

Learning Interpretable and Reliable Classifiers Using Local Algorithms

by

Jane Lange

S.B., Stanford University (2016)

S.M., Massachusetts Institute of Technology (2025)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2026

© 2026 Jane Lange. All rights reserved.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Jane Lange
Department of Electrical Engineering and Computer Science
August 11, 2026

Certified by: Ronitt Rubinfeld
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Leslie Kolodziejcki
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

THESIS COMMITTEE

THESIS SUPERVISOR

Ronitt Rubinfeld

*Professor of Electrical Engineering and Computer Science
Department of Electrical Engineering and Computer Science*

THESIS READERS

Mohsen Ghaffari

*Associate Professor of Electrical Engineering and Computer Science
Department of EECS*

Ankur Moitra

*Norbert Wiener Professor of Mathematics
Department of Mathematics*

Li-Yang Tan

*Assistant Professor of Computer Science
Department of Computer Science, Stanford University*

Learning Interpretable and Reliable Classifiers Using Local Algorithms

by

Jane Lange

Submitted to the Department of Electrical Engineering and Computer Science
on August 11, 2026 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

The safe and responsible deployment of machine learning in real-world applications requires learned models to satisfy critical safety, reliability, and interpretability constraints without compromising predictive accuracy or computational efficiency. This work addresses the challenge of enforcing three such critical properties: properness, adversarial robustness, and discrete Lipschitz continuity. Expressing these properties as global constraints to be enforced throughout training makes the problem computationally expensive.

To overcome this, we propose a framework based on “local correctors”—efficient query-time procedures that transform a base hypothesis satisfying a relaxed constraint into one that strictly satisfies the target property. This localized approach reduces the complex globally constrained optimization task during training to a relaxed version, which can then be solved efficiently using polynomial approximation methods and standard optimization techniques. We design and analyze several such local correction algorithms and build learners and testers around them, producing provably safe and interpretable learned models with efficiency comparable to that of traditional learning.

Thesis supervisor: Ronitt Rubinfeld

Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I am deeply grateful to my family for their love and support throughout my life, and for encouraging me to pursue my research interests. My sincere appreciation goes to my advisor, Ronitt Rubinfeld, for her support, advice, and many fruitful research discussions, particularly on local computation algorithms, which feature heavily in this work. I also wish to thank my committee members Mohsen Ghaffari and Ankur Moitra, and particularly Li-Yang Tan, who has been an important mentor and collaborator throughout my time at MIT.

The work included in this thesis was done in collaboration with Ronitt; her student, Arsen Vasilyan; Boston University professor Sofya Raskhodnikova; and her student, Ephraim Linder. Chapter 3 is co-authored with Arsen, Chapter 4 with Ronitt and Arsen, and Chapter 5 with Ephraim, Sofya, and Arsen. I am very grateful to all of them, who made this project a joy to work on, and without whom this work would not have been possible. I would also like to acknowledge our reviewers for FOCS 2022, FOCS 2023, SODA 2025, and NeurIPS 2025, as well as Adam Smith, for their helpful comments on earlier versions of this work.

This work was supported by NSF Awards CCF-2006664, DMS-2022448, and CCF-2310818, NSF Graduate Research Fellowships Program under Grant No. 2141064, Microsoft Research, and the Big George Fellowship.

Contents

1	Introduction	13
1.1	Property Relaxation and Local Correction	14
1.1.1	LCAs and Succinct Representations of Large Objects	15
1.2	Organization of the Thesis	16
1.3	Adversarially Robust Learning of Halfspaces	17
1.3.1	Main Result	18
1.3.2	Technical Overview and Intermediate Results	18
1.3.3	Verifiable Robustness	22
1.4	Proper and Agnostic Learning of Monotone Functions	23
1.4.1	Main Results	23
1.4.2	Technical Overview	24
1.5	Local Lipschitz Filters	26
1.5.1	Main Results	26
1.5.2	Applications	28
1.6	Related Work	31
2	Preliminaries	33
2.1	Basic Notation	34
2.1.1	Distances and Errors	34
2.1.2	Graphs and Posets	34
2.2	Learning Algorithms	35
2.2.1	Agnostic Learning	35
2.2.2	Convex Optimization	36
2.3	Local Computation Algorithms and Succinct Representations	37
3	Agnostically Approximating Halfspaces by Robust Classifiers	41
3.1	Chapter-Specific Preliminaries	42
3.1.1	Perturbation and Robustness Models	42
3.1.2	Distances and Errors	43
3.1.3	Randomized Implementation of $\hat{\phi}$	44
3.1.4	Uniform Convergence of Empirical Approximations	45
3.2	Results and Pseudocode of Our Main Algorithm	50

3.2.1	Proof of Theorem 10	52
3.3	Pseudocode and Analysis of LearnRealValued	53
3.3.1	Facts About Log-Concave Distributions	54
3.3.2	Approximating Halfspaces with Polynomials	55
3.3.3	Analyzing LearnRealValued	57
3.3.4	Proving Claim 5	59
3.4	Pseudocode and Analysis of ComputeClassifier	63
3.4.1	Local Correction of Adversarial Robustness	63
3.4.2	Finding Good Rounding Thresholds	64
3.4.3	Randomized Set Partitioning	68
3.4.4	Correctness and Complexity of Algorithm 5	76
3.5	Verifiable Robustness	79
4	Agnostically Learning Monotone Functions	83
4.1	Chapter-Specific Preliminaries	84
4.2	Poset-Sorting and Monotonicity Correction for Boolean Labels	84
4.3	The LCA for Poset Sorting	85
4.3.1	A Global View	86
4.3.2	Correctness of Algorithm 7	86
4.3.3	Local Implementation	89
4.3.4	Refined Local Implementation	91
4.4	Correcting Real-Valued Functions Over Posets in ℓ_1 Distance	94
4.5	Analysis of the Matching Algorithm	98
4.5.1	Details and Correctness of MATCHVIOLATIONS	99
4.6	Algorithm for Agnostic Learning	103
4.7	Analysis of the Agnostic Learning Algorithm	106
4.7.1	Finishing the Proof of the Main Theorem (Theorem 4)	111
4.8	Rounding of Real-Valued Functions to Boolean	113
4.9	Agnostic Learning Algorithms Handling Randomized Labels	114
4.9.1	Case 1: ε Is Very Small	115
4.9.2	Case 2: ε Is Not Too Small	115
5	Local Correction of Discrete Lipschitz Functions	117
5.1	Chapter-Specific Preliminaries	118
5.2	ℓ_1 -Respecting Local Lipschitz Filter	118
5.2.1	Analysis of the Global Filter	119
5.2.2	Analysis of the Local Filter	122
5.3	ℓ_0 -Respecting Local Lipschitz Filter	124
5.3.1	Analysis of the Global Filter	124
5.3.2	Analysis of the Local Filter	126
5.4	Lower Bounds	127
5.4.1	Testing Definitions and the Lower Bound for Local Lipschitz Filters	127

5.4.2	Distribution-Free Testing Lower Bound	128
5.5	Application to Differential Privacy	134
5.5.1	Preliminaries on Differentially Private Mechanisms	135
5.5.2	Mechanism for Bounded-Range Functions	135
5.5.3	Mechanism for Unbounded Range Functions	137
5.6	Application to Tolerant Testing	138
<i>References</i>		143

Chapter 1

Introduction

Modern learning algorithms are often asked to do more than predict accurately. In many applications, a hypothesis is also expected to satisfy a structural guarantee: a credit model should respect monotonic relationships between features and outcomes, a data-analysis model should not reveal sensitive information about any individual in the dataset, and an image classifier should, for most inputs, not change its decision under imperceptible perturbations of the input. Such properties are desired by users and regulators alike as aspects of safety and interpretability, and they are often formalized as global constraints on the function computed by the learned model. They constrain the behavior of the learned hypothesis not only at a small collection of test points, but across a large domain of possible inputs.

This thesis studies a unified algorithmic approach to learning predictors that comply with such constraints, in settings where the learner runs much faster than the time it would take to check the constraints at every point in the domain. Rather than explicitly constructing the entire compliant hypothesis, we build efficient *local correction algorithms* that make queries to a hypothesis that is “mostly” compliant and answer queries to a nearby hypothesis that is strictly compliant. The algorithmic toolbox for this approach is that of *local computation algorithms* (LCAs) [3,103] and *local reconstruction* or *local correction* [76,104]. An LCA computes a small portion of a large object while inspecting only a small portion of its input. A local corrector is a special type of LCA that, given query access to an object that is close to satisfying a property, provides query access to a nearby object that satisfies the property exactly.

The central theme of this thesis is that local correction can turn efficient algorithms for solving an easier, relaxed version of the problem into efficient algorithms for the original problem, without paying the cost of constructing the entire corrected object in a more explicit form. In the chapters that follow, this paradigm appears in two well-studied learning settings: adversarially robust learning of halfspaces and proper learning of monotone functions. In each case, the computational obstacle is a global structural constraint that would be, in general, computationally intractable to even verify. In each case, the solution is to combine an efficient *relaxed* optimization or approximation step with a local algorithm that enforces the desired structure at query time. Finally, we show that a similar approach can be used in a third, non-learning setting: local correction of the discrete Lipschitz property, which allows for black-box differentially private function evaluation and tolerant testing of Lipschitzness.

1.1 Property Relaxation and Local Correction

The properties we study in this work can each be stated as the intersection of many local constraints on nearby points in a domain. For example, a classifier over $\mathbb{R}^d$ is adversarially robust at radius r if most points in the data distribution are not within distance r of the classification boundary; the corresponding local constraint at a point x is that no other point within distance r of x has a different label. A classifier is monotone if increasing a coordinate never decreases the label value; the corresponding local constraint at x is that no successor to x has a smaller label. These definitions are local in form, but satisfying them globally imposes an enormous system of constraints. In the case of monotonicity, the number of constraints is

at least exponential in the dimension of the domain, and in the case of adversarial robustness, it is infinite. An efficient learner cannot hope to even read the value of the function at every point in the domain, let alone check all of the constraints.

An efficient learner can, however, produce hypotheses that are *close* to satisfying these constraints. In both of the learning problems that we consider, the target function classes (linear threshold functions, monotone functions) are well-approximated in ℓ_1 distance by low-degree polynomials, over the data distributions that we consider. Furthermore, the set of low-degree polynomials that are ε -close in ℓ_1 distance to a function satisfying the property is a convex set, so convex optimization can find an error-minimizing polynomial within that set.

This reduces the problem of learning a robust or monotone classifier to the problem of finding a nearby robust or monotone classifier, given a polynomial that is ε -close in ℓ_1 distance to a classifier that has the property. From a sample complexity perspective, then, there is no obstacle to learning a function that satisfies these properties, since with unbounded running time a learner could search for such a nearby classifier by brute force. Thus, the focus of this work is to make that process more efficient in running time. More specifically, we aim for the running time of this step not to exceed the running time of the convex optimization step, up to polynomial factors. To this end, we employ a strategy of local reconstruction, which takes advantage of the fact that the properties we consider can be expressed as a system of local constraints.

The framework of local property reconstruction, which is the framework used in this thesis, traces its roots as far back as random self-reducibility in cryptography [108], and was introduced in the non-unique-decoding setting by Ailon, Chazelle, Seshadhri, and Liu [1]. It is one of the fundamental tasks studied in the area of local computation [3, 103], property testing, and sublinear-time algorithms. It has been studied for properties of functions, including monotonicity [10, 16, 84, 85, 104] and the Lipschitz property [10, 76], as well as for properties of graphs [27].

Intuitively, a local reconstructor for some property of functions is a randomized algorithm that gets oracle access to a function f and provides query access to a related function g that is guaranteed to have the property. The goal in the design of local reconstructors is to minimize the running time and the number of *lookups*, i.e., oracle calls to the input function f . In our learning problems, the oracle to f is the “almost-compliant” hypothesis produced by the convex optimization step, and the output function g is the hypothesis we actually return; that is, the fully compliant hypothesis.

1.1.1 LCAs and Succinct Representations of Large Objects

Among the many distributed and local computation models, the one that turns out relevant to us is the *local computation algorithm* (LCA) model [3, 103], defined formally in Section 2.3. In brief, to be an LCA, an algorithm should be local in the sense that it should not need to read the entire assignment of labels to vertices in order to determine which label will end up at some particular vertex x . It should suffice to read only the labels of vertices that lie in a restricted neighborhood of x .

In this work we also employ the concept of a *succinct representation* (Definition 8) as a tool for analyzing the time complexity of correcting a hypothesis using LCAs and the description size of the final hypothesis. For example, in Chapter 4, the succinct representations we deal with will have size and evaluation time $2^{\tilde{O}(\sqrt{d}/\varepsilon)}$. To be fully specific, we consider succinct representations of two types of objects:

- A succinct representation of a function $f : \{\pm 1\}^d \rightarrow \mathbb{R}$ is an algorithm that, given $x \in \{\pm 1\}^d$, computes $f(x)$ in time $2^{\tilde{O}(\sqrt{d}/\varepsilon)}$.
- A succinct representation of a (possibly weighted) graph G with the vertex set $\{\pm 1\}^d$ is an algorithm that, given $v \in \{\pm 1\}^d$, outputs all its neighbors and the weights of corresponding edges in time $2^{\tilde{O}(\sqrt{d}/\varepsilon)}$.

A polynomial of degree $O(\sqrt{d})$ is an example of a succinct representation, but another type of representation that makes frequent appearances in this work is in fact an LCA. An LCA efficiently computes a function over a large domain. For example, an LCA for an independent set takes as input some vertex v , makes some lookups to the adjacency list of the graph, then outputs “yes” or “no” so that the set of vertices for which the LCA would output “yes” forms an independent set. Typically, its running time and query complexity are each sublinear in the domain size.

Though not a requirement of LCAs in general, we require that all LCAs used in this work have outputs consistent with one global object, regardless of the order of user queries, and without remembering any history from previous queries. This property allows us to use the LCA, in conjunction with any succinct representation of the object it makes oracle queries to, as a succinct representation of the object it computes. This yields an efficient “global correction by local correction” strategy, since an LCA for local correction makes oracle queries to a function that nearly has the desired property, and we have a succinct representation of such a function as the output of our convex optimization step. We formalize this relationship in Section 2.3.

In addition to [64], on which much of this thesis builds, there have been many exciting recent works on local computation algorithms (LCAs). Some examples include [103], [3], [90], [70], [101], [58], [63], [34], [57], [99], [65], [89], [7], [24] and [72].

1.2 Organization of the Thesis

This thesis is organized into a preliminaries section and three technical chapters, each of which presents a different application of local correction.

Chapter 2, the preliminaries, collects definitions and background used throughout the thesis, including more formal facts about local computation algorithms, standard results from learning theory, and common notation.

Chapter 3, outlined in Section 1.3, presents the adversarially robust agnostic learner for halfspaces. This work is joint with Arsen Vasilyan, and was published in [86].

Chapter 4, outlined in Section 1.4, develops the proper and agnostic learning algorithm for monotone functions and the local corrector for monotonicity. This chapter combines material from two publications; [84], which is joint with Ronitt Rubinfeld and Arsen Vasilyan, and [85], which is joint with Arsen Vasilyan.

Chapter 5, outlined in Section 1.5, gives the local Lipschitz filters for bounded-range functions, their lower bounds, and applications to differential privacy and tolerant testing. This work is joint with Ephraim Linder, Sofya Raskhodnikova, and Arsen Vasilyan, and was published in [83].

1.3 Adversarially Robust Learning of Halfspaces

The first technical chapter studies adversarial robustness for classifiers over $\mathbb{R}^d$. A classifier is robust at radius r if, with probability at least $1 - O(r)$ over x drawn from the input distribution, every point within distance r of x receives the same label. This probability is the complement of what we call the *boundary volume* of the classifier at radius r : the probability that a random point lies within distance r of the classification boundary. When combined with classification error, it gives an upper bound on the *robust risk*, which is the total probability that a random point is misclassified under adversarial perturbations.

Formally, for a distribution $\mathcal{D}$ over inputs and a Boolean classifier f , the boundary volume at radius r is

$$\text{Boundary}_{\mathcal{D},r}(f) = \Pr_{x \sim \mathcal{D}}[\exists z : \|z\|_2 \leq r \text{ and } f(x) \neq f(x+z)].$$

There is a large body of research on adversarial robustness in machine learning, the focus of which is to assess the robustness of classifiers commonly deployed in practice — see [17,69,107], as well as [82] for an overview of the topic. It is well known that deep networks trained on classic benchmark data sets, such as ImageNet, can be “tricked” into misclassifying a test input by making a perturbation so small that it cannot be detected by humans, and that training robust models is a challenge in practice.

The objective of the robust-learning chapter is to compete with linear threshold functions (halfspaces), one of the most basic and well-known concept classes. These functions have boundary volume $O(r)$ under the distributions we consider (subgaussian isotropic log-concave distributions). As a robustness benchmark, we consider the robustness of a *proper* learning algorithm — one that outputs a hypothesis that is a halfspace. A proper agnostic learner would output a hypothesis with the following guarantees:

- **Agnostic approximation:** $\Pr_{(x,y) \sim \mathcal{D}}[y \neq h(x)] \leq \text{opt} + \varepsilon$, where **opt** is the classification error of the best halfspace, and
- **Adversarial robustness:** $\text{Boundary}_{\mathcal{D},r}(h) \leq O(r)$.

However, the state of the art for agnostically learning halfspaces is an improper algorithm — an algorithm that does not output a halfspace, but instead outputs a polynomial threshold

function (PTF). Polynomial regression with randomized rounding learns halfspaces over $\mathcal{N}(0, I_d)$ with time and sample complexity $d^{\tilde{O}(1/\varepsilon^2)}$ and satisfies the accuracy guarantee [46, 77]. In fact, the efficiency of polynomial regression is due to the inherent robustness of halfspaces — their small surface area! However, there exist degree- $1/\varepsilon^2$ PTFs with boundary volume $\Omega(1)$ even at a small radius of perturbation,¹ and this algorithm makes no guarantee that its output is robust.

No proper agnostic learning algorithm with $2^{o(d)}$ running time for general subgaussian isotropic log-concave distributions is known, and for the Gaussian distribution the state of the art for proper learning is $d^{O(1/\varepsilon^4)}$ [48]. The complexity of proper learning in these settings is still an open question. We circumvent the possible difficulty of proper learning by giving an improper algorithm that still satisfies both the accuracy and robustness properties. Our algorithm has time and sample complexity $d^{\tilde{O}(1/\varepsilon^2)}$, matching the run-time of the best (improper and non-robust) agnostic learning algorithm [46]. Up to polylogarithmic factors in the exponent, this also matches the statistical query lower bound for agnostically learning halfspaces [49].

1.3.1 Main Result

Our main result is an improper agnostic learner (Algorithm 1) for halfspaces that outputs an adversarially robust hypothesis.

Theorem 1 (Adversarially robust learning of halfspaces, informal version of Theorem 10). *Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{-1, 1\}$ such that the $\mathbb{R}^d$ -marginal is subgaussian, isotropic, and log-concave. There is an algorithm that takes a robustness radius r , an accuracy parameter ε , and a sample of size $d^{\tilde{O}(1/\varepsilon^2)}$ from $\mathcal{D}$. With high probability, it outputs a hypothesis $h : \mathbb{R}^d \rightarrow \{-1, 1\}$ with the following guarantees:*

- **Agnostic approximation:** $\Pr_{(x,y) \sim \mathcal{D}}[h(x) \neq y] \leq \text{opt} + O(\varepsilon)$, where opt is the classification error of the best halfspace.
- **Adversarial robustness:** $\text{Boundary}_{\mathcal{D}, r}(h) \leq O(r)$.

1.3.2 Technical Overview and Intermediate Results

Our algorithm and its analysis have three main components, which may be of independent interest. Two of the components solve a relaxation of the robust learning problem, and the third transforms the almost-robust hypothesis into a robust one. The relaxed learner produces a hypothesis with small *noise sensitivity* (a notion of boundary volume with random perturbations instead of adversarial perturbations) and small *isolation probability* (the probability that the local noise sensitivity around a point is very high). To formally define these two quantities, we need the following notion:

¹Consider the PTF $\text{sign}(\prod_{i=1}^{1/\varepsilon^2} x_i)$, which has boundary volume $\Omega(1)$ over $\mathcal{N}(0, I_d)$ at any radius $\geq \varepsilon^2$.

Definition (Local noise sensitivity). *We define the local noise sensitivity $\phi_{f,\eta}(x)$ at noise scale η as*

$$\phi_{f,\eta}(x) := \mathbb{E}_{z \sim \mathcal{N}(0, I_d)} [\tfrac{1}{2} |f(x) - f(x + \eta z)|].$$

We remark that when f is Boolean-valued, this is equivalently

$$\phi_{f,\eta}(x) := \Pr_{z \sim \mathcal{N}(0, I_d)} [f(x) \neq f(x + \eta z)].$$

We now define the noise sensitivity and the isolation probability:

Definition (Distributional noise sensitivity). *The noise sensitivity of a function f at noise scale η on distribution $\mathcal{D}$ is defined as*

$$\text{NS}_{\mathcal{D},\eta}(f) = \mathbb{E}_{x \sim \mathcal{D}} [\phi_{f,\eta}(x)]. \quad (1.1)$$

We remark that when $\mathcal{D} = \mathcal{N}(0, I_d)$, our definition of distributional noise sensitivity differs from the standard definition of Gaussian noise sensitivity.

Definition (Isolation probability). *We call a point isolated if its local noise sensitivity is over a threshold. The isolation probability of a function f relative to a threshold t is*

$$\text{iso}_{\mathcal{D},\eta}(f, t) := \Pr_{x \sim \mathcal{D}} [\phi_{f,\eta}(x) > t].$$

Intuitively, a function can have low distributional noise sensitivity and still not be adversarially robust, if the classification boundary lies near many isolated points scattered throughout the domain. For this reason, after learning a hypothesis with low noise sensitivity, we apply the local correction algorithm which we describe below, which eliminates regions of high local noise sensitivity (isolated points).

Local correction of adversarial robustness (Algorithm 3)

We analyze a very simple LCA that makes queries to a function and outputs queries to a nearby function with reduced boundary volume. This algorithm estimates the local noise sensitivity at x , and changes the label of x if the local noise sensitivity is too high. This is, in essence, the smoothing procedure discussed in [39,87,91], shown to be quite successful experimentally. We give formal guarantees on the boundary volume and the error introduced by the smoothing procedure in terms of the relaxed robustness properties of the input function (the noise sensitivity and isolation probability).

Lemma 1.3.1 (Local corrector for adversarial robustness, informal version of Theorem 16). *Let $\varepsilon, \alpha, \beta, r \in (0, 1)$. Let $f : \mathbb{R}^d \rightarrow \{-1, 1\}$ be a degree- k polynomial threshold function with noise sensitivity $\leq \alpha$ and isolation probability $\leq \beta$. There is an efficient randomized algorithm **ROBUSTNESSLCA** that makes black-box queries to f and answers black-box queries to a function $g : \mathbb{R}^d \rightarrow \{-1, 1\}$ that satisfies the following with high probability:*

- **Adversarial robustness:** $\text{Boundary}_{\mathcal{D},r} \leq O(\alpha + \varepsilon)$.
- **Small distance:** $\Pr_{x \sim \mathcal{D}}[g(x) \neq f(x)] \leq O(\beta + \varepsilon)$.

We use this algorithm in a “deterministic” fashion, where all calls to the local corrector share the same random seed, which is good with high probability. When the random seed is good, the guarantees hold for all $x \in \mathbb{R}^d$.² This allows us to append the local corrector and its fixed seed to some degree- k PTF, creating a deterministic robust hypothesis that can be evaluated in $d^{O(k)}$ time.

From the guarantees of the local corrector and the desired guarantees of the overall algorithm, we determine that we must feed the local corrector a PTF of error $\text{opt} + O(\varepsilon)$, noise sensitivity $O(r)$, and isolation probability $O(\varepsilon)$.

Polynomial regression under noise sensitivity constraints (Algorithm 2)

Our first step in finding such a PTF is to learn a polynomial with low error, noise sensitivity, and isolation probability in the ℓ_1 -distance regime.

Theorem 2 (Learning a polynomial, informal version of Theorem 11). *There is an algorithm with time and sample complexity $d^{\tilde{O}(1/\varepsilon^2)}$ that takes samples from the distribution $\mathcal{D}$ and, with high probability, returns a degree- $\tilde{O}(1/\varepsilon^2)$ polynomial p with the following properties:*

- **Accuracy:** $\text{err}(p) \leq \text{opt} + O(\varepsilon)$,
- **Low noise sensitivity:** $\text{NS}(p) \leq O(r + \varepsilon)$, and
- **Low isolation probability:** $\text{iso}(p) \leq O(\varepsilon)$.

We achieve this with convex programming. The ℓ_1 noise sensitivity is a convex constraint, and we use a convex upper bound on the ℓ_1 analogue of the isolation probability. We minimize error over the set of degree- $\tilde{O}(1/\varepsilon^2)$ polynomials under these constraints. To show that a feasible polynomial of error $\text{opt} + O(\varepsilon)$ exists, we show that the halfspace-approximating polynomial given in [46] satisfies the constraints under any subgaussian isotropic log-concave distribution.

Randomized partitioning and rounding (Algorithm 5, Algorithm 4)

With p in hand, we must then round p to some polynomial threshold function $\text{sign}(p - t)$ that satisfies the error, noise sensitivity, and isolation probability constraints. Simply rounding at a uniformly random threshold $t \sim [-1, 1]$, as in [77], does result in error $\text{opt} + O(\varepsilon)$, noise sensitivity $O(r)$, and isolation probability $O(\varepsilon)$ in expectation, but does not guarantee that

²We remark that in a setting where each call to the corrector is allowed to fail *independently* with probability δ , the query complexity can be reduced from $d^{O(k)} \cdot \log(1/\delta)/\varepsilon^2$ to $O(\log(1/\delta)/\varepsilon^2)$ and the assumption that f is a degree- k PTF can be dropped. This is not applicable in our setting but might be desired in a non-learning application of local correction for robustness.

the conditions ever hold simultaneously for the same t . Thus, running the local corrector on just one rounded function would not guarantee a hypothesis with the right properties. In this step, we find a “deterministic mixture” of rounded functions that simultaneously satisfies the three conditions: a partition of the domain into parts, where each part is rounded at a different threshold. The corrector is run on each part separately.

A first attempt at finding such a partition would be to allow it to have $O(1/\varepsilon^2)$ parts. Observe that the error, noise sensitivity, and isolation probability concentrate with deviation ε when averaging over $O(1/\varepsilon^2)$ random thresholds. Thus, an equal-weighted mixture of $O(1/\varepsilon^2)$ independent random roundings would satisfy all of the guarantees simultaneously with high probability. But suppose we partitioned the domain into $1/\varepsilon^2$ sets of equal mass — we cannot guarantee robustness for any point near the boundaries of these sets, and the total volume of these boundaries scales inversely with ε , which is undesirable. To minimize the increase in boundary volume due to partitioning, it is necessary to find a partition of constant size.

We show that there exists a mixture of four rounded functions satisfying all the constraints simultaneously by applying Carathéodory’s theorem ([29]), and we find the mixture by linear programming.

To understand how to turn the mixture into a partition, consider the example of handling just the error when y is a deterministic function of x . The error of the mixture is

$$\sum_{i \in [4]} w_i \cdot \text{err}(\text{sign}(p - t_i)) \leq \text{opt} + O(\varepsilon),$$

where w_i is the mixing weight of the i^{th} PTF. For each i in the mixture, there is a set of volume $\text{err}(\text{sign}(p - t_i))$ consisting of the points misclassified by $\text{sign}(p - t_i)$. We want to partition the domain such that a w_i -fraction of this set falls into the i^{th} part, so that

$$\sum_{i \in [4]} \mathbb{1}[x \text{ is in the } i^{\text{th}} \text{ part and } y \neq \text{sign}(p(x) - t_i)] = \sum_{i \in [4]} w_i \cdot \text{err}(\text{sign}(p - t_i)).$$

We cite a theorem of [43], which says, essentially, that projecting on a random unit vector causes the set to be very close to normally-distributed with high probability. Thus, our partition is according to the inner product with a random unit vector u , and the parts are intervals $J_1, \dots, J_4$ of Gaussian mass $w_1, \dots, w_4$ respectively. With high probability, the sets of misclassified points and the sets for which robustness is guaranteed by the corrector are all partitioned with the correct weights.

Theorem 3 (Randomized partitioning, informal version of Theorem 12). *Let p be a polynomial satisfying the guarantees of Theorem 2. There is an algorithm that outputs a unit vector u , rounding thresholds $t_1, \dots, t_4$, and a partition of the real line into intervals $J_1, \dots, J_4$ such that the hypothesis*

$$h(x) = \sum_{i=1}^4 \text{ROBUSTNESSLCA}(\text{sign}(p(x) - t_i)) \cdot \mathbb{1}[\langle x, u \rangle \in J_i]$$

has the following properties with high probability:

- **Accuracy:** $\text{err}(h) \leq \text{opt} + O(\varepsilon)$,
- **Robustness:** $\text{Boundary}_{\mathcal{D},r}(h) \leq O(r + \varepsilon)$.

The time complexity is $d^{\tilde{O}(1/\varepsilon^2)}$.

1.3.3 Verifiable Robustness

The work of [67] discusses the planting of “backdoors” in learned classifiers — structured violations of the robustness condition — and the impossibility of efficiently distinguishing a robust model from a backdoored one in the most general case. The takeaway is that when training is performed by an untrusted service, it is not generally possible to verify that a hypothesis is robust, even if a description of the hypothesis is available. However, our learning algorithm produces a hypothesis with a specific structure that can be checked, and the reduced expressivity of this structure allows robustness to be verifiable. Under complexity assumptions, the robustness of our algorithm’s output can be verified in the following way:

Corollary 1 (Deterministic verifiability, informal version of Corollary 6). *If $P = BPP$, then there is a learning algorithm $\mathcal{B}$ that, given access to labeled samples from a subgaussian isotropic log-concave distribution, runs in time $d^{\tilde{O}(1/\varepsilon^2)}$ and produces a hypothesis h with the following guarantees³:*

- **Agnostic approximation:** h satisfies $\Pr_{(x,y) \sim \mathcal{D}}[h(x) \neq y] \leq \text{opt} + O(\varepsilon)$, where opt is the misclassification error of the best halfspace.
- **Verifiable robustness:** There is an efficient deterministic verifier that takes as input a hypothesis g and a point $x \in \mathbb{R}^d$, and always rejects if

$$\exists z : \|z\|_2 \leq r \text{ and } g(x) \neq g(x + z).$$

If g is the output of $\mathcal{B}$, then the verifier accepts with probability at least $1 - O(r + \varepsilon)$ over x drawn from $\mathcal{D}$.

As a result, training can be done entirely by an untrusted service that claims to be using our learning algorithm, but may not be. The verifier checks whether the hypothesis matches the format our learning algorithm produces, rejects if it does not match, and performs a robustness test that is sound if it does match. The user must trust that the error of the hypothesis really is $\text{opt} + O(\varepsilon)$, but does not have to place any trust in the service to guarantee robustness — there is a deterministic algorithm that the user can perform to verify that most points are not near the boundary. Furthermore, in a setting where the user performs some of the construction of the hypothesis (but need not access any training data), verifiable robustness can be made unconditional, but the soundness of the verifier can fail with small probability over the randomness of the hypothesis construction.

³Agnostic approximation holds with high probability, whereas verifiable robustness always holds (deterministically).

1.4 Proper and Agnostic Learning of Monotone Functions

The study of learning and testing algorithms for monotone functions [1,4,12,23,25,32,33,36–38,54,66,79,81,84,96,97] and various subclasses of monotone functions [5,22,75,111] is a major research direction. In Chapter 4, we consider two fundamental problems in this line of work: *approximating the distance* of unknown functions to monotone, and *agnostic proper learning* of monotone functions. For each of these problems we are given independent uniform samples $\{\mathbf{x}_i\}$ labeled by an arbitrary function $f : \{\pm 1\}^d \rightarrow \{\pm 1\}$, and we perform the following tasks:

1. **Estimating distance to monotonicity** is the task of estimating up to some additive error ε the distance $\text{dist}(f, f_{\text{mon}})$ from f to the monotone function f_{mon} that is closest to f .
2. **Agnostic proper learning of monotone functions** is the task of obtaining a description of a monotone function g_{mon} , whose distance $\text{dist}(f, g_{\text{mon}})$ approximates $\text{dist}(f, f_{\text{mon}})$ up to additive error ε .

For over 25 years there has been a large statistical-to-computational gap in our understanding of the agnostic learning problem. A $2^{\tilde{O}(\sqrt{d}/\varepsilon)}$ -time *improper* learning algorithm — that is, an algorithm that outputs a predictor f that is accurate but not guaranteed to be monotone — is given in [25]. One could use the output of this algorithm to obtain a monotone $\hat{g}$ by computing f on every element of $\{-1, 1\}^d$ and solving a linear program to obtain the closest monotone function to f . Although this gives a $2^{\tilde{O}(\sqrt{d}/\varepsilon)}$ -sample algorithm, the running time is $2^{\Omega(d)}$. No proper learning algorithm with faster run-time (i.e. $2^{o(d)}$) was known.

Through a new connection with LCAs, we close this gap by giving a $2^{\tilde{O}(\sqrt{d}/\varepsilon)}$ -time algorithm for this problem. Note that the running time essentially matches that of the aforementioned improper learning algorithm of [25]. Moreover, our algorithm is essentially optimal, due to the $2^{\tilde{\Omega}(\sqrt{d})}$ query lower bound of [20].

1.4.1 Main Results

The following are our main results: learning and distance approximation of Boolean functions, and local correction of real-valued functions.

Theorem 4. [*Agnostic proper learning of monotone functions*⁴] *There is an algorithm that runs in time $2^{\tilde{O}(\frac{\sqrt{d}}{\varepsilon})}$ and, given uniform sample access to an unknown function $f : \{\pm 1\}^d \rightarrow \{\pm 1\}$, with probability at least $1 - \frac{1}{2^d}$, outputs a succinct representation of a monotone function $g : \{\pm 1\}^d \rightarrow \{\pm 1\}$ that is $\text{opt} + O(\varepsilon)$ -close to f , where opt is the distance from f to the closest monotone function (i.e. the fraction of elements of $\{\pm 1\}^d$ on which f and its closest monotone function disagree).*

The corollary below follows immediately by the standard method of [98] that runs the learning algorithm in Theorem 4 and estimates the distance between g and f .

⁴See Section 4.9 for an extension to functions with randomized labels.

Corollary 2 (Additive distance-to-monotonicity approximation). *There is an algorithm with running time and sample complexity $2^{\tilde{O}(\frac{\sqrt{d}}{\varepsilon})}$ that outputs some estimate est of the distance from f to the closest monotone function f_{mon} . With probability at least $1 - 2^{-d+1}$, this estimate satisfies the guarantee*

$$\text{dist}(f, f_{mon}) \leq est \leq \text{dist}(f, f_{mon}) + O(\varepsilon).$$

Our main result, [Theorem 4](#), builds on an algorithm that is also of independent interest. It is a local corrector for real-valued functions that are close to monotone in ℓ_1 distance.

Theorem 5. [*Local monotonicity correction of real-valued functions*] *Let P be a poset with N elements, such that every element has at most Δ predecessors or successors and the longest directed path has length h . Let $f : P \rightarrow [-1, 1]$ be α -close to monotone in ℓ_1 distance. There is an LCA that makes queries to f and outputs queries to $g : P \rightarrow [-1, 1]$, such that g is monotone and $\|f - g\|_1 \leq 2\alpha + 3\varepsilon$. The LCA makes $(\Delta \log N)^{O(\log h \log(1/\varepsilon))}$ queries, uses a random seed of length $\text{poly}(\Delta \log N)$, and succeeds with probability $1 - N^{-10}$.*

1.4.2 Technical Overview

As in [Chapter 3](#), there is an relaxed learning phase and a correction phase. We have essentially three steps:

1. Find a real-valued polynomial P that is ε -close to some monotone function, $(\text{opt} + \varepsilon)$ -close⁵ to the unknown function f in ℓ_1 distance, and bounded in $[-1, 1]$.
2. Obtain a succinct description of a real-valued function $P_{\text{CORRECTED}}$ that is monotone, and $O(\varepsilon)$ -close to P in ℓ_1 distance.
3. Round the real-valued function $P_{\text{CORRECTED}}$ to be $\{\pm 1\}$ -valued, while preserving monotonicity and closeness to f .

For step (3) we use the rounding procedure of [\[77\]](#) that rounds real-valued functions to $\{\pm 1\}$ -valued functions, and we show that this procedure also preserves monotonicity. Our main contributions are in steps (1) and (2).

Learning a polynomial that is close to monotone. The improper learning phase is constrained to produce a good predictor that is ε -close to some monotone function, regardless of how far f may be from monotone. Existing improper learning algorithms are far from satisfying this new requirement. We design a new improper learner by combining the polynomial-approximation based techniques of [\[25, 59, 77\]](#) with graph LCAs and the *ellipsoid method* for convex optimization.

⁵Since opt is unknown, we instead guess values of opt in increments of ε .

The improper learning task is a convex feasibility problem; the set of polynomials satisfying the constraints we give in step (1) is a convex subset of the initial convex set of low-degree real polynomials. The ellipsoid method requires a *separation oracle*, i.e. some way to efficiently generate a hyperplane separating a given infeasible polynomial from the feasible region. Such hyperplanes are themselves low-degree real polynomials, which have high inner product with the infeasible polynomial and low inner product with every point in the feasible region. The separator for the set of polynomials that are $(\alpha + \varepsilon)$ -close to f is just the gradient of the prediction error; the more interesting case is the separator for the set of polynomials that are ε -close to monotone.

With an argument inspired by the characterization of Lipschitz functions given in [14], we observe that if a real-valued polynomial P is far from monotone, this can be witnessed by a large matching on the pairs of elements on which P violates monotonicity. Given any description of the matching, we show how to extract a separating hyperplane for P by evaluating the matching on a set of sample points. Therefore, the challenge is to find a description of a sufficiently large matching that can also be evaluated quickly. We elaborate on this in the next section.

Local monotonicity correction by sorting on partially ordered sets. Our algorithm for step (2) solves a new problem that we call the *poset-sorting problem*.

Let P be a poset of N elements with longest chain length at most h and such that every element has at most Δ predecessors or successors. Given a labeling f of elements of P , output a new labeling f_{mon} that (i) is monotone with respect to the partial order in P (ii) can be obtained from f by a sequence of swaps of monotonicity-violating label pairs.

The greedy algorithm for this task simply swaps monotonicity-violating pairs of labels until there are none left; it is not hard to show that this algorithm produces a monotone function whose distance to f is at most twice the distance of f to monotone. The challenge is to improve the efficiency by doing this in a local fashion. We accomplish this by bringing in recent exciting developments on parallelizing greedy algorithms [63–65]. These developments accomplish speedups for classic graph problems in the setting of local and distributed computing. They build on a large body of work in local computation algorithms and distributed graph algorithms: some examples include [3,7,24,34,57,58,70,72,90,99,101,103].

We reduce the problem first to the case where the labels are in $\{0, 1\}$. In this case, when the longest chain length h is 1, the poset-sorting problem is equivalent to the classical problem of finding a maximal matching on a bipartite graph with N vertices and maximum degree at most Δ . We note that a recent LCA by Ghaffari [64] handles this problem using a run-time of only $\text{poly}(\Delta, \log(N/\delta))$.

For larger values of h , a naive approach would be to execute a sequence of phases, in each of which a maximal matching between monotonicity-violating labels is produced and the labels that are matched with each other are swapped. One can show that $O(h)$ such phases suffice and sometimes necessary if the matchings are arbitrary. If the algorithm by Ghaffari [64] is used to implement each of the phases, this yields a running time of $(\Delta \log(N/\delta))^{\Theta(h)}$.

For properly learning monotone functions over the Boolean cube, the parameters⁶ we are primarily interested in are $h = \Theta(\sqrt{d})$, $\Delta = 2^{\tilde{\Theta}(\sqrt{d})}$ and $N = \Theta(2^d)$. The naive approach then gives us a running time of $(\Delta \log(N/\delta))^{\Theta(h)} = 2^{\Omega(d)}$, which is too slow for our purposes.

We beat the naive approach by enforcing that the maximal matchings at each step only include pairs of vertices that are sufficiently far away in the graph. We show that after each matching step, the greatest distance between pairs that violate monotonicity reduces by a factor of 2. This allows us to reduce the exponent from h to $\log h$, which is sufficiently fast to yield an essentially optimal proper learning algorithm for monotone functions.

1.5 Local Lipschitz Filters

In this next chapter, we study local Lipschitz filters for real-valued functions, with the main application being *differentially-private evaluation* of black-box functions. A local filter is a specific type of local corrector, introduced by Jha and Raskhodnikova [76], who motivated their research by an application in private data analysis.

The special condition that makes a local corrector a filter is that when the input function f has the desired property, then the output function g is guaranteed to be equal to f . In other words, a local filter is a local corrector that does not modify functions that already have the desired property. If, in addition, the distance between f and g is relatively small compared to the distance from f to the nearest function with the desired property, the filter is called *distance-respecting*.

1.5.1 Main Results

Motivated by the applications, we consider functions over $[n]^d$, where $[n]$ is a shorthand for $\{1, \dots, n\}$. A function $f : [n]^d \rightarrow \mathbb{R}$ is called *c-Lipschitz* if increasing or decreasing any coordinate by one can only change the function value by c . The parameter c is called the *Lipschitz constant* of f . A 1-Lipschitz function is simply referred to as Lipschitz⁷. Intuitively, changing the argument to the Lipschitz function by a small amount does not significantly change the value of the function.

In previous work, only unbounded-range functions were considered in the context of Lipschitz reconstruction. Jha and Raskhodnikova [76] obtained a deterministic local filter that runs in time $O((\log n + 1)^d)$. This direction of research was halted by a strong lower bound obtained by Awasthi et al. [11]. They showed that every local Lipschitz filter, even with significant additive error, needs exponential in the dimension d number of lookups.

⁶These are the parameters for small constant ε . A slight subtlety in our argument is that we need to work over truncated hypercube, i.e. handle separately $O(\varepsilon)$ fraction of the elements with too high or too low Hamming weight. This is a standard technique, and in our case it makes the parameter Δ in our poset small enough for us to get a good run-time guarantee using our poset sorting algorithm.

⁷In this work, we focus on reconstruction to Lipschitz functions. All our results extend to the class of *c-Lipschitz* functions via scaling all function values by a factor of c .

We demonstrate that important applications of local Lipschitz filters can be accomplished with filters for functions whose range is bounded in $[0, r]$. By focusing on this class of functions, we circumvent the lower bound from [11] and achieve running time polynomial in d for constant r . Moreover, our filters satisfy additional accuracy guarantees compared to the filter in [76], which is only required to (1) give access to a Lipschitz function g ; (2) ensure that $g = f$ if the input function f is Lipschitz. Our filters achieve an additional feature of being *distance-respecting*, i.e., they ensure that g is close to f . We provide this feature with respect to both ℓ_1 -distance ($\|f - g\|_1 = \mathbb{E}_x[|f(x) - g(x)|]$) and ℓ_0 -distance ($\|f - g\|_0 = \Pr_x[f(x) \neq g(x)]$), where the expectation and the probability are taken over a uniformly distributed point in the domain. The distance of f to Lipschitzness is defined as the minimum over all Lipschitz functions g of the distance from f to g , and can be considered with respect to both norms. Our filters are *distance-respecting* in the following sense: the distance between the input function f and the output function g is at most twice the distance of f to the Lipschitz property (w.h.p.).

Our algorithms work for functions over general graphs. To facilitate comparison with prior work and our lower bound, we state their guarantees only for the $[n]^d$ domain. (It can be represented by the d -dimensional hypergrid graph $\mathcal{H}_n^d$.) Our first local Lipschitz filter is distance-respecting with respect to the ℓ_1 -distance.

Theorem 25 (ℓ_1 -filter, informal). *There is an algorithm $\mathcal{A}$ that, given lookup access to a function $f : [n]^d \rightarrow [0, r]$, a query $x \in [n]^d$, and a random seed $\rho \in \{0, 1\}^*$, has the following properties:*

- **Efficiency:** $\mathcal{A}$ has lookup and time complexity $(d^r \cdot \text{polylog } n)^{O(\log r)}$ per query.
- **Consistency:** With probability at least $1 - n^{-d}$ over the choice of ρ , algorithm $\mathcal{A}$ provides query access to a 1.01-Lipschitz function g_ρ with $\|g_\rho - f\|_1$ at most twice the ℓ_1 -distance from f to Lipschitzness.

If f is Lipschitz, then the filter outputs $f(x)$ for all queries x and random seeds.

Our second local Lipschitz filter is distance-respecting with respect to ℓ_0 . Unlike the filter in Theorem 25, the ℓ_0 -respecting filter provides access to a 1-Lipschitz function.

Theorem 26 (ℓ_0 -filter, informal). *There is an algorithm $\mathcal{A}$ that, given lookup access to a function $f : [n]^d \rightarrow [0, r]$, a query $x \in [n]^d$, and a random seed $\rho \in \{0, 1\}^*$, has the following properties:*

- **Efficiency:** $\mathcal{A}$ has lookup and time complexity $d^{O(r)} \text{polylog}(n)$.
- **Consistency:** With probability at least $1 - n^{-d}$ over the choice of ρ , algorithm $\mathcal{A}$ provides query access to a 1-Lipschitz function g_ρ with $\|g_\rho - f\|_0$ at most twice the ℓ_0 -distance from f to Lipschitzness.

If f is Lipschitz, then the filter outputs $f(x)$ for all queries x and random seeds.

An important special case of the hypergrid domain is the hypercube $\{0, 1\}^d$. Prior to our work, no local Lipschitz filter for the hypercube domain could avoid lookups on the entire domain (in the worst case). Our algorithms do that for the case when $r \leq d/\log^2 d$. Moreover, we show that our filters are nearly optimal in terms of their dependence on r for this domain. We present a theorem showing that the running time of $d^{\tilde{\Omega}(r)}$ is unavoidable in Theorems 25 and 26; thus, our filters are nearly optimal. Moreover, even local Lipschitz filters that are not distance-respecting (as in [76]) must still run in time $d^{\tilde{\Omega}(r)}$.

Theorem 27 (Filter lower bound, informal). *Let $\mathcal{A}(x, \rho)$ be an algorithm that, given lookup access to a function $f : \{0, 1\}^d \rightarrow [0, r]$, a query $x \in \{0, 1\}^d$, and a random seed $\rho \in \{0, 1\}^*$, has the following property:*

- **Weak Consistency:** *With probability at least $\frac{3}{4}$ over the choice of ρ , the algorithm $\mathcal{A}(\cdot, \rho)$ provides query access to a 1-Lipschitz function g_ρ such that whenever f is 1-Lipschitz $g_\rho = f$.*

Then, for all integers $r \geq 4$ and $d \geq \Omega(r)$, there exists a function $f : \{0, 1\}^d \rightarrow [0, r]$ for which the lookup complexity of $\mathcal{A}$ is $(\frac{d}{r})^{\Omega(r)}$.

The lower bound for local Lipschitz filters in [10] applies only to filters that are guaranteed to output a Lipschitz function (or a Lipschitz function with small error in values) for all random seeds. They can still have a constant error probability, but the only mode of failure allowed is returning a function that is far from f when f already satisfied the property. Awasthi et al. [10] suggest as a future research direction to consider local filters whose output does not satisfy the desired property $\mathcal{P}$ with small probability and mention that their techniques do not work for this case. We overcome this difficulty by providing different techniques that work for filters whose output fails to satisfy the Lipschitz property with constant error probability. In particular, Theorem 27 applied with $r = \Theta(d)$ yields the lookup lower bound of $2^{\Omega(d)}$, as in [10], but without their restriction on the mode of failure, thus answering their open question.

1.5.2 Applications

We showcase two applications of our local Lipschitz filters: to private data analysis and to tolerant testing. Both of them deal with real-valued functions with no a priori bound on the range.

Application to black-box privacy. Our first application is for providing differentially private mechanisms for releasing outputs of black-box functions, even in the presence of malicious clients. Differential privacy, introduced by [56], is an accepted standard of privacy protection for releasing information about sensitive datasets. A sensitive dataset can be modeled as a point x in $\{0, 1\}^d$, representing whether each of the d possible types of individuals

is present in the data⁸. More generally, it is modeled as a point x in $[n]^d$ that represents a histogram counting the number of individuals of each type.

Definition 1 (Differential Privacy). *Two datasets $x, x' \in [n]^d$ are neighbors if vertices x, x' are neighbors in the hypergrid $\mathcal{H}_n^d$. For privacy parameters $\varepsilon > 0$ and $\delta \in (0, 1)$, a randomized mechanism $\mathcal{M} : [n]^d \rightarrow \mathbb{R}$ is (ε, δ) -differentially private if, for all neighboring $x, x' \in [n]^d$ and all measurable sets $Y \subset \mathbb{R}$,*

$$\Pr[\mathcal{M}(x) \in Y] \leq e^\varepsilon \Pr[\mathcal{M}(x') \in Y] + \delta.$$

When $\delta = 0$, then we call $\mathcal{M}$ purely differentially private; otherwise, it is approximately differentially private.

A statistic (or any information about the sensitive dataset) is modeled as a function $f(x)$. One of the most commonly used building blocks in the design of differentially private algorithms is the Laplace mechanism⁹. The Laplace mechanism computing on a sensitive dataset x can approximate the value $f(x)$ for a desired c -Lipschitz function f by adding Laplace noise proportional to c to the true value $f(x)$. The noisy value is safe to release while satisfying differential privacy.

When f (supplied by a distrusted client) is given as a general purpose program, it is hard to compute its least Lipschitz constant, or even an upper bound on it. The data curator can ask the client to supply the Lipschitz constant for the query function f . However, as noted in [76], even deciding if f has Lipschitz constant at most c is NP-hard for functions over the finite domains we study (if f is specified by a circuit). Applying the Laplace mechanism with c smaller than a Lipschitz constant (if the client supplied incorrect c) would result in a privacy breach, while applying it with a generic upper bound on the least Lipschitz constant of f would result in overwhelming noise.

To the best of our knowledge, no existing (ε, δ) -DP mechanisms for the black-box privacy problem simultaneously achieve a runtime that is polynomial in d and $\log \frac{1}{\delta}$ while providing an accuracy guarantee that is comparable to the Laplace mechanism. One solution to the black-box privacy problem was proposed by Jha and Raskhodnikova [76] who designed the following *filter mechanism*: *A client who does not have direct access to x can ask the data curator for information about the dataset by specifying a Lipschitz¹⁰ function f . The data curator can run a local filter to obtain a value $g(x)$, where g is guaranteed to be Lipschitz. Then the curator can use the Laplace mechanism and release the obtained noisy value.* If the client is truthful (i.e., the function is Lipschitz), then, by the filter properties, the accuracy guarantee of the filter mechanism is inherited from the Laplace mechanism. However, even if f is not Lipschitz, the filter ensures that privacy is still preserved. Observe that the running time and accuracy of the filter mechanism directly depends on the running time and accuracy

⁸A point $x \in \{0, 1\}^d$ can also represent a dataset containing data of d individuals, with one bit per individual: e.g., $x_i = 1$ could indicate that the individual i has a criminal record or some illness.

⁹Gaussian mechanism is another popular differentially private algorithm that calibrates noise to the Lipschitz constant of f ; for simplicity, we focus on the Laplace mechanism as a canonical example.

¹⁰If the function is c -Lipschitz, it can be rescaled by dividing by c .

of the local Lipschitz filter. The lower bound by Awasthi et al. [11] on the complexity of local Lipschitz filters implies $2^{\Omega(d)}$ running time for filter mechanisms, even for releasing functions of the form $f : \{0, 1\}^d \rightarrow \mathbb{R}$.

We provide a mechanism for privately releasing outputs of black-box functions, even in the presence of malicious clients, in time that is quasi-polynomial in the dimension d while providing accuracy comparable to the Laplace mechanism. For bounded-range functions, the running time of our mechanism is polynomial in d and $\log \frac{1}{\delta}$. We bypass the lower bound in [11] by using the filter mechanism for bounded-range functions repeatedly to simulate a noisy binary search. Our mechanism needs only query access to the input function, that is, it can be specified as a black box (e.g., as a complicated or obfuscated program).

Theorem 31 (Binary search filter mechanism, informal). *For all $\varepsilon > 0$ and $\delta \in (0, 1)$, there exists an (ε, δ) -differentially private mechanism $\mathcal{M}$ that gets lookup access to a function $f : [n]^d \rightarrow [0, r]$ and has the following properties. Let $\kappa = \log \min(r, nd)$.*

- **Efficiency:** *The lookup and time complexity of $\mathcal{M}$ are $d^{O(\frac{1}{\varepsilon} \kappa \log \kappa)}$ $\text{polylog } \frac{n}{\delta}$.*
- **Accuracy:** *If f is Lipschitz, then for all $x \in [n]^d$, we have $\mathcal{M}(x) \sim f(x) + \text{Laplace}(\frac{\kappa}{\varepsilon})$ with probability at least 0.99.*

Application to tolerant testing. The second application we present is to tolerant testing of the Lipschitz property of real-valued functions on the hypercube domains. In the standard property testing terminology, a *property* $\mathcal{P}$ is a set of functions. Given a parameter $\varepsilon \in (0, 1)$, a function f is ε -far from $\mathcal{P}$ if at least an ε fraction of function values have to change to make $f \in \mathcal{P}$; otherwise, f is ε -close to $\mathcal{P}$. Given parameters $\varepsilon_0, \varepsilon \in (0, 1)$ with $\varepsilon_0 < \varepsilon$ and query access to an input function f , an $(\varepsilon_0, \varepsilon)$ -tolerant tester for $\mathcal{P}$ accepts with probability at least $2/3$ if f is ε_0 -close to $\mathcal{P}$ and rejects with probability at least $2/3$ if f is ε -far from $\mathcal{P}$. For the special case when $\varepsilon_0 = 0$, the corresponding computational task is referred to as (standard) *testing*.

Testing of the Lipschitz property was introduced in [76] and subsequently studied in [11, 14, 30, 32, 52, 53, 78]. Lipschitz testing of functions $f : \{0, 1\}^d \rightarrow \mathbb{R}$ can be performed with $O(\frac{d}{\varepsilon})$ queries [32, 76]. In contrast, prior to our work, no nontrivial tolerant tester was known for this property. As shown in [60], tolerant testing can have drastically higher query complexity than standard testing: some properties have constant-query testers, but no sublinear-time tolerant testers. Moreover, $\exp(d^{1/4})$ queries are required for tolerantly testing the Lipschitz property with nonadaptive algorithms (i.e., algorithms that specify all queries in advance, before receiving any answers). This result follows from the monotonicity-to-Lipschitzness reduction of Chakrabarty et al. [30] and the lower bounds for tolerantly testing monotonicity [35, 97]). Though our algorithm is adaptive, the existence of this lower bound provides evidence that the problem may be inherently hard.

As an application of our local filters, we construct the first nontrivial tolerant Lipschitz tester (see Theorem 32) for functions $f : \{0, 1\}^d \rightarrow \mathbb{R}$. Our tester is $(\varepsilon, 2.01\varepsilon)$ -tolerant and has query and time complexity $\frac{1}{\varepsilon^2} d^{O(\sqrt{d \log(d/\varepsilon)})}$. We stress that this tester handles functions

with any range. Given that Lipschitz functions on $\{0, 1\}^d$ can have range $[0, d]$, and our ℓ_0 -filter has time complexity $d^{O(r)}$ for functions $f : \{0, 1\}^d \rightarrow [0, r]$, one might expect our tester to run in time $\exp(d)$ when no a priori upper bound on f is available. We leverage additional structural properties of Lipschitz functions to reduce this to $\exp(\sqrt{d})$.

1.6 Related Work

Adversarially robust learning. Several recent works study adversarially robust learning in the distribution-free PAC model. This line of work considers a range of perturbation models, including ℓ_p balls and fully arbitrary perturbation sets. The works of [40,95] show that robust learning and standard learning are essentially equivalent from a statistical perspective; therefore, separations between the two tasks must be computational. Such computational separations were established in [9,26,44], which exhibit concept classes and perturbation sets for which robust learning is computationally harder than standard learning under standard hardness assumptions. The work of [106] gives examples where high robust risk, i.e., high prediction error under adversarial perturbations, is unavoidable.

For halfspaces, [94] gives an algorithm for robust learning in a realizable setting with random classification noise. The works of [47,51] give algorithms for robust, proper, agnostic learning of halfspaces, but assume that the input distribution is supported on the unit ball. When these algorithms are scaled to the regime considered here, where distributions are concentrated on a radius- $\sqrt{d}$ ball, their running times have an exponential dependence on $\sqrt{d}$. The work of [9] also studies robust learning in the distribution-free setting, giving algorithms for learning halfspaces and degree-2 PTFs with robustness to ℓ_∞ -bounded perturbations. In distribution-specific settings, [71] gives an algorithm for learning monotone conjunctions robust to perturbations of $O(\log n)$ Hamming distance under log-Lipschitz distributions on the Hamming cube. As mentioned earlier, the proper agnostic learner of [48] is a robust learner for halfspaces under the Gaussian distribution.

Learning halfspaces. Under general log-concave distributions, [8] gives a semiagnostic proper learner for origin-centered halfspaces with error $O(\text{opt}) + \varepsilon$, where opt is the error of the best halfspace. In contrast, our work aims for the optimal $\text{opt} + \varepsilon$ error guarantee. For every constant α , [42] gives a polynomial-time improper learner with error $(1 + \alpha)\text{opt} + \varepsilon$ using polynomial regression, but the algorithm applies only to the Gaussian distribution and not to general sub-Gaussian log-concave distributions. The work of [48] gives a $d^{\tilde{O}(1/\varepsilon^4)}$ -time algorithm achieving error $\text{opt} + \varepsilon$ for halfspaces under the Gaussian distribution, as discussed in Section 1.3, and also gives a polynomial-time proper learner for origin-centered halfspaces with error $(1 + \alpha)\text{opt} + \varepsilon$. These algorithms are highly specific to Gaussian distributions and do not extend, for example, to the uniform distribution over $[-1, 1]^d$ or to general sub-Gaussian log-concave distributions.

Monotonicity correction, testing, and learning. Local correction of monotonicity has been studied in several parameter regimes [1,2,10,15,16,84,104]. The work of [2] introduced online property reconstruction and gave an algorithm for correcting monotonicity of real-valued functions on the discrete line. The work of [98] gives a tolerant tester in the same

setting. Lower bounds for monotonicity correction in other error regimes are given in [10,15]; see also [84] for an overview of previously available monotonicity-correction algorithms and lower bounds. The related problem of approximating distance to monotonicity, which is closely connected to tolerant testing, has been studied in [1,97,98].

Proper learning algorithms are known for restricted classes of monotone functions [5,22,75,111], and [28] give learning algorithms for function classes that generalize monotone functions. Weak learning of monotone functions has also been studied [4,23,25,79,96]; this line of work obtains very fast algorithms at the cost of accuracy only $\frac{1}{2} + \frac{1}{\text{poly}(n)}$. Monotonicity testing has been studied beyond the Boolean cube, including on hypergrids [14,18,19,21,31,32] and for functions with bounded influence [33].

Private data analysis. Several systems allow analysts to query a sensitive dataset while satisfying differential privacy, including PINQ [93], Airavat [102], Fuzz [74], and PSI [62]. These systems support the release of noisy approximations to certain real-valued functions of the dataset. Typically, the analyst sends a program to the server, requesting that it be evaluated on the private dataset, and receives the output with noise added. The program may be composed from a limited set of trusted built-in functions, such as sum and count, together with a limited set of untrusted data transformations, such as combining several types of individuals into one type. The Lipschitzness of these transformations can be enforced using programming-language tools.

Chapter 2

Preliminaries

2.1 Basic Notation

In this work, the main objects of interest are classifiers, which we will sometimes express as labeling functions ($f : \mathcal{X} \rightarrow \mathcal{Y}$) and sometimes as joint distributions over the input and output domains ($\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$). We say that f is a *randomized function* over $\mathcal{X}$ if f maps every point in $\mathcal{X}$ to a random variable $f(x)$ over $\mathcal{Y}$.

When dealing with a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, we will sometimes overload the notation to use the symbol $\mathcal{D}$ to refer to the $\mathcal{X}$ -marginal of $\mathcal{D}$. Analogously, when dealing with a collection S of pairs $\{(x_i, y_i)\}$ in $\mathcal{X} \times \mathcal{Y}$ we will sometimes write $\Pr_{x \sim S}[\dots]$ to refer to $\Pr_{(x,y) \sim S}[\dots]$ when the values of y are not referenced.

which we will refer to as the labeling function.

2.1.1 Distances and Errors

In this work, we will typically use the ℓ_1 distance to denote the distance between functions over a distribution:

$$\|f_1 - f_2\|_{1,\mathcal{D}} = \mathbb{E}_{x \sim \mathcal{D}} [|f_1(x) - f_2(x)|].$$

We will also sometimes use the ℓ_0 distance to denote the distance between Boolean-valued functions.

$$\|f_1 - f_2\|_{0,\mathcal{D}} = \Pr_{x \sim \mathcal{D}} [f_1(x) \neq f_2(x)].$$

When $\mathcal{Y} = \{0, 1\}$, these distances are equivalent, and when $\mathcal{Y} = \{-1, 1\}$ they differ by a factor of 2. When a different distance function is used it will be specified explicitly, and when the distribution or the distance function is clear from context, the subscript may be omitted.

Given a distance function `dist`, we denote the distance of a function f to a class $\mathcal{F}$ as

$$\text{dist}(f, \mathcal{F}) = \inf_{g \in \mathcal{F}} \text{dist}(f, g).$$

The terms “distance to monotonicity” and “distance to Lipschitzness” are used to refer to the distance of a function to the class of monotone functions and the class of Lipschitz functions, respectively. In learning contexts, we will also sometimes use the notation `opt` to refer to the distance between the ground truth labeling function and the concept class of interest (linear threshold functions in [Chapter 3](#), monotone functions in [Chapter 4](#)).

2.1.2 Graphs and Posets

Let x and y be elements of a poset P .¹ We use $\preceq$ to denote the ordering relation on P . We say $x \prec y$ (“ x is a predecessor of y ”) if $x \preceq y$ and $x \neq y$. Also, we say $x \succeq y$ if $y \preceq x$, and $x \succ y$ (“ x is a successor of y ”) if $y \prec x$. We say x and y are *incomparable* if neither $x \prec y$ nor

¹In this work all posets are assumed to be finite.

$x \succ y$ holds. We say that x is an *immediate predecessor* of y if $x \prec y$ and there is no z in P for which $x \prec z \prec y$. The notion of an *immediate successor* is defined analogously.

The *graph* of a poset P (a.k.a. *Hasse diagram* of P) is a directed graph, in which elements of P are the vertices, and there is an edge from x to y whenever x is an immediate predecessor of y . Clearly, the graph of any poset is a DAG. Additionally, it is immediate that the poset itself is unambiguously determined by its graph, and we will refer to the poset and its graph interchangeably.

Definition 2 (Monotone functions). *For a poset P and a totally ordered set Y , a function $f : P \rightarrow Y$ is monotone if whenever $x \preceq y$ we have $f(x) \leq f(y)$.*

Definition 3 (Discrete c -Lipschitz functions). *Fix a constant $c > 0$ and a graph $G = (V, E)$. A function $f : V \rightarrow \mathbb{R}$ is c -Lipschitz w.r.t. G if $|f(x) - f(y)| \leq c \cdot \text{dist}_G(x, y)$ for all $x, y \in V$. A 1-Lipschitz function is simply referred to as Lipschitz. We denote by $\mathcal{Lip}(G)$ the set of Lipschitz functions w.r.t. G .*

Definition 4 (Hypergrid, hypercube, and truncated hypercube). *The d -dimensional n -hypergrid $\mathcal{H}_n^d$ is the set $[n]^d$. For $x, y \in [n]^d$, we say $x \preceq y$ if $x_i \leq y_i$ for all $i \in \{1, \dots, d\}$. It is immediate that $[n]^d$ is a poset with n^d elements, and that in its graph, two vertices are adjacent if they differ by one in one coordinate and agree everywhere else. We will typically refer to $\mathcal{H}_2^d$ as the d -dimensional Boolean hypercube and denote it simply $\mathcal{H}^d$.*

We also define the truncated hypercube

$$\mathcal{H}_\varepsilon^d := \left\{ x \in \{0, 1\}^d : \left| \sum_i x_i - \frac{d}{2} \right| \leq \sqrt{\frac{d}{2} \log \frac{2}{\varepsilon}} \right\},$$

Via Hoeffding's bound, we have that the fraction of elements in $\{0, 1\}^d$ that are not also in $\mathcal{H}_\varepsilon^d$ is at most $2 \exp\left(-\frac{2t^2}{d}\right) = \varepsilon$.

2.2 Learning Algorithms

2.2.1 Agnostic Learning

In distribution-specific agnostic learning, a learning algorithm is given i.i.d. example-label pairs from some distribution over $\mathcal{X} \times \mathcal{Y}$, where the marginal distribution over $\mathcal{X}$ is known. The *generalization error* (which we also refer to simply as *error*) of a predictor is its distance to the ground truth labeling function; for instance if the distance function is ℓ_0 distance, this is the probability it misclassifies a fresh example-label pair. The goal of agnostic learning is to produce a predictor $\hat{g}$ with good generalization error. Guarantees are produced in terms of **opt**, the lowest generalization error among all hypotheses in the function class.

We state some known results about learning monotone functions over the hypercube.

Theorem 6 (Learnability of monotone functions [25]). *There is an algorithm that, for any monotone function $g : \{0, 1\}^d \rightarrow \{0, 1\}$, given i.i.d. example-label pairs $(x_i, g(x_i))$, with x_i uniform in $\{0, 1\}^d$, returns a circuit computing a predictor $\hat{g}$, such that $\|g - \hat{g}\|_0 \leq \varepsilon$ over the uniform distribution. The algorithm uses $d^{O(\frac{\sqrt{d}}{\varepsilon})} \log(\frac{1}{\delta})$ samples and run-time, where δ is the failure probability bound.*

From Remark 4 on page 6 of [77] and standard failure probability reduction via repetition, we also have:

Theorem 7 (Agnostic learnability of monotone functions). *In the agnostic setting with examples distributed uniformly on $\{0, 1\}^d$, there is an algorithm that returns a circuit with generalization error at most $\text{opt} + \varepsilon$, where opt is the error of the best monotone function. The algorithm uses $d^{O(\frac{\sqrt{d}}{\varepsilon^2})} \log(\frac{1}{\delta})$ samples and run-time, where δ is the failure probability bound.*

We will also use a bound on how well monotone functions can be approximated by low-degree polynomials. This fact follows from [25, 77] and a refinement by [59]:

Fact 1. *For every monotone $f : \{-1, 1\}^d \rightarrow \{-1, 1\}$ and $\varepsilon > 0$, there exists a multilinear polynomial p of degree $\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \rceil$ such that*

$$\|f - p\|_1 \leq \varepsilon.$$

2.2.2 Convex Optimization

The following notion is standard in convex optimization.

Definition 5. A **separation oracle** for a convex set $\mathcal{C}_{\text{convex}}$ is an oracle that given a point x does one of the following things:

- If $x \in \mathcal{C}_{\text{convex}}$, then the oracle outputs “Yes”.
- If $x \notin \mathcal{C}_{\text{convex}}$, then the oracle outputs $(\text{No}, Q_{\text{separation}})$, where $Q_{\text{separation}} \in \mathbb{R}^d$ represents a direction along which x is separated from $\mathcal{C}_{\text{convex}}$. Formally, $\langle Q_{\text{separation}}, x \rangle > \langle Q_{\text{separation}}, x' \rangle$ for any x' in $\mathcal{C}_{\text{convex}}$.

We will need the following well-known fact from convex optimization:

Fact 2. [80] *There is an algorithm ELLIPSOIDALGORITHM that takes as inputs positive real values r and R , and access to a separation oracle for some convex set $\mathcal{C}_{\text{convex}} \subset \{x \in \mathbb{R}^d : \|x\| \leq R\}$. The algorithm runs in time $\text{poly}(d, \log \frac{R}{r})$ and either outputs an element in $\mathcal{C}_{\text{convex}}$ or outputs FAIL. Furthermore, if $\mathcal{C}_{\text{convex}}$ contains a ball of radius r , the algorithm is guaranteed to succeed.*

Also see [88] for an overview of algorithms building on [80].

2.3 Local Computation Algorithms and Succinct Representations

The Local Computation Algorithm (LCA) model captures the ability to provide query access to parts of an output in sublinear time. In this work, we use the LCA model to describe algorithms for various tasks, including violation graph access, maximal matching, and local property correction. An LCA is given access to a random bit-string and to the input: for example, in the case of maximal matching, this input is the adjacency list of the graph. Upon receiving an edge query e in G , the LCA should respond “yes” or “no.” Responses to different edges must be consistent with a single legal maximal matching. Similarly, for poset sorting on poset P , a query to the LCA is any element $x \in P$, and the LCA, which is provided with access to function $f : P \rightarrow \{0, 1\}$, must respond with $f_{\text{mon}}(x)$ so that the function f_{mon} is monotone. Responses to different inputs must be consistent with a single legal sorting of f . In both cases, the LCA’s answers may depend on a random bit-string² that persists between queries, but otherwise, responses must be³ *memoryless* – i.e., they cannot depend on previous queries to or responses from the LCA.

Definition 6 (Graph LCA). *Fix $\delta \in (0, 1)$. A graph LCA $\mathcal{A}(x, \rho)$ is a randomized algorithm that gets adjacency list access⁴ to an input graph $G = (V, E)$, a query $x \in V$, and a random seed $\rho \in \{0, 1\}^*$. For each ρ , the set of outputs $\{\mathcal{A}(x, \rho) : x \in V\}$ is consistent with some object defined with respect to G , such as a maximal matching in G . The fraction of possible random strings for which $\mathcal{A}$ fails (i.e., defines an object that does not satisfy the constraints) of the problem, is at most δ .*

Local filters were introduced by Saks and Seshadhri [104] and first studied for Lipschitz functions by Jha and Raskhodnikova [76].

Definition 7 (Local Lipschitz filter). *For all $c > 0$ and $\delta \in (0, 1)$, a local (c, δ) -Lipschitz filter⁵ over a graph $G = (V, E)$ is an algorithm $\mathcal{A}(x, \rho)$ that gets a query $x \in V$ and a random seed $\rho \in \{0, 1\}^*$, as well as lookup access to a function $f : V \rightarrow \mathbb{R}$ and adjacency lists access to G . With probability at least $1 - \delta$ (over the random seed), the filter $\mathcal{A}$ provides query access to a c -Lipschitz function $g_\rho : V \rightarrow \mathbb{R}$ such that whenever f is c -Lipschitz $g_\rho = f$. In addition, for all $\lambda > 0$, the filter is ℓ_p -respecting with blowup λ if $\|f - g_\rho\|_p \leq \lambda \cdot \ell_p(f, \mathcal{Lip}(G))$ whenever g_ρ is c -Lipschitz.*

In terms of performance, we want the following quantities to be as small as possible (i) the *query complexity*, i.e. the number of probes to the input the LCA makes to respond to a

²In all cases we consider, the random bit-string is short enough for the LCA to read as a whole (in contrast, in some work exponentially long bit-strings are considered).

³Sometimes LCAs are considered that are not memoryless, but in this work whenever we refer to an LCA, we imply it is memoryless.

⁴An adjacency list lookup takes a vertex x and returns the set of vertices adjacent to x .

⁵While the graph is hardcoded in this definition, our filters work when given adjacency list access to any graph.

single query, (ii) the *run-time* the LCA needs to respond to a query, (iii) the length of the random bit-string used by the LCA, (iv) the probability over the random bit-string that the LCA fails to satisfy the problem specifications.

The monotonicity- and Lipschitz-correction algorithms will make use of a recent powerful result⁶ of Ghaffari [64], which gives an efficient algorithm for answering membership queries to a maximal independent set. (We emphasize that *maximal* independent set is defined to be an independent set that cannot be made into a larger independent set by adding an extra vertex, and *maximal matching* is defined analogously.)

Theorem 8 ([64]). *There is an LCA that takes adjacency-lists⁷ access to a graph G with N vertices and largest degree at most Δ , and gives membership access to a maximal independent set on the graph. The query complexity and the run-time of the LCA are $\text{poly}(\Delta, \log(N/\delta))$, the length of the random bit-string is also $\text{poly}(\Delta, \log(N/\delta))$ and the failure probability is at most δ .*

Corollary 3. *There is an LCA that takes adjacency-lists access to a graph G , with N vertices and largest degree at most Δ , and gives membership access to a maximal matching on the graph. The query complexity and the run-time of the LCA is $\text{poly}(\Delta, \log(N/\delta))$, the length of the random bit-string is also $\text{poly}(\Delta, \log(N/\delta))$ and the failure probability is at most δ .*

Proof. This reduction is standard; for completeness we include a proof. For a graph G , one forms the so-called line graph of G , denoted as G' , as follows: (i) the vertex set of G' is the edge set of G , (ii) two vertices are connected in G' if the corresponding edges in G share a vertex. One sees immediately that maximal matchings on G translate to maximal independent sets on G' , and vice versa. Therefore, one can use the LCA of [65] (described here in Theorem 8) to get access to a maximal independent set in G' , which will translate to a maximal matching on G .

An all-neighbor query to G' can be simulated via two all-neighbor queries to G . The graph G' has at most ΔN vertices and degree at most 2Δ . Overall, this means that the query complexity and the run-time of the LCA are still $\text{poly}(\Delta, \log(N/\delta))$, the length of the random bit-string is still $\text{poly}(\Delta, \log(N/\delta))$ and the failure probability is still at most δ . $\square$

⁶We need to comment on some superficial differences between Theorem 8 and the main theorem of [64], which does not mention run-time and bit-string length explicitly and also has $\delta = \text{poly}(\frac{1}{N})$. The following observations are not novel in any way, and some of them are alluded to in [64,65], but we explain them here for completeness. The bound on run-time follows by direct inspection of their algorithm. The length of the bit-string can be reduced to $\text{poly}(\Delta, \log(N/\delta))$ via the standard method [3,90] of replacing i.i.d. random bits with k -wise independent random bits for k large enough ($k = \text{poly}(\Delta, \log(N/\delta))$ suffices, as number of those i.i.d. random bits accessed per query is $\text{poly}(\Delta, \log(N/\delta))$). Although, the failure probability bound δ in the original theorem of [64] is set to be $\text{poly}(\frac{1}{N})$, it can be boosted to arbitrary δ by adding extra disconnected vertices to our graph until this value reaches δ for the new number of vertices N' . Query access to this new graph can be simulated via query access to the original graph with inconsequential overhead. Overall, this costs one an extra $\text{polylog}(1/\delta)$ factor in query complexity and run-time.

⁷I.e. when queried a vertex v , the oracle returns a list of all the neighbors of v .

An important idea in this work is that LCAs (i.e. algorithms that achieve global consistency) can be used to operate on succinct representations of combinatorial objects. To explain further, we need the following definition:

Definition 8 (Succinct representation). *A succinct representation of a function f of size s is a description of f that is stored in s bits of memory and can be evaluated on an input in $O(s)$ time.*

For example, circuits of size s and polynomials of degree $\log s$ are examples of succinct representations of size s and $n^{\log s}$ respectively. The following fact follows immediately from the definition:

Fact 3 (Composition of representations). *If a function f has a description that uses t bits of memory and evaluates in time $O(t)$ given q oracle queries to a function g , and g has a succinct representation of size s , then there is a succinct representation of f of size $O(t + sq)$.*

Now, for example, combining⁸ Fact 3 and Corollary 3 we see immediately that for a graph G , with N vertices and largest degree at most Δ , using the algorithm in Corollary 3 we can transform a size- s representation⁹ of a function computing adjacency-lists access to G into a size- $(\Delta^{O(\log h)} \cdot \text{polylog}(N/\delta) \cdot s)$ representation¹⁰ of a function that determines membership in some maximal matching over G . Note that this transformation itself runs in time $\Delta^{O(\log h)} \cdot \text{polylog}(N/\delta) \cdot s$.

⁸A note on the description sizes of LCAs: because LCAs are uniform (i.e. Turing-machine) algorithms, they can be simulated with a uniform circuit family. For each input size, the size of the corresponding circuit is polynomial in the running time of the LCA for that input size.

⁹For simplicity, in the rest of the paper we will refer to such functions as a “succinct representation of G .”

¹⁰For simplicity, in the rest of the paper we will refer to such functions simply as “representation of a maximal matching.”

Chapter 3

Agnostically Approximating Halfspaces by Robust Classifiers

In this chapter, we present our algorithm for agnostically learning halfspaces under subgaussian log-concave distributions. The algorithm outputs a hypothesis with *boundary volume* $O(r)$ at radius r , providing a level of robustness comparable to that of halfspaces themselves.

3.1 Chapter-Specific Preliminaries

3.1.1 Perturbation and Robustness Models

Definition 9 (r -boundary volume). *The boundary volume of a function f at radius r on distribution $\mathcal{D}$ is defined as*

$$\text{Boundary}_{\mathcal{D},r}(f) = \Pr_{x \sim \mathcal{D}} [\exists z : \|z\|_2 \leq r \text{ and } f(x) \neq f(x+z)].$$

Definition 10 (Local noise sensitivity). *We define the local noise sensitivity $\phi_{f,\eta}(x)$ at noise scale η as*

$$\phi_{f,\eta}(x) := \mathbb{E}_{z \sim \mathcal{N}(0, I_d)} \left[\frac{1}{2} |f(x) - f(x + \eta z)| \right].$$

We remark that when f is Boolean-valued, this is equivalently

$$\phi_{f,\eta}(x) := \Pr_{z \sim \mathcal{N}(0, I_d)} [f(x) \neq f(x + \eta z)].$$

Definition 11 (Distributional noise sensitivity). *The noise sensitivity of a function f at noise scale η on distribution $\mathcal{D}$ is defined as*

$$\text{NS}_{\mathcal{D},\eta}(f) = \mathbb{E}_{x \sim \mathcal{D}} [\phi_{f,\eta}(x)].$$

Definition 12 (Isolation probability). *We call a point isolated if its local noise sensitivity is over a threshold. The isolation probability of a function f relative to a threshold t is*

$$\text{iso}_{\mathcal{D},\eta}(f, t) := \Pr_{x \sim \mathcal{D}} [\phi_{f,\eta}(x) > t].$$

We also use a convex relaxation of the isolation indicator and define ψ to be its expectation:

$$\psi_{\mathcal{D},\eta}(f) := 10 \mathbb{E}_{x \sim \mathcal{D}} [\mathbf{1}[\phi_{f,\eta}(x) > 0.6] \cdot (\phi_{f,\eta}(x) - 0.6)].$$

Note that $\psi(f)$ is an upper bound on $\text{iso}(f, 0.7)$.

Noise sensitivity approximators

Below, we introduce the notion of a local noise sensitivity approximator. Many of the algorithms we introduce will use such an approximator as a black box they can query (see [Algorithm 1](#), [Algorithm 5](#), [Algorithm 4](#) and [Algorithm 3](#)). We first introduce a local noise

sensitivity approximator and some related notions, and present a randomized method for efficiently instantiating it.¹

Definition 13 (Local noise sensitivity approximator for PTFs). *For a degree parameter k , an algorithm $\hat{\phi}$ is an ε -accurate noise sensitivity approximator if whenever it is given (i) a degree- k polynomial threshold function f , (ii) a noise rate $\eta \in (0, 1)$, and (iii) a point $x \in \mathbb{R}^d$, it outputs a value $\hat{\phi}_{f,\eta}(x)$ such that*

$$\left| \hat{\phi}_{f,\eta}(x) - \phi_{f,\eta}(x) \right| \leq \varepsilon.$$

Definition 14. *Let $\hat{\phi}$ be as in the definition above. Then we define the empirical noise sensitivity and isolation probabilities as*

$$\widehat{\text{NS}}_{\mathcal{D},\eta}(f) := \mathbb{E}_{x \sim \mathcal{D}} \hat{\phi}_{f,\eta}(x) \quad \text{and} \quad \widehat{\text{iso}}_{\mathcal{D},\eta}(f, t) := \mathbb{E}_{x \sim \mathcal{D}} \mathbb{1}[\hat{\phi}_{f,\eta}(x) > t].$$

We also define

$$\hat{\psi}_{\mathcal{D},\eta}(f) = 10 \mathbb{E}_{x \sim \mathcal{D}} [\mathbb{1}[\hat{\phi}_{f,T,\eta}(x) > 0.6] \cdot (\phi_{f,\eta}(x) - 0.6)].$$

3.1.2 Distances and Errors

Definition 15 (Error and optimal error). *We will denote the error of a hypothesis on a distribution*

$$\text{err}_{\mathcal{D}}(h) := \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\frac{1}{2} |h(x) - y| \right]$$

and remark that when h is Boolean-valued this is equivalently

$$\text{err}_{\mathcal{D}}(h) := \Pr_{(x,y) \sim \mathcal{D}} [h(x) \neq y].$$

Relative to a sample set we use

$$\widehat{\text{err}}_S(h) = \frac{1}{|S|} \sum_{(x,y) \in S} \frac{1}{2} |h(x) - y|.$$

We will often denote by **opt** the optimal error of a function f with respect to a concept class $\mathcal{F}$:

$$\text{opt} := \inf_{g \in \mathcal{F}} \text{err}_{\mathcal{D}}(g).$$

In this chapter, $\mathcal{F}$ is always the class of linear threshold functions over $\mathbb{R}^d$.

The following facts about the error and local noise sensitivity after rounding a real-valued function to Boolean follow directly from linearity of expectation.

¹In [Section 3.1.3](#) a randomized construction of $\hat{\phi}$ is given and in [Section 3.5](#) a deterministic method is presented assuming $\text{BPP} = \text{P}$.

Fact 4 (Randomized rounding). *Let f and g be real-valued functions. Let the rounded function f_t be defined as*

$$f_t(x) = \text{sign}(f(x) - t),$$

and likewise for g_t . Then,

$$\begin{aligned} \mathbb{E}_{t \sim [-1,1]} \left[\Pr_{(x,y) \sim \mathcal{D}} [f_t(x) \neq y] \right] &\leq \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\frac{|f(x) - y|}{2} \right]. \\ \mathbb{E}_{t \sim [-1,1]} \left[\Pr_{(x,y) \sim \mathcal{D}} [f_t(x) \neq g_t(x)] \right] &\leq \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\frac{|f(x) - g(x)|}{2} \right]. \end{aligned}$$

*Similarly,*²

$$\mathbb{E}_{t \sim [-1,1]} [\phi_{f_t, \eta}(x)] \leq \phi_{f, \eta}(x) \quad \text{and} \quad \mathbb{E}_{t \sim [-1,1]} [\tilde{\phi}_{f_t, T, \eta}(x)] \leq \tilde{\phi}_{f, T, \eta}(x)$$

for any x , η and T .

Finally, the following is a standard fact, see for example [45].

Fact 5 (TV distance between one-dimensional Gaussians). *For all μ_1, μ_2 in $\mathbb{R}$ and σ_1, σ_2 in $\mathbb{R}_{>0}$ we have*

$$d_{\text{TV}}(\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2)) \leq \frac{3|\sigma_1^2 - \sigma_2^2|}{2\sigma_1^2} + \frac{|\mu_1 - \mu_2|}{2\sigma_1}$$

3.1.3 Randomized Implementation of $\hat{\phi}$

Let T be a sufficiently large set of i.i.d. samples from $\mathcal{N}(0, I_d)$. Below, we show that the choice of $\hat{\phi}$ as the following empirical estimate will satisfy [Definition 13](#) with high probability over the choice of T :

$$\tilde{\phi}_{f, T, \eta}(x) = \frac{1}{|T|} \sum_{z \in T} \frac{|f(x) - f(x + \eta z)|}{2}.$$

Fact 6 ([6]). *The VC dimension of the class of polynomial threshold functions of degree k is $\Theta(\binom{d}{\leq k}) = O(d^k)$.*

Fact 7 (Concentration of expectations from VC dimension). *Let $\mathcal{C}$ be a concept class and $\mathcal{D}$ be a distribution over $\mathbb{R}^d$. For some sufficiently large absolute constant C , the following is true. With probability at least $1 - \delta$ over a sample S of i.i.d. samples from $\mathcal{D}$, with $|S| \geq C \cdot VC(\mathcal{C}) \log(1/\delta)/\varepsilon^2$, the following holds:*

$$\sup_{f \in \mathcal{C}} \left| \mathbb{E}_{x \sim \mathcal{D}} [f(x)] - \frac{1}{|T|} \sum_{x \in T} f(x) \right| \leq \varepsilon.$$

We will call a sample set “representative” of distribution $\mathcal{D}$ for a concept class if it satisfies the above success condition.

²See [Section 3.1.3](#) for the definition of $\tilde{\phi}$.

Claim 1 (Representative samples provide accurate noise sensitivity estimates). *Let $\mathcal{C}$ be the class of degree- k PTFs. Let T be a set of points in $\mathbb{R}^d$ that is representative for $\mathcal{N}(0, I_d)$ for $\mathcal{C}$ (in the sense of [Fact 7](#)). Then with probability $\geq 1 - \delta$ over T , the following holds for all $f \in \mathcal{C}$, $x \in \mathbb{R}^d$, and $\eta \in (0, 1)$:*

$$\left| \tilde{\phi}_{f,T,\eta}(x) - \phi_{f,\eta}(x) \right| \leq \varepsilon.$$

Proof. Observe that for any fixed x , the function $g(z) = -f(x) \cdot f(x + \eta z)$ is a member of $\mathcal{C}$. Thus by [Fact 7](#), we have with probability $1 - \delta$ that

$$\left| \mathbb{E}_{z \sim T}[g(z)] - \mathbb{E}_{z \sim \mathcal{N}(0, I_d)}[g(z)] \right| \leq \varepsilon.$$

The claim follows from observing that $\phi_{f,r}(x) = \frac{1}{2}(\mathbb{E}[g(z)] + 1)$, and thus

$$\left| \tilde{\phi}_{f,T,\eta}(x) - \phi_{f,\eta}(x) \right| = \frac{1}{2} \left| \mathbb{E}_{z \sim T}[g(z)] - \mathbb{E}_{z \sim \mathcal{N}(0, I_d)}[g(z)] \right| \leq \varepsilon.$$

□

3.1.4 Uniform Convergence of Empirical Approximations

Finally, we will need the following observation about the uniform convergence of approximations of local noise sensitivity ϕ . First, we need the following fact:

Fact 8 ([109], also see lecture notes [55]). *A function class $\mathcal{C}$ of VC dimension Δ and every distribution D_0 , there is an ε -cover H of $\mathcal{C}$ of size at most $\beta := (O(1)/\varepsilon)^{O(\Delta)}$. I.e. H is a discrete subset of $\mathcal{C}$ of size β and for every f in $\mathcal{C}$ we have h in H for which*

$$\Pr_{x \sim D_0}[f(x) \neq h(x)] \leq \varepsilon,$$

Claim 2. *Let $\mathcal{C}$ be the class of degree- k PTFs over $\mathbb{R}^d$, let D be a probability distribution over $\mathbb{R}^d$ and let $\eta \in [0, 1]$ be fixed. Then, for some constant C , if S is a collection of $(d^k/\varepsilon)^C \log 1/\delta$ i.i.d. examples from D , then with probability at least $1 - O(\delta)$,*

$$\max_{f \in \mathcal{C}} \left| \mathbb{E}_{x \sim S}[\phi_{f,\eta}(x)] - \mathbb{E}_{x \sim D}[\phi_{f,\eta}(x)] \right| \leq O(\varepsilon) \quad (3.1)$$

Proof. We now use [Fact 8](#). For us, $\mathcal{C}$ is the class of degree- k PTFs, and we have $\Delta = d^{O(k)}$. Taking the distribution D_0 to be an equal-weight mixture of (i) the distribution D in the premise of this claim and (ii) the convolution of D with the normal distribution $\mathcal{N}(0, \eta I_d)$ we see that for every f in $\mathcal{C}$ there is h in H for which³

$$\Pr_{x \sim D}[f(x) \neq h(x)] \leq 2\varepsilon, \quad (3.2)$$

³Note that the error over D_0 is the average of the two errors in the inequalities below.

$$\Pr_{\substack{x \sim D \\ z \sim \mathcal{N}(0, I_d)}} [f(x + \eta z) \neq h(x + \eta z)] \leq 2\varepsilon, \quad (3.3)$$

which via the definition of ϕ and the triangle inequality implies that

$$\begin{aligned} |\mathbb{E}_{x \sim D}[\phi_{f,\eta}(x)] - \mathbb{E}_{x \sim D}[\phi_{h,\eta}(x)]| &\leq \mathbb{E}_{x \sim D} \left[\left| \phi_{f,\eta}(x) - \phi_{h,\eta}(x) \right| \right] \\ &\leq 2 \mathbb{E}_{x \sim D}[|f(x) - h(x)|] + \mathbb{E}_{\substack{x \sim D \\ z \sim \mathcal{N}(0, I_d)}} [|f(x + \eta z) - h(x + \eta z)|] \leq O(\varepsilon). \end{aligned} \quad (3.4)$$

By the standard Hoeffding bound, since ϕ and ξ are always in $[0, 1]$, with probability $1 - \delta/2$ for every h in H we have

$$\left| \mathbb{E}_{x \sim S}[\phi_{h,\eta}(x)] - \mathbb{E}_{x \sim D}[\phi_{h,\eta}(x)] \right| \leq \frac{1}{\sqrt{|S|}} O(\log(|H|/\delta)) \leq \varepsilon, \quad (3.5)$$

where the last step for both inequalities follows by substituting $|H|$, $|S|$, Δ and taking C to be a sufficiently large absolute constant. We also know that with for a sufficiently large absolute constant C , with probability $1 - \delta$ the set S satisfies [Fact 10](#), which means that:

$$\sup_{f_1, f_2 \in \mathcal{C}} |\Pr_{x \sim S}[f_1(x) \neq f_2(x)] - \Pr_{x \sim D}[f_1(x) \neq f_2(x)]| \leq O(\varepsilon). \quad (3.6)$$

From the above, we can also conclude that with probability $1 - \delta$ over S for every pair f_1, f_2 in $\mathcal{C}$ we have the following:

$$\begin{aligned} &\left| \Pr_{\substack{x \sim S \\ z \sim \mathcal{N}(0, I_d)}} [f_1(x + \eta z) \neq f_2(x + \eta z)] - \Pr_{\substack{x \sim D \\ z \sim \mathcal{N}(0, I_d)}} [f_1(x + \eta z) \neq f_2(x + \eta z)] \right| \leq \\ \mathbb{E}_{z \sim \mathcal{N}(0, I_d)} &\underbrace{\left| \Pr_{x \sim S}[f_1(x + \eta z) \neq f_2(x + \eta z)] - \Pr_{x \sim D}[f_1(x + \eta z) \neq f_2(x + \eta z)] \right|}_{\substack{\leq O(\varepsilon) \text{ by defining } f_3(x) := f_1(x + \eta z), f_4(x) := f_2(x + \eta z), \\ \text{observing } f_3 \text{ and } f_4 \text{ are also degree-}k \text{ PTFs and using Equation (3.6)}}} \leq O(\varepsilon) \end{aligned} \quad (3.7)$$

Using the definition of ϕ , the triangle inequality, and Equations [3.6](#) and [3.7](#) we also see that

$$\begin{aligned} |\mathbb{E}_{x \sim S}[\phi_{h,\eta}(x)] - \mathbb{E}_{x \sim S}[\phi_{f,\eta}(x)]| &\leq \\ 2 \Pr_{x \sim S}[h(x) \neq f(x)] + \Pr_{\substack{x \sim S \\ z \sim \mathcal{N}(0, I_d)}} [h(x + \eta z) \neq f(x + \eta z)] &\leq \\ 2 \Pr_{x \sim D}[h(x) \neq f(x)] + \Pr_{\substack{x \sim D \\ z \sim \mathcal{N}(0, I_d)}} [h(x + \eta z) \neq f(x + \eta z)] &\leq O(\varepsilon), \end{aligned} \quad (3.8)$$

where in the last step we substituted [Equation \(3.2\)](#) and [Equation \(3.3\)](#). Finally, we put all the inequalities together. By combining the triangle inequality with Equations [3.4](#), [3.5](#) and [3.8](#) we derive [Equation \(3.1\)](#). $\square$

Claim 3. Let $\mathcal{C}$ be the class of degree- k PTFs over $\mathbb{R}^d$, let D be a probability distribution over $\mathbb{R}^d$ and let $\eta \in [0, 1]$ be fixed. Then, for some constant C , if S is a collection of $(d^k/\varepsilon)^C \log 1/\delta$ i.i.d. examples from D , then with probability at least $1 - O(\delta)$, for every polynomial p it holds that

$$\Pr_{x \sim D} [\mathbb{E}_{t \in [-1, 1]} [\phi_{\text{sign}(p(x)-t), \eta}(x)] \geq 0.7] \leq \Pr_{x \sim D} [\mathbb{E}_{t \in [-1, 1]} [\phi_{\text{sign}(p-t), \eta}(x)] \geq 0.67] + O(\varepsilon) \quad (3.9)$$

Proof. Let $\mathcal{C}^{\text{clipped}}$ be the class of degree- k clipped polynomials, i.e. functions $p^{\text{clipped}} : \mathbb{R}^d \rightarrow [-1, 1]$ of the form

$$p^{\text{clipped}}(x) = \begin{cases} 1 & \text{if } p(x) \geq 1 \\ -1 & \text{if } p(x) \leq -1 \\ p(x) & \text{otherwise} \end{cases}$$

where p is a degree- k polynomial. We first argue the following

Observation 9. For any distribution D_0 over $\mathbb{R}^d$, there is a subset H of $\mathcal{C}^{\text{clipped}}$ of size $(O(1)/\varepsilon)^{O(d^k/\varepsilon)}$, i.e. such that for every p^{clipped} in $\mathcal{C}^{\text{clipped}}$ we have

$$\min_{h \in H} [\mathbb{E}_{x \sim D_0} [|p^{\text{clipped}}(x) - h(x)|]] \leq O(\varepsilon). \quad (3.10)$$

Proof. We form H by considering an ε^2 -net $H_{\text{PTF}}^{\varepsilon^2}$ for degree- $2k$ PTFs, and taking H to consist of functions of the form

$$h(x) = \sum_{\tau \in \{-1, -1+\varepsilon, -1+2\varepsilon, \dots, +1\}} \tau \cdot f_{\tau}(x),$$

where each $f_{\tau}(x)$ is some function in $\mathcal{F}_{\text{PTF}}$. By [Fact 8](#), we see that $H_{\text{PTF}}^{\varepsilon^2}$ has size $(O(1)/\varepsilon)^{O(d^k)}$, and therefore (by a counting argument) the size of H defined above is indeed $(O(1)/\varepsilon)^{O(d^k/\varepsilon)}$.

We can write for every x and p^{clipped} in $\mathcal{C}^{\text{clipped}}$ the following inequality:

$$\begin{aligned} \sum_{\tau \in \{-1, -1+\varepsilon, -1+2\varepsilon, \dots, +1\}} \tau \cdot \mathbb{1}[p^{\text{clipped}}(x) \in (\tau, \tau + \varepsilon)] &\leq p^{\text{clipped}}(x) \leq \\ &\sum_{\tau \in \{-1, -1+\varepsilon, -1+2\varepsilon, \dots, +1\}} (\tau + \varepsilon) \cdot \mathbb{1}[p^{\text{clipped}}(x) \in (\tau, \tau + \varepsilon)], \end{aligned} \quad (3.11)$$

we observe that each indicator $\mathbb{1}[p^{\text{clipped}}(x) \in (\tau, \tau + \varepsilon)]$ equals to a degree- $2k$ polynomial threshold function, and therefore there is some f_{τ} in $H_{\text{PTF}}^{\varepsilon^2}$ for which

$$\mathbb{E}_{x \sim D_0} |\mathbb{1}[p^{\text{clipped}}(x) \in (\tau, \tau + \varepsilon)] - f_{\tau}| \leq O(\varepsilon^2),$$

which substituted into [Equation \(3.11\)](#) allows us to conclude that

$$\mathbb{E}_{x \sim D_0} \left| p^{\text{clipped}}(x) - \sum_{\tau \in \{-1, -1+\varepsilon, -1+2\varepsilon, \dots, +1\}} \tau \cdot f_{\tau}(x) \right| \leq O(\varepsilon),$$

which concludes the proof that for p^{clipped} in $\mathcal{C}^{\text{clipped}}$ we have $h : \mathbb{R}^d$ in H for which Equation (3.10) holds.

We finish the proof of the observation, by resolving one last issue: as defined the set H is not a subset of $\mathcal{C}^{\text{clipped}}$. However, if we consider the set H' consisting of functions f of the form

$$H' = \{f = \underset{g \in \mathcal{C}^{\text{clipped}}}{\operatorname{argmin}} [\mathbb{E}_{x \sim D_0} [|g(x) - h(x)|]] : h \in H, \}$$

then we see by the triangle inequality that H' still satisfies Equation (3.10), has a size of at most $|H|$ and H' is a subset of $\mathcal{C}^{\text{clipped}}$ $\square$

We continue the proof of Claim 3 We can write:

$$\begin{aligned} \mathbb{E}_{t \in [-1,1]} [\phi_{\operatorname{sign}(p(x)-t), \eta}(x)] &= \Pr_{t \in [-1,1], z \sim \mathcal{N}(0, I_d)} [|\operatorname{sign}(p(x) - t) \neq \operatorname{sign}(p(x + \eta z) - t)|] = \\ &\mathbb{E}_{z \sim \mathcal{N}(0, I_d)} \left[\frac{|p^{\text{clipped}}(x) - p^{\text{clipped}}(x + \eta z)|}{2} \right] := \phi_{p^{\text{clipped}}, \eta}(x) \end{aligned} \quad (3.12)$$

We will overload the notation for ϕ and define $\phi_{p^{\text{clipped}}, \eta}(x)$ to be the expression above. We further define the following auxiliary quantity:

$$\xi_{p^{\text{clipped}}, \eta}(x) = \begin{cases} 0 & \text{if } \phi_{p^{\text{clipped}}, \eta}(x) \leq 0.67 \\ 1 & \text{if } \phi_{p^{\text{clipped}}, \eta}(x) \geq 0.7 \\ \frac{100}{3}(\phi_{p^{\text{clipped}}, \eta}(x) - 0.67) & \text{if } \phi_{p^{\text{clipped}}, \eta}(x) \in (0.67, 0.7) \end{cases}$$

By construction, the function satisfies the following properties for every x ,

$$\mathbb{1}[\phi_{p^{\text{clipped}}, \eta}(x) \geq 0.67] \leq \xi_{p^{\text{clipped}}, \eta}(x) \leq \mathbb{1}[\phi_{p^{\text{clipped}}, \eta}(x) \geq 0.7] \quad (3.13)$$

$$\left| \xi_{p_1^{\text{clipped}}, \eta}(x) - \xi_{p_2^{\text{clipped}}, \eta}(x) \right| \leq O(1) \cdot \left| \phi_{p_1^{\text{clipped}}, \eta}(x) - \phi_{p_2^{\text{clipped}}, \eta}(x) \right| \quad (3.14)$$

Taking the distribution D_0 to be an equal-weight mixture of (i) the distribution D in the premise of this claim and (ii) the convolution of D with the normal distribution $\mathcal{N}(0, \eta I_d)$ we see that for every p^{clipped} in $\mathcal{C}^{\text{clipped}}$ there is h in H for which⁴

$$\Pr_{x \sim D} [|p^{\text{clipped}}(x) - h(x)|] \leq 2\varepsilon, \quad (3.15)$$

$$\Pr_{\substack{x \sim D \\ z \sim \mathcal{N}(0, I_d)}} [|p^{\text{clipped}}(x + \eta z) - h(x + \eta z)|] \leq 2\varepsilon, \quad (3.16)$$

which via the definition of ϕ for clipped polynomials and the triangle inequality implies that

$$\begin{aligned} \left| \mathbb{E}_{x \sim D} [\phi_{p^{\text{clipped}}, \eta}(x)] - \mathbb{E}_{x \sim D} [\phi_{h, \eta}(x)] \right| &\leq \mathbb{E}_{x \sim D} \left[\left| \phi_{p^{\text{clipped}}, \eta}(x) - \phi_{h, \eta}(x) \right| \right] \\ &\leq 2 \mathbb{E}_{x \sim D} [|p^{\text{clipped}}(x) - h(x)|] + \mathbb{E}_{\substack{x \sim D \\ z \sim \mathcal{N}(0, I_d)}} [|p^{\text{clipped}}(x + \eta z) - h(x + \eta z)|] \leq O(\varepsilon). \end{aligned} \quad (3.17)$$

⁴Note that the error over D_0 is the average of the two errors in the inequalities below.

Together with Equation (3.14), this implies that

$$\left| \mathbb{E}_{x \sim D}[\xi_{p^{\text{clipped}}, \eta}(x)] - \mathbb{E}_{x \sim D}[\xi_{h, \eta}(x)] \right| \leq O(1) \cdot \mathbb{E}_{x \sim D} \left[\left| \phi_{p^{\text{clipped}}, \eta}(x) - \phi_{h, \eta}(x) \right| \right] \leq O(\varepsilon). \quad (3.18)$$

By the standard Hoeffding bound, since ϕ and ξ are always in $[0, 1]$, with probability $1 - \delta/2$ for every h in H we have

$$\left| \mathbb{E}_{x \sim S}[\phi_{h, \eta}(x)] - \mathbb{E}_{x \sim D}[\phi_{h, \eta}(x)] \right| \leq \frac{1}{\sqrt{|S|}} O(\log(|H|/\delta)) \leq \varepsilon, \quad (3.19)$$

$$\left| \mathbb{E}_{x \sim S}[\xi_{h, \eta}(x)] - \mathbb{E}_{x \sim D}[\xi_{h, \eta}(x)] \right| \leq \frac{1}{\sqrt{|S|}} O(\log(|H|/\delta)) \leq \varepsilon, \quad (3.20)$$

where the last step for both inequalities follows by substituting $|H|$, $|S|$, Δ and taking C to be a sufficiently large absolute constant.

We observe that Fact 10 and Equation (3.11) together imply that⁵:

$$\sup_{p_1^{\text{clipped}}, p_2^{\text{clipped}} \in \mathcal{C}^{\text{clipped}}} \left| \mathbb{E}_{x \sim S} \left[\left| p_1^{\text{clipped}}(x) - p_2^{\text{clipped}}(x) \right| \right] - \mathbb{E}_{x \sim D} \left[\left| p_1^{\text{clipped}}(x) - p_2^{\text{clipped}}(x) \right| \right] \right| \leq O(\varepsilon). \quad (3.21)$$

From the above, we can also conclude that with probability $1 - \delta$ over S for every pair $p_1^{\text{clipped}}, p_2^{\text{clipped}}$ in $\mathcal{C}^{\text{clipped}}$ we have the following:

$$\begin{aligned} & \left| \mathbb{E}_{\substack{x \sim S \\ z \sim \mathcal{N}(0, I_d)}} \left[\left| p_1^{\text{clipped}}(x + \eta z) - p_2^{\text{clipped}}(x + \eta z) \right| \right] - \mathbb{E}_{\substack{x \sim D \\ z \sim \mathcal{N}(0, I_d)}} \left[\left| p_1^{\text{clipped}}(x + \eta z) - p_2^{\text{clipped}}(x + \eta z) \right| \right] \right| \leq \\ & \mathbb{E}_{z \sim \mathcal{N}(0, I_d)} \left[\underbrace{\left| \mathbb{E}_{x \sim S} \left[\left| p_1^{\text{clipped}}(x + \eta z) - p_2^{\text{clipped}}(x + \eta z) \right| \right] - \mathbb{E}_{x \sim D} \left[\left| p_1^{\text{clipped}}(x + \eta z) - p_2^{\text{clipped}}(x + \eta z) \right| \right] \right|}_{\leq O(\varepsilon) \text{ by defining } p_3^{\text{clipped}}(x) := p_1^{\text{clipped}}(x + \eta z), p_4^{\text{clipped}}(x) := p_2^{\text{clipped}}(x + \eta z), \text{ and using Equation (3.21)}} \right] \leq O(\varepsilon) \end{aligned} \quad (3.22)$$

We now come back to the setting of Equation (3.15). p^{clipped} is a function in $\mathcal{C}^{\text{clipped}}$ and h in H satisfies Equation (3.15). Using the definition of ϕ , the triangle inequality, and Equations 3.21 and 3.22 we also see that

$$\begin{aligned} & \left| \mathbb{E}_{x \sim S}[\phi_{h, \eta}(x)] - \mathbb{E}_{x \sim S}[\phi_{p^{\text{clipped}}, \eta}(x)] \right| \leq \\ & 2 \Pr_{x \sim S} [|h(x) - p^{\text{clipped}}(x)|] + \Pr_{\substack{x \sim S \\ z \sim \mathcal{N}(0, I_d)}} [|h(x + \eta z) - p^{\text{clipped}}(x + \eta z)|] \leq \\ & 2 \Pr_{x \sim D} [h(x) \neq f(x)] + \Pr_{\substack{x \sim D \\ z \sim \mathcal{N}(0, I_d)}} [h(x + \eta z) \neq f(x + \eta z)] \leq O(\varepsilon), \end{aligned} \quad (3.23)$$

⁵Equation (3.11) tells us that the difference $p_1 - p_2$ can be ε -approximated in L_∞ norm by a function of the form $\sum_{\tau \in \{-2, -2+\varepsilon, -1+2\varepsilon, \dots, +2\}} \tau \cdot f_\tau(x)$, where each f_τ is a degree- $2k$ PTF. Then, Fact 10 together with triangle inequality tells us that the inequality above holds when C is a sufficiently large absolute constant.

where in the last step we substituted Equation (3.15) and Equation (3.16). By Equation 3.14, we see that

$$|\mathbb{E}_{x \sim S}[\xi_{h,\eta}(x)] - \mathbb{E}_{x \sim S}[\xi_{f,\eta}(x)]| \leq O(1) \cdot |\mathbb{E}_{x \sim S}[\phi_{h,\eta}(x)] - \mathbb{E}_{x \sim S}[\phi_{f,\eta}(x)]| \leq O(\varepsilon), \quad (3.24)$$

Analogously, by combining the triangle inequality with Equations 3.18, 3.20 and 3.24 we derive the following:

$$\max_{f \in \mathcal{C}} \left| \mathbb{E}_{x \sim S}[\xi_{f,\eta}(x)] - \mathbb{E}_{x \sim D}[\xi_{f,\eta}(x)] \right| \leq O(\varepsilon), \quad (3.25)$$

which together with Equation (3.13) and Equation (3.12) implies Equation (3.9). $\square$

The following claims follow from VC theory (see e.g. [110] and the references therein):

Fact 9 (Generalization bound from VC dimension). *Let $\mathcal{C}$ be a concept class and $\mathcal{D}$ be a distribution over $\mathbb{R}^d$. For some sufficiently large absolute constant C , the following is true. With probability at least $1 - \delta$ over a sample S of i.i.d. samples from $\mathcal{D}$, with $|S| \geq C \cdot VC(\mathcal{C}) \log(1/\delta)/\varepsilon^2$ the following holds:*

$$\sup_{f \in \mathcal{C}} |\text{err}_{\mathcal{D}}(f) - \widehat{\text{err}}_S(f)| \leq O(\varepsilon).$$

Fact 10. *Let $\mathcal{C}$ be a concept class with VC dimension Δ and D be a distribution over $\mathbb{R}^d$. For a sufficiently large absolute constant C , let S be a collection of $\frac{C\Delta}{\varepsilon^2} \log(1/\delta)$ i.i.d. examples from D . Then, the following holds with probability at least $1 - \delta$*

$$\sup_{f_1, f_2 \in \mathcal{C}} |\Pr_{x \sim S}[f_1(x) \neq f_2(x)] - \Pr_{x \sim D}[f_1(x) \neq f_2(x)]| \leq O(\varepsilon). \quad (3.26)$$

In Section 3.1.4, miscellaneous claims about uniform convergence are shown, which are used in the analysis of Algorithm 2.

We will use the notation of the form $a = b \pm c$ as a shorthand for $|a - b| \leq c$. We will also drop subscripts, particularly on ϕ and $\hat{\phi}$, when they are clear from context. Throughout this work, wherever there is noise, the noise scale η is always $10r$, where r is the desired robustness radius. We use the notation $t \sim [-1, 1]$ to denote that t is drawn uniformly from $[-1, 1]$.

Sometimes we will write $f(x) = \text{Ber}(p)$, to denote that $f(x) = 1$ with probability p and 0 with probability $1 - p$.

3.2 Results and Pseudocode of Our Main Algorithm

Our main result is the following. It states the correctness and running time of the main algorithm, ROBUSTLEARN.

Theorem 10 (Correctness and complexity of Algorithm 1). Let $\varepsilon \geq d^{-1/7}$, let $r, \delta \in (0, 1)$, and let C be a sufficiently large absolute constant. Furthermore, let $\hat{\phi}$ be an ε -accurate local noise sensitivity approximator for degree- k PTFs over $\mathbb{R}^d$, where $k := \frac{C \log^2(1/\varepsilon)}{\varepsilon^2}$, and let the running time and query complexity of $\hat{\phi}$ be τ .

Let $\mathcal{F}$ be the class of linear threshold functions over $\mathbb{R}^d$ and $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{-1, 1\}$ whose $\mathbb{R}^d$ -marginal is isotropic, subgaussian and log-concave. The algorithm ROBUSTLEARN, given i.i.d. sample access to the distribution $\mathcal{D}$, with probability at least $1 - O(\delta)$ returns a hypothesis $h : \mathbb{R}^d \rightarrow \{\pm 1\}$ for which the following hold:

$$\text{err}_{\mathcal{D}}(h) \leq \text{opt} + O(\varepsilon)$$

$$\text{Boundary}_{\mathcal{D}, r}(h) \leq O(r + \varepsilon).$$

The running time and sample complexity are poly $\left(d^{\log^2(1/\varepsilon)/\varepsilon^2} \cdot \log(1/\delta) \cdot \tau\right)$.

Corollary 4. By instantiating $\hat{\phi}$ with the algorithm of Section 3.1.3, which evaluates queries in poly $\left(d^{\log^2(1/\varepsilon)/\varepsilon^2} \cdot \log(1/\delta)\right)$ time, ROBUSTLEARN runs in total time poly $\left(d^{\log^2(1/\varepsilon)/\varepsilon^2} \cdot \log(1/\delta)\right)$.

Algorithm 1 ROBUSTLEARN($\mathcal{D}, \varepsilon, \delta, r$):

- 1: **Input:** sample access to distribution $\mathcal{D}$ over $\mathbb{R}^d \times \{\pm 1\}$, error bound ε ,
 - 2: confidence bound δ , robustness radius r
 - 3: **Uses:** local noise sensitivity approximator $\hat{\phi}$. (See Definition 13)
 - 4: **Output:** hypothesis $h : \mathbb{R}^d \rightarrow \{-1, 1\}$
 - 5: $p := \text{LEARNREALVALUED}(\mathcal{D}, \varepsilon, r)$. (See Algorithm 2)
 - 6: **return** COMPUTECLASSIFIER($\mathcal{D}, p, r, \varepsilon, \delta$). (See Algorithm 5)
-

The algorithm has two “phases”: LEARNREALVALUED, which solves a convex relaxation of the learning task, and COMPUTECLASSIFIER, which rounds the real-valued hypothesis to a Boolean one with small boundary volume. We present and analyze these algorithms in Section 3.3.3 and Section 3.4.3 respectively.

Theorem 11 (Correctness of Algorithm 2, LEARNREALVALUED). Let $r \in (0, 1)$, and let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{\pm 1\}$ whose $\mathbb{R}^d$ -marginal is a subgaussian isotropic log-concave distribution over $\mathbb{R}^d$. Let $\mathcal{F}$ be the class of halfspaces on $\mathbb{R}^d$. Then, for some sufficiently large absolute constant C , the algorithm LearnRealValued, given ε, δ and sample access to $\mathcal{D}$, runs in time poly $\left(d^{O(\log^2(1/\varepsilon)/\varepsilon^2)}\right)$ and will with probability at least $1 - O(\delta)$ return a degree- $O(\log^2(1/\varepsilon)/\varepsilon^2)$ polynomial p for which the following hold:

$$\mathbb{E}_{t \sim [-1, 1]} \left[\mathbb{E}_{(x, y) \sim \mathcal{D}} [\phi_{\text{sign}(p-t), 10r}(x)] \right] \leq 100r + O(\varepsilon), \quad (3.27)$$

$$\Pr_{(x, y) \sim \mathcal{D}} \left[\mathbb{E}_{t \sim [-1, 1]} [\phi_{\text{sign}(p(x)-t), 10r}(x)] \geq 0.7 \right] \leq O(\varepsilon), \quad (3.28)$$

$$\mathbb{E}_{t \sim [-1,1]} \left[\Pr_{(x,y) \sim D} [\text{sign}(p(x) - t) \neq y] \right] \leq \text{opt} + O(\varepsilon). \quad (3.29)$$

Theorem 12 (Correctness of [Algorithm 5](#), COMPUTECLASSIFIER). *Let $\varepsilon \geq d^{-1/7}$ and $\delta, r > 0$. Let $\hat{\phi}$ be an ε -accurate local noise sensitivity approximator for degree- k PTFs over $\mathbb{R}^d$. Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{-1, 1\}$ such that the $\mathbb{R}^d$ -marginal is isotropic and log-concave. Let $p : \mathbb{R}^d \rightarrow \mathbb{R}$ be a polynomial of degree at most k satisfying*

$$\mathbb{E}_{t \sim [-1,1]} \left[\mathbb{E}_{(x,y) \sim \mathcal{D}} [\phi_{\text{sign}(p-t), 10r}(x)] \right] \leq 100r + O(\varepsilon), \quad (3.30)$$

$$\Pr_{(x,y) \sim \mathcal{D}} \left[\mathbb{E}_{t \sim [-1,1]} [\phi_{\text{sign}(p(x)-t), 10r}(x)] \geq 0.7 \right] \leq O(\varepsilon). \quad (3.31)$$

Then the algorithm COMPUTECLASSIFIER($\mathcal{D}, p, r, \varepsilon, \delta$), given sample access to $\mathcal{D}$ and query access to $\hat{\phi}$, outputs a hypothesis $h : \mathbb{R}^d \rightarrow \{-1, 1\}$ such that the following properties hold with probability at least $1 - O(\delta)$:

$$\text{err}_{\mathcal{D}}(h) \leq \mathbb{E}_{t \sim [-1,1]} [\text{err}_{\mathcal{D}}(\text{sign}(p - t))] + O(\varepsilon)$$

$$\text{Boundary}_{\mathcal{D}, r}(h) \leq O(r + \varepsilon).$$

The running time and number of queries to $\hat{\phi}$ are $\text{poly}(d^k \cdot 1/\varepsilon \cdot \log(1/\delta))$.

