2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS) | 979-8-3503-1894-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/FOCS57990.2023.00068

Agnostic proper learning of monotone functions: beyond the black-box correction barrier

Jane Lange
CSAIL
MIT
Cambridge, USA
jlange@mit.edu

Arsen Vasilyan
CSAIL
MIT
Cambridge, USA
vasilyan@mit.edu

***Abstract*—We give the first agnostic, efficient, proper learning algorithm for monotone Boolean functions. Given $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$ uniformly random examples of an unknown function $f : \{\pm 1\}^n \to \{\pm 1\}$, our algorithm outputs a hypothesis $g : \{\pm 1\}^n \to \{\pm 1\}$ that is monotone and $(\mathrm{opt} + \varepsilon)$-close to f, where opt is the distance from f to the closest monotone function. The running time of the algorithm (and consequently the size and evaluation time of the hypothesis) is also $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$, nearly matching the lower bound of [13]. We also give an algorithm for estimating up to additive error ε the distance of an unknown function f to monotone using a run-time of $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$. Previously, for both of these problems, sample-efficient algorithms were known, but these algorithms were not run-time efficient. Our work thus closes this gap in our knowledge between the run-time and sample complexity.**

This work builds upon the improper learning algorithm of [17] and the proper semiagnostic learning algorithm of [40], which obtains a non-monotone Boolean-valued hypothesis, then "corrects" it to monotone using query-efficient local computation algorithms on graphs. This black-box correction approach can achieve no error better than $2\mathrm{opt} + \varepsilon$ information-theoretically; we bypass this barrier by

a) augmenting the improper learner with a convex optimization step, and
b) learning and correcting a real-valued function before rounding its values to Boolean.

Our real-valued correction algorithm solves the "poset sorting" problem of [40] for functions over general posets with non-Boolean labels.

***Index Terms*—learning theory, monotone functions, property testing, sublinear algorithms, Boolean functions**

Jane is supported in part by NSF Graduate Research Fellowship under Grant No. 2141064 and NSF Awards CCF-2006664, DMS-2022448. Arsen is supported in part by NSF awards CCF-2006664, DMS-2022448, CCF-1565235, CCF-1955217, Big George Fellowship and Fintech@CSAIL.

I. Introduction

The class of monotone functions over $\{\pm 1\}^n$ is an object of major interest in theoretical computer science. In consequence, the study of learning and testing algorithms for monotone functions [1], [4], [8], [15], [17], [19], [20], [22]–[25], [32], [37], [39], [40], [45], [46] and various subclasses of monotone functions [5], [14], [35], [52] is a major research direction. In this work, 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 $\{\boldsymbol{x}_i\}$ labeled by an arbitrary function $f : \{\pm 1\} \to \{\pm 1\}$ and we are required to perform the following tasks:

1) **Estimating distance to monotonicity** is the task of estimating up to some additive error ε the distance $\mathrm{dist}(f, f_{\mathrm{mon}})$ 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 $\mathrm{dist}(f, g_{\mathrm{mon}})$ approximates $\mathrm{dist}(f, f_{\mathrm{mon}})$ up to additive error ε.

Prior to our work, it was known that information-theoretically these tasks can be solved using only $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$ samples. However, all known algorithms had a run-time of $2^{\Omega(n)}$, thus dramatically exceeding the known sample complexity of $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$. In this work, we close this gap in our knowledge and give algorithms for the two tasks above that not only use $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$ samples, but also run in time $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$. This nearly matches the $2^{\tilde{\Omega}(\sqrt{n})}$ lower bound of [13].

979-8-3503-1894-4/23/$31.00 ©2023 IEEE
DOI 10.1109/FOCS57990.2023.00068

A. Previous work

We note that the work of [40] largely concerns itself with the problem of *realizable learning* of monotone functions, i.e. learning a function f that is itself promised to be monotone. In contrast, the focus of our work is the harder setting when the function f we access is *arbitrary* and we want to obtain a description of a monotone function g_{mon} that predicts f best among monotone functions (up to an additive slack of ε).

Still, as noted in [40], their work does give mixed additive-multiplicative approximation guarantees in the settings we study here. Specifically, [40] gives algorithms that also run in time $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$ and achieve the following:

1) Obtain a $(3,\varepsilon)$-approximation of $\text{dist}(f, f_{\text{mon}})$. In other words, the estimate is in the interval between $\text{dist}(f, f_{\text{mon}})$ and $3 \cdot \text{dist}(f, f_{\text{mon}}) + \varepsilon$. (We also note that [40] additionally present an algorithm that gives a distance estimate in $[\text{dist}(f, f_{\text{mon}}), 2 \cdot \text{dist}(f, f_{\text{mon}}) + \varepsilon]$ but also requires query access to function f).
2) Obtain a succinct description of a monotone function g_{mon}, whose distance $\text{dist}(f, g_{\text{mon}})$ is a $(3,\varepsilon)$-approximation to $\text{dist}(f, f_{\text{mon}})$. In other words, it is in the interval between $\text{dist}(f, f_{\text{mon}})$ and $3 \cdot \text{dist}(f, f_{\text{mon}}) + \varepsilon$. As it is noted in [40], this yields a fully agnostic learning algorithm only if $\text{dist}(f, f_{\text{mon}}) \leq O(\varepsilon)$.

B. Main results

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

Theorem 1. *[Agnostic proper learning of monotone functions[1]] There is an algorithm that runs in time $2^{\tilde{O}\left(\frac{\sqrt{n}}{\varepsilon}\right)}$ and, given uniform sample access to an unknown function $f : \{\pm 1\}^n \to \{\pm 1\}$, with probability at least $1 - \frac{1}{2^n}$, outputs a succinct representation of a monotone function $g : \{\pm 1\}^n \to \{\pm 1\}$ that is 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 $\{\pm 1\}^n$ on which f and its closest monotone function disagree).*

The corollary below follows immediately by the standard method of [47] that runs the learning algorithm in Theorem 1 and estimates the distance between g and f.

[1]See Appendix B for an extension to functions with randomized labels.

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

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

Our main result, Theorem 1, builds on an algorithm that is also of independent interest. It it a local computation algorithm for solving the "poset sorting problem" as described in [41] for real-valued functions (note that [41] only handled Boolean-valued functions). In other words, the algorithm gives local access to a monotone approximation of a real-valued function that is close to the optimal monotone approximation in ℓ_1 distance. (See Section I-C2 for background on local computation algorithms.)

Theorem 2. *[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 $f : P \to [-1, 1]$ be α-close to monotone in ℓ_1 distance. There is an LCA that makes queries to f and outputs queries to $g : P \to [-1, 1]$, 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 poly$(\Delta \log N)$, and succeeds with probability $1 - N^{-10}$.*

C. Our techniques: beyond the black-box correction barrier.

The algorithms of [40] follow the following pattern (which we also summarize in Figure 1):

1) Use [17], [28], [36] to obtain a succinct description of a (possibly non-monotone) function f_{improper} whose distance $\text{dist}(f, f_{\text{improper}})$ is at most $\text{dist}(f, f_{\text{mon}}) + \varepsilon$. The issue now is that f_{improper} is not necessarily monotone, and therefore the distance $\text{dist}(f, f_{\text{improper}})$ might dramatically underestimate the true distance to monotonicity $\text{dist}(f, f_{\text{mon}})$.
2) Design and use a monotonicity corrector, in order to transform the succinct description of f_{improper} into a succinct description of some **monotone** function g_{mon} that is close to f_{improper}. Formally, [40] develop a corrector that guarantees that the distance $\text{dist}(f_{\text{improper}}, g_{\text{mon}})$ satisfies

$$\text{dist}(f_{\text{improper}}, g_{\text{mon}}) \leq c \min_{\text{monotone } f'} \text{dist}(f_{\text{improper}}, f') + \varepsilon, \quad (1)$$

where the constant c is 2. They achieve this by a novel use of **Local Computation Algorithms** (LCAs) on graphs.

This way, [40] obtain a succinct polytime-evaluable description of a monotone function g_{mon} for which[2] $\text{dist}(f, g_{\text{mon}}) \leq 3 \cdot \text{dist}(f, f_{\text{mon}}) + \varepsilon$.

However, one can see that even if the correction constant c in Equation (1) were equal to 1 (which is the best it can be) this approach could only yield a guarantee of $\text{dist}(f, g_{\text{mon}}) \leq 2 \cdot \text{dist}(f, f_{\text{mon}}) + \varepsilon$.

1) Description of our approach: We overcome this barrier by using a different approach, summarized in Figure 2. As before, there is an improper learning phase and a correction phase; however in both phases we work with real-valued functions. We have essentially three steps:

1) Find a real-valued polynomial P that is ε-close to some monotone function, $(\text{opt} + \varepsilon)$[3]-close 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.

In contrast to the approach of [40], 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 [17], [28], [36] with graph LCAs and the *ellipsoid method* for convex optimization.

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, as shown in Figure 2, 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 [9], 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.

Step (2) requires another technical contribution, which is an extension of the poset-sorting LCA of [40] to real-valued functions. This extension is crucial for us to achieve the overall agnostic learning guarantee, because in the improper learning phase we obtain a real-valued function that is only close to monotone in ℓ_1 distance.[4] For step (3) we use the rounding procedure of [36] that rounds real-valued functions to $\{\pm 1\}$-valued functions, and we show that this procedure also preserves monotonicity.

2) LCAs and succinct representations of large objects: In this work we employ heavily the concept of a *succinct representation*. The succinct representations we deal with will have size and evaluation time $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$. To be fully specific, we consider succinct representations of two types of objects:

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

A polynomial of degree $O(\sqrt{n})$ is an example of a succinct representation, but another type of representation that makes frequent appearances in this work is a *local computation algorithm*, or LCA [3], [50]. An LCA efficiently computes a function over a large domain. For example, an LCA for an independent set takes as input

[2]Strictly speaking, the properties of the corrector described so far yield only a guarantee of $\text{dist}(f, g_{\text{mon}}) \leq 4 \cdot \text{dist}(f, f_{\text{mon}}) + \varepsilon$. To improve the multiplicative error constant from 4 to 3 the work of [40] uses an additional property of the corrector.

[3]Since opt is unknown, we instead guess values of opt in increments of ε.

[4]One can construct functions that are arbitrarily close to monotone in ℓ_1 norm but a constant fraction of their values needs to be changed for them to become monotone. Because of this, the corrector of [40] was not fit for our correction stage.

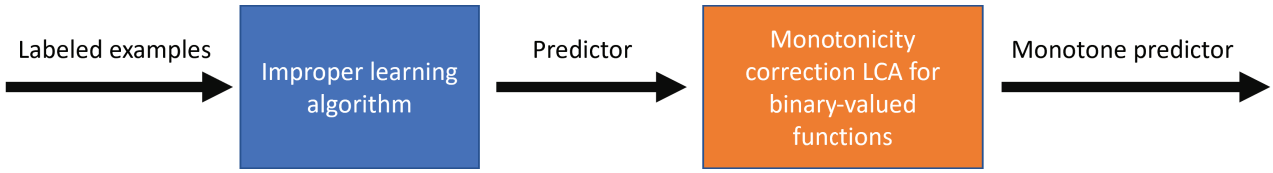

Fig. 1: Control-flow diagram of the semiagnostic algorithm of [40]

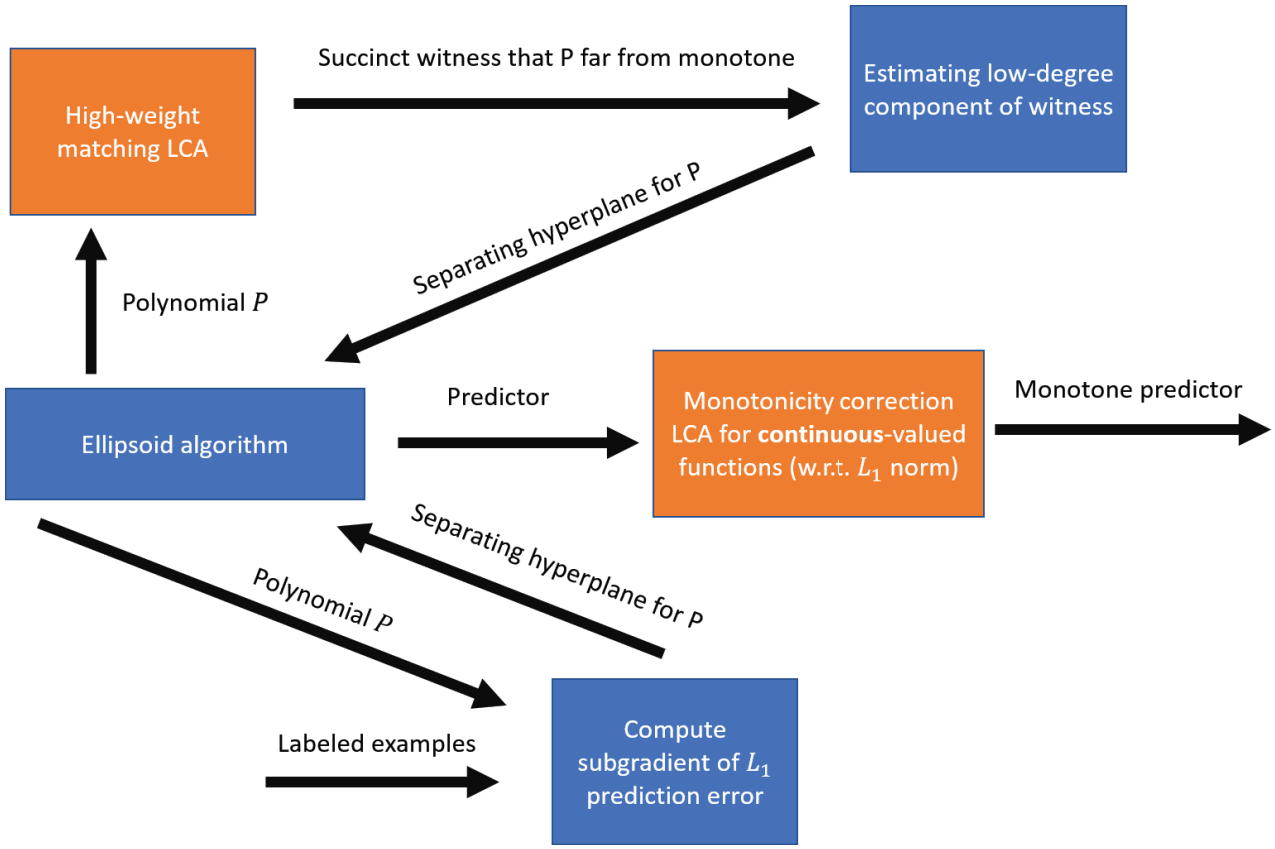

Fig. 2: Control-flow diagram of the fully agnostic learning algorithm presented in this work (the final rounding step is omitted).

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" form an independent set. Typically, its running time and query complexity are each sublinear in the domain size. 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 graph, as a succinct representation of the object it computes. We formalize this relationship in Section II-D.

D. Other related work

The local correction of monotonicity was studied in [2], [7], [10], [51] and [40] (see [40] for an overview of previously available algorithms for monotonicity correction and lower bounds).

The work of [18] gives an improper learning algorithm for a function class that is larger than monotone functions. Additionally, we note that testing of monotone functions has also been studied over hypergrids [9], [11], [12], [19].

In addition to [30], there have been many exciting recent works on local computation algorithms (LCAs). Some examples include [50], [3], [44], [34], [49], [27],

[29], [21], [26] , [48], [31], [43], [6], [16] and [33].

II. Preliminaries

A. Posets and $\{-1,1\}^n$

Let P be a partially-ordered set. 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 $x \neq y$, and use the analogous symbols $\succeq$ and $\succ$ for successorship. If $x \prec y$ and there is no z in P for which $x \prec z \prec y$, then x is an *immediate predecessor* of y and y is an *immediate successor* of x. We refer to the poset P and its Hasse diagram (DAG) interchangeably. The transitive closure $TC(P)$ is the graph on the elements of P that has an edge from each vertex to each of its successors. A *succinct representation* of P with size s is any computational procedure whose description is stored in s bits of memory that takes a vertex as input, outputs the sets of immediate predecessors and immediate successors, and runs in time $O(s)$ in the worst case over vertices.

Specific posets of interest in this work are the Boolean cube and the weight-truncated cube. We give a definition and a size-$O(n/\varepsilon)$ representation computing the truncated cube.[5]

Definition 1. *The n-*dimensional Boolean hypercube *is the set $\{-1,1\}^n$. For $x, y \in \{-1,1\}^n$, we say $x \preceq y$ if for all $i \in \{1, \cdots, n\}$ one has $x_i \leq y_i$. It is immediate that $\{-1,1\}^n$ is a poset with 2^n elements.*

We also define the truncated hypercube

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

Via Hoeffding's bound, we have that the fraction of elements in $\{0,1\}^n$ that are not also in H^ε_n is at most $2\exp\left(-\frac{2t^2}{4n}\right) = \varepsilon$.

Algorithm 1 LCA: TruncatedCube(x, ε)

Given: Input $x \in \{-1,1\}^n$, truncation parameter ε
return $\{y \mid y$ differs from x in one bit and $|\sum_j y_j| \leq \sqrt{2n \log \frac{2}{\varepsilon}}\}$

1) Fourier analysis over $\{\pm 1\}^n$.: Let $[n]$ denote the set $\{1, 2, \cdots, n\}$. We define for every $S \subseteq [n]$ the function $\chi_S : \{\pm 1\}^n \to \mathbb{R}$ as $\chi_S(\boldsymbol{x}) := \prod_{i \in S} x_i$. We define the inner product between two functions $g_1, g_2 : \{\pm 1\}^n \to \mathbb{R}$ as follows: $\langle g_1, g_2 \rangle := \mathbb{E}_{\boldsymbol{x} \sim \{\pm 1\}^n}[g_1(\boldsymbol{x}) g_2(\boldsymbol{x})]$. It is known that $\langle \chi_{S_1}, \chi_{S_2} \rangle = \mathbb{1}_{S_1 = S_2}$. For a function $g : \{\pm 1\}^n \to \mathbb{R}$ we denote $\widehat{g}(S) := \langle g, \chi_S \rangle$. It is known that

$$g(\boldsymbol{x}) = \sum_{S \subseteq [n]} \widehat{g}(S) \chi_S(\boldsymbol{x}) \quad \langle g_1, g_2 \rangle = \sum_{S \subseteq [n]} \widehat{g_1}(S) \widehat{g_2}(S).$$

B. Monotone functions

Part of our algorithm concerns monotonicity of functions over general posets. For a function $f : P \to \mathbb{R}$, we say that a pair of elements $x, y \in P$ forms a *violated pair* if we have $x \preceq y$ but $f(x) > f(y)$, and we define the *violation score* $\mathrm{vs}(x,y) := f(x) - f(y)$. The *violation graph* $\mathrm{viol}(f)$ is the subgraph of $TC(P)$ induced by violated pairs in f. The weight of an edge is the difference $f(x) - f(y)$.

The ℓ_1 distance of f to monotonicity $\mathrm{dist}(f, \mathrm{mono})$ is the ℓ_1 distance of f to the closest real-valued monotone function.

Definition 2 (Distance to monotonicity)**.** *The ℓ_1 distance of $f : P \to \mathbb{R}$ to monotonicity is its distance to the closest real-valued monotone function.*

$$\mathrm{dist}_1(f, \mathrm{mono}) := \min_{\text{monotone } g: P \to \mathbb{R}} \left[\frac{1}{|P|} \sum_{x \in P} |f(x) - g(x)| \right]$$

The Hamming distance to monotonicity of $f : P \to \{-1,1\}$ is defined analogously.

$$\mathrm{dist}_0(f, \mathrm{mono}) := \min_{\substack{\text{monotone } g: \\ P \to \{-1,1\}}} \left[\frac{1}{|P|} \sum_{x \in P} \mathbb{1}[f(x) \neq g(x)] \right]$$

We will need a bound on how well monotone functions can be approximated by low-degree polynomials. The following fact follows[6] from [17], [36] and a refinement by [28].

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

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

C. Convex optimization

The following notion is standard in convex optimization.

Definition 3. *A **separation oracle** for a convex set $\mathcal{C}_{convex}$ is an oracle that given a point $\boldsymbol{x}$ does one of the following things:*

[5] See Algorithm 1 for the computational procedure that provides access to immediate successors and predecessor of a given element. Note that only size $O(n/\varepsilon)$ is necessary because one can, for example, store a circuit that implements Algorithm 1.

[6] see [41] for more explanation on how these references yield the fact below.

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

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

Fact II.2. *[38] 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 $\mathcal{C}_{convex} \subset \{x \in \mathbb{R}^d : \|x\| \leq R\}$. The algorithm runs in time $poly\left(d, \log \frac{R}{r}\right)$ and either outputs an element in $\mathcal{C}_{convex}$ or outputs FAIL. Furthermore, if $\mathcal{C}_{convex}$ contains a ball of radius r, the algorithm is guaranteed to succeed.*

Also see [42] for an overview of algorithms building on [38].

D. LCAs and succinct representations

We use the following LCAs in this work:

Theorem 3 (LCA for maximal matching[7] [30])**.** *There is an algorithm GhaffariMatching that takes adjacency lists access to a graph G, with N vertices and largest degree at most Δ, a random string $r \in \{0,1\}^{poly(\Delta, \log(N/\delta))}$, parameter $\delta \in (0,1)$ and a vertex $v \in G$. The algorithm outputs the identity of a vertex $u : (u,v) \in E(G)$ or $\perp$. The algorithm runs in time $poly(\Delta, \log(N/\delta))$ and with probability at least $1-\delta$ over the choice of r the condition of **global consistency holds** i.e. the set of edges $\{(u,v) \in G : GhaffariMatching(G, r, \delta, u) = v\}$ is a maximal matching in the graph G.*

Theorem 4 (LCA for monotonicity correction of Boolean-valued functions [40])**.** *There is an algorithm BooleanCorrector that takes access to a function $f : P \to \{-1,1\}$ and adjacency lists access to a poset P with N vertices, such that each element has at most Δ predecessors and successors and the longest directed path has length h, a random string $r \in \{0,1\}^{poly(\Delta, \log(N/\delta))}$, a parameter $\delta \in (0,1)$ and an element x in P. The algorithm outputs a value in $\{-1,1\}$. The algorithm runs in time $\Delta^{O(\log h)} \cdot polylog(N/\delta)$ and with probability at least $1-\delta$ over the choice of r the condition of **global consistency holds** i.e. the function $g : P \to \{-1,1\}$ defined as $g(x) :=$ BooleanCorrector(P, r, δ, x) is monotone and is such that $\Pr_{x \sim P}[g(x) \neq f(x)] \leq 2 \cdot \mathrm{dist}(f, \mathrm{mono})$.*

An important idea in [40] 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 4 (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 II.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, combining[8] Fact II.3 and Theorem 3 we see immediately that for a graph G, with N vertices and largest degree at most Δ, using the algorithm in Theorem 3 we can transform a size-s representation[9] of a function computing all-neighbor access to G into a size-$\left(\Delta^{O(\log h)} \cdot polylog(N/\delta) \cdot s\right)$ representation[10] of a function that determines membership in some maximal matching over G. Note that this transformation itself runs in time $\Delta^{O(\log h)} \cdot polylog(N/\delta) \cdot s$. Analogously, in an exact same fashion it is possible to combine Fact II.3 and Theorem 4.

III. Our algorithms

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 MonotoneLearner makes calls to EllipsoidAlgorithm, where the optimization domain is the $\leq n^{\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil}$-dimensional space of degree-$\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomials over $\mathbb{R}^n$, and constraints

[7] To be fully precise, [30] gives an LCA for the task of maximal independent set. The reduction to maximal matching is standard, see e.g. [40].

[8] A 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.

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

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

given by ORACLE$_{\alpha,n,\varepsilon}$. It also makes calls to HYPERCUBECORRECTOR, which is given in Corollary IV.7.

The subroutine ORACLE takes as input a polynomial and provides the separating hyperplane required by ELLIPSOIDALGORITHM. It makes calls to HYPERCUBEMATCHING (see Lemma V.4), which provides a high-weight matching over the pairs of labels that violate monotonicity.

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 Theorem 3. More implementation details and analysis are given in Section V.

The following is the core of HYPERCUBECORRECTOR, given as a "global overview" for convenience. Analysis and local implementation are given in Section IV. 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.

IV. ANALYSIS OF THE LOCAL CORRECTOR

In this section, we prove Theorem 2 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 [40] to handle functions with non-Boolean ranges.

Lemma IV.1 (ℓ_1 correction of k-valued functions)**.** *Let P be a poset and $f : P \to [k]$ be α-close to monotone in ℓ_1 distance. There is an LCA that makes queries to f and outputs queries to $g : P \to [k]$, 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 $poly(\Delta \log N)$, and succeeds with probability $1 - N^{-10}$.*

The following lemmas are used in the proof of correctness of our algorithm. Their proofs are deferred to the appendix.

Lemma IV.2 (Equivalence of k-valued and bitwise monotonicity)**.** *Let $f : P \to [k]$ 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.

Lemma IV.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, $||g - m||_1 \leq ||f - m||_1$.*

The corollary follows from repeated application of Lemma IV.3 and the triangle inequality.

Corollary IV.4 (ℓ_1 error preservation)**.** *Let g be obtained from f by a series of swaps of label pairs that violate monotonicity in f. Then $||g - f||_1 \leq 2 \cdot \mathrm{dist}_1(f, \mathrm{mono})$.*

We also require a modification to the LCA claimed in Theorem 4 for correcting Boolean functions. That 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.

Fact IV.5 (Poset sorting algorithm implicit in [40])**.** *Let P be a poset with N vertices such that every element has at most Δ predecessors and successors, and the longest directed path has length h. Let $f : P \to \{-1, 1\}$ be α-close to monotone in Hamming distance. There is an algorithm* BOOLEANCORRECTOR *that gives query access to a permutation π of P such that $f\pi$ is a monotone function and $\Pr_{x \sim P}[f(x) \neq (f\pi)(x)] \leq 2\alpha$. The LCA implementation of* BOOLEANCORRECTOR *uses $(\Delta \log N)^{O(\log h)}$ queries and running time, has a random seed of length $poly(\Delta \log N)$, and succeeds with probability $1 - N^{-11}$.*

Here we present the LCA implementation of Algorithm 5.

Lemma IV.6 (Correctness and query complexity of Algorithm 6)**.** *With probability $1 - i \cdot N^{-11}$ over a random seed r of length $poly(\Delta \log N)$, the algorithm k-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*

Algorithm 2 Algorithm MONOTONELEARNER (n, ε, T)

Given: Integer n, $\varepsilon \in (0,1)$, and uniform sample access to an unknown function $f : \{\pm 1\}^n \to \{\pm 1\}$.
Output: Circuit $\mathcal{C} : \{\pm 1\}^n \to \{\pm 1\}$.
for $\alpha \in \{\varepsilon, 2\varepsilon, 3\varepsilon, \cdots 1-\varepsilon, 1+200\varepsilon\}$ **do**
  OptimizationResult $\leftarrow$ ELLIPSOIDALGORITHM $\left(1, \varepsilon \cdot n^{-\frac{1}{2}\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil}, \text{ORACLE}_{\alpha,n,\varepsilon}\right)$.
  if OptimizationResult$\neq$FAIL **then**
    $P^{\text{GOOD}} =$ OptimizationResult
    $P^{\text{GOOD}}_{\text{TRIMMED}} \leftarrow$ representation of a function that takes input $\boldsymbol{x}$ and outputs the value
$$\begin{cases} P^{\text{GOOD}}(x) & \text{if } P^{\text{GOOD}}(x) \in [-1,+1] \\ 1 & \text{if } P^{\text{GOOD}}(x) > 1 \\ -1 & \text{if } P^{\text{GOOD}}(x) < -1 \end{cases}$$
    $P^{\text{GOOD}}_{\text{CORRECTED}} \leftarrow$ representation of a function that takes input $\boldsymbol{x}$ and returns the value
$$\text{HYPERCUBECORRECTOR}(x, P^{\text{GOOD}}_{\text{TRIMMED}}, r)$$
    $T \leftarrow \frac{200}{\varepsilon^2} \log\left(\frac{20}{\varepsilon}\right) \log(20n)$ i.i.d. pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$, with $\boldsymbol{x}_i$ sampled uniformly from $\{-1,1\}^n$.
    ThresholdCandidates$\leftarrow \left\{\frac{1}{\varepsilon}\right.$ i.i.d. uniformly random elements in $\left.[-1,1]\right\}$.
    $t^* := \arg\min_{t \in \text{ThresholdCandidates}} \left[\frac{1}{|T|} \sum_{\boldsymbol{x} \in T} \left[\left|\text{sign}(P^{\text{GOOD}}_{\text{CORRECTED}}(\boldsymbol{x}) - t) - f(\boldsymbol{x})\right|\right]\right]$
    return representation of a function that takes input $\boldsymbol{x}$ and returns the value
$$\text{sign}(P^{\text{GOOD}}_{\text{CORRECTED}}(\boldsymbol{x}) - t^*)$$
  end if
end for

$(\Delta \log N)^{O(i \log h + 1)}$, *and* $||g - f||_1 \le 2\alpha$, *where* α *is the* ℓ_1 *distance of* f *to the nearest monotone function.*

Proof. Fix the random seed $\boldsymbol{r}$ and assume all calls to BOOLEANCORRECTOR succeed with $\boldsymbol{r}$, 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-CORRECTOR$(y, P, f, i-1, r_1 \circ \ldots \circ r_{i-1})$ 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-CORRECTOR is called with iteration number i, the function f'_j is monotone over P'_j for all $j < i$. 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-CORRECTOR satisfies the conditions of Lemma IV.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 $\le i \cdot N^{-11}$ as desired. The fact that $||g - f||_1 \le 2\alpha$ follows from Corollary IV.4. □

We can now prove Theorem 2.

Theorem 2. *[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* $f : P \to [-1,1]$ *be* α*-close to monotone in* ℓ_1 *distance. There is an LCA that makes queries to* f *and outputs queries to* $g : P \to [-1,1]$*, such that* g *is monotone and* $||f - g||_1 \le 2\alpha + 3\varepsilon$*. The LCA makes* $(\Delta \log N)^{O(\log h \log(1/\varepsilon))}$ *queries, uses a random seed of length* $poly(\Delta \log N)$*, and succeeds with probability* $1 - N^{-10}$.

Proof of Theorem 2. 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-CORRECTOR$(x, P, f_\varepsilon, \lceil \log(2/\varepsilon) \rceil, r)$ with a random seed $\boldsymbol{r}$ of length $poly(\Delta \log N)$. By Lemma IV.6, 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 $||g_\varepsilon - f_\varepsilon||_1 \le 2 \cdot \text{dist}_1(f_\varepsilon, \text{mono})$. Since f is α-close to some monotone function m, we have $\text{dist}_1(f_\varepsilon, \text{mono}) \le ||f_\varepsilon - m/\varepsilon||_1 \le ||f/\varepsilon - m/\varepsilon||_1 + ||f/\varepsilon - f_\varepsilon||_1 \le \alpha/\varepsilon + 1$.

Algorithm 3 Subroutine $\textsc{Oracle}_{\alpha,n,\varepsilon}(P)$

Given: $\varepsilon, \alpha \in (0,1)$, degree-$\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial P over $\mathbb{R}^n$ with $\|P\|_2 \le 1$, and uniform sample access to an unknown function $f : \{\pm 1\}^n \to \{\pm 1\}$.

Output: "Yes" or ("No", $Q_{\text{separator}}$), where $Q_{\text{separator}}$ is a degree-$\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial over $\mathbb{R}^n$.

$P_{\text{TRIMMED}} \leftarrow$ representation of a function that takes input $\boldsymbol{x}$ and outputs $\begin{cases} P(x) & \text{if } P(x) \in [-1,+1] \\ 1 & \text{if } P(x) > 1 \\ -1 & \text{if } P(x) < -1 \end{cases}$.

$T \leftarrow$ set of $n^{\frac{C\sqrt{n}}{\varepsilon} \log \frac{1}{\varepsilon}}$ i.i.d. pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$, with $\boldsymbol{x}_i$ sampled uniformly from $\{-1,1\}^n$ (for sufficiently large constant C).

$r \leftarrow$ string of $2^{C\sqrt{n}(\log n \cdot \log \frac{1}{\varepsilon})^C}$ random i.i.d. bits (for sufficiently large constant C).

$M_{\text{separator}} \leftarrow$ representation of a function that takes input x and outputs

$$\begin{cases} 0 & \text{if } \textsc{HypercubeMatching}(P_{\text{TRIMMED}}, \varepsilon/4, r) \text{ does not match } x \text{ to any other vertex} \\ 1 & \text{if } \textsc{HypercubeMatching}(P_{\text{TRIMMED}}, \varepsilon/4, r) \text{ matches } x \text{ to some vertex } z, \text{ s.t. } z \preceq x \\ -1 & \text{if } \textsc{HypercubeMatching}(P_{\text{TRIMMED}}, \varepsilon/4, r) \text{ matches } x \text{ to some vertex } z, \text{ s.t. } z \succeq x \end{cases}$$

if $\frac{1}{|T|} \sum_{\boldsymbol{x} \in T} [M_{\text{separator}}(\boldsymbol{x}) \cdot P_{\text{TRIMMED}}(\boldsymbol{x})] > 5\varepsilon$ **then**

  $Q_{\text{separator}} \leftarrow \sum_{S \subset [n]:\, |S| \le \left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \left(\frac{1}{|T|} \sum_{\boldsymbol{x} \in T} [M_{\text{separator}}(\boldsymbol{x}) \cdot \chi_S(\boldsymbol{x})] \right) \chi_S$

  return ("No", $Q_{\text{separator}}$)

else if $\frac{1}{|T|} \sum_{\boldsymbol{x} \in T} [|f(\boldsymbol{x}) - P(\boldsymbol{x})|] > \alpha + 50\varepsilon$ **then**

  $Q_{\text{separator}} \leftarrow \sum_{S \subset [n] | S| \le \left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \left(\mathbb{E}_{\boldsymbol{x} \sim T} \left[\widehat{P}(S) \chi_S(\boldsymbol{x}) \text{sign}(P(\boldsymbol{x}) - f(\boldsymbol{x})) \right] \right) \chi_S$

  return ("No", $Q_{\text{separator}}$)

else

  return "Yes"

end if

Return $g(x) := \varepsilon \cdot g_\varepsilon(x)$. Then

$$||g - f||_1 = ||\varepsilon g_\varepsilon - f||_1 \le ||\varepsilon g_\varepsilon - \varepsilon f_\varepsilon||_1 + ||\varepsilon f_\varepsilon - f||_1 \le \\ \le 2\varepsilon(\alpha/\varepsilon + 1) + \varepsilon \le 2\alpha + 3\varepsilon.$$

The failure probability is $N^{-11} \cdot \lceil \log(2/\varepsilon) \rceil$ by Lemma IV.6, 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}. □

Corollary IV.7 (Monotonizing a representation of a function on the Boolean cube)**.** *Let $f : \{-1,1\}^n \to [-1,1]$ 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 $2^{\tilde{O}(\sqrt{n}\log^{3/2}(1/\varepsilon))} \cdot s_f$ time and outputs a monotone function g such that $||f - g||_1 \le 2\alpha + 4\varepsilon$. The size of the representation of g is $2^{\tilde{O}(\sqrt{n}\log^{3/2}(1/\varepsilon))} \cdot s_f$. The algorithm uses a random seed of length $2^{\tilde{O}(\sqrt{n}\log(1/\varepsilon))}$ and succeeds with probability $1 - 2^{-10n}$.*

The proof of Corollary IV.7 is deferred to Appendix E.

V. 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 V.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 [9]; see Appendix F. □

Algorithm 4 MATCHVIOLATIONS$(P, f, \varepsilon, \boldsymbol{r} = r_1 \circ \ldots \circ r_{\lceil \log 2/\varepsilon \rceil})$

Given: Poset P and function $f : P \to [-1, 1]$ given as succinct representations, weight threshold ε, random seed $\boldsymbol{r} = r_1 \circ \ldots \circ r_{\lceil \log 2/\varepsilon \rceil}$
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$(P', r_i, x))$
    $t \leftarrow t/2$
    $i \leftarrow i + 1$
  end while
end if
return M

Algorithm 5 Global view of sorting k-valued labels in a poset

Given: Poset P of height h, function $f : P \to [k]$
Output: monotone function $g : P \to [k]$
Let $i \leftarrow 0$
for $0 \leq i \leq \lceil \log k \rceil$ **do**
  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.
  Let P_i be the poset on the elements of P with the relation

$$x \prec_{P_i} y := x \prec_P y \text{ and } f_j(x) = f_j(y) \text{ for all } j < i.$$

  Let $\pi_i \leftarrow$ BOOLEANCORRECTOR(f_i, P_i).
  Let $f \leftarrow f\pi_i$.
end for
return f

Algorithm 6 LCA implementation of Algorithm 5, k-CORRECTOR$(x, P, f, i, \boldsymbol{r})$

Given: Target vertex x, all-neighbors (immediate predecessor and successor) oracle for P, query access to $f : P \to [k]$, iteration number i, random seed $\boldsymbol{r} = r_1 \circ \ldots \circ r_i$.
Output: query access to function $g : P \to [k]$ which is monotone when truncated to the first i most significant bits.
if $i = 0$ **then return** $f(x)$
else
  $S \leftarrow$ the set of all predecessors and successors of x in P
  for $y \in S$ **do**
    Let $f'(y) \leftarrow k$-CORRECTOR$(y, P, f, i - 1, r_1 \circ \ldots \circ r_{i-1})$.
  end for
  Let f'_i be defined as in Algorithm 5, and P'_i be similarly defined with respect to f'_i.
  Remove any y from S such that $f'_i(y) = f'_i(x)$ or y and x are incomparable in P'_i.
  Let $z \leftarrow$ BOOLEANCORRECTOR(x, P'_i, f'_i, r_i)
  return $f'(z)$
end if

A. Details and correctness of MATCHVIOLATIONS

The algorithm MATCHVIOLATIONS given in Section III 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.

Lemma V.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 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; see Appendix F. □

Lemma V.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 MATCHVIOLATIONS(P, f, ε, r) *runs in time $(\Delta \log N)^{O(\log(1/\varepsilon))}(s_P + s_f)$ and outputs a representation of size $(\Delta \log N)^{O(\log(1/\varepsilon))}(s_P + s_f)$.*

Algorithm 7 HYPERCUBECORRECTOR$(f, \varepsilon, \boldsymbol{r})$

Given: function $f : \{-1,1\} \to [-1,1]$ given as succinct representation, additive error parameter $\varepsilon > 0$, random seed $\boldsymbol{r} = r_1 \circ \ldots \circ r_{\lceil \log 1/\varepsilon \rceil}$.
Output: succinct representation of monotone function $g : \{-1,1\} \to [-1,1]$.
$P \leftarrow$ 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

$$\begin{cases} \varepsilon \cdot k\text{-CORRECTOR}(x, P, f', \lceil \log(1/\varepsilon) \rceil, \boldsymbol{r}) & -\sqrt{2n \log 2/\varepsilon} \le |x| \le \sqrt{2n \log 2/\varepsilon} \\ 1 & |x| \ge \sqrt{2n \log 2/\varepsilon} \\ -1 & |x| \le -\sqrt{2n \log 2/\varepsilon} \end{cases}$$

return f''

Algorithm 8 LCA: FILTEREDGES(P, f, t, M, x)

1: **Given:** Poset P, function $f : P \to [-1,1]$, and matching M given as succinct representations, weight threshold t, vertex x
2: **Output:** All neighbors of x in the graph of violation score $\ge t$ and not in M
3: **return**

$$\{y \in P(x) \mid M(y) = \perp \text{ and } [(x < y \text{ and } f(x) \ge f(y) + t) \text{ or } (x > y \text{ and } f(x) \le f(y) - t)]\}$$

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 $O(\Delta N) \le (\Delta \log N)^{O(\log(1/\varepsilon))}$, and the running time of MATCHVIOLATIONS is polynomial in this representation size.

If $\varepsilon \ge 1/N$, 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

$$O(\Delta) \cdot (\Delta \log N)^{O(i)}(s_P + s_f) + O(\log N) + O(\Delta \cdot s_P) \le (\Delta \log N)^{O(i+1)}(s_P + s_f).$$

Then we set M to be the function that applies GHAFFARIMATCHING to P'. GHAFFARIMATCHING has constant overhead and makes $poly(\Delta, \log N)$ queries to P'. Then the new size of M is $poly(\Delta, \log N) \cdot (\Delta \log N)^{O(i)}(s_P + s_f) = (\Delta \log N)^{O(i+1)}(s_P + s_f)$.

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.

□

Lemma V.4. *With a random seed of length* $2^{\tilde{O}(\sqrt{n} \log(1/\varepsilon))}$, *Algorithm 9 outputs a representation of a matching on the weighted violation graph* $\mathrm{viol}(f)$, *of weight at least* $2^n \cdot (\frac{1}{4}\mathrm{dist}_1(f, \mathrm{mono}) - 4\varepsilon)$, *with probability at least* $1 - 2^{-10n}$. *The size of the representation is* $2^{\tilde{O}(\sqrt{n} \log(1/\varepsilon))} \cdot s_f$, *where* s_f *is the size of the representation of* f.

Proof. HYPERCUBEMATCHING calls MATCHVIOLATIONS on the truncated hypercube, which has parameters $N < 2^n$ and $\Delta = 2^{O(\sqrt{n} \log n \log(1/\varepsilon))}$. The size of the representation of TRUNCATEDCUBE is $O(n)$. So by Lemma V.3, the running time and output size of HYPERCUBEMATCHING are $2^{O(\sqrt{n} \log n \log(1/\varepsilon))} \cdot s_f$, and the random seed length is $2^{O(\sqrt{n} \log n \log(1/\varepsilon))}$.

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 $\mathrm{dist}_1(f', \mathrm{mono}) \ge \mathrm{dist}_1(f, \mathrm{mono}) - 2\varepsilon$. By Lemma V.2, the weight of the matching is at least $(1 - \varepsilon) \cdot 2^n(\frac{1}{4}\mathrm{dist}_1(f', \mathrm{mono}) - \varepsilon) \ge (1 - \varepsilon) \cdot 2^n(\frac{1}{4}\mathrm{dist}_1(f, \mathrm{mono}) - 3\varepsilon/2) \ge 2^n(\frac{1}{4}\mathrm{dist}_1(f, \mathrm{mono}) - 4\varepsilon)$.

□

Algorithm 9 HYPERCUBEMATCHING$(f, \varepsilon, \boldsymbol{r})$

Given: Function $f : \{-1,1\}^n \to [-1,1]$ given as succinct representation, weight threshold ε, random seed $\boldsymbol{r} = r_1 \circ \ldots \circ r_{\lceil \log 2/\varepsilon \rceil}$
Output: Succinct representation of a high-weight matching on the violating pairs w.r.t. f
$P \leftarrow$ TRUNCATEDCUBE(n, ε)
$M \leftarrow$ representation of a function that takes x and outputs
$$\begin{cases} \text{MATCHVIOLATIONS}(P, f, \varepsilon, \boldsymbol{r}) & -\sqrt{2n \log 2/\varepsilon} \le |x| \le \sqrt{2n \log 2/\varepsilon} \\ \perp & \text{otherwise} \end{cases}$$
return M

VI. ANALYSIS OF THE AGNOSTIC LEARNING ALGORITHM

By inspecting algorithm MONOTONELEARNER (i.e. Algorithm 2 on page 8), we see immediately that the runtime is $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$. We proceed to argue that the algorithm indeed satisfies the guarantee of Theorem 1. First, we will need the following standard proposition.

Claim VI.1. *For any positive integers n and d, real $\varepsilon, \delta \in (0,1)$, and any function $f : \{\pm 1\}^n \to [-1,1]$, let T be a collection of at least $n^{5d} \cdot \frac{100}{\varepsilon^2} \ln \frac{1}{\varepsilon} \ln \frac{1}{\delta}$ i.i.d. uniformly random elements of $\{\pm 1\}^n$. Then, with probability at least $1 - \delta$*

$$\max_{\substack{\text{degree-}d \text{ polynomial } P \\ \text{with } \|P\|_2 \le 1}} \left| \|f - P\|_1 - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim T} [|f(\boldsymbol{x}) - P(\boldsymbol{x})|] \right| \le \varepsilon,$$

Proof. See Appendix G for the proof of this proposition. □

Now, in the following lemma we prove that subroutine $\textbf{Oracle}_{\alpha,n,\varepsilon}(P)$ (i.e. Algorithm 3 on page 9) satisfies some precise specifications with high probability. Informally, we show that $\textbf{Oracle}_{\alpha,n,\varepsilon}(P)$ 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 VI.2. *For sufficiently large constant C in Section III and Section III of procedure $\textbf{Oracle}_{\alpha,n,\varepsilon}(P)$, sufficiently large integer n, any function $f : \{-1,1\}^n \to \{-1,1\}$, parameters $\varepsilon, \alpha \in (0,1)$, and a degree-$\left\lceil \frac{4 \cdot \sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial P satisfying $\|P\|_2 \le 1$ the following is true. The procedure $\textbf{Oracle}_{\alpha,n,\varepsilon}(P)$ runs in time $n^{\tilde{O}\left(\frac{\sqrt{n}}{\varepsilon}\right)}$ and will with probability at least $1 - \frac{1}{2^{5n}}$ conform to the following specification:*

1) If $\textbf{Oracle}_{\alpha,n,\varepsilon}(P)$ outputs "yes", then:

a) The function $P_{\text{TRIMMED}} = \begin{cases} 1 & \text{if } P(x) > 1, \\ -1 & \text{if } P(x) < -1, \\ P(\boldsymbol{x}) & \text{otherwise.} \end{cases}$ 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 $\textbf{Oracle}_{\alpha,n,\varepsilon}(P)$ instead outputs ("No", $Q_{\text{separator}}$), where $Q_{\text{separator}}$ is a degree-$\left\lceil \frac{4 \cdot \sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial over $\mathbb{R}^n$, then we have $\langle P', Q_{\text{separator}} \rangle < \langle P, Q_{\text{separator}} \rangle$ for any degree-$\left\lceil \frac{4 \cdot \sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial P' with $\|P'\|_2 \le 1$ that satisfies the following two conditions:

- *P' is ε-close in L_1 distance to some monotone function $f_{\text{monotone}} : \{\pm 1\}^n \to [-1,1]$ 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 $\textbf{Oracle}_{\alpha,n,\varepsilon}(P)$ will say "yes" with probability at least $1 - \frac{1}{2^{10n}}$.

Proof. We use the union bound to conclude that with probability at least $1 - \frac{1}{2^{5n}}$ all the following events hold:

(a) The LCA from Lemma V.4 works as advertised and the weight W of the resulting matching satisfies

$$\frac{W}{2^n} \ge 0.1 \operatorname{dist}_1(P_{\text{TRIMMED}}, \text{mono}) - \varepsilon.$$

Another way to write the same thing is

$$\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle \ge 0.1 \operatorname{dist}_1(P_{\text{TRIMMED}}, \text{mono}) - \varepsilon. \tag{2}$$

From Lemma V.4 it follows that this holds with probability at least $1 - \frac{1}{2^{10n}}$.

(b) The estimate of $\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle$ in Section III is indeed ε-close to the true value. From the

standard Hoeffding bound, this holds with probability at least $1 - \frac{1}{2^{10n}}$.

(c) It is the case that

$$\left\| \sum_{\substack{S \subset [n]: \\ |S| \leq \left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil}} \widehat{M_{\text{separator}}}(S) \chi_S - Q_{\text{separator}} \right\|_2 \leq \varepsilon$$

Substituting the expression for $Q_{\text{separator}}$, and using the orthogonality of $\{\chi_S\}$ we see this is equivalent to

$$\underbrace{\sum_{S \subset [n]:\ |S| \leq \left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \left(\widehat{M_{\text{separator}}}(S) - \frac{1}{|T|} \sum_{\boldsymbol{x} \in T} \left[M_{\text{separator}}(\boldsymbol{x}) \cdot \chi_S(\boldsymbol{x}) \right] \right)^2}_{\leq \varepsilon n^{-\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil} \text{ in absolute value w.p. } \geq \frac{1}{2^{10n}} \text{ via Hoeffding's bound}} \leq \varepsilon$$

Overall, the above holds with probability at least $1 - \frac{1}{2^{9n}}$ by taking a Hoeffding bound for each individual summand and taking a union bound over them.

(d) The set $T \subset \{\pm 1\}^n$ is such that

$$\max_{\substack{\text{degree-}\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil \text{ polynomial } P' \text{ over } \{\pm 1\}^n \\ \text{with } \|P'\|_2 \leq 1}} \left| \|f - P'\|_1 - \mathbb{E}_{(x, f(x)) \sim T} \left[|f(\boldsymbol{x}) - P'(\boldsymbol{x})| \right] \right| \leq \varepsilon. \quad (3)$$

It follows from Claim A.2 that this happens with probability at least to $1 - \frac{1}{2^{10n}}$.

Now, we argue that if these conditions indeed hold, then $\textbf{Oracle}_{\alpha,n,\varepsilon}(P)$ will satisfy the specification given.

First, suppose $\textbf{Oracle}_{\alpha,n,\varepsilon}(P)$ answered "yes". Then, since the estimate of $\langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle$ in Section III 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 V.4 works as advertised, this means that

$$\begin{aligned} 6\varepsilon &\geq \langle M_{\text{separator}}, P_{\text{TRIMMED}} \rangle \\ &\geq 0.1 \cdot \text{dist}_1(P_{\text{TRIMMED}}, \text{mono}) - \varepsilon \end{aligned}$$

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 III we have

$$\mathbb{E}_{\boldsymbol{x} \sim T} \left[|f(\boldsymbol{x}) - P(\boldsymbol{x})| \right] \leq \alpha + 50\varepsilon.$$

From Equation (3) it then follows that

$$\begin{aligned} \|f - P\|_1 &\leq \mathbb{E}_{\boldsymbol{x} \sim T} \left[|f(\boldsymbol{x}) - P(\boldsymbol{x})| \right] + \varepsilon \\ &\leq \alpha + 51\varepsilon \leq \alpha + 100\varepsilon, \end{aligned}$$

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 $\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon} \log \frac{4}{\varepsilon} \right\rceil$ polynomial with $\|P'\|_2 \leq 1$ that satisfies the following two conditions[11]:

- P' is ε-close in L_1 distance to some monotone function $f_{\text{monotone}} : \{\pm 1\}^n \to [-1, 1]$ 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 $f_{\text{monotone}} : \{\pm 1\}^n \to [-1, 1]$. Since $M_{\text{separator}}$ is defined to be so for every matched pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ with $\boldsymbol{x}_i \prec \boldsymbol{y}_i$ we have $M_{\text{separator}}(\boldsymbol{x}_i) = 1$ and $M_{\text{separator}}(\boldsymbol{y}_i) = -1$ and is 0 otherwise, and for each such pair $f_{\text{monotone}}(\boldsymbol{x}_i) \leq f_{\text{monotone}}(\boldsymbol{y}_i)$ we have $\langle M_{\text{separator}}, f_{\text{monotone}} \rangle \leq 0$. This allows us to conclude

$$\begin{aligned} 0 &\geq \langle M_{\text{separator}}, f_{\text{monotone}} \rangle \\ &= \langle M_{\text{separator}}, P' \rangle + \langle M_{\text{separator}}, f_{\text{monotone}} - P' \rangle \geq \\ &\qquad \langle M_{\text{separator}}, P' \rangle \\ &\quad - \left(\max_{x \in \{-1,1\}^n} |M_{\text{separator}}(x)| \right) \|f_{\text{monotone}} - P'\|_1 \\ &\qquad \geq \langle M_{\text{separator}}, P' \rangle - \varepsilon, \end{aligned}$$

[11] If no polynomial satisfying these conditions exists, the statement we are seeking to prove holds vacuously.

which means

$$\begin{aligned}\varepsilon &\geq \langle M_{\text{separator}}, P'\rangle \\ &= \left\langle \sum_{\substack{S\subset[n]\\ |S|\leq\left\lceil\frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil}} \widehat{M_{\text{separator}}}(S)\left(\prod_{i\in S}x_i\right), P'\right\rangle \\ &= \langle Q_{\text{separator}}, P'\rangle - \\ &- \left\| Q - \sum_{\substack{S\subset[n]:\\ |S|\leq\left\lceil\frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil}} \widehat{M_{\text{separator}}}(S)\left(\prod_{i\in S}x_i\right)\right\|_2 \|P'\|_2 \\ &\geq \langle Q_{\text{separator}}, P'\rangle - \varepsilon. \quad (4)\end{aligned}$$

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

$$\begin{aligned}4\varepsilon &\leq \overbrace{\langle M_{\text{separator}}, P_{\text{TRIMMED}}\rangle \leq \langle M_{\text{separator}}, P\rangle}^{\text{Trimming the values of a function only decreases weights of violated edges.}} \\ &= \left\langle \sum_{S\subset[n]:\,|S|\leq\left\lceil\frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil} \widehat{M_{\text{separator}}}(S)\left(\prod_{i\in S}x_i\right), P\right\rangle \\ &\leq \langle Q_{\text{separator}}, P\rangle + \\ &\left\| Q - \sum_{\substack{S\subset[n]:\\ |S|\leq\left\lceil\frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil}} \widehat{M_{\text{separator}}}(S)\left(\prod_{i\in S}x_i\right)\right\|_2 \|P\|_2 \\ &\geq \langle Q_{\text{separator}}, P\rangle + \varepsilon. \quad (5)\end{aligned}$$

Combining Equation 5 and Equation 4 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 III. Since P' is $(\alpha+\varepsilon)$-close in L_1 distance to the function f, by Equation (3) we have that

$$\begin{aligned}\alpha+\varepsilon &\leq \|f(\boldsymbol{x}) - P'(\boldsymbol{x})\|_1 \\ &\leq \mathbb{E}_{(\boldsymbol{x},f(\boldsymbol{x}))\sim T}\left[|f(\boldsymbol{x}) - P'(\boldsymbol{x})|\right] - \varepsilon,\end{aligned}$$

which we can rewrite as $\mathbb{E}_{(\boldsymbol{x},f(\boldsymbol{x}))\sim T}\left[|f(\boldsymbol{x}) - P'(\boldsymbol{x})|\right] \leq \alpha + 2\varepsilon$. At the same time, we have $\mathbb{E}_{(\boldsymbol{x},f(\boldsymbol{x}))\sim T}\left[|f(\boldsymbol{x}) - P(\boldsymbol{x})|\right] > \alpha + 50\varepsilon$, which means that

$$\begin{aligned}&\mathop{\mathbb{E}}_{(\boldsymbol{x},f(\boldsymbol{x}))\sim T}\left[|f(\boldsymbol{x}) - P(\boldsymbol{x})|\right] > \\ &\qquad \mathbb{E}_{(\boldsymbol{x},f(\boldsymbol{x}))\sim T}\left[|f(\boldsymbol{x}) - P'(\boldsymbol{x})|\right].\end{aligned}$$

Therefore, as the function mapping a polynomial H to the value $\mathbb{E}_{(\boldsymbol{x},f(\boldsymbol{x}))\sim T}\left[|f(\boldsymbol{x}) - H(\boldsymbol{x})|\right]$ is convex , it has to be the case that[12]

$$\begin{aligned}&\left\langle P' - P, \sum_{\substack{S\subset[n]\ :\\ |S|\leq\left\lceil\frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil}} \left(\mathop{\mathbb{E}}_{\boldsymbol{x}\sim T}\left[\widehat{P}(S)\chi_S(\boldsymbol{x})\cdot\right.\right.\right. \\ &\qquad \left.\left.\left.\cdot\,\text{sign}(P(\boldsymbol{x}) - f(\boldsymbol{x}))\right]\right)\chi_S\right\rangle \\ &= \left\langle P' - P, \nabla_H\left(\mathop{\mathbb{E}}_{(\boldsymbol{x},f(\boldsymbol{x}))\sim T}\left[|H(\boldsymbol{x}) - f(\boldsymbol{x})|\right]\right)\Big|_{H=P}\right\rangle \\ &< 0.\end{aligned}$$

This implies that $\langle Q_{\text{separator}}, P'\rangle \leq \langle Q_{\text{sepatator}}, P\rangle$, which completes the proof. □

A. Finishing the proof of the Main Theorem (Theorem 1).

Recall that earlier by inspecting Algorithm 2 we concluded that this algorithm runs in time $2^{\tilde{O}\left(\frac{\sqrt{n}}{\varepsilon}\right)}$. Here we use Lemma VI.2 to finish the proof of Theorem 1 by showing that with probability at least $1-\frac{1}{2^n}$ the function $\text{sign}(P^{\text{GOOD}}_{\text{TRIMMED}}(\boldsymbol{x}) - t^*)$ is monotone and is 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 $1-\frac{1}{2^{3n}}$ the following events hold:

1) Every time an oracle $\textbf{Oracle}_{\alpha,n,\varepsilon}$ is invoked (for various values of α), its behavior will conform to the specifications in Lemma VI.2.
2) The algorithm HypercubeCorrector from Corollary IV.7 used on line 11 works as advertised, so the function $P^{\text{GOOD}}_{\text{CORRECTED}} : \{\pm 1\} \to [-1,1]$ is monotone and we indeed have

$$\begin{aligned}&\left\|P^{\text{GOOD}}_{\text{CORRECTED}} - P^{\text{GOOD}}_{\text{TRIMMED}}\right\|_1 \\ &\qquad \leq 10\cdot\text{dist}_1(P^{\text{GOOD}}_{\text{TRIMMED}}, \text{mono}) + \varepsilon. \quad (6)\end{aligned}$$

[12]To be fully precise, the expression above is a subgradient of the convex function mapping a polynomial H to $\mathbb{E}_{(\boldsymbol{x},f(\boldsymbol{x}))\sim T}\left[|f(\boldsymbol{x}) - H(\boldsymbol{x})|\right]$.

3) In step (4), the function $\text{sign}(P^{\text{GOOD}}_{\text{CORRECTED}}(\boldsymbol{x}) - t^*)$ satisfies the guarantee from Fact A.1, i.e.

$$\Pr_{\boldsymbol{x}\sim\{\pm1\}^n}\left[\text{sign}(P^{\text{GOOD}}_{\text{CORRECTED}}(\boldsymbol{x}) - t^*) \neq f\right] \leq \frac{1}{2}\left\|P^{\text{GOOD}}_{\text{CORRECTED}} - f\right\|_1 + \varepsilon \quad (7)$$

We argue that each of these events takes place with probability at least $1 - \frac{1}{2^{4n}}$:

- Note that the oracles $\textbf{Oracle}_{\alpha,n,\varepsilon}$ for various values of α are invoked at most $2^{\tilde{O}\left(\frac{\sqrt{n}}{\varepsilon}\right)}$ times. Therefore, Lemma VI.2 tells us that for each of this invocations the algorithm $\textbf{Oracle}_{\alpha,n,\varepsilon}$ conforms to its specification with probability at least $1 - \frac{1}{2^{5n}}$. Via union bound we see that event (1) holds with probability at least[13] $1 - \frac{1}{2^{4n}}$.
- Event (2) holds with probability at least $1 - \frac{1}{2^{4n}}$ via Corollary IV.7.
- Event (3) holds with probability at least $1 - \frac{1}{2^{4n}}$ via Fact A.1

Via union bound, we see that with probability at least $1 - \frac{1}{2^{3n}}$ 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 III for some value $\alpha^* \leq 2\text{opt} + 150\varepsilon$, which we argue as follows: If $\alpha^* > 2\text{opt}+150\varepsilon$, then for some[14] $\alpha \in [2\text{opt}+100\varepsilon, 2\text{opt}+150\varepsilon]$ the ellipsoid algorithm failed to find some polynomial P on which $\textbf{Oracle}_{\alpha,n,\varepsilon}$ returns "Yes". We claim that this is impossible. Indeed, let $\mathcal{C}_{\text{convex}}$ be the set consisting of degree-$\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil$ 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 $f_{\text{monotone}} : \{\pm1\}^n \to [-1,1]$, 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 $f_{\text{monotone}} : \{\pm1\}^n \to [-1,1]$ 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 $\{\pm1\}^n \to [-1,1]$ that are $(\alpha+\varepsilon)$-close in L_1 distance a monotone functions and the other convex set is is the set of all degree-$\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil$ polynomials with with $\|P'\|_2 \leq 1$).
- The set $\mathcal{C}_{\text{convex}}$ contains an L_2 ball of radius at least $\varepsilon \cdot n^{-\frac{1}{2}\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil}$. In other words, in $\mathcal{C}_{\text{convex}}$ there is some degree$\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil$ polynomial P_0 such that any degree-$\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil$ polynomial P' that is ε-close to P_0 in L_2 norm is also in $\mathcal{C}_{\text{convex}}$. Let $f_{\text{monotone, optimal}} : \{\pm1\}^n \to \{\pm1\}$ be the monotone function for which it is the case that $\Pr_{\boldsymbol{x}\sim\{\pm1\}^n}[f_{\text{monotone, optimal}}(\boldsymbol{x}) \neq f(\boldsymbol{x})] = \text{opt}$, and let P_0 be a degree-$\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil$ polynomial that is ε-close to $f_{\text{monotone, optimal}}$ in L_1 norm (such polynomial has to exist by Fact II.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 or radius ε contains an L_2 ball of radius at most $\varepsilon/\sqrt{d}$. Our claim follows, since the space of degree-$\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil$ over $\mathbb{R}^d$ has dimension at most $n^{\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil}$.
- Since the procedure $\text{Oracle}_{\alpha,n,\varepsilon}$ is assumed to satisfy the specifications given in Lemma VI.2 and for this specific value of α it never gave the response "yes", then for every query P to $\text{Oracle}_{\alpha,n,\varepsilon}$, the oracle returned some halfspace that separates P from the convex set $\mathcal{C}_{\text{convex}}$.

From Fact II.2 we know that under these conditions the ellipsoid algorithm will necessarily in time $\text{poly}\left(n^{\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil}, \log(R/r)\right) = n^{O\left(\left\lceil \frac{4\cdot\sqrt{n}}{\varepsilon}\log\frac{4}{\varepsilon}\right\rceil\right)}$ find some polynomial P that is in $\mathcal{C}_{\text{convex}}$. For this particular polynomial, the specifications in Lemma VI.2 require the oracle $\text{Oracle}_{\alpha,n,\varepsilon}$ to give a response "yes", which gives us a contradiction. Thus, the function $P^{\text{GOOD}}_{\text{TRIMMED}}$ will be $O(\varepsilon)$-close to monotone in L_1 norm and will satisfy $\left\|P^{\text{GOOD}}_{\text{TRIMMED}} - f\right\|_1 \leq 2\text{opt} + O(\varepsilon)$. Combining this with Equation (6) yields

$$\left\|P^{\text{GOOD}}_{\text{CORRECTED}} - f\right\|_1 \leq 2\text{opt} + O(\varepsilon) + \left\|P^{\text{GOOD}}_{\text{TRIMMED}} - P^{\text{GOOD}}_{\text{CORRECTED}}\right\|_1 = 2\text{opt} + O(\varepsilon).$$

[13] We assume that ε is such that $2^{0.1n}$ exceeds the number $2^{\tilde{O}\left(\frac{\sqrt{n}}{\varepsilon}\right)}$ of times that $\textbf{Oracle}_{\alpha,n,\varepsilon}$ is invoked (for different values of α. Otherwise, the run-time budget is sufficient to store entire truth-tables of functions over $\{-1,1\}^n$ and statement in Algorithm 7 is achieved by the trivial algorithm that uses a linear program to fit the best montone real-valued function and then rounds it to be $\{-1,1\}$-valued. See Appendix C for further details.

[14] Note that 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 $\alpha = \varepsilon, 2\varepsilon, 3\varepsilon, \cdots 1-\varepsilon, 1+200\varepsilon$.

We know that $\left\|P^{\text{GOOD}}_{\text{TRIMMED}} - P^{\text{GOOD}}_{\text{CORRECTED}}\right\|_1 \leq O(\varepsilon)$ because $P^{\text{GOOD}}_{\text{TRIMMED}}$ is $O(\varepsilon)$-close to monotone by Equation (6). Now, combining the inequality above with Equation 7 gives us

$$\Pr_{\boldsymbol{x}\sim\{\pm1\}^n}\left[\text{sign}(P^{\text{GOOD}}_{\text{CORRECTED}}(\boldsymbol{x}) - t^*) \neq f\right] \leq \frac{1}{2}\left\|P^{\text{GOOD}}_{\text{CORRECTED}} - f\right\|_1 + \varepsilon \leq \text{opt} + O(\varepsilon).$$

Finally, we see that since the function $P^{\text{GOOD}}_{\text{CORRECTED}}\{\pm1\}^n \to [-1,+1]$ is monotone we have that the $\{\pm1\}$-valued function $\text{sign}(P^{\text{GOOD}}_{\text{CORRECTED}}(\boldsymbol{x}) - t^*)$ is also monotone, which finishes our argument.

VII. Acknowledgments

We thank Ronitt Rubinfeld and Mohsen Ghaffari for helpful conversations about local computation algorithms. We additionally thank Ronitt Rubinfeld for useful comments regarding the manuscript and Adam Klivans for a helpful discussion of the algorithm of [28]. Finally, we thank Ephraim Linder for pointing out an inaccuracy in a previous version of this work.

References

[1] Nir Ailon, Bernard Chazelle, C. Seshadhri, and Ding Liu. Estimating the distance to a monotone function. *Random Structures & Algorithms*, 31(3):371–383, 2007. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rsa.20167.

[2] Nir Ailon, Bernard Chazelle, C. Seshadhri, and Ding Liu. Property-Preserving Data Reconstruction. *Algorithmica*, 51(2):160–182, 2008.

[3] Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient Local Computation Algorithms. In *Proceedings of the 2012 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 1132–1139. Society for Industrial and Applied Mathematics, January 2012.

[4] Kazuyuki Amano and Akira Maruoka. On learning monotone Boolean functions under the uniform distribution. *Theor. Comput. Sci.*, 350(1):3–12, 2006.

[5] Dana Angluin. Queries and Concept Learning. *Mach. Learn.*, 2(4):319–342, April 1988. Place: USA Publisher: Kluwer Academic Publishers.

[6] Rubi Arviv and Reut Levi. Improved LCAs for constructing spanners. *CoRR*, abs/2105.04847, 2021.

[7] Pranjal Awasthi, Madhav Jha, Marco Molinaro, and Sofya Raskhodnikova. Limitations of local filters of Lipschitz and monotone functions. *ACM Transactions on Computation Theory*, 7(1), December 2014. Publisher: Association for Computing Machinery (ACM).

[8] Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 1021–1032, 2016.

[9] Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-testing. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 164–173, 2014.

[10] Arnab Bhattacharyya, Elena Grigorescu, Madhav Jha, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Lower bounds for local monotonicity reconstruction from transitive-closure spanners. In *Approximation, Randomization, and Combinatorial Optimization*, pages 448–461, 2010.

[11] Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. A o(d) · 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*, pages 2133–2151. SIAM, 2018.

[12] Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. Domain Reduction for Monotonicity Testing: A o(d) Tester for Boolean Functions in d-Dimensions. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1975–1994, 2020.

[13] Eric Blais, Clément L Canonne, Igor C Oliveira, Rocco A Servedio, and Li-Yang Tan. Learning Circuits with Few Negations. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, page 512, 2015.

[14] Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Properly learning decision trees in almost polynomial time. *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 920–929, 2022.

[15] Avrim Blum, Carl Burch, and John 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*, pages 408–415. IEEE Computer Society, 1998.

[16] Sebastian Brandt, Christoph Grunau, and Václav Rozhon. The randomized local computation complexity of the Lovász local lemma. *CoRR*, abs/2103.16251, 2021.

[17] Nader H Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM (JACM)*, 43(4):747–770, 1996. Publisher: ACM New York, NY, USA.

[18] Clément L. Canonne, Elena Grigorescu, Siyao Guo, Akash Kumar, and Karl Wimmer. Testing k-Monotonicity. *CoRR*, abs/1609.00265, 2016.

[19] Deeparnab 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*, pages 419–428. ACM, 2013.

[20] Deeparnab Chakrabarty and C. Seshadhri. Adaptive Boolean Monotonicity Testing in Total Influence Time. In *Proceedings of Innovations in Theoretical Computer Science (ITCS)*, pages 20:1–20:7, 2019.

[21] Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan 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*, pages 471–480. ACM, 2019.

[22] Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New Algorithms and Lower Bounds for Monotonicity Testing. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, October 2014.

[23] Xi Chen and Erik Waingarten. Testing unateness nearly optimally. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 547–558, 2019.

[24] Xi Chen, Erik Waingarten, and Jinyu 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, June 2017. Association for Computing Machinery.

[25] Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved Testing Algorithms for Monotonicity. In *RANDOM-APPROX'99, Berkeley, CA, USA, August 8-11, 1999, Proceedings*, volume 1671 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 1999.

[26] Guy Even, Reut Levi, Moti Medina, and Adi Rosén. Sublinear Random Access Generators for Preferential Attachment Graphs. *ACM Trans. Algorithms*, 17(4):28:1–28:26, 2021.

[27] Guy Even, Moti Medina, and Dana Ron. Best of Two Local Models: Local Centralized and Local Distributed Algorithms. *CoRR*, abs/1402.3796, 2014. arXiv: 1402.3796.

[28] Vitaly Feldman, Pravesh Kothari, and Jan Vondrák. Tight bounds on l1 approximation and learning of self-bounding functions. *Theoretical Computer Science*, 808:86–98, February 2020.

[29] Mohsen Ghaffari. An Improved Distributed Algorithm for Maximal Independent Set. In *Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 270–277. Society for Industrial and Applied Mathematics, December 2015.

[30] Mohsen Ghaffari. Local Computation of Maximal Independent Set. In *2022 IEEE 62nd Annual Symposium on Foundations of Computer Science*, 2022.

[31] Mohsen Ghaffari and Jara 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*, pages 1636–1653. SIAM, 2019.

[32] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property Testing and its Connection to Learning and Approximation. *J. ACM*, 45(4):653–750, 1998.

[33] Jan Grebík and Václav Rozhon. Classification of Local Problems on Paths from the Perspective of Descriptive Combinatorics. *CoRR*, abs/2103.14112, 2021.

[34] Mika Göös, Juho Hirvonen, Reut Levi, Moti Medina, and Jukka Suomela. Non-Local Probes Do Not Help with Graph Problems. *CoRR*, abs/1512.05411, 2015.

[35] Jeffrey C. Jackson, Homin K. Lee, Rocco A. Servedio, and Andrew Wan. Learning random monotone DNF. *Discret. Appl. Math.*, 159(5):259–271, 2011.

[36] A. T. Kalai, A. R. Klivans, Yishay Mansour, and R. A. Servedio. Agnostically learning halfspaces. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 11–20, October 2005.

[37] Michael J. Kearns and Leslie 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*, pages 433–444. ACM, 1989.

[38] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[39] Subhash Khot, Dor Minzer, and Muli Safra. On Monotonicity Testing and Boolean Isoperimetric Type Theorems. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, October 2015.

[40] Jane Lange, Ronitt Rubinfeld, and Arsen Vasilyan. Properly learning monotone functions via local correction. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 75–86, October 2022. ISSN: 2575-8454.

[41] Jane Lange, Ronitt Rubinfeld, and Arsen Vasilyan. Properly learning monotone functions via local reconstruction, March 2023. arXiv:2204.11894 [cs].

[42] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai 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*, pages 1049–1065. IEEE, 2015.

[43] Reut Levi, Dana Ron, and Ronitt Rubinfeld. Local Algorithms for Sparse Spanning Graphs. *Algorithmica*, 82(4):747–786, 2020.

[44] Reut Levi, Ronitt Rubinfeld, and Anak Yodpinyanee. Local Computation Algorithms for Graphs of Non-constant Degrees. *Algorithmica*, 77(4):971–994, 2017.

[45] Ryan O'Donnell and Karl 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*, pages 725–734. IEEE Computer Society, 2009.

[46] Ramesh Krishnan S Pallavoor, Sofya Raskhodnikova, and Erik Waingarten. Approximating the distance to monotonicity of Boolean functions. *Random Structures & Algorithms*, 60(2):233–260, 2022. Publisher: Wiley Online Library.

[47] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Electron. Colloquium Comput. Complex.*, 2004.

[48] Merav Parter, Ronitt Rubinfeld, Ali Vakilian, and Anak Yodpinyanee. Local Computation Algorithms for Spanners. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 58:1–58:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[49] Omer Reingold and Shai Vardi. New techniques and tighter bounds for local computation algorithms. *J. Comput. Syst. Sci.*, 82(7):1180–1200, 2016.

[50] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast Local Computation Algorithms. In *ICS*, 2011.

[51] Michael Saks and C. Seshadhri. Local Monotonicity Reconstruction. *SIAM J. Comput.*, 39:2897–2926, January 2010.

[52] Liu Yang, Avrim Blum, and Jaime Carbonell. Learnability of DNF with Representation-Specific Queries. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 37–46, New York, NY, USA, 2013. Association for Computing Machinery. event-place: Berkeley, California, USA.

APPENDIX

A. Rounding of real-valued functions to Boolean.

Fact A.1. *Suppose we have two functions $g : \{\pm 1\}^n \to \mathbb{R}$ and $f : \{\pm 1\}^n \to \{\pm 1\}$. 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 $\{-1,1\}^n$, and let 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*

$$t^* := \operatorname*{arg\,min}_{t \in \mathit{ThresholdCandidates}} \frac{1}{|T|}\sum_{\boldsymbol{x}\in T} |\mathit{sign}(g(\boldsymbol{x}) - t) - f(\boldsymbol{x})|$$

Then, with probability at least $1-\delta$ it is the case that

$$\Pr_{\boldsymbol{x}\sim\{\pm1\}^n}\left[\mathit{sign}(g(\boldsymbol{x}) - t^*) \neq f\right] \leq \tfrac{1}{2}\left\|f - g\right\|_1 + \varepsilon$$

Proof. We get that

$$\mathbb{E}_{t\sim[-1,1]}\left[\mathbb{E}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[|\mathrm{sign}(g(\boldsymbol{x}) - t) - f(\boldsymbol{x})|\right]\right] \leq \left\|f - g\right\|_1$$

directly via linearity of expectation. Now, the random variable $\mathbb{E}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[|\mathrm{sign}(g(\boldsymbol{x}) - t) - f(\boldsymbol{x})|\right]$ (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

$$\Pr_{t\sim[-1,1]}\left[\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[|\mathrm{sign}(g(\boldsymbol{x})-t)-g(\boldsymbol{x})|\right]\geq E+\varepsilon/2\right] \leq \frac{E}{E+\varepsilon/2}\leq\frac{2}{2+\varepsilon/2}\leq 1-\frac{\varepsilon}{4}.$$

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 $\mathbb{E}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[|\mathrm{sign}(g(\boldsymbol{x})-t)-g(\boldsymbol{x})|\right]$ 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

$$\max_{t\in\text{ThresholdCandidates}}\left|\frac{1}{|T|}\sum_{\boldsymbol{x}\in T}|\mathrm{sign}(g(\boldsymbol{x})-t)-f(\boldsymbol{x})|-E_{\boldsymbol{x}\sim\{-1,1\}^n}|\mathrm{sign}(g(\boldsymbol{x})-t)-f(\boldsymbol{x})|\right|\leq\frac{\varepsilon}{4}.$$

Overall, we see that with probability at least $1-\delta$ it is the case that

$$\Pr_{\boldsymbol{x}\sim\{\pm1\}^n}[\mathrm{sign}(g(\boldsymbol{x})-t^*)\neq f] \leq \frac{1}{|T|}\sum_{\boldsymbol{x}\in T}|\mathrm{sign}(g(\boldsymbol{x})-t^*)-f(\boldsymbol{x})|+\frac{\varepsilon}{4} \leq \tfrac{1}{2}\|f-g\|_1+\varepsilon$$

This finishes the proof. □

B. 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 1. 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

$$\Pr_{(\boldsymbol{x},\boldsymbol{y})\sim D_{\text{pairs}}}[g(\boldsymbol{x})\neq\boldsymbol{y}] \leq \min_{\substack{\text{monotone } f_{\text{mon}}:\\ \{-1,1\}^n\to\{-1,1\}}}\Pr_{(\boldsymbol{x},\boldsymbol{y})\sim D_{\text{pairs}}}[f_{\text{mon}}(\boldsymbol{x})\neq\boldsymbol{y}]+O(\varepsilon). \tag{8}$$

The only difference between this setting and the one in Theorem 1 is that here the label y doesn't 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 1 extends directly into this slightly more general setting. Formally, we show that

Theorem 5. *For all sufficiently large integers n the following holds. There is an algorithm that runs in time $2^{\tilde{O}\left(\frac{\sqrt{n}}{\varepsilon}\right)}$ 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 $1-\frac{1}{2^{0.5n}}$ the algorithm outputs a representation of a monotone function $g:\{\pm1\}^n\to\{\pm1\}$ of size $2^{\tilde{O}\left(\frac{\sqrt{n}}{\varepsilon}\right)}$ that satisfies Equation (8).*

C. Case 1: ε is very small.

We will consider two cases. First of all, suppose ε is so small that the run-time of the algorithm in Theorem 1 exceeds $2^{0.1n}$. In this case, the following algorithm runs in time $poly(2^n,1/\varepsilon)$ and outputs and efficiently-evaluable succinct representation of a function g for which Equation (8) holds:

1) Draw two sets T_1 and T_2, each of $100n^5\cdot 2^n/\varepsilon^2$ example-label pairs from D_{pairs}.
2) For each $x\in\{-1,1\}^n$ 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-$2^{O(n)}$ linear program, find the monotone function $q:\{-1,1\}^n\to[-1,1]$ that is closest to h is ℓ_1 distance.
4) Output the function g defined so $g(x):=\mathrm{sign}(q(x)-t^*)$, where t^* is obtained as in Fact A.1 using the samples in T_2.

The function g we output above with high probability satisfies Theorem 1 for the following reason. First of all, via the standard coupon-collector argument with probability at least $1-\frac{1}{2^{5n}}$ for every $x\in\{-1,1\}^n$ 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 $1-\frac{1}{2^{2n}}$ we have

$$\left|h(x)-\mathbb{E}_{(\boldsymbol{x}',\boldsymbol{y}')\sim D_{\text{pairs}}}\left[\boldsymbol{y}'\middle|\boldsymbol{x}'=x\right]\right|\leq\frac{\varepsilon}{2}. \tag{9}$$

Now, from steps (3) and (4) we have

$$\frac{\|h-g\|_1}{2}\leq\tfrac{1}{2}\mathrm{dist}_1(h,\text{mono})+\varepsilon. \tag{10}$$

Therefore, we can combine Equation (9) and Equation (10) to obtain

$$\Pr_{(\boldsymbol{x},\boldsymbol{y})\sim D_{\text{pairs}}}[g(\boldsymbol{x})\neq\boldsymbol{y}]\leq \min_{\substack{\text{monotone } f_{\text{mon}}:\\ \{-1,1\}^n\to\{-1,1\}}}\Pr_{(\boldsymbol{x},\boldsymbol{y})\sim D_{\text{pairs}}}[f_{\text{mon}}(\boldsymbol{x})\neq\boldsymbol{y}]+O(\varepsilon), \tag{11}$$

which finishes the proof for this case.

D. 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 1 runs in time at most $2^{0.1n}$ (and therefore uses at most $2^{0.1n}$ samples). In this case, we claim that simply running the algorithm in Theorem 1 will give an efficiently evaluable succinct description of a function g that satisfies the guarantee in Equation (8).

We now proceed to show that the guarantee in Equation (8) will indeed be achieved. Define a random function $f_{\text{random}} : \{-1,1\}^n \to \{-1,1\}$, so for all $x \in \{-1,1\}^n$ the value $f_{\text{random}}(x)$ is chosen independently such that $f_{\text{random}}(x) = 1$ with probability $\Pr_{(\boldsymbol{x}',\boldsymbol{y}')\sim D_{\text{pairs}}}[\boldsymbol{y}' = 1 \mid \boldsymbol{x}' = x]$ and $f_{\text{random}}(x) = -1$ with probability $\Pr_{(\boldsymbol{x}',\boldsymbol{y}')\sim D_{\text{pairs}}}[\boldsymbol{y}' = -1 \mid \boldsymbol{x}' = x]$. Consider the following two scenarios:

- **Scenario I:** The samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$ given to the algorithm from Theorem 1 are indeed i.i.d. samples coming from D_{pairs}.
- **Scenario II:** The samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$ given to the algorithm from Theorem 1 are sampled as follows: (i) $\boldsