadhan. "Regularity, Boosting, and Efficiently Simulating Every High-Entropy Distribution". In: *IEEE Conference on Computational Complexity*. 2009, pp. 126–136.

[140] T.Tao and V. H. Vu. "Inverse Littlewood-Offord theorems and the condition number of random discrete matrices". In: *Annals of Mathematics* 169 (2009), pp. 595–632.

[141] P. Vaidya. "A new algorithm for minimizing convex functions over convex sets". In: *Proceedings of the Thirtheth Symposium on Foundations of Computer Science*. 1989, pp. 338–343.

[142] P. M. Vaidya. "A new algorithm for minimizing convex functions over convex sets". In: *Math. Prog.* 73.3 (1996), pp. 291–341.

[143] G. Valiant. "Finding Correlations in Subquadratic Time, with Applications to Learning Parities and Juntas". In: *FOCS*. 2012.

[144] Leslie G. Valiant. "A Theory of the Learnable". In: *Commun. ACM* 27.11 (1984), pp. 1134–1142.

[145] Leslie G. Valiant and V. V. Vazirani. "NP Is as Easy as Detecting Unique Solutions". In: *Theoretical Computer Science* 47 (1986), pp. 85–93.

[146] Karsten A. Verbeurgt. "Learning DNF under the uniform distribution in quasi-polynomial time". In: *Conference on Learning Theory*. 1990, pp. 314–326.

[147] Eric Vigoda. "Improved Bounds for Sampling Colorings". In: *FOCS*. 1999, pp. 51–59.

[148] Thomas Watson. *The complexity of estimating Min-entropy*. Tech. rep. 70. Electronic Colloquium in Computational Complexity, 2012.

[149] R.O. Winder. "Chow parameters in threshold logic". In: *Journal of the ACM* 18.2 (1971), pp. 265–289.

[150] R.O. Winder. *Threshold Functions Through* $n = 7$. Tech. rep. 7. Air Force Cambridge Research Laboratories, 1964.

[151] R.O. Winder. "Threshold Gate Approximations Based on Chow Parameters". In: *IEEE Transactions on Computers* (1969), pp. 372–375.

[152] R.O. Winder. "Threshold logic in artificial intelligence". In: *Artificial Intelligence* IEEE Publication S-142 (1963), pp. 107–128.

[153] M. Zuckerman et al. "Manipulating the Quota in Weighted Voting Games". In: *AAAI*. 2008.

Appendix A

Missing proofs from Chapter 2

A.1 Near-Optimality of Lemma 22

The following lemma shows that in any statement like Lemma 22 in which the hyperplane $\mathbf{H}'$ passes through *all* the points in S, the distance bound on β can be no larger than $n^{-1/2}$ as a function of n. This implies that the result obtained by taking $\kappa = 1/2^{n+1}$ in Lemma 22, which gives a distance bound of $n^{-(1/2+o(1))}$ as a function of n, is optimal up to the $o(1)$ in the exponent.

Lemma 150. *Fix $\epsilon > 8n^{-1/2}$. There is a hyperplane $\mathbf{H} \in \mathbb{R}^n$ and a set $S \subseteq \{-1,1\}^n$ such that $|S| \geq \frac{\epsilon}{8}2^n$ and the following properties both hold:*

- *For every $x \in S$ we have $d(x, \mathbf{H}) \leq 2\epsilon n^{-1/2}$; and*
- *There is no hyperplane $\mathbf{H}'$ which passes through all the points in S.*

Proof. Without loss of generality, let us assume $K = 4/\epsilon^2$ is an even integer; note that by assumption $K < n/2$. Now let us define the hyperplane $\mathbf{H}$ by

$$\mathbf{H} = \left\{ x \in \mathbb{R}^n : (x_1 + \ldots + x_K) + \frac{2(x_{K+1} + \ldots + x_n)}{(n-K)} = 0 \right\}$$

Let us define $S = \{x \in \{-1,1\}^n : d(x, \mathbf{H}) \leq 4/\sqrt{K(n-K)}\}$. It is easy to verify that every $x \in S$ indeed satisfies $d(x, \mathbf{H}) \leq 2\epsilon n^{-1/2}$ as claimed. Next, let us define A as follows:

$$A = \{x \in \{-1,1\}^n : x_1 + \ldots + x_K = 0$$

and

$$|x_{K+1} + \ldots + x_n| \leq 2\sqrt{n-K}\}.$$

It is easy to observe that $A \subseteq S$. Also, we have

$$\mathop{\mathbf{P}r}_{x_1,\ldots,x_K} [x_1 + \ldots + x_K = 0] \geq (2\sqrt{K})^{-1}$$

and

$$\mathop{\mathbf{Pr}}_{x_{K+1},\dots,x_n}[|x_{K+1}+\ldots+x_n| \le 2\sqrt{n-K}] \ge 1/2.$$

Hence we have that $|S| \ge \epsilon 2^n/8$. We also observe that the point $z \in \{-1,1\}^n$ defined as

$$z := (1,1,\underbrace{1,-1,\ldots,1,-1}_{K-2},-1,\ldots,-1) \tag{A.1}$$

(whose first two coordinates are 1, next $K-2$ coordinates alternate between 1 and -1, and final $n-K$ coordinates are -1) lies on $\mathbf{H}$ and hence $z \in S$.

We next claim that the dimension of the affine span of the points in $A \cup z$ is n. This obviously implies that there is no hyperplane which passes through all points in $A \cup z$, and hence no hyperplane which passes through all points in S. Thus to prove the lemma it remains only to prove the following claim:

Claim 151. *The dimension of the affine span of the elements of $A \cup z$ is n.*

To prove the claim, we observe that if we let Y denote the affine span of elements in $A \cup z$ and Y' denote the linear space underlying Y, then it suffices to show that the dimension of Y' is n. Each element of Y' is obtained as the difference of two elements in Y.

First, let $y \in \{-1,1\}^n$ be such that

$$\sum_{i \le K} y_i = \sum_{K+1 \le i \le n} y_i = 0.$$

Let $y^{\oplus i} \in \{-1,1\}^n$ be obtained from y by flipping the i-th bit. For each $i \in \{K+1,\ldots,n\}$ we have that y and $y^{\oplus i}$ are both in A, so subtracting the two elements, we get that the basis vector e_i belongs to Y' for each $i \in \{K+1,\ldots,n\}$.

Next, let $i \ne j \le K$ be positions such that $y_i = 1$ and $y_j = -1$. Let y^{ij} denote the vector which is the same as y except that the signs are flipped at coordinates i and j. Since y^{ij} belongs to A, by subtracting y from y^{ij} we get that for every vector e_{ij} ($i \ne j \le K$) which has 1 in coordinate i, -1 in coordinate j, and 0 elsewhere, the vector e_{ij} belongs to Y'.

The previous two paragraphs are easily seen to imply that the linear space Y' contains all vectors $x \in \mathbb{R}^n$ that satisfy the condition $x_1 + \cdots + x_K = 0$. Thus to show that the dimension of Y' is n, it suffices to exhibit any vector in Y' that does not satisfy this condition. But it is easy to see that the vector $y - z$ (where z is defined in (A.1)) is such a vector. This concludes the proof of the claim and of Lemma 150. □

A.2 Useful extensions of Goldberg's theorems

To allow an application of Lemma 22 in affine subspaces of $\mathbb{R}^n$ we require an extension of Theorem 21 (Theorem 3 of [57]) which roughly speaking is as follows: the hypothesis is that not only does the set $S \subset \{-1,1\}^n$ lie close to hyperplane $\mathbf{H}$ but so also does a (small) set R of points in

$\{0,1\}^n$; and the conclusion is that not only does "almost all" of S (the subset S^*) lie on $\mathbf{H}'$ *but so also does all of* R. To obtain this extension we need a corresponding extension of an earlier result of Goldberg (Theorem 2 of [57]), which he uses to prove his Theorem 3; similar to our extension of Theorem 21 our extension of Theorem 2 of [57] deals with points from both $\{-1,1\}^n$ and $\{0,1\}^n$. The simplest approach we have found to obtain our desired extension of Theorem 2 of [57] uses the "Zeroth Inverse Theorem" of Tao and Vu [140]. We begin with a useful definition from their paper:

Definition 152. *Given a vector $w = (w_1, \dots, w_k)$ of real values, the* cube $S(w)$ *is the subset of* $\mathbb{R}$ *defined as* [1]

$$S(w) = \left\{ \sum_{i=1}^{k} \epsilon_i w_i : (\epsilon_1, \dots, \epsilon_n) \in \{-1,0,1\}^n \right\}.$$

The "Zeroth Inverse Theorem" of [140] is as follows:

Theorem 153. *Suppose $w \in \mathbb{R}^n$, $d \in \mathbb{N}$ and $\theta \in \mathbb{R}$ satisfy $\mathbf{P}r_{x\in\{-1,1\}^n}[w \cdot x = \theta] > 2^{-d-1}$. Then there exists a d-element subset $A = \{i_1, \dots, i_d\} \subset [n]$ such that for $v = (w_{i_1}, \dots, w_{i_d})$ we have $\{w_1, \dots, w_n\} \subseteq S(v)$.*

For convenience of the reader, we include the proof here.

Proof of Theorem 153. Towards a contradiction, assume that there is no $v = (w_{i_1}, \dots, w_{i_d})$ such that $\{w_1, \dots, w_n\} \subseteq S(v)$. Then an obvious greedy argument shows that there are distinct integers $i_1, \dots, i_{d+1} \in [n]$ such that $w_{i_1}, \dots, w_{i_{d+1}}$ is *dissociated*, i.e. there does not exist $j \in [n]$ and $\epsilon_i \in \{-1,0,1\}$ such that $w_j = \sum_{i \neq j} \epsilon_i w_i$.

Let $v = (w_{i_1}, \dots, w_{i_{d+1}})$. By an averaging argument, it is easy to see that if $\mathbf{P}r_{x\in\{-1,1\}^n}[w \cdot x = \theta] > 2^{-d-1}$, then $\exists \nu \in \mathbb{R}$ such that $\mathbf{P}r_{x\in\{-1,1\}^{d+1}}[v \cdot x = \nu] > 2^{-d-1}$. By the pigeon hole principle, this means that there exist $x, y \in \{-1,1\}^{d+1}$ such that $x \neq y$ and $v \cdot ((x-y)/2) = 0$. Since entries of $(x-y)/2$ are in $\{-1,0,1\}$, and not all the entries in $(x-y)/2$ are zero, this means that v is not dissociated resulting in a contradiction. $\square$

Armed with this result, we now prove the extension of Goldberg's Theorem 2 that we will need later:

Theorem 154. *Let $w \in \mathbb{R}^n$ have $\|w\|_2 = 1$ and let $\theta \in \mathbb{R}$ be such that $\mathbf{P}r_{x\in\{-1,1\}^n}[w \cdot x = \theta] = \alpha$. Let $\mathbf{H}$ denote the hyperplane $\mathbf{H} = \{x \in \mathbb{R}^n \mid w \cdot x = \theta\}$. Suppose that* $\mathrm{span}(\mathbf{H} \cap (\{-1,1\}^n \cup \{0,1\}^n)) = \mathbf{H}$, *i.e. the affine span of the points in $\{-1,1\}^n \cup \{0,1\}^n$ that lie on $\mathbf{H}$ is $\mathbf{H}$. Then all entries of w are integer multiples of $f(n,\alpha)^{-1}$, where*

$$f(n,\alpha) \leq (2n)^{\lfloor \log(1/\alpha) \rfloor + 3/2} \cdot (\lfloor \log(1/\alpha) \rfloor)!$$

[1] In [140] the cube is defined only allowing $\epsilon_i \in \{-1,1\}$ but this is a typographical error; their proof uses the $\epsilon_i \in \{-1,0,1\}$ version that we state.

Proof. We first observe that $w \cdot (x-y) = 0$ for any two points x, y that both lie on $\mathbf{H}$. Consider the system of homogeneous linear equations in variables $w'_1, \dots, w'_n$ defined by

$$w' \cdot (x-y) = 0 \qquad \text{for all } x, y \in \mathbf{H} \cap (\{-1,1\}^n \cup \{0,1\}^n). \tag{A.2}$$

Since $\operatorname{span}(\mathbf{H} \cap (\{-1,1\}^n \cup \{0,1\}^n))$ is by assumption the entire hyperplane $\mathbf{H}$, the system (A.2) must have rank $n-1$; in other words, every solution w' that satisfies (A.2) must be some rescaling $w' = cw$ of the vector w defining $\mathbf{H}$.

Let A denote a subset of $n-1$ of the equations comprising (A.2) which has rank $n-1$ (so any solution to A must be a vector $w' = cw$ as described above). We note that each coefficient in each equation of A lies in $\{-2,-1,0,1,2\}$. Let us define $d = \lfloor \log(1/\alpha) \rfloor + 1$. By Theorem 153, there is some $w_{i_1}, \dots, w_{i_{d'}}$ with $d' \le d$ such that for $v \overset{\text{def}}{=} (w_{i_1}, \dots, w_{i_{d'}})$, we have $\{w_1, \dots, w_n\} \subseteq S(v)$; in other words, for all $j \in [n]$ we have $w_j = \sum_{\ell=1}^{d'} \epsilon_{\ell,j} w_{i_\ell}$ where each $\epsilon_{\ell,j}$ belongs to $\{-1,0,1\}$. Substituting these relations into the system A, we get a new system of homogenous linear equations, of rank $d'-1$, in the variables $w'_{i_1}, \dots, w'_{i_{d'}}$, where all coefficients of all variables in all equations of the system are integers of magnitude at most $2n$.

Let M denote a subset of $d'-1$ equations from this new system which has rank $d'-1$. In other words, viewing M as a $d' \times (d'-1)$ matrix, we have the equation $M \cdot v^T = 0$ where all entries in the matrix M are integers in $[-2n, 2n]$. Note that at least one of the values $w_{i_1}, \dots, w_{i_{d'}}$ is non-zero (for if all of them were 0, then since $\{w_1, \dots, w_n\} \subseteq S(v)$ it would have to be the case that $w_1 = \cdots = w_n = 0$.). Without loss of generality we may suppose that w_{i_1} has the largest magnitude among $w_{i_1}, \dots, w_{i_{d'}}$. We now fix the scaling constant c, where $w' = cw$, to be such that $w'_{i_1} = 1$. Rearranging the system $M(cv)^T = M(1, w'_{i_2}, \dots, w'_{i_{d'}})^T = 0$, we get a new system of $d'-1$ linear equations $M'(w'_{i_2}, \dots, w'_{i_{d'}})^T = b$ where M' is a $(d'-1) \times (d'-1)$ matrix whose entries are integers in $[-2n, 2n]$ and b is a vector whose entries are integers in $[-2n, 2n]$.

We now use Cramer's rule to solve the system

$$M'(w'_{i_2}, \dots, w'_{i_{d'}})^T = b.$$

This gives us that $w'_{i_j} = \det(M'_j)/\det(M')$ where M'_j is the matrix obtained by replacing the j^{th} column of M' by b. So each w'_{i_j} is an integer multiple of $1/\det(M')$ and is bounded by 1 (by our earlier assumption about w_{i_1} having the largest magnitude). Since $\{w'_1, \dots, w'_n\} \subseteq S(v)$, we get that each value w'_i is an integer multiple of $1/\det(M')$, and each $|w'_i| \le n$. Finally, since M' is a $(d'-1) \times (d'-1)$ matrix where every entry is an integer of magnitude at most $2n$, we have that $|\det(M')| \le (2n)^{d'-1} \cdot (d'-1)! \le (2n)^{d-1} \cdot (d-1)!$. Moreover, the ℓ_2 norm of the vector w' is bounded by $n^{3/2}$. So renormalizing (dividing by c) to obtain the unit vector w back from $w' = cw$, we see that every entry of w is an integer multiple of $1/N$, where N is a quantity at most $(2n)^{d+1/2} \cdot d!$. Recalling that $d = \lfloor \log(1/\alpha) \rfloor + 1$, the theorem is proved. $\square$

We next prove the extension of Theorem 3 from [57] that we require. The proof is almost identical to the proof in [57] except for the use of Theorem 154 instead of Theorem 2 from [57] and a few other syntactic changes. For the sake of clarity and completeness, we give the complete proof here.

Theorem 155. *Given any hyperplane $\mathbf{H}$ in $\mathbb{R}^n$ whose β-neighborhood contains a subset S of vertices of $\{-1,1\}^n$ where $S = \alpha \cdot 2^n$, there exists a hyperplane which passes through all the points of $(\{-1,1\}^n \cup \{0,1\}^n)$ that are contained in the β-neighborhood of $\mathbf{H}$ provided that*

$$0 \le \beta \le \left((2/\alpha) \cdot n^{5+\lfloor \log(n/\alpha)\rfloor} \cdot (2 + \lfloor \log(n/\alpha)\rfloor)!\right)^{-1}.$$

Before giving the proof, we note that the hypothesis of our theorem is the same as the hypothesis of Theorem 3 of [57]. The only difference in the conclusion is that while Goldberg proves that all points of $\{-1,1\}^n$ in the β-neighborhood of $\mathbf{H}$ lie on the new hyperplane, we prove this for all the points of $(\{-1,1\}^n \cup \{0,1\}^n)$ in the β-neighborhood of $\mathbf{H}$.

Proof. Let $\mathbf{H} = \{x \mid w \cdot x - t = 0\}$ with $\|w\| = 1$. Also, let $S = \{x \in \{-1,1\}^n \mid d(x,\mathbf{H}) \le \beta\}$ and $S' = \{x \in (\{-1,1\}^n \cup \{0,1\}^n) \mid d(x,\mathbf{H}) \le \beta\}$. For any $x \in S'$ we have that $w \cdot x \in [t-\beta, t+\beta]$. Following [57] we create a new weight vector $w' \in \mathbb{R}^n$ by rounding each coordinate w_i of w to the nearest integer multiple of β (rounding up in case of a tie). Since every $x \in S'$ has entries from $\{-1,0,1\}$, we can deduce that for any $x \in S'$, we have

$$t - \beta - n\beta/2 < w \cdot x - n\beta/2 < w' \cdot x < w \cdot x + n\beta/2 \le t + \beta + n\beta/2.$$

Thus for every $x \in S'$, the value $w' \cdot x$ lies in a semi-open interval of length $\beta(n+2)$; moreover, since it only takes values which are integer multiples of β, there are at most $n+2$ possible values that $w' \cdot x$ can take for $x \in S'$. Since $S \subset S'$ and $|S| \ge \alpha 2^n$, there must be at least one value $t' \in (t - n\beta/2 - \beta, t + n\beta/2 + \beta]$ such that at least $\alpha 2^n/(n+2)$ points in S lie on the hyperplane $\mathbf{H}_1$ defined as $\mathbf{H}_1 = \{x : w' \cdot x = t'\}$. We also let $A_1 = \text{span}\{x \in S' : w' \cdot x = t'\}$. It is clear that $A_1 \subset \mathbf{H}_1$. Also, since at least $\alpha 2^n/(n+2)$ points of $\{-1,1\}^n$ lie on A_1, by Fact 16 we get that $\dim(A_1) \ge n - \log(n+2) - \log(1/\alpha)$.

It is easy to see that $\|w' - w\| \le \sqrt{n}\beta/2$, which implies that $\|w'\| \ge 1 - \sqrt{n}\beta/2$. Note that for any $x \in S'$ we have $|w' \cdot x - t'| \le (n+2)\beta$. Recalling Fact 20, we get that for any $x \in S'$ we have $d(x,\mathbf{H}_1) \le (\beta(n+2))/(1 - \sqrt{n}\beta/2)$. Since $\sqrt{n}\beta \ll 1$, we get that $d(x,\mathbf{H}_1) \le 2n\beta$ for every $x \in S'$.

At this point our plan for the rest of the proof of Theorem 155 is as follows: First we will construct a hyperplane $\mathbf{H}_k$ (by an inductive construction) such that $\text{span}(\mathbf{H}_k \cap (\{-1,1\}^n \cup \{0,1\}^n)) = \mathbf{H}_k$, $A_1 \subseteq \mathbf{H}_k$, and all points in S' are very close to $\mathbf{H}_k$ (say within Euclidean distance γ). Then we will apply Theorem 154 to conclude that any point $\{-1,1\}^n \cup \{0,1\}^n$ which is not on $\mathbf{H}_k$ must have Euclidean distance at least some γ' from $\mathbf{H}_k$. If $\gamma' > \gamma$ then we can infer that every point in S' lies on $\mathbf{H}_k$, which proves the theorem. We now describe the construction that gives $\mathbf{H}_k$.

If $\dim(A_1) = n-1$, then we let $k = 1$ and stop the process, since as desired we have $\text{span}(\mathbf{H}_k \cap (\{-1,1\}^n \cup \{0,1\}^n)) = \mathbf{H}_k$, $A_1 = H_k$, and $d(x,\mathbf{H}_k) \le 2n\beta$ for every $x \in S'$. Otherwise, by an inductive hypothesis, we may assume that for some $j \ge 1$ we have an affine space A_j and a hyperplane $\mathbf{H}_j$ such that

- $A_1 \subseteq A_j \subsetneq \mathbf{H}_j$;
- $\dim(A_j) = \dim(A_1) + j - 1$, and

- for all $x \in S'$ we have $d(x, \mathbf{H}_j) \leq 2^j n\beta$.

Using this inductive hypothesis, we will construct an affine space A_{j+1} and a hyperplane $\mathbf{H}_{j+1}$ such that $A_1 \subset A_{j+1} \subseteq \mathbf{H}_{j+1}$, $\dim(A_{j+1}) = \dim(A_1) + j$, and for all $x \in S'$ we have

$$d(x, \mathbf{H}_{j+1}) \leq 2^{j+1} n\beta.$$

If $A_{j+1} = \mathbf{H}_{j+1}$, we stop the process, else we continue.

We now describe the inductive construction. Since $A_j \subsetneq \mathbf{H}_j$, there must exist an affine subspace A'_j such that $A_j \subseteq A'_j \subsetneq \mathbf{H}_j$ and $\dim(A'_j) = n-2$. Let x_j denote $\arg\max_{x \in S'} d(x, A'_j)$. (We assume that $\max_{x \in S'} d(x, A'_j) > 0$; if not, then choose x_j to be an arbitrary point in $\{-1,1\}^n$ not lying on A'_j. In this case, the properties of the inductive construction will trivially hold.) Define $\mathbf{H}_{j+1} = \operatorname{span}(A'_j \cup x_j)$. It is clear that $\mathbf{H}_{j+1}$ is a hyperplane. We claim that for $x \in S'$ we have

$$d(x, \mathbf{H}_{j+1}) \leq d(x, \mathbf{H}_j) + d(x_j, \mathbf{H}_j) \leq 2^j n\beta + 2^j n\beta = 2^{j+1} n\beta.$$

To see this, observe that without loss of generality we may assume that $\mathbf{H}_j$ passes through the origin and thus A'_j is a linear subspace. Thus we have that $\|x_{\perp A'_j}\| \leq \|(x_j)_{\perp A'_j}\|$ for all $x \in S'$, where for a point $z \in \mathbb{R}^n$ we write $z_{\perp A'_j}$ to denote the component of x orthogonal to A'_j. Let $r = \|x_{\perp A'_j}\|$ and $r_1 = \|x_{j,\perp A'_j}\|$, where $r_1 \geq r$. Let θ denote the angle that $x_{\perp A'_j}$ makes with $\mathbf{H}_j$ and let ϕ denote the angle that $x_{\perp A'_j}$ makes with $(x_j)_{\perp A'_j}$. Then it is easy to see that $d(x, \mathbf{H}_{j+1}) = |r \cdot \sin(\theta - \phi)|$, $d(x, \mathbf{H}_j) = |r \cdot \sin(\theta)|$ and $d(x_j, \mathbf{H}_j) = |r_1 \cdot \sin(\phi)|$. Thus, we only need to check that if $r_1 \geq r$, then $|r \cdot \sin(\theta - \phi)| \leq |r \cdot \sin(\theta)| + |r_1 \cdot \sin(\phi)|$ which is straightforward to check.

Let $A_{j+1} = \operatorname{span}(A_j \cup x_j)$ and note that $A_1 \subset A_{j+1} \subseteq \mathbf{H}_{j+1}$ and $\dim(A_{j+1}) = \dim(A_j) + 1$. As shown above, for all $x \in S'$ we have $d(x, \mathbf{H}_{j+1}) \leq 2^{j+1} n\beta$. This completes the inductive construction.

Since $\dim(A_1) \geq n - \log(n+2) - \log(1/\alpha)$, the process must terminate for some $k \leq \log(n+2) + \log(1/\alpha)$. When the process terminates, we have a hyperplane $\mathbf{H}_k$ satisfying the following properties:

- $\operatorname{span}(\mathbf{H}_k \cap (\{-1,1\}^n \cup \{0,1\}^n)) = \mathbf{H}_k$; and
- $|\mathbf{H}_k \cap S| \geq \alpha 2^n/(n+2)$; and
- for all $x \in S'$ we have $d(x, \mathbf{H}_k) \leq 2^k n\beta \leq (1/\alpha) n(n+2)\beta$.

We can now apply Theorem 154 to the hyperplane $\mathbf{H}_k$ to get that if $\mathbf{H}_k = \{x \mid v \cdot x - \nu = 0\}$ with $\|v\| = 1$, then all the entries of v are integral multiples of a quantity E^{-1} where

$$E \leq (2n)^{\lfloor \log((n+2)/\alpha) \rfloor + 3/2} \cdot (\lfloor \log((n+2)/\alpha) \rfloor)!.$$

Consequently $v \cdot x$ is an integral multiple of E^{-1} for every $x \in (\{-1,1\}^n \cup \{0,1\}^n)$. Since there are points of $\{-1,1\}^n$ on $\mathbf{H}_k$, it must be the case that ν is also an integral multiple of E. So if any $x \in (\{-1,1\}^n \cup \{0,1\}^n)$ is such that $d(x, \mathbf{H}_k) < E$, then $d(x, \mathbf{H}_k) = 0$ and hence x actually lies on $\mathbf{H}_k$. Now recall that for any $x \in S'$ we have $d(x, \mathbf{H}_k) \leq (n/\alpha)(n+2)\beta$. Our upper bound on β from the theorem statement ensures that $(n/\alpha)(n+2)\beta < E^{-1}$, and consequently every $x \in S'$ must lie on $\mathbf{H}_k$, proving the theorem. □

A.3 Proof of Claim 31

We will need the following technical tool for the proof:

Lemma 156. *Let $S \subseteq \{-1,1\}^n$ and $\mathcal{W} : S \to [0,2]$ such that $\mathcal{W}(S) = \delta' 2^n$. Also, let $v \in \mathbb{R}^n$ have $\|v\| = 1$. Then*

$$\sum_{x \in S} \mathcal{W}(x) \cdot |v \cdot x| = \delta'(\sqrt{2\ln(1/\delta')} + 4) \cdot 2^n.$$

Proof. For any $x \in S$, let $D(x) \overset{\text{def}}{=} \mathcal{W}(x)/\mathcal{W}(S)$. Clearly, D defines a probability distribution over S. By definition, $\mathbf{E}_{x \sim D}[|v \cdot x|] = (\sum_{x \in S} \mathcal{W}(x) \cdot |v \cdot x|)/\mathcal{W}(S)$. Since $\mathcal{W}(S) = \delta' \cdot 2^n$, to prove the lemma it suffices to show that $\mathbf{E}_{x \sim D}[|v \cdot x|] = \sqrt{2\ln(1/\delta')} + 4$. Recall that for any non-negative random variable Y, we have the identity $\mathbf{E}[Y] = \int_{t \geq 0} \mathbf{P}r[Y > t]\, dt$. Thus, we have

$$\mathbf{E}_{x \sim D}[|v \cdot x|] = \int_{t \geq 0} \mathop{\mathbf{P}r}_{x \sim D}[|v \cdot x| > t]\, dt.$$

To bound this quantity, we exploit the fact that the integrand is concentrated. Indeed, by the Hoeffding bound we have that

$$\mathop{\mathbf{P}r}_{x \sim \{-1,1\}^n}[|v \cdot x| > t] \leq 2e^{-t^2/2}.$$

This implies that the set $A = \{x \in \{-1,1\}^n : |v \cdot x| > t\}$ is of size at most $2e^{-t^2/2}2^n$. Since $\mathcal{W}(x) \leq 2$ for all $x \in S$, we have that $\sum_{x \in A \cap S} \mathcal{W}(x) \leq 4e^{-t^2/2}2^n$. This implies that $\mathbf{P}r_{x \sim D}[|v \cdot x| > t] \leq (4/\delta') \cdot e^{-t^2/2}$. The following chain of inequalities completes the proof:

$$\begin{aligned}
\mathbf{E}_{x \sim D}\left[|v \cdot x|\right] &= \int_{t=0}^{\sqrt{2\ln(1/\delta')}} \mathop{\mathbf{P}r}_{x \sim D}[|w \cdot x| > t]\, dt + \int_{t \geq \sqrt{2\ln(1/\delta')}} \mathop{\mathbf{P}r}_{x \sim D}[|v \cdot x| > t]\, dt \\
&\leq \sqrt{2\ln(1/\delta')} + \int_{t \geq \sqrt{2\ln(1/\delta')}} \mathop{\mathbf{P}r}_{x \sim D}[|v \cdot x| > t]\, dt \\
&\leq \sqrt{2\ln(1/\delta')} + \int_{t \geq \sqrt{2\ln(1/\delta')}} \frac{4e^{-t^2/2}}{\delta'}\, dt \\
&\leq \sqrt{2\ln(1/\delta')} + \int_{t \geq \sqrt{2\ln(1/\delta')}} \frac{4te^{-t^2/2}}{\delta'}\, dt = \sqrt{2\ln(1/\delta')} + 4.
\end{aligned}$$

□

Recall the statement of Claim 31:

Claim 31. *For $j \leq \log(8/\epsilon)$, suppose that $\mathcal{W}(S_j) \geq \gamma_j \cdot 2^n$ where $\gamma_j = \beta^{4\log(1/\epsilon) - 2(j+1)} \cdot \epsilon$. Then $d_{\text{Chow}}(f, g) \geq \delta$, where $\delta = \beta^{4\log(1/\epsilon)}$.*

Proof. We start by observing that

$$\left(\epsilon - 4\sum_{\ell=0}^{j-1}\gamma_\ell - \delta\right)2^{n-1} \le \mathcal{W}(V_+^j), \mathcal{W}(V_-^j) \le (\epsilon+\delta)2^{n-1}.$$

The upper bound is obvious because $V_+^j \subseteq V_+^0$ and $V_-^j \subseteq V_-^0$ and the range of $\mathcal{W}$ is non-negative. To see the lower bound, note that $\mathcal{W}(V^0 \setminus V^j) \le 2(\sum_{\ell=0}^{j-1}\gamma_\ell)2^n$. As $V_+^0 \setminus V_+^j$ and $V_-^0 \setminus V_-^j$ are both contained in $V^0 \setminus V^j$, we get the stated lower bound. We also note that

$$\begin{aligned} 2\left(\sum_{\ell=0}^{j-1}\gamma_\ell\right)2^n &= 2\left(\sum_{\ell=0}^{j-1}\beta^{4\log(1/\epsilon)-2\ell-2}\right)2^n \\ &\le 4\beta^{4\log(1/\epsilon)-2j}2^n. \end{aligned}$$

This implies that the sets V_+^j and V_-^j are $(\epsilon, 4\beta^{4\log(1/\epsilon)-2j}+\delta)$ balanced. In particular, using that $\delta \le 4\beta^{4\log(1/\epsilon)-2j}$, we can say that the sets V_+^j and V_-^j are $(\epsilon, 8\beta^{4\log(1/\epsilon)-2j})$-balanced. We also observe that for $j \le \log(8/\epsilon)$, we have that $8\beta^{4\log(1/\epsilon)-2j} \le \epsilon/8$. Let us define $\mu_+^j = \sum_{x\in V_+^j}\mathcal{W}(x)\cdot x$, $\mu_-^j = \sum_{x\in V_-^j}\mathcal{W}(x)\cdot x$, $\Delta_+^j = V_+^0 \setminus V_+^j$ and $\Delta_-^j = V_-^0 \setminus V_-^j$. An application of Proposition 29 yields that $|(\mu_+^j - \mu_-^j)\cdot\widehat{\ell_j}| \ge (\beta\gamma_j - 8\beta^{4\log(1/\epsilon)-2j}\sqrt{2\ln(16/\epsilon)})2^n$.

We now note that

$$(\mu_+ - \mu_-)\cdot\widehat{\ell_j} = (\mu_+^j - \mu_-^j)\cdot\widehat{\ell_j} + \left(\sum_{x\in\Delta_+^j}\mathcal{W}(x) - \sum_{x\in\Delta_-^j}\mathcal{W}(x)\right)\cdot\widehat{\ell_j}.$$

Defining $\mu_+'^j = \sum_{x\in\Delta_+^j}\mathcal{W}(x)\cdot x$ and $\mu_-'^j = \sum_{x\in\Delta_-^j}\mathcal{W}(x)\cdot x$, the triangle inequality implies that

$$\left|(\mu_+ - \mu_-)\cdot\widehat{\ell_j}\right| \ge \left|(\mu_+^j - \mu_-^j)\cdot\widehat{\ell_j}\right| - \left|\mu_+'^j\cdot\widehat{\ell_j}\right| - \left|\mu_-'^j\cdot\widehat{\ell_j}\right|.$$

Using Lemma 156 and that $\mathcal{W}(\Delta_+^j), \mathcal{W}(\Delta_-^j) \le \mathcal{W}(V^0\setminus V^j) \le 8\beta^{4\log(1/\epsilon)-2j}\cdot 2^n$, we get that

$$\begin{aligned} \left|\mu_+'^j\cdot\widehat{\ell_j}\right| &= \sum_{x\in\Delta_+^j}\mathcal{W}(x)\cdot x\cdot\widehat{\ell_j} \\ &= O\left(|\Delta_+^j|\cdot\sqrt{\log(2^n/|\Delta_+^j|)}\right) \\ &= O\left(\beta^{4\log(1/\epsilon)-2j}\cdot\log^{3/2}(1/\epsilon)\cdot 2^n\right) \end{aligned}$$

and similarly

$$\left|\mu_-'^j\cdot\widehat{\ell_j}\right| = O\left(\beta^{4\log(1/\epsilon)-2j}\cdot\log^{3/2}(1/\epsilon)\cdot 2^n\right).$$

This implies that

$$\left|(\mu_+ - \mu_-) \cdot \widehat{\ell_j}\right| \geq (\beta\gamma_j - 8\beta^{4\log(1/\epsilon)-2j}\sqrt{2\ln(8/\epsilon)})2^n$$
$$-O\left(\beta^{4\log(1/\epsilon)-2j} \cdot \log^{3/2}(1/\epsilon) \cdot 2^n\right).$$

Plugging in the value of γ_j, we see that for ϵ smaller than a sufficiently small constant, we have that

$$\left|(\mu_+ - \mu_-) \cdot \widehat{\ell_j}\right| \geq \beta\gamma_j 2^{n-1}.$$

An application of Proposition 27 finally gives us that

$$d_{\text{Chow}}(f,g) \geq 2^{-n}\|\mu_+ - \mu_-\| \geq 2^{-n}(\mu_+ - \mu_-) \cdot \widehat{\ell_j} = \beta\gamma_j/2 \geq \delta$$

which establishes Claim 31. □

Appendix B

Missing proofs from Chapter 3

B.1 LTF representations with "nice" weights

This chapter contains the missing proofs from Chapter 3. The reason for having these proofs in the appendix as opposed to the main chapter is that the proofs In this section, we prove Theorem 40. This theorem essentially says that given any η-reasonable LTF, there is an equivalent representation of the LTF which is also η-reasonable and is such that the weights of the linear form (when arranged in decreasing order of magnitude) decrease somewhat "smoothly." For convenience we recall the exact statement of the theorem:

Theorem 40. *Let $f : \{-1,1\}^n \to \{-1,1\}$ be an η-reasonable LTF and $k \in [2,n]$. There exists a representation of f as $f(x) = \mathrm{sign}(v_0 + \sum_{i=1}^n v_i x_i)$ such that (after reordering coordinates so that condition (i) below holds) we have: (i) $|v_i| \geq |v_{i+1}|$, $i \in [n-1]$; (ii) $|v_0| \leq (1-\eta)\sum_{i=1}^n |v_i|$; and (iii) for all $i \in [0,k-1]$ we have $|v_i| \leq (2/\eta)\cdot\sqrt{n}\cdot k^{\frac{k}{2}}\cdot\sigma_k$, where $\sigma_k \overset{def}{=} \sqrt{\sum_{j\geq k} v_j^2}$.*

Proof of Theorem 40. The proof proceeds along similar lines as the proof of Lemma 5.1 from [120] (itself an adaptation of the argument of Muroga et. al. from [116]) with some crucial modifications.

Since f is η-reasonable, there exists a representation as $f(x) = \mathrm{sign}(w_0 + \sum_{i=1}^n w_i x_i)$ (where we assume w.l.o.g. that $|w_i| \geq |w_{i+1}|$ for all $i \in [n-1]$) such that $|w_0| \leq (1-\eta)\sum_{i=1}^n |w_i|$. Of course, this representation may not satisfy condition (iii) of the theorem statement. We proceed to construct the desired alternate representation as follows: First, we set $v_i = w_i$ for all $i \geq k$. We then set up a feasible linear program $\mathcal{LP}$ with variables $u_0, \dots, u_{k-1}$ and argue that there exists a feasible solution to $\mathcal{LP}$ with the desired properties.

Let $h : \{\pm 1\}^{k-1} \to \mathbb{R}$ denote the affine form $h(x) = w_0 + \sum_{j=1}^{k-1} w_j x_j$. We consider the following linear system $\mathcal{S}$ of 2^{k-1} equations in k unknowns $u_0, \dots, u_{k-1}$: For each $x \in \{\pm 1\}^{k-1}$ we include the equation

$$u_0 + \sum_{i=1}^{k-1} u_i x_i = h(x).$$

It is clear that the system $\mathcal{S}$ is satisfiable, since $(u_0, \dots, u_{k-1}) = (w_0, \dots, w_{k-1})$ is a solution.

We now relax the above linear system into the linear program $\mathcal{LP}$ (over the same variables) as follows: Let $C \overset{\text{def}}{=} \sqrt{n}\sigma_k$. Our linear program has the following constraints:

- For each $x \in \{\pm 1\}^{k-1}$ we include the (in)equality:

$$u_0 + \sum_{i=1}^{k-1} u_i x_i \begin{cases} \geq C & \text{if } h(x) \geq C, \\ = h(x) & \text{if } |h(x)| < C, \\ \leq -C & \text{if } h(x) \leq -C. \end{cases} \tag{B.1}$$

- For each $i \in [0, k-1]$, we add the constraints $\text{sign}(u_i) = \text{sign}(w_i)$. Since the w_i's are known, these are linear constraints, i.e., constraints like $u_1 \leq 0$, $u_2 \geq 0$, etc.

- We also add the constraints of the form $|u_i| \geq |u_{i+1}|$ for $1 \leq i \leq k-2$ and also $|u_{k-1}| \geq |w_k|$. Note that these constraints are equivalent to the linear constraints: $u_i \cdot \text{sign}(w_i) \geq u_{i+1} \cdot \text{sign}(w_{i+1})$ and $\text{sign}(w_{k-1}) \cdot u_{k-1} \geq |w_k|$.

- We let $q = \lceil 1/\eta \rceil$ and $\eta' = 1/q$. Clearly, $\eta' \leq \eta$. We now add the constraint $|u_0| \leq (1-\eta') \cdot \left(\sum_{j=1}^{k-1} |u_j| + \sum_{j=k}^{n} |w_j| \right)$. Note that this is also a linear constraint over the variables $u_0, u_1, \dots, u_{k-1}$. Indeed, it can be equivalently written as:

$$\text{sign}(w_0) \cdot u_0 - (1-\eta') \sum_{j=1}^{k-1} \text{sign}(w_j) \cdot u_j \leq (1-\eta') \sum_{j=k}^{n} |w_j|.$$

  Note that the RHS is strictly bounded from above by C, since

$$\sum_{j=k}^{n} |w_j| \leq \sqrt{n-k+1} \cdot \sigma_k < \sqrt{n}\sigma_k,$$

  where the first inequality is Cauchy-Schwarz and the second uses the fact that $k \geq 2$.

We observe that the above linear program is feasible. Indeed, it is straightforward to verify that all the constraints are satisfied by the vector $(w_0, \dots, w_{k-1})$. In particular, the last constraint is satisfied because $|w_0| \leq (1-\eta) \cdot \left(\sum_{j=1}^{k-1} |w_j| + \sum_{j=k}^{n} |w_j| \right)$ and hence *a fortiori*, $|w_0| \leq (1-\eta') \cdot \left(\sum_{j=1}^{k-1} |w_j| + \sum_{j=k}^{n} |w_j| \right)$.

Claim 157. *Let $(v_0, \dots, v_{k-1})$ be* any *feasible solution to $\mathcal{LP}$ and consider the LTF*

$$f'(x) = \text{sign}(v_0 + \sum_{j=1}^{k-1} v_j x_j + \sum_{j=k}^{n} w_j x_j).$$

Then $f'(x) = f(x)$ for all $x \in \{-1, 1\}^n$.

Proof. Given $x \in \{-1,1\}^n$, we have

$$h(x) = h(x_1, \ldots, x_{k-1}) = w_0 + \sum_{j=1}^{k-1} w_j x_j;$$

Let us also define

$$h'(x) = h'(x_1, \ldots, x_{k-1}) = v_0 + \sum_{j=1}^{k-1} v_j x_j$$

$$t(x) = \sum_{j \geq k} w_j x_j$$

Then, we have $f(x) = \operatorname{sign}(h(x) + t(x))$ and $f'(x) = \operatorname{sign}(h'(x) + t(x))$. Now, if $x \in \{-1,1\}^n$ is an input such that $|h(x)| < C$, then we have $h'(x) = h(x)$ by construction, and hence $f(x) = f'(x)$. If $x \in \{-1,1\}^n$ is such that $|h(x)| \geq C$, then by construction we also have that $|h'(x)| \geq C$. Also, note that $h(x)$ and $h'(x)$ always have the same sign. Hence, in order for f and f' to disagree on x, it must be the case that $|t(x)| \geq C$. But this is not possible, since $|t(x)| \leq \sum_{j=k}^{n} |w_j| \leq \sqrt{n-1} \cdot \sigma_k < C$. This completes the proof of the claim. $\square$

We are almost done, except that we need to choose a solution $(v_0, \ldots, v_{k-1})$ to $\mathcal{LP}$ satisfying property (iii) in the statement of the theorem. The next claim ensures that this can always be achieved.

Claim 158. *There is a feasible solution $v = (v_0, \ldots, v_{k-1})$ to the $\mathcal{LP}$ which satisfies property (iii) in the statement of the theorem.*

Proof. We select a feasible solution $v = (v_0, \ldots, v_{k-1})$ to the $\mathcal{LP}$ that maximizes the number of *tight* inequalities (i.e., satisfied with equality). If more than one feasible solutions satisfy this property, we choose one arbitrarily. We require the following fact from [116] (a proof can be found in [64, 39]).

Fact 159. *There exists a linear system $A \cdot v = b$ that uniquely specifies the vector v. The rows of (A, b) correspond to rows of the constraint matrix of $\mathcal{LP}$ and the corresponding RHS respectively.*

At this point, we use Cramer's rule to complete the argument. In particular, note that $v_i = \det(A_i)/\det(A)$ where A_i is the matrix obtained by replacing the i-th column of A by b. In particular, we want to give an upper bound on the magnitude of v_i; we do this by showing a lower bound on $|\det(A)|$ and an upper bound on $|\det(A_i)|$.

We start by showing that $|\det(A)| \geq \eta'$. First, since A is invertible, $\det(A) \neq 0$. Now, note that all rows of A have entries in $\{-1, 0, 1\}$ except potentially one "special" row which has entries from the set $\{\pm 1, \pm(1-\eta')\}$. If the special row does not appear, it is clear that $|\det(A)| \geq 1$, since it is not zero and the entries of A are all integers. If, on the other hand, the special row appears, simply expanding $\det(A)$ along that row gives that $\det(A) = a \cdot (1 - \eta') + b$ where $a, b \in \mathbb{Z}$. As $\eta' = 1/q$ for some $q \in Z$ and $\det(A) \neq 0$, we deduce that $|\det(A)| \geq \eta'$, as desired.

We bound $|\det(A_i)|$ from above by recalling the following fact.

Fact 160. *(Hadamard's inequality) If $A \in \mathbb{R}^{n \times n}$ and $v_1, \dots, v_n \in \mathbb{R}^n$ are the columns of A, then $|\det(A)| \le \prod_{j=1}^n \|v_j\|_2$.*

Now, observe that for all i, the i-th column of A_i (i.e., vector b) has all its entries bounded by C, hence $\|v_i\|_2 \le C\sqrt{k}$. All other columns have entries bounded from above by 1 and thus for $j \neq i$, $\|v_j\|_2 \le \sqrt{k}$. Therefore, $\det(A_i) \le C \cdot k^{k/2}$. Thus, we conclude that $|v_i| \le (C \cdot k^{k/2})/\eta'$. Further, as $(1/\eta') = \lceil(1/\eta)\rceil \le (2/\eta)$, we get $|v_i| \le 2C \cdot k^{k/2}/\eta$, completing the proof of the claim. □

The proof of Theorem 40 is now complete. □

B.2 Estimating correlations and Shapley values

Our algorithms need to estimate expectations of the form $f^*(i) = \mathbf{E}_{x \sim \mu}[f(x)x_i]$ and to estimate Shapley values $\tilde{f}(i)$, where $f : \{-1,1\}^n \to [-1,1]$ is an explicitly given function (an LBF). This is quite straightforward using standard techniques (see e.g. [8]) but for completeness we briefly state and prove the estimation guarantees that we will need.

Estimating correlations with variables. We will use the following:

Proposition 161. *There is a procedure* Estimate-Correlation *with the following properties: The procedure is given oracle access to a function $f : \{-1,1\}^n \to [-1,1]$, a desired accuracy parameter γ, and a desired failure probability δ. The procedure makes $O(n \log(n/\delta)/\gamma^2)$ oracle calls to f and runs in time $O(n^2 \log(n/\delta)/\gamma^2)$ (counting each oracle call to f as taking one time step). With probability $1-\delta$ it outputs a list of numbers $a^*(0), a^*(1), \dots, a^*(n)$ such that $|a^*(j) - f^*(j)| \le \gamma/\sqrt{n+1}$ for all $j = 0, \dots, n$. (Recall that $f^*(j)$ equals $\mathbf{E}_{x \sim \mu}[f(x)x_j]$, where $x_0 \equiv 1$).*

Proof. The procedure works simply by empirically estimating all the values $f^*(j) = \mathbf{E}_{x \sim \mu}[f(x)x_j]$, $j = 0, \dots, n$, using a single sample of m independent draws from μ. Since the random variable $(f(x)x_j))_{x \sim \mu}$ is bounded by 1 in absolute value, a straightforward Chernoff bound gives that for $m = O(n \log(n/\delta)/\gamma^2)$, each estimate $a^*(j)$ of $f^*(j)$ is accurate to within an additive $\pm\gamma/\sqrt{n+1}$ with failure probability at most $\delta/(n+1)$. A union bound over $j = 0, \dots, n$ finishes the argument. □

Estimating Shapley values. This is equally straightforward:

Proposition 162. *There is a procedure* Estimate-Shapley *with the following properties: The procedure is given oracle access to a function $f : \{-1,1\}^n \to [-1,1]$, a desired accuracy parameter γ, and a desired failure probability δ. The procedure makes $O(n \log(n/\delta)/\gamma^2)$ oracle calls to f and runs in time $O(n^2 \log(n/\delta)/\gamma^2)$ (counting each oracle call to f as taking one time step). With probability $1-\delta$ it outputs a list of numbers $\tilde{a}(1), \dots, \tilde{a}(n)$ such that $d_{\text{Shapley}}(a, f) \le \gamma$.*

Proof. The procedure empirically estimates each $\tilde{f}(j)$, $j = 1, \ldots, n$, to additive accuracy $\gamma/\sqrt{n}$ using Equation (3.1). This is done by generating a uniform random $\pi \sim \mathbb{S}_n$ and then, for each $i = 1, \ldots, n$, constructing the two inputs $x^+(\pi, i)$ and $x(\pi, i)$ and calling the oracle for f twice to compute $f(x^+(\pi, i)) - f(x(\pi, i))$. Since $|f(x^+(\pi, i)) - f(x(\pi, i))| \leq 2$ always, a sample of $m = O(n \log(n/\delta)/\gamma^2)$ permutations suffices to estimate all the $\tilde{f}(i)$ values to additive accuracy $\pm\gamma/\sqrt{n}$ with total failure probability at most δ. If each estimate $\tilde{a}(i)$ is additively accurate to within $\pm\gamma/\sqrt{n}$, then $d_{\text{Shapley}}(a, f) \leq \gamma$ as desired. $\square$

Appendix C

Missing proofs from Chapter 4

This appendix collects some of the missing proofs from Chapter 4. The reason the

C.1 Proofs from Section 4.1

Proof of Proposition 77. At a high level, the algorithm $\mathcal{T}^D$ performs a tournament by running a "competition" `Choose-Hypothesis`^{D} for every pair of distinct distributions in the collection $\mathcal{D}_\epsilon$. It outputs a distribution $D^\star \in \mathcal{D}_\epsilon$ that was never a loser (i.e., won or achieved a draw in all its competitions). If no such distribution exists in $\mathcal{D}_\epsilon$ then the algorithm outputs "failure." We start by describing and analyzing the competition subroutine between a pair of distributions in the collection.

Lemma 163. *In the context of Proposition 77, there is an algorithm* `Choose-Hypothesis`$^D(D_i, D_j, \epsilon', \delta')$ *which is given access to*

(i) independent samples from D and D_k, for $k \in \{i, j\}$,

(ii) an evaluation oracle $\mathrm{EVAL}_{D_k}(\beta)$, for $k \in \{i, j\}$,

an accuracy parameter ϵ' and a confidence parameter δ', and has the following behavior: It uses $m' = O\left((1/\epsilon'^2)\log(1/\delta')\right)$ samples from each of D, D_i and D_j, it makes $O(m')$ calls to the oracles $\mathrm{EVAL}_{D_k}(\beta)$, $k \in \{i, j\}$, performs $O(m')$ arithmetic operations, and if some D_k, $k \in \{i, j\}$, has $d_{\mathrm{TV}}(D_k, D) \leq \epsilon'$ then with probability $1 - \delta'$ it outputs an index $k^\star \in \{i, j\}$ that satisfies $d_{\mathrm{TV}}(D, D_{k^\star}) \leq 6\epsilon'$.

Proof. To set up the competition between D_i and D_j, we consider the following subset of $\mathcal{W}$:

$$H_{ij} = H_{ij}(D_i, D_j) \overset{\text{def}}{=} \{w \in \mathcal{W} \mid D_i(w) \geq D_j(w)\}$$

and the corresponding probabilities $p_{i,j} \overset{\text{def}}{=} D_i(H_{ij})$ and $q_{i,j} \overset{\text{def}}{=} D_j(H_{ij})$. Clearly, it holds $p_{i,j} \geq q_{i,j}$ and by definition of the total variation distance we can write

$$d_{\mathrm{TV}}(D_i, D_j) = p_{i,j} - q_{i,j}.$$

For the purposes of our algorithm, we would ideally want oracle access to the set H_{ij}. Unfortunately though, this is not possible since the evaluation oracles are only approximate. Hence, we will need to define a more robust version of the set H_{ij} which will turn out to have similar properties. In particular, we consider the set

$$H_{ij}^{\beta} \stackrel{\text{def}}{=} \{w \in \mathcal{W} \mid \widetilde{D}_i^{\beta}(w) \geq \widetilde{D}_j^{\beta}(w)\}$$

and the corresponding probabilities $p_{i,j}^{\beta} \stackrel{\text{def}}{=} D_i(H_{ij}^{\beta})$ and $q_{i,j}^{\beta} \stackrel{\text{def}}{=} D_j(H_{ij}^{\beta})$. We claim that the difference $\Delta \stackrel{\text{def}}{=} p_{i,j}^{\beta} - q_{i,j}^{\beta}$ is an accurate approximation to $d_{\mathrm{TV}}(D_i, D_j)$. In particular, we show:

Claim 164. *We have*

$$\Delta \leq d_{\mathrm{TV}}(D_i, D_j) \leq \Delta + \epsilon/4. \tag{C.1}$$

Before we proceed with the proof, we stress that (C.1) crucially uses our assumption that the evaluation oracles provide a *multiplicative* approximation to the exact probabilities.

Proof. To show (C.1) we proceed as follows: Let $A = H_{ij} \cap H_{ij}^{\beta}$, $B = H_{ij} \cap \overline{H_{ij}^{\beta}}$ and $C = \overline{H_{ij}} \cap H_{ij}^{\beta}$. Then we can write

$$d_{\mathrm{TV}}(D_i, D_j) = (D_i - D_j)(A) + (D_i - D_j)(B)$$

and

$$\Delta = (D_i - D_j)(A) + (D_i - D_j)(C).$$

We will show that

$$0 \leq (D_i - D_j)(B) \leq \epsilon/8 \tag{C.2}$$

and similarly

$$-\epsilon/8 \leq (D_i - D_j)(C) \leq 0 \tag{C.3}$$

from which the claim follows. We proceed to prove (C.2), the proof of (C.3) being very similar. Let $w \in B$. Then $D_i(w) \geq D_j(w)$ (since $w \in H_{ij}$) which gives $(D_i - D_j)(B) \geq 0$, establishing the LHS of (C.2). We now establish the RHS. For $w \in B$ we also have that $\widetilde{D}_i^{\beta}(w) < \widetilde{D}_j^{\beta}(w)$ (since $w \in \overline{H_{ij}^{\beta}}$). Now by the definition of the evaluation oracles, it follows that $\widetilde{D}_i^{\beta}(w) \geq \frac{D_i(w)}{(1+\beta)}$ and $\widetilde{D}_j^{\beta}(w) \leq (1+\beta) D_j(w)$. Combining these inequalities yields

$$D_i(w) \leq (1+\beta)^2 D_j(w) \leq (1+\epsilon/8) D_j(w)$$

where the second inequality follows by our choice of β. Therefore,

$$(D_i - D_j)(B) = \sum_{w \in B} (D_i(w) - D_j(w)) \leq (\epsilon/8) \cdot D_j(B) \leq \epsilon/8$$

as desired. $\square$

Note that the probabilities $p^{\beta}_{i,j}$ and $q^{\beta}_{i,j}$ are not available to us explicitly. Hence, the algorithm `Choose-Hypothesis` requires a way to empirically estimate each of these probability values (up to a small additive accuracy). This task can be done efficiently because we have sample access to the distributions D_i, D_j and oracle access to the set H^{β}_{ij} thanks to the $\mathrm{EVAL}_{D_k}(\beta)$ oracles. The following claim provides the details:

Claim 165. *There exists a subroutine* `Estimate`$(D_i, H^{\beta}_{ij}, \gamma, \delta)$ *which is given access to*

(i) independent samples from D_i,

(ii) an evaluation oracle $\mathrm{EVAL}_{D_k}(\beta)$, for $k \in \{i, j\}$,

an accuracy parameter γ and a confidence parameter δ, and has the following behavior: It makes $m = O\left((1/\gamma^2)\log(1/\delta)\right)$ draws from D_i and $O(m)$ calls to the oracles $\mathrm{EVAL}_{D_k}(\beta)$, $k = i, j$, performs $O(m)$ arithmetic operations, and with probability $1-\delta$ outputs a number $\widetilde{p}^{\beta}_{i,j}$ such that $|\widetilde{p}^{\beta}_{i,j} - p^{\beta}_{i,j}| \le \gamma$.

Proof. The desired subroutine amounts to a straightforward random sampling procedure, which we include here for the sake of completeness. We will use the following elementary fact, a simple consequence of the additive Chernoff bound.

Fact 166. *Let X be a random variable taking values in the range $[-1, 1]$. Then $\mathbf{E}[X]$ can be estimated to within an additive $\pm\tau$, with confidence probability $1-\delta$, using $m = \Omega((1/\tau^2)\log(1/\delta))$ independent samples from X. In particular, the empirical average $\widehat{X}_m = (1/m)\sum_{i=1}^{m} X_i$, where the X_i's are independent samples of X, satisfies $\mathbf{P}r\left[|\widehat{X}_m - \mathbf{E}[X]| \le \tau\right] \ge 1-\delta$.*

We shall refer to this as "empirically estimating" the value of $\mathbf{E}[X]$.

Consider the indicator function $I_{H^{\beta}_{ij}}$ of the set H^{β}_{ij}, i.e., $I_{H^{\beta}_{ij}} : \mathcal{W} \to \{0, 1\}$ with $I_{H^{\beta}_{ij}}(x) = 1$ if and only if $x \in H^{\beta}_{ij}$. It is clear that $\mathbf{E}_{x\sim D_i}\left[I_{H^{\beta}_{ij}}(x)\right] = D_i(H^{\beta}_{ij}) = p^{\beta}_{i,j}$. The subroutine is described in the following pseudocode:

```
Subroutine Estimate($D_i, H^{\beta}_{ij}, \gamma, \delta$):
**Input:** Sample access to $D_i$ and oracle access to $\mathrm{EVAL}_{D_k}(\beta)$, $k = i, j$.
**Output:** A number $\widetilde{p}^{\beta}_{ij}$ such that with probability $1-\delta$ it holds $|\widetilde{p}^{\beta}_{ij} - D_i(H^{\beta}_{ij})| \le \gamma$.

    1. Draw $m = \Theta\left((1/\gamma^2)\log(1/\delta)\right)$ samples $\mathrm{s} = \{s_\ell\}_{\ell=1}^{m}$ from $D_i$.
    2. For each sample $s_\ell$, $\ell \in [m]$:
        (a) Use the oracles $\mathrm{EVAL}_{D_i}(\beta)$, $\mathrm{EVAL}_{D_j}(\beta)$, to approximately evaluate $D_i(s_\ell)$, $D_j(s_\ell)$.
        (b) If $\widetilde{D}^{\beta}_i(s_\ell) \ge \widetilde{D}^{\beta}_j(s_\ell)$ set $I_{H^{\beta}_{ij}}(s_\ell) = 1$, otherwise $I_{H^{\beta}_{ij}}(s_\ell) = 0$.
```

```
3. Set $\widetilde{p}^{\beta}_{ij} = \frac{1}{m}\sum_{\ell=1}^{m} I_{H^{\beta}_{ij}}(s_\ell)$.

4. Output $\widetilde{p}^{\beta}_{ij}$.
```

The computational efficiency of this simple random sampling procedure follows from the fact that we can efficiently decide membership in H^{β}_{ij}. To do this, for a given $x \in \mathcal{W}$, we make a query to each of the oracles $\mathrm{EVAL}_{D_i}(\beta)$, $\mathrm{EVAL}_{D_j}(\beta)$ to obtain the probabilities $\widetilde{D}^{\beta}_i(x)$, $\widetilde{D}^{\beta}_j(x)$. We have that $x \in H^{\beta}_{ij}$ (or equivalently $I_{H^{\beta}_{ij}}(x) = 1$) if and only if $\widetilde{D}^{\beta}_i(x) \geq \widetilde{D}^{\beta}_j(x)$. By Fact 166, applied for the random variable $I_{H^{\beta}_{ij}}(x)$, where $x \sim D_i$, after $m = \Omega((1/\gamma^2)\log(1/\delta))$ samples from D_i we obtain a $\pm\gamma$-additive estimate to $p^{\beta}_{i,j}$ with probability $1-\delta$. For each sample, we make one query to each of the oracles, hence the total number of oracle queries is $O(m)$ as desired. The only non-trivial arithmetic operations are the $O(m)$ comparisons done in Step 2(b), and Claim 165 is proved. □

Now we are ready to prove Lemma 163. `Choose-Hypothesis`$^{D}(D_i, D_j, \epsilon', \delta')$ performs a competition between D_i and D_j in the following way:

```
Algorithm Choose-Hypothesis$^{D}(D_i, D_j, \epsilon', \delta')$:
**Input:** Sample access to $D$ and $D_k$, $k = i, j$, oracle access to $\mathrm{EVAL}_{D_k}(\beta)$, $k = i, j$.

1. Set $\widetilde{p}^{\beta}_{i,j}$ =Estimate$(D_i, H^{\beta}_{ij}, \epsilon'/8, \delta'/4)$ and $\widetilde{q}^{\beta}_{i,j}$ =Estimate$(D_j, H^{\beta}_{ij}, \epsilon'/8, \delta'/4)$.

2. If $\widetilde{p}^{\beta}_{i,j} - \widetilde{q}^{\beta}_{i,j} \leq 9\epsilon'/2$, declare a draw and return either $i$ or $j$. Otherwise:

3. Draw $m' = \Theta\left((1/\epsilon'^2)\log(1/\delta')\right)$ samples $\mathrm{s}' = \{s_\ell\}_{\ell=1}^{m'}$ from $D$.

4. For each sample $s_\ell$, $\ell \in [m']$:

    (a) Use the oracles $\mathrm{EVAL}_{D_i}(\beta)$, $\mathrm{EVAL}_{D_j}(\beta)$ to evaluate $\widetilde{D}^{\beta}_i(s_\ell)$, $\widetilde{D}^{\beta}_j(s_\ell)$.
    (b) If $\widetilde{D}^{\beta}_i(s_\ell) \geq \widetilde{D}^{\beta}_j(s_\ell)$ set $I_{H^{\beta}_{ij}}(s_\ell) = 1$, otherwise $I_{H^{\beta}_{ij}}(s_\ell) = 0$.

5. Set $\tau = \frac{1}{m'}\sum_{\ell=1}^{m'} I_{H^{\beta}_{ij}}(s_\ell)$, i.e., $\tau$ is the fraction of samples that fall inside $H^{\beta}_{ij}$.

6. If $\tau > \widetilde{p}^{\beta}_{i,j} - \frac{13}{8}\epsilon'$, declare $D_i$ as winner and return $i$; otherwise,

7. if $\tau < \widetilde{q}^{\beta}_{i,j} + \frac{13}{8}\epsilon'$, declare $D_j$ as winner and return $j$; otherwise,

8. declare a draw and return either $i$ or $j$.
```

It is not hard to check that the outcome of the competition does not depend on the ordering of the pair of distributions provided in the input; that is, on inputs (D_i, D_j) and (D_j, D_i) the competition outputs the same result for a fixed set of samples $\{s_1, \ldots, s_{m'}\}$ drawn from D.

The upper bounds on sample complexity, query complexity and number of arithmetic operations can be straightforwardly verified. Hence, it remains to show correctness. By Claim 165 and a union bound, with probability at least $1 - \delta'/2$, we will have that $|\widetilde{p}^{\beta}_{i,j} - p^{\beta}_{i,j}| \le \epsilon'/8$ and $|\widetilde{q}^{\beta}_{i,j} - q^{\beta}_{i,j}| \le \epsilon'/8$. In the following, we condition on this good event. The correctness of `Choose-Hypothesis` is then an immediate consequence of the following claim.

Claim 167. *Suppose that $d_{\mathrm{TV}}(D, D_i) \le \epsilon'$. Then:*

(i) If $d_{\mathrm{TV}}(D, D_j) > 6\epsilon'$, then the probability that the competition between D_i and D_j does not declare D_i as the winner is at most $e^{-m'\epsilon'^2/8}$. (Intuitively, if D_j is very far from D then it is very likely that D_i will be declared winner.)

(ii) The probability that the competition between D_i and D_j declares D_j as the winner is at most $e^{-m'\epsilon'^2/8}$. (Intuitively, since D_i is close to D, a draw with some other D_j is possible, but it is very unlikely that D_j will be declared winner.)

Proof. Let $r^{\beta} = D(H^{\beta}_{ij})$. The definition of the variation distance implies that $|r^{\beta} - p^{\beta}_{i,j}| \le d_{\mathrm{TV}}(D, D_i) \le \epsilon'$. Therefore, we have that $|r^{\beta} - \widetilde{p}^{\beta}_{i,j}| \le |r^{\beta} - p^{\beta}_{i,j}| + |\widetilde{p}^{\beta}_{i,j} - p^{\beta}_{i,j}| \le 9\epsilon'/8$. Consider the indicator (0/1) random variables $\{Z_\ell\}_{\ell=1}^{m'}$ defined as $Z_\ell = 1$ if and only if $s_\ell \in H^{\beta}_{ij}$. Clearly, $\tau = \frac{1}{m'}\sum_{\ell=1}^{m'} Z_\ell$ and $\mathbf{E}_{\mathbf{s}'}[\tau] = \mathbf{E}_{s_\ell \sim D}[Z_\ell] = r^{\beta}$. Since the Z_ℓ's are mutually independent, it follows from the Chernoff bound that $\mathbf{P}r[\tau \le r^{\beta} - \epsilon'/2] \le e^{-m'\epsilon'^2/8}$. Using $|r^{\beta} - \widetilde{p}^{\beta}_{i,j}| \le 9\epsilon'/8$. we get that $\mathbf{P}r[\tau \le \widetilde{p}^{\beta}_{i,j} - 13\epsilon'/8] \le e^{-m'\epsilon'^2/8}$.

- For part (i): If $d_{\mathrm{TV}}(D, D_j) > 6\epsilon'$, from the triangle inequality we get that $p_{i,j} - q_{i,j} = d_{\mathrm{TV}}(D_i, D_j) > 5\epsilon'$ Claim 164 implies that $p^{\beta}_{i,j} - q^{\beta}_{i,j} > 19\epsilon'/4$ and our conditioning finally gives $\widetilde{p}^{\beta}_{i,j} - \widetilde{q}^{\beta}_{i,j} > 9\epsilon'/2$. Hence, the algorithm will go beyond Step 2, and with probability at least $1 - e^{-m'\epsilon'^2/8}$, it will stop at Step 6, declaring D_i as the winner of the competition between D_i and D_j.

- For part (ii): If $\widetilde{p}^{\beta}_{i,j} - \widetilde{q}^{\beta}_{i,j} \le 9\epsilon'/2$ then the competition declares a draw, hence D_j is not the winner. Otherwise we have $\widetilde{p}^{\beta}_{i,j} - \widetilde{q}^{\beta}_{i,j} > 9\epsilon'/2$ and the argument of the previous paragraph implies that the competition between D_i and D_j will declare D_j as the winner with probability at most $e^{-m'\epsilon'^2/8}$.

This concludes the proof of Claim 167. □

This completes the proof of Lemma 163. □

We now proceed to describe the algorithm $\mathcal{T}^D$ and establish Proposition 77. The algorithm performs a tournament by running the competition `Choose-Hypothesis`$^D(D_i, D_j, \epsilon, \delta/(2N))$

for every pair of distinct distributions D_i, D_j in the collection $\mathcal{D}_\epsilon$. It outputs a distribution $D^\star \in \mathcal{D}_\epsilon$ that was never a loser (i.e., won or achieved a draw in all its competitions). If no such distribution exists in $\mathcal{D}_\epsilon$ then the algorithm outputs "failure." A detailed pseudocode follows:

```
Algorithm $\mathcal{T}^D(\{D_j\}_{j=1}^N, \epsilon, \delta)$:
Input: Sample access to $D$ and $D_k$, $k \in [N]$, and oracle access to $\mathrm{EVAL}_{D_k}$, $k \in [N]$.
    1. Draw $m = \Theta\left((1/\epsilon^2)(\log N + \log(1/\delta))\right)$ samples from $D$ and each $D_k$, $k \in [N]$.
    2. For all $i, j \in [N]$, $i \neq j$, run Choose-Hypothesis$^D(D_i, D_j, \epsilon, \delta/(2N))$ using this sample.
    3. Output an index $i^\star$ such that $D_{i^\star}$ was never declared a loser, if one exists.
    4. Otherwise, output "failure".
```

We now proceed to analyze the algorithm. The bounds on the sample complexity, running time and query complexity to the evaluation oracles follow from the corresponding bounds for `Choose-Hypothesis`. Hence, it suffices to show correctness. We do this below.

By definition, there exists some $D_i \in \mathcal{D}_\epsilon$ such that $d_{\mathrm{TV}}(D, D_i) \leq \epsilon$. By Claim 167, the distribution D_i never loses a competition against any other $D_j \in \mathcal{D}_\epsilon$ (so the algorithm does not output "failure"). A union bound over all N distributions in $\mathcal{D}_\epsilon$ shows that with probability $1-\delta/2$, the distribution D' never loses a competition.

We next argue that with probability at least $1-\delta/2$, every distribution $D_j \in \mathcal{D}_\epsilon$ that never loses has small variation distance from D. Fix a distribution D_j such that $d_{\mathrm{TV}}(D_j, D) > 6\epsilon$; Claim 167(i) implies that D_j loses to D_i with probability $1 - 2e^{-m\epsilon^2/8} \geq 1 - \delta/(2N)$. A union bound yields that with probability $1-\delta/2$, every distribution D_j that has $d_{\mathrm{TV}}(D_j, D) > 6\epsilon$ loses some competition.

Thus, with overall probability at least $1-\delta$, the tournament does not output "failure" and outputs some distribution $D^\star$ such that $d_{\mathrm{TV}}(D, D^\star)$ is at most 6ϵ. The proof of Proposition 77 is now complete. □

C.2 Proofs from Section 4.2

Proofs from Section 4.2

Proof of Lemma 84. To prove the lemma, we start by rewriting the expectation in (4.1) as follows:

$$\mathbf{E}_{x\sim D}\left[\chi(x, f(x))\right] = \mathbf{E}_{x\sim D_{f,+}}\left[\chi(x,1)\right] \cdot \Pr_{x\sim D}[f(x)=1] + \mathbf{E}_{x\sim D_{f,-}}\left[\chi(x,-1)\right] \cdot \Pr_{x\sim D}[f(x)=-1].$$

We also observe that

$$\mathbf{E}_{x\sim D}\left[\chi(x,-1)\right] = \mathbf{E}_{x\sim D_{f,+}}\left[\chi(x,-1)\right]\cdot \underset{x\sim D}{\mathbf{P}r}[f(x)=1] + \mathbf{E}_{x\sim D_{f,-}}\left[\chi(x,-1)\right]\cdot \underset{x\sim D}{\mathbf{P}r}[f(x)=-1].$$

Combining the above equalities we get

$$\mathbf{E}_{x\sim D}\left[\chi(x,f(x))\right] = \mathbf{E}_{x\sim D}\left[\chi(x,-1)\right] + \mathbf{E}_{x\sim D_{f,+}}\left[\chi(x,1)-\chi(x,-1)\right]\cdot \underset{x\sim D}{\mathbf{P}r}[f(x)=1]. \quad \text{(C.4)}$$

Given the above identity, the algorithm `Simulate-STAT`$_f^D$ is very simple: We use random sampling from D to empirically estimate the expectations $\mathbf{E}_{x\sim D}\left[\chi(x,-1)\right]$ (recall that D is assumed to be a samplable distribution), and we use the independent samples from $D_{f,+}$ to empirically estimate $\mathbf{E}_{x\sim D_{f,+}}\left[\chi(x,1)-\chi(x,-1)\right]$. Both estimates are obtained to within an additive accuracy of $\pm\tau/2$ (with confidence probability $1-\delta/2$ each). We combine these estimates with our estimate $\widetilde{b_f}$ for $\mathbf{P}r_{x\sim D}[f(x)=1]$ in the obvious way (see Step 2 of pseudocode below).

```
Subroutine Simulate-STAT$_f^D(D, D_{f,+}, \chi, \tau, \widetilde{b_f}, \delta)$:
Input: Independent samples from $D$ and $D_{f,+}$, query access to $\chi : \{-1,1\}^n \to \{-1,1\}$, accuracy $\tau$, $\widetilde{b_f} \in [0,1]$ and confidence $\delta$.
Output: If $|\widetilde{b_f} - \mathbf{P}r_{x\sim D}[f(x)=1]| \le \tau'$, a number $v$ that with probability $1-\delta$ satisfies $|\mathbf{E}_{x\sim D}[\chi(x,f(x))] - v| \le \tau+\tau'$.

    1. Empirically estimate the values $\mathbf{E}_{x\sim D}[\chi(x,-1)]$ and $\mathbf{E}_{x\sim D_{f,+}}[\chi(x,1)-\chi(x,-1)]$ to within an additive $\pm\tau/2$ with confidence probability $1-\delta/2$. Let $\widetilde{E_1}, \widetilde{E_2}$ be the corresponding estimates.
    2. Output $v = \widetilde{E_1} + \widetilde{E_2}\cdot\widetilde{b_f}$.
```

By Fact 166, we can estimate each expectation using $m = \Theta\left((1/\tau^2)\log(1/\delta)\right)$ samples (from D, $D_{f,+}$ respectively). For each such sample the estimation algorithm needs to evaluate the function χ (once for the first expectation and twice for the second). Hence, the total number of queries to χ is $O(m)$, i.e., the subroutine `Simulate-STAT`$_f^D$ runs in time $O(m\cdot t(n))$ as desired.

By a union bound, with probability $1-\delta$ both estimates will be $\pm\tau/2$ accurate. The bound (4.1) follows from this latter fact and (C.4) by a straightforward application of the triangle inequality. This completes the proof of Lemma 84. □

Proof of Proposition 85. The simulation procedure is very simple. We run the algorithm $\mathcal{A}_{\mathrm{SQ}}$ by simulating its queries using algorithm `Simulate-STAT`$_f^D$. The algorithm is described in the following pseudocode:

```
Algorithm $\mathcal{A}_{\mathrm{SQ-SIM}}(D, D_{f,+}, \epsilon, \widetilde{b_f}, \delta)$:
**Input:** Independent samples from $D$ and $D_{f,+}$, $\widetilde{b_f} \in [0,1]$, $\epsilon$, $\delta > 0$.
**Output:** If $|\widetilde{b_f} - \mathbf{P}r_{x\sim D}[f(x)=1]| \le \tau/2$, a hypothesis $h$ that with probability $1-\delta$ satisfies $\mathbf{P}r_{x\sim D}[h(x) \neq f(x)] \le \epsilon$.

    1. Let $\tau = \tau(n, 1/\epsilon)$ be the minimum accuracy ever used in a query to $\mathrm{STAT}(f,D)$ during
       the execution of $\mathcal{A}_{\mathrm{SQ}}(\epsilon, \delta/2)$.
    2. Run the algorithm $\mathcal{A}_{\mathrm{SQ}}(\epsilon, \delta/2)$, by simulating each query to $\mathrm{STAT}(f,D)$ as follows:
       whenever $\mathcal{A}_{\mathrm{SQ}}$ makes a query $(\chi, \tau)$ to $\mathrm{STAT}(f,D)$, the simulation algorithm runs
       Simulate-STAT$_f^D(D, D_{f,+}, \chi, \tau/2, \tau/2, \delta/(2T_1))$.
    3. Output the hypothesis $h$ obtained by the simulation.
```

Note that we run the algorithm $\mathcal{A}_{\mathrm{SQ}}$ with confidence probability $1-\delta/2$. Moreover, each query to the $\mathrm{STAT}(f,D)$ oracle is simulated with confidence $1-\delta/(2T_1)$. Since $\mathcal{A}_{\mathrm{SQ}}$ runs for at most T_1 time steps, it certainly performs at most T_1 queries in total. Hence, by a union bound over these events, with probability $1-\delta/2$ all answers to its queries will be accurate to within an additive $\pm\tau/2$. By the guarantee of algorithm $\mathcal{A}_{\mathrm{SQ}}$ and a union bound, with probability $1-\delta$, the algorithm $\mathcal{A}_{\mathrm{SQ-SIM}}$ will output a hypothesis $h : \{-1,1\}^n \to \{-1,1\}$ such that $\mathbf{P}r_{x\sim D}[h(x) \neq f(x)] \le \epsilon$. The sample complexity and running time follow from the bounds for `Simulate-STAT`$_f^D$. This completes the proof of Proposition 85. ☐

Proof of Fact 86. By definition we have that

$$\begin{aligned}
|\mathbf{E}_{x\sim D}[\phi(x)] - \mathbf{E}_{x\sim D'}[\phi(x)]| &= \left|\sum_{x\in\{-1,1\}^n} (D(x)-D'(x))\,\phi(x)\right| \\
&\le \sum_{x\in\{-1,1\}^n} |(D(x)-D'(x))|\,|\phi(x)| \\
&\le \max_{x\in\{-1,1\}^n} |\phi(x)| \cdot \sum_{x\in\{-1,1\}^n} |D(x)-D'(x)| \\
&\le 1\cdot \|D-D'\|_1 \\
&= 2d_{\mathrm{TV}}(D,D') \\
&\le 2\tau'
\end{aligned}$$

as desired. ☐

Proof of Claim 88. To simulate a sample from $D_{f,+}$ we simply draw samples from $\mathcal{U}_{f^{-1}(1)}$ until we obtain a sample x with $g(x)=1$. The following pseudocode makes this precise:

Subroutine `Simulate-sample`$^{D_{f,+}}(\mathcal{U}_{f^{-1}(1)}, g, \epsilon', \delta)$:
Input: Independent samples from $\mathcal{U}_{f^{-1}(1)}$, a circuit computing g, a value $\epsilon' > 0$ such that $\epsilon' \leq \mathbf{P}r_{x \sim \mathcal{U}_{f^{-1}(1)}}[g(x) = 1]$ and confidence parameter δ.
Output: A point $x \in \{-1,1\}^n$ that with probability $1-\delta$ satisfies $x \sim D_{f,+}$.

1. Repeat the following at most $m = \Theta\left((1/\epsilon')\log(1/\delta)\right)$ times:
   a) Draw a sample $x \sim \mathcal{U}_{f^{-1}(1)}$.
   b) If the circuit for g evaluates to 1 on input x then output x.
2. If no point x with $g(x) = 1$ has been obtained, halt and output "failure."

Since $\mathbf{P}r_{x \sim \mathcal{U}_{f^{-1}(1)}}[g(x) = 1] \geq \epsilon'$, after repeating this process $m = \Omega\left((1/\epsilon')\log(1/\delta)\right)$ times, we will obtain a satisfying assignment to g with probability at least $1-\delta$. It is clear that such a sample x is distributed according to $D_{f,+}$. For each sample we need to evaluate g once, hence the running time follows. □

Proof of Claim 89. The procedure `Estimate-Bias` is very simple. It runs $\mathcal{A}^{\mathcal{C}'}_{\text{count}}$ on inputs $\epsilon^\star = \tau'/2, \delta'$, using the representation for $g \in \mathcal{C}'$. Let p_g be the value returned by the approximate counter; `Estimate-Bias` returns $\widehat{p}/p_g$.

The claimed running time bound is obvious. To see that the procedure is correct, first observe that by Definition 73, with probability $1-\delta'$ we have that

$$\frac{|g^{-1}(1)|}{2^n} \cdot \frac{1}{1+\epsilon^\star} \leq p_g \leq \frac{|g^{-1}(1)|}{2^n} \cdot (1+\epsilon^\star).$$

For the rest of the argument we assume that the above inequality indeed holds. Let A denote $|g^{-1}(1)|$, let B denote $|f^{-1}(1) \cap g^{-1}(1)|$, and let C denote $|f^{-1}(1) \setminus g^{-1}(1)|$, so the true value $\mathbf{P}r_{x\sim D}[f(x) = 1]$ equals $\frac{B}{A}$ and the above inequality can be rephrased as

$$\frac{A}{1+\epsilon^\star} \leq p_g \cdot 2^n \leq A \cdot (1+\epsilon^\star).$$

By our assumption on $\widehat{p}$ we have that

$$B + C \leq \widehat{p} \cdot 2^n \leq (1+\epsilon')(B+C);$$

since $\mathbf{P}r_{x \sim \mathcal{U}_{f^{-1}(1)}}[g(x) = 1] \geq 1-\epsilon'$ we have

$$\frac{C}{B+C} \leq \epsilon' \qquad \text{(i.e., } C \leq \frac{\epsilon'}{1-\epsilon'} \cdot B \text{);}$$

and since $\mathbf{P}r_{x\sim\mathcal{U}_{g^{-1}(1)}}[f(x)=1]\geq\gamma'$ we have

$$\frac{B}{A}\geq\gamma'.$$

Combining these inequalities we get

$$\frac{1}{1+\epsilon^\star}\cdot\frac{B}{A}\leq\frac{1}{1+\epsilon^\star}\cdot\frac{B+C}{A}\leq\frac{\widehat{p}}{p_g}\leq\frac{B}{A}\cdot(1+\epsilon')(1+\epsilon^\star)\left(1+\frac{\epsilon'}{1-\epsilon'}\right)=\frac{B}{A}\cdot(1+\epsilon^\star)$$

Hence

$$\left|\frac{B}{A}-\frac{\widehat{p}}{p_g}\right|\leq\frac{B}{A}\left(1+\epsilon^\star-\frac{1}{1+\epsilon^\star}\right)\leq\frac{2\epsilon^\star}{1+\epsilon^\star}\leq 2\epsilon^\star,$$

where we have used $B\leq A$. Recalling that $\epsilon^\star=\tau'/2$, the lemma is proved. □

Details from Section 4.2

In this section, we complete the proof of Theorem 81. Recall that as described in Proposition 77, the hypothesis testing algorithm requires the following:

1. independent samples from the target distribution $\mathcal{U}_{f^{-1}(1)}$ (this is not a problem since such samples are available in our framework);
2. independent samples from $\hat{D}_i$ for each i (also not a problem since the i-th run of algorithm $\mathcal{A}'^{\mathcal{C}}_{\text{inv}}$ outputs a sampler for distribution $\hat{D}_i$; and
3. a $(1+O(\epsilon))$-approximate evaluation oracle $\mathrm{EVAL}_{\hat{D}_i}$ for each distribution $\hat{D}_i$.

In this subsection we show how to construct item (3) above, the approximate evaluation oracle. In more detail, we first describe a randomized procedure `Check` which is applied to the output of each execution of $\mathcal{A}'^{\mathcal{C}}_{\text{inv}}$ (across all k different settings of the input parameter $\widehat{p}_i$). We show that with high probability the "right" value $\widehat{p}_{i^*}$ (the one which satisfies $p\leq\widehat{p}_{i^*}<(1+\epsilon)p$) will pass the procedure `Check`. Then we show that for each value $\widehat{p}_{i^*}$ that passed the check a simple deterministic algorithm gives the desired approximate evaluation oracle for $\hat{D}_i$.

We proceed to describe the `Check` procedure and characterize its performance.

Algorithm `Check`(g,h,δ',ϵ) :
Input: functions g and h as described in Lemma 168, a confidence parameter δ', and an accuracy parameter ϵ
Output: If $|h^{-1}(1)\cap g^{-1}(1)|/|g^{-1}(1)|\geq\gamma/2$, with probability $1-\delta'$ outputs a pair (α,κ) such that $|\alpha-|h^{-1}(1)\cap g^{-1}(1)|/|g^{-1}(1)||\leq\mu\cdot|h^{-1}(1)\cap g^{-1}(1)|/|g^{-1}(1)|$ and $\frac{|g^{-1}(1)|}{1+\tau}\leq\kappa\leq(1+\tau)|g^{-1}(1)|$, where $\mu=\tau=\epsilon/40000$.

1. Sample $m = O(\log(2/\delta')/(\gamma\mu^2))$ points $x^1, \dots, x^m$ from $\mathcal{A}_{\text{gen}}^{\mathcal{C}'}(g, \gamma/4, \delta'/(2m))$. If any $x^j = \perp$ halt and output "failure."
2. Let α be $(1/m)$ times the number of points x^j that have $h(x) = 1$.
3. Call $\mathcal{A}_{\text{count}}^{\mathcal{C}'}(\tau, \delta'/2)$ on g and set κ to 2^n times the value it returns.

Lemma 168. *Fix $i \in [k]$. Consider a sequence of k runs of $\mathcal{A}_{\text{inv}}'^{\mathcal{C}}$ where in the i-th run it is given $\widehat{p_i} \overset{def}{=} (1+\epsilon)^{i-1}/2^n$ as its input parameter. Let g_i be the function in $\mathcal{C}'$ constructed by $\mathcal{A}_{\text{inv}}'^{\mathcal{C}}$ in Step 1 of its i-th run and h_i be the hypothesis function constructed by $\mathcal{A}_{\text{inv}}'^{\mathcal{C}}$ in Step 2(e) of its i-th run. Suppose* `Check` *is given as input g_i, h_i, a confidence parameter δ', and an accuracy parameter ϵ'. Then it either outputs "no" or a pair $(\alpha_i, \kappa_i) \in [0,1] \times [0, 2^{n+1}]$, and satisfies the following performance guarantee: If $|h_i^{-1}(1) \cap g_i^{-1}(1)|/|g_i^{-1}(1)| \geq \gamma/2$ then with probability at least $1 - \delta'$* `Check` *outputs a pair (α_i, κ_i) such that*

$$\left| \alpha_i - \frac{|h_i^{-1}(1) \cap g_i^{-1}(1)|}{|g_i^{-1}(1)|} \right| \leq \mu \cdot \frac{|h_i^{-1}(1) \cap g_i^{-1}(1)|}{|g_i^{-1}(1)|} \tag{C.5}$$

and

$$\frac{|g_i^{-1}(1)|}{1+\tau} \leq \kappa_i \leq (1+\tau)|g_i^{-1}(1)|, \tag{C.6}$$

where $\mu = \tau = \epsilon/40000$.

Proof. Suppose that i is such that $|h_i^{-1}(1) \cap g_i^{-1}(1)|/|g_i^{-1}(1)| \geq \gamma/2$. Recall from Definition 74 that each point x^j drawn from $\mathcal{A}_{\text{gen}}^{\mathcal{C}'}(g_i, \gamma/4, \delta'/(2m))$ in Step 1 is with probability $1 - \delta'/(2m)$ distributed according to $D_{g_i,\gamma/4}$; by a union bound we have that with probability at least $1 - \delta'/2$ all m points are distributed this way (and thus none of them are $\perp$). We condition on this going forward. Definition 74 implies that $d_{\text{TV}}(D_{g_i,\gamma/4}, \mathcal{U}_{g_i^{-1}(1)}) \leq \gamma/4$; together with the assumption that $|h_i^{-1}(1) \cap g_i^{-1}(1)|/|g_i^{-1}(1)| \geq \gamma/2$, this implies that each x^j independently has proability at least $\gamma/4$ of having $h(x) = 1$. Consequently, by the choice of m in Step 1, a standard multiplicative Chernoff bound implies that

$$\left| \alpha_i - \frac{|h^{-1}(1) \cap g^{-1}(1)|}{|g^{-1}(1)|} \right| \leq \mu \cdot \frac{|h^{-1}(1) \cap g^{-1}(1)|}{|g^{-1}(1)|}$$

with failure probability at most $\delta'/4$, giving (C.5).

Finally, Definition 73 gives that (C.6) holds with failure probability at most $\delta'/4$. This concludes the proof. □

Next we show how a high-accuracy estimate α_i of $|h_i^{-1}(1) \cap g_i^{-1}(1)|/|g_i^{-1}(1)|$ yields a deterministic approximate evaluation oracle for $\hat{D}_i'$.

Lemma 169. *Algorithm* `Simulate-Approx-Eval` *(which is deterministic) takes as input a value* $\alpha \in [0,1]$*, a string* $x \in \{-1,1\}^n$*, a parameter* κ*, (a circuit for)* $h : \{-1,1\}^n \to \{-1,1\}$*, and (a representation for)* $g : \{-1,1\}^n \to \{-1,1\}$*,* $g \in \mathcal{C}'$*, where* h, g *are obtained from a run of* $\mathcal{A}'^{\mathcal{C}}_{\mathrm{inv}}$*. Suppose that*

$$\left|\alpha - \frac{|h^{-1}(1) \cap g^{-1}(1)|}{|g^{-1}(1)|}\right| \le \mu \cdot \frac{|h^{-1}(1) \cap g^{-1}(1)|}{|g^{-1}(1)|}$$

and

$$\frac{|g^{-1}(1)|}{1+\tau} \le \kappa \le (1+\tau)|g^{-1}(1)|$$

where $\mu = \tau = \epsilon/40000$*. Then* `Simulate-Approx-Eval` *outputs a value* ρ *such that*

$$\frac{\hat{D}'(x)}{1+\beta} \le \rho \le (1+\beta)\hat{D}'(x), \tag{C.7}$$

where $\beta = \epsilon/192$*,* $\hat{D}$ *is the output distribution constructed in Step 3 of the run of* $\mathcal{A}^{\mathcal{C}}_{\mathrm{inv}}$ *that produced* h, g*, and* $\hat{D}'$ *is* $\hat{D}$ *conditioned on* $\{-1,1\}^n$ *(excluding* $\perp$*).*

Proof. The `Simulate-Approx-Eval` procedure is very simple. Given an input $x \in \{-1,1\}^n$ it evaluates both g and h on x, and if either evaluates to -1 it returns the value 0. If both evaluate to 1 then it returns the value $1/(\kappa\alpha)$.

For the correctness proof, note first that it is easy to see from the definition of the sampler C_f (Step 3 of $\mathcal{A}'^{\mathcal{C}}_{\mathrm{inv}}$) and Definition 74 (recall that the approximate uniform generator $\mathcal{A}^{\mathcal{C}'}_{\mathrm{gen}}(g)$ only outputs strings that satisfy g) that if $x \in \{-1,1\}^n$, $x \notin h^{-1}(1) \cap g^{-1}(1)$ then $\hat{D}$ has zero probability of outputting x, so `Simulate-Approx-Eval` behaves appropriately in this case.

Now suppose that $h(x) = g(x) = 1$. We first show that the value $1/(\kappa\alpha)$ is multiplicatively close to $1/|h^{-1}(1) \cap g^{-1}(1)|$. Let us write A to denote $|g^{-1}(1)|$ and B to denote $|h^{-1}(1) \cap g^{-1}(1)|$. With this notation we have

$$\left|\alpha - \frac{B}{A}\right| \le \mu \cdot \frac{B}{A} \qquad \text{and} \qquad \frac{A}{1+\tau} \le \kappa \le (1+\tau)A.$$

Consequently, we have

$$B(1-\mu-\tau) \le B \cdot \frac{1-\mu}{1+\tau} = \frac{B}{A}(1-\mu) \cdot \frac{A}{1+\tau} \le \kappa\alpha \le \frac{B}{A}(1+\mu) \cdot (1+\tau)A \le B(1+2\mu+2\tau),$$

and hence

$$\frac{1}{B} \cdot \frac{1}{1+2\mu+2\tau} \le \frac{1}{\kappa\alpha} \le \frac{1}{B} \cdot \frac{1}{1-\mu-\tau}. \tag{C.8}$$

Now consider any $x \in h^{-1}(1) \cap g^{-1}(1)$. By Definition 74 we have that

$$\frac{1}{1+\epsilon_3} \cdot \frac{1}{|g^{-1}(1)|} \le D_{g,\epsilon_3}(x) \le (1+\epsilon_3) \cdot \frac{1}{|g^{-1}(1)|}.$$

Since a draw from $\hat{D}'$ is obtained by taking a draw from D_{g,ϵ_3} and conditioning on it lying in $h^{-1}(1)$, it follows that we have

$$\frac{1}{1+\epsilon_3} \cdot \frac{1}{B} \le \hat{D}'(x) \le (1+\epsilon_3) \cdot \frac{1}{B}.$$

Combining this with (C.8) and recalling that $\mu = \tau = \epsilon/40000$ and $\epsilon_3 = \epsilon\gamma/48000$, we get (C.7) as desired. $\square$

The final algorithm: Proof of Theorem 81.

Finally we are ready to give the inverse approximate uniform generation algorithm $\mathcal{A}^{\mathcal{C}}_{\text{inv}}$ for $\mathcal{C}$.

```
Algorithm $\mathcal{A}^{\mathcal{C}}_{\text{inv}}(\mathcal{U}_{f^{-1}(1)}, \epsilon, \delta)$
Input: Independent samples from $\mathcal{U}_{f^{-1}(1)}$, accuracy and confidence parameters $\epsilon, \delta$.
Output: With probability $1-\delta$ outputs an $\epsilon$-sampler $C_f$ for $\mathcal{U}_{f^{-1}(1)}$ .

  1. For $i = 1$ to $k = O(n/\epsilon)$:
     a) Set $\widehat{p}_i \overset{\text{def}}{=} (1+\epsilon)^{i-1}/2^n$.
     b) Run $\mathcal{A}'^{\mathcal{C}}_{\text{inv}}(\mathcal{U}_{f^{-1}(1)}, \epsilon/12, \delta/3, \widehat{p}_i)$. Let $g_i \in \mathcal{C}'$ be the function constructed in Step 1,
        $h_i$ be the hypothesis function constructed in Step 2(e), and $(C_f)_i$ be the sampler for
        distribution $\hat{D}_i$ constructed in Step 3.
     c) Run Check$(g_i, h_i, \delta/3, \epsilon)$. If it returns a pair $(\alpha_i, \kappa_i)$ then add $i$ to the set $S$ (initially
        empty).
  2. Run the hypothesis testing procedure $\mathcal{T}^{\mathcal{U}_{f^{-1}(1)}}$ over the set $\{\hat{D}'_i\}_{i\in S}$ of hypothesis distri-
     butions, using accuracy parameter $\epsilon/12$ and confidence parameter $\delta/3$. Here $\mathcal{T}^{\mathcal{U}_{f^{-1}(1)}}$ is
     given access to $\mathcal{U}_{f^{-1}(1)}$, uses the samplers $(C_f)_i$ to generate draws from distributions $\hat{D}'_i$
     (see Remark 78), and uses the procedure Simulate-Approx-Eval$(\alpha_i, \kappa_i, h_i, g_i)$ for
     the $(1+\epsilon/192)$-approximate evaluation oracle $\text{EVAL}_{\hat{D}'_i}$ for $\hat{D}'_i$. Let $i^\star \in S$ be the index
     of the distribution that it returns.
  3. Output the sampler $(C_f)_{i^\star}$.
```

Proof of Theorem 81: Let $p \equiv \mathbf{P}r_{x\in\mathcal{U}_n}[f(x) = 1]$ denote the true fraction of satisfying assignments for f in $\{-1,1\}^n$. Let i^* be the element of $[k]$ such that $p \le \widehat{p}_{i^*} < (1+\epsilon/6)p$. By Theorem 90, with probability at least $1-\delta/3$ we have that both

(i) $(C_f)_{i^*}$ is a sampler for a distribution $\hat{D}_{i^*}$ such that $d_{\mathrm{TV}}(\hat{D}_{i^*}, \mathcal{U}_{f^{-1}(1)}) \leq \epsilon/6$; and

(ii) $|h_{i^*}^{-1}(1) \cap g_{i^*}^{-1}(1)|/|g_{i^*}^{-1}(1)| \geq \gamma/2$.

We condition on these two events holding. By Lemma 168, with probability at least $1-\delta/3$ the procedure `Check` outputs a value α_{i^*} such that

$$\left|\alpha_{i^*} - \frac{|h_{i^*}^{-1}(1) \cap g_{i^*}^{-1}(1)|}{|g_{i^*}^{-1}(1)|}\right| \leq \mu \cdot \frac{|h_{i^*}^{-1}(1) \cap g_{i^*}^{-1}(1)|}{|g_{i^*}^{-1}(1)|}$$

for $\mu = \epsilon/40000$. We condition on this event holding. Now Lemma 169 implies that the algorithm `Simulate-Approx-Eval`$((C_f)_{i^*})$ meets the requirements of a $(1+\beta)$-approximate evaluation oracle for $\mathrm{EVAL}_{\hat{D}'_{i^*}}$ from Proposition 77, for $\beta = \frac{\epsilon}{192}$. Hence by Proposition 77 (or more precisely by Remark 78) with probability at least $1-\delta/3$ the index $i^\star$ that $\mathcal{T}^{\mathcal{U}_{f^{-1}(1)}}$ returns is such that $\hat{D}'_{i^\star}$ is an $\epsilon/2$-sampler for $\mathcal{U}_{f^{-1}(1)}$ as desired.

As in the proof of Theorem 90, the claimed running time bound is a straightforward consequence of the various running time bounds established for all the procedures called by $\mathcal{A}_{\mathrm{inv}}^{\mathcal{C}}$. This concludes the proof of our general positive result, Theorem 81. $\square$