3.2.1 Proof of [Theorem 10](#)

In this section we prove the main theorem assuming [Theorem 12](#) and [Theorem 11](#).

Proof of [Theorem 10](#). By [Theorem 11](#), with probability $1 - O(\delta)$, the output p of LEARNREALVALUED is a degree- k polynomial satisfying

$$\mathbb{E}_{t \sim [-1,1]} \left[\mathbb{E}_{(x,y) \sim D} [\phi_{\text{sign}(p-t), 10r}(x)] \right] \leq 100r + O(\varepsilon), \quad (3.32)$$

$$\Pr_{(x,y) \sim \mathcal{D}} \left[\mathbb{E}_{t \sim [-1,1]} [\phi_{\text{sign}(p(x)-t), 10r}(x)] \geq 0.7 \right] \leq O(\varepsilon), \quad (3.33)$$

$$\mathbb{E}_{t \sim [-1,1]} [\Pr_{(x,y) \sim D} [\text{sign}(p(x) - t) \neq y]] \leq \text{opt} + O(\varepsilon). \quad (3.34)$$

This polynomial p is the input to COMPUTECLASSIFIER. We see that the premises of [Theorem 12](#) are satisfied by the guarantees of [Equation \(3.32\)](#) and [Equation \(3.33\)](#). Taking the conclusions of [Theorem 12](#) and combining it with [Equation \(3.34\)](#) we see that the hypothesis h we return satisfies

$$\text{err}_{\mathcal{D}}(h) \leq \mathbb{E}_{t \sim [-1,1]} [\text{err}_{\mathcal{D}}(\text{sign}(p - t))] + O(\varepsilon) \leq \text{opt} + O(\varepsilon),$$

$$\text{Boundary}_{\mathcal{D},r}(h) \leq O(r + \varepsilon),$$

which concludes the proof of correctness.

We now analyze the sample complexity and running time. By [Theorem 11](#), `LEARNREALVALUED` takes $d^{O(\log^2(1/\varepsilon)/\varepsilon^2)} \text{poly}(\log(1/\delta))$ time and samples. By [Theorem 12](#), since the degree of p is $O(\log^2(1/\varepsilon)/\varepsilon^2)$ and $\varepsilon \geq d^{-1/7}$, `COMPUTECCLASSIFIER` takes $d^{O(\log^2(1/\varepsilon)/\varepsilon^2)} \text{poly}(\log(1/\delta))$ time and queries to $\hat{\phi}$. Evaluations of $\hat{\phi}$ take time τ , so the total running time is

$$\text{poly}(d^{\log^2(1/\varepsilon)/\varepsilon^2} \cdot \log(1/\delta) \cdot \tau),$$

and all components succeed with probability $\geq 1 - O(\delta)$. □

3.3 Pseudocode and Analysis of LearnRealValued

In this section we present and analyze our algorithm for learning a polynomial with small noise sensitivity and isolation probability.

Algorithm 2 `LEARNREALVALUED`(D, ε, r):

- 1: **Input:** Sample access to distribution D , error bound ε , robustness radius r
 - 2: **Output:** hypothesis $h : \{-1, 1\}^d \rightarrow \mathbb{R}$
 - 3: Set $k := \frac{C \log^2(1/\varepsilon)}{\varepsilon^2}$.
 - 4: **for** i in $\{1, \dots, \log(1/\delta)\}$ **do**
 - 5: $S_i := \{C d^{Ck} \cdot \log(1/\delta)/\varepsilon^2 \text{ i.i.d. draws from } D\}$.
 - 6: $T_i := \{C d^{Ck} \cdot \log(1/\delta)/\varepsilon^2 \text{ i.i.d. draws from } \mathcal{N}(0, I_d)\}$
 - 7: Let p_i be the output of the following convex program:
 - Domain: polynomials over $\mathbb{R}^d$ of degree k .
 - Minimize: $\sum_{(x,y) \in S} |p(x) - y|$
 - Constraints:
 - Define $\tilde{\phi}(x)_{p,T,10r} := \frac{1}{|T|} \sum_{z \in T} \frac{|f(x) - f(x+\eta z)|}{2}$
 - Define $\tilde{\psi}_{p,T,10r} := 10 \cdot \mathbb{1}[\tilde{\phi}(x)_{p,T,10r} > 0.6] \cdot (\tilde{\phi}(x)_{p,T,10r} - 0.6)$
 - Constraint 1: $\frac{1}{|S|} \sum_{x \in S} \tilde{\phi}_{p,T,10r}(x) \leq 100r + \varepsilon$
 - Constraint 2: $\frac{1}{|S|} \sum_{x \in S} \tilde{\psi}_{p,T,10r}(x) \leq \varepsilon$
 - 8: $i^* := \text{argmin}_i \left(\mathbb{E}_{(x,y) \sim S_i} \left[\frac{|p_i(x) - y|}{2} \right] \right)$
 - 9: $p := p_{i^*}$.
 - 10: **return** p
-

3.3.1 Facts About Log-Concave Distributions

Definition 16. A distribution D over $\mathbb{R}^d$ is called C -sub-Gaussian, if for any unit vector u :

$$\Pr_{x \sim D}[|u \cdot x| > t] \leq \exp(-Ct^2)$$

for any $t \geq 0$. We say D is sub-Gaussian, if D is C -sub-Gaussian for some absolute constant C .

See e.g. [105] for the following fact:

Fact 11. If D_1 and D_2 are log-concave distributions over $\mathbb{R}^d$, then the convolution of D_1 and D_2 is also a log-concave distribution over $\mathbb{R}^d$.

Likewise, the following fact follows directly from the definition of a sub-Gaussian distribution:

Fact 12. If D_1 and D_2 are sub-Gaussian distributions over $\mathbb{R}^d$, then the convolution of D_1 and D_2 is also a sub-Gaussian distribution over $\mathbb{R}^d$.

The following fact can be found in [105]:

Fact 13. If a distribution D over $\mathbb{R}^d$ is isotropic log-concave, then the projection of D on any linear subspace in $\mathbb{R}^d$ is likewise log-concave.

The following two facts can be found in [68], [8] and the references therein.

Fact 14. If a distribution D over $\mathbb{R}^d$ is isotropic log-concave, then for some absolute constant C_3 , for any unit vector u and every interval $J \subset \mathbb{R}$, we have $\Pr_{x \sim D}[\langle u, x \rangle \in J] \leq |J|$.

Remark 13. It follows from a direct change-of-basis argument that Fact 14 still holds if the premise that the distribution D is isotropic⁶ is replaced by a slightly more general premise that D mean zero and satisfies:

$$I_{d \times d} \preceq \mathbb{E}_{x \sim D}[xx^T] \preceq 200I_{d \times d}$$

For the following fact, see e.g. [50] and the references therein:

Fact 15. If a distribution D over $\mathbb{R}^d$ is log-concave, then for some absolute constant C_4 , if p is a degree- k polynomial for which $\mathbb{E}_{x \sim D}[(p(x))^2] \leq 1$, then for any $B > e^k$ it is the case that

$$\Pr_{x \sim D}[(p(x))^2 > B] \leq \exp(-C_4 B^{1/k})$$

One can use the fact above to conclude the following:

⁶Recall that the condition that D is isotropic means that D has zero mean and covariance $I_{d \times d}$.

Observation 14. If a distribution D over $\mathbb{R}^d$ is log-concave, then for some absolute constant C_5 , for every degree- k polynomial p it is the case that

$$\mathbb{E}_{x \sim D}[|p(x)|] \geq \frac{\sqrt{\mathbb{E}_{x \sim D}[(p(x))^2]}}{(C_5 k)^{C_5 k}}$$

Proof. Without loss of generality, we can assume that $\mathbb{E}_{x \sim D}[(p(x))^2] = 1$. For any $B \geq 0$, we have

$$\mathbb{E}_{x \sim D}[|p(x)|] \geq \frac{\mathbb{E}_{x \sim D}[|p(x)|^2 \mathbb{1}_{|p(x)| \leq B}]}{B} = \frac{1 - \mathbb{E}_{x \sim D}[|p(x)|^2 \mathbb{1}_{|p(x)| > B}]}{B} \quad (3.35)$$

Using [Fact 15](#), we have

$$\begin{aligned} \mathbb{E}_{x \sim D}[|p(x)|^3] &\leq e^{O(k)} + \int_{\beta=0}^{+\infty} \Pr_{x \sim D}[|p(x)|^3 > \beta] d\beta \leq \\ &e^{O(k)} + \int_{\beta'=0}^{+\infty} (\beta')^{3k/2-1} \exp(-C_4 \beta') d\beta' \leq (O(k))^{O(k)}, \end{aligned}$$

which allows us to conclude that

$$\mathbb{E}_{x \sim D}[|p(x)|^2 \mathbb{1}_{|p(x)| > B}] \leq \frac{\mathbb{E}_{x \sim D}[|p(x)|^3]}{B} = \frac{(O(k))^{O(k)}}{B}.$$

Taking B to equal $(C'k)^{C'k}$ for a sufficiently large absolute constant C' , and substituting the bound above into [Equation \(3.35\)](#) finishes the proof. $\square$

3.3.2 Approximating Halfspaces with Polynomials

Claim 4. Let h be a linear threshold function over $\mathbb{R}^d$. Then, for every absolute constant C , there exists a polynomial p of degree $O(1/\varepsilon^2 \log^2(1/\varepsilon))$, such that for any C -sub-Gaussian log-concave distribution D over $\mathbb{R}^d$ satisfying $I_{d \times d} \preceq \mathbb{E}_{x \sim D}[xx^T] \preceq 200I_{d \times d}$ we have

$$\mathbb{E}_{x \sim D}[|h(x) - p(x)|] \leq O(\varepsilon)$$

Remark 15. Recall that a distribution D is log-concave if the logarithm of the probability density function of D is concave. Note that the constants hidden by the $O(\cdot)$ notation in [Claim 4](#) are allowed to depend on the sub-Gaussianity constant C .

To prove the claim above, we will need the following proposition from [\[46\]](#):

Fact 16 ([\[46\]](#)). There exist absolute constants C_4 and C_5 , such that for every $\varepsilon \in (0, 0.1)$, there exists a univariate polynomial P satisfying the following. Denoting $a := \frac{\varepsilon^2}{C_4 \log(1/\varepsilon)}$ and $K := C_5 \frac{\log \frac{1}{\varepsilon}}{a} = O\left(\frac{\log^2 1/\varepsilon}{\varepsilon^2}\right)$, it is the case that

- The degree of P is at most K .

- $P(t) \in [\text{sign}(t), \text{sign}(t) + \varepsilon]$ for all t in $[-1/2, -2a] \cup [0, 1/2]$.
- $P(t) \in [-1, 1 + \varepsilon]$ for t in $[-2a, 0]$
- $|P(t)| \leq 2 \cdot (4t)^K$ for all $|t| \geq 1/2$.

The proof below follows closely the line of reasoning in [46] and is included for completeness.

Proof of Claim 4. For a unit vector u $h(x) = \text{sign}(u \cdot x - \tau)$ (assuming $\tau = 0$ for now)

If $|\tau| > \log(1/\varepsilon)$ the statement follows immediately by taking P to be the constant polynomial that equals to $\text{sign}(\tau)$ everywhere in $\mathbb{R}^d$. The error between h and P will be upper-bounded by $O(\varepsilon)$ via Definition 16 and Remark 13.

Otherwise, we have $|\tau| \leq \log(1/\varepsilon)$. Let P be the polynomial in Fact 16 and take $w := C_6 \left(\frac{\log^2 1/\varepsilon}{\varepsilon^2} \right)^{1/2}$ for a sufficiently large absolute constant C_6 . Denoting by $D_{\text{projected}}$ the projection of distribution D on the unit vector u , we can bound the error as follows using Fact 16:

$$\begin{aligned} \mathbb{E}_{x \sim D} [|\text{sign}(u \cdot x - \tau) - P((u \cdot x - \tau)/w)|] &= \mathbb{E}_{z \sim D_{\text{projected}}} [|\text{sign}(z - \tau) - P((z - \tau)/w)|] \leq \\ \varepsilon + 2 \Pr_{z \sim D_{\text{projected}}} [-2a \leq z/w - \tau \leq 0] &+ \mathbb{E}_{z \sim D_{\text{projected}}} \left[\left(2 \cdot \left(4 \cdot \frac{z - \tau}{w} \right)^K + 1 \right) \mathbb{1}_{|(z - \tau)/w| \geq 1/2} \right] \end{aligned} \quad (3.36)$$

The second term above is bounded by $O(aw)$ due to Fact 14 and Remark 13. We also note that since $|\tau| \leq \log(1/\varepsilon)$ and $w := C_6 \left(\frac{\log^2 1/\varepsilon}{\varepsilon^2} \right)^{1/2}$, for sufficiently large absolute constant C_6 we have $|\tau|/w \leq 0.1$. Therefore, whenever $(z - \tau)/w \geq 1/2$ we also have $z \geq 0.4w$. This allows us to upper bound

$$\begin{aligned} \mathbb{E}_{x \sim D} [|\text{sign}(u \cdot x - \tau) - P((u \cdot x - \tau)/w)|] &\leq \\ \varepsilon + O(aw) + \mathbb{E}_{z \sim D_{\text{projected}}} \left[\left(2 \cdot \left(4 \cdot \frac{z}{w} + 0.1 \right)^K + 1 \right) \mathbb{1}_{|z| \geq 0.4w} \right] &\leq \\ \varepsilon + O(aw) + \mathbb{E}_{z \sim D_{\text{projected}}} \left[\left(3 \cdot \left(5 \cdot \frac{z}{w} \right)^K \right) \mathbb{1}_{|z| \geq 0.4w} \right] \end{aligned} \quad (3.37)$$

Decomposing the interval $[0.4, \infty]$ into a union $\bigcup_{i=0}^{\infty} [0.4 \cdot 2^i, 0.4 \cdot 2^{i+1}]$ and using Definition 16 and Remark 13 we can now upper-bound the last term above as follows:

$$\begin{aligned} \mathbb{E}_{z \sim D_{\text{projected}}} \left[\left(3 \cdot \left(5 \cdot \frac{z}{w} \right)^K \right) \mathbb{1}_{|z| \geq 0.4w} \right] &\leq \\ 2 \sum_{i=0}^{\infty} (3 \cdot (4 \cdot 2^i)^K \Pr_{z \sim D_{\text{projected}}} [|z| \geq 2^{i-1} \cdot 0.4w]) &\leq \\ 6 \sum_{i=0}^{\infty} ((4 \cdot 2^i)^K \exp(-C(2^{i-3}w)^2)) \end{aligned} \quad (3.38)$$

Substituting $K = O\left(\frac{\log^2 1/\varepsilon}{\varepsilon^2}\right)$ and $w = C_6 \left(\frac{\log^2 1/\varepsilon}{\varepsilon^2}\right)^{1/2}$, we can bound each term in the sum above

$$(4 \cdot 2^i)^K \exp(-C(2^{i-3}w)^2) \leq \exp\left((i+2) \cdot O\left(\frac{\log^2 1/\varepsilon}{\varepsilon^2}\right) - C \cdot C_6 2^{i-3} \frac{\log^2 1/\varepsilon}{\varepsilon^2}\right)$$

For a sufficiently large absolute constant C_6 the above is at most $\varepsilon/2^{i+1}$, which substituted into Equation (3.38) gives us

$$\mathbb{E}_{z \sim D_{\text{projected}}} \left[\left(3 \cdot \left(5 \cdot \frac{z}{w} \right)^K \right) \mathbb{1}_{|z| \geq 0.4w} \right] \leq \varepsilon.$$

Finally, substituting the inequality above into Equation (3.37), we get

$$\mathbb{E}_{x \sim D} [|\text{sign}(u \cdot x) - P(u \cdot x/w)|] = O(\varepsilon),$$

finishing the proof. $\square$

3.3.3 Analyzing LearnRealValued

Theorem 11 (Correctness of Algorithm 2, LEARNREALVALUED). *Let $r \in (0, 1)$, and let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{\pm 1\}$ whose $\mathbb{R}^d$ -marginal is a subgaussian isotropic log-concave distribution over $\mathbb{R}^d$. Let $\mathcal{F}$ be the class of halfspaces on $\mathbb{R}^d$. Then, for some sufficiently large absolute constant C , the algorithm LearnRealValued, given ε, δ and sample access to $\mathcal{D}$, runs in time $\text{poly}\left(d^{O(\log^2(1/\varepsilon)/\varepsilon^2)}\right)$ and will with probability at least $1 - O(\delta)$ return a degree- $O(\log^2(1/\varepsilon)/\varepsilon^2)$ polynomial p for which the following hold:*

$$\mathbb{E}_{t \sim [-1, 1]} \left[\mathbb{E}_{(x, y) \sim \mathcal{D}} [\phi_{\text{sign}(p-t), 10r}(x)] \right] \leq 100r + O(\varepsilon), \quad (3.27)$$

$$\Pr_{(x, y) \sim \mathcal{D}} \left[\mathbb{E}_{t \sim [-1, 1]} [\phi_{\text{sign}(p(x)-t), 10r}(x)] \geq 0.7 \right] \leq O(\varepsilon), \quad (3.28)$$

$$\mathbb{E}_{t \sim [-1, 1]} \left[\Pr_{(x, y) \sim \mathcal{D}} [\text{sign}(p(x) - t) \neq y] \right] \leq \text{opt} + O(\varepsilon). \quad (3.29)$$

We first argue that the run-time is indeed $d^{O(\log^2(1/\varepsilon)/\varepsilon^2)}$. The algorithm operates with polynomials of degree $k = O(\log^2(1/\varepsilon)/\varepsilon^2)$ over $\mathbb{R}^d$, which are described via $d^O(k)$ coefficients. It remains to show that the constraints for the optimization task are indeed convex in the coefficients in the polynomial p . Recall that the constraints are:

- Constraint 1: $\frac{1}{|S|} \sum_{x \in S} \tilde{\phi}_{p, T, 10r}(x) \leq 100r + \varepsilon$
- Constraint 2: $\frac{1}{|S|} \sum_{x \in S} \tilde{\psi}_{p, T, 10r}(x) \leq \varepsilon$

To see that these constraints are indeed convex, first note that, by definition of $\tilde{\phi}_{p,T,10r}(x)$ and the triangle inequality, we have for any pair of polynomials p_1, p_2 , x in R^d and $\alpha \in [0, 1]$

$$\tilde{\phi}_{(\alpha p_1 + (1-\alpha)p_2), T, 10r}(x) \leq \alpha \tilde{\phi}_{p_1, T, 10r}(x) + (1-\alpha) \tilde{\phi}_{p_2, T, 10r}(x), \quad (3.39)$$

which implies that Constraint 1 is convex. Likewise, recalling the definition of ψ

$$\tilde{\psi}_{p, T, 10r} := 10 \cdot \mathbb{1}[\tilde{\phi}(x)_{p, T, 10r} > 0.6] \cdot (\tilde{\phi}(x)_{p, T, 10r} - 0.6),$$

and combining it with Equation (3.39) we see that

$$\tilde{\psi}_{\alpha p_1 + (1-\alpha)p_2, T, 10r}(x) \leq \alpha \tilde{\psi}_{p_1, T, 10r}(x) + (1-\alpha) \tilde{\psi}_{p_2, T, 10r}(x).$$

We now proceed to the proof of correctness, the following claim will be key to our argument, which will be proven in the next subsection (Section 3.3.4):

Claim 5. *For any specific iteration i of the algorithm, the polynomial p_i will satisfy the following:*

$$\mathbb{E}_{(x,y) \sim S_i} [\tilde{\phi}_{p_i, T_i, 10r}(x)] \leq 100r + O(\varepsilon), \quad (3.40)$$

$$\mathbb{E}_{(x,y) \sim S_i} [\tilde{\psi}_{p_i, T_i, 10r}(x)] \leq O(\varepsilon), \quad (3.41)$$

and the following holds with probability at least 0.9:

$$\mathbb{E}_{(x,y) \sim S_i} \left[\frac{|p_i(x) - y|}{2} \right] \leq \min_{f \in \mathcal{F}} (\Pr_{(x,y) \sim D} [f(x) \neq y]) + O(\varepsilon), \quad (3.42)$$

We now prove Theorem 11 assuming Claim 5.

Proof of Theorem 11. First we note that for sufficiently large value of the absolute constant C , with probability at least $1 - O(\delta)$ for all i in $\{1, \dots, \log 1/\delta\}$, the set S_i will satisfy Claim 2, Claim 3 and Fact 9, and the set T_i will satisfy Claim 2. Furthermore, since the main iteration is repeated $O(\log 1/\delta)$ times, we see that with probability $1 - O(\delta)$ the polynomial $p = p_{i^*}$ will satisfy Equations 3.40, 3.41 and 3.42. Denoting $S = S_{i^*}$ and $T = T_{i^*}$, we have

$$\begin{aligned} \mathbb{E}_{t \sim [-1, 1]} [\mathbb{E}_{(x,y) \sim D} [\phi_{\text{sign}(p-t), 10r}(x)]] &\stackrel{\text{Since } S \text{ satisfies Claim 2}}{\leq} \mathbb{E}_{t \sim [-1, 1]} [\mathbb{E}_{(x,y) \sim S} [\phi_{\text{sign}(p-t), 10r}(x)]] + O(\varepsilon) \leq \\ &\leq \underbrace{\mathbb{E}_{t \sim [-1, 1]} [\mathbb{E}_{(x,y) \sim S} [\tilde{\phi}_{\text{sign}(p-t), T, 10r}(x)]]}_{\text{Since } T \text{ satisfies Claim 1, and WLOG } \varepsilon < 0.02.} + O(\varepsilon) \leq \underbrace{\mathbb{E}_{(x,y) \sim S} [\tilde{\phi}_{p, T, 10r}(x)]}_{\text{By Fact 4}} + O(\varepsilon) \stackrel{\text{By Equation (3.40)}}{\leq} 100r + O(\varepsilon), \end{aligned}$$

which gives us Equation (3.27). We now derive Equation (3.28) as follows:

$$\begin{aligned}
& \Pr_{(x,y) \sim \mathcal{D}} [\mathbb{E}_{t \in [-1,1]} [\phi_{\text{sign}(p(x)-t), 10r}] \geq 0.7] \leq \overbrace{\Pr_{(x,y) \sim S} [\mathbb{E}_{t \in [-1,1]} [\phi_{\text{sign}(p-t), 10r}(x)] \geq 0.67] + O(\varepsilon)}^{\text{Since } S \text{ satisfies Claim 3}} \leq \\
& \leq \underbrace{\Pr_{(x,y) \sim S} [\mathbb{E}_{t \sim [-1,1]} [\tilde{\phi}_{\text{sign}(p-t), T, 10r}(x)] \geq 0.65] + O(\varepsilon)}_{\text{Since } T \text{ satisfies Claim 1, and WLOG } \varepsilon < 0.02.} \leq \underbrace{O(1) \mathbb{E}_{(x,y) \sim S} [\tilde{\psi}_{p,T,10r}(x)] + O(\varepsilon)}_{\text{By Fact 4 and definition of } \tilde{\psi} \text{ (Section 3.1.1)}} \stackrel{\text{By Equation (3.41)}}{\leq} O(\varepsilon)
\end{aligned}$$

Finally, we derive Equation (3.29) as follows:

$$\begin{aligned}
\mathbb{E}_{t \sim [-1,1]} [\text{err}_D [\text{sign}(p-t)]] & \leq \overbrace{\mathbb{E}_{t \sim [-1,1]} [\widehat{\text{err}}_S [\text{sign}(p-t)]] + O(\varepsilon)}^{\text{Since } S \text{ satisfies Fact 9}} \leq \\
& \leq \underbrace{\mathbb{E}_{(x,y) \sim S} \left[\frac{|p(x) - y|}{2} \right] + O(\varepsilon)}_{\text{By Fact 4}} \stackrel{\text{By Equation (3.42) and defn of opt}}{\leq} \text{opt} + O(\varepsilon).
\end{aligned}$$

□

3.3.4 Proving Claim 5

Finally, we finish this section by proving Claim 5. Inspecting the algorithm LearnRealValued, we see that the linear program is always feasible for any input dataset S_i , because the all-zeros polynomial will satisfy the constraints in the linear program (i.e. Equations 3.40 and 3.41). It remains to show that with probability at least 0.9 there exists a polynomial p_i satisfying all three of Equations 3.40, 3.41 and 3.42.

Suppose f^* is the optimal halfspace for D , i.e.

$$\Pr_{(x,y) \sim D} [f^*(x) \neq y] = \min_{f \in \mathcal{F}} (\Pr_{(x,y) \sim D} [f(x) \neq y]). \quad (3.43)$$

Let D_{marginal} denote the $\mathbb{R}^d$ -marginal of D , and let D'_{marginal} denote distribution of $x + 10rz$, where x is sampled from D_{marginal} and z is sampled from $\mathcal{N}(0, I_d)$. Since D_{marginal} is assumed to be sub-Gaussian log-concave, Fact 11 and Fact 12 tell us that D'_{marginal} is also sub-Gaussian log-concave. Furthermore, since D_{marginal} is isotropic (i.e. has mean zero and identity covariance) and r is in $(0, 1)$ we see that

$$I_d \preceq \mathbb{E}_{x \sim D'_{\text{marginal}}} [xx^T] \preceq 101I_d.$$

Therefore, Claim 4 tells us that for a sufficiently large absolute constant C' there is a polynomial of p^* of degree $\frac{C' \log^2(1/\varepsilon)}{\varepsilon^2}$ for which

$$\mathbb{E}_{x \sim D_{\text{marginal}}} [|f^*(x) - p^*(x)|] \leq O(\varepsilon) \quad (3.44)$$

$$\mathbb{E}_{x \sim D'_{\text{marginal}}} [|f^*(x) - p^*(x)|] \leq O(\varepsilon) \quad (3.45)$$

Recall that for each element (x, y) in S_i , the variable x is distributed according to D_{marginal} . Similarly, for each (x, y) in S_i and z in T_i the sum $x + 10rz$ is distributed D'_{marginal} . This allows us to use inequalities above to conclude that with probability at least 0.99 we have

$$\mathbb{E}_{(x,y) \sim S_i} [|p^*(x) - f^*(x)|] \leq O(\varepsilon) \quad (3.46)$$

$$\mathbb{E}_{(x,y) \sim S_i, z \sim T_i} [|p^*(x + 10rz) - f^*(x + 10rz)|] \leq O(\varepsilon) \quad (3.47)$$

In the remainder of this section we show that with probability at least 0.9 over the choice of S_i and T_i the choice $p_i = p^*$ will indeed satisfy Equations 3.40, 3.41 and 3.42.

Bounding $\mathbb{E}_{x \sim D_{\text{marginal}}}[\phi_{f^*, 10r}(x)]$ and $\mathbb{E}_{x \sim D_{\text{marginal}}}[\psi_{f^*}(x)]$

Recall that $f^*(x) = \text{sign}(u \cdot x - \tau)$ for some unit vector u and real τ . Here, we show that f^* satisfies the following two inequalities

$$\mathbb{E}_{x \sim D_{\text{marginal}}}[\phi_{f^*, 10r}(x)] \leq 60r, \quad (3.48)$$

$$\mathbb{E}_{x \sim D_{\text{marginal}}}[\psi_{f^*}(x)] = 0. \quad (3.49)$$

Indeed, fix a specific value of x . Recall that $\phi_{f^*, 10r}(x)$ is the probability that $\text{sign}(u \cdot x - \tau) \neq \text{sign}(u \cdot (x + 10rz) - \tau)$ where z is sampled from $\mathcal{N}(0, I_d)$. With probability at least 0.5 over the choice of Z we have $\text{sign}(u \cdot z) = \text{sign}(u \cdot x - \tau)$ which leads to $\text{sign}(u \cdot x - \tau) \neq \text{sign}(u \cdot (x + 10rz) - \tau)$. Thus, $\phi_{f^*, 10r}(x) \leq 0.5$ and substituting this into the definition of ψ we see that $\psi(x) = 0$, which proves Equation (3.49).

Now consider points x in $\mathbb{R}^d$ such that $|u \cdot x - \tau|$ is in the interval $[10r \cdot (i - 1), 10r \cdot i]$ for an integer i . We have the following two observations:

- By Fact 14, we see that the probability over x sampled from D_{marginal} that $|u \cdot x - \tau|$ is in the interval $[10r \cdot (i - 1), 10r \cdot i]$ can be upper-bounded by $20r$.
- For a fixed x for which $|u \cdot x - \tau|$ is in the interval $[10r \cdot (i - 1), 10r \cdot i]$, via Chebyshev's inequality, it follows that

$$\Pr_{z \sim \mathcal{N}(0, I_d)}[\text{sign}(u \cdot x - \tau) \neq \text{sign}(u \cdot (x + 10rz) - \tau)] \leq \max \left(1, \left(\frac{10r}{10r(i - 1)} \right)^2 \right) = \max \left(1, \left(\frac{1}{i - 1} \right)^2 \right)$$

Combining the two observations above, we see that

$$\mathbb{E}_{x \sim D_{\text{marginal}}}[\phi_{f^*, 10r}(x)] \leq \sum_{i=0}^{\infty} 20r \max \left(1, \left(\frac{1}{i - 1} \right)^2 \right) \leq 60r,$$

which yields Equation (3.48).

Bounding $\mathbb{E}_{(x,y) \sim S_i} [\tilde{\phi}_{p^*, T_i, 10r}(x)]$ and $\mathbb{E}_{(x,y) \sim S_i} [\tilde{\psi}_{p^*, T_i, 10r}(x)]$.

We observe that

$$\begin{aligned} \mathbb{E}_{(x,y) \sim S_i} [\tilde{\phi}_{p^*, T_i}(x)] &= \mathbb{E}_{(x,y) \sim S_i} \left[\left| \frac{p^*(x) - p^*(x + 10rz)}{2} \right| \right] \leq \\ &\mathbb{E}_{(x,y) \sim S_i} \left[\left| \frac{f^*(x) - f^*(x + 10rz)}{2} \right| \right] + \mathbb{E}_{(x,y) \sim S_i} \left[\left| \frac{p^*(x) - f^*(x)}{2} \right| \right] + \mathbb{E}_{(x,y) \sim S_i} \left[\left| \frac{p^*(x + 10rz) - f^*(x + 10rz)}{2} \right| \right] \\ &\mathbb{E}_{(x,y) \sim S_i} [\tilde{\phi}_{f^*, T_i, 10r}(x)] + \underbrace{\mathbb{E}_{(x,y) \sim S_i} |p^*(x) - f^*(x)| + \mathbb{E}_{(x,y) \sim S_i, z \sim T_i} |p^*(x + 10rz) - f^*(x + 10rz)|}_{=O(\varepsilon) \text{ by Equations 3.46 and 3.47}}. \end{aligned} \quad (3.50)$$

Analogously, by inspecting the definition of ψ , we see that for any x we have

$$\left| \tilde{\psi}_{p^*, T_i}(x) - \tilde{\psi}_{f^*, T_i}(x) \right| \leq |f^*(x) - p^*(x)| + O(1) \cdot \mathbb{E}_{z \sim T_i} [|f^*(x + 10z) - p^*(x + 10z)|]$$

Averaging over x in S_i , we have

$$\begin{aligned} \mathbb{E}_{(x,y) \sim S_i} [\tilde{\psi}_{p^*, T_i}(x)] &\leq \\ \mathbb{E}_{(x,y) \sim S_i} [\tilde{\phi}_{f^*, T_i, 10r}(x)] &+ O(1) \cdot \underbrace{(\mathbb{E}_{(x,y) \sim S_i} |p^*(x) - f^*(x)| + \mathbb{E}_{(x,y) \sim S_i, z \sim T_i} |p^*(x + 10rz) - f^*(x + 10rz)|)}_{=O(\varepsilon) \text{ by Equations 3.46 and 3.47}}. \end{aligned} \quad (3.51)$$

Recall that f^* is a $\{\pm 1\}$ -valued function, which implies that $\tilde{\phi}_{f^*, T_i, 10r}(x + 10rz)$ is in $[0, 1]$ for all x and z . This allows us to use the Chebyshev's inequality to conclude that:

$$\mathbb{E}_{T_i} \left[\left| \tilde{\phi}_{f^*, T_i, 10r}(x) - \phi_{f^*, 10r}(x) \right| \right] \leq O \left(\frac{1}{\sqrt{|T_i|}} \right).$$

Inspecting the definition of $\tilde{\psi}$, we see that for any x it is the case that

$$\left| \tilde{\psi}_{f^*, T_i}(x) - \psi_{f^*}(x) \right| \leq O(1) \cdot \left| \tilde{\phi}_{f^*, T_i, 10r}(x) - \phi_{f^*, 10r}(x) \right|,$$

which implies that

$$\mathbb{E}_{T_i} \left[\left| \tilde{\psi}_{f^*, T_i}(x) - \psi_{f^*}(x) \right| \right] \leq O \left(\frac{1}{\sqrt{|T_i|}} \right)$$

Averaging the inequalities above over x in S_i , we have:

$$\mathbb{E}_{T_i} \left[\mathbb{E}_{(x,y) \sim S_i} \left[\left| \tilde{\phi}_{f^*, T_i, 10r}(x) - \phi_{f^*, 10r}(x) \right| \right] \right] \leq O \left(\frac{1}{\sqrt{|T_i|}} \right)$$

$$\mathbb{E}_{T_i} \left[\mathbb{E}_{(x,y) \sim S_i} \left[\left| \tilde{\psi}_{f^*, T_i}(x) - \psi_{f^*}(x) \right| \right] \right] \leq O \left(\frac{1}{\sqrt{|T_i|}} \right)$$

Thus, with probability at least 0.99 over the choice of T_i , we have

$$\left| \mathbb{E}_{(x,y) \sim S_i} \left[\tilde{\phi}_{f^*, T_i, 10r}(x) \right] - \mathbb{E}_{(x,y) \sim S_i} \left[\phi_{f^*, 10r}(x) \right] \right| \leq \mathbb{E}_{(x,y) \sim S_i} \left[\left| \tilde{\phi}_{f^*, T_i, 10r}(x) - \phi_{f^*, 10r}(x) \right| \right] \leq O \left(\frac{1}{\sqrt{|T_i|}} \right) \quad (3.52)$$

$$\left| \mathbb{E}_{(x,y) \sim S_i} \left[\tilde{\psi}_{f^*, T_i}(x) \right] - \mathbb{E}_{(x,y) \sim S_i} \left[\psi_{f^*}(x) \right] \right| \leq \mathbb{E}_{(x,y) \sim S_i} \left[\left| \tilde{\psi}_{f^*, T_i}(x) - \psi_{f^*}(x) \right| \right] \leq O \left(\frac{1}{\sqrt{|T_i|}} \right) \quad (3.53)$$

Again, recalling that f^* is a $\{\pm 1\}$ -valued function, we see that $\phi_{f^*, 10r}(x)$ is in $[0, 1]$ for all x and $\psi_{f^*}(x)$ is likewise bounded by $O(1)$ in absolute value. This allows us to use the Chebyshev's inequality to conclude that with probability at least 0.99 over the choice of S_i we have:

$$\left| \mathbb{E}_{(x,y) \sim S_i} \left[\phi_{f^*, 10r}(x) \right] - \mathbb{E}_{x \sim D_{\text{marginal}}} \left[\phi_{f^*, 10r}(x) \right] \right| \leq O \left(\frac{1}{\sqrt{|S_i|}} \right) \quad (3.54)$$

$$\left| \mathbb{E}_{(x,y) \sim S_i} \left[\psi_{f^*}(x) \right] - \mathbb{E}_{x \sim D_{\text{marginal}}} \left[\psi_{f^*}(x) \right] \right| \leq O \left(\frac{1}{\sqrt{|S_i|}} \right) \quad (3.55)$$

Overall, combining Equation (3.48), Equation (3.54), Equation (3.52) and Equation (3.50) we see that with probability at least 0.97 over the choice of S_i and T_i it is the case that

$$\mathbb{E}_{(x,y) \sim S_i} [\tilde{\phi}_{p^*, 10r}(x)] \leq 60r + O(\varepsilon).$$

Similarly, combining Equation (3.49), Equation (3.55), Equation (3.53) and Equation (3.51) we see that with probability at least 0.97 over the choice of S_i and T_i it is the case that

$$\mathbb{E}_{(x,y) \sim S_i} [\tilde{\psi}_{p^*, T_i}(x)] \leq O(\varepsilon).$$

p^* has a small empirical error.

Finally, we argue that with probability 0.99 we have $\mathbb{E}_{(x,y) \sim S_i} \left[\frac{|p^*(x) - y|}{2} \right] \leq \mathbb{E}_{(x,y) \sim D} \left[\frac{|f^*(x) - y|}{2} \right] + O(\varepsilon)$. Indeed, Hoeffding's inequality implies that with probability at least 0.995 we have

$$\mathbb{E}_{(x,y) \sim S_i} \left[\frac{|f^*(x) - y|}{2} \right] \leq \mathbb{E}_{(x,y) \sim D} \left[\frac{|f^*(x) - y|}{2} \right] + \varepsilon,$$

whereas Markov's inequality tells us that with probability 0.995 we have

$$\mathbb{E}_{(x,y) \sim S_i} \left[\frac{|f^*(x) - p^*(x)|}{2} \right] \leq O(1) \cdot \mathbb{E}_{(x,y) \sim D} \left[\frac{|f^*(x) - p^*(x)|}{2} \right] = O(\varepsilon).$$

Overall, we see that with probability at least 0.99 it holds that

$$E_{(x,y) \sim S_i} \left[\frac{|p^*(x) - y|}{2} \right] \leq E_{(x,y) \sim S_i} \left[\frac{|f^*(x) - y|}{2} \right] + E_{(x,y) \sim S_i} \left[\frac{|p^*(x) - f^*(x)|}{2} \right] \leq \mathbb{E}_{(x,y) \sim D} \left[\frac{|f^*(x) - y|}{2} \right] + O(\epsilon)$$

3.4 Pseudocode and Analysis of ComputeClassifier

3.4.1 Local Correction of Adversarial Robustness

Theorem 16 (Correctness and complexity of [Algorithm 3](#)). *Let $\hat{\phi}$ be an ε -accurate local noise sensitivity approximator for degree- k PTFs over $\mathbb{R}^d$, and let τ be the run-time and query complexity of $\hat{\phi}$. Then there exists an algorithm $\text{ROBUSTNESSLCA}(x, g, r)$ that takes x in $\mathbb{R}^d$, query access to a function $g : \mathbb{R}^d \rightarrow \{-1, 1\}$, perturbation size parameter r in $(0, 1)$. The algorithm makes $O(\tau)$ queries to g , runs in time $O(\tau + d)$, and satisfies the specifications below.*

Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d$, and let g be a degree- k PTF. Let the function $h : \mathbb{R}^d \rightarrow \{-1, 1\}$ be defined as $h(x) := \text{ROBUSTNESSLCA}(x, g, r)$. Then the following properties hold:

- a) $\Pr_{x \sim \mathcal{D}}[g(x) \neq h(x)] \leq \widehat{\text{iso}}_{\mathcal{D}, 10r}(g, 0.8)$.
- b) $\Pr_{x \sim \mathcal{D}}[\phi_{g, 10r}(x) \leq 0.1] \geq 1 - O\left(\widehat{\text{NS}}_{\mathcal{D}, 10r}(g)\right)$.
- c) *For every x in $\mathbb{R}^d$, if $\hat{\phi}_{g, 10r}(x) \leq 0.1$, then for every x' with $\|x' - x\| \leq r$ we have $h(x') = h(x)$.*

Algorithm 3 $\text{ROBUSTNESSLCA}(x, g, r)$:

- 1: **Input:** point $x \in \mathbb{R}^d$, black-box representation of a function $g : \mathbb{R}^d \rightarrow \{-1, 1\}$,
 - 2: robustness radius r
 - 3: **Uses:** local noise sensitivity approximator $\hat{\phi}$. (See [Definition 13](#))
 - 4: **Output:** $b \in \{-1, 1\}$
 - 5: **if** $\hat{\phi}_{g, 10r}(x) > 0.8$ **then**
 - 6: **return** $-g(x)$
 - 7: **else**
 - 8: **return** $g(x)$
-

Proof. Inspecting the algorithm ROBUSTNESSLCA , we see that condition (a) follows directly, as only isolated points have their labels changed. Meanwhile, condition (b) follows directly from Markov's inequality and the definition of $\widehat{\text{NS}}$ ([Section 3.1.1](#)).

To show condition (c), we first recall that by the assumption that $\hat{\phi}$ is an ε -accurate approximator, for every degree- k polynomial threshold function g and every x in $\mathbb{R}^d$ we have

$$\left| \hat{\phi}_{g, 10r}(x) - \phi_{g, 10r}(x) \right| \leq \varepsilon \leq 0.01, \quad (3.56)$$

where for the last we assumed without loss of generality that $\varepsilon < 0.01$. This implies that whenever $\hat{\phi}_{g,10r}(x) \leq 0.1$ we have $\phi_{g,10r}(x) \leq 0.11$. Suppose x' has distance at most r from x . Since $\phi_{g,r}(x)$ is defined as

$$\phi_{g,10r}(x) := \Pr_{z \sim N(0, I_d)} [g(x) \neq g(x + 10rz)],$$

we see that

$$\begin{aligned} \Pr_{z \sim N(0, I_d)} [g(x) \neq g(x' + 10rz)] &\leq \\ \Pr_{z \sim N(0, I_d)} [g(x) \neq g(x + 10rz)] + d_{TV}(N(x, 100r^2 I_d), N(x', 100r^2 I_d)) &\leq \\ \Pr_{z \sim N(0, I_d)} [g(x) \neq g(x + 10rz)] + \frac{\|x - x'\|}{20r}, \end{aligned}$$

where the last inequality is implied by [Fact 5](#). Overall, since we know that $\phi_{g,10r}(x) \leq 0.11$ and $\|x - x'\| \leq r$, we have

$$\Pr_{z \sim N(0, I_d)} [g(x) \neq g(x' + 10rz)] \leq 0.16.$$

Now, we consider the following two cases:

- Suppose $g(x') = g(x)$: then the above implies that $\phi_{g,10r}(x') \leq 0.16$, which combined with [Equation \(3.56\)](#) implies that $\hat{\phi}_{g,10r}(x') \leq 0.17$. Therefore, by inspecting the algorithm ROBUSTNESSLCA we see that $h(x') = g(x') = g(x) = h(x)$.
- On the other hand, suppose $g(x') = -g(x)$. Then, we have $\phi_{g,10r}(x') \geq 1 - 0.16 = 0.84$, which combined with [Equation \(3.56\)](#) implies that $\hat{\phi}_{g,10r}(x') \geq 0.83$. Therefore, by inspecting the algorithm ROBUSTNESSLCA we see that $h(x') = -g(x') = g(x) = h(x)$.

In either case, we have $h(x') = h(x)$. □

3.4.2 Finding Good Rounding Thresholds

Theorem 17 (Correctness of [Algorithm 4](#)). *Let $\varepsilon \in (0, 1)$ and $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{\pm 1\}$. Let $\hat{\phi}$ be an ε -accurate local noise sensitivity approximator for degree- k PTFs over $\mathbb{R}^d$. Let $p : \mathbb{R}^d \rightarrow \mathbb{R}$ be a polynomial of degree $\leq k$ satisfying the following:*

$$\mathbb{E}_{t \sim [-1, 1]} \left[\mathbb{E}_{(x, y) \sim \mathcal{D}} [\phi_{\text{sign}(p-t), 10r}(x)] \right] \leq 100r + O(\varepsilon), \quad (3.57)$$

$$\Pr_{(x, y) \sim \mathcal{D}} \left[\mathbb{E}_{t \sim [-1, 1]} [\phi_{\text{sign}(p(x)-t), 10r}(x)] \geq 0.7 \right] \leq O(\varepsilon). \quad (3.58)$$

The algorithm `COMPUTEROUNDINGTHRESHOLDS` outputs real numbers t_1, t_2, t_3, t_4 and $w_1, w_2, w_3, w_4 \in [0, 1]$ such that $\sum_i w_i = 1$. Let the hypotheses g_1, g_2, g_3, g_4 be defined as

$$g_i(x) = \text{sign}(p(x) - t_i).$$

Then the following properties hold with probability at least $1 - O(\delta)$:

- $\sum_i w_i \cdot \text{err}_{\mathcal{D}}(g_i) \leq \mathbb{E}_{t \sim [-1, 1]}[\text{err}_{\mathcal{D}}(\text{sign}(p - t))] + O(\varepsilon)$
- $\sum_i w_i \cdot \widehat{\text{NS}}_{\mathcal{D}, 10r}(g_i) \leq 200r + O(\varepsilon)$.
- $\sum_i w_i \cdot \widehat{\text{iso}}_{\mathcal{D}, 10r}(g_i, 0.8) \leq O(\varepsilon)$.

Algorithm 4 `COMPUTEROUNDINGTHRESHOLDS`($D, p, r, \varepsilon, \delta$):

- 1: **Input:** sample access to D , robustness radius r ,
 - 2: error bound ε , confidence bound δ
 - 3: **Uses:** local noise sensitivity approximator $\hat{\phi}$. (See [Definition 13](#))
 - 4: **Output:** thresholds $\vec{t} = t_1, t_2, t_3, t_4 \in [-1, 1]$ and weights $\vec{w} = w_1, w_2, w_3, w_4 \in [0, 1]$
 - 5: Initialize $Q := \emptyset$.
 - 6: $M \leftarrow 100/\varepsilon^3 \cdot \log^2(1/\delta)$ i.i.d. samples (x, y) from $\mathcal{D}$
 - 7: **for** $i \in [\log(1/\delta)]$ **do**
 - 8: **for** $j \in [100/\varepsilon^2]$ **do**
 - 9: Draw rounding threshold t u.a.r from $[-1, 1]$ and define rounded function $p_t(x) := \text{sign}(p(x) - t)$.
 - 10: Let $\text{err}_t := \text{err}_M(p_t)$.
 - 11: Let $\text{NS}_t := \frac{1}{|M|} \cdot \sum_{x \in M} \hat{\phi}_{p_t, 10r}(x)$.
 - 12: Let $\text{iso}_t := \frac{1}{|M|} \cdot \sum_{x \in M} \mathbb{1}[\hat{\phi}_{p_t, 10r}(x) \geq 0.8]$.
 - 13: Let $q_t := (\text{err}_t, \text{NS}_t, \text{iso}_t)$.
 - 14: Add q_t to Q .
 - 15: $\widehat{\text{opt}} := \frac{1}{|Q|} \sum_{(\text{err}_t, \text{NS}_t, \text{iso}_t) \in Q} (\text{err}_t)$
 - 16: **for** tuple $q_1, q_2, q_3, q_4 \in Q^4$ **do**
 - 17: Solve the linear program with variables w_1, w_2, w_3, w_4 defined by the following constraints:
 - $\sum_{i=1}^4 w_i = 1$
 - $\sum_{i=1}^4 w_i q_{i,1} \leq \widehat{\text{opt}} + C\varepsilon$ C is a sufficiently large absolute constant.
 - $\sum_{i=1}^4 w_i q_{i,2} \leq 200r + C\varepsilon$
 - $\sum_{i=1}^4 w_i q_{i,3} \leq C\varepsilon$
 - 18: If a solution is found, **return** $\vec{t} = t_1, t_2, t_3, t_4$ and $\vec{w} = w_1, w_2, w_3, w_4$.
 - 19: **return** $\perp$
-

We argue that with high probability over the choice of the $O(1/\varepsilon^2)$ random rounding thresholds, the equal-weighted mixture of the rounded functions has each of the desired properties.

Claim 6 (Properties of the equal mixture of all rounding thresholds). *Let $\varepsilon \in (0, 1)$. Let $p : \mathbb{R}^d \rightarrow \mathbb{R}$ be a polynomial of degree $\leq k$ satisfying the following:*

- $\mathbb{E}_{t \sim [-1, 1]} \left[\mathbb{E}_{(x, y) \sim \mathcal{D}} [\hat{\phi}_{\text{sign}(p-t), 10r}(x)] \right] \leq \alpha$
- $\Pr_{(x, y) \sim \mathcal{D}} \left[\mathbb{E}_{t \in [-1, 1]} [\hat{\phi}_{\text{sign}(p(x)-t), 10r}(x)] \geq 0.75 \right] \leq O(\varepsilon).$

Let Θ be a set of $100/\varepsilon^2$ real numbers drawn uniformly and independently from $[-1, 1]$. Then the following hold with probability ≥ 0.99 :

- (a) $\mathbb{E}_{t \in \Theta} [\text{err}_{\mathcal{D}}(\text{sign}(p-t))] \leq \mathbb{E}_{t \sim [-1, 1]} [\text{err}_{\mathcal{D}}(\text{sign}(p-t))] + O(\varepsilon)$
- (b) $\mathbb{E}_{t \in \Theta} [\mathbb{E}_{x \sim \mathcal{D}} [\hat{\phi}_{\text{sign}(p-t), 10r}(x)]] \leq 2\alpha + O(\varepsilon)$
- (c) $\mathbb{E}_{t \in \Theta} [\Pr_{x \sim \mathcal{D}} [\hat{\phi}_{\text{sign}(p-t), 10r}(x) > 0.8]] \leq O(\varepsilon).$

Proof of Claim 6. We will show that each of the conditions holds with high probability, then union bound.

- (a) Let $X = \sum_{t \in \Theta} \text{err}_{\mathcal{D}}(\text{sign}(p-t))$. We have $\mathbb{E}_{\Theta}[X] = |\Theta| \cdot \mathbb{E}_{t \sim [-1, 1]} [\text{err}_{\mathcal{D}}(\text{sign}(p-t))]$ and we apply Hoeffding's inequality:

$$\begin{aligned} \Pr \left[\mathbb{E}_{t \in \Theta} [\text{err}_{\mathcal{D}}(\text{sign}(p-t))] > \mathbb{E}_{t \sim [-1, 1]} [\text{err}_{\mathcal{D}}(\text{sign}(p-t))] + \varepsilon \right] \\ = \Pr [(X - \mathbb{E}[X]) / |\Theta| > \varepsilon] \leq \exp(-2\varepsilon^2 \cdot 100/\varepsilon^2) \leq 0.001. \end{aligned}$$

- (b) We have by assumption that

$$\mathbb{E}_{t \sim [-1, 1]} \mathbb{E}_{x \sim \mathcal{D}} [\hat{\phi}_{\text{sign}(p-t)}(x)] \leq \alpha.$$

Let $X = \sum_{t \in \Theta} \mathbb{E}_{x \sim \mathcal{D}} [\hat{\phi}_{\text{sign}(p-t), 10r}(x)]$, which has expectation $\leq |\Theta| \cdot \alpha$. Then by Hoeffding's inequality, we have

$$\Pr[X/|\Theta| > \alpha + \varepsilon] \leq \Pr[(X - \mathbb{E}[X]) / |\Theta| > \varepsilon] \leq \exp(-2\varepsilon^2 \cdot 100/\varepsilon^2) \leq 0.001.$$

- (c) We have by assumption that $\Pr_{x \sim \mathcal{D}} [\mathbb{E}_{t \sim [-1, 1]} [\hat{\phi}_{\text{sign}(p-t)}(x)] \geq 0.75] \leq O(\varepsilon)$. We want to bound

$$\mathbb{E}_{t \in \Theta} \Pr_{x \sim \mathcal{D}} [\hat{\phi}_{\text{sign}(p-t)}(x) > 0.75].$$

We can upper bound this by

$$O(\varepsilon) + \mathbb{E}_{t \sim \Theta} \Pr_{x \sim \mathcal{D}} [\hat{\phi}_{\text{sign}(p-t)}(x) - \mathbb{E}_{t' \sim [-1,1]} [\hat{\phi}_{\text{sign}(p-t)}(x)] > 0.05] = \\ O(\varepsilon) + \Pr_{x \sim \mathcal{D}} \Pr_{t \sim \Theta} [\hat{\phi}_{\text{sign}(p-t)}(x) - \mathbb{E}_{t' \sim [-1,1]} [\hat{\phi}_{\text{sign}(p-t')}(x)] > 0.05].$$

Let $X = \sum_{t \in \Theta} \hat{\phi}_{\text{sign}(p-t)}(x)$; by Hoeffding's inequality we have

$$\Pr_{\Theta} \left[\mathbb{E}_{t \sim \Theta} [\hat{\phi}_{\text{sign}(p-t)}(x)] > \mathbb{E}_{t' \sim [-1,1]} [\hat{\phi}_{\text{sign}(p-t')}(x)] + 0.05 \right] = \Pr[(X - \mathbb{E}[X])/|\Theta| > 0.05] \\ \leq \exp(-2 \cdot 0.025 \cdot (100/\varepsilon^2)) \\ \leq \varepsilon/1000.$$

Thus, combining all three probabilities, we have:

$$\Pr_{x \sim \mathcal{D}} \Pr_{\Theta} \Pr_{t \sim \Theta} [\hat{\phi}_{\text{sign}(p-t), 10r}(x) - \mathbb{E}_{t' \sim [-1,1]} [\hat{\phi}_{\text{sign}(p-t'), 10r}(x)] > 0.05] \leq \varepsilon/1000$$

Then we apply a Markov bound:

$$\Pr_{\Theta} \left[\Pr_{x \sim \mathcal{D}, t \sim \Theta} [\hat{\phi}_{\text{sign}(p-t), 10r}(x) - \mathbb{E}_{t' \sim [-1,1]} [\hat{\phi}_{\text{sign}(p-t'), 10r}(x)] > 0.05] > \varepsilon \right] \leq 0.001$$

Union bounding over all the conditions, we have that all three conditions hold simultaneously with probability at least 0.99 over the choice of Θ . □

Now we will show the existence of a mixture using four rounding thresholds from Θ , and show that COMPUTE ROUNDING THRESHOLDS finds it. We will make use of the following well-known theorem of Carathéodory.

Theorem 18 (Carathéodory's theorem [29]). *Let $P \subset \mathbb{R}^k$ be a set of points and $\text{Conv}(P)$ be its convex hull. For any $p \in \text{Conv}(P)$, there exists a set $S \subseteq P$ of $k+1$ points such that p can be written as a convex combination of points in S .*

Proof of Theorem 17. We first note that since $\hat{\phi}$ is an ε -accurate local noise sensitivity approximator for degree- k PTFs, Equations 3.57 and 3.58 respectively imply that⁷

$$\mathbb{E}_{t \sim [-1,1]} \left[\mathbb{E}_{(x,y) \sim \mathcal{D}} [\hat{\phi}_{\text{sign}(p-t), 10r}(x)] \right] \leq 100r + O(\varepsilon), \quad (3.59)$$

$$\Pr_{\substack{x \sim \mathcal{D} \\ t \sim [-1,1]}} [\hat{\phi}_{\text{sign}(p-t), 10r}(x) > 0.75] \leq O(\varepsilon). \quad (3.60)$$

⁷We assume $\varepsilon \leq 0.05$ without loss of generality.

Thus, the assumptions of [Claim 6](#) hold with $\alpha = 100r$. For a threshold t , let the three-dimensional point $q_t := (\text{err}_t, \text{NS}_t, \text{iso}_t)$ be defined as in `COMPUTEROUNDINGTHRESHOLDS`. First we argue that each of these estimates is $O(\varepsilon)$ -accurate. Each estimate is the expectation of a random variable bounded in $[0, 1]$, so by Hoeffding's inequality we have

$$\Pr[|\text{err}_t - \text{err}_{\mathcal{D}}(\text{sign}(p - t))| \geq \varepsilon] \leq \exp(-2\varepsilon^2 \cdot |M|),$$

and likewise for each NS_t and iso_t . We set M large enough that this probability is $\leq \delta/2 \cdot \exp(-100/\varepsilon)$, giving us more than enough room to union bound over the $\log(1/\delta) \cdot 300/\varepsilon^2$ estimates. Likewise, by the Hoeffding bound, we have with probability $1 - \delta$ that

$$|\widehat{\text{opt}} - \mathbb{E}_{t \sim [-1, 1]}[\text{err}_{\mathcal{D}}(\text{sign}(p - t))]| \leq \varepsilon.$$

Thus with probability at least $1 - \delta$, all estimates are ε -accurate. Then, by [Claim 6](#), we have that with large constant probability over the randomness of the set Θ , there exists a convex combination q^* of the points $\{q_t : t \in \Theta\}$ that satisfies the linear constraints

- $q^*[1] \leq \mathbb{E}_{t \sim [-1, 1]}[\text{err}_{\mathcal{D}}(\text{sign}(p - t))] + O(\varepsilon) \leq \text{opt} + C\varepsilon$
- $q^*[2] \leq 200r + C\varepsilon$
- $q^*[3] \leq C\varepsilon$,

for a sufficiently large constant C . That linear combination is the equal-weighted mixture $\mathbb{E}_{t \in \Theta}[q_t]$. By [Theorem 18](#), since the q_t 's are points in $\mathbb{R}^3$, it is possible to write q^* as a convex combination of four members of $\{q_t : t \in \Theta\}$. Thus, for some 4-tuple in Q^4 , the linear program is feasible. `COMPUTEROUNDINGTHRESHOLDS` searches every possible tuple, guaranteeing that a solution is returned.

By repeating $\log(1/\delta)$ times with independent draws of Θ , the large constant success probability is boosted to $1 - \delta$. $\square$

3.4.3 Randomized Set Partitioning

In this section, we will prove the correctness of `COMPUTECLASSIFIER`. As explained in the previous section, the output of `COMPUTERANDOMTHRESHOLDS` is a list of four weights and four thresholds for rounding. The goal is for a small collection of sets — the sets of points misclassified by each threshold and the sets of points for which `ROBUSTNESSLCA` guarantees robustness — to each be partitioned such that w_i of their mass falls into the i^{th} part for each i .

The partition is a set of intervals along a unit vector chosen uniformly from the subspace orthogonal to the estimated mean vectors of the sets we want to partition. The intervals are chosen to have mass $w_1, \dots, w_4$ under the one-dimensional standard Gaussian. We will argue that with high probability over the choice of the random unit vector, all of our sets will be approximately normally distributed in their projections on this vector, and thus the intervals will contain the right proportion of their mass. The key fact underlying this argument comes from the following theorem of [\[43\]](#):

Algorithm 5 COMPUTECLASSIFIER($\mathcal{D}, p, r, \varepsilon, \delta$):

- 1: **Input:** sample access to $\mathcal{D}$, polynomial p ,
 - 2: robustness radius r , error bound ε , confidence parameter δ
 - 3: **Uses:** local noise sensitivity approximator $\hat{\phi}$. (See [Definition 13](#))
 - 4: **Output:** Classifier $h : \mathbb{R}^d \rightarrow \{\pm 1\}$.
 - 5: $(t_1, t_2, t_3, t_4, w_1, w_2, w_3, w_4) \leftarrow \text{COMPUTEROUNDINGTHRESHOLDS}(\mathcal{D}, p, r, \varepsilon, \delta)$.
(See [Algorithm 4](#)).
 - 6: $S_{\text{test}} \leftarrow C \log^2(1/\delta)/\varepsilon^2$ samples from $\mathcal{D}$ (for sufficiently large constant C)
 - 7: **for** $i \in [\log(1/\delta)]$ **do**
 - 8: $M \leftarrow 10d^3$ i.i.d. samples (x, y) from $\mathcal{D}$
 - 9: For $i \in \{1, \dots, 4\}$, define the Boolean functions
 $h_i(x) = \text{ROBUSTNESSLCA}(x, \text{sign}(p - t_i), r)$
 $\text{RobustIndicator}_i(x) = \mathbb{1}[\hat{\phi}_{\text{sign}(p-t_i), 10r}(x) \leq 0.1]$
 - 10: For each $i \in \{1, \dots, 4\}$, let
 $\hat{\mu}_i \leftarrow \mathbb{E}_{(x,y) \in M} \mathbb{1}[y \neq h_i(x)]$
 $\hat{\mu}'_i \leftarrow \mathbb{E}_{(x,y) \in M} \text{RobustIndicator}_i(x)$.
 - 11: Obtain an orthonormal collection of vectors $\{u_1, \dots, u_{d/2}\}$ orthogonal to
 $\text{span}(\mu_1, \dots, \mu_4, \mu'_1, \dots, \mu'_4)$.
 - 12: Generate a uniformly random unit vector u^* in $\text{span}(u_1, \dots, u_{d/2})$.
 - 13: Compute a partition of the real line into intervals $J_1, \dots, J_4$ of Gaussian mass
 $w_1, \dots, w_4$ respectively.
 - 14: Check if the following hold for each $i \in [4]$:
 $\mathbb{E}_{(x,y) \sim S_{\text{test}}} [\mathbb{1}[y \neq h_i(x)] \wedge \langle x, u^* \rangle \in J_i]$
 $= \mathbb{E}_{(x,y) \sim S_{\text{test}}} [\mathbb{1}[y \neq h_i(x)]] \cdot w_i \pm C'\varepsilon$
(for sufficiently large constant C').
 $\mathbb{E}_{(x,y) \sim S_{\text{test}}} [\text{RobustIndicator}_i(x) \wedge \langle x, u^* \rangle \in J_i]$
 $= \mathbb{E}_{x \sim S_{\text{test}}} [\text{RobustIndicator}_i(x)] \cdot w_i \pm C'\varepsilon$
 - 15: If so, **return** the function h defined by
 $h(x) = \sum_{i=1}^4 h_i(x) \cdot \mathbb{1}[\langle u^*, x \rangle \in J_i]$
 - 16: **return** $\perp$
-

Theorem 19 (Concentration of random projections: special case of Theorem 11 of [43]). *Let D be a distribution over $\mathbb{R}^d$ with mean zero and covariance bounded by $\lambda_{\max}$ in operator norm. Assume also that D is supported only on points x satisfying $\|x\|_2 \geq \nu\sqrt{d}$. For a unit vector v , let D_v denote the distribution of $\langle x, v \rangle$, $x \sim D$, and let $D_{\|\cdot\|}$ denote the distribution of $\|x\|_2 / \sqrt{d}$, $x \sim D$. With probability at least*

$$1 - \frac{\lambda_{\max} \ln(1/\varepsilon)}{\varepsilon^3 \nu^2} \cdot \exp \left(-\Omega \left(\frac{\varepsilon^4 \nu^2 d}{\lambda_{\max} \ln(1/\varepsilon)} \right) \right)$$

over a uniformly random unit vector v , the following holds for all intervals J :

$$\left| \Pr_{z \sim D_v} [z \in J] - \Pr_{\substack{\sigma \sim D_{\|\cdot\|} \\ z \sim \mathcal{N}(0, \sigma^2)}} [z \in J] \right| \leq \varepsilon.$$

We apply this theorem to the distributions induced by a set we aim to partition. This is our underlying distribution $\mathcal{D}$ conditioned on a (potentially randomized) indicator, such as the indicator of a point being misclassified by a hypothesis; we will represent that indicator as $r(x)$. After projecting on the subspace orthogonal to our mean estimates, these distributions will still not have mean exactly zero, so we incorporate an error term into our analysis to account for that. The resulting claim is the following:

Claim 7 (Masses of intervals under random projections). *Let $r(x)$ be a randomized Boolean function over $\mathbb{R}^d$, P be a $d/2$ -dimensional linear subspace in $\mathbb{R}^d$, and $J \subseteq \mathbb{R}$ be an interval. Let $\mathcal{P}$ be a $d/2 \times d$ matrix whose rows are an orthonormal basis for P . Assume the following hold for some $\beta_1, \beta_2 \in (0, 1)$ and $\beta_3 \in ((d/2)^{-1/2}, 1)$:*

- **Bounded mean:** $\|\mathcal{P} \mathbb{E}_{x \sim \mathcal{D}}[r(x)x]\|_2 \leq \beta_1$.
- **Thin-shell support:** Every x for which $r(x)$ can be nonzero satisfies $\frac{\|\mathcal{P}x\|_2}{\sqrt{d/2}} = 1 \pm \beta_2$.
- **Large support:** $\mathbb{E}_{x \sim \mathcal{D}}[r(x)] \geq \beta_3$.

For unit vector v , let $\mathcal{D}_v^r$ denote the distribution of $\langle x, v \rangle$, $x \sim \mathcal{D}$ conditioned on $r(x) = 1$. With probability at least

$$1 - \frac{(1 + \beta_1/\beta_3) \ln(1/\varepsilon)}{\varepsilon^3 \beta_3 (1 - 2\beta_2 - 2\beta_1)} \cdot \exp \left(-\Omega \left(\frac{\varepsilon^4 \beta_3 (1 - 2\beta_2 - 2\beta_1) d}{\ln(1/\varepsilon) (1 + \beta_1/\beta_3)} \right) \right)$$

over v drawn uniformly from unit vectors in P , we have

$$\left| \Pr_{z \sim \mathcal{D}_v^r} [z \in J] - \Pr_{z \sim \mathcal{N}(0, 1)} [z \in J] \right| \leq O(\beta_1/\beta_3 + \beta_2 + \varepsilon).$$

Proof. Let $\mu := \mathbb{E}_{x \sim \mathcal{D}, r}[x \mid r(x) = 1]$. We will define $D_{\mu, \mathcal{P}}^r$ to be the distribution of $\mathcal{P}(x - \mu)$ and $\mathcal{D}_{\|\cdot\|}$ to be the distribution of $\|\mathcal{P}(x - \mu)\|_2 / \sqrt{d/2}$, where x is drawn from $\mathcal{D}$ conditioned on $r(x) = 1$.

We aim to apply [Theorem 19](#) to the zero-mean distribution $\mathcal{D}_{\mu, \mathcal{P}}^r$ in the $d/2$ dimensional space P , and find the appropriate bounds for ν and $\lambda_{\max}$.

- **Bounding ν :** By the thin-shell assumption, every x for which $r(x)$ can be nonzero satisfies $\|\mathcal{P}x\|_2 \geq \sqrt{d/2}(1 - \beta_2)$. Thus for any such x , we have $\|\mathcal{P}(x - \mu)\|_2 \geq \sqrt{d/2}(1 - \beta_2) - \|\mathcal{P}\mu\|_2$ by triangle inequality. Now we just need to bound $\|\mathcal{P}\mu\|_2$:

$$\begin{aligned} \|\mathcal{P}\mu\|_2 &= \left\| \mathcal{P} \mathbb{E}_{x, r}[x \mid r(x) = 1] \right\|_2 \\ &= \left\| \mathcal{P} \int_{x \in \mathbb{R}^n} x \cdot \frac{\mathcal{D}(x) \Pr_r[r(x) = 1]}{\Pr_{x, r}[r(x) = 1]} dx \right\|_2 \\ &= \left\| \mathcal{P} \frac{\mathbb{E}_{x, r}[r(x)x]}{\mathbb{E}_{x, r}[r(x)]} \right\|_2 \\ &= \frac{\|\mathcal{P} \mathbb{E}_{x, r}[r(x)x]\|_2}{\mathbb{E}_{x, r}[r(x)]} \leq \frac{\beta_1}{\beta_3} \end{aligned}$$

by the bounded-mean and large support assumptions. This gives us $\nu \geq 1 - \beta_2 - \frac{\beta_1}{\beta_3 \sqrt{d/2}}$, and thus $\nu^2 \geq 1 - 2\beta_2 - 2\frac{\beta_1}{\beta_3 \sqrt{d/2}} \geq 1 - 2\beta_2 - 2\beta_1$, by the assumption that $\beta_3 \geq 1/\sqrt{d/2}$.

- **Bounding $\lambda_{\max}$:** We want a spectral bound on $\mathbb{E}_{x \sim \mathcal{D}_{\mu, \mathcal{P}}^r}[xx^T]$. We have:

$$\begin{aligned} \mathbb{E}_{x \sim \mathcal{D}_{\mu, \mathcal{P}}^r}[xx^T] &= \frac{\mathbb{E}_{x \sim \mathcal{D}, r}[r(x)(\mathcal{P}(x - \mu))(\mathcal{P}(x - \mu))^T]}{\mathbb{E}_{x, r}[r(x)]} \\ &\leq \frac{1}{\beta_3} \mathcal{P} \mathbb{E}_{x \sim \mathcal{D}}[(xx^T + \mu\mu^T - x\mu^T - \mu x^T)] \mathcal{P}^T \end{aligned}$$

Since $\mathcal{D}$ has mean zero, $\mathcal{P}$ and $\mathbb{E}[xx^T]$ has operator norm 1, and $\mathcal{P}\mu\mu^T\mathcal{P}^T$ has operator norm $\leq \|\mathcal{P}\mu\|_2$, we have

$$\lambda_{\max} = \left\| \mathbb{E}_{x \sim \mathcal{D}_{\mu, \mathcal{P}}^r}[xx^T] \right\|_{op} \leq \frac{1 + \|\mathcal{P}\mu\|_2}{\beta_3} \leq \frac{1 + \beta_1/\beta_3}{\beta_3}.$$

Substituting these parameters into [Theorem 19](#) gives us the claimed failure probability bound. The condition that holds with high probability is that for all intervals J , we have

$$\left| \Pr_{z \sim (\mathcal{D}_{\mu, \mathcal{P}}^r)_v}[z \in J] - \Pr_{\substack{\sigma \sim \mathcal{D}_{\|\cdot\|} \\ z \sim \mathcal{N}(0, \sigma^2)}}[z \in J] \right| \leq \varepsilon.$$

By the thin-shell assumption, the bound on $\|\mathcal{P}\mu\|_2$, and the triangle inequality, $\mathcal{D}_{\|\cdot\|}$ is supported on $[1 - \beta_1/\beta_3 - \beta_2, 1 + \beta_1/\beta_3 + \beta_2]$. Then by the TV distance bound for Gaussians, we have for any σ in the support of $\mathcal{D}_{\|\cdot\|}$:

$$d_{TV}(\mathcal{N}(0, 1), \mathcal{N}(0, \sigma^2)) \leq \frac{3|1 - \sigma^2|}{2} \leq O(\beta_1/\beta_3 + \beta_2),$$

so we have

$$\left| \Pr_{z \sim (\mathcal{D}_{\mu, \mathcal{P}}^r)_v} [z \in J] - \Pr_{z \sim \mathcal{N}(0, 1)} [z \in J] \right| \leq O(\varepsilon + \beta_1/\beta_3 + \beta_2).$$

We now substitute the definition of $(\mathcal{D}_{\mu, \mathcal{P}}^r)_v$ and use the fact that v is in P .

$$\begin{aligned} \Pr_{z \sim (\mathcal{D}_{\mu, \mathcal{P}}^r)_v} [z \in J] &= \Pr_{x \sim \mathcal{D}^r} [\langle \mathcal{P}(x - \mu), v \rangle \in J] \\ &= \Pr_{x \sim \mathcal{D}^r} [\langle x, v \rangle + \langle \mu, v \rangle \in J] \\ &= \Pr_{y \sim \mathcal{D}_v^r} [y + \langle \mu, v \rangle \in J]. \end{aligned}$$

So we have

$$\left| \Pr_{z \sim \mathcal{D}_v^r} [z + \langle \mu, v \rangle \in J] - \Pr_{z \sim \mathcal{N}(0, 1)} [z \in J] \right| \leq O(\varepsilon + \beta_1/\beta_3 + \beta_2).$$

By applying the uniform convergence property of [Theorem 19](#) for all intervals to shift J , we have

$$\left| \Pr_{z \sim \mathcal{D}_v^r} [z \in J] - \Pr_{z \sim \mathcal{N}(0, 1)} [z - \langle \mu, v \rangle \in J] \right| \leq O(\varepsilon + \beta_1/\beta_3 + \beta_2).$$

By anticoncentration of the Gaussian distribution and the fact that $|\langle \mu, v \rangle| \leq \beta_1/\beta_3$, we have

$$\Pr_{z \sim \mathcal{N}(0, 1)} [z - \langle \mu, v \rangle \in J] = \Pr_{z \sim \mathcal{N}(0, 1)} [z \in J] \pm 2\beta_1/\beta_3.$$

Combining everything, we have

$$\left| \Pr_{z \sim \mathcal{D}_v^r} [z \in J] - \Pr_{z \sim \mathcal{N}(0, 1)} [z \in J] \right| \leq O(\varepsilon + \beta_1/\beta_3 + \beta_2)$$

with probability at least

$$1 - \frac{(1 + \beta_1/\beta_3) \ln(1/\varepsilon)}{\varepsilon^3 \beta_3 (1 - 2\beta_2 - 2\beta_1)} \cdot \exp \left(-\Omega \left(\frac{\varepsilon^4 \beta_3 (1 - 2\beta_2 - 2\beta_1) d}{\ln(1/\varepsilon) (1 + \beta_1/\beta_3)} \right) \right)$$

as desired. $\square$

We now argue that for r indicating a set of large Gaussian volume and $\mathcal{P}$ being orthogonal to a good approximation to the mean of r , the assumptions of [Claim 7](#) are satisfied with small values of β .

The following facts are relevant to this proof.

Fact 17 (Thin-shell concentration of log-concave variables [73]). *Let X be an isotropic random vector with log-concave density in $\mathbb{R}^d$. Then there are universal constants $c, C > 0$ such that for all $t \geq 0$,*

$$\Pr \left[\left| \|X\|_2 - \sqrt{d} \right| \geq t \right] \leq C \exp(-cd^{1/2} \min((t/\sqrt{d})^3, t/\sqrt{d})).$$

Corollary 5. *Let X be an isotropic random vector with log-concave density in $\mathbb{R}^d$. Then there is a universal constant C such that for any $\varepsilon > 0$,*

$$\Pr \left[\left| \|X\|_2 - \sqrt{d} \right| \geq C(d \log(1/\varepsilon))^{1/3} \right] \leq \varepsilon.$$

Fact 18 (Thin-shell concentration of Gaussian variables (standard fact)). *Let X be a standard Gaussian in $\mathbb{R}^d$. Then*

$$\Pr \left[\left| \|X\|_2 - \sqrt{d} \right| \geq t \right] \leq 2 \exp(-t/4).$$

Claim 8 (Log-concave distributions satisfy assumptions of Claim 7). *Let $\mathcal{D}$ be an isotropic log-concave distribution over $\mathbb{R}^d$ and K be a sufficiently large constant depending on those given by Fact 17. Let r be a randomized Boolean function such that $\mathbb{E}_{x \sim \mathcal{D}}[r(x)] \geq \beta$, and let $\hat{\mu}(r)$ be such that for all $i \in [d]$,*

$$\left| \hat{\mu}(r)_i - \left(\mathbb{E}_{x \sim \mathcal{D}}[r(x)x] \right)_i \right| \leq 1/d.$$

Let P be a $d/2$ -dimensional linear subspace orthogonal to $\hat{\mu}(r)$, and $\mathcal{P}$ be a $d/2 \times d$ matrix whose rows are an orthonormal basis for P . Let r_{trunc} be the truncation of r defined as

$$r_{\text{trunc}}(x) := r(x) \wedge \mathbb{1} \left[\frac{\|\mathcal{P}x\|_2}{\sqrt{d/2}} = 1 \pm \frac{10Kd^{1/3} \ln d}{\sqrt{d/2}} \right].$$

Then the following conditions hold:

- **Bounded mean:** $\|\mathcal{P} \mathbb{E}_{x \sim \mathcal{D}}[r_{\text{trunc}}(x)x]\|_2 \leq O(\sqrt{1/d})$.
- **Thin-shell support:** *Every x for which $r_{\text{trunc}}(x)$ can be nonzero satisfies $\frac{\|\mathcal{P}x\|_2}{\sqrt{d/2}} = 1 \pm \frac{10Kd^{1/3} \ln d}{\sqrt{d/2}}$.*
- **Large support:** $\mathbb{E}_{x \sim \mathcal{D}}[r_{\text{trunc}}(x)] \geq \beta - O(1/d)$.

Furthermore, $\Pr_{x \sim \mathcal{D}}[r(x) \neq r_{\text{trunc}}(x)] \leq O(1/d)$.

Proof. First we will prove the final inequality,

$$\Pr_{x \sim \mathcal{D}}[r(x) \neq r_{\text{trunc}}(x)] \leq O(1/d).$$

Since the projection of $\mathcal{D}$ by $\mathcal{P}$ is a log-concave distribution over $\mathbb{R}^{d/2}$ (Fact 13), this inequality follows from Corollary 5 applied to $\mathcal{P}(\mathcal{D})$ with $\varepsilon = 1/d$. We now prove the other conditions.

- **Bounded mean:** Let

$$\mu(r_{\text{trunc}}) := \mathbb{E}_{x \sim \mathcal{D}} [r_{\text{trunc}}(x)x] \quad \text{and} \quad \mu(r) := \mathbb{E}_{x \sim \mathcal{D}} [r(x)x].$$

First we claim that for every i , $|\mu(r_{\text{trunc}})_i - \mu(r)_i| \leq 1/d$. We have

$$\begin{aligned} \mu(r)_i &= \mu(r_{\text{trunc}})_i + \mathbb{E}[r(x)x_i \cdot \mathbf{1}[r(x) \neq r_{\text{trunc}}(x)]] \\ &\leq \mu(r_{\text{trunc}})_i + \mathbb{E}[\|x\| \cdot \mathbf{1}[r(x) \neq r_{\text{trunc}}(x)]] \\ &\leq \mu(r_{\text{trunc}})_i + \int_{t=10Kd^{1/3} \ln d}^{\infty} t \cdot C \exp(-ct^3/d) dt \\ &\leq \mu(r_{\text{trunc}})_i + C \int_{t=10Kd^{1/3} \ln d}^{\infty} t^3 \cdot \exp(-ct^3/d) dt \\ &\leq \mu(r_{\text{trunc}})_i + C \int_{u=(10K \ln d)^3 d}^{\infty} u \cdot \exp(-cu/d) du \\ &\leq \mu(r_{\text{trunc}})_i + \exp(-(10K \ln d)^3 \cdot c) \cdot ((10K \ln d)^3 \cdot c + 1) \cdot (d/c)^2 \\ &\leq \mu(r_{\text{trunc}})_i + 1/d. \quad (\text{by setting } K \text{ sufficiently large}) \end{aligned}$$

A symmetric argument lower bounds $\mu(r)_i$ by $\mu(r_{\text{trunc}})_i - 1/d$. Now we bound $\|\mathcal{P}\mu(r_{\text{trunc}})\|_2$. We have $\|\mathcal{P}\mu(r_{\text{trunc}})\|_2 \leq \sup_{u \in P, \|u\|_2=1} \langle u, \mu(r_{\text{trunc}}) \rangle$. Consider a unit vector $u \in P$. We have

$$\begin{aligned} \langle u, \mu(r_{\text{trunc}}) \rangle &= \langle u, \mu(r_{\text{trunc}}) - \mu(r) \rangle + \langle u, \mu(r) - \hat{\mu}(r) \rangle + \langle u, \hat{\mu}(r) \rangle \\ &\leq \sum_{i \in [d]} \left| \frac{u_i}{d} \right| + \sum_{i \in [d]} \left| \frac{u_i}{d} \right| + 0 \\ &\leq 2/\sqrt{d}. \end{aligned}$$

- **Thin-shell support:** This follows immediately from the definition of r_{trunc} .
- **Large support:** This follows immediately from the assumption that $\mathbb{E}_{x \sim \mathcal{N}(0, I_d)}[r(x)] \geq \beta$ and the fact that $\Pr_{x \sim \mathcal{N}(0, I_d)}[r(x) \neq r_{\text{trunc}}(x)] \leq O(1/d)$.

□

Claim 9 (Accuracy of the mean estimates). *Let S be a set of $10d^3$ points drawn from a distribution $\mathcal{D}$ with covariance I_d . Let $\mathcal{F}$ be a collection of eight (possibly randomized) Boolean functions over $\mathbb{R}^d$. Then with probability at least $1 - O(d^{-3})$, the following holds for all $f \in \mathcal{F}$ and $i \in [d]$:*

$$\left| \frac{1}{T} \sum_{x \in T} x_i f(x) - \mathbb{E}_{x \sim \mathcal{D}} x_i f(x) \right| \leq 1/d.$$

Proof. By the fact that $\mathcal{D}$ has covariance I_d , we have $\text{Var}(x_i) = 1$ and $\text{Var}(\frac{1}{|T|} \sum x_i) = 1/|T|$, so by Chebyshev's inequality

$$\Pr_T \left[\left| \mathbb{E}_{x \sim \mathcal{D}} [x_i f(x)] - \frac{1}{|T|} \sum_{x \in T} x_i f(x) \right| > \frac{10d^2}{|T|} \right] \leq 0.01d^{-4}.$$

Set $|T| := 10d^3$; then the deviation becomes $\frac{10d^2}{|T|} = \frac{1}{d}$. Union bounding over $\mathcal{F}$ and $i \in [d]$, we have with probability at least $1 - O(d^{-3})$, all f and all i satisfy

$$\left| \frac{1}{|T|} \sum_{x \in T} x_i f(x) - \mathbb{E}_{x \sim \mathcal{D}} x_i f(x) \right| \leq 1/d.$$

□

Claim 10 (Accurate partitioning of (randomized) sets). *Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{-1, 1\}$ such that the $\mathbb{R}^d$ -marginal is isotropic log-concave. Let $\varepsilon > d^{-1/7}$. Let $\mathcal{F}$ be the set of eight (possibly randomized) Boolean functions over $\mathbb{R}^d$ given by RobustIndicator_i and $\text{ErrorIndicator}_i(x) := \mathbb{1}[y \neq h_i(x)]$ for $i \in [4]$. Let $w_1, \dots, w_4$ be the weights returned by $\text{COMPUTEROUNDINGTHRESHOLDS}$ and $J_1, \dots, J_4$ be the intervals generated by $\text{COMPUTECCLASSIFIER}$. Then with probability at least $1 - O(\delta)$, the following holds for all $f \in \mathcal{F}$ and all $i \in [4]$:*

$$\Pr_{x \sim \mathcal{D}} [\langle x, u \rangle \in J_i \wedge f(x) = 1] = \Pr_{x \sim \mathcal{D}} [f(x) = 1] \cdot w_i \pm O(\varepsilon).$$

Proof. First we will claim that each iteration succeeds with probability $1 - 1/\text{poly}(n)$. Consider only the functions $f \in \mathcal{F}$ such that $\Pr[f(x) = 1] \geq \varepsilon$; for the others, the statement holds trivially. $\text{COMPUTECCLASSIFIER}$ estimates the means $\mu_i, \mu'_i; i \in [4]$ of the functions from its sample of size $10d^3$, then sets P to be a linear subspace orthogonal to all of these. By [Claim 9](#) we have that with probability $1 - O(d^{-3})$ the mean-accuracy assumption of [Claim 8](#) is satisfied. Thus, by [Claim 8](#), the assumptions of [Claim 7](#) are satisfied with the following parameters:

- $\beta_1 = O(\sqrt{1/d})$
- $\beta_2 = O(d^{-1/6} \ln d)$
- $\beta_3 = \varepsilon - O(1/d)$.

$\text{COMPUTECCLASSIFIER}$ then generates a uniform random unit vector $u \in P$ and four intervals $J_1 \dots J_4$ of Gaussian volume $w_1, \dots, w_4$. By [Claim 7](#), with high probability over u , the following holds for each J_i :

$$\Pr_{x \sim \mathcal{D} | f(x)=1} [\langle x, u \rangle \in J_i] = w_i \pm O(\beta_1/\beta_3 + \beta_2 + \varepsilon) = w_i \pm O(\varepsilon);$$

taking the conjunction with the event that $f(x) = 1$ gives us the desired

$$\Pr_{x \sim \mathcal{D}}[\langle x, u \rangle \in J_i \wedge f(x) = 1] = \Pr_{x \sim \mathcal{D}}[f(x) = 1] \cdot (w_i \pm O(\varepsilon)) \leq \Pr_{x \sim \mathcal{D}}[f(x) = 1] \cdot w_i \pm O(\varepsilon).$$

The failure probability is

$$\begin{aligned} & \frac{(1 + \beta_1/\beta_3) \ln(1/\varepsilon)}{\varepsilon^3 \beta_3 (1 - 2\beta_2 - 2\beta_1)} \cdot \exp \left(-\Omega \left(\frac{\varepsilon^4 \beta_3 (1 - 2\beta_2 - 2\beta_1) d}{\ln(1/\varepsilon) (1 + \beta_1/\beta_3)} \right) \right) \\ & \frac{(1 + o(1)) \ln(1/\varepsilon)}{\varepsilon^4 (1 - o(1))} \cdot \exp \left(-\Omega \left(\frac{\varepsilon^5 (1 - o(1)) d}{\ln(1/\varepsilon) (1 + o(1))} \right) \right) \\ & \frac{\ln(1/\varepsilon)}{\varepsilon^4} \cdot \exp \left(-\Omega \left(\frac{\varepsilon^5 d}{\ln(1/\varepsilon)} \right) \right) \\ & 1/\varepsilon^5 \cdot \exp(-\Omega(\varepsilon^6 d)) \end{aligned}$$

By our assumption that $\varepsilon > \Omega(d^{-1/7})$, this term is dominated by $\exp(-\Omega(d^{1/7}))$. The total failure probability is dominated by the $O(d^{-3})$ probability of failure for the mean estimation. Thus, we have that with probability $\geq 1 - 1/\text{poly}(d)$, no bad tail events occur, and the guarantee holds for all $f \in \mathcal{F}$ and $i \in [4]$.

The following boosting argument is relevant only when δ is smaller than this $\approx d^{-3}$ failure probability. In each iteration we test the accuracy of the partition by comparing

$$\Pr[\langle x, u \rangle \in J_i \wedge f(x) = 1] \quad \text{to} \quad \Pr[f(x) = 1] \cdot w_i$$

for each function f . It suffices for these estimates to be accurate up to ε additive error. By a Chernoff bound, for a test set of size $\Omega(\log(1/\delta)/\varepsilon^2)$, all estimates are ε -accurate with probability at least $1 - \delta^2$, so that after union bounding over all iterations the estimates are still accurate with probability at least $1 - \delta$. Since $d \geq 2$, the probability that a good partition is not found in any of the $\log(1/\delta)$ independent attempts is at most $2^{-\log(1/\delta)} = \delta$. Thus, the total failure probability is at most 2δ . $\square$

3.4.4 Correctness and Complexity of Algorithm 5

Now we will finish the analysis of COMPUTECLASSIFIER, restated here for convenience.

Theorem 12 (Correctness of Algorithm 5, COMPUTECLASSIFIER). *Let $\varepsilon \geq d^{-1/7}$ and $\delta, r > 0$. Let $\hat{\phi}$ be an ε -accurate local noise sensitivity approximator for degree- k PTFs over $\mathbb{R}^d$. Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{-1, 1\}$ such that the $\mathbb{R}^d$ -marginal is isotropic and log-concave. Let $p : \mathbb{R}^d \rightarrow \mathbb{R}$ be a polynomial of degree at most k satisfying*

$$\mathbb{E}_{t \sim [-1, 1]} \left[\mathbb{E}_{(x, y) \sim \mathcal{D}} [\phi_{\text{sign}(p-t), 10r}(x)] \right] \leq 100r + O(\varepsilon), \quad (3.30)$$

$$\Pr_{(x, y) \sim \mathcal{D}} \left[\mathbb{E}_{t \sim [-1, 1]} [\phi_{\text{sign}(p(x)-t), 10r}(x)] \geq 0.7 \right] \leq O(\varepsilon). \quad (3.31)$$

Then the algorithm $\text{COMPUTECLASSIFIER}(\mathcal{D}, p, r, \varepsilon, \delta)$, given sample access to $\mathcal{D}$ and query access to $\hat{\phi}$, outputs a hypothesis $h : \mathbb{R}^d \rightarrow \{-1, 1\}$ such that the following properties hold with probability at least $1 - O(\delta)$:

$$\text{err}_{\mathcal{D}}(h) \leq \mathbb{E}_{t \sim [-1, 1]} [\text{err}_{\mathcal{D}}(\text{sign}(p - t))] + O(\varepsilon)$$

$$\text{Boundary}_{\mathcal{D}, r}(h) \leq O(r + \varepsilon).$$

The running time and number of queries to $\hat{\phi}$ are $\text{poly}(d^k \cdot 1/\varepsilon \cdot \log(1/\delta))$.

Proof of Theorem 12. By Theorem 16 and the triangle inequality, we have that all $i \in [4]$ satisfy

- (i) $\Pr_{(x, y) \sim \mathcal{D}}[h_i(x) \neq y] \leq \text{err}_{\mathcal{D}}(\text{sign}(p - t_i)) + \widehat{\text{iso}}_{\mathcal{D}, 10r}(\text{sign}(p - t_i), 0.8)$
- (ii) $\mathbb{E}_{x \sim \mathcal{D}}[\text{RobustIndicator}_i(x)] \geq 1 - O(\widehat{\text{NS}}_{\mathcal{D}, 10r}(\text{sign}(p - t_i)))$
- (iii) For every x such that $\text{RobustIndicator}_i(x) = 1$ and $x' : \|x - x'\| \leq r$, we have $h_i(x') = h_i(x)$.

First we will handle the error condition. By Theorem 17, we have that with probability $1 - O(\delta)$,

$$\begin{aligned} \sum_{i \in [4]} w_i \cdot (\text{err}_{\mathcal{D}}(\text{sign}(p - t_i)) + \widehat{\text{iso}}_{\mathcal{D}, 10r}(\text{sign}(p - t_i), 0.8)) \\ \leq \mathbb{E}_{t \sim [-1, 1]} [\text{err}_{\mathcal{D}}(\text{sign}(p - t))] + O(\varepsilon). \end{aligned}$$

Thus, we have

$$\sum_{i \in [4]} w_i \cdot \Pr_{(x, y) \sim \mathcal{D}}[h_i(x) \neq y] \leq \mathbb{E}_{t \sim [-1, 1]} [\text{err}_{\mathcal{D}}(\text{sign}(p - t))] + O(\varepsilon).$$

By Claim 10, we then have with probability $\geq 1 - O(\delta)$,

$$\sum_{i \in [4]} \Pr_{(x, y) \sim \mathcal{D}}[h_i(x) \neq y \wedge \langle x, u^* \rangle \in J_i] \leq \mathbb{E}_{t \sim [-1, 1]} [\text{err}_{\mathcal{D}}(\text{sign}(p - t))] + O(\varepsilon).$$

Therefore, applying the definition of h , we have

$$\text{err}_{\mathcal{D}}(h) \leq \mathbb{E}_{t \sim [-1, 1]} [\text{err}_{\mathcal{D}}(\text{sign}(p - t))] + O(\varepsilon).$$

Now we analyze the robustness condition. By Theorem 17, we have that with probability $\geq 1 - O(\delta)$,

$$\sum_{i \in [4]} w_i \cdot \widehat{\text{NS}}_{\mathcal{D}, 10r}(\text{sign}(p - t_i)) \leq O(r + \varepsilon).$$

By a Markov bound and the definitions of RobustIndicator and $\widehat{\text{NS}}$, we have the following for each i :

$$\begin{aligned} \mathbb{E}[\text{RobustIndicator}_i(x)] &= \Pr[\hat{\phi}_{\text{sign}(p-t_i)}(x) \leq 0.1] = 1 - \Pr[\hat{\phi}_{\text{sign}(p-t_i)}(x) > 0.9] \\ &= 1 - \Pr\left[\hat{\phi}(x) > 0.9/\widehat{\text{NS}}(\text{sign}(p-t_i)) \cdot \mathbb{E}[\hat{\phi}(x)]\right] \\ &\geq 1 - \frac{10}{9}\widehat{\text{NS}}(\text{sign}(p-t_i)). \end{aligned}$$

Thus we have

$$\begin{aligned} \sum_{i \in [4]} w_i \cdot \mathbb{E}_{\mathcal{D}}[\text{RobustIndicator}_i(x)] &\geq \sum_{i \in [4]} w_i \cdot \left(1 - \frac{10}{9} \mathbb{E}_{\mathcal{D}}[\hat{\phi}_{\text{sign}(p-t_i), 10r}(x)]\right) \\ &\geq 1 - \sum_{i \in [4]} w_i \cdot \frac{10}{9} \mathbb{E}_{\mathcal{D}}[\hat{\phi}_{\text{sign}(p-t_i), 10r}(x)] \\ &\geq 1 - O(r + \varepsilon). \end{aligned}$$

By [Claim 10](#) we then have with probability $\geq 1 - \delta$,

$$\sum_{i \in [4]} \mathbb{E}_{x \sim \mathcal{D}} [\text{RobustIndicator}_i(x) \cdot \mathbb{1}[\langle x, u^\star \rangle \in J_i]] \geq 1 - O(r + \varepsilon).$$

We now claim that for each J_i , all x with $\text{RobustIndicator}_i(x) = 1$ such that $\langle x, u^\star \rangle$ is at least r away from the interval boundary satisfy the adversarial robustness condition

$$\forall x' : \|x - x'\|_2 \leq r \text{ and } h(x) = h(x').$$

Since x is r away from the interval boundary, all x' such that $\|x - x'\|_2 \leq r$ satisfy $\langle x', u^\star \rangle \in J_i$, so they are also labeled by h_i . By [Theorem 16](#), all x with $\text{RobustIndicator}_i(x) = 1$ satisfy the robustness condition in h_i , so, since they are also r away from the boundary, they satisfy the robustness condition in h . Thus we have

$$\begin{aligned} \text{Boundary}_{\mathcal{D}, r}(h) &\geq \sum_{i \in [4]} \left(\mathbb{E}_{x \sim \mathcal{D}} [\text{RobustIndicator}_i(x) \cdot \mathbb{1}[\langle x, u^\star \rangle \in J_i]] \right) \\ &\quad - \Pr_{x \sim \mathcal{D}}[\langle x, u^\star \rangle \text{ within } r \text{ of an interval boundary}] \\ &\geq 1 - O(r + \varepsilon) - O(r), \end{aligned}$$

where the boundary probability is bounded by $O(r)$ due to [Fact 14](#). Thus when all subroutines are successful, the guarantees of the theorem hold. The total success probability is $\geq 1 - O(\delta)$ after union bounding the failure probabilities of all subroutines. This concludes the proof of correctness.

We now analyze the running time and query complexity to $\hat{\phi}$. `COMPUTEROUNDINGTHRESHOLDS` is called once. For $\log(1/\delta)$ iterations, it estimates $\text{err}_t, \text{NS}_t, \text{iso}_t$ for each of $100/\varepsilon^2$ rounding thresholds t ; this step makes $O(\log(1/\delta)/\varepsilon^2)$ queries to $\hat{\phi}$ and takes

$\text{poly}(d^k \cdot 1/\varepsilon \cdot \log(1/\delta))$ additional time, due to the evaluations of the degree- k polynomial p . Then it solves a linear program of constant size for $(100/\varepsilon^2)^4$ iterations. The total running time and query complexity of COMPUTE ROUNDING THRESHOLDS are dominated by the first term, which is $\text{poly}(d^k \cdot 1/\varepsilon \cdot \log(1/\delta))$.

The rest of COMPUTE CLASSIFIER repeats $\log(1/\delta)$ times and does the following each repetition:

- a) evaluate $\mathbb{1}[h_i(x) \neq y]$ and $\text{RobustIndicator}_i(x)$ for each $i \in [4]$ and (x, y) in M ,
- b) obtain a basis orthogonal to the estimated mean vectors, a random unit vector in this space, and a partition of the real line into intervals
- c) check accuracy of the partition with respect to S_{test} .

Each evaluation of $\mathbb{1}[h_i(x) \neq y]$ makes one query to $\hat{\phi}$ and takes $d^{O(k)}$ time, as it simply calls ROBUSTNESSLCA, which makes one evaluation of a degree- k PTF and one call to $\hat{\phi}$. Each evaluation of $\text{RobustIndicator}_i(x)$ makes one query to $\hat{\phi}$ and takes $O(1)$ additional time. Thus items (a) and (c) take $d^{O(k)}$ time and queries. For item (b), obtaining the basis takes $\text{poly}(d)$ time by Gaussian elimination, and approximating the interval boundaries can be done in $\text{poly}(1/\varepsilon)$ time by using a numerical ε -approximation to the error function⁸

$$\text{erf}(t) = \frac{1}{\sqrt{\pi}} \int_0^t e^{-t^2/2}.$$

Overall, the total time and query complexity of COMPUTE CLASSIFIER is $\text{poly}(d^k \cdot 1/\varepsilon \cdot \log(1/\delta))$, as desired. $\square$

3.5 Verifiable Robustness

In this section we prove that under complexity assumptions, the robustness guarantee of our learning algorithm can be made efficiently verifiable in the sense formalized below. The verifier certifies that a point satisfies the robustness condition. We formally state this result:

Corollary 6 (Deterministic verifiability). *If $P = \text{BPP}$, then there is a learning algorithm $\mathcal{B}$ that, given access to labeled samples from a subgaussian isotropic log-concave distribution, runs in time $d^{\tilde{O}(1/\varepsilon^2)} \cdot \log(1/\delta)$ and produces a hypothesis h with the following guarantees:*

- **Agnostic approximation:** *With probability at least $1 - O(\delta)$, $\Pr_{(x,y) \sim \mathcal{D}}[h(x) \neq y] \leq \text{opt} + O(\varepsilon)$, where opt is the misclassification error of the best halfspace.*

⁸Such numerical approximations are standard and take $\text{poly}(1/\varepsilon)$ time. Alternatively, one could use a randomized algorithm that takes $\text{poly}(1/\varepsilon)$ samples from the gaussian and uses them to approximate the relevant values w_i . Note that we presented the w_i 's as exact for conciseness, but since Claim 10 already guarantees only an $O(\varepsilon)$ -accurate partition, we see that an additional ε error in the estimation of the w_i 's is asymptotically irrelevant.

- **Verifiable robustness:** There is a verifier that runs in time $d^{\tilde{O}(1/\varepsilon^2)}$ that takes as input a circuit g and a point $x \in \mathbb{R}^d$ that always rejects if

$$\exists z : \|z\|_2 \leq r \text{ and } g(x) \neq g(x+z).$$

If $g = h$, then with probability at least $1 - O(\delta)$ over the randomness of $\mathcal{B}$, the verifier accepts with probability at least $1 - O(r + \varepsilon)$ over $x \sim D$.

We use the following fact:

Fact 19 (Derandomized estimation of $\hat{\phi}$). *If $P = BPP$, then there exists a deterministic algorithm running in time $d^{O(k)} \cdot \text{poly}(1/\varepsilon)$ that takes as input a degree- k PTF f , a radius r , and an input $x \in \mathbb{R}^d$ and outputs an estimate $\hat{\phi}_{f,10r}(x)$ such that $\hat{\phi}_{f,10r}(x) = \phi_{f,10r}(x) \pm \varepsilon$.*

Proof. Observe that there is a randomized algorithm running in $d^{O(k)}/\varepsilon^2$ time that takes as input a threshold t and decides if $\phi_f(x) \geq t$, succeeding with probability $\geq 2/3$ whenever $\phi_f(x) > t + \varepsilon$ or $\phi_f(x) < t - \varepsilon$. This is the algorithm that samples $O(1/\varepsilon^2)$ points z from $\mathcal{N}(0, I_d)$ and evaluates $f(x + 10rz)$ on each of them to estimate ϕ ; its analysis is a Chernoff bound. If $P = BPP$, then there exists a deterministic algorithm for this decision problem also running in time $d^{O(k)} \cdot \text{poly}(1/\varepsilon)$. Since $\phi_f(x) \in [0, 1]$, we can binary search with $\log(1/\varepsilon)$ iterations to find a t such that $\phi_f(x) \in t \pm \varepsilon$. $\square$

The learning algorithm is simply ROBUSTLEARN, augmented to provide some extra information, with the estimates of $\hat{\phi}$ provided by the deterministic estimator. The verifier checks that the hypothesis matches a “template;” this will prove that the unknown circuit is in fact of the form that ROBUSTLEARN is supposed to return, and for which our correctness analysis holds. Since for any hypothesis matching the template, any point x for which $\phi(x) \leq 0.1$ satisfies the robustness condition, the verifier will then deterministically estimate $\phi(x)$.

We define the template below:

Lemma 3.5.1 (Hypothesis template). *Fix a set B of basis functions for the set of degree- k polynomials over $\mathbb{R}^d$. There is an algorithm COMPILE that takes as input a $|B|$ -length vector $\vec{v}$ of real-valued coefficients, real numbers $t_1, \dots, t_4$, a unit vector $u \in \mathbb{R}^d$, and real numbers $c_1 < c_2 < c_3$. For each $i \in [4]$, let the PTF h_i be defined*

$$h_i(x) = \text{sign} \left(\left(\sum_{b \in B} v_b \cdot b(x) \right) - t_i \right).$$

Let the intervals $J_1, \dots, J_4$ be the partition of the real line induced by c_1, c_2, c_3 . The algorithm outputs a circuit computing the hypothesis

$$h(x) = \sum_{i \in [4]} \text{ROBUSTNESSLCA}(x, h_i, r) \cdot \mathbf{1}[\langle x, u \rangle \in J_i].$$

It also outputs each of the PTFs $h_1, \dots, h_4$. The running time is $d^{O(k)}$.

Proof. Observe that there is a deterministic algorithm running in $d^{O(k)}$ time that takes as input all of the given parameters and the input x , and evaluates $h(x)$, using the deterministic implementation of $\hat{\phi}$ in ROBUSTNESSLCA. Therefore there is a circuit of size $d^{O(k)}$ with the same behavior. The algorithm COMPILE takes this circuit and hardcodes all the input parameters except for x , then outputs the resulting circuit. This takes time $d^{O(k)}$ and the output is a circuit that takes x as input and evaluates $h(x)$. By the same argument, in $d^{O(k)}$ time COMPILE can also output the PTFs $h_1, \dots, h_4$. $\square$

We include pseudocode of the verifier below. For brevity we will refer to the package of data taken as input by COMPILE as **data**.

Algorithm 6 VERIFY($g, x, r, \varepsilon, \mathbf{data}$):

- 1: **Input:** circuit g , point $x \in \mathbb{R}^n$, robustness radius r , error tolerance ε , hypothesis parameters **data**
 - 2: $h, h_1, h_2, h_3, h_4 \leftarrow \text{COMPILE}(\mathbf{data})$
 - 3: **if** $g \neq h$ **then return** reject.
 - 4: Let u be the unit vector in **data** and $J_1, \dots, J_4$ be the partition induced by the interval boundaries c_1, c_2, c_3 in **data**.
 - 5: Let i be the interval such that $\langle x, u \rangle \in J_i$.
 - 6: **if** $\hat{\phi}_{h_i, 10r}(x) > 0.1 - \varepsilon$ **then return** reject.
 - 7: **if** $|\langle x, u \rangle - c_j| \leq r$ for any of the interval boundaries c_1, c_2, c_3 in **data** **then return** reject.
 - 8: **return** accept.
-

Proof of Corollary 6. The learning algorithm $\mathcal{B}$ is ROBUSTLEARN, but with the following modification: it records its parameters in a **data** package, outputs COMPILE(**data**) as its final hypothesis, and outputs **data** as well. By the guarantees of ROBUSTLEARN, with probability at least $1 - O(\delta)$, h satisfies the agnostic approximation guarantee. When VERIFY is called on the output h of $\mathcal{B}$, since it is the output of COMPILE, it always passes the first check. By inspecting the proof of Theorem 16, we see that in fact robustness holds for all x such that $\hat{\phi}_{h_i}(x) \leq 0.1 - \varepsilon$ and x is at least r distance from each of the interval boundaries. By Fact 19, we have $|\hat{\phi}_{h_i}(x) - \phi_{h_i}(x)| \leq \varepsilon$, thus the verifier rejects all points such that $\phi_{h_i}(x) > 0.1$. Since it also rejects if x is within r of a boundary, all accepted points satisfy the robustness condition. Furthermore, by inspecting the proofs of Theorem 12 and Theorem 16, we see that whenever ROBUSTLEARN succeeds (probability $1 - O(\delta)$), we have that at least $1 - O(r + \varepsilon)$ fraction of points $x \sim \mathcal{D}$ satisfy $\hat{\phi}_{h_i}(x) \leq 0.1 - \varepsilon$ and are at least r away from any boundary, and are thus verifiably robust. Thus, the verifiable robustness guarantee holds.

The running time of COMPILE for PTFs of degree $\tilde{O}(1/\varepsilon^2)$ is $d^{\tilde{O}(1/\varepsilon^2)}$ (Lemma 3.5.1). The running time of VERIFY is this plus the running time of the deterministic estimator for $\hat{\phi}$, which is also $d^{\tilde{O}(1/\varepsilon^2)}$ (Fact 19). Thus the total running time is $d^{\tilde{O}(1/\varepsilon^2)}$. $\square$

Remark 20. We note that without the assumption that $\mathsf{P} = \mathsf{BPP}$, there is a randomized analogue of Corollary 6 where the user compiles the final hypothesis, rather than verifying

the one provided by ROBUSTLEARN. Rather than outputting a circuit h , ROBUSTLEARN can just output **data**, and the user can COMPILE it with a randomized implementation of ROBUSTNESSLCA in time $d^{\tilde{O}(1/\varepsilon^2)} \cdot \log(1/\delta)$, in which case the soundness of the verifier holds with probability $1 - \delta$ for the compiled hypothesis.

Chapter 4

Agnostically Learning Monotone Functions

In this chapter, we present our algorithm for agnostically learning monotone functions, as well as the local monotonicity corrector that it is built on.

4.1 Chapter-Specific Preliminaries

Definition 17 (ℓ_0 distance for functions on a poset). *For a pair of functions $f_1, f_2 : P \rightarrow \{0, 1\}$, the **distance** $\|f_1 - f_2\|$ is the fraction of elements x in P on which $f_1(x) \neq f_2(x)$.*

Definition 18 (Graph distance and height). *We will write $\text{dist}(x, y)$ to denote the length of the **longest** directed path¹ between x and y in the Hasse diagram of P . The **height** of a poset P is the length of the longest directed path between any two elements in P .*

4.2 Poset-Sorting and Monotonicity Correction for Boolean Labels

We first present the formal statement for our LCA for the poset sorting problem, from which every other result in this section is derived. The algorithm and analysis are presented in [Section 4.3](#).

Theorem 21. *Let P be a poset of N elements and height h , such that each element in P has at most Δ predecessors or successors. Suppose we are given query access to the graph of P , i.e. for any $x \in P$ we can obtain the immediate predecessors or successors of x . Also, suppose we are given query access to some Boolean function f over P . Then, there is an LCA that solves the poset-sorting problem for f over P : in other words, it provides query access to a monotone function $f_{\text{mon}} : P \rightarrow \{0, 1\}$ that can be obtained from f by a sequence of swaps of monotonicity-violating label pairs. The LCA has query complexity and run-time of $(\Delta \log(\frac{N}{\delta}))^{O(\log h)}$, and it uses a random bit-string of length $\text{poly}(\Delta \log \frac{N}{\delta})$. The failure probability of the LCA is at most δ .*

Remark 22. In the setting of [Theorem 21](#), suppose each element of P has at most Γ immediate predecessors or successors, and furthermore, that P is graded (all paths between u and v for any u, v have the same length). Then query complexity and the run-time of our LCA is also bounded by $\Gamma^{O(h)}(\log(N/\delta))^{\log h}$. The number of random bits used is at most $\Gamma^{O(h)} \text{polylog}(N/\delta)$. In particular, when P is the truncated hypercube H_ϵ^d and $\delta = 2^{-10d}$, the query and time complexity are $d^{O(\sqrt{d \log \frac{1}{\epsilon}})}$.

The proof of [Remark 22](#) is given in [Section 4.3.4](#).

Proposition 1 (Local correction). *In the setting of the previous problem, the distance between f and f_{mon} is at most twice the distance of f to monotonicity. Furthermore, the following extra property holds: for any monotone $q : P \rightarrow \{0, 1\}$ we have $\|f_{\text{mon}} - q\| \leq \|f - q\|$.*

¹If x and y are incomparable, then $\text{dist}(x, y)$ is undefined.

Proof. We first show the extra property. Recall that f_{mon} can be obtained from f via a sequence of swaps of monotonicity-violating labels. Since q is monotone, as a result of every single of this swaps the distance to q will either decrease or stay the same. Overall across all the swaps, this means that $\|f_{\text{mon}} - q\| \leq \|f - q\|$.

Taking q to be the closest monotone function to f and using the triangle inequality, we see that the distance between f and f_{mon} is at most twice the distance of f to monotonicity. $\square$

The corollary about tolerant testing of monotone functions follows from our theorem above. We note that the success probability of $2/3$ can be improved via repetition to $1 - \delta$ at the cost of $\log(\frac{1}{\delta})$ multiplicative factor in run-time and query complexity. We also note that the inverse tolerance ratio, given as 0.49 , can be improved to any absolute constant less than 0.5 .

Corollary 7 (Tolerant testing for general posets). *Suppose we are in the setting of Theorem 21, and we also have access to an oracle giving us i.i.d. uniform elements in P . Then, there is an algorithm that uses $\Delta^{O(\log h \log \log \Delta)} (\log(N))^{O(\log h)} \frac{1}{\varepsilon^2}$ queries and run-time, and distinguishes whether the function f is 0.49ε -close or ε -far from monotone. The failure probability is at most $2/3$.*

Proof. First of all, without loss of generality we can assume $\delta = 2/3$, because error probability can be reduced via repetition.

Theorem 21 (via Remark 1) allows us to gain query access to f_{mon} , such that distance of f to f_{mon} is at most twice the distance of f to monotonicity. Then, obtaining the values of both these functions on i.i.d. uniformly random points of P , we estimate $\|f - f_{\text{mon}}\|$ up to error 0.005ε . Then, if the distance of f to monotonicity is at least ε , the distance $\|f - f_{\text{mon}}\|$ will be also at least ε , so the value of the estimate will be at least 0.995ε . On the other hand, if the distance of f to monotonicity is at most 0.49ε , then $\|f - f_{\text{mon}}\|$ will at most 0.98ε and the value of the estimate will be at most 0.985ε . Overall, checking if the value of the estimate is greater than 0.99ε we can see which of the two cases we are in.

For the estimation to succeed, we need to evaluate f and f_{mon} on $O(\frac{1}{\varepsilon^2})$ i.i.d. random elements of P^2 . Overall, this will take $\Delta^{O(\log h \log \log \Delta)} (\log(N))^{O(\log h)} \frac{1}{\varepsilon^2}$ queries and run-time. $\square$

4.3 The LCA for Poset Sorting

In this section, we prove Theorem 21. First, we give a “global” algorithm for the poset sorting problem, which reads all the values of f and writes all the values of f_{mon} . The global algorithm is inefficient, but lends itself easily to a proof of correctness. We prove correctness for the global algorithm, then give our local implementation and show that it simulates the global algorithm.

²For the LCA of f_{mon} we can set the overall success probability parameter to be 0.1 . A Chernoff bound and union bound argument then shows that overall success probability is at least $2/3$.

4.3.1 A Global View

We first present [Algorithm 7](#), which sorts the labels of f in stages by swapping the labels of pairs of vertices that violate monotonicity. We will show that each stage reduces the maximum distance between violated vertices by a factor of 2, which produces a monotone function after $\log h$ stages, where h is the height of the input graph.

Before we present the algorithm, we define the following objects that it constructs during its execution.

Definition 19 (Violation set). *We define the set of violated pairs $\text{viol}_P(f)$ as follows:*

$$\begin{aligned} \text{viol}_P(f) := \\ \{(v, w) \in P \times P : v \succ w, f(v) = 0 \text{ and } f(w) = 1\}. \end{aligned}$$

Definition 20 (k -violation graph). *For a poset P of height h , a function $f : P \rightarrow \{0, 1\}$, and some $k \in [h]$, we define the k -violation graph B_k as follows:*

- $V(B_k) = P$, and
- For $(v, w) \in \text{viol}_P(f)$, $(v, w) \in E(B_k)$ iff $\text{dist}(v, w) \geq k$.

Note that B_k is bipartite and undirected.

Algorithm 7 LCA for sorting labels in a poset: the global view

Given: Poset P of height h , function $f_0 : P \rightarrow \{0, 1\}$

Output: monotone function over P

Let $i \leftarrow 0$

for $0 \leq i \leq \lceil \log h \rceil + 1$ **do**

Let $k \leftarrow \lceil h/2^{i+1} \rceil$

Construct the k -violation graph B_k from P and f_i .

Compute a maximal matching in B_k and let λ map matched vertices to each other, and unmatched vertices to themselves.

For all $x \in P$, let $f_{i+1}(x) = f_i(\lambda(x))$

$i \leftarrow i + 1$

return f_i

4.3.2 Correctness of [Algorithm 7](#)

Recall that we are required to show that our algorithm outputs a monotone function that can be obtained from f by a sequence of monotonicity-violating label swaps. Since it is evident from the pseudocode that this algorithm only performs such swaps, it remains to show only that the output is monotone.

Our algorithm works by finding a maximal matching over the k -violation graph B_k , and swapping the matched labels. We first claim that performing this swap reduces the distance (length of the longest path) between violated labels by at least k .

Lemma 4.3.1 (Distance shortening lemma). *Let P be any poset. Let f be a $\{0, 1\}$ -valued function over P and k be a positive integer. Let B_k be as defined in [Definition 20](#). Suppose one picks some maximal matching M over B_k and obtains a new function f' as follows*

$$f'(x) = \begin{cases} f(x) & \text{if } x \text{ was not matched,} \\ f(y) & \text{if } x \text{ was matched to some } y. \end{cases}$$

Then, we have

$$\max_{(v,w) \in \text{viol}_P(f')} \text{dist}(v, w) \leq \max \left(k - 1, \left(\max_{(v,w) \in \text{viol}_P(f)} \text{dist}(v, w) \right) - k \right).$$

Proof. Let $\lambda : P \rightarrow P$ map x to (i) y if x was mapped to some y by M (ii) x itself otherwise. Note that $f'(x) = f(\lambda(x))$ and also λ is one-to-one.

Let x, y in P be such that $x \succ y$, $f(x) = 0$ and $f(y) = 1$. If $\text{dist}(x, y) \geq k$, it cannot be the case that both $\lambda(x) = x$ and $\lambda(y) = y$, because then M would not be a maximal matching since we could also match x to y . Besides, note that if $\lambda(x) \neq x$ then $\lambda(x) \prec x$ and $\text{dist}(x, \lambda(x)) \geq k$. Analogously, if $\lambda(y) \neq y$ then $\lambda(y) \succ y$ and $\text{dist}(y, \lambda(y)) \geq k$. Additionally, for any $a, b, c \in P$ if $a \succ b \succ c$ then $\text{dist}(a, c) \geq \text{dist}(a, b) + \text{dist}(b, c)$, as there exists a path from a to c that is the union of the longest paths from a to b and b to c . Taking these observations together, we see that only following eight cases are possible:

1. $\text{dist}(x, y) \leq k - 1$.
 - (a) $\lambda(x) \succ \lambda(y)$ and $\text{dist}(\lambda(x), \lambda(y)) \leq \text{dist}(x, y) \leq k - 1$.
 - (b) It is not the case that $\lambda(x) \succ \lambda(y)$.
2. $\text{dist}(x, y) \geq k$.
 - (a) $\lambda(x) = x$ and $\text{dist}(y, \lambda(y)) \geq k$
 - i. $\lambda(x) \succ \lambda(y)$ and $\text{dist}(\lambda(x), \lambda(y)) \leq \text{dist}(x, y) - k$.
 - ii. It is not the case that $\lambda(x) \succ \lambda(y)$.
 - (b) $\lambda(y) = y$ and $\text{dist}(x, \lambda(x)) \geq k$
 - i. $\lambda(x) \succ \lambda(y)$ and $\text{dist}(\lambda(x), \lambda(y)) \leq \text{dist}(x, y) - k$.
 - ii. It is not the case that $\lambda(x) \succ \lambda(y)$.
 - (c) $\text{dist}(x, \lambda(x)) \geq k$ and $\text{dist}(y, \lambda(y)) \geq k$
 - i. $\lambda(x) \succ \lambda(y)$ and $\text{dist}(\lambda(x), \lambda(y)) \leq \text{dist}(x, y) - 2k$.

ii. It is not the case that $\lambda(x) \succ \lambda(y)$.

In the whole, if $\lambda(x) \succ \lambda(y)$ then $\text{dist}(\lambda(x), \lambda(y)) \leq \max(k-1, \text{dist}(x, y) - k)$.

Now let's consider what happens after the swap. Let v_0, w_0 in P maximize $\text{dist}(v_0, w_0)$ subject to $v_0 \succ w_0$, $f'(v_0) = 0$ and $f'(w_0) = 1$. Let $x = \lambda^{-1}(v_0)$ and $y = \lambda^{-1}(w_0)$. Since $f'(v_0) = 0$, we have $x \succeq v_0$ and since $f'(w_0) = 1$, we have $y \preceq w_0$. Therefore, $x \succ y$. Also, $f(x) = f'(v_0) = 0$, $f(y) = f'(w_0) = 1$ and $\lambda(x) \succ \lambda(y)$. The conclusion of the previous paragraph tells us that $\text{dist}(v_0, w_0) = \text{dist}(\lambda(x), \lambda(y)) \leq \max(k-1, \text{dist}(x, y) - k)$. Overall,

$$\begin{aligned} \max_{(v,w) \in \text{viol}_P(f')} \text{dist}(v, w) &= \text{dist}(v_0, w_0) \\ &\leq \max(k-1, \text{dist}(x, y) - k) \\ &\leq \max\left(k-1, \left(\max_{(v,w) \in \text{viol}_P(f)} \text{dist}(v, w)\right) - k\right), \end{aligned}$$

which finishes the proof. $\square$

The following invariant, which will be useful for proving that the output is monotone, is a consequence of [Lemma 4.3.1](#).

Corollary 8 (Distance shortening invariant). *The following holds for all f_i , $0 \leq i \leq \lceil \log h \rceil$:*

$$\max_{(v,w) \in \text{viol}_P(f)} \text{dist}(v, w) \leq \left\lceil \frac{h}{2^i} \right\rceil$$

Proof. We proceed by induction on i . For $i = 0$, the distance must be at most h , because h is the height of P . Assume as an inductive hypothesis that the claim holds for some $i \leq \lceil \log h \rceil$.

B_k is a graph with the properties described in [Definition 20](#): it has an edge joining each pair of vertices in P that violates monotonicity and has distance at least k , for $k = \lceil h/2^{i+1} \rceil$. By the inductive hypothesis, all such distances are between $\lceil h/2^{i+1} \rceil$ and $\lceil h/2^i \rceil$ inclusive. Then by [Lemma 4.3.1](#) we guarantee that

$$\begin{aligned} \max_{(v,w) \in \text{viol}_P(f)} \text{dist}(v, w) &\leq \max\left(\left\lceil \frac{h}{2^{i+1}} \right\rceil - 1, \left\lceil \frac{h}{2^i} \right\rceil - \left\lceil \frac{h}{2^{i+1}} \right\rceil\right) \\ &\leq \left\lceil \frac{h}{2^{i+1}} \right\rceil. \end{aligned}$$

This completes the induction. $\square$

With [Corollary 8](#) in hand, we continue with the proof of correctness.

Lemma 4.3.2 (Correctness). *For any Boolean function f_0 , poset P , and any choice of maximal matchings over B_k in [Algorithm 7](#), the output $f_{\lceil \log h \rceil + 1}$ is monotone over P .*

Proof. By [Corollary 8](#), we have

$$\max_{(v,w) \in \text{viol}_P(f_{\lceil \log h \rceil})} \text{dist}(v, w) \leq \left\lceil \frac{h}{2^{\lceil \log h \rceil}} \right\rceil = 1$$

Then $f_{\lceil \log h \rceil}$ has the property that all pairs that violate monotonicity are immediate neighbors in P . By one more application of [Lemma 4.3.1](#), we have

$$\begin{aligned} & \max_{(v,w) \in \text{viol}_P(f_{\lceil \log h \rceil + 1})} \text{dist}(v, w) \\ & \leq \max \left(1 - 1, 1 - \left\lceil \frac{h}{2^{\lceil \log h \rceil + 1}} \right\rceil \right) = 0, \end{aligned}$$

indicating that $f_{\lceil \log h \rceil + 1}$ is monotone. □

4.3.3 Local Implementation

In this section, we provide an LCA that gives membership query access to the output of [Algorithm 7](#), and we analyze its complexity. To better explain how our LCA simulates [Algorithm 7](#), we present it as a system of three LCAs, each parameterized by the iteration number i . [Algorithm 8](#) makes queries to the i_{th} function f_i and an all-neighbors oracle for P , and answers queries to the i_{th} k -violation graph B_i . [Algorithm 9](#) makes queries to B_i and answers queries to a maximal matching λ_i over it. [Algorithm 10](#) makes queries to f_i and λ_i and answers queries to f_{i+1} , which swaps the matched labels.

Algorithm 8 An LCA for undirected all-neighbors queries $B_i(x, P, f_i, h, i)$

Given: Target vertex x , all-neighbors (immediate predecessor and successor) oracle for P , membership query oracle f_i , height h , iteration number i .

Initialize $k \leftarrow \lceil h/2^{i+1} \rceil$

if $f_i(x) = 1$ **then**

$S \leftarrow$ the set of all successors of x

else

$S \leftarrow$ the set of all predecessors of x

Compute longest path $\text{dist}(x, y)$ for each $y \in S$ by dynamic programming

Remove any y from S such that $\text{dist}(x, y) < k$ or $f(y) = f(x)$

return S

Algorithm 9 An LCA for maximal matchings $\lambda_i(x, B_i, r, \delta)$

Given: Target vertex x , undirected all-neighbors query oracle B_i , random seed r , and confidence parameter δ .

Call the algorithm described in [Corollary 3](#) with B_i as the graph, x as the target, and r as the random seed and Δ as the degree bound. If x is in the matching, return the vertex that it is matched to; otherwise return x .

Algorithm 10 An LCA for membership queries $f_{i+1}(x, \lambda_i, f_i)$

Given: Target vertex x , matching query oracle λ_i , membership query oracle f_i
return $f_i(\lambda_i(x))$

Analysis of our implementation

Throughout this section, for any algorithm A , the notation $T(A)$ denotes the running time of A .

Claim 11 (Behavior of B_i). *For any $0 \leq i \leq \lceil \log h \rceil + 1$, B_i provides all-neighbors query access to the $\lceil h/2^{i+1} \rceil$ -violation graph of P with respect to f_i . Furthermore, if each element of P has at most Δ successors or predecessors, then $T(B_i) \leq O(\Delta \cdot T(f_i))$.*

Proof. If $f_i(x) = 1$, then all neighbors of x in the $\lceil h/2^{i+1} \rceil$ -violation graph of P are successors of x . Finding all the successors takes $O(\Delta)$ time by depth-first search. Computing $\text{dist}(x, y)$ for each successor y of x takes $O(\Delta)$ time by standard dynamic programming techniques for finding longest paths in a DAG. Comparing $f_i(x)$ to $f_i(y)$ takes $O(\Delta)$ queries to f_i , and therefore $O(\Delta \cdot T(f_i))$ time. The case of $f_i(x) = 0$ is symmetric. $\square$

Claim 12 (Behavior of λ_i). *For any $0 \leq i \leq \lceil \log h \rceil + 1$, and $\delta \in (0, 1]$, λ_i provides query access to a maximal matching over B_i with probability $1 - \delta/(\lceil \log h \rceil + 1)$, using a random seed of length $\text{poly}(\Delta \log(N/\delta))$. Furthermore, $T(\lambda_i) \leq \text{poly}(\Delta \log(N/\delta)) \cdot T(B_i)$.*

Proof. Let $\delta' = \delta/(\lceil \log h \rceil + 1)$. By [Corollary 3](#), λ_i fails with probability at most δ' , where the query complexity and the length of the random seed are each $\text{poly}(\Delta \log(N/\delta'))$. Since h is always at most Δ , this is still $\text{poly}(\Delta \log(N/\delta))$. The claim follows from the fact that since the queries are made to B_i , each query takes time $T(B_i)$. $\square$

We now proceed with our proof of [Theorem 21](#), which we restate here for convenience.

Theorem 21. *Let P be a poset of N elements and height h , such that each element in P has at most Δ predecessors or successors. Suppose we are given query access to the graph of P , i.e. for any $x \in P$ we can obtain the immediate predecessors or successors of x . Also, suppose we are given query access to some Boolean function f over P . Then, there is an LCA that solves the poset-sorting problem for f over P : in other words, it provides query access to a monotone function $f_{\text{mon}} : P \rightarrow \{0, 1\}$ that can be obtained from f by a sequence of swaps of monotonicity-violating label pairs. The LCA has query complexity and run-time of $(\Delta \log(\frac{N}{\delta}))^{O(\log h)}$, and it uses a random bit-string of length $\text{poly}(\Delta \log \frac{N}{\delta})$. The failure probability of the LCA is at most δ .*

Proof. Assume that for each $i \leq \lceil \log h \rceil + 1$, the matching provided by λ_i is maximal for B_i . Under this condition, f_i , λ_i , and B_i implement LCAs for all the objects expected by [Algorithm 7](#) (by [Claim 11](#) and [Claim 12](#)). The correctness result of [Lemma 4.3.2](#) implies that our implementation of $f_{\lceil \log h \rceil + 1}$ is an LCA for a function with the properties claimed in this theorem. We will bound the running time and query complexity by a recurrence relation.

As a base case, we will let $T(f_0)$ be 1. We have three recurrences: $T(B_i) \leq O(\Delta \cdot T(f_i))$, $T(\lambda_i) \leq \text{poly}(\Delta \log(N/\delta)) \cdot T(B_i)$, and $T(f_{i+1}) \leq O(T(f_i) + T(\lambda_i))$. To simplify:

$$\begin{aligned}
T(f_{i+1}) &\leq O(T(f_i) + T(\lambda_i)) \\
&\leq T(f_i) + \text{poly}(\Delta \log(N/\delta)) \cdot T(B_i) \\
&\leq T(f_i) + \text{poly}(\Delta \log(N/\delta)) \cdot \Delta T(f_i) \\
&\leq \text{poly}(\Delta \log(N/\delta)) \cdot T(f_i)
\end{aligned}$$

This recurrence resolves to

$$T(f_i) = (\Delta \log(N/\delta))^{O(i)}$$

for both running time and query complexity. Letting $i = \lceil \log h \rceil + 1$, we have a total running time and query complexity of $(\Delta \log(N/\delta))^{O(\log h)}$.

We initialize our algorithm with a random bit string of length $\text{poly}(\Delta \log(N/\delta))$ as required by [Claim 12](#). Each call to λ_i fails with probability $\delta/(\lceil \log h \rceil + 1)$; this gives a total failure probability of δ . Therefore, with probability at least $1 - \delta$ all the matchings are maximal and our analysis holds. $\square$

Remark 23. We remark that the LCA implementation of [Algorithm 7](#) can be modified to give query access to a label permutation π such that $f\pi$ is monotone and $\Pr_{x \sim P}[f(x) \neq (f\pi)(x)] \leq 2\alpha$. Since the algorithm works by performing a sequence of label-swaps on pairs that violate monotonicity in the poset, then outputting the function value that ends up at the queried vertex x , it can instead track the swaps and output the identity of the vertex that x receives its final label from. The modified algorithm can be thought of as an LCA that gives query access to a label permutation.

4.3.4 Refined Local Implementation

Here we explain how an improved run-time can be achieved for posets with additional characteristics. We shall assume that each element of the poset P has at most Γ immediate predecessors and at most Γ immediate successors. Additionally, we assume that the poset P is graded. This is achieved again by implementing the algorithm in [Section 4.3.1](#), but the run-time is somewhat faster than the one achieved in in [Section 4.3.3](#). To be fully specific, [Section 4.3.1](#) would give us a run-time of $\Gamma^{O(h \log h)} (\log(N/\delta))^{\log h}$ in this setting, which is here improved to $\Gamma^{O(h)} (\log(N/\delta))^{\log h}$.

Similarly to [Section 4.3.3](#), we provide an LCA that gives membership query access to the output of [Algorithm 7](#), and we analyze its complexity. This is again presented it as a system of three LCAs, each parameterized by the iteration number i . [Algorithm 11](#) makes queries to the i_{th} function f_i and an all-neighbors oracle for P , and answers queries to the i_{th} k -violation graph B_i . [Algorithm 12](#) makes queries to B_i and answers queries to a maximal

matching λ_i over it. [Algorithm 13](#) makes queries to f_i and λ_i and answers queries to f_{i+1} , which swaps the matched labels.

Algorithm 11 An LCA for undirected all-neighbors queries $B_i(x, P, f_i, h, i)$

Given: Target vertex x , immediate predecessor and successor oracle for P , membership query oracle f_i , height h , iteration number i .

Initialize $k \leftarrow \lceil h/2^{i+1} \rceil$

if $f_i(x) = 1$ **then**

$S \leftarrow$ the set of all successors y of x such that $k \leq \text{dist}(x, y) \leq 2k$.

else

$S \leftarrow$ the set of all successors y of x such that $k \leq \text{dist}(x, y) \leq 2k$.

Note: because the poset is graded, the collections of successors and predecessors above can be found via breath-first search.

Remove any y from S such that $f(y) = f(x)$

return S

Algorithm 12 An LCA for maximal matchings $\lambda_i(x, B_i, r, \delta)$

Given: Target vertex x , undirected all-neighbors query oracle B_i , random seed r , and confidence parameter δ .

Call the algorithm described in [Corollary 3](#) with B_i as the graph, x as the target, and r as the random seed and $\Gamma^{\lceil h/2^i \rceil}$ as the degree bound. If x is in the matching, return the vertex that it is matched to; otherwise return x .

Algorithm 13 An LCA for membership queries $f_{i+1}(x, \lambda_i, f_i)$

Given: Target vertex x , matching query oracle λ_i , membership query oracle f_i

return $f_i(\lambda_i(x))$

Analysis of our implementation

Throughout this section, for any algorithm A , the notation $T(A)$ denotes the running time of A . This is the same convention used in [Section 4.3.3](#). Furthermore, the following two claims are analogous to [Claim 11](#) and [Claim 12](#) respectively.

Claim 13 (Behavior of B_i). *For any $0 \leq i \leq \lceil \log h \rceil + 1$, suppose*

$$\max_{(v,w) \in \text{viol}_P(f_i)} \text{dist}(v, w) \leq \lceil h/2^i \rceil.$$

Then, B_i provides all-neighbors query access to the $\lceil h/2^{i+1} \rceil$ -violation graph of P with respect to f_i . Furthermore, the degrees of all vertices in B_i are bounded by $\Gamma^{\lceil h/2^i \rceil}$ and $T(B_i) \leq O(\Gamma^{\lceil h/2^i \rceil} \cdot T(f_i))$.

Proof. If $f_i(x) = 1$, then all neighbors of x in the $\lceil h/2^{i+1} \rceil$ -violation graph of P are successors of x . As, $\lceil h/2^i \rceil \leq 2\lceil h/2^{i+1} \rceil = 2k$, we see that initializing S to have only elements of distance at most $2k$ does not leave out any neighbors of x in the $\lceil h/2^{i+1} \rceil$ -violation graph of P . Comparing with the definition of the $\lceil h/2^{i+1} \rceil$ -violation graph of P , we see that the elements given by [Algorithm 11](#) are precisely the neighbors of x in the $\lceil h/2^{i+1} \rceil$ -violation graph of P .

Finally, the bound of $\Gamma^{\lceil h/2^i \rceil}$ on the degree of B_i follows, because each element in P has at most Γ immediate predecessors or successors. The bound on the run-time follows from the bound on degree of B_i . $\square$

Claim 14 (Behavior of λ_i). *For any $0 \leq i \leq \lceil \log h \rceil + 1$, and $\delta \in (0, 1]$, if degrees of all vertices in B_i are bounded by $\Gamma^{\lceil h/2^i \rceil}$, then λ_i provides query access to a maximal matching over B_i with probability $1 - \delta/(\lceil \log h \rceil + 1)$, using a random seed of length $\text{poly}(\Gamma^{\lceil h/2^i \rceil} \log(N/\delta))$. Furthermore, the run-time $T(\lambda_i)$ is bounded by $\text{poly}(\Gamma^{\lceil h/2^i \rceil} \log(N/\delta)) \cdot T(B_i)$.*

Proof. Let $\delta' = \delta/(\lceil \log h \rceil + 1)$. By [Corollary 3](#), λ_i fails with probability at most δ' , where the query complexity and the length of the random seed are each $\text{poly}(\Gamma^{\lceil h/2^i \rceil} \log(N/\delta'))$. The claim follows from the fact that since the queries are made to B_i , each query takes time $T(B_i)$. $\square$

We now proceed with our proof of [Remark 22](#), that is to proving an overall run-time bound of $\Gamma^{O(h)}(\log(N/\delta))^{\log h}$.

Proof. We first argue, using an induction over i , that f_i , λ_i , and B_i implement LCAs for all the objects expected by [Algorithm 7](#) (with overall probability of at least $1 - \delta$). The base case $i = 0$ is immediate. Suppose, this holds up to iteration i (i.e. condition on this event). Then, by [Corollary 8](#) we have

$$\max_{(v,w) \in \text{viol}_P(f_i)} \text{dist}(v, w) \leq \lceil h/2^i \rceil,$$

so the premise of [Claim 13](#) holds. Now, one of the conclusions of [Claim 13](#) is that degrees of all vertices in B_i are bounded by $\Gamma^{\lceil h/2^i \rceil}$, which is the premise of [Claim 14](#). Together, the conclusions of [Claim 13](#) and [Claim 14](#), imply that, with probability $1 - \delta/(\lceil \log h \rceil + 1)$, B_i , λ_i and f_{i+1} implement the corresponding quantities expected by [Algorithm 7](#). Via a union bound over all i we see that with overall probability of at least $1 - \delta$ this indeed holds for all i .

We will bound the running time and query complexity by a recurrence relation. As a base case, we will let $T(f_0)$ be 1. We have three recurrences: $T(B_i) \leq O(\Gamma^{\lceil h/2^i \rceil} \cdot T(f_i))$, $T(\lambda_i) \leq \text{poly}(\Gamma^{\lceil h/2^i \rceil} \log(N/\delta)) \cdot T(B_i)$, and $T(f_{i+1}) \leq O(T(f_i) + T(\lambda_i))$. To simplify:

$$\begin{aligned}
T(f_{i+1}) &\leq O(T(f_i) + T(\lambda_i)) \\
&\leq T(f_i) + \text{poly}(\Gamma^{\lceil h/2^i \rceil} \log(N/\delta)) \cdot T(B_i) \\
&\leq T(f_i) + \text{poly}(\Gamma^{\lceil h/2^i \rceil} \log(N/\delta)) \cdot \Gamma^{\lceil h/2^i \rceil} T(f_i) \\
&\leq \text{poly}(\Gamma^{\lceil h/2^i \rceil} \log(N/\delta)) \cdot T(f_i)
\end{aligned}$$

Thus, we obtain

$$\begin{aligned}
T(f_i) &= \prod_{i=0}^{\lceil \log h \rceil} \text{poly}(\Gamma^{\lceil h/2^i \rceil} \log(N/\delta)) \\
&= \Gamma^{O(h)} (\log(N/\delta))^{\log h}
\end{aligned}$$

for both running time and query complexity.

We note that our algorithm is initialized with a random bit string of length $\Gamma^{O(h)} \text{polylog}(N/\delta)$, which is as required. $\square$

4.4 Correcting Real-Valued Functions Over Posets in ℓ_1 Distance

In this section, we prove [Theorem 5](#) by analyzing our algorithm for correcting a real-valued function over a poset in a way that preserves the ℓ_1 distance to monotonicity within a factor of 2. This extends the monotonicity corrector of [\[84\]](#) to handle functions with non-Boolean ranges.

Lemma 4.4.1 (ℓ_1 correction of k -valued functions). *Let P be a poset and $f : P \rightarrow [k]$ be α -close to monotone in ℓ_1 distance. There is an LCA that makes queries to f and outputs queries to $g : P \rightarrow [k]$, such that g is monotone and $\|f - g\|_1 \leq 2\alpha$. The LCA makes $(\Delta \log N)^{O(\log h \log k)}$ queries, where Δ is the maximum number of predecessors or successors of any element in P , N is the number of vertices, and h is the length of the longest directed path.. It uses a random seed of length $\text{poly}(\Delta \log N)$, and succeeds with probability $1 - N^{-10}$.*

The following lemmas are used in the proof of correctness of our algorithm.

Lemma 4.4.2 (Equivalence of k -valued and bitwise monotonicity). *Let $f : P \rightarrow [k]$ be a function and f_i be the projection of f onto the i th most significant bit of k , i.e. $f_i(x) = 1$ if the i th bit of $f(x)$ is 1, for each $i \in [\lceil \log k \rceil]$. Let P_i be the poset on the elements of P with the relation*

$$x \prec_{P_i} y := x \prec_P y \text{ and } f_j(x) = f_j(y) \text{ for all } j < i.$$

Then f is monotone if and only if each f_i is monotone over the corresponding P_i .

Proof. Let x and y be comparable elements of P ; w.l.o.g. $x \prec_P y$. It is sufficient to show that $f(x) > f(y)$ if and only if there is some i for which $x \prec_{P_i} y$ and $f_i(x) > f_i(y)$. We claim that this i is the most significant bit in which $f(x)$ and $f(y)$ differ. It is certainly true that $f(x) > f(y)$ if and only if $f_i(x) > f_i(y)$ for this i , and since $f_j(x) = f_j(y)$ for all $j < i$ by the choice of i , we have $x \prec_{P_i} y$ as well. $\square$

Lemma 4.4.3 (Preservation of closeness to monotone functions). *Let g be obtained from f by swapping the labels of a pair $x \prec_P y$ that violates monotonicity. Then for any monotone function m , $\|g - m\|_1 \leq \|f - m\|_1$.*

Proof of Lemma 4.4.3. Since m is monotone, certainly $m(x) \leq m(y)$, and since f violates monotonicity on this pair, certainly $f(x) \geq f(y)$ (and therefore $g(y) \geq g(x)$). We will examine the contribution of x and y to each of $\|f - m\|_1$ and $\|g - m\|_1$. We have the following cases:

- $f(y) \leq f(x) \leq m(x) \leq m(y)$: then

$$\begin{aligned} |m(x) - f(x)| + |m(y) - f(y)| &= m(x) + m(y) - (f(x) + f(y)) \\ &= m(x) + m(y) - (g(x) + g(y)) \\ &= |m(x) - g(x)| + |m(y) - g(y)|. \end{aligned}$$

The distance of this pair does not change. The case of $m(x) \leq m(y) \leq f(x) \leq f(y)$ is symmetric.

- $f(y) \leq m(x) \leq m(y) \leq f(x)$: then

$$\begin{aligned} |m(x) - f(x)| + |m(y) - f(y)| &= (f(x) - m(x)) + (m(y) - f(y)) \\ &\geq (f(x) - m(y)) + (m(x) - f(y)) \\ &= |g(y) - m(y)| + |g(x) - m(x)|. \end{aligned}$$

The distance of this pair does not increase. The case of $m(x) \leq f(y) \leq f(x) \leq m(y)$ is symmetric.

- $f(y) \leq m(x) \leq f(x) \leq m(y)$: then

$$\begin{aligned} |m(x) - f(x)| + |m(y) - f(y)| &= (f(x) - m(x)) + (m(y) - f(y)) \\ &\geq (m(x) - f(y)) + (m(y) - f(x)) \\ &= |g(x) - m(x)| + |g(y) - m(y)|. \end{aligned}$$

The distance of this pair does not increase. The case of $m(x) \leq f(y) \leq m(y) \leq f(x)$ is symmetric. $\square$

The corollary follows from repeated application of Lemma 4.4.3 and the triangle inequality.

Corollary 9 (ℓ_1 error preservation). *Let g be obtained from f by a series of swaps of label pairs that violate monotonicity in f . Then $\|g - f\|_1 \leq 2 \cdot \text{dist}_1(f, \text{mono})$.*

Here we present the LCA implementation of [Algorithm 21](#). Here, `BOOLEANCORRECTOR` refers to the LCA described in [Theorem 21](#) and [Remark 23](#).

Algorithm 14 LCA implementation of [Algorithm 21](#), $k\text{-CORRECTOR}(x, P, f, i, \mathbf{r})$

```

1: Given: Target vertex  $x$ , all-neighbors (immediate predecessor and successor) oracle for
    $P$ , query access to  $f : P \rightarrow [k]$ , iteration number  $i$ , random seed  $\mathbf{r} = r_1 \circ \dots \circ r_i$ .
2: Output: query access to function  $g : P \rightarrow [k]$  which is monotone when truncated to the
   first  $i$  most significant bits.
3: if  $i = 0$  then return  $f(x)$ 
4: else
5:    $S \leftarrow$  the set of all predecessors and successors of  $x$  in  $P$ 
6:   for  $y \in S$  do
7:     Let  $f'(y) \leftarrow k\text{-CORRECTOR}(y, P, f, i - 1, r_1 \circ \dots \circ r_{i-1})$ .
8:   Let  $f'_i$  be defined as in Algorithm 21, and  $P'_i$  be similarly defined with respect to  $f'_i$ .
9:   Remove any  $y$  from  $S$  such that  $f'_i(y) = f'_i(x)$  or  $y$  and  $x$  are incomparable in  $P'_i$ .
10:  Let  $z \leftarrow \text{BOOLEANCORRECTOR}(x, P'_i, f'_i, r_i)$ 
11:  return  $f'(z)$ 

```

Algorithm 15 `HYPERCUBECORRECTOR`($f, \varepsilon, \mathbf{r}$)

Given: function $f : \{-1, 1\} \rightarrow [-1, 1]$ given as succinct representation, additive error parameter $\varepsilon > 0$, random seed $\mathbf{r} = r_1 \circ \dots \circ r_{\lceil \log 1/\varepsilon \rceil}$.

Output: succinct representation of monotone function $g : \{-1, 1\} \rightarrow [-1, 1]$.

$P \leftarrow$ representation of a function that takes x and outputs `TRUNCATEDCUBE`(x, ε)

$f' \leftarrow$ representation of a function that takes x and outputs $\lfloor f(x)/\varepsilon \rfloor$

$f'' \leftarrow$ representation of a function that takes x and outputs

$$\begin{cases} \varepsilon \cdot k\text{-CORRECTOR}(x, P, f', \lceil \log(1/\varepsilon) \rceil, \mathbf{r}) & -\sqrt{2d \log 2/\varepsilon} \leq |x| \leq \sqrt{2d \log 2/\varepsilon} \\ 1 & |x| \geq \sqrt{2d \log 2/\varepsilon} \\ -1 & |x| \leq -\sqrt{2d \log 2/\varepsilon} \end{cases}$$

return f''

Lemma 4.4.4 (Correctness and query complexity of [Algorithm 14](#)). *With probability $1 - i \cdot N^{-11}$ over a random seed r of length $\text{poly}(\Delta \log N)$, the algorithm $k\text{-CORRECTOR}(x, P, f, i, r)$ gives query access to a function g that is monotone when truncated to the first i most significant bits. Its query complexity is $(\Delta \log N)^{O(i \log h + 1)}$, and $\|g - f\|_1 \leq 2\alpha$, where α is the ℓ_1 distance of f to the nearest monotone function.*

Proof. Fix the random seed $\mathbf{r}$ and assume all calls to `BOOLEANCORRECTOR` succeed with $\mathbf{r}$, then we proceed by induction. In the base case, f is certainly monotone when truncated to 0

bits and the algorithm makes only 1 query. In the inductive case, suppose the claim holds for $i - 1$; in other words $k\text{-CORRECTOR}(y, P, f, i - 1, r_1 \circ \dots \circ r_{i-1})$ makes $(\Delta \log N)^{O((i-1) \log h + 1)}$ queries and returns a function that is monotone in the first $i - 1$ bits. Then when $k\text{-CORRECTOR}$ is called with iteration number i , the function f'_j is monotone over P'_j for all $j < i$. $\text{BOOLEANCORRECTOR}(x, P'_i, f'_i, r_i)$ returns a vertex to swap labels with x such that the resulting function is monotone in the i_{th} bit, over the poset P'_i . Then the function returned by $k\text{-CORRECTOR}$ satisfies the conditions of [Lemma 4.4.2](#) for the first i bits, so it must be monotone in the first i bits.

We now bound the failure probability and distance to f . The failure probability of BOOLEANCORRECTOR is N^{-11} and we call BOOLEANCORRECTOR on i different graphs, so by union bound the total failure probability is $\leq i \cdot N^{-11}$ as desired. The fact that $\|g - f\|_1 \leq 2\alpha$ follows from [Corollary 9](#). $\square$

We can now prove [Theorem 5](#).

Theorem 5. *[Local monotonicity correction of real-valued functions] Let P be a poset with N elements, such that every element has at most Δ predecessors or successors and the longest directed path has length h . Let $f : P \rightarrow [-1, 1]$ be α -close to monotone in ℓ_1 distance. There is an LCA that makes queries to f and outputs queries to $g : P \rightarrow [-1, 1]$, such that g is monotone and $\|f - g\|_1 \leq 2\alpha + 3\varepsilon$. The LCA makes $(\Delta \log N)^{O(\log h \log(1/\varepsilon))}$ queries, uses a random seed of length $\text{poly}(\Delta \log N)$, and succeeds with probability $1 - N^{-10}$.*

Proof of Theorem 5. Given some $\varepsilon \in (0, 1/2)$, let $f_\varepsilon(x) := \lfloor f(x)/\varepsilon \rfloor$; certainly queries to f_ε can be simulated by queries to f . On input x , run $k\text{-CORRECTOR}(x, P, f_\varepsilon, \lceil \log(2/\varepsilon) \rceil, r)$ with a random seed r of length $\text{poly}(\Delta \log N)$. By [Lemma 4.4.4](#), this makes $(\Delta \log N)^{O(\log(1/\varepsilon) \log h)}$ queries to f_ε and outputs $g_\varepsilon(x)$, where g is monotone and $\|g_\varepsilon - f_\varepsilon\|_1 \leq 2 \cdot \text{dist}_1(f_\varepsilon, \text{mono})$. Since f is α -close to some monotone function m , we have $\text{dist}_1(f_\varepsilon, \text{mono}) \leq \|f_\varepsilon - m/\varepsilon\|_1 \leq \|f/\varepsilon - m/\varepsilon\|_1 + \|f/\varepsilon - f_\varepsilon\|_1 \leq \alpha/\varepsilon + 1$.

Return $g(x) := \varepsilon \cdot g_\varepsilon(x)$. Then

$$\|g - f\|_1 = \|\varepsilon g_\varepsilon - f\|_1 \leq \|\varepsilon g_\varepsilon - \varepsilon f_\varepsilon\|_1 + \|\varepsilon f_\varepsilon - f\|_1 \leq 2\varepsilon(\alpha/\varepsilon + 1) + \varepsilon \leq 2\alpha + 3\varepsilon.$$

The failure probability is $N^{-11} \cdot \lceil \log(2/\varepsilon) \rceil$ by [Lemma 4.4.4](#), but we will assume that $\lceil \log(2/\varepsilon) \rceil < N$. Otherwise, the allowed query complexity and running time would exceed Δ^N , which is $> \Delta N$ for any $\Delta, N > 1$. With $O(\Delta N)$ query complexity and running time, a trivial algorithm would suffice: one could solve the linear program with ΔN monotonicity constraints, minimizing $\|g - f\|_1$. Under our assumption, the failure probability is at most N^{-10} . $\square$

Corollary 10 (Monotonizing a representation of a function on the Boolean cube). *Let $f : \{-1, 1\}^d \rightarrow [-1, 1]$ be α -close to monotone in ℓ_1 distance, given as a succinct representation of size s_f . There is an algorithm that runs in time $2^{\tilde{O}(\sqrt{d} \log^{3/2}(1/\varepsilon))} \cdot s_f$ time and outputs a monotone function g such that $\|f - g\|_1 \leq 2\alpha + 4\varepsilon$. The size of the representation of g is*

$2^{\tilde{O}(\sqrt{d} \log^{3/2}(1/\varepsilon))} \cdot s_f$. The algorithm uses a random seed of length $2^{\tilde{O}(\sqrt{d} \log(1/\varepsilon))}$ and succeeds with probability $1 - 2^{-10d}$.

Proof. Let $f : \{-1, 1\}^d \rightarrow [-1, 1]$ be α -close to monotone in ℓ_1 distance. We call the algorithm $\text{HYPERCUBECORRECTOR}(f, \varepsilon, r)$ with a random seed r of length $2^{O(\sqrt{d} \log(1/\varepsilon) \log d)}$. First we set the poset to be the truncated cube of width $\sqrt{2d \log 2/\varepsilon}$, which is a poset such that every element has at most $2^{O(\sqrt{d \log(1/\varepsilon)} \log d)}$ predecessors and successors. The representation of this poset (not its transitive closure) has size $\text{poly}(d, \log(1/\varepsilon))$. Then we set f' to be a function that discretizes f to $2/\varepsilon$ possible values. This representation has size $O(s_f/\varepsilon)$. Then we set f'' to be a function that computes the Hamming weight of x , then either calls k -CORRECTOR or outputs a constant. So its size is the size of the k -CORRECTOR representation times some overhead that is polynomial in d and $1/\varepsilon$. Since the Δ parameter for the truncated cube is $2^{O(\sqrt{d \log(1/\varepsilon)} \log d)}$, the h parameter is $O(\sqrt{d})$, and the N parameter is $< 2^d$, the worst-case running time and query complexity of this instance of k -CORRECTOR is $2^{O(\sqrt{d} \log d \log^{3/2}(1/\varepsilon))}$ by Lemma 4.4.4. Thus the representation size of the k -CORRECTOR instance is $2^{\tilde{O}(\sqrt{d} \log^{3/2}(1/\varepsilon))}$, and so the representation size of f'' is $2^{\tilde{O}(\sqrt{d} \log^{3/2}(1/\varepsilon))} \cdot s_f$. With the random seed of length $2^{O(\sqrt{d \log(1/\varepsilon)} \log d)} = \text{poly}(\Delta \log N)$, k -CORRECTOR succeeds with probability $N^{-10} \leq 2^{-10d}$. $\square$

4.5 Analysis of the Matching Algorithm

In this section we give an algorithm for generating a succinct representation of a matching over the violated pairs of the hypercube whose weight is a constant factor of the distance to monotonicity. The core of the algorithm is an LCA for finding such a matching over the violated pairs of an arbitrary poset.

Lemma 4.5.1 (Equivalence of distance to monotonicity and maximum-weight matching). *Let W be the total weight of the maximum-weight matching of the violation graph of f . Then $\text{dist}_1(f, \text{mono}) = W/N$.*

Proof. This proof is analogous to the proof of Lemma 3.1 of [14]; for completeness we include a proof. The proof of $\text{dist}_1(f, \text{mono}) \geq W/N$ is straightforward; for any edge (x, y) , $x \prec y$ in the matching, any monotone function must have $g(y) \geq g(x)$ and thus $(f(x) - g(x)) + (g(y) - f(y)) \geq f(x) - f(y)$. So the contribution of x and y to the ℓ_1 distance is at least the weight of (x, y) .

For the other direction, we give a proof exactly analogous to the max-weight matching characterization of distance to the class of Lipschitz functions as presented in [14]. Let g be the closest monotone function to f in ℓ_1 -distance. We will partition the vertices of the cube into three classes: $V_> := \{x \mid f(x) > g(x)\}$, $V_< := \{x \mid f(x) < g(x)\}$, and $V_= := \{x \mid f(x) = g(x)\}$. We will duplicate the vertices of $V_=$ and group one copy with $V_>$ and one copy with $V_<$, to form vertex sets $V_{\geq}$ and $V_{\leq}$. The duplicated copies of x will be denoted $x_{\geq}$ and $x_{\leq}$. We define the bipartite graph $B_{f,g}$ to be the graph on $V_{\geq} \times V_{\leq}$ with an edge (x, y) if $x \prec y$ and

$g(x) = g(y)$. The weight of the edge (x, y) is the same as it is in $\text{viol}(f)$; it is just $f(x) - f(y)$. Intuitively, a matching in $B_{f,g}$ will represent a set of edges along which some a minimal amount of label mass is transferred to correct monotonicity. First, we claim that $B_{f,g}$ has a matching which matches every vertex in $V_{>} \cup V_{<}$. This will follow from Hall's marriage theorem if we can show that for every $A \subseteq V_{>}$ or $A \subseteq V_{<}$, we have $|A| \leq |N(A)|$.

Suppose for contradiction that the marriage condition is false, and without loss of generality let A be the largest subset of $V_{>}$ for which $|A| > |N(A)|$. We would like to claim that for any $x \in A \cup N(A)$ and $y \notin A \cup N(A)$, if $x \prec y$ then $g(x) < g(y)$. We consider four possible cases:

- a) If $x \in A$, $y \in V_{>}$, $x \prec y$, and $g(x) = g(y)$, then $y \in A$ as well, by the choice of A to be the largest set that fails the marriage condition. This is because $N(y) \subseteq N(x)$: any neighbor z of y must have $g(z) = g(y) = g(x)$, have $x \prec y \prec z$, and be in $V_{\leq}$, which makes it a neighbor of x .
- b) If $x \in N(A)$, $y \in V_{\leq}$, $x \prec y$, and $g(x) = g(y)$, then $g(y) = g(x) = g(z)$ and $z \prec x \prec y$ for some $z \in A$, so $y \in N(A)$.
- c) If $x \in A$, $y \in V_{\leq}$, $x \prec y$, and $g(x) = g(y)$, then $y \in N(A)$.
- d) If $x \in N(A)$, $y \in V_{>}$, $x \prec y$, and $g(x) = g(y)$, then $g(y) = g(x) = g(z)$ and $z \prec x \prec y$ for some $z \in A$, so as in case (a) we have $N(y) \subseteq N(z)$ and therefore $y \in A$.

We have shown that for any $x \in A \cup N(A)$ and $y \notin A \cup N(A)$, if $x \prec y$ then $g(x) < g(y)$. Then there is some $\delta > 0$ for which $g(x)$ can be increased by δ for every $x \in A \cup N(A)$ without breaking monotonicity. This decreases $\|f - g\|_1$ by $\delta(|A| - |N(A)|) > 0$, which contradicts the assumption that g is the closest monotone function.

Having proven that $B_{f,g}$ contains a matching M' on all vertices in $V_{>} \cup V_{<}$, we will now show that its weight is equal to $N\|f - g\|_1$, using the fact that $g(x) = g(y)$ for all $(x, y) \in M'$:

$$\sum_{(x,y) \in M'} f(x) - f(y) = \sum_{(x,y) \in M'} f(x) - g(x) + g(y) - f(y) = \sum_{x \in V_{>} \cup V_{<}} |f(x) - g(x)| = N\|f - g\|_1.$$

We will now find a matching M in $\text{viol}(f)$ of equal weight. First replace each $x_{\leq}$ and $x_{\geq}$ with x , obtaining an edge set in $\text{viol}(f)$ of equal weight that is not necessarily a matching, but is a set of disjoint paths. We replace each path with the edge between its endpoints; i.e. if there is some pair of edges $(y, x_{\leq})$ and $(x_{\geq}, z)$, then we know that $y \prec x \prec z$ and $f(y) - f(z) = ((f(y) - f(x)) + (f(x) - f(z)))$, so the matching edge (y, z) has weight equal to the total weight of the path it replaces. Then M is a matching in $\text{viol}(f)$ of weight equal to $N\|f - g\|_1$, which is equal to $N \cdot \text{dist}_1(f, \text{mono})$. $\square$

4.5.1 Details and Correctness of MATCHVIOLATIONS

The algorithm MATCHVIOLATIONS given in [Section 4.6](#) makes calls to an algorithm called FILTEREDGES, which removes vertices that have already been matched or are not incident to any heavy edges. We give the pseudocode for FILTEREDGES here.

Algorithm 16 LCA: FILTEREDGES(P, f, t, M, x)

- 1: **Given:** Poset P , function $f : P \rightarrow [-1, 1]$, and matching M given as succinct representations, weight threshold t , vertex x
- 2: **Output:** All neighbors of x in the graph of violation score $\geq t$ and not in M
- 3: **return**

$$\{y \in P(x) \mid M(y) = \perp \text{ and } [(x < y \text{ and } f(x) \geq f(y) + t) \text{ or } (x > y \text{ and } f(x) \leq f(y) - t)]\}$$

Lemma 4.5.2. *Let P be a poset with N vertices, and let Δ be an upper bound on the number of predecessors and successors of any vertex in P . Then the output of the LCA MATCHVIOLATIONS(P, f, ε, r) with a random seed r of length $\text{poly}(\Delta, \log N)$, is a matching of weight at least $N(\frac{1}{4}\text{dist}_1(f, \text{mono}) - \varepsilon)$ with probability at least $1 - N^{-10}$.*

Proof. This is a small modification to the standard greedy algorithm for high-weight matching. Fix the random seed r and assume all calls to the algorithm of [64] using r succeed. Let M' be a maximum-weight matching over $\text{viol}(f)$, and let M be a matching returned by MATCHVIOLATIONS. We will use M to refer to the matching and its succinct representation interchangeably. For each edge $e \in M'$, let w_e be the weight of e (i.e. the violation score of its endpoints), and δ_e be the total weight of edges in $M \setminus M'$ that share an endpoint with e .

First we show by induction that at the start of each iteration i , M is maximal over the subgraph of $TC(P)$ induced by edges of weight greater than $2^{-(i-1)}$. In the base case, M is initialized to be the empty matching, which is maximal on the edges of weight > 2 , as there are no such edges. In the inductive case, we assume the invariant is still true at the start of iteration i . Then when FILTEREDGES (Section 4.5.1) is called in iteration $i + 1$, the vertices removed are exactly those that are either already in M , or not incident to any edges of weight greater than $t = 2^{-i}$. Then by the maximality of the matching computed by GHAFFARIMATCHING on the filtered subgraph, any edge not in that matching must satisfy one of the following criteria:

- it has weight at most 2^{-i} ,
- it has an endpoint in M ,
- it shares an endpoint with another edge in GHAFFARIMATCHING.

So after the new edges of in GHAFFARIMATCHING are added to M , M is maximal over the 2^{-i} -heavy edges as desired.

Now we claim that $\delta_e \geq w_e/2$ for any edge $e \in M' \setminus M$ of weight at least ε . This is because after the first round for which $t < w_e$, M' must be maximal over the t -heavy edges. This t is at least $w_e/2$, so if $e \notin M$, then either it shares an endpoint with some edge of

weight at least $w_e/2$ or its own weight is $\leq \varepsilon$. We then have

$$\begin{aligned}
w(M') &= w(M \cap M') + \sum_{e \in M' \setminus M} w_e \\
&\leq w(M \cap M') + \sum_{e \in M' \setminus M} \max(2\delta_e, \varepsilon) \\
&\leq w(M \cap M') + 2 \sum_{e \in M' \setminus M} \delta_e + \varepsilon N
\end{aligned}$$

We claim that $\sum_{e \in M' \setminus M} \delta_e \leq 2 \cdot w(M \setminus M')$. This is because each edge in $M \setminus M'$ shares an endpoint with at most 2 edges of $M' \setminus M$, otherwise M' would not be a matching. Therefore,

$$\begin{aligned}
w(M') &\leq w(M \cap M') + 4 \sum_{e \in M \setminus M'} w_e + \varepsilon N \\
&\leq 4 \cdot w(M) + \varepsilon N
\end{aligned}$$

By [Lemma 4.5.1](#), $w(M') = N \cdot \text{dist}_1(f, \text{mono})$; therefore $w(M) \geq N(\frac{1}{4}\text{dist}_1(f, \text{mono}) - \varepsilon)$ as desired.

We now bound the failure probability. When called with a random seed of length $\text{poly}(\log N, \log \log(1/\varepsilon))$ the algorithm of [\[64\]](#) can be made to succeed with probability $1 - (N^{-10}/\log(4/\varepsilon))$. We use the random seed on at most $\log(4/\varepsilon)$ different graphs, so by union bound, with probability $1 - N^{-10}$ all the calls succeed. By the same argument as in the proof of [Theorem 5](#), we may assume that $\log(1/\varepsilon) \leq N$, and so the randomness complexity is $\text{poly}(\Delta, \log N)$. $\square$

Lemma 4.5.3 (Running time and output size). *Let $P, f, \varepsilon, N, \Delta$, and r be as described in the lemma above. Let s_P be the size of the succinct representation of P , and s_f be the size of the succinct representation of f .*

Then $\text{MATCHVIOLATIONS}(P, f, \varepsilon, r)$ runs in time $(\Delta \log N)^{O(\log(1/\varepsilon))}(s_P + s_f)$ and outputs a representation of size $(\Delta \log N)^{O(\log(1/\varepsilon))}(s_P + s_f)$.

Proof. If $\varepsilon < 1/N$, then MATCHVIOLATIONS constructs and outputs a representation of the standard global greedy algorithm for 2-approximate maximum matching. The representation size of this algorithm is $O(\Delta N) \leq (\Delta \log N)^{O(\log(1/\varepsilon))}$, and the running time of MATCHVIOLATIONS is polynomial in this representation size.

If $\varepsilon \geq 1/N$, then by induction on the number of iterations i , we will show that the representation size of M at the start of iteration i is at most $(\Delta \log N)^{O(i)}(s_P + s_f)$. In the base case, we have an empty matching M which has constant representation size.

In the inductive case, suppose the claim holds at the start of iteration i . Then we set P' to be the function that applies FILTEREDGES to $TC(P)$. $TC(P)$ has size $O(\Delta \cdot s_P)$, as it makes $O(\Delta)$ calls to P . FILTEREDGES makes one call to $TC(P)$ and at most $O(\Delta)$ calls to

M and f . It also has overhead of size $O(\log t) = O(\log(1/\varepsilon)) = O(\log N)$. By the inductive hypothesis, the size of P' is then

$$\begin{aligned} & O(\Delta) \cdot (\Delta \log N)^{O(i)}(s_P + s_f) + O(\log N) + O(\Delta \cdot s_P) \\ & \leq (\Delta \log N)^{O(i+1)}(s_P + s_f). \end{aligned}$$

Then we set M to be the function that applies `GHAFFARIMATCHING` to P' . `GHAFFARIMATCHING` has constant overhead and makes $\text{poly}(\Delta, \log N)$ queries to P' . Then the new size of M is $\text{poly}(\Delta, \log N) \cdot (\Delta \log N)^{O(i)}(s_P + s_f) = (\Delta \log N)^{O(i+1)}(s_P + s_f)$.

The size bounds follow from the fact that there are $O(\log 1/\varepsilon)$ iterations. The corresponding running time bound for `MATCHVIOLATIONS` comes from the fact that since it only constructs the succinct representations, its running time in each iteration is polynomial in the size of the representations it constructs. □

Algorithm 17 `HYPERCUBEMATCHING`($f, \varepsilon, \mathbf{r}$)

- 1: **Given:** Function $f : \{-1, 1\}^d \rightarrow [-1, 1]$ given as succinct representation, weight threshold ε , random seed $\mathbf{r} = r_1 \circ \dots \circ r_{\lceil \log 2/\varepsilon \rceil}$
 - 2: **Output:** Succinct representation of a high-weight matching on the violating pairs w.r.t. f
 - 3: $P \leftarrow \text{TRUNCATEDCUBE}(d, \varepsilon)$
 - 4: $M \leftarrow$ representation of a function that takes x and outputs
 - 5: $\begin{cases} \text{MATCHVIOLATIONS}(P, f, \varepsilon, \mathbf{r}) & -\sqrt{2d \log 2/\varepsilon} \leq |x| \leq \sqrt{2d \log 2/\varepsilon} \\ \perp & \text{otherwise} \end{cases}$
 - 6: **return** M
-

Lemma 4.5.4. *With a random seed of length $2^{\tilde{O}(\sqrt{d} \log(1/\varepsilon))}$, [Algorithm 17](#) outputs a representation of a matching on the weighted violation graph $\text{viol}(f)$, of weight at least $2^d \cdot (\frac{1}{4} \text{dist}_1(f, \text{mono}) - 4\varepsilon)$, with probability at least $1 - 2^{-10d}$. The size of the representation is $2^{\tilde{O}(\sqrt{d} \log(1/\varepsilon))} \cdot s_f$, where s_f is the size of the representation of f .*

Proof. `HYPERCUBEMATCHING` calls `MATCHVIOLATIONS` on the truncated hypercube, which has parameters $N < 2^d$ and $\Delta = 2^{O(\sqrt{d} \log d \log(1/\varepsilon))}$. The size of the representation of `TRUNCATEDCUBE` is $O(d)$. So by [Lemma 4.5.3](#), the running time and output size of `HYPERCUBEMATCHING` are $2^{O(\sqrt{d} \log d \log(1/\varepsilon))} \cdot s_f$, and the random seed length is $2^{O(\sqrt{d} \log d \log(1/\varepsilon))}$.

Let f' be the restriction of f to the truncated cube. Since f is bounded in $[-1, 1]$ and the truncated cube covers all but an ε fraction of vertices, we have $\text{dist}_1(f', \text{mono}) \geq \text{dist}_1(f, \text{mono}) - 2\varepsilon$. By [Lemma 4.5.2](#), the weight of the matching is at least $(1 - \varepsilon) \cdot 2^d (\frac{1}{4} \text{dist}_1(f', \text{mono}) - \varepsilon) \geq (1 - \varepsilon) \cdot 2^d (\frac{1}{4} \text{dist}_1(f, \text{mono}) - 3\varepsilon/2) \geq 2^d (\frac{1}{4} \text{dist}_1(f, \text{mono}) - 4\varepsilon)$. □

4.6 Algorithm for Agnostic Learning

In this section we give descriptions of the agnostic learning algorithm and its major components (we will analyze the algorithms in the subsequent sections). The algorithm MONOTONE-LEARNER makes calls to

ELLIPSOIDALGORITHM, where the optimization domain is the $\leq d^{\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \rceil}$ -dimensional space of degree- $\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \rceil$ polynomials over $\mathbb{R}^d$, and the feasible region is the subset of such polynomials that are ε -close to monotone and α -close to f . Its output, P^{GOOD} , is such a polynomial (see Fact 2 for details). It also makes calls to HYPERCUBECORRECTOR, which is given in Corollary 10.

Algorithm 18 Algorithm MONOTONELEARNER (d, ε, T)

- 1: **Given:** Integer d , $\varepsilon \in (0, 1)$, and uniform sample access to an unknown function $f : \{\pm 1\}^d \rightarrow \{\pm 1\}$.
 - 2: **Output:** Circuit $\mathcal{C} : \{\pm 1\}^d \rightarrow \{\pm 1\}$.
 - 3: **for** $\alpha \in \{\varepsilon, 2\varepsilon, 3\varepsilon, \dots, 1 - \varepsilon, 1 + 200\varepsilon\}$ **do**
 - 4: OptimizationResult $\leftarrow$ ELLIPSOIDALGORITHM $\left(1, \varepsilon \cdot d^{-\frac{1}{2} \lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \rceil}, \text{ORACLE}_{\alpha, d, \varepsilon}\right)$.
 - 5: **if** OptimizationResult $\neq$ FAIL **then**
 - 6: $P^{\text{GOOD}} = \text{OptimizationResult}$
 - 7: $P^{\text{GOOD}}_{\text{TRIMMED}} \leftarrow$ representation of a function that takes input $\mathbf{x}$ and outputs the value

$$\begin{cases} P^{\text{GOOD}}(x) & \text{if } P^{\text{GOOD}}(x) \in [-1, +1] \\ 1 & \text{if } P^{\text{GOOD}}(x) > 1 \\ -1 & \text{if } P^{\text{GOOD}}(x) < -1 \end{cases}$$
 - 8: $P^{\text{GOOD}}_{\text{CORRECTED}} \leftarrow$ representation of a function that takes input $\mathbf{x}$ and returns the value

$$\text{HYPERCUBECORRECTOR}(x, P^{\text{GOOD}}_{\text{TRIMMED}}, r)$$
 - 9: $T \leftarrow \frac{200}{\varepsilon^2} \log\left(\frac{20}{\varepsilon}\right) \log(20d)$ i.i.d. pairs $(\mathbf{x}_i, f(\mathbf{x}_i))$, with $\mathbf{x}_i$ sampled uniformly from $\{-1, 1\}^d$.
 - 10: ThresholdCandidates $\leftarrow \left\{ \frac{1}{\varepsilon} \text{ i.i.d. uniformly random elements in } [-1, 1] \right\}$.
 - 11: $t^* := \operatorname{argmin}_{t \in \text{ThresholdCandidates}} \left[\frac{1}{|T|} \sum_{\mathbf{x} \in T} [|\operatorname{sign}(P^{\text{GOOD}}_{\text{CORRECTED}}(\mathbf{x}) - t) - f(\mathbf{x})|] \right]$
 - 12: **return** representation of a function that takes input $\mathbf{x}$ and returns the value

$$\operatorname{sign}(P^{\text{GOOD}}_{\text{CORRECTED}}(\mathbf{x}) - t^*)$$
-

The subroutine ORACLE takes as input a polynomial and provides the separating hyperplane required by ELLIPSOIDALGORITHM. It makes calls to HYPERCUBEMATCHING (see

Lemma 4.5.4), which provides a high-weight matching over the pairs of labels that violate monotonicity.

Algorithm 19 Subroutine $\text{ORACLE}_{\alpha,d,\varepsilon}(P)$

- 1: **Given:** $\varepsilon, \alpha \in (0, 1)$, degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial P over $\mathbb{R}^d$ with $\|P\|_2 \leq 1$, and uniform sample access to an unknown function $f : \{\pm 1\}^d \rightarrow \{\pm 1\}$.
 - 2: **Output:** "Yes" or ("No", $Q_{\text{separator}}$), where $Q_{\text{separator}}$ is a degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial over $\mathbb{R}^d$.
 - 3: $P_{\text{TRIMMED}} \leftarrow$ representation of a function that takes input $\mathbf{x}$ and outputs

$$\begin{cases} P(\mathbf{x}) & \text{if } P(\mathbf{x}) \in [-1, +1] \\ 1 & \text{if } P(\mathbf{x}) > 1 \\ -1 & \text{if } P(\mathbf{x}) < -1 \end{cases}.$$
 - 4: $T \leftarrow$ set of $d^{\frac{C\sqrt{d}}{\varepsilon} \log \frac{1}{\varepsilon}}$ i.i.d. pairs $(\mathbf{x}_i, f(\mathbf{x}_i))$, with $\mathbf{x}_i$ sampled uniformly from $\{-1, 1\}^d$ (for sufficiently large constant C).
 - 5: $r \leftarrow$ string of $2^{C\sqrt{d}(\log d \cdot \log \frac{1}{\varepsilon})^C}$ random i.i.d. bits (for sufficiently large constant C).
 - 6: $M_{\text{separator}} \leftarrow$ representation of a function that takes input x and outputs

$$\begin{cases} 0 & \text{if } \text{HYPERCUBEMATCHING}(P_{\text{TRIMMED}}, \varepsilon/4, r) \text{ does not match } x \text{ to any other vertex} \\ 1 & \text{if } \text{HYPERCUBEMATCHING}(P_{\text{TRIMMED}}, \varepsilon/4, r) \text{ matches } x \text{ to some vertex } z, \text{ s.t. } z \preceq x \\ -1 & \text{if } \text{HYPERCUBEMATCHING}(P_{\text{TRIMMED}}, \varepsilon/4, r) \text{ matches } x \text{ to some vertex } z, \text{ s.t. } z \succeq x \end{cases}$$
 - 7: **if** $\frac{1}{|T|} \sum_{\mathbf{x} \in T} [M_{\text{separator}}(\mathbf{x}) \cdot P_{\text{TRIMMED}}(\mathbf{x})] > 5\varepsilon$ **then**
 - 8: $Q_{\text{separator}} \leftarrow \sum_{S \subseteq [d]: |S| \leq \left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \left(\frac{1}{|T|} \sum_{\mathbf{x} \in T} [M_{\text{separator}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})] \right) \chi_S$
 - 9: **return** ("No", $Q_{\text{separator}}$)
 - 10: **else if** $\frac{1}{|T|} \sum_{\mathbf{x} \in T} [|f(\mathbf{x}) - P(\mathbf{x})|] > \alpha + 50\varepsilon$ **then**
 - 11: $Q_{\text{separator}} \leftarrow \sum_{S \subseteq [d]: |S| \leq \left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \left(\mathbb{E}_{\mathbf{x} \sim T} \left[\hat{P}(S) \chi_S(\mathbf{x}) \text{sign}(P(\mathbf{x}) - f(\mathbf{x})) \right] \right) \chi_S$
 - 12: **return** ("No", $Q_{\text{separator}}$)
 - 13: **else**
 - 14: **return** "Yes"
-

The algorithm `MATCHVIOLATIONS` finds a high-weight matching on the violation graph of a poset. It is the main component of `HYPERCUBEMATCHING`, which is just a wrapper that calls `MATCHVIOLATIONS` on the truncated cube. `FILTEREDGES` removes vertices that are either incident to M or have weight below the threshold t , and `GHAFFARIMATCHING` is the maximal matching algorithm of Corollary 3. More implementation details and analysis are given in Section 4.5.

Algorithm 20 MATCHVIOLATIONS($P, f, \varepsilon, \mathbf{r} = r_1 \circ \dots \circ r_{\lceil \log 2/\varepsilon \rceil}$)

Given: Poset P and function $f : P \rightarrow [-1, 1]$ given as succinct representations, weight threshold ε , random seed $\mathbf{r} = r_1 \circ \dots \circ r_{\lceil \log 2/\varepsilon \rceil}$

Output: Succinct representation of a high-weight matching on the violating pairs of P w.r.t. f

if $\varepsilon < 1/|P|$ **then**

$M \leftarrow$ representation of the greedy algorithm that adds each edge (x, y) of $TC(P)$ in decreasing order of $f(x) - f(y)$.

else

$t \leftarrow 2$

$i \leftarrow 1$

$M \leftarrow$ representation of a function computing the empty matching

while $t > \varepsilon/2$ **do**

$P' \leftarrow$ representation of a function that takes input x and outputs

 FILTEREDGES($TC(P), f, t, M, x$)

$M \leftarrow$ representation of a function that takes input x and outputs $M(x)$ if $M(x) \neq \perp$, otherwise GHAFFARIMATCHING(P', r_i, x)

$t \leftarrow t/2$

$i \leftarrow i + 1$

return M

The following is the core of HYPERCUBECORRECTOR, given as a “global overview” for convenience. Analysis and local implementation are given in [Section 4.4](#). The algorithm corrects monotonicity of a k -valued function over a poset. HYPERCUBECORRECTOR is a wrapper that discretizes a real-valued function and then calls this corrector with the truncated hypercube as the poset.

Algorithm 21 Global view of sorting k -valued labels in a poset

1: **Given:** Poset P of height h , function $f : P \rightarrow [k]$

2: **Output:** monotone function $g : P \rightarrow [k]$

3: Let $i \leftarrow 0$

4: **for** $0 \leq i \leq \lceil \log k \rceil$ **do**

5: Let f_i be the projection of f onto the i_{th} most significant bit of k , i.e. $f_i(x) = 1$ if the i_{th} bit of $f(x)$ is 1.

6: Let P_i be the poset on the elements of P with the relation

$$x \prec_{P_i} y := x \prec_P y \text{ and } f_j(x) = f_j(y) \text{ for all } j < i.$$

7: Let $\pi_i \leftarrow$ BOOLEANCORRECTOR(f_i, P_i).

8: Let $f \leftarrow f\pi_i$.

9: **return** f

4.7 Analysis of the Agnostic Learning Algorithm

By inspecting algorithm MONOTONELEARNER (i.e. Algorithm 18 on page 103), we see immediately that the run-time is $2^{\tilde{O}(\sqrt{d}/\varepsilon)}$. We proceed to argue that the algorithm indeed satisfies the guarantee of Theorem 4. First, we will need the following standard proposition.

Claim 15. *For any positive integers d and r , real $\varepsilon, \delta \in (0, 1)$, and any function $f : \{\pm 1\}^d \rightarrow [-1, 1]$, let T be a collection of at least $d^{5r} \cdot \frac{100}{\varepsilon^2} \ln \frac{1}{\varepsilon} \ln \frac{1}{\delta}$ i.i.d. uniformly random elements of $\{\pm 1\}^d$. Then, with probability at least $1 - \delta$*

$$\max_{\substack{\text{degree-}r \text{ polynomial } P \text{ over } \{\pm 1\}^d \\ \text{with } \|P\|_2 \leq 1}} \|f - P\|_1 - \mathbb{E}_{\mathbf{x} \sim T} [|f(\mathbf{x}) - P(\mathbf{x})|] \leq \varepsilon,$$

Proof. First we bound the probability that the condition above holds for one specific P with $\|P\|_2 \leq 1$. The condition $\|P\|_2 \leq 1$ implies that $\max_{\mathbf{x} \in \{\pm 1\}^d} |P(\mathbf{x})| \leq d^r$. This implies, via the Hoeffding bound, that

$$\Pr_{\text{choice of } T} \left[\left| \|f - P\|_1 - \mathbb{E}_{\mathbf{x} \sim T} [|f(\mathbf{x}) - P(\mathbf{x})|] \right| > \frac{\varepsilon}{4} \right] \leq \exp \left(-\frac{\varepsilon^2 |T|}{32 d^{2r}} \right).$$

We now move on to bounding the maximum over all degree- r polynomials P over $\{\pm 1\}^d$ with $\|P\|_2 \leq 1$. We will need a collection $\mathcal{C}$ of degree r polynomials over $\{\pm 1\}^d$, such that $|\mathcal{C}| \leq \exp(d^r \ln \frac{8d^r}{\varepsilon})$ so for every degree r polynomial P with $\|P\|_2 \leq 1$ there is some element $P_{\text{closest}} \in \mathcal{C}$ for which it is the case that

$$\max_{\mathbf{x} \in \{\pm 1\}^d} |P(\mathbf{x}) - P_{\text{closest}}(\mathbf{x})| \leq \frac{\varepsilon}{4}.$$

Also, the L_2 norm of every element in $\mathcal{C}$ is at most 1. Such a set can be constructed by putting into $\mathcal{C}$ all polynomials of the form $\sum_{\substack{S \subseteq [d] \\ |S| \leq r}} c_S (\chi_S(x))$ with the coefficients c_S taking values in $[-1, +1]$ rounded to the nearest multiple of $\frac{\varepsilon}{8d^r}$, while discarding the polynomials whose L_2 norm is larger than 1. This way, since $\chi_S(x) \in \{\pm 1\}$, when we round the coefficients of P to a multiple of $\frac{\varepsilon}{8d^r}$ the value at any $\mathbf{x} \in \{\pm 1\}^d$ cannot change by more than $\frac{\varepsilon}{4}$, as there are at most d^r contributing monomials³. The total number of such polynomials is at most $\left(\frac{8d^r}{\varepsilon}\right)^{d^r} = e^{d^r \ln \frac{8d^r}{\varepsilon}}$.

Now, by taking a union bound on all elements of $\mathcal{C}$ we get

$$\Pr_{\text{choice of } T} \left[\max_{P \in \mathcal{C}} \left| \|f - P\|_1 - \mathbb{E}_{\mathbf{x} \sim T} [|f(\mathbf{x}) - P(\mathbf{x})|] \right| \leq \frac{\varepsilon}{2} \right] \geq 1 - \exp \left(-\frac{\varepsilon^2 |T|}{32 d^{2r}} + d^r \ln \frac{8d^r}{\varepsilon} \right)$$

³To have $\|P_{\text{closest}}(\mathbf{x})\|_2 \leq \|P\|_2 \leq 1$ we should round to the closest multiple of $\frac{\varepsilon}{8d^r}$ that is smaller in the absolute value of the coefficient being rounded

Finally, if the above holds, by choosing a polynomial P_{closest} from $\mathcal{C}$ to minimize $\max_{\mathbf{x} \in \{\pm 1\}^d} |P(\mathbf{x}) - P_{\text{closest}}(\mathbf{x})|$ we get that

$$\Pr_{\text{choice of } T} \left[\max_{\substack{\text{degree-}r \text{ polynomial } P \text{ over } \{\pm 1\}^d \\ \text{with } \|P\|_2 \leq 1}} \left| \|f - P\|_1 - \mathbb{E}_{\mathbf{x} \sim T} [|f(\mathbf{x}) - P(\mathbf{x})|] \right| \leq \varepsilon \right] \geq 1 - \exp \left(-\frac{\varepsilon^2 |T|}{8 d^{2r}} + d^r \ln \frac{4d^r}{\varepsilon} \right).$$

Substituting $|T| \geq d^{5r} \frac{100}{\varepsilon^2} \ln \frac{1}{\varepsilon} \ln \frac{1}{\delta}$ we see that the above expression is at least $1 - \delta$. $\square$

Now, in the following lemma we prove that subroutine **Oracle** $_{\alpha, d, \varepsilon}(P)$ (i.e. [Algorithm 19](#) on page 104) satisfies some precise specifications with high probability. Informally, we show that **Oracle** $_{\alpha, d, \varepsilon}(P)$ either

- Certifies that the polynomial P is both close to monotone in L_1 distance and has L_1 prediction error of $\alpha + O(\varepsilon)$.
- Outputs a hyperplane separating P from all such polynomials.

Formally, we prove the following:

Lemma 4.7.1. *For sufficiently large constant C in [Section 4.6](#) and [Section 4.6](#) of procedure **Oracle** $_{\alpha, d, \varepsilon}(P)$, sufficiently large integer d , any function $f : \{-1, 1\}^d \rightarrow \{-1, 1\}$, parameters $\varepsilon, \alpha \in (0, 1)$, and a degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial P satisfying $\|P\|_2 \leq 1$ the following is true.*

*The procedure **Oracle** $_{\alpha, d, \varepsilon}(P)$ runs in time $d^{\tilde{O}(\frac{\sqrt{d}}{\varepsilon})}$ and will with probability at least $1 - \frac{1}{2^{5d}}$ conform to the following specification:*

1. *If **Oracle** $_{\alpha, d, \varepsilon}(P)$ outputs “yes”, then:*

$$(a) \text{ The function } P_{\text{TRIMMED}} = \begin{cases} 1 & \text{if } P(x) > 1, \\ -1 & \text{if } P(x) < -1, \\ P(\mathbf{x}) & \text{otherwise.} \end{cases}$$

is 100ε -close to monotone in L_1 norm.

- (b) *The L_1 distance between P and the function f is at most $\alpha + 100\varepsilon$.*

2. *If **Oracle** $_{\alpha, d, \varepsilon}(P)$ instead outputs (“No”, $Q_{\text{separator}}$), where $Q_{\text{separator}}$ is a degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial over $\mathbb{R}^d$, then we have $\langle P', Q_{\text{separator}} \rangle < \langle P, Q_{\text{separator}} \rangle$ for any degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial P' with $\|P'\|_2 \leq 1$ that satisfies the following two conditions:*

- P' is ε -close in L_1 distance to some monotone function $f_{\text{monotone}} : \{\pm 1\}^d \rightarrow [-1, 1]$ and

- P' is $(\alpha + \varepsilon)$ -close in L_1 distance to the function f which we are trying to learn.

In particular, this implies that if P itself is ε -close in L_1 distance to some monotone function and is $(\alpha + \varepsilon)$ -close in L_1 distance to the function f , then $\mathbf{Oracle}_{\alpha,d,\varepsilon}(P)$ will say “yes” with probability at least $1 - \frac{1}{2^{10d}}$.

Proof. We use the union bound to conclude that with probability at least $1 - \frac{1}{2^{5d}}$ all the following events hold:

- The LCA from [Lemma 4.5.4](#) works as advertised and the weight W of the resulting matching satisfies

$$\frac{W}{2^d} \geq 0.1 \text{dist}_1(P_{\text{TRIMMED}}, \text{mono}) - \varepsilon.$$

Another way to write the same thing is

$$\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle \geq 0.1 \text{dist}_1(P_{\text{TRIMMED}}, \text{mono}) - \varepsilon. \quad (4.1)$$

From [Lemma 4.5.4](#) it follows that this holds with probability at least $1 - \frac{1}{2^{10d}}$.

- The estimate of $\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle$ in [Section 4.6](#) is indeed ε -close to the true value. From the standard Hoeffding bound, this holds with probability at least $1 - \frac{1}{2^{10d}}$.
- It is the case that

$$\left\| \sum_{S \subset [d]: |S| \leq \left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \widehat{M_{\text{separator}}}(S) \chi_S - Q_{\text{separator}} \right\|_2 \leq \varepsilon$$

Substituting the expression for $Q_{\text{separator}}$, and using the orthogonality of $\{\chi_S\}$ we see this is equivalent to

$$\sum_{S \subset [d]: |S| \leq \left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \underbrace{\left(\widehat{M_{\text{separator}}}(S) - \frac{1}{|T|} \sum_{\mathbf{x} \in T} [M_{\text{separator}}(\mathbf{x}) \cdot \chi_S(\mathbf{x})] \right)^2}_{\leq \varepsilon d^{-\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \text{ in absolute value w.p. } \geq \frac{1}{2^{10d}} \text{ via Hoeffding's bound}} \leq \varepsilon$$

Overall, the above holds with probability at least $1 - \frac{1}{2^{9d}}$ by taking a Hoeffding bound for each individual summand and taking a union bound over them.

- The set $T \subset \{\pm 1\}^d$ is such that

$$\max_{\substack{\text{degree-}\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil \\ \text{polynomial } P' \text{ over } \{\pm 1\}^d \\ \text{with } \|P'\|_2 \leq 1}} \left| \|f - P'\|_1 - \mathbb{E}_{(x, f(x)) \sim T} [|f(\mathbf{x}) - P'(\mathbf{x})|] \right| \leq \varepsilon. \quad (4.2)$$

It follows from [Claim 15](#) that this happens with probability at least $1 - \frac{1}{2^{10d}}$.

Now, we argue that if these conditions indeed hold, then $\mathbf{Oracle}_{\alpha,d,\varepsilon}(P)$ will satisfy the specification given.

First, suppose $\mathbf{Oracle}_{\alpha,d,\varepsilon}(P)$ answered “yes”. Then, since the estimate of $\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle$ in [Section 4.6](#) is within ε of its true value, we have

$$\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle \leq 6\varepsilon.$$

Now, since we are assuming the matching LCA from [Lemma 4.5.4](#) works as advertised, this means that

$$6\varepsilon \geq \langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle \geq 0.1 \cdot \text{dist}_1(P_{\text{TRIMMED}}, \text{mono}) - \varepsilon$$

which can be rewritten as

$$\text{dist}_1(P_{\text{TRIMMED}}, \text{mono}) \leq 70\varepsilon \leq 100\varepsilon,$$

which is one of the two things we wanted to show. The other one was showing that the L_1 distance between P and the function f , which we are trying to learn, is at most $\alpha + 100\varepsilon$. Since the algorithm returned “yes”, it has to be that in [Section 4.6](#) we have

$$\mathbb{E}_{\mathbf{x} \sim T} [|f(\mathbf{x}) - P(\mathbf{x})|] \leq \alpha + 50\varepsilon.$$

From [Equation \(4.2\)](#) it then follows that

$$\|f - P\|_1 \leq \mathbb{E}_{\mathbf{x} \sim T} [|f(\mathbf{x}) - P(\mathbf{x})|] + \varepsilon \leq \alpha + 51\varepsilon \leq \alpha + 100\varepsilon,$$

which is the other condition we wanted to show for the case when the oracle says “yes”.

Now, assume the oracle outputs “no” along with some polynomial $Q_{\text{separator}}$ and let P' be a degree $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial with $\|P'\|_2 \leq 1$ that satisfies the following two conditions⁴:

- P' is ε -close in L_1 distance to some monotone function $f_{\text{monotone}} : \{\pm 1\}^d \rightarrow [-1, 1]$ and
- P' is $(\alpha + \varepsilon)$ -close in L_1 distance to the function f which we are trying to learn.

Here, again, there are two cases. First, suppose we have the case where $Q_{\text{separator}}$ is generated from $M_{\text{separator}}$. We have that the oracle’s estimate of $\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle$ is at least 5ε , which means that $\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle \geq 4\varepsilon$. We know that P' is ε -close in L_1 distance to some monotone function $f_{\text{monotone}} : \{\pm 1\}^d \rightarrow [-1, 1]$. Since $M_{\text{separator}}$ is defined to be so for every matched pair $(\mathbf{x}_i, \mathbf{y}_i)$ with $\mathbf{x}_i \prec \mathbf{y}_i$ we have $M_{\text{separator}}(\mathbf{x}_i) = 1$ and $M_{\text{separator}}(\mathbf{y}_i) = -1$ and is 0 otherwise, and for each such pair $f_{\text{monotone}}(\mathbf{x}_i) \leq f_{\text{monotone}}(\mathbf{y}_i)$ we have $\langle M_{\text{separator}}, f_{\text{monotone}} \rangle \leq 0$. This allows us to conclude

$$\begin{aligned} 0 &\geq \langle M_{\text{separator}}, f_{\text{monotone}} \rangle = \langle M_{\text{separator}}, P' \rangle + \langle M_{\text{separator}}, f_{\text{monotone}} - P' \rangle \geq \\ &\quad \langle M_{\text{separator}}, P' \rangle - \left(\max_{x \in \{-1, 1\}^d} |M_{\text{separator}}(x)| \right) \|f_{\text{monotone}} - P'\|_1 \geq \langle M_{\text{separator}}, P' \rangle - \varepsilon, \end{aligned}$$

⁴If no polynomial satisfying these conditions exists, the statement we are seeking to prove holds vacuously.

which means

$$\begin{aligned} \varepsilon &\geq \langle M_{\text{separator}}, P' \rangle = \left\langle \sum_{S \subset [d]: |S| \leq \left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \widehat{M_{\text{separator}}}(S) \left(\prod_{i \in S} x_i \right), P' \right\rangle = \\ &\langle Q_{\text{separator}}, P' \rangle - \left\| Q - \sum_{S \subset [d]: |S| \leq \left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \widehat{M_{\text{separator}}}(S) \left(\prod_{i \in S} x_i \right) \right\|_2 \|P'\|_2 \geq \langle Q_{\text{separator}}, P' \rangle - \varepsilon. \end{aligned} \quad (4.3)$$

On the other hand, the oracle's estimate of $\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle$ is at least 5ε , which means that it is the case that $\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle \geq 4\varepsilon$. This allows us to conclude

$$\begin{aligned} 4\varepsilon &\leq \overbrace{\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle}^{\text{Trimming the values of a function only decreases weights of violated edges.}} \leq \langle M_{\text{separator}}, P \rangle = \left\langle \sum_{S \subset [d]: |S| \leq \left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \widehat{M_{\text{separator}}}(S) \left(\prod_{i \in S} x_i \right), P \right\rangle \leq \\ &\langle Q_{\text{separator}}, P \rangle + \left\| Q - \sum_{S \subset [d]: |S| \leq \left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \widehat{M_{\text{separator}}}(S) \left(\prod_{i \in S} x_i \right) \right\|_2 \|P\|_2 \geq \langle Q_{\text{separator}}, P \rangle + \varepsilon. \end{aligned} \quad (4.4)$$

Combining Equation 4.4 and Equation 4.3 we get

$$\langle Q_{\text{separator}}, P' \rangle \leq 2\varepsilon < 3\varepsilon \leq \langle Q_{\text{separator}}, P \rangle$$

as required.

Finally, we consider the case when $Q_{\text{separator}}$ is generated on Section 4.6. Since P' is $(\alpha + \varepsilon)$ -close in L_1 distance to the function f , by Equation (4.2) we have that

$$\alpha + \varepsilon \leq \|f(\mathbf{x}) - P'(\mathbf{x})\|_1 \leq \mathbb{E}_{(\mathbf{x}, f(\mathbf{x})) \sim T} [|f(\mathbf{x}) - P'(\mathbf{x})|] - \varepsilon,$$

which we can rewrite as $\mathbb{E}_{(\mathbf{x}, f(\mathbf{x})) \sim T} [|f(\mathbf{x}) - P'(\mathbf{x})|] \leq \alpha + 2\varepsilon$. At the same time, we have $\mathbb{E}_{(\mathbf{x}, f(\mathbf{x})) \sim T} [|f(\mathbf{x}) - P(\mathbf{x})|] > \alpha + 50\varepsilon$, which means that

$$\mathbb{E}_{(\mathbf{x}, f(\mathbf{x})) \sim T} [|f(\mathbf{x}) - P(\mathbf{x})|] > \mathbb{E}_{(\mathbf{x}, f(\mathbf{x})) \sim T} [|f(\mathbf{x}) - P'(\mathbf{x})|].$$

Therefore, as the function mapping a polynomial H to the value $\mathbb{E}_{(\mathbf{x}, f(\mathbf{x})) \sim T} [|f(\mathbf{x}) - H(\mathbf{x})|]$

is convex, it has to be the case that⁵

$$\begin{aligned} \left\langle P' - P, \sum_{S \subset [d] : |S| \leq \left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \left(\mathbb{E}_{\mathbf{x} \sim T} \left[\widehat{P}(S) \chi_S(\mathbf{x}) \text{sign}(P(\mathbf{x}) - f(\mathbf{x})) \right] \right) \chi_S \right\rangle = \\ = \left\langle P' - P, \nabla_H \left(\mathbb{E}_{(\mathbf{x}, f(\mathbf{x})) \sim T} [|H(\mathbf{x}) - f(\mathbf{x})|] \right) \Big|_{H=P} \right\rangle < 0. \end{aligned}$$

This implies that $\langle Q_{\text{separator}}, P' \rangle \leq \langle Q_{\text{separator}}, P \rangle$, which completes the proof. $\square$

4.7.1 Finishing the Proof of the Main Theorem (Theorem 4)

Recall that earlier by inspecting Algorithm 18 we concluded that this algorithm runs in time $2^{\tilde{O}(\frac{\sqrt{d}}{\varepsilon})}$. Here we use Lemma 4.7.1 to finish the proof of Theorem 4 by showing that with probability at least $1 - \frac{1}{2^d}$ the function $\text{sign}(P_{\text{TRIMMED}}^{\text{GOOD}}(\mathbf{x}) - t^*)$ is monotone and is $\text{opt} + O(\varepsilon)$ -close to f (where opt is the distance of f to the closest monotone function).

We can further conclude that with probability at least $1 - \frac{1}{2^{3d}}$ the following events hold:

1. Every time an oracle $\mathbf{Oracle}_{\alpha, d, \varepsilon}$ is invoked (for various values of α), its behavior will conform to the specifications in Lemma 4.7.1.
2. The algorithm HypercubeCorrector from Corollary 10 used on line 11 works as advertised, so the function $P_{\text{CORRECTED}}^{\text{GOOD}} : \{\pm 1\} \rightarrow [-1, 1]$ is monotone and we indeed have

$$\|P_{\text{CORRECTED}}^{\text{GOOD}} - P_{\text{TRIMMED}}^{\text{GOOD}}\|_1 \leq 10 \cdot \text{dist}_1(P_{\text{TRIMMED}}^{\text{GOOD}}, \text{mono}) + \varepsilon. \quad (4.5)$$

3. In step (4), the function $\text{sign}(P_{\text{CORRECTED}}^{\text{GOOD}}(\mathbf{x}) - t^*)$ satisfies the guarantee from Fact 20, i.e.

$$\Pr_{\mathbf{x} \sim \{\pm 1\}^d} [\text{sign}(P_{\text{CORRECTED}}^{\text{GOOD}}(\mathbf{x}) - t^*) \neq f] \leq \frac{1}{2} \|P_{\text{CORRECTED}}^{\text{GOOD}} - f\|_1 + \varepsilon \quad (4.6)$$

We argue that each of these events takes place with probability at least $1 - \frac{1}{2^{4d}}$:

- Note that the oracles $\mathbf{Oracle}_{\alpha, d, \varepsilon}$ for various values of α are invoked at most $2^{\tilde{O}(\frac{\sqrt{d}}{\varepsilon})}$ times. Therefore, Lemma 4.7.1 tells us that for each of this invocations the algorithm $\mathbf{Oracle}_{\alpha, d, \varepsilon}$ conforms to its specification with probability at least $1 - \frac{1}{2^{5d}}$. Via union bound we see that event (1) holds with probability at least⁶ $1 - \frac{1}{2^{4d}}$.

⁵To be fully precise, the expression above is a subgradient of the convex function mapping a polynomial H to $\mathbb{E}_{(\mathbf{x}, f(\mathbf{x})) \sim T} [|f(\mathbf{x}) - H(\mathbf{x})|]$.

⁶We assume that ε is such that $2^{0.1d}$ exceeds the number $2^{\tilde{O}(\frac{\sqrt{d}}{\varepsilon})}$ of times that $\mathbf{Oracle}_{\alpha, d, \varepsilon}$ is invoked (for different values of α). Otherwise, the run-time budget is sufficient to store entire truth-tables of functions over $\{-1, 1\}^d$ and statement in Algorithm 15 is achieved by the trivial algorithm that uses a linear program to fit the best monotone real-valued function and then rounds it to be $\{-1, 1\}$ -valued. See Section 4.9.1 for further details.

- Event (2) holds with probability at least $1 - \frac{1}{2^{4d}}$ via [Corollary 10](#).
- Event (3) holds with probability at least $1 - \frac{1}{2^{4d}}$ via [Fact 20](#)

Via union bound, we see that with probability at least $1 - \frac{1}{2^{3d}}$ all these events hold, which we will assume for the rest of the proof.

Recall that opt stands for the distance of f to the closest monotone function. We first claim that the algorithm will break out of the loop in [Section 4.6](#) for some value $\alpha^* \leq 2\text{opt} + 150\varepsilon$, which we argue as follows: If $\alpha^* > 2\text{opt} + 150\varepsilon$, then for some⁷ $\alpha \in [2\text{opt} + 100\varepsilon, 2\text{opt} + 150\varepsilon]$ the ellipsoid algorithm failed to find some polynomial P on which **Oracle** $_{\alpha,d,\varepsilon}$ returns “Yes”. We claim that this is impossible. Indeed, let $\mathcal{C}_{\text{convex}}$ be the set consisting of degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomials P' with $\|P'\|_2 \leq 1$ that satisfies the following two conditions:

- P' is ε -close in L_1 distance to some monotone function $f_{\text{monotone}} : \{\pm 1\}^d \rightarrow [-1, 1]$, and
- P' is $(\alpha + \varepsilon)$ -close in L_1 distance to the function f which we are trying to learn.

We make the following observations:

- The set $\mathcal{C}_{\text{convex}}$ is a convex set, because (a) the set of all monotone functions $f_{\text{monotone}} : \{\pm 1\}^d \rightarrow [-1, 1]$ is convex, (b) the set of points $(\alpha + \varepsilon)$ -close in L_1 distance to some specific convex set is itself convex, and (c) the intersection of two convex sets is a convex set (in this case one convex set is the set functions $\{\pm 1\}^d \rightarrow [-1, 1]$ that are $(\alpha + \varepsilon)$ -close in L_1 distance a monotone functions and the other convex set is the set of all degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomials with $\|P'\|_2 \leq 1$).
- The set $\mathcal{C}_{\text{convex}}$ contains an L_2 ball of radius at least $\varepsilon \cdot d^{-\frac{1}{2}} \left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$. In other words, in $\mathcal{C}_{\text{convex}}$ there is some degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial P_0 such that any degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial P' that is ε -close to P_0 in L_2 norm is also in $\mathcal{C}_{\text{convex}}$. Let $f_{\text{monotone, optimal}} : \{\pm 1\}^d \rightarrow \{\pm 1\}$ be the monotone function for which it is the case that $\Pr_{\mathbf{x} \sim \{\pm 1\}^d} [f_{\text{monotone, optimal}}(\mathbf{x}) \neq f(\mathbf{x})] = \text{opt}$, and let P_0 be a degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial that is ε -close to $f_{\text{monotone, optimal}}$ in L_1 norm (such polynomial has to exist by [Fact 1](#)). Then, P_0 is $(2\text{opt} + \varepsilon)$ -close to f in L_1 norm and ε -close to monotone in L_1 norm. In other words, the set $\mathcal{C}_{\text{convex}}$ contains an L_1 -ball of radius ε . Via the standard inequality between the L_1 and L_2 norms, in d dimensions every L_1 ball of radius ε contains an L_2 ball of radius at most $\varepsilon/\sqrt{d}$. Our claim follows, since the space of degree- $\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ over $\mathbb{R}^d$ has dimension at most $d^{\left\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil}$.

⁷Note that $\text{opt} \leq 1/2$, because the function f is at least $1/2$ -close to either the all-ones or all-zeroes functions, which are both monotone. Therefore some value of α in the range $[2\text{opt} + 100\varepsilon, 2\text{opt} + 150\varepsilon]$ is necessarily considered by the algorithm as it is trying all values $\alpha = \varepsilon, 2\varepsilon, 3\varepsilon, \dots, 1 - \varepsilon, 1 + 200\varepsilon$.

- Since the procedure $\text{Oracle}_{\alpha,d,\varepsilon}$ is assumed to satisfy the specifications given in [Lemma 4.7.1](#) and for this specific value of α it never gave the response “yes”, then for every query P to $\text{Oracle}_{\alpha,d,\varepsilon}$, the oracle returned some halfspace that separates P from the convex set $\mathcal{C}_{\text{convex}}$.

From [Fact 2](#) we know that under these conditions the ellipsoid algorithm will necessarily in time

$\text{poly}\left(d^{\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \rceil}, \log(R/r)\right) = d^{O(\lceil \frac{4\sqrt{d}}{\varepsilon} \log \frac{4}{\varepsilon} \rceil)}$ find some polynomial P that is in $\mathcal{C}_{\text{convex}}$. For this particular polynomial, the specifications in [Lemma 4.7.1](#) require the oracle $\text{Oracle}_{\alpha,d,\varepsilon}$ to give a response “yes”, which gives us a contradiction. Thus, the function $P_{\text{TRIMMED}}^{\text{GOOD}}$ will be $O(\varepsilon)$ -close to monotone in L_1 norm and will satisfy

$$\|P_{\text{TRIMMED}}^{\text{GOOD}} - f\|_1 \leq 2\text{opt} + O(\varepsilon).$$

Combining this with [Equation \(4.5\)](#) yields

$$\|P_{\text{CORRECTED}}^{\text{GOOD}} - f\|_1 \leq 2\text{opt} + O(\varepsilon) + \|P_{\text{TRIMMED}}^{\text{GOOD}} - P_{\text{CORRECTED}}^{\text{GOOD}}\|_1 = 2\text{opt} + O(\varepsilon).$$

We know that $\|P_{\text{TRIMMED}}^{\text{GOOD}} - P_{\text{CORRECTED}}^{\text{GOOD}}\|_1 \leq O(\varepsilon)$ because $P_{\text{TRIMMED}}^{\text{GOOD}}$ is $O(\varepsilon)$ -close to monotone by [Equation \(4.5\)](#). Now, combining the inequality above with [Equation 4.6](#) gives us

$$\Pr_{\mathbf{x} \sim \{\pm 1\}^d} [\text{sign}(P_{\text{CORRECTED}}^{\text{GOOD}}(\mathbf{x}) - t^*) \neq f] \leq \frac{1}{2} \|P_{\text{CORRECTED}}^{\text{GOOD}} - f\|_1 + \varepsilon \leq \text{opt} + O(\varepsilon).$$

Finally, we see that since the function $P_{\text{CORRECTED}}^{\text{GOOD}} \{\pm 1\}^d \rightarrow [-1, +1]$ is monotone we have that the $\{\pm 1\}$ -valued function $\text{sign}(P_{\text{CORRECTED}}^{\text{GOOD}}(\mathbf{x}) - t^*)$ is also monotone, which finishes our argument.

4.8 Rounding of Real-Valued Functions to Boolean

Fact 20. Suppose we have two functions $g : \{\pm 1\}^d \rightarrow \mathbb{R}$ and $f : \{\pm 1\}^d \rightarrow \{\pm 1\}$. Let T be a set of at least $\frac{40}{\varepsilon^2} \log\left(\frac{20}{\varepsilon\delta} \log \frac{1}{\delta}\right)$ i.i.d. uniformly random elements of $\{-1, 1\}^d$, and let $\text{ThresholdCandidates} \subset [-1, 1]$ be a set of $\frac{20}{\varepsilon} \log \frac{1}{\delta}$ i.i.d. uniformly random elements of $[-1, 1]$. Let

$$t^* := \underset{t \in \text{ThresholdCandidates}}{\text{argmin}} \frac{1}{|T|} \sum_{\mathbf{x} \in T} |\text{sign}(g(\mathbf{x}) - t) - f(\mathbf{x})|$$

Then, with probability at least $1 - \delta$ it is the case that

$$\Pr_{\mathbf{x} \sim \{\pm 1\}^d} [\text{sign}(g(\mathbf{x}) - t^*) \neq f] \leq \frac{1}{2} \|f - g\|_1 + \varepsilon$$

Proof. We get that

$$\mathbb{E}_{t \sim [-1, 1]} \left[\mathbb{E}_{\mathbf{x} \sim \{\pm 1\}^d} [|\text{sign}(g(\mathbf{x}) - t) - f(\mathbf{x})|] \right] \leq \|f - g\|_1$$

directly via linearity of expectation. Now, the random variable $\mathbb{E}_{\mathbf{x} \sim \{\pm 1\}^d} [|\text{sign}(g(\mathbf{x}) - t) - f(\mathbf{x})|]$ (with randomness taken over t) is always in $[0, 2]$ and has some expectation $E \in [0, 2]$ which is at most $\|f - g\|_1$. By Markov's inequality, we have

$$\Pr_{t \sim [-1, 1]} \left[\mathbb{E}_{\mathbf{x} \sim \{\pm 1\}^d} [|\text{sign}(g(\mathbf{x}) - t) - g(\mathbf{x})|] \geq E + \varepsilon/2 \right] \leq \frac{E}{E + \varepsilon/2} \leq \frac{2}{2 + \varepsilon/2} \leq 1 - \frac{\varepsilon}{4}.$$

Since the set `ThresholdCandidates` consists of $\frac{20}{\varepsilon} \log \frac{1}{\delta}$ i.i.d. uniform elements in $[-1, 1]$, then with probability $1 - \delta$ or more, some t in `ThresholdCandidates` will satisfy the condition that $\mathbb{E}_{\mathbf{x} \sim \{\pm 1\}^d} [|\text{sign}(g(\mathbf{x}) - t) - g(\mathbf{x})|]$ is in $[0, E + \varepsilon/2]$.

Finally, from the Hoeffding bound and union bound we observe that with probability at least $1 - \frac{\delta}{2}$ it is the case that

$$\max_{t \in \text{ThresholdCandidates}} \left| \frac{1}{|T|} \sum_{\mathbf{x} \in T} |\text{sign}(g(\mathbf{x}) - t) - f(\mathbf{x})| - \mathbb{E}_{\mathbf{x} \sim \{-1, 1\}^d} |\text{sign}(g(\mathbf{x}) - t) - f(\mathbf{x})| \right| \leq \frac{\varepsilon}{4}.$$

Overall, we see that with probability at least $1 - \delta$ it is the case that

$$\Pr_{\mathbf{x} \sim \{\pm 1\}^d} [\text{sign}(g(\mathbf{x}) - t^*) \neq f] \leq \frac{1}{|T|} \sum_{\mathbf{x} \in T} |\text{sign}(g(\mathbf{x}) - t^*) - f(\mathbf{x})| + \frac{\varepsilon}{4} \leq \frac{1}{2} \|f - g\|_1 + \varepsilon$$

This finishes the proof. $\square$

4.9 Agnostic Learning Algorithms Handling Randomized Labels

It is customary in the agnostic learning literature to consider a setting that is slightly more general than the one in [Theorem 4](#). Specifically, one is given pairs of i.i.d. elements $\{(x_i, y_i)\}$ from a distribution D_{pairs} , where the distribution of each x_i by itself is uniform. The aim here is to output an efficiently-evaluable succinct representation of a function g for which

$$\Pr_{(\mathbf{x}, \mathbf{y}) \sim D_{\text{pairs}}} [g(\mathbf{x}) \neq \mathbf{y}] \leq \min_{\text{monotone } f_{\text{mon}} : \{-1, 1\}^d \rightarrow \{-1, 1\}} \Pr_{(\mathbf{x}, \mathbf{y}) \sim D_{\text{pairs}}} [f_{\text{mon}}(\mathbf{x}) \neq \mathbf{y}] + O(\varepsilon). \quad (4.7)$$

The only difference between this setting and the one in [Theorem 4](#) is that here the label y does not have to be a function of example x ; it is possible to receive the same example x twice accompanied by different labels. Here we argue that [Theorem 4](#) extends directly into this slightly more general setting. Formally, we show that

Theorem 24. *For all sufficiently large integers d the following holds. There is an algorithm that runs in time $2^{\tilde{O}(\frac{\sqrt{d}}{\varepsilon})}$ and given i.i.d. samples of pairs $\{(x_i, y_i)\}$ from a distribution D_{pairs} , where the marginal distribution over x is uniform, does the following. With probability at least $1 - \frac{1}{2^{0.5d}}$ the algorithm outputs a representation of a monotone function $g : \{\pm 1\}^d \rightarrow \{\pm 1\}$ of size $2^{\tilde{O}(\frac{\sqrt{d}}{\varepsilon})}$ that satisfies [Equation \(4.7\)](#).*

4.9.1 Case 1: ε Is Very Small

We consider two cases. First, suppose ε is so small that the run-time of the algorithm in [Theorem 4](#) exceeds $2^{0.1d}$. In this case, the following algorithm runs in time $\text{poly}(2^d, 1/\varepsilon)$ and outputs an efficiently evaluable succinct representation of a function g for which [Equation \(4.7\)](#) holds:

1. Draw two sets T_1 and T_2 , each of $100d^5 \cdot 2^d/\varepsilon^2$ example-label pairs from D_{pairs} .
2. For each $x \in \{-1, 1\}^d$ let $h(x)$ be $\frac{1}{|\{(x_i, y_i) \in T_1 \text{ s.t. } x_i = x\}|} \sum_{(x_i, y_i) \in T_1 \text{ s.t. } x_i = x} y_i$.
3. Via a size- $2^{O(d)}$ linear program, find the monotone function $q : \{-1, 1\}^d \rightarrow [-1, 1]$ that is closest to h in ℓ_1 distance.
4. Output the function g defined so $g(x) := \text{sign}(q(x) - t^*)$, where t^* is obtained as in [Fact 20](#) using the samples in T_2 .

The function g we output above with high probability satisfies [Theorem 4](#) for the following reason. First of all, via the standard coupon-collector argument with probability at least $1 - \frac{1}{2^{5d}}$ for every $x \in \{-1, 1\}^d$ there will be at least $10^2/\varepsilon^2$ elements in (x_i, y_i) in T for which $x_i = x$. Using the Hoeffding bound and the union bound, we see that with probability at least $1 - \frac{1}{2^{2d}}$ we have

$$\left| h(x) - \mathbb{E}_{(\mathbf{x}', \mathbf{y}') \sim D_{\text{pairs}}} \left[\mathbf{y}' \mid \mathbf{x}' = x \right] \right| \leq \frac{\varepsilon}{2}. \quad (4.8)$$

Now, from steps (3) and (4) we have

$$\frac{\|h - g\|_1}{2} \leq \frac{1}{2} \text{dist}_1(h, \text{mono}) + \varepsilon. \quad (4.9)$$

Therefore, we can combine [Equation \(4.8\)](#) and [Equation \(4.9\)](#) to obtain

$$\Pr_{(\mathbf{x}, \mathbf{y}) \sim D_{\text{pairs}}} [g(\mathbf{x}) \neq \mathbf{y}] \leq \min_{\text{monotone } f_{\text{mon}} : \{-1, 1\}^d \rightarrow \{-1, 1\}} \Pr_{(\mathbf{x}, \mathbf{y}) \sim D_{\text{pairs}}} [f_{\text{mon}}(\mathbf{x}) \neq \mathbf{y}] + O(\varepsilon), \quad (4.10)$$

which finishes the proof for this case.

4.9.2 Case 2: ε Is Not Too Small

Now, we proceed to the other case when ε is not too small and the algorithm in [Theorem 4](#) runs in time at most $2^{0.1d}$ (and therefore uses at most $2^{0.1d}$ samples). In this case, we claim that simply running the algorithm in [Theorem 4](#) will give an efficiently evaluable succinct description of a function g that satisfies the guarantee in [Equation \(4.7\)](#).

We now proceed to show that the guarantee in [Equation \(4.7\)](#) will indeed be achieved. Define a random function $f_{\text{random}} : \{-1, 1\}^d \rightarrow \{-1, 1\}$, so for all $x \in \{-1, 1\}^d$ the value $f_{\text{random}}(x)$ is chosen independently such that $f_{\text{random}}(x) = 1$ with probability $\Pr_{(\mathbf{x}', \mathbf{y}') \sim D_{\text{pairs}}} [\mathbf{y}' = 1 \mid \mathbf{x}' = x]$

and $f_{\text{random}}(x) = -1$ with probability $\Pr_{(\mathbf{x}', \mathbf{y}') \sim D_{\text{pairs}}} [\mathbf{y}' = -1 \mid \mathbf{x}' = x]$. Consider the following two scenarios:

- **Scenario I:** The samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ given to the algorithm from [Theorem 4](#) are indeed i.i.d. samples coming from D_{pairs} .
- **Scenario II:** The samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ given to the algorithm from [Theorem 4](#) are sampled as follows: (i) $\mathbf{x}_i$ are i.i.d. uniform from $\{-1, 1\}^d$ (ii) $\mathbf{y}_i = f_{\text{random}}(\mathbf{x}_i)$.

First we argue that in Scenario II with probability at least $1 - \frac{2}{2^d}$ the function g given by the algorithm from [Theorem 4](#) satisfies [Equation \(4.7\)](#), (here the probability is over the choice of f_{random} , choice of the samples, and the randomness of the algorithm itself). Indeed, let f_{mon}^* be the function that minimizes the right side of [Equation \(4.7\)](#). From the Hoeffding's bound, it follows that with probability at least⁸ $1 - \frac{1}{2^d}$ over the choice of f_{random} it is the case that

$$\left| \Pr_{\mathbf{x} \sim \{-1, 1\}^d} [f_{\text{random}}(\mathbf{x}) \neq f_{\text{mon}}^*(\mathbf{x})] - \Pr_{(\mathbf{x}, \mathbf{y}) \sim D_{\text{pairs}}} [f_{\text{mon}}^*(\mathbf{x}) \neq \mathbf{y}] \right| \leq \varepsilon. \quad (4.11)$$

Now, [Theorem 4](#) implies that with probability at least $1 - \frac{1}{2^d}$

$$\Pr_{\mathbf{x} \sim \{-1, 1\}^d} [g(\mathbf{x}) \neq f_{\text{random}}(\mathbf{x})] \leq \text{dist}_0(f_{\text{random}}, \text{mono}) + O(\varepsilon) \leq \Pr_{\mathbf{x} \sim \{-1, 1\}^d} [f_{\text{mon}}^*(\mathbf{x}) \neq f_{\text{random}}(\mathbf{x})] + O(\varepsilon). \quad (4.12)$$

Combining [Equations 4.11 and 4.12](#) we see that with probability at least $1 - \frac{2}{2^d}$, the function g given by the algorithm from [Theorem 4](#) satisfies [Equation \(4.7\)](#) in Scenario II.

Finally, we argue that [Equation \(4.7\)](#) will be satisfied also in Scenario I with probability at least $1 - \frac{1}{2^{0.5d}}$ for sufficiently large d . Conditioned on the absence of sample pairs $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ with $\mathbf{x}_i = \mathbf{x}_j$, the distributions over samples in Scenario I and Scenario II are the same, Hence it suffices to argue that the collision probability is low, given that the value of ε is such that the algorithm from [Theorem 4](#) uses at most $2^{0.1d}$ samples. By taking a union bound over all pairs of samples, we bound the probability of such collision by $\frac{2^{0.2d}}{2^d} = 2^{-0.8d}$. Thus, information-theoretically, any algorithm can distinguish between Scenario I and Scenario II with an advantage of only at most $2^{-0.8d}$. In particular, this is true of the algorithm that checks whether [Equation \(4.7\)](#) applies. Thus, indeed [Equation \(4.7\)](#) will be satisfied also in Scenario I with probability at least $1 - \frac{2}{2^d} - \frac{1}{2^{0.8d}} \geq 1 - \frac{1}{2^{0.5d}}$, which finishes the proof of [Theorem 24](#).

⁸Here we used that $\varepsilon \geq \frac{1}{\sqrt{d} \text{poly} \log d}$, because otherwise ε would be too small and we would be in the other case when the run-time of the algorithm in [Theorem 4](#) exceeds $2^{0.1d}$. Also, we note that a much stronger bound can be deduced from the Hoeffding bound, but we only need a bound of $1 - \frac{1}{2^d}$.

Chapter 5

Local Correction of Discrete Lipschitz Functions

In this chapter, we present our local correction algorithms for the discrete Lipschitz property, along with lower bounds and applications for tolerant testing and private function evaluation.

5.1 Chapter-Specific Preliminaries

When we discuss the range (or image) of functions, we often refer to its diameter. The *diameter* of a closed and bounded $S \subset \mathbb{R}$ is $\max_{y \in S}(y) - \min_{y \in S}(y)$. Let $G = (V, E)$ be an undirected graph and let $f : V \rightarrow \mathbb{R}$.

Next, we define the violation score of a pair of points, and the violation graph of a function.

Definition 21 (Violated pair, violation score). *For $x, y \in V$, let $\text{dist}_G(x, y)$ denote the shortest path distance from x to y in G . A pair (x, y) of vertices is violated with respect to f if $|f(x) - f(y)| > \text{dist}_G(x, y)$. The violation score of a pair (x, y) with respect to f , denoted $VS_f(x, y)$, is*

$$VS_f(x, y) = |f(x) - f(y)| - \text{dist}_G(x, y)$$

if (x, y) is violated and 0 otherwise. We extend these definitions to partial functions $g : V \rightarrow \mathbb{R} \cup \{?\}$, where $?$ denotes an undefined value, by stipulating that if x or y is in $g^{-1}(?)$ then (x, y) is not violated.

Definition 22 (Violation Graph). *The τ -violation graph with respect to f is a directed graph, denoted $B_{\tau, f}$, with vertex set V and edge set $\{(x, y) : VS_f(x, y) > \tau \text{ and } f(x) < f(y)\}$.*

We specify an LCA for accessing the violation graph. To simplify notation, we assume that any algorithm used as a subroutine gets access to the inputs of the algorithm which calls it; only the inputs that change in recursive calls are explicitly passed as parameters.

Algorithm 22 LCA: $\text{VIOL}(f(\cdot), \tau, x)$

Input: Adjacency lists access to $G = (V, E)$, lookup access to $f : V \rightarrow [0, r]$, range diameter $r \in \mathbb{R}$, threshold $\tau \leq r$, vertex $x \in V$

Output: Neighbor list of x in $B_{\tau, f}$

1: **return** $\{y : \text{dist}_G(x, y) < |f(x) - f(y)| - \tau\}$ $\triangleright$ Compute by performing a BFS from x

5.2 ℓ_1 -Respecting Local Lipschitz Filter

The d -dimensional hypergrid of side length n is the undirected graph, denoted $\mathcal{H}_n^d$, with the vertex set $[n]^d$ and the edge set $\{(x, y) : |x - y| = 1\}$.

Theorem 25. For all $\gamma > 0$ and $\delta \in (0, 1)$, there is an ℓ_1 -respecting local $(1 + \gamma, \delta)$ -Lipschitz filter with blowup 2 over the d -dimensional hypergrid $\mathcal{H}_n^d$. Given lookup access to a function $f : [n]^d \rightarrow [0, r]$, and a random seed ρ of length $d^{O(r)} \cdot \text{polylog}(n \log(r/\gamma)/\delta)$, the filter has lookup and time complexity $(d^r \cdot \text{polylog}(n/\delta))^{O(\log(r/\gamma))}$ for each query $x \in [n]^d$. If f is Lipschitz, then the filter outputs $f(x)$ for all queries x and random seeds.

We first give a global Lipschitz filter (Algorithm 23) and then show how to simulate it locally (in Algorithm 24) by using the result of [64] stated in Theorem 8.

5.2.1 Analysis of the Global Filter

Algorithm 23 GLOBALFILTER₁

Input: Graph $G = (V, E)$, function $f : V \rightarrow [0, r]$, range diameter $r \in \mathbb{R}$, and approximation parameter $\gamma > 0$
Output: $(1 + \gamma)$ -Lipschitz function $g : V \rightarrow [0, r]$

- 1: Let $g_1 \leftarrow f$
- 2: **for** $t \leftarrow 2$ to $\log_{3/2}(\frac{r}{\gamma}) + 1$ **do** ▷ Start at $t = 2$ for GLOBALFILTER₁-LOCALFILTER₁ analogy.
- 3: Set threshold $\tau \leftarrow r \cdot (\frac{2}{3})^{t-1}$ and move-amount $\Delta \leftarrow \frac{r}{3} \cdot (\frac{2}{3})^{t-2}$
- 4: Construct $B_{\tau, g_{t-1}}$ (Definition 22) and compute a maximal matching M_t of $B_{\tau, g_{t-1}}$
- 5: Set $g_t \leftarrow g_{t-1}$
- 6: **for** $(x, y) \in M_t$ **do** ▷ Recall: $f(x) < f(y)$
- 7: Set $g_t(x) \leftarrow g_t(x) + \Delta$
- 8: Set $g_t(y) \leftarrow g_t(y) - \Delta$
- 9: **return** g_t , where $t = \log_{3/2}(\frac{r}{\gamma}) + 1$

The guarantees of GLOBALFILTER (Algorithm 23) are summarized in the following lemma.

Lemma 5.2.1. For all input graphs $G = (V, E)$, functions $f : V \rightarrow [0, r]$, and $\gamma > 0$, if g is the output of GLOBALFILTER₁, then g is a $(1 + \gamma)$ -Lipschitz function and $\|g - f\|_1 \leq 2\ell_1(f, \text{Lip}(G))$.

We prove Lemma 5.2.1 via a sequence of claims. Claim 16 makes an important observation about the violation scores on adjacent edges in the violation graph. Claim 17 argues that the violation scores decrease after each iteration of the loop. Claim 18 converts the guarantee for each iteration to the guarantee on the Lipschitz constant for the output function. Finally, Claim 19 bounds the ℓ_1 -distance between the input and output functions.

Claim 16. If (x, y) and (y, z) are edges in the violation graph $B_{0, f}$ then $VS_f(x, z) \geq VS_f(x, y) + VS_f(y, z)$.

Proof. Since (x, y) and (y, z) are edges in the violation graph $B_{0,f}$, then $f(x) < f(y) < f(z)$. Therefore,

$$\begin{aligned} VS_f(x, z) &= f(z) - f(x) - \text{dist}_G(x, z) \\ &\geq f(z) - f(y) + f(y) - f(x) - \text{dist}_G(x, y) - \text{dist}_G(y, z) \\ &= VS_f(x, y) + VS_f(y, z), \end{aligned}$$

by the definition of the violation score and the triangle inequality. $\square$

Next, we abstract out and analyze the change to the function values made in each iteration of the loop in [Algorithm 23](#).

Claim 17 (Maximum Violation Score Reduction). *Let $G = (V, E)$ be a graph and let $g : V \rightarrow [0, r]$ be a function such that $VS_g(x, y) \leq v$ for all $x, y \in V$. Let M be a maximal matching in $B_{2v/3, g}$ such that $g(x) < g(y)$ for all $(x, y) \in M$. Obtain h as follows: set $h = g$ and then, for every edge $(x, y) \in M$, set $h(x) \leftarrow h(x) + v/3$ and $h(y) \leftarrow h(y) - v/3$. Then $VS_h(x, y) \leq 2v/3$ for all $x, y \in V$.*

Proof. Suppose $x, y \in V$, assume w.l.o.g. that $g(x) \leq g(y)$, and consider the following two cases. Recall that edges in the violation graph (x, y) are directed from the smaller value x to the larger value y .

Case 1: (x, y) is an edge in $B_{2v/3, g}$. By [Claim 16](#), we cannot have vertices a, b, c such that (a, b) and (b, c) are both in $B_{2v/3, g}$, since otherwise $VS_g(a, c)$ would be at least $VS_g(a, b) + VS_g(b, c) > 2v/3 + 2v/3 > v$, contradicting the upper bound of v on violation scores stated in [Claim 17](#). Thus, in $B_{2v/3, g}$, each edge incident on x is outgoing and each edge incident on y is incoming. Consequently, $h(x) \geq g(x)$ and $h(y) \leq g(y)$.

Moreover, since (x, y) is in $B_{2v/3, g}$ and M is a maximal matching, at least one of x, y is matched in M . W.l.o.g. assume that M contains an edge (x, z) . Then $h(x) = g(x) + v/3$. Since $h(y) \leq g(y)$, we have $VS_h(x, y) \leq VS_g(x, y) - v/3 \leq 2v/3$.

Case 2: (x, y) is not an edge in $B_{2v/3, g}$. Then $VS_g(x, y) \leq 2v/3$. Consider how the values of x and y change when we go from g to h . Observe that $|h(x) - g(x)|$ is 0 or $v/3$ for all $x \in V$. If both values for x and for y stay the same, or move in the same direction (both increase or both decrease), then $VS_h(x, y) = VS_g(x, y) \leq 2v/3$. If they move towards each other, then $VS_h(x, y) \leq 2v/3$, whether $h(x) \leq h(y)$ or not.

Now consider the case when they move away from each other, that is, $h(x) \leq g(x)$ and $h(y) \geq g(y)$, and at least one of the inequalities is strict. First, suppose both inequalities are strict. Then there are vertices z_x, z_y such that $(z_x, x), (y, z_y) \in M$. By [Claim 16](#), pair (x, y) is not violated in g (since otherwise $VS_g(z_x, z_y)$ would be at least $VS_g(z_x, x) + VS_g(x, y) + VS_g(y, z_y) > 4v/3$, contradicting the assumption on violation scores in the claim). Since the values of the endpoints move by $v/3$ each, the new violation score $VS_h(x, y) \leq 2v/3$.

Finally, consider the case when only one of the inequalities is strict. W.l.o.g. suppose $h(y) > g(y)$. Then there is $z \in V$ such that $(y, z) \in M$. By [Claim 16](#), the violation score $VS_g(x, y) \leq v/3$, since otherwise $VS_g(x, z)$ would be at least $VS_g(x, y) + VS_g(y, z) > v/3 + 2v/3 = v$, contradicting the assumption on violation scores in the claim. Thus, $VS_h(x, y) \leq VS_g(x, y) + v/3 \leq 2v/3$, as required. $\square$

Next, we use [Claim 17](#) to bound the Lipschitz constant of the function output by the global filter.

Claim 18. *For all $t \geq 1$, the function g_t computed in [Algorithm 23](#) is $(1 + r(\frac{2}{3})^{t-1})$ -Lipschitz. In particular, if $t^* = \log_{3/2}(r/\gamma) + 1$ then g_{t^*} is $(1 + \gamma)$ -Lipschitz.*

Proof. Fix a graph $G = (V, E)$ and a function $f : V \rightarrow [0, r]$. Then $VS_f(x, y) < r$ for all $x, y \in V$.

For all $t \geq 1$, let $v_t = r(\frac{2}{3})^{t-1}$. Notice that, in Line 24 of [Algorithm 23](#), we set $\tau = \frac{2}{3}v_{t-1}$ and $\Delta = \frac{r}{3}(\frac{2}{3})^{t-2} = \frac{1}{3}v_{t-1}$. To prove the claim, it suffices to show that for all $t \geq 1$ and $x, y \in V$,

$$VS_{g_t}(x, y) \leq v_t, \quad (5.1)$$

since then $|g_t(x) - g_t(y)| \leq \text{dist}_G(x, y)(1 + v_t)$. We prove [Equation \(5.1\)](#) by induction on t .

In the base case of $t = 1$, we have $VS_{g_1}(x, y) < r = v_1$ for all $x, y \in V$. Assume [Equation \(5.1\)](#) holds for some $t \geq 1$. Then, instantiating [Claim 17](#) with $g = g_t, h = g_{t+1}$, and $v = v_t$ yields $VS_{g_{t+1}}(x, y) \leq 2v_t/3 = v_{t+1}$ for all $x, y \in V$.

In conclusion, since $v_{t^*} = \gamma$, the function g_{t^*} is $(1 + \gamma)$ -Lipschitz. $\square$

Finally, we argue that the ℓ_1 -distance between the input and the output functions of [Algorithm 23](#) is small.

Claim 19. *Fix a graph $G = (V, E)$ and a function $f : V \rightarrow [0, r]$. Suppose h is the closest (in ℓ_1 -distance) Lipschitz function to f . Then for all $t \geq 1$, the functions g_t computed by [Algorithm 23](#) satisfy $\|g_{t+1} - h\|_1 \leq \|g_t - h\|_1$ and $\|g_t - f\|_1 \leq 2\ell_1(f, \mathcal{Lip}(G))$.*

Proof. Fix $t \geq 1$. Since g_t and g_{t+1} only differ on the endpoints of the edges in the matching M_{t+1} , we restrict our attention to those points. For each edge $(x, y) \in M_{t+1}$, we will show

$$|g_{t+1}(x) - h(x)| + |g_{t+1}(y) - h(y)| \leq |g_t(x) - h(x)| + |g_t(y) - h(y)|. \quad (5.2)$$

Let $\tau = r(\frac{2}{3})^t$ and $\Delta = \frac{r}{3}(\frac{2}{3})^{t-1} = \frac{\tau}{2}$. Suppose $(x, y) \in M_{t+1}$. Recall that this implies that $VS_{g_t}(x, y) > \tau = 2\Delta$ and $g_t(x) < g_t(y)$. By construction, $g_{t+1}(x) = g_t(x) + \Delta$ and $g_{t+1}(y) = g_t(y) - \Delta$. Thus, the violation score of (x, y) decreased by 2Δ , so (x, y) is still violated by g_{t+1} , i.e.,

$$g_{t+1}(x) + \text{dist}_G(x, y) < g_{t+1}(y). \quad (5.3)$$

Define $\Phi(z) = |g_{t+1}(z) - h(z)| - |g_t(z) - h(z)|$ for all $z \in V$. (Intuitively, it captures how much further from $h(z)$ the value on z moved when we changed g_t to g_{t+1} .) Then [Equation \(5.2\)](#) is equivalent to $\Phi(x) + \Phi(y) \leq 0$. If both $\Phi(x) \leq 0$ and $\Phi(y) \leq 0$, then [Equation \(5.2\)](#) holds. Otherwise, $\Phi(x) > 0$ or $\Phi(y) > 0$. Suppose w.l.o.g. $\Phi(x) > 0$. Since $g_{t+1}(x) = g_t(x) + \Delta$, we know that $\Phi(x) \leq \Delta$. To demonstrate that [Equation \(5.2\)](#) holds, it remains to show that $\Phi(y) \leq -\Delta$.

Since $\Phi(x) > 0$, the value $h(x)$ is closer to $g_t(x)$ than to $g_{t+1}(x)$. Since $g_{t+1}(x) = g_t(x) + \Delta$, it implies that $h(x)$ must be below the midpoint between $g_t(x)$ and $g_{t+1}(x)$, which is $g_{t+1}(x) = \Delta/2$. That is,

$$h(x) < g_{t+1}(x) - \Delta/2. \quad (5.4)$$

We use that h is Lipschitz, then apply Equations (5.4) and (5.3) to obtain

$$h(y) < h(x) + \text{dist}_G(x, y) < g_{t+1}(x) - \Delta/2 + \text{dist}_G(x, y) < g_{t+1}(y) - \Delta/2.$$

Since $h(y) < g_{t+1}(y) - \Delta/2$ and $g_{t+1}(y) = g_t(y) - \Delta$, we get that $g_t(y)$ and $g_{t+1}(y)$ are both greater than $h(y)$. Thus, $|g_{t+1}(y) - h(y)| = |g_t(y) - h(y)| - \Delta$ and hence, $\Phi(y) = -\Delta$, so Equation (5.2) holds.

We proved that Equation (5.2) holds for every edge in M_{t+1} . Moreover, for all vertices z outside of M_{t+1} , we have $g_{t+1}(z) = g_t(z)$ and, consequently, $\Phi(z) = 0$. Summing over all vertices, we get that $\sum_{x \in V} \Phi(x) \leq 0$. Thus, $\|g_{t+1} - h\|_1 \leq \|g_t - h\|_1$. By the triangle inequality, $\|g_t - f\|_1 \leq \|g_t - h\|_1 + \|h - f\|_1 \leq \|g_1 - h\|_1 + \|f - h\|_1 = 2\ell_1(f, \mathcal{Lip}(G))$. $\square$

Lemma 5.2.1 follows from Claims 18 and 19.

5.2.2 Analysis of the Local Filter

In this section, we present a local implementation of Algorithm 23 and complete the proof of Theorem 25. We claim that for each $t \in [\log(r/\gamma) + 1]$, Algorithm 24 simulates round t of Algorithm 23 and, for graphs on N vertices with maximum degree D , has lookup complexity $(D^r \cdot \text{polylog}(N/\delta))^{O(\log(r/\gamma))}$.

Algorithm 24 LCA: LOCALFILTER₁($x, t, \rho_1 \circ \dots \circ \rho_t$)

Input: Adjacency lists access to graph $G = (V, E)$, lookup access to $f : V \rightarrow [0, r]$, range diameter $r \in \mathbb{R}$, vertex $x \in V$, iteration number t , approximation parameter $\gamma > 0$, and random seed $\rho = \rho_1 \circ \dots \circ \rho_t$

Subroutines: GHAMATCH (see Theorem 8) and VIOL (see Algorithm 22)

Output: Query access to $(r \cdot (\frac{2}{3})^{t-1})$ -Lipschitz function $g_\rho : V \rightarrow [0, r]$

- 1: **if** $t = 1$ or $r \cdot (\frac{2}{3})^{t-1} < \gamma$ **then**
 - 2: **return** $f(x)$
 - 3: Set threshold $\tau \leftarrow r \cdot (\frac{2}{3})^{t-1}$ and move amount $\Delta \leftarrow \frac{r}{3} \cdot (\frac{2}{3})^{t-2}$
 - 4: Set $f_t(x) \leftarrow \text{LOCALFILTER}_1(x, t-1, \rho_1 \circ \dots \circ \rho_{t-1})$
 - 5: Set $y \leftarrow \text{GHAMATCH}(\text{VIOL}(\text{LOCALFILTER}_1(\cdot, t-1, \rho_1 \circ \dots \circ \rho_{t-1}), \tau, \cdot), x, \rho_t)$
 - 6: **if** $y \neq \perp$ **then**
 - 7: $f_{t-1}(y) \leftarrow \text{LOCALFILTER}_1(y, t-1, \rho_1 \circ \dots \circ \rho_{t-1})$
 - 8: $f_t(x) \leftarrow f_t(x) + \text{sign}(f_{t-1}(y) - f_t(x)) \cdot \Delta$
 - 9: **return** $f_t(x)$
-

Definition 23 (Good seed). Let $G = (V, E)$ be a graph and fix $t \geq 1$. Consider a function $f : V \rightarrow [0, r]$. A string $\boldsymbol{\rho} = \rho_1 \circ \dots \circ \rho_t$ is a good seed for G and f if, for all $i \in [t]$, the matching computed by GHAMATCH in LOCALFILTER₁($\cdot, i, \rho_1 \circ \dots \circ \rho_i$) is maximal.

Claim 20. Fix a graph $G = (V, E)$, a function $f : V \rightarrow [0, r]$, and $\gamma > 0$. Let $t^* = \log_{3/2}(r/\gamma) + 1$ and fix a good seed $\boldsymbol{\rho} = \rho_1 \circ \dots \circ \rho_{t^*}$. Let $g(x)$ denote LOCALFILTER₁($x, t^*, \boldsymbol{\rho}$) for all $x \in V$. Then g is a $(1 + \gamma)$ -Lipschitz function with range $[0, r]$ and $\|f - g\|_1 \leq 2\ell_1(f, \mathcal{Lip}(G))$.

Proof. For all $t \in [t^*]$, let g_t be the function computed by GLOBALFILTER₁ on G, r, f , and γ after iteration t using the matching computed by the call to GHAMATCH in LOCALFILTER₁($x, t, \rho_1 \circ \dots \circ \rho_t$). Recall that the matching computed by each call to GHAMATCH in LOCALFILTER₁($x, t, \rho_1 \circ \dots \circ \rho_t$) is maximal and therefore can be used as the matching in the iteration t of the loop in GLOBALFILTER₁.

By an inductive argument, LOCALFILTER₁($x, t, \rho_1 \circ \dots \circ \rho_t$) = $g_t(x)$ for all $x \in V$ and $t \in [t^*]$. The base case is LOCALFILTER₁($x, 1, \rho_1$) = $f(x) = g_1$, and every subsequent g_t computed by GLOBALFILTER₁ is the same as LOCALFILTER₁($\cdot, t, \rho_1 \circ \dots \circ \rho_t$). Hence, LOCALFILTER₁($x, t^*, \boldsymbol{\rho}$) provides query access to g_{t^*} . By Lemma 5.2.1, g_{t^*} is $(1 + \gamma)$ -Lipschitz and satisfies $\|g_{t^*} - f\| \leq 2\ell_1(f, \mathcal{Lip}(G))$. $\square$

Lemma 5.2.2. Fix $\gamma > 0$ and $\delta \in (0, 1)$. Let $G = (V, E)$ be a graph with $|V| = N$ and maximum degree D . Let $f : V \rightarrow [0, r]$ and $t^* = \log_{3/2}(r/\gamma) + 1$. Then, for a random seed $\boldsymbol{\rho} = \rho_1 \circ \dots \circ \rho_{t^*}$, which is a concatenation of t^* strings of length $D^{O(r)}$ polylog(Nt^*/δ) each, the algorithm LOCALFILTER₁($\cdot, t^*, \boldsymbol{\rho}$) is an ℓ_1 -respecting local $(1 + \gamma, \delta)$ -Lipschitz filter with blowup 2 and lookup and time complexity $(D^r \cdot \text{polylog}(N/\delta))^{O(\log(r/\gamma))}$.

Proof. Since the range of f is at most r , two vertices in a violated pair can be at distance at most $r - 1$. Hence, the maximum degree of the violation graph $B_{\tau, f}$ is at most D^r . By Theorem 8 instantiated with $D_0 = D^r$ and $\delta_0 = \frac{t^*}{\delta}$, the failure probability of each call to GHAMATCH is at most $\frac{\delta}{t^*}$. Since there are at most t^* calls to GHAMATCH, the probability that any call fails is at most δ . It follows that a random string $\boldsymbol{\rho}$ of length specified in the lemma is a good seed (see Definition 23) with probability at least $1 - \delta$. This allows us to apply Claim 20, and conclude that LOCALFILTER₁($x, t^*, \boldsymbol{\rho}$) provides query access to a $(1 + \gamma)$ -Lipschitz function and fails with probability at most δ over the choice of $\boldsymbol{\rho}$.

Let $Q(t)$ be the lookup complexity of LOCALFILTER₁($x, t, \rho_1 \circ \dots \circ \rho_t$). Then $Q(1) = 1$ and, since the max degree of $B_{\tau, f}$ is D^r , each lookup made by GHAMATCH to the violation graph oracle in the $(t - 1)$ -st iteration requires at most $D^r Q(t - 1)$ lookups to compute. Since GHAMATCH makes $D^{O(r)}$ polylog(N/δ) such lookups, $Q(t) \leq D^{O(r)}$ polylog(n/δ) $Q(t - 1)$. Thus, the final lookup complexity is $Q(t^*) \leq (D^r \cdot \text{polylog}(N/\delta))^{O(\log(r/\gamma))}$. By inspection of the pseudocode, we see that the running time is polynomial in the number of lookups. $\square$

Proof of Theorem 25. The theorem follows as a special case of Lemma 5.2.2 with G equal to the hypergrid $\mathcal{H}_n^d$. The hypergrid has n^d vertices and maximum degree $2d$. This gives lookup and time complexity $(d^r \cdot \text{polylog}(n/\delta))^{O(\log(r/\gamma))}$. If f is Lipschitz, then all violation

graphs are empty; therefore, any local matching algorithm returns an empty matching (or can otherwise be amended to do so by checking whether the returned edge is in the graph and returning $\perp$ if it is not). Thus, when f is Lipschitz, the returned value is always $f(x)$. $\square$

5.3 ℓ_0 -Respecting Local Lipschitz Filter

In this section, we present a local Lipschitz filter that respects ℓ_0 -distance rather than ℓ_1 -distance. Unlike the ℓ_1 -respecting filter, the ℓ_0 -respecting filter outputs a function that is 1-Lipschitz.

Theorem 26. *For all $\delta \in (0, 1)$, there exists an ℓ_0 -respecting local $(1, \delta)$ -Lipschitz filter with blowup 2 over the d -dimensional hypergrid $\mathcal{H}_n^d$. Given lookup access to a function $f : [n]^d \rightarrow [0, r]$, and a random seed ρ of length $d^{O(r)} \cdot \text{polylog}(n/\delta)$, the filter has lookup and time complexity $d^{O(r)} \cdot \text{polylog}(n/\delta)$ for each query $x \in [n]^d$. If f is Lipschitz, then the filter outputs $f(x)$ for all queries x and random seeds. If for all $y \in [n]^d$ we have $|f(x) - f(y)| \leq |x - y|$ then the filter outputs $f(x)$.*

We give a global view (Algorithm 25) and prove its correctness before presenting a local implementation (Algorithm 26). We use the convention that $\max_{y \in S}(\cdot)$ is defined to be zero when S is the empty set.

5.3.1 Analysis of the Global Filter

Algorithm 25 GLOBALFILTER₀

Input: Graph $G = (V, E)$, function $f : V \rightarrow [0, r]$
Output: Lipschitz function $g : V \rightarrow [0, r]$

- 1: Construct $B_{0,f}$ (see Definition 22) and compute a vertex cover C of $B_{0,f}$
- 2: Set $g_C \leftarrow f$
- 3: **for** every vertex $u \in C$ **do**
- 4: Set $g_C(u) \leftarrow \max(0, \max_{v \in V \setminus C} (g_C(v) - \text{dist}_G(u, v)))$
- 5: **return** g_C

Algorithm 25 reassigns the labels on a vertex cover C of the violation graph $B_{0,f}$. Observe that the partial function f on the domain $V \setminus C$ is Lipschitz w.r.t. G , because its violation graph has no edges. We claim that this algorithm extends this partial function to a Lipschitz function defined on all of G . It is well known that for a function $f : X \rightarrow \mathbb{R}$ with a metric space domain, if f is Lipschitz on some subset $Y \subset X$, then f can be made Lipschitz while only modifying points in $X \setminus Y$. See, for example, [76] and [13]. We include a proof for completeness.

Claim 21 (Lipschitz extension). *Let $G = (V, E)$ be a graph, and $f : V \rightarrow [0, r]$ a function. Then, for all vertex covers C of $B_{0,f}$, the function g_C returned by Algorithm 25 is Lipschitz.*

Proof. Let $f : V \rightarrow [0, r] \cup \{?\}$ be a partial Lipschitz function and let A_f be the set of points on which f is defined. Fix a vertex $x \notin A_f$ and obtain the function g as follows: Set $g(y) = f(y)$ for all $y \in A_f$. Set $g(x) = \max_{v \in A_f} (f(v) - \text{dist}_G(x, v))$. Note that in the case where A_f is empty, the function f is nowhere defined, and hence setting $g(x) = 0$ will always result in a Lipschitz function. Thus, assume w.l.o.g. that A_f is not empty.

We will first argue that g is Lipschitz. Let $v^* = \arg \max_{v \in A_f} (f(v) - \text{dist}_G(x, v))$, i.e., a vertex such that $g(x) = f(v^*) - \text{dist}_G(x, v^*)$. Then, for all $v \in A_f$,

$$g(x) - g(v) = f(v^*) - \text{dist}_G(x, v^*) - f(v) \leq \text{dist}_G(v, v^*) - \text{dist}_G(x, v^*) \leq \text{dist}_G(x, v).$$

Similarly, $g(v) - g(x) \leq f(v) + \text{dist}_G(x, v) - f(v) = \text{dist}_G(x, v)$, so g is Lipschitz. Notice that if g is a Lipschitz function, then $\max(0, g)$ is also a Lipschitz function (truncating negative values can only decrease the distance between $g(x)$ and $g(y)$ for all pairs x, y in the domain). Thus, setting $g(x) = \max(0, \max_{v \in A_f} (f(v) - \text{dist}_G(x, v)))$ will also yield a Lipschitz function.

Next, we argue that the order of assignment does not affect the extension. Let A_g be set of points on which g is defined and note that $A_g = A_f \cup \{x\}$. We will show that for all $z \notin A_g$ we have

$$\max(0, \max_{v \in A_f} (f(v) - \text{dist}_G(z, v))) = \max(0, \max_{v \in A_g} (g(v) - \text{dist}_G(z, v))).$$

Let $f(z) = \max(0, \max_{v \in A_f} (f(v) - \text{dist}_G(z, v)))$ and $g(z) = \max(0, \max_{v \in A_g} (g(v) - \text{dist}_G(z, v)))$. Then, since $A_g = A_f \cup \{x\}$, we obtain $g(z) = \max(f(z), g(x) - \text{dist}_G(x, z))$. If $g(x) = 0$ then $f(z) = g(z)$ since by definition $f(z) \geq 0$. On the other hand, if $g(x) > 0$ then $g(x) = f(v^*) - \text{dist}_G(x, v^*)$ and hence,

$$\begin{aligned} f(z) &\geq f(v^*) - \text{dist}_G(v^*, z) \geq (f(v^*) - \text{dist}_G(x, v^*)) - \text{dist}_G(x, z) \\ &= g(x) - \text{dist}_G(x, z), \end{aligned}$$

which implies $f(z) = g(z)$. To complete the proof of [Claim 21](#), Let $f : V \rightarrow [0, r]$ be a function with violation graph $B_{0,f}$. Notice that if C is a vertex cover of $B_{0,f}$ then $f : V \setminus C \rightarrow [0, r]$ is Lipschitz, and thus, setting $f(x) = ?$ for all $x \in C$ and applying the extension procedure inductively, we see that g_C is a Lipschitz function. $\square$

The following claim relates the distance to Lipschitzness to the vertex cover of the underlying graph. This relationship is standard for the Lipschitz and related properties, such as monotonicity over general partially ordered sets [\[61, Corollary 2\]](#). See [\[32, Theorem 5\]](#) for the statement and proof for the special case of hypergrid domains. The arguments in these papers extend immediately to the setting of general domains.

Claim 22 (Distance to Lipschitz). *For all graphs $G = (V, E)$ on n vertices and functions $f : V \rightarrow \mathbb{R}$, the size of the minimum vertex cover of the violation graph $B_{0,f}$ is exactly $n \cdot \ell_0(f, \mathcal{Lip}(G))$.*

Proof of Claim 22. Let C be any minimum vertex cover of $B_{0,f}$. We first argue that $n \cdot \ell_0(f, \mathcal{Lip}(G)) \leq |C|$. If g is a partial function that is equal to f on vertices in C and equals f elsewhere, then g is Lipschitz outside of C . Then, by Claim 21 with C as the cover in Algorithm 25, there exist values $y_1, \dots, y_{|C|}$ such that setting $g(x_i) = y_i$ for each $x_i \in C$ yields a Lipschitz function. It follows that $n \cdot \ell_0(f, \mathcal{Lip}(G)) \leq |C|$.

Next, using the function g defined in the previous paragraph, suppose the set $P = \{x : g(x) \neq f(x)\}$ is not a vertex cover for $B_{0,f}$. Then, there exists some edge (x, y) in $B_{0,f}$ such that $f(x) = g(x)$ and $f(y) = g(y)$ (i.e. $x, y \notin P$). But by definition of $B_{0,f}$, this implies that $|g(x) - g(y)| > \text{dist}_G(x, y)$ which contradicts the fact that g is Lipschitz. It follows that $n \cdot \ell_0(f, \mathcal{Lip}(G)) = |C|$. $\square$

5.3.2 Analysis of the Local Filter

Using Algorithm 22 and GHAMATCH from Theorem 8, we construct Algorithm 26, an LCA which provides query access to a Lipschitz function close to the input function. It is analyzed in Lemma 5.3.1.

Algorithm 26 LCA: LOCALFILTER₀(x, ρ)

Input: Adjacency lists access to graph $G = (V, E)$, lookup access to $f : V \rightarrow [0, r]$, range diameter $r \in \mathbb{R}$, vertex $x \in V$, random seed ρ

Subroutines: GHAMATCH (see Theorem 8) and VIOL (see Algorithm 22).

Output: Query access to Lipschitz function $g : V \rightarrow [0, r]$.

- 1: **if** GHAMATCH(VIOL($f, 0, \cdot$), x, ρ) = $\perp$ **then**
 - 2: **return** $f(x)$
 - 3: **else**
 - 4: $S \leftarrow \{y : \text{dist}_G(x, y) \leq r \text{ and } \text{GHAMATCH}(\text{VIOL}(f, 0, \cdot), y, \rho) = \perp\}$
 - 5: **return** $\max(0, \max_{y \in S}(f(y) - \text{dist}_G(x, y)))$
-

Lemma 5.3.1 (LOCALFILTER₀). *Fix $\delta \in (0, 1)$. Let $G = (V, E)$ be a graph with N vertices and maximum degree D . Then, for a random seed ρ of length $D^{O(r)} \cdot \text{polylog}(N/\delta)$, the algorithm LOCALFILTER₀(x, ρ) (Algorithm 26) is an ℓ_0 -respecting local $(1, \delta)$ -Lipschitz filter with blowup 2 and lookup and time complexity $D^{O(r)} \cdot \text{polylog}(N/\delta)$.*

Proof. Since the range of f is bounded by r , a pair of violated vertices x, y must have $\text{dist}_G(x, y) < r$, and thus, the maximum degree of $B_{0,f}$ is at most D^r . By Theorem 8 instantiated with $D_0 = D^r$ and $\delta_0 = \delta$, the algorithm GHAMATCH has lookup and time complexity $D^{O(r)} \cdot \text{polylog}(N/\delta)$ per query, and fails to provide query access to a maximal matching with probability at most δ over the choice of ρ . Since LOCALFILTER₀ makes at most D^r queries to GHAMATCH and only fails when GHAMATCH fails, LOCALFILTER₀ has lookup and time complexity $D^{O(r)} \cdot \text{polylog}(N/\delta)$ and failure probability at most δ . Let ρ be a seed for which GHAMATCH does not fail, and let C be the set of vertices that are matched by GHAMATCH when given adjacency lists access to $B_{0,f}$. Since the matching is maximal, C

is a 2-approximate vertex cover of $B_{0,f}$. Hence, we can run GLOBALFILTER_0 (Algorithm 25) and use C as the vertex cover. Since LOCALFILTER_0 and GLOBALFILTER_0 apply the same procedure to every vertex in V , and since C has at most twice as many vertices as a minimum vertex cover, Claims 21 and 22 imply that LOCALFILTER_0 provides query access to some Lipschitz function g satisfying $\|f - g\|_0 = |C| \leq 2\ell_0(f, \mathcal{Lip}(G))$. $\square$

Proof of Theorem 26. This is an application of Lemma 5.3.1 to the d -dimensional hypergrid $\mathcal{H}_n^d$. The hypergrid has n^d vertices and a maximum degree of $2d$, therefore, the lookup and time complexity are $d^{O(r)} \cdot \text{polylog}(n^d/\delta) = d^{O(r)} \cdot \text{polylog}(n/\delta)$. Similarly, the length of the random seed is also $d^{O(r)} \cdot \text{polylog}(n/\delta)$. If f is Lipschitz, then all violation graphs are empty; therefore, any local matching algorithm returns an empty matching (or can otherwise be amended to do so by checking whether the returned edge is in the graph and returning $\perp$ if it is not). Thus, when f is Lipschitz, the returned value is always $f(x)$.

If for all $y \in [n]^d$ we have $|f(x) - f(y)| \leq |x - y|$ then no edges in the violation graph are incident on x . Therefore, every local matching algorithm returns a matching that does not contain x , or can otherwise be amended to do so by checking whether the returned edge is in the graph and returning $\perp$ if it is not. Hence, the returned value is always $f(x)$. $\square$

5.4 Lower Bounds

In this section, we prove our lower bound for local filters, stated in Section 1.5. We start with a more detailed statement of the lower bound.

Theorem 27 (Local Lipschitz filter lower bound). *For all local $(1, \frac{1}{4})$ -Lipschitz filters $\mathcal{A}$ over the hypercube $\mathcal{H}^d$, for all even $r \geq 4$ and integer $d \geq 2^{16}r$, there exists a function $f : \{0, 1\}^d \rightarrow [0, r]$ for which the lookup complexity of $\mathcal{A}$ is $(\frac{d}{r})^{\Omega(r)}$.*

Note that the same bound on lookup complexity applies to local $(1 + \frac{1}{2r}, \frac{1}{4})$ -Lipschitz filters over $\mathcal{H}^d$. This can be seen by observing the Lipschitz constant in the hard distributions constructed in the proof of the theorem (in Definition 25). Our proof is via a reduction from distribution-free testing. In Section 5.4.1, we state our lower bound on distribution-free testing of Lipschitz functions and use it to derive the lower bound on local Lipschitz filters stated in Theorem 27. In Section 5.4.2, we prove our lower bound for distribution-free testing.

5.4.1 Testing Definitions and the Lower Bound for Local Lipschitz Filters

We start by defining distribution-free testing of the Lipschitz property.

Definition 24 (Distribution-free Lipschitz testing). *Fix $\varepsilon \in (0, 1/2]$ and $r \in \mathbb{R}$. A distribution-free Lipschitz ε -tester $\mathcal{T}$ is an algorithm that gets query access to the input function $f : \{0, 1\}^d \rightarrow [0, r]$ and sample access to the input distribution D over $\{0, 1\}^d$. If f is Lipschitz, then $\mathcal{T}(f, D)$ accepts with probability at least $2/3$, and if $\ell_{0,D}(f, \mathcal{Lip}(\mathcal{H}^d)) \geq \varepsilon$, then it rejects with probability at least $2/3$.*

We give a sample and query lower bound for this task.

Theorem 28 (Distribution-free testing lower bound). *Let $\mathcal{T}$ be a distribution-free Lipschitz $\frac{1}{2}$ -tester. Then, for all sufficiently large $d \in \mathbb{N}$ and even integers $4 \leq r \leq 2^{-16}d$, there exists a function $f : \{0, 1\}^d \rightarrow [0, r]$ and a distribution D , such that $\mathcal{T}(f, D)$ either has sample complexity $2^{\Omega(d)}$, or query complexity $(\frac{d}{r})^{\Omega(r)}$.*

Before proving [Theorem 28](#), we use it to prove the lower bound on the lookup complexity of local Lipschitz filters, stated in [Theorem 27](#). Recall that it says that every local $(1, \frac{1}{4})$ -Lipschitz filter over the hypercube $\mathcal{H}^d$ w.r.t. ℓ_0 -distance has worst-case lookup complexity $(\frac{d}{r})^{\Omega(r)}$.

Proof of [Theorem 27](#). Let $\mathcal{A}$ be a local $(1, \frac{1}{4})$ -Lipschitz filter for functions $f : \{0, 1\}^d \rightarrow [0, r]$ over the hypercube $\mathcal{H}^d$. Then, given an instance (f, D) of the distribution-free Lipschitz testing problem with proximity parameter ε , we can run the following algorithm, denoted $\mathcal{T}(f, D)$:

1. Sample a set S of $3/\varepsilon$ points from D .
2. If $\mathcal{A}(x, \rho) \neq f(x)$ for some $x \in S$ then **reject**; otherwise, **accept**.

If f is Lipschitz then, with probability at least $3/4 > 2/3$, we have $\mathcal{A}(x, \rho) = f(x)$ for all $x \in S$ and, consequently, $\mathcal{T}(f, D)$ accepts. Now suppose f is ε -far from Lipschitz with respect to D . Then $\mathcal{A}$ fails with probability at most $1/4$. With the remaining probability, it provides query access to some Lipschitz function g_ρ . This function disagrees with f on a point sampled from D with probability at least ε . In this case, $\mathcal{T}$ incorrectly accepts with probability at most $(1 - \varepsilon)^{3/\varepsilon} \leq e^{-3}$. By a union bound, $\mathcal{T}$ fails or accepts with probability at most $\frac{1}{4} + e^{-3} \leq \frac{1}{3}$. Therefore, $\mathcal{T}$ satisfies [Definition 24](#). By [Theorem 28](#), $\mathcal{T}$ needs at least $(\frac{d}{r})^{\Omega(r)}$ queries, so $\mathcal{A}$ must make at least $(\frac{d}{r})^{\Omega(r)}$ lookups. $\square$

5.4.2 Distribution-Free Testing Lower Bound

We prove [Theorem 28](#) by constructing two distributions, D_0 and D_1 , on pairs (f, D) and then applying Yao's Minimax Principle [112]. We show that D_0 has most of its probability mass on positive instances and D_1 has most of its mass on negative instances of distribution-free Lipschitz testing. The crux of the proof of [Theorem 28](#) is demonstrating that every deterministic (potentially adaptive) tester with insufficient sample and query complexity distinguishes D_0 and D_1 only with small probability.

We start by defining our hard distributions. In both distributions, D is uniform over a large set of points, called *anchor points*, partitioned into sets A and A' , both of size $2^{d/64}$. We treat A (and A'), both as a set and as an ordered sequence indexed by $i \in [2^{d/64}]$. Points in A and A' with the same index are paired up; specifically, the pairs are $(A[i], A'[i])$ for all $i \in [2^{d/64}]$. For every point in $x \in \{0, 1\}^d$ and radius $t > 0$, let $\text{BALL}_t(x)$ denote the **open** ball centered at x , that is, the set $\{y \in \{0, 1\}^d : |x - y| < t\}$. For each point $x \in A$, the function value of every point $y \in \text{BALL}_{\frac{r}{2}}(x)$ is equal to the distance from x to y . For each

point $x \in A'$, the function value of every point $y \in \text{BALL}_{\frac{r}{2}}(x)$ is equal to r minus the distance from x to y where r is the desired image diameter of the functions. The points in A' are chosen so that every pair $(A[i], A'[i])$ satisfies the Lipschitz condition with equality in D_0 and violates the Lipschitz condition in D_1 .

Definition 25 (Hard Distributions). *Fix sufficiently large $d \in \mathbb{N}$ and $4 \leq r \leq 2^{-16}d$. For all $b \in \{0, 1\}$, let D_b be the distribution given by the following sampling procedure:*

1. *Sample a list A of $2^{d/64}$ elements in $\{0, 1\}^d$ independently and uniformly at random.*
2. *Sample a list A' of the same length as A as follows. For each $i \in [2^{d/64}]$, pick the element $A'[i]$ uniformly and independently from $\{y \in \{0, 1\}^d : |A[i] - y| = r - b\}$. The elements of $A \cup A'$ are called anchor points. Additionally, for each $i \in [2^{d/64}]$, we call $A[i]$ and $A'[i]$ corresponding anchor points.*
3. *Define $f : \{0, 1\}^d \rightarrow [0, r]$ by*

$$f(x) = \begin{cases} |x - A[i]| & \text{if } x \in \text{BALL}_{\frac{r}{2}}(A[i]) \text{ for some } i \in [2^{d/64}]; \\ r - |x - A'[i]| & \text{else if } x \in \text{BALL}_{\frac{r}{2}}(A'[i]) \text{ for some } i \in [2^{d/64}]; \\ r/2 & \text{otherwise.} \end{cases}$$

4. *Output (f, U) , where U is the uniform distribution over $A \cup A'$.*

Next, we define a bad event B that occurs with small probability and analyze the distance to Lipschitzness of functions arising in the support of distributions D_b , conditioned on $\overline{B}$. For a distribution D and an event E , let $D|_E$ denote the conditional distribution of a sample from D given E .

Lemma 5.4.1 (Distance to Lipschitzness). *Let B_0 be the event that $|A[i] - A[j]| \leq d/4$ for some distinct $i, j \in [2^{d/64}]$. Then*

1. $\Pr_{D_b}[B_0] \leq 2^{-d/32}$ for all $b \in \{0, 1\}$.
2. *If $(f, U) \sim D_0|_{\overline{B_0}}$ then f is Lipschitz, and if $(f, U) \sim D_1|_{\overline{B_0}}$ then $\ell_{0,U}(f, \mathcal{Lip}(\mathcal{H}^d)) \geq \frac{1}{2}$.*

Proof. To prove Item 1, choose $x, y \in \{0, 1\}^d$ by setting each coordinate to one independently with probability $p = \frac{1}{2}$. Let $\mu = \mathbb{E}_{x,y}[d(x, y)] = 2dp(1-p) = \frac{d}{2}$. By Chernoff bound, $\Pr_{x,y}[d(x, y) \leq \frac{\mu}{2}] \leq e^{-\frac{\mu}{8}} \leq 2^{-d/16}$. There are at most $2^{d/32}$ pairs of points in A . By a union bound over all such pairs, $\Pr[B_0] \leq 2^{-d/16} \cdot 2^{d/32} = 2^{-d/32}$. To prove Item 2, recall that $r \leq 2^{-16}d$. Suppose that B_0 did not occur. If $b = 0$ then f is Lipschitz because balls $\text{BALL}_{\frac{r}{2}}(x)$ are disjoint for all anchor points x . When $b = 1$, every pair $(A[i], A'[i])$ violates the Lipschitz condition, so f is $1/2$ -far from Lipschitz w.r.t. U . $\square$

Indistinguishability of the Hard Distributions by a Deterministic Algorithm

Fix a deterministic distribution-free Lipschitz $\frac{1}{2}$ -tester $\mathcal{T}$ that gets access to input (f, U) , takes $s = \frac{1}{8} \cdot 2^{d/128}$ samples from U and makes q queries to f . Since the samples from U are independent (and, in particular, do not depend on query answers), we assume w.l.o.g. that $\mathcal{T}$ receives all samples from U prior to making its queries. One of the challenges in proving that the distributions D_0 and D_1 are hard to distinguish for $\mathcal{T}$ is dealing with adaptivity. We overcome this challenge by showing that $\mathcal{T}$ can be simulated by a *nonadaptive algorithm* $\mathcal{T}_{na}$ that is provided with extra information. In addition to its samples: $\mathcal{T}_{na}$ gets at least one point from each pair $(A[i], A'[i])$, as well as function values on these points. Next, we define the extended sample given to $\mathcal{T}_{na}$ and the associated event B_1 that indicates that the sample is bad. We analyze the probability of B_1 immediately after the definition.

Definition 26 (Sample set, extended sample, bad sample event B_1). *Fix $b \in \{0, 1\}$ and sample $(f, U) \sim D_b$. Let S denote the sample set of $\frac{1}{8} \cdot 2^{d/128}$ points obtained i.i.d. from U by the tester $\mathcal{T}$. The extended sample S^+ is the set $S \cup \{A[i] : i \in [2^{d/64}] \wedge A[i] \notin S \wedge A'[i] \notin S\}$. Let S^- denote the set $(A \cup A') \setminus S^+$. A set S^+ is good if all distinct $x, y \in S^+$ satisfy $|x - y| > d/5$ and bad otherwise. Define B_1 as the event that S^+ is bad.*

Lemma 5.4.2 (B_1 bound). *Fix $b \in \{0, 1\}$. Then $\Pr_{D_b, S}[B_1] \leq \frac{1}{30}$.*

Proof. Recall the bad event B_0 from Lemma 5.4.1. By the law of total probability,

$$\Pr_{D_b, S}[B_1] = \Pr_{D_b, S}[B_1|B_0] \cdot \Pr_{D_b}[B_0] + \Pr_{D_b, S}[B_1|\overline{B_0}] \cdot \Pr_{D_b}[\overline{B_0}] \leq \Pr_{D_b}[B_0] + \Pr_{D_b, S}[B_1|\overline{B_0}]. \quad (5.5)$$

To bound $\Pr_{D_b, S}[B_1|\overline{B_0}]$, observe that if B_0 did not occur, then all pairs (x, y) of anchor points, except for the corresponding pairs, satisfy $|x - y| > \frac{d}{4} - 2r > \frac{d}{5}$ because $r < d \cdot 2^{-16}$. In particular, it means that all anchor points are distinct. Now, condition on $\overline{B_0}$. Then event B_1 can occur only if both $A[i]$ and $A'[i]$ for some $i \in [2^{d/64}]$ appear in the extended sample S^+ . By Definition 26, this is equivalent to the event that both $A[i]$ and $A'[i]$ for some $i \in [2^{d/64}]$ appear in S . Each pair of samples in S is a pair of corresponding anchor points with probability at most $2^{-d/64}$. By a union bound over the at most $\frac{1}{64} \cdot 2^{d/64}$ pairs of samples taken for S , the probability that $A[i], A'[i] \in S$ for some i is at most $\frac{1}{64} \cdot 2^{d/64} \cdot 2^{-d/64} = \frac{1}{64}$. Hence, $\Pr_{D_b, S}[B_1|\overline{B_0}] \leq \frac{1}{64}$. The lemma follows from Equation (5.5) and Lemma 5.4.1. $\square$

The Simulator

One of the key ideas in the analysis is that our hard distributions, and the sampling done by the tester, can be simulated by first obtaining the set S^+ using steps which are identical for $b = 0$ and $b = 1$, and only then selecting points in S^- to obtain the full description of the function f and the distribution U . Next, we state the simulation procedure. Note that the first 3 steps of the procedure do not use bit b , that is, are the same for simulating D_0 and D_1 .

Definition 27 (Simulator). *Fix $b \in \{0, 1\}$. Let $\hat{D}_b$ be the distribution given by the following procedure:*

1. Sample a list S^+ of $2^{d/64}$ elements in $\{0, 1\}^d$ independently and uniformly at random.
2. For each $i \in [2^{d/64}]$, do the following: if $i \leq \frac{1}{8} \cdot 2^{d/128}$, then assign $S^+[i]$ to either $A[i]$ or $A'[i]$ uniformly and independently at random; if $i > \frac{1}{8} \cdot 2^{d/128}$ then assign $S^+[i]$ to $A[i]$.
3. Proceed as in Step 3 of the procedure in Definition 25 to set $f(x)$ for all $x \in S^+$.
4. For each $i \in [2^{d/64}]$, pick the element $S^-[i]$ uniformly and independently from $\{y \in \{0, 1\}^d : |S^+[i] - y| = r - b\}$. Assign it to $A'[i]$ if $S^+[i]$ was assigned to $A[i]$ and vice versa.
5. Proceed as in Step 3 of the procedure in Definition 25 to set $f(x)$ for all $x \notin S^+$ and output (f, U, S^+) , where U is the uniform distribution over $A \cup A'$.

Observation 29 states that, conditioned on $\overline{B_1}$, the simulator produces identical distributions on the extended sample S^+ and function values $f(x)$ on points $x \in S^+$, regardless of whether it is run with $b = 0$ or $b = 1$. Moreover, conditioned on $\overline{B_1}$, it faithfully simulates sampling (f, U) from D_b and S from U , and then extending S to S^+ according to the procedure described in Definition 26.

Observation 29 (Simulator facts). Let $(f_b, U_b, S_b^+) \sim \hat{D}_b|_{\overline{B_1}}$ for each $b \in \{0, 1\}$. Let $f(S^+)$ denote function f restricted to the set S^+ . Then the distribution of $(S_0^+, f(S_0^+))$ is identical to the distribution of $(S_1^+, f(S_1^+))$.

Now fix $b \in \{0, 1\}$. Sample $(f, U) \sim D_b|_{\overline{B_1}}$ and $S \sim U$. Then the distribution of (f, U, S^+) is identical to the distribution of (f_b, U_b, S_b^+) .

Proof. Since the first 3 steps in Definition 27 do not depend on b , the distribution of $(S_0^+, f(S_0^+))$ is the same as the distribution of $(S_1^+, f(S_1^+))$. Now, fix $b \in \{0, 1\}$. Notice that in the procedure for sampling from D_b (Definition 25), for all $i \in [2^{d/64}]$ the anchor point $A[i]$ is a uniformly random point in $\{0, 1\}^d$, and the anchor point $A'[i]$ is sampled uniformly from $\{y : |y - A[i]| = r - b\}$. By the symmetry of the hypercube, the marginal distribution of $A'[i]$ is uniform over $\{0, 1\}^d$. Hence, sampling $A'[i]$ uniformly at random from $\{0, 1\}^d$ and then $A[i]$ uniformly at random from $\{y : |y - A'[i]| = r - b\}$ yields the same distribution. Thus, the distribution of anchor points is the same under $\hat{D}_b$ as under D_b for each $b \in \{0, 1\}$. Now, conditioned on $\overline{B_1}$, the set S^+ contains exactly one anchor point from each corresponding pair. By the preceding remarks, we can assume the anchor point in S^+ was sampled uniformly at random from $\{0, 1\}^d$, and the corresponding anchor point was sampled uniformly from the set of points at distance $r - b$. Thus, conditioned on $\overline{B_0}$, the distribution over S^+ and S^- is the same as the distribution over S_b^+ and S_b^- . Since (U, f) and (U_b, f_b) are (respectively), uniquely determined by (S^+, S^-) and (S_b^+, S_b^-) , the distribution of (U, f, S^+) is the same as the distribution of (U_b, f_b, S_b^+) . $\square$

The Bad Query Event and the Proof that Adaptivity Does Not Help

Observation 29 assures us that distributions of $\{(x, f(x)) : x \in S^+\}$ are the same (conditioned on $\overline{B_1}$) for both hard distributions. The function values $f(y)$ for $y \in \bigcup_{x \in S^+} \text{BALL}_{\frac{r}{2}}(x)$ are

determined by $\{(x, f(x)) : x \in S^+\}$. Moreover, the function values $f(y)$ are set to $r/2$ for all $y \notin \bigcup_{x \in (S^+ \cup S^-)} \text{BALL}_{\frac{r}{2}}(x)$. So, intuitively, the tester can distinguish the distributions only if it queries a point in $\text{BALL}_{\frac{r}{2}}(x)$ for some $x \in S^-$. Our last bad event, introduced next, captures this possibility.

Definition 28 (Revealing point, bad query event $B_{\mathcal{T}}$). *Fix $b \in \{0, 1\}$ and sample $(f, U) \sim D_b$. A point x is revealing if $x \in \text{BALL}_{\frac{r}{2}}(y)$ for some $y \in S^-$. Let $B_{\mathcal{T}}$ be the event that $\mathcal{T}$ queries a revealing point.*

To bound the probability of $B_{\mathcal{T}}$, we first introduce the nonadaptive tester $\mathcal{T}_{na}$ that simulates $\mathcal{T}$ to decide on all of its queries. Tester $\mathcal{T}_{na}$ gets query access to a function f sampled from D_b , a sample $S \sim U$ and $\{(x, f(x)) : x \in S^+\}$. Subsequently, in [Claim 23](#), we argue that if $\mathcal{T}$ queries a revealing point then $\mathcal{T}_{na}$ queries such a point as well. This implies that $\Pr_{D_b, S}[B_{\mathcal{T}}] \leq \Pr_{D_b, S}[B_{\mathcal{T}_{na}}]$, where $B_{\mathcal{T}_{na}}$ is defined analogously to $B_{\mathcal{T}}$, but for the tester $\mathcal{T}_{na}$. Finally, in [Lemma 5.4.3](#), we upper bound the probability of $B_{\mathcal{T}_{na}}$ by first arguing that, conditioned on $\overline{B_1}$, the probability that $\mathcal{T}_{na}$ queries a revealing point is small. Combining this fact with the bound on $\overline{B_1}$ yields an upper bound on $B_{\mathcal{T}_{na}}$ and, consequently, $B_{\mathcal{T}}$.

Definition 29 ($\mathcal{T}_{na}$). *Let $\mathcal{T}_{na}$ be a nonadaptive deterministic algorithm that gets query access to f sampled from D_b , sets $S \sim U$ and $\{(x, f(x)) : x \in S^+\}$, and selects its queries by simulating $\mathcal{T}$ as follows:*

1. *Provide S as the sample and answer each query $x \in \{0, 1\}^d$ with $g(x)$ defined by*

$$g(x) = \begin{cases} |x - y| & \text{if } x \in \text{BALL}_{\frac{r}{2}}(y) \text{ for some } y \in S^+ \text{ satisfying } f(y) = 0; \\ r - |x - y| & \text{else if } x \in \text{BALL}_{\frac{r}{2}}(y) \text{ for some } y \in S^+ \text{ satisfying } f(y) = r; \\ r/2 & \text{otherwise.} \end{cases}$$

2. *Let $x_1, \dots, x_q$ be the queries made by $\mathcal{T}$ in the simulation. Query f on $x_1, \dots, x_q$.*

Claim 23 (Adaptivity does not help). *If $\mathcal{T}$ queries a revealing point then $\mathcal{T}_{na}$ queries a revealing point.*

Proof. Let $x_1, \dots, x_q$ be the queries made by $\mathcal{T}$ and $y_1, \dots, y_q$ be the queries made by $\mathcal{T}_{na}$. Since $\mathcal{T}$ is deterministic and the sample set S is the same in both $\mathcal{T}_{na}$ and in $\mathcal{T}$, we have $x_1 = y_1$. Assume $\mathcal{T}$ queries a revealing point. Let $m \in [q]$ be the smallest index such that x_m is revealing. By definition of g and revealing point, $g(x_i) = f(x_i)$ for all $i \in [m - 1]$. Consequently, $y_m = x_m$ and $\mathcal{T}_{na}$ queries a revealing point. $\square$

Lemma 5.4.3 ($B_{\mathcal{T}}$ bound). *Fix $b \in \{0, 1\}$. There exists a constant $\alpha > 0$ such that, for all sufficiently large d , if $\mathcal{T}$ makes $q = 2^{\alpha r \log(d/r)}$ queries then $\Pr_{D_b, S}[B_{\mathcal{T}}] < \frac{2}{30}$.*

Proof. By [Claim 23](#), $\Pr_{D_b, S}[B_{\mathcal{T}}] \leq \Pr_{D_b, S}[B_{\mathcal{T}_{na}}]$. Applying the law of total probability we obtain the inequality $\Pr_{D_b, S}[B_{\mathcal{T}_{na}}] \leq \Pr_{D_b, S}[B_{\mathcal{T}_{na}} | \overline{B_1}] + \Pr_{D_b, S}[B_1]$. Next, we compute $\Pr_{D_b, S}[B_{\mathcal{T}_{na}} | \overline{B_1}]$, which is equal to $\Pr_{\hat{D}_b}[B_{\mathcal{T}_{na}} | \overline{B_1}]$, since by [Observation 29](#), the simulator

faithfully simulates sampling $(f, U) \sim D_b$ and then obtaining $S \sim U$, conditioned on $\overline{B_1}$. By the principle of deferred decisions, we can stop the simulator after Step 3, then consider queries from $\mathcal{T}_{na}$, and only then run the rest of the simulator. Since $\mathcal{T}_{na}$ is a nonadaptive q -query algorithm, it is determined by the collection $(x_1, \dots, x_q)$ of query points that it chooses as a function of its input (sets S and $\{(x, f(x)) : x \in S^+\}$). We will argue that the probability (over the randomness of the simulator) that the set $\{x_1, \dots, x_q\}$ contains a revealing point (i.e., a point on which f and g disagree) is small. Consider some query x made by $\mathcal{T}_{na}$. By Definition 28, a point x is revealing if $x \in \text{BALL}_{\frac{r}{2}}(S^-[i])$ for some $i \in [2^{d/64}]$. Recall that each $S^-[i]$ satisfies $|S^+[i] - S^-[i]| = r - b$, and thus each revealing point is in $\text{BALL}_{3r/2}(S^+[i])$ for some $i \in [2^{d/64}]$. All such balls around anchor points in S^+ are disjoint, because $r \leq 2^{-16} \cdot d$ and we are conditioning on $\overline{B_1}$ (the event that all pairs of points in S^+ are at distance greater than $d/5$).

Suppose $x \in \text{BALL}_{3r/2}(S^+[i])$ for some $i \in [2^{d/64}]$. (If not, x cannot be a revealing point.) For x to be revealing, it must be in $\text{BALL}_{\frac{r}{2}}(S^-[i])$ or equivalently, $S^-[i]$ must be in $\text{BALL}_{\frac{r}{2}}(x)$. The simulator chooses $S^-[i]$ uniformly and independently from $\{y \in \{0, 1\}^d : |S^+[i] - y| = r - b\}$. The number of points at distance $r - b \geq r - 1$ from $S^+[i]$ is at least $\binom{d}{r-b} > \left(\frac{d}{r}\right)^{r-1} \geq \left(\frac{d}{r}\right)^{3r/4}$, where the last inequality holds because $r \geq 4$. Out of these choices, only those that are in $\text{BALL}_{\frac{r}{2}}(x)$ will make x a revealing point. The number of points in $\text{BALL}_{\frac{r}{2}}(x)$ is $\sum_{i=0}^{r/2} \binom{d}{i} \leq r \binom{d}{r/2} \leq r \left(\frac{2de}{r}\right)^{r/2}$. Then

$$\begin{aligned} \Pr_{\hat{D}_b}[x \text{ is revealing} \mid \overline{B_1}] &\leq r \left(\frac{2de}{r}\right)^{r/2} \left(\frac{r}{d}\right)^{3r/4} \leq r(2e)^{r/2} \left(\frac{r}{d}\right)^{r/4} = 2^{\log(r) + \frac{r}{2} \log(2e) - \frac{r}{4} \log(d/r)} \\ &\leq 2^{2r - \frac{r}{4} \log(d/r)} \leq 2^{-\frac{r}{8} \log(d/r)} < \frac{1}{30} \cdot 2^{-\alpha r \log(d/r)}, \end{aligned}$$

where the first inequality in the second line holds because $r \geq 4$, the next inequality holds since $r \leq 2^{-16}d$ and, for the last inequality, we set $\alpha = \frac{1}{32}$ and use both bounds on r . By a union bound over the $q = 2^{-\alpha r \log(d/r)}$ queries, $\Pr_{D_b, S}[B_{\mathcal{T}_{na}} | \overline{B_1}] = \Pr_{\hat{D}_b}[B_{\mathcal{T}_{na}} | \overline{B_1}] < \frac{1}{30}$. Using the bound from Lemma 5.4.2 on the probability of $\overline{B_1}$, we obtain

$$\Pr_{D_b, S}[B_{\mathcal{T}_{na}}] \leq \Pr_{D_b, S}[B_{\mathcal{T}_{na}} | \overline{B_1}] + \Pr_{D_b, S}[\overline{B_1}] \leq \frac{2}{30},$$

completing the proof of Lemma 5.4.2. $\square$

Proof of Distribution-Free Testing Lower Bound

Before proving Theorem 28, we argue that conditioned on $\overline{B_1 \cup B_{\mathcal{T}}}$, the distribution of samples and query answers seen by $\mathcal{T}$ is the same whether $(f, U) \sim D_0$ or $(f, U) \sim D_1$.

Definition 30 (*D-view*). For all distributions D over instances of distribution-free testing, and all t -sample, q -query deterministic algorithms, let *D-view* be the distribution over samples $s_1, \dots, s_t$ and query answers $a_1, \dots, a_q$ seen by the algorithm on input (f, U) when $(f, U) \sim D$.

Lemma 5.4.4 (Equal conditional distributions). $D_0\text{-view}|_{\overline{B_1 \cup B_{\mathcal{T}}}} = D_1\text{-view}|_{\overline{B_1 \cup B_{\mathcal{T}}}}$.

Proof. Conditioned on $\overline{B_1}$ and $\overline{B_{\mathcal{T}}}$, every query answer $f(x)$ given to $\mathcal{T}$ is determined by the function g in [Definition 29](#). In particular, every query answer is a deterministic function of the points in S^+ , the restricted function $f(S^+)$, and (possibly) previous query answers. By [Observation 29](#), the distribution of $(S^+, f(S^+))$ is the same under both $D_0|_{\overline{B_1}}$ and $D_1|_{\overline{B_1}}$. Hence, the distribution of query answers $f(x_1), \dots, f(x_q)$ is identical. The lemma follows. $\square$

Next, we recall some standard definitions and facts that are useful for proving query lower bounds.

Definition 31 (Notation for statistical distance). *For two distributions D_1 and D_2 and a constant δ , let $D_1 \approx_\delta D_2$ denote that the statistical distance between D_1 and D_2 is at most δ .*

Fact 21 (Claim 4 [100]). *Let E be an event that happens with probability at least $1 - \delta$ under the distribution D and let B denote the conditional distribution $D|_E$. Then $B \approx_{\delta'} D$ where $\delta' = \frac{1}{1-\delta} - 1$.*

We use the version of Yao's principle with two distributions from [100].

Fact 22 (Claim 5 [100]). *To prove a lower bound q on the worst-case query complexity of a randomized property testing algorithm, it is enough to give two distributions on inputs: $\mathcal{P}$ on positive instances, and $\mathcal{N}$ on negative instances, such that $\mathcal{P}\text{-view} \approx_\delta \mathcal{N}\text{-view}$ for some $\delta < \frac{1}{3}$.*

We now complete the proof of [Theorem 28](#), the main theorem on distribution-free testing.

Proof of Theorem 28. We apply [Fact 22](#) (Yao's principle) with $\mathcal{P} = D_0|_{\overline{B_0}}$ and $\mathcal{N} = D_1|_{\overline{B_0}}$. By [Lemma 5.4.1](#), $D_0|_{\overline{B_0}}$ is over positive instances and $D_1|_{\overline{B_0}}$ is over negative instances of distribution-free Lipschitz $\frac{1}{2}$ -testing. Let $\delta_0 = \Pr[B_0]$ and $\delta_1 = \Pr[B_1 \cup B_{\mathcal{T}}]$. Set $\delta'_0 = \frac{1}{1-\delta_0} - 1$ and $\delta'_1 = \frac{1}{1-\delta_1} - 1$. By [Fact 21](#), we have the following chain of equivalences:

$$D_0\text{-view}|_{\overline{B_0}} \approx_{\delta'_0} D_0\text{-view} \approx_{\delta'_1} D_0\text{-view}|_{\overline{B_1 \cup B_{\mathcal{T}}}} = D_1\text{-view}|_{\overline{B_1 \cup B_{\mathcal{T}}}} \approx_{\delta'_1} D_1\text{-view} \approx_{\delta'_0} D_1\text{-view}|_{\overline{B}},$$

where the equality follows from [Lemma 5.4.4](#). By [Lemmas 5.4.1](#), [5.4.2](#) and [5.4.3](#) (that upper bound the probabilities of bad events), for sufficiently large d , we have $\delta'_0 \leq \frac{1}{27}$, and $\delta'_1 \leq \frac{1}{9}$. Hence, $2(\delta'_0 + \delta'_1) < \frac{1}{3}$. [Theorem 28](#) now follows from Yao's principle (as stated in [Fact 22](#)). $\square$

5.5 Application to Differential Privacy

In this section, we show how to use a local Lipschitz filter for bounded-range functions to construct a mechanism ([Theorem 30](#)) for privately releasing outputs of bounded-range functions even when the client is malicious (i.e., lies about the range or Lipschitz constant of the function). Then, we show how the mechanism can be extended to privately release outputs of unbounded-range functions ([Theorem 31](#)).

5.5.1 Preliminaries on Differentially Private Mechanisms

We start by defining the Laplace mechanism, used in the proofs of Theorems 30 and 31. It is based on the Laplace distribution, denoted $\text{Laplace}(\lambda)$, that has probability density function $f(x) = \frac{1}{2\lambda}e^{-|x|/\lambda}$. We use abbreviation (ε, δ) -DP for “ (ε, δ) -differentially private” (see Definition 1).

Lemma 5.5.1 (Laplace Mechanism [56]). *Fix $\varepsilon > 0$ and $c > 1$. Let $f : [n]^d \rightarrow \mathbb{R}$ be a c -Lipschitz function. Then the mechanism that gets a query $x \in [n]^d$ as input, samples $N \sim \text{Laplace}(\frac{c}{\varepsilon})$, and outputs $L(x) = f(x) + N$, is $(\varepsilon, 0)$ -DP. Furthermore, for all $\alpha \in (0, 1)$, the mechanism satisfies $|L(x) - f(x)| \leq \frac{c}{\varepsilon} + \ln \frac{1}{\alpha}$ with probability at least $1 - \alpha$.*

In addition to the Laplace mechanism (Lemma 5.5.1), the proof of Theorem 31 uses the following well known facts about differentially private algorithms. These can be found in [41].

Fact 23 (Composition). *Fix $\varepsilon_1, \varepsilon_2 > 0$ and $\delta_1, \delta_2 \in (0, 1)$. Suppose $\mathcal{M}_1$ and $\mathcal{M}_2$ are (respectively) $(\varepsilon_1, \delta_1)$ -DP and $(\varepsilon_2, \delta_2)$ -DP. Then, the mechanism that, on input x , outputs $(\mathcal{M}_1(x), \mathcal{M}_2(x))$ is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$ -DP.*

Fact 24 (Post-processing). *Fix $\varepsilon > 0$ and $\delta \in (0, 1)$. Suppose $\mathcal{M} : D \rightarrow R$ is an (ε, δ) -DP mechanism. If $\mathcal{A}$ is an algorithm with input space R then the algorithm given by $\mathcal{A} \circ \mathcal{M}$ is (ε, δ) -DP.*

Fact 25. *If $X \sim \text{Laplace}(\lambda)$ then $\Pr[|X| \geq t\lambda] \leq e^{-t}$ for all $t > 0$.*

In particular, if $\lambda = \frac{\log r}{\varepsilon}$ and $t = \log(200 \log r)$ then $\Pr[|X| \geq \frac{\log(r) \log(200 \log r)}{\varepsilon}] \leq \frac{1}{200 \log r}$.

5.5.2 Mechanism for Bounded-Range Functions

The filter mechanism can be instantiated with either one of our filters (from Theorem 25 or from Theorem 26), providing slightly different accuracy guarantees. In Theorem 30, we state the guarantees for the mechanism based on the l_1 -respecting filter. Next, we establish the terminology used in the theorem. Recall that $\text{BALL}_R(x)$ denotes the set $\{y : |x - y| < R\}$. We say a vertex $x \in \{0, 1\}^d$ is *dangerous* w.r.t. f if there exists a vertex $y \in \{0, 1\}^d$ such that $|f(x) - f(y)| > \text{dist}_G(x, y)$. A client that submits a Lipschitz function is called *honest*; a client that submits a non-Lipschitz function that is close to Lipschitz is called *clumsy* (the distance measure could be ℓ_1 or ℓ_0 , depending on the filter used). Finally, we assume that sampling from the Laplace distribution requires unit time.

Theorem 30 (Filter mechanism). *For all $\varepsilon > 0$ and $\delta \in (0, 1)$, there exists an (ε, δ) -differentially private mechanism $\mathcal{M}$ that, given a query $x \in [n]^d$, lookup access to a function $f : [n]^d \rightarrow [0, r]$, and range diameter $r \in \mathbb{R}$, outputs a value $h(x) \in \mathbb{R}$, and has the following properties.*

- **Efficiency:** *The lookup and time complexity of $\mathcal{M}$ are $(d^r \cdot \text{polylog}(n/\delta))^{O(\log r)}$.*

- **Accuracy for an honest client:** If f is Lipschitz then for all $x \in [n]^d$ we have $h(x) \sim f(x) + \text{Laplace}(\frac{2}{\varepsilon})$.
- **Accuracy for a clumsy client:** For all $x \in [n]^d$ such that $\text{BALL}_{r \log_{3/2}(r)}(x)$ does not contain any dangerous vertices, the “accuracy for an honest client” guarantee holds. Moreover, with probability at least $1 - 2\delta$, the mechanism satisfies $\mathbb{E}_{z \sim [n]^d} [|h(z) - f(z)|] \leq 2\ell_1(f, \mathcal{Lip}(\mathcal{H}_n^d)) + O(\frac{1}{\varepsilon})$.

We stress that the differential privacy guarantee in [Theorem 30](#) holds whether or not the client is honest.

Proof of [Theorem 30](#). Fix $\varepsilon > 0$ and $\delta \in (0, 1)$. Let $\mathcal{A}$ denote LOCALFILTER_1 ([Algorithm 24](#)) run with iteration parameter $t = \log_{3/2}(r/2) + 1$. Recall that by [Theorem 25](#) instantiated with $\gamma = 1$ and failure probability δ , the algorithm $\mathcal{A}$ is an ℓ_1 -respecting local $(2, \delta)$ -Lipschitz filter with blowup 2 over the d -dimensional hypergrid $\mathcal{H}_n^d$. The “efficiency” and “accuracy for honest client” guarantees hold for any local Lipschitz filter of the type stated in [Theorem 25](#). However, the first guarantee of “accuracy for a clumsy client” requires properties specific to the construction of LOCALFILTER_1 .

Let $\mathcal{M}$ be the following mechanism: *Sample a random seed ρ of length specified in [Theorem 25](#), run $\mathcal{A}(x, \rho)$ to obtain $g_\rho(x)$, sample $N \sim \text{Laplace}(\frac{2}{\varepsilon})$, and output $g_\rho(x) + N$.*

First, we prove that $\mathcal{M}$ is (ε, δ) -differentially private. If the function g_ρ is 2-Lipschitz, then, by [Lemma 5.5.1](#) instantiated with $c = 2$ and privacy parameter ε , the mechanism $\mathcal{M}$ is $(\varepsilon, 0)$ -DP. Conditioned on the event that g_ρ is 2-Lipschitz, we obtain that for all measurable sets $Y \subset \mathbb{R}$,

$$\Pr[\mathcal{M}(x, \rho) \in Y \mid g_\rho \text{ is 2-Lipschitz}] \leq e^\varepsilon \Pr[\mathcal{M}(x', \rho) \in Y \mid g_\rho \text{ is 2-Lipschitz}].$$

By [Theorem 25](#), $\mathcal{A}$ fails to output a 2-Lipschitz function with probability at most δ . By the law of total probability,

$$\begin{aligned} \Pr[\mathcal{M}(x, \rho) \in Y] &\leq e^\varepsilon \Pr[\mathcal{M}(x', \rho) \in Y \mid g_\rho \text{ is 2-Lipschitz}] \Pr[g_\rho \text{ is 2-Lipschitz}] + \delta \\ &\leq e^\varepsilon \Pr[\mathcal{M}(x', \rho) \in Y] + \delta. \end{aligned}$$

The efficiency guarantee follows directly from [Theorem 25](#). The accuracy guarantee for an honest client holds since [Theorem 25](#) guarantees that if f is Lipschitz, then $\mathcal{A}(x, \rho) = f(x)$ for all x and ρ . The average accuracy guarantee for the clumsy client follows from the ℓ_1 -respecting, 2-blowup guarantee of [Theorem 25](#) and the fact that $\mathbb{E}[|\text{Laplace}(\frac{2}{\varepsilon})|] \leq O(\frac{1}{\varepsilon})$. To demonstrate that the stronger accuracy guarantee holds when no dangerous vertices are in $\text{BALL}_{r \log_{3/2}(r)}(x)$, we make the following observation: if no vertex $y \in \text{BALL}_{r \log_{3/2}(r)}(x)$ is dangerous, then $\mathcal{A}(x, \rho) = f(x)$. We prove this claim as follows. Recall that $t = \log_{3/2}(r/2) + 1$ and, for each $i \in [t]$, let $\mathcal{A}(x, i, \rho)$ denote the output after iteration i of [Algorithm 24](#). By construction, $\mathcal{A}(x, 1, \rho) = f(x)$. Suppose no vertex $y \in \text{BALL}_{r(t-1)}(x)$ is dangerous. Then, since the range of f is $[0, r]$, every dangerous vertex v can, in a single iteration, only create

new dangerous vertices in $\text{BALL}_r(v)$. Thus, in $t - 1$ iterations, no dangerous vertices can be introduced in $\text{BALL}_r(x)$. Since $\mathcal{A}(x, 1, \rho) = f(x)$, and in every subsequent iteration $1 < i \leq t$, no dangerous vertices are in $\text{BALL}_{r(i-1)}(x)$, we obtain $\mathcal{A}(x, t, \rho) = \mathcal{A}(x, 1, \rho) = f(x)$ for each $i \in [t]$. Thus, if no dangerous vertex is in $\text{BALL}_{r \log_{3/2}(r)}(x)$ then $\mathcal{A}(x, \rho) = f(x)$, and hence, the “accuracy for an honest client” guarantee holds. $\square$

5.5.3 Mechanism for Unbounded Range Functions

In this section, we use the mechanism for bounded-range functions to construct a mechanism for arbitrary-range functions.

Theorem 31 (Binary search filter mechanism). *For all $\varepsilon > 0$ and $\delta \in (0, \frac{1}{200})$, there exists an (ε, δ) -differentially private mechanism $\mathcal{M}$ that, given a query $x \in [n]^d$, lookup access to a function $f : [n]^d \rightarrow [0, \infty)$ and an optional range parameter $r \in \mathbb{R}$, outputs value $h(x) \in \mathbb{R}$ and has the following properties.*

Let $\kappa = \log \min(r, nd)$, where the optional parameter r is set to ∞ by default.

- **Efficiency:** *The lookup and time complexity of $\mathcal{M}$ are $d^{O(\frac{1}{\varepsilon} \kappa \log \kappa)} \text{polylog}(\frac{n}{\delta})$.*
- **Accuracy for an honest client:** *If f is Lipschitz then, for all $x \in [n]^d$, we have $h(x) \sim f(x) + \text{Laplace}(\frac{\kappa}{\varepsilon})$ with probability at least 0.99.*
- **Accuracy for a clumsy client:** *There exists a constant $c > 0$ such that for all $x \in [n]^d$, if $f(x) \leq nd$ and $|f(x) - f(y)| \leq |x - y|$ for all $y \in \text{BALL}_{\frac{c}{\varepsilon} \kappa \log \kappa}(x)$, then the “accuracy for an honest client” guarantee holds.*

As in [Theorem 30](#), we emphasize that the differential privacy guarantee holds whether or not the client is honest. Note that the accuracy guarantee for an honest client is subsumed by the guaranty for a clumsy client, but we state the former guarantee separately for clarity.

Proof of [Theorem 31](#). Our private mechanism is presented in [Algorithm 27](#). It uses the following “clipping” operation to truncate the range of a function.

Definition 32 (Clipped function). *For any $f : V \rightarrow \mathbb{R}$ and interval $[\ell, u] \subset \mathbb{R}$, the clipped function $f[\ell, u]$ is defined by*

$$f[\ell, u](x) = \begin{cases} f(x) & f(x) \in [\ell, u]; \\ \ell & f(x) < \ell; \\ u & f(x) > u. \end{cases}$$

Algorithm 27 Binary search filter mechanism $\mathcal{M}(x, f, \varepsilon)$

Input: Dataset $x \in [n]^d$, lookup access to $f : [n]^d \rightarrow [0, \infty)$, range diameter $r \in \mathbb{R}$, adjacency lists access to the hypercube $\mathcal{H}^d$, $\varepsilon > 0$, and $\delta \in (0, 1)$

Subroutines: Local $(1, \frac{\delta}{\log r})$ -Lipschitz filter $\mathcal{A}$ obtained in [Theorem 26](#)

Output: Noisy value $h(x)$ satisfying the guarantees of [Theorem 31](#)

```
1: set  $r \leftarrow \min(r, nd)$  and  $f \leftarrow f[0, r]$  ▷ If the client is honest then  $f[0, r] = f$ .
2: set  $t \leftarrow r/2$  and  $\alpha \leftarrow \frac{1}{\varepsilon} \log(r) \log(200 \log r)$ 
3: for  $i = 2$  to  $\lceil \log r \rceil$  do
4:   let  $h(x) \leftarrow \mathcal{A}(x, f[t - 2\alpha, t + 2\alpha]) + \text{Laplace}(\frac{\log r}{\varepsilon})$ 
5:   if  $h(x) \in [t - \alpha, t + \alpha]$  then
6:     return  $h(x)$ 
7:   else  $t \leftarrow t + \text{sign}(h(x) - t) \cdot \lceil r/2^i \rceil$ 
8: return  $h(x)$ 
```

Next, we complete the analysis of the binary search filter mechanism. We first argue that $\mathcal{M}$ ([Algorithm 27](#)) is (ε, δ) -DP. In every iteration of the **for**-loop, $\mathcal{M}$ uses Laplace mechanism on a function that is 1-Lipschitz with probability at least $1 - \frac{\delta}{\log r}$. By [Lemma 5.5.1](#) and an argument similar to the proof of [Theorem 30](#), each iteration is $(\frac{\varepsilon}{\log r}, \frac{\delta}{\log r})$ -DP. It follows by [Facts 23](#) and [24](#) that [Algorithm 27](#) is (ε, δ) -DP.

Next, we prove the accuracy guarantee for the clumsy client. Observe that it subsumes the accuracy guarantee for the honest client. Suppose $f : [n]^d \rightarrow [0, r]$ and that x satisfies $|f(x) - f(y)| \leq |x - y|$ for all $y \in \text{BALL}_\alpha(x)$ (the α in line 2 of [Algorithm 27](#)). Then, for all intervals $\mathcal{I}$ of diameter α , the point x satisfies $|f[\mathcal{I}](x) - f[\mathcal{I}](y)| \leq |x - y|$ for all $y \in [n]^d$. By [Theorem 26](#), $\mathcal{A}(x, f[\mathcal{I}]) = f[\mathcal{I}](x)$, which is equal to $f(x)$ whenever $f(x) \in \mathcal{I}$.

Condition on the event that $\mathcal{A}$ does not fail and that the Laplace noise added is strictly less than α in every iteration of $\mathcal{M}$. Then, if $h(x) \in [t - \alpha, t + \alpha]$, we must have $f[t - 2\alpha, t + 2\alpha](x) \in (t - 2\alpha, t + 2\alpha)$, and therefore $f[t - 2\alpha, t + 2\alpha](x) = f(x)$. Next, suppose $h(x) \notin [t - \alpha, t + \alpha]$. If $f(x) < t$ then $h(x) < t + \alpha$ and thus $h(x) < t - \alpha$. Similarly, if $f(x) > t$ then $h(x) > t - \alpha$ and thus $h(x) > t + \alpha$. It follows that in every iteration $\mathcal{M}$ either continues the binary search in the correct direction, or halts and outputs $h(x)$ such that $|h(x) - f(x)| \leq \alpha$. By the union bound and the guarantee obtained in [Theorem 26](#), the algorithm $\mathcal{A}$ fails in some iteration of $\mathcal{M}$ with probability at most δ . Moreover, by [Fact 25](#) and the union bound, the Laplace noise added is at least α in some iteration of $\mathcal{M}$ with probability at most $\frac{1}{200}$. It follows that for sufficiently small δ , the mechanism $\mathcal{M}$ outputs $h(x)$ such that $h(x) \sim f(x) + \text{Laplace}(\frac{\log \min(r, nd)}{\varepsilon})$ with probability at least $\frac{99}{100}$. $\square$

5.6 Application to Tolerant Testing

In this section, we give an efficient algorithm for tolerant Lipschitz testing of real-valued functions over the d -dimensional hypercube $\mathcal{H}^d$ and prove [Theorem 32](#), which we restate here

for convenience.

Theorem 32. *For all $\varepsilon \in (0, \frac{1}{3})$ and all sufficiently large $d \in \mathbb{N}$, there exists an $(\varepsilon, 2.01\varepsilon)$ -tolerant tester for the Lipschitz property of functions on the hypercube $\mathcal{H}^d$. The tester has query and time complexity $\frac{1}{\varepsilon^2} d^{O(\sqrt{d \log(d/\varepsilon)})}$.*

Our tester utilizes the fact that the image of a function which is close to Lipschitz exhibits a strong concentration about its mean on most of the points in the domain. Hence, if a function is close to Lipschitz, it can be truncated to a small interval around its mean without modifying too many points. This truncation guarantees that the local filter in [Theorem 26](#) runs in time subexponential in d . A key idea in the truncation procedure is that if a function f is ε -close to Lipschitz then either not very many values are truncated, or the truncated function is close to Lipschitz.

To prove [Theorem 32](#), we design an algorithm ([Algorithm 28](#)) that, for functions $f : \{0, 1\}^d \rightarrow \mathbb{R}$, accepts if f is ε -close to Lipschitz, rejects if f is 2.01ε -far from Lipschitz and fails with probability at most $\frac{45}{100}$. The success probability can then be amplified to at least $\frac{2}{3}$ by repeating the algorithm $\Theta(1)$ times and taking the majority answer. Before presenting [Algorithm 28](#), we introduce some additional notation. For all functions f and intervals $\mathcal{I} \subset \mathbb{R}$, the partial function $f_{\mathcal{I}}$ is defined by $f_{\mathcal{I}}(x) = f(x)$ whenever $f(x) \in \mathcal{I}$ and $f_{\mathcal{I}}(x) = ?$ otherwise. Additionally, for all events E , let $\mathbf{1}_E$ denote the indicator for the event E . [Algorithm 28](#) runs the ℓ_0 -filter given by [Algorithm 26](#) with lookup access to a partial function $f_{\mathcal{I}}$. Our analysis of [Algorithm 26](#) presented in [Theorem 26](#) is for total functions. In [Observation 33](#), we extend it to partial functions.

Observation 33. Let $h : [n]^d \rightarrow \mathbb{R} \cup \{?\}$ be a partial function with ℓ_0 -distance to the nearest Lipschitz partial function equal to ε_h . Let $\mathcal{A}$ denote [Algorithm 26](#). Then, for all $\delta \in (0, 1)$, the algorithm $\mathcal{A}^h$ provides query access to a Lipschitz partial function g such that $g(x) = ?$ if and only if $h(x) = ?$, and $\|g - h\|_0 \leq 2\varepsilon_h$. The runtime and failure probability guarantees are as in [Theorem 26](#).

Proof. Consider $x \in h^{-1}(?)$. By [Definition 21](#), $VS_h(x, y) = 0$ for all $y \in [n]^d$, and hence the vertex x is not incident to any edge of the violation graph of h . By construction, $\mathcal{A}^h(x) = h(x) = ?$. Next, consider the induced subgraph G of $\mathcal{H}^d$ with vertex set $V = \{x : h(x) \neq ?\}$. By [Claim 22](#), the size of the minimum vertex cover of the violation graph of h is at most $|V|\varepsilon_h$. It follows that $\mathcal{A}^h$ provides query access to a Lipschitz partial function g such that $\|g - h\|_0 \leq \varepsilon_h$. The runtime and failure probability are the same as for total functions by definition of the algorithm. $\square$

Algorithm 28 Tolerant Lipschitz tester $\mathcal{T}(f, \varepsilon)$

Input: Query access to $f : \{0, 1\}^d \rightarrow \mathbb{R}$, adjacency lists access to $\mathcal{H}^d$, and $\varepsilon \in (0, \frac{1}{3})$
Subroutines: Local $(1, \frac{1}{100})$ -Lipschitz filter $\mathcal{A}$ given by [Algorithm 26](#)
Output: accept or reject

- 1: sample a point $p \sim \{0, 1\}^d$ uniformly at random and a random seed ρ of length specified in [Theorem 26](#)
- 2: set $t \leftarrow 2\sqrt{d \log(d/\varepsilon)}$ and $\mathcal{I} \leftarrow [f(p) - t, f(p) + t]$
- 3: sample a set S of $(\frac{1500}{\varepsilon})^2$ points uniformly and independently from $\{0, 1\}^d$
- 4: **for all** $x_i \in S$ **do**
- 5: **if** $f_{\mathcal{I}}(x_i) = ?$ **then** set $y_i \leftarrow ?$
- 6: **else** set $y_i \leftarrow \mathcal{A}(x_i, \rho)$, where $\mathcal{A}$ is run with lookup access to $f_{\mathcal{I}}$ and adjacency lists access to $\mathcal{H}^d$
- 7: **if** $\frac{1}{|S|} \sum_{x_i \in S} \mathbf{1}_{f(x_i) \neq y_i} < 2.005\varepsilon$ **then accept**
- 8: **else reject**

We use McDiarmid's inequality [\[92\]](#), stated here for the special case of the $\{0, 1\}^d$ domain.

Fact 26 (McDiarmid's Inequality[\[92\]](#)). *Fix $d \geq 2$ and let $g : \{0, 1\}^d \rightarrow \mathbb{R}$ be a Lipschitz function w.r.t. $\mathcal{H}^d$. Let $\mu_g = \mathbb{E}_{x \sim \{0, 1\}^d} [g(x)]$. Then, for all $\gamma \in (0, 1)$,*

$$\Pr_{x \sim \{0, 1\}^d} [|g(x) - \mu_g| \geq \sqrt{d \log(d/\gamma)}] \leq \frac{\gamma}{d}.$$

Next, we introduced a definition which, for each function f , attributes some part of its ℓ_0 -distance to Lipschitz to a particular interval $\mathcal{I}$ in the range of f .

Definition 33. *Let $f : \{0, 1\}^d \rightarrow \mathbb{R}$ and C be a minimum vertex cover of the violation graph of f . (If there are multiple vertex covers, use any rule to pick a canonical one.) For an interval $\mathcal{I} \subset \mathbb{R}$ define $\varepsilon[\mathcal{I}]$ as $|\{x \in C : f(x) \in \mathcal{I}\}|/2^d$.*

For a function f , let ε_f denote the ℓ_0 -distance from f to Lipschitz. Then $\varepsilon_f = \varepsilon[\mathcal{I}] + \varepsilon[\overline{\mathcal{I}}]$ for all intervals $\mathcal{I}$. Moreover, since $f_{\mathcal{I}}$ is a partial function defined only on points x such that $f(x) \in \mathcal{I}$, the distance from $f_{\mathcal{I}}$ to the nearest Lipschitz partial function is at most $\varepsilon[\mathcal{I}]$. Using [Definition 33](#), we argue that if $\varepsilon_f \leq \varepsilon$, then with high probability the interval $\mathcal{I}$ chosen in [Algorithm 28](#) satisfies $\|f_{\mathcal{I}} - f\|_0 \leq \varepsilon[\overline{\mathcal{I}}] + \frac{\varepsilon}{d}$. Since $\varepsilon[\mathcal{I}] + \varepsilon[\overline{\mathcal{I}}] \leq \varepsilon$, [Lemma 5.6.1](#) implies that if $\|f_{\mathcal{I}} - f\|_0$ is very close to ε , then the distance of $f_{\mathcal{I}}$ to Lipschitz, which is at most $\varepsilon[\mathcal{I}]$, must be small. Leveraging this fact, we can approximate the ℓ_0 -distance from f to Lipschitz using the distance from f to $f_{\mathcal{I}}$ and the distance from $f_{\mathcal{I}}$ to Lipschitz.

Lemma 5.6.1. *Fix $\varepsilon \in (0, \frac{1}{3})$ and $d \geq 4$. Let $f : \{0, 1\}^d \rightarrow \mathbb{R}$ be ε -close to Lipschitz over $\mathcal{H}^d$. Choose $p \in \{0, 1\}^d$ uniformly at random. Set $t \leftarrow 2\sqrt{d \log(d/\varepsilon)}$ and $\mathcal{I} \leftarrow [f(p) - t, f(p) + t]$. Then $\|f - f_{\mathcal{I}}\|_0 > \varepsilon[\overline{\mathcal{I}}] + \frac{\varepsilon}{d}$ with probability at most $\frac{5}{12}$.*

Proof. Let C be a minimum vertex cover of the violation graph $B_{0,f}$ (see [Definition 22](#)). Let g be a Lipschitz function obtained by extending f from $\{0, 1\}^d \setminus C$ to $\{0, 1\}^d$ (such an

extension exists by [Claim 21](#)). By [Fact 26](#), $\Pr_{x \sim \{0,1\}^d} [|g(x) - \mu_g| \geq \frac{t}{2}] \leq \frac{\varepsilon}{d}$. Notice that if $|f(p) - \mu_g| \leq \frac{t}{2}$ then $[\mu_g - \frac{t}{2}, \mu_g + \frac{t}{2}] \subset \mathcal{I}$. Conditioned on this occurring,

$$\Pr_x[f(x) \notin \mathcal{I}] \leq \varepsilon[\overline{\mathcal{I}}] + \Pr_x[g(x) \notin \mathcal{I}] \leq \varepsilon[\overline{\mathcal{I}}] + \frac{\varepsilon}{d}.$$

Since $\Pr_{x \sim \{0,1\}^d}[x \in C] \leq \varepsilon$, we have $|f(p) - \mu_g| \leq \frac{t}{2}$ with probability at most $\varepsilon + \frac{\varepsilon}{d} \leq \frac{5}{12}$. $\square$

Next, we argue that, after boosting the success probability via standard amplification techniques, we obtain a $(\varepsilon, 2.01\varepsilon)$ -tolerant Lipschitz tester.

Proof of [Theorem 32](#). Fix $\varepsilon \in (0, \frac{1}{3})$ and let $f : \{0,1\}^d \rightarrow \mathbb{R}$ and let $\mathcal{T}$ denote [Algorithm 28](#). Define the following events: Let E_1 be the event that local filter $\mathcal{A}$ fails. Set $\omega = \Pr_x[f(x) \neq \mathcal{A}(x, \rho)]$ and $\hat{\omega} = \frac{1}{|S|} \sum_{x_i \in S} \mathbf{1}_{f(x_i) \neq y_i}$, and let E_2 be the event that $|\omega - \hat{\omega}| \geq \frac{\varepsilon}{300}$.

Suppose f is ε -close to Lipschitz, and let E_3 be the event that the interval $\mathcal{I}$ chosen in $\mathcal{T}$ satisfies $\|f - f_{\mathcal{I}}\| > \varepsilon[\overline{\mathcal{I}}] + \frac{\varepsilon}{d}$. Condition on the event that none of E_1, E_2 , and E_3 occur. Then $f_{\mathcal{I}}$ is at distance at most $\varepsilon[\mathcal{I}]$ from some Lipschitz partial function and, by [Observation 33](#), $\mathcal{A}$ provides query access to a Lipschitz partial function g such that $\|f - g\|_0 \leq 2\varepsilon[\mathcal{I}]$. Using the fact that $\varepsilon[\mathcal{I}] + \varepsilon[\overline{\mathcal{I}}] \leq \varepsilon$ we obtain

$$\hat{\omega} \leq 2\varepsilon[\mathcal{I}] + \varepsilon[\overline{\mathcal{I}}] + \frac{\varepsilon}{d} + \frac{\varepsilon}{300} < 2.005\varepsilon$$

for sufficiently large d . Hence $\mathcal{T}$ accepts.

Now consider the case that f is 2.01ε -far from Lipschitz and suppose neither of the events E_1 and E_2 occur. Since $\mathcal{A}$ provides query access to a Lipschitz function, and the nearest Lipschitz function is at distance at least 2.01ε , we must have $\hat{\omega} \geq 2.01\varepsilon - \frac{\varepsilon}{300} > 2.005\varepsilon$. Hence $\mathcal{T}$ rejects.

Next, we show that the events E_1, E_2 and E_3 all occur with small probability. By [Observation 33](#), $\Pr[E_1] \leq \frac{1}{100}$. To bound $\Pr[E_2]$, notice that $\mathbb{E}[\hat{\omega}] = \omega$ and $\text{Var}[\hat{\omega}] \leq \frac{1}{4|S|}$. By Chebyshev's inequality and our choice of $|S|$ we have, $\Pr_S[|\omega - \hat{\omega}| \geq \frac{\varepsilon}{300}] \leq \frac{300^2}{4|S|\varepsilon^2} \leq \frac{1}{100}$. Moreover, if f is ε -close to Lipschitz, then by [Lemma 5.6.1](#), $\Pr[E_3] \leq \frac{5}{12}$. Thus, the failure probability of $\mathcal{T}$ can be bounded above by $\Pr[E_1 \cup E_2 \cup E_3] \leq \frac{2}{100} + \frac{5}{12} \leq \frac{45}{100}$. The success probability can be boosted to $\frac{2}{3}$ by running the algorithm $O(1)$ times and taking the majority answer. Finally, we bound the query and time complexity of tester by bounding the query and time complexity of $\mathcal{T}$. The algorithm $\mathcal{T}$ runs the local filter $\mathcal{A}$ with lookup access to $f_{\mathcal{I}}$, a function with range $\mathcal{I}$ of diameter $O(\sqrt{d \log(d/\varepsilon)})$, and sets $\mathcal{A}$'s failure probability to $\delta = \frac{1}{100}$. Consequently, [Observation 33](#) implies $\mathcal{T}$ has query and time complexity bound of $\frac{1}{\varepsilon^2} d^{O(\sqrt{d \log(d/\varepsilon)})}$. $\square$

References

- [1] N. Ailon, B. Chazelle, C. Seshadhri, and D. Liu. “Estimating the distance to a monotone function.” *Random Structures & Algorithms*, **31**(3), 2007, pp. 371–383. DOI: <https://doi.org/10.1002/rsa.20167>.
- [2] N. Ailon, B. Chazelle, C. Seshadhri, and D. Liu. “Property-Preserving Data Reconstruction.” *Algorithmica*, **51**(2), 2008, pp. 160–182.
- [3] N. Alon, R. Rubinfeld, S. Vardi, and N. Xie. “Space-efficient local computation algorithms.” *Transactions on Computation*, **3**(1), 2012, pp. 9–24.
- [4] K. Amano and A. Maruoka. “On learning monotone Boolean functions under the uniform distribution.” *Theor. Comput. Sci.*, **350**(1), 2006, pp. 3–12. DOI: [10.1016/j.tcs.2005.10.012](https://doi.org/10.1016/j.tcs.2005.10.012). URL: <https://doi.org/10.1016/j.tcs.2005.10.012>.
- [5] D. Angluin. “Queries and Concept Learning.” *Mach. Learn.*, **2**(4), Apr. 1988, pp. 319–342. ISSN: 0885-6125. DOI: [10.1023/A:1022821128753](https://doi.org/10.1023/A:1022821128753). URL: <https://doi.org/10.1023/A:1022821128753>.
- [6] M. Anthony. “Classification by polynomial surfaces.” *Discrete Appl. Math.*, **61**(2), July 1995, pp. 91–103. ISSN: 0166-218X. DOI: [10.1016/0166-218X\(94\)00008-2](https://doi.org/10.1016/0166-218X(94)00008-2). URL: [https://doi.org/10.1016/0166-218X\(94\)00008-2](https://doi.org/10.1016/0166-218X(94)00008-2).
- [7] R. Arviv and R. Levi. “Improved LCAs for constructing spanners.” *CoRR*, **abs/2105.04847**, 2021.
- [8] P. Awasthi, M. F. Balcan, and P. M. Long. “The power of localization for efficiently learning linear separators with noise.” *Journal of the ACM (JACM)*, **63**(6), 2017, pp. 1–27.
- [9] P. Awasthi, A. Dutta, and A. Vijayaraghavan. “On Robustness to Adversarial Examples and Polynomial Optimization.” In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett. 2019, pp. 13737–13747.
- [10] P. Awasthi, M. Jha, M. Molinaro, and S. Raskhodnikova. “Limitations of Local Filters of Lipschitz and Monotone Functions.” *ACM Trans. Comput. Theory*, **7**(1), Jan. 2015. ISSN: 1942-3454. DOI: [10.1145/2692372.2692373](https://doi.org/10.1145/2692372.2692373). URL: <https://doi.org/10.1145/2692372.2692373>.

- [11] P. Awasthi, M. Jha, M. Molinaro, and S. Raskhodnikova. “Testing Lipschitz Functions on Hypergrid Domains.” *Algorithmica*, **74**(3), 2016, pp. 1055–1081. DOI: [10.1007/s00453-015-9984-y](https://doi.org/10.1007/s00453-015-9984-y). URL: <https://doi.org/10.1007/s00453-015-9984-y>.
- [12] A. Belovs and E. Blais. “A Polynomial Lower Bound for Testing Monotonicity.” *SIAM J. Comput.*, **50**(3), 2021. DOI: [10.1137/16M1097006](https://doi.org/10.1137/16M1097006). URL: <https://doi.org/10.1137/16M1097006>.
- [13] Y. Benyamini and J. Lindenstrauss. *Geometric nonlinear functional analysis. Vol. 1*. Vol. 48. Colloquium Publications. Amer. Math. Soc., 2000, pp. xii+488. ISBN: 0-8218-0835-4.
- [14] P. Berman, S. Raskhodnikova, and G. Yaroslavtsev. “ L_p -testing.” In: *Proceedings, ACM Symposium on Theory of Computing (STOC)*. 2014, pp. 164–173. DOI: [10.1145/2591796.2591887](https://doi.org/10.1145/2591796.2591887). URL: <https://doi.org/10.1145/2591796.2591887>.
- [15] A. Bhattacharyya, E. Grigorescu, M. Jha, K. Jung, S. Raskhodnikova, and D. Woodruff. “Lower bounds for local monotonicity reconstruction from transitive-closure spanners.” In: *Approximation, Randomization, and Combinatorial Optimization*. 2010, pp. 448–461. ISBN: 3642153682. DOI: [10.1007/978-3-642-15369-3_34](https://doi.org/10.1007/978-3-642-15369-3_34).
- [16] A. Bhattacharyya, E. Grigorescu, M. Jha, K. Jung, S. Raskhodnikova, and D. P. Woodruff. “Lower Bounds for Local Monotonicity Reconstruction from Transitive-Closure Spanners.” *SIAM J. Discrete Math.*, **26**(2), 2012, pp. 618–646.
- [17] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli. “Evasion Attacks against Machine Learning at Test Time.” In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*. Ed. by H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezný. Vol. 8190. Lecture Notes in Computer Science. Springer, 2013, pp. 387–402. DOI: [10.1007/978-3-642-40994-3_25](https://doi.org/10.1007/978-3-642-40994-3_25). URL: https://doi.org/10.1007/978-3-642-40994-3_25.
- [18] H. Black, D. Chakrabarty, and C. Seshadhri. “A $o(d) \cdot \text{polylog } n$ Monotonicity Tester for Boolean Functions over the Hypergrid $[n]^d$.” In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. SIAM, 2018, pp. 2133–2151. DOI: [10.1137/1.9781611975031.139](https://doi.org/10.1137/1.9781611975031.139).
- [19] H. Black, D. Chakrabarty, and C. Seshadhri. “Domain Reduction for Monotonicity Testing: A $o(d)$ Tester for Boolean Functions in d -Dimensions.” In: *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*. SIAM, 2020, pp. 1975–1994. DOI: [10.1137/1.9781611975994.122](https://doi.org/10.1137/1.9781611975994.122).
- [20] E. Blais, C. L. Canonne, I. C. Oliveira, R. A. Servedio, and L.-Y. Tan. “Learning Circuits with Few Negations.” *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2015, p. 512.

- [21] E. Blais, S. Raskhodnikova, and G. Yaroslavtsev. “Lower Bounds for Testing Properties of Functions over Hypergrid Domains.” In: *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*. IEEE Computer Society, 2014, pp. 309–320. DOI: [10.1109/CCC.2014.38](https://doi.org/10.1109/CCC.2014.38).
- [22] G. Blanc, J. Lange, M. Qiao, and L.-Y. Tan. “Properly learning decision trees in almost polynomial time.” *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2022, pp. 920–929.
- [23] A. Blum, C. Burch, and J. Langford. “On Learning Monotone Boolean Functions.” In: *39th Annual Symposium on Foundations of Computer Science, FOCS ’98, November 8-11, 1998, Palo Alto, California, USA*. IEEE Computer Society, 1998, pp. 408–415. DOI: [10.1109/SFCS.1998.743491](https://doi.org/10.1109/SFCS.1998.743491). URL: <https://doi.org/10.1109/SFCS.1998.743491>.
- [24] S. Brandt, C. Grunau, and V. Rozhon. “The randomized local computation complexity of the Lovász local lemma.” *CoRR*, **abs/2103.16251**, 2021.
- [25] N. H. Bshouty and C. Tamon. “Fourier methods for learning monotone Boolean functions.” *Theoretical Computer Science*, **155**(1), 1996, pp. 157–180.
- [26] S. Bubeck, Y. T. Lee, E. Price, and I. Razenshteyn. “Adversarial examples from computational constraints.” In: *International Conference on Machine Learning*. PMLR, 2019, pp. 831–840.
- [27] A. Campagna, A. Guo, and R. Rubinfeld. “Local reconstructors and tolerant testers for connectivity and diameter.” In: *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, 2013, pp. 411–424.
- [28] C. L. Canonne, E. Grigorescu, S. Guo, A. Kumar, and K. Wimmer. “Testing k-Monotonicity.” *CoRR*, **abs/1609.00265**, 2016.
- [29] C. Carathéodory. “Über den Variabilitätsbereich der Koeffizienten von Potenzreihen, die gegebene Werte nicht annehmen. (Mit 2 Figuren im Text).” *Mathematische Annalen*, **64**, 1907, pp. 95–115. URL: <http://eudml.org/doc/158305>.
- [30] D. Chakrabarty, K. Dixit, M. Jha, and C. Seshadhri. “Property Testing on Product Distributions: Optimal Testers for Bounded Derivative Properties.” *ACM Transactions on Algorithms (TALG)*, **13**(2), 2017, 20:1–20:30. DOI: [10.1145/3039241](https://doi.org/10.1145/3039241).
- [31] D. Chakrabarty and C. Seshadhri. “An Optimal Lower Bound for Monotonicity Testing over Hypergrids.” In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*. Vol. 8096. Lecture Notes in Computer Science. Springer, 2013, pp. 425–435. DOI: [10.1007/978-3-642-40328-6_30](https://doi.org/10.1007/978-3-642-40328-6_30).
- [32] D. Chakrabarty and C. Seshadhri. “Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids.” In: *Symposium on Theory of Computing Conference, STOC’13, Palo Alto, CA, USA, June 1-4, 2013*. ACM, 2013, pp. 419–428. DOI: [10.1145/2488608.2488661](https://doi.org/10.1145/2488608.2488661).

- [33] D. Chakrabarty and C. Seshadhri. “Adaptive Boolean Monotonicity Testing in Total Influence Time.” In: *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*. Vol. 124. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 20:1–20:7. DOI: [10.4230/LIPIcs.ITCS.2019.20](https://doi.org/10.4230/LIPIcs.ITCS.2019.20).
- [34] Y.-J. Chang, M. Fischer, M. Ghaffari, J. Uitto, and Y. Zheng. “The Complexity of $(\Delta+1)$ Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation.” In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. ACM, 2019, pp. 471–480.
- [35] X. Chen, A. De, Y. Li, S. Nadimpalli, and R. A. Servedio. “Mildly Exponential Lower Bounds on Tolerant Testers for Monotonicity, Unateness, and Juntas.” In: *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*. Ed. by D. P. Woodruff. SIAM, 2024, pp. 4321–4337. DOI: [10.1137/1.9781611977912.151](https://doi.org/10.1137/1.9781611977912.151). URL: <https://doi.org/10.1137/1.9781611977912.151>.
- [36] X. Chen, R. A. Servedio, and L.-Y. Tan. “New Algorithms and Lower Bounds for Monotonicity Testing.” In: *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. ISSN: 0272-5428. Oct. 2014. DOI: [10.1109/FOCS.2014.38](https://doi.org/10.1109/FOCS.2014.38).
- [37] X. Chen and E. Waingarten. “Testing unateness nearly optimally.” In: *Proceedings of ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 547–558.
- [38] X. Chen, E. Waingarten, and J. Xie. “Beyond Talagrand functions: new lower bounds for testing monotonicity and unateness.” In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2017. New York, NY, USA: Association for Computing Machinery, June 2017. ISBN: 978-1-4503-4528-6. DOI: [10.1145/3055399.3055461](https://doi.org/10.1145/3055399.3055461). URL: <https://doi.org/10.1145/3055399.3055461> (visited on 03/23/2022).
- [39] J. Cohen, E. Rosenfeld, and Z. Kolter. “Certified adversarial robustness via randomized smoothing.” In: *international conference on machine learning*. PMLR. 2019, pp. 1310–1320.
- [40] D. Cullina, A. N. Bhagoji, and P. Mittal. “Pac-learning in the presence of adversaries.” *Advances in Neural Information Processing Systems*, **31**, 2018.
- [41] A. R. Cynthia Dwork. *The Algorithmic Foundations of Differential Privacy*. NOW, 2014.
- [42] A. Daniely. “A PTAS for agnostically learning halfspaces.” In: *Conference on Learning Theory*. PMLR. 2015, pp. 484–502.
- [43] S. Dasgupta, D. Hsu, and N. Verma. “A concentration theorem for projections.” In: *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*. 2006, pp. 114–121.

- [44] A. Degwekar, P. Nakkiran, and V. Vaikuntanathan. “Computational limitations in robust classification and win-win results.” In: *Conference on Learning Theory*. PMLR. 2019, pp. 994–1028.
- [45] L. Devroye, A. Mehrabian, and T. Reddad. “The total variation distance between high-dimensional Gaussians with the same mean.” *arXiv preprint arXiv:1810.08693*, 2018.
- [46] I. Diakonikolas, P. Gopalan, R. Jaiswal, R. Servedio, and E. Viola. *Bounded Independence Fools Halfspaces*. Number: arXiv:0902.3757 arXiv:0902.3757 [cs]. Feb. 2009. DOI: [10.48550/arXiv.0902.3757](https://arxiv.org/abs/0902.3757). URL: <http://arxiv.org/abs/0902.3757> (visited on 09/07/2022).
- [47] I. Diakonikolas, D. Kane, and P. Manurangsi. “Nearly Tight Bounds for Robust Proper Learning of Halfspaces with a Margin.” In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett. 2019, pp. 10473–10484.
- [48] I. Diakonikolas, D. M. Kane, V. Kontonis, C. Tzamos, and N. Zarifis. “Agnostic Proper Learning of Halfspaces under Gaussian Marginals.” In: *Proceedings of Thirty Fourth Conference on Learning Theory*. Ed. by M. Belkin and S. Kpotufe. Vol. 134. Proceedings of Machine Learning Research. PMLR, Aug. 2021, pp. 1522–1551. URL: <https://proceedings.mlr.press/v134/diakonikolas21b.html>.
- [49] I. Diakonikolas, D. M. Kane, T. Pittas, and N. Zarifis. “The optimality of polynomial regression for agnostic learning under gaussian marginals in the SQ model.” In: *Conference on Learning Theory*. PMLR. 2021, pp. 1552–1584.
- [50] I. Diakonikolas, D. M. Kane, and A. Stewart. “Learning geometric concepts with nasty noise.” In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 2018, pp. 1061–1073.
- [51] I. Diakonikolas, D. M. Kane, and P. Manurangsi. “The complexity of adversarially robust proper learning of halfspaces with agnostic noise.” In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS ’20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.
- [52] K. Dixit, M. Jha, S. Raskhodnikova, and A. Thakurta. “Testing the Lipschitz Property over Product Distributions with Applications to Data Privacy.” In: *Theory of Cryptography Conference (TCC)*. 2013, pp. 418–436. DOI: [10.1007/978-3-642-36594-2_24](https://doi.org/10.1007/978-3-642-36594-2_24). URL: https://doi.org/10.1007/978-3-642-36594-2_24.
- [53] K. Dixit, S. Raskhodnikova, A. Thakurta, and N. Varma. “Erasure-Resilient Property Testing.” *SIAM Journal on Computing (SICOMP)*, **47**(2), 2018, pp. 295–329. DOI: [10.1137/16M1075661](https://doi.org/10.1137/16M1075661). URL: <https://doi.org/10.1137/16M1075661>.

- [54] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. “Improved Testing Algorithms for Monotonicity.” In: *RANDOM-APPROX’99, Berkeley, CA, USA, August 8-11, 1999, Proceedings*. Vol. 1671. Lecture Notes in Computer Science. Springer, 1999, pp. 97–108.
- [55] J. Duchi. *VC-Dimension, Covering, and Packing*. Accessed: 2025-05-11 <https://web.stanford.edu/class/dimension.pdf>. 2023.
- [56] C. Dwork, F. McSherry, K. Nissim, and A. Smith. “Calibrating Noise to Sensitivity in Private Data Analysis.” In: *TCC*. 2006, pp. 265–284.
- [57] G. Even, R. Levi, M. Medina, and A. Rosén. “Sublinear Random Access Generators for Preferential Attachment Graphs.” *ACM Trans. Algorithms*, **17**(4), 2021, 28:1–28:26.
- [58] G. Even, M. Medina, and D. Ron. “Best of Two Local Models: Local Centralized and Local Distributed Algorithms.” *CoRR*, **abs/1402.3796**, 2014. arXiv: [1402.3796](https://arxiv.org/abs/1402.3796). URL: <http://arxiv.org/abs/1402.3796>.
- [59] V. Feldman, P. Kothari, and J. Vondrák. “Tight bounds on l_1 approximation and learning of self-bounding functions.” en. *Theoretical Computer Science*. Special Issue on Algorithmic Learning Theory, **808**, Feb. 2020, pp. 86–98. ISSN: 0304-3975. DOI: [10.1016/j.tcs.2019.11.013](https://doi.org/10.1016/j.tcs.2019.11.013). URL: <https://www.sciencedirect.com/science/article/pii/S030439751930725X> (visited on 03/06/2023).
- [60] E. Fischer and L. Fortnow. “Tolerant Versus Intolerant Testing for Boolean Properties.” In: *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*. IEEE Computer Society, 2005, pp. 135–140. DOI: [10.1109/CCC.2005.30](https://doi.org/10.1109/CCC.2005.30). URL: <https://doi.org/10.1109/CCC.2005.30>.
- [61] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. “Monotonicity testing over general poset domains.” In: *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*. Ed. by J. H. Reif. ACM, 2002, pp. 474–483. DOI: [10.1145/509907.509977](https://doi.org/10.1145/509907.509977). URL: <https://doi.org/10.1145/509907.509977>.
- [62] M. Gaboardi, J. Honaker, G. King, K. Nissim, J. R. Ullman, and S. P. Vadhan. “PSI (Ψ): a Private data Sharing Interface.” *CoRR*, **abs/1609.04340**, 2016. arXiv: [1609.04340](https://arxiv.org/abs/1609.04340). URL: <http://arxiv.org/abs/1609.04340>.
- [63] M. Ghaffari. “An Improved Distributed Algorithm for Maximal Independent Set.” In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. SIAM, 2016, pp. 270–277.
- [64] M. Ghaffari. “Local Computation of Maximal Independent Set.” In: *2022 IEEE 62nd Annual Symposium on Foundations of Computer Science*. 2022.

- [65] M. Ghaffari and J. Uitto. “Sparsifying Distributed Algorithms with Ramifications in Massively Parallel Computation and Centralized Local Computation.” In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. SIAM, 2019, pp. 1636–1653.
- [66] O. Goldreich, S. Goldwasser, and D. Ron. “Property Testing and its Connection to Learning and Approximation.” *J. ACM*, **45**(4), 1998, pp. 653–750.
- [67] S. Goldwasser, M. P. Kim, V. Vaikuntanathan, and O. Zamir. “Planting undetectable backdoors in machine learning models.” In: *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2022, pp. 931–942.
- [68] A. Gollakota, A. R. Klivans, and P. K. Kothari. “A moment-matching approach to testable learning and a new characterization of rademacher complexity.” In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 1657–1670.
- [69] I. J. Goodfellow, J. Shlens, and C. Szegedy. “Explaining and Harnessing Adversarial Examples.” *CoRR*, **abs/1412.6572**, 2014. URL: <https://api.semanticscholar.org/CorpusID:6706414>.
- [70] M. Göös, J. Hirvonen, R. Levi, M. Medina, and J. Suomela. “Non-Local Probes Do Not Help with Graph Problems.” *CoRR*, **abs/1512.05411**, 2015.
- [71] P. Gourdeau, V. Kanade, M. Kwiatkowska, and J. Worrell. “On the hardness of robust classification.” *Journal of Machine Learning Research*, **22**(273), 2021, pp. 1–29.
- [72] J. Grebík and V. Rozhon. “Classification of Local Problems on Paths from the Perspective of Descriptive Combinatorics.” *CoRR*, **abs/2103.14112**, 2021.
- [73] O. Guédon and E. Milman. *Interpolating Thin-Shell and Sharp Large-Deviation Estimates For Isotropic Log-Concave Measures*. 2011. arXiv: [1011.0943](https://arxiv.org/abs/1011.0943) [math.FA]. URL: <https://arxiv.org/abs/1011.0943>.
- [74] A. Haeberlen, B. C. Pierce, and A. Narayan. “Differential Privacy Under Fire.” In: *20th USENIX Security Symposium (USENIX Security 11)*. San Francisco, CA: USENIX Association, Aug. 2011. URL: <https://www.usenix.org/conference/usenix-security-11/differential-privacy-under-fire>.
- [75] J. C. Jackson, H. K. Lee, R. A. Servedio, and A. Wan. “Learning random monotone DNF.” *Discret. Appl. Math.*, **159**(5), 2011, pp. 259–271. DOI: [10.1016/j.dam.2010.08.022](https://doi.org/10.1016/j.dam.2010.08.022). URL: <https://doi.org/10.1016/j.dam.2010.08.022>.
- [76] M. Jha and S. Raskhodnikova. “Testing and Reconstruction of Lipschitz Functions with Applications to Data Privacy.” *SIAM Journal on Computing*, **42**(2), 2013, pp. 700–731. DOI: [10.1137/110840741](https://doi.org/10.1137/110840741).
- [77] A. T. Kalai, A. R. Klivans, Y. Mansour, and R. A. Servedio. “Agnostically learning halfspaces.” *SIAM Journal on Computing*, **37**(6), 2008, pp. 1777–1805.

- [78] I. Kalemaj, S. Raskhodnikova, and N. Varma. “Sublinear-Time Computation in the Presence of Online Erasures.” *Theory of Computing*, **19**(1), 2023, pp. 1–48. DOI: [10.4086/toc.2023.v019a001](https://doi.org/10.4086/toc.2023.v019a001). URL: <https://theoryofcomputing.org/articles/v019a001>.
- [79] M. J. Kearns and L. G. Valiant. “Cryptographic Limitations on Learning Boolean Formulae and Finite Automata.” In: *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*. ACM, 1989, pp. 433–444. DOI: [10.1145/73007.73049](https://doi.org/10.1145/73007.73049). URL: <https://doi.org/10.1145/73007.73049>.
- [80] L. G. Khachiyan. “Polynomial algorithms in linear programming.” *USSR Computational Mathematics and Mathematical Physics*, **20**(1), 1980, pp. 53–72.
- [81] S. Khot, D. Minzer, and M. Safra. “On Monotonicity Testing and Boolean Isoperimetric Type Theorems.” In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. ISSN: 0272-5428. Oct. 2015. DOI: [10.1109/FOCS.2015.13](https://doi.org/10.1109/FOCS.2015.13).
- [82] Z. Kolter and A. Madry. “Adversarial robustness — theory and practice.” 2018. URL: <https://adversarial-ml-tutorial.org/introduction/>.
- [83] J. Lange, E. Linder, S. Raskhodnikova, and A. Vasilyan. “Local Lipschitz filters for bounded-range functions with applications to arbitrary real-valued functions.” In: *Proceedings of the 36th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2025, pp. 2881–2907. DOI: [10.1137/1.9781611978322.93](https://doi.org/10.1137/1.9781611978322.93).
- [84] J. Lange, R. Rubinfeld, and A. Vasilyan. “Properly learning monotone functions via local computation algorithms.” In: *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 1010–1021.
- [85] J. Lange and A. Vasilyan. “Agnostic proper learning of monotone functions: beyond the black-box correction barrier.” *SIAM Journal on Computing*, **0**(FOCS23-1), 2025.
- [86] J. Lange and A. Vasilyan. “Robust learning of halfspaces under log-concave marginals.” *Advances in Neural Information Processing Systems*, **38**, 2026, pp. 100700–100744.
- [87] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana. “Certified Robustness to Adversarial Examples with Differential Privacy.” In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 656–672. DOI: [10.1109/SP.2019.00044](https://doi.org/10.1109/SP.2019.00044).
- [88] Y. T. Lee, A. Sidford, and S. C.-w. Wong. “A faster cutting plane method and its implications for combinatorial and convex optimization.” In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 1049–1065.
- [89] R. Levi, D. Ron, and R. Rubinfeld. “Local Algorithms for Sparse Spanning Graphs.” *Algorithmica*, **82**(4), 2020, pp. 747–786.
- [90] R. Levi, R. Rubinfeld, and A. Yodpinyanee. “Local Computation Algorithms for Graphs of Non-constant Degrees.” *Algorithmica*, **77**(4), 2017, pp. 971–994. DOI: [10.1007/s00453-016-0126-y](https://doi.org/10.1007/s00453-016-0126-y). URL: <https://doi.org/10.1007/s00453-016-0126-y>.
- [91] B. Li, C. Chen, W. Wang, and L. Carin. “Second-order adversarial attack and certifiable robustness.”

- [92] C. McDiarmid. “On the method of bounded differences.” In: *Surveys in Combinatorics*. London Math. Soc. Lecture Notes 141, 1989.
- [93] F. McSherry. “Privacy integrated queries: an extensible platform for privacy-preserving data analysis.” *Commun. ACM*, **53**(9), 2010, pp. 89–97. URL: <http://doi.acm.org/10.1145/1810891.1810916>.
- [94] O. Montasser, S. Goel, I. Diakonikolas, and N. Srebro. “Efficiently learning adversarially robust halfspaces with noise.” In: *International Conference on Machine Learning*. PMLR, 2020, pp. 7010–7021.
- [95] O. Montasser, S. Hanneke, and N. Srebro. “Vc classes are adversarially robustly learnable, but only improperly.” In: *Conference on Learning Theory*. PMLR, 2019, pp. 2512–2530.
- [96] R. O’Donnell and K. Wimmer. “KKL, Kruskal-Katona, and Monotone Nets.” In: *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*. IEEE Computer Society, 2009, pp. 725–734. DOI: [10.1109/FOCS.2009.78](https://doi.org/10.1109/FOCS.2009.78).
- [97] R. K. S. Pallavoor, S. Raskhodnikova, and E. Waingarten. “Approximating the distance to monotonicity of Boolean functions.” *Random Struct. Algorithms*, **60**(2), 2022, pp. 233–260. DOI: [10.1002/rsa.21029](https://doi.org/10.1002/rsa.21029). URL: <https://doi.org/10.1002/rsa.21029>.
- [98] M. Parnas, D. Ron, and R. Rubinfeld. “Tolerant property testing and distance approximation.” *Electron. Colloquium Comput. Complex.*, 2004.
- [99] M. Parter, R. Rubinfeld, A. Vakilian, and A. Yodpinyanee. “Local Computation Algorithms for Spanners.” In: *10th Innovations in Theoretical Computer Science Conference, ITCs 2019, January 10-12, 2019, San Diego, California, USA*. Vol. 124. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 58:1–58:21.
- [100] S. Raskhodnikova and A. D. Smith. “A Note on Adaptivity in Testing Properties of Bounded Degree Graphs.” *Electron. Colloquium Computational Complexity*, **13**(089), 2006. URL: <http://eccc.hpi-web.de/eccc-reports/2006/TR06-089/index.html>.
- [101] O. Reingold and S. Vardi. “New techniques and tighter bounds for local computation algorithms.” *J. Comput. Syst. Sci.*, **82**(7), 2016, pp. 1180–1200. DOI: [10.1016/j.jcss.2016.05.007](https://doi.org/10.1016/j.jcss.2016.05.007). URL: <https://doi.org/10.1016/j.jcss.2016.05.007>.
- [102] I. Roy, S. T. V. Setty, A. Kilzer, V. Shmatikov, and E. Witchel. “Airavat: Security and Privacy for MapReduce.” In: *NSDI*. 2010, pp. 297–312.
- [103] R. Rubinfeld, G. Tamir, S. Vardi, and N. Xie. “Fast, local computation algorithms.” In: *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS)*. 2011, pp. 223–238.
- [104] M. Saks and C. Seshadhri. “Local Monotonicity Reconstruction.” *SIAM Journal on Computing*, **39**(7), 2010, pp. 2897–2926. DOI: [10.1137/080728561](https://doi.org/10.1137/080728561).

- [105] A. Saumard and J. A. Wellner. “Log-concavity and strong log-concavity: a review.” *Statistics surveys*, **8**, 2014, p. 45.
- [106] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein. “Are adversarial examples inevitable?” *arXiv preprint arXiv:1809.02104*, 2018.
- [107] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. “Intriguing properties of neural networks.” In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2014. URL: <http://arxiv.org/abs/1312.6199>.
- [108] M. Tompa and H. Woll. “Random self-reducibility and zero knowledge interactive proofs of possession of information.” In: *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (FOCS)*. 1987, pp. 472–482.
- [109] A. v. d. Vaart and J. A. Wellner. “Weak convergence and empirical processes with applications to statistics.” *Journal of the Royal Statistical Society-Series A Statistics in Society*, **160**(3), 1997, pp. 596–608.
- [110] A. Van Der Vaart and J. A. Wellner. “A note on bounds for VC dimensions.” *Institute of Mathematical Statistics collections*, **5**, 2009, p. 103.
- [111] L. Yang, A. Blum, and J. Carbonell. “Learnability of DNF with Representation-Specific Queries.” In: *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*. ITCS ’13. Berkeley, California, USA: Association for Computing Machinery, 2013, pp. 37–46. ISBN: 9781450318594. DOI: [10.1145/2422436.2422441](https://doi.org/10.1145/2422436.2422441). URL: <https://doi.org/10.1145/2422436.2422441>.
- [112] A. C. Yao. “Probabilistic Computations: Toward a Unified Measure of Complexity (Extended Abstract).” In: *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*. 1977, pp. 222–227. DOI: [10.1109/SFCS.1977.24](https://doi.org/10.1109/SFCS.1977.24). URL: <https://doi.org/10.1109/SFCS.1977.24>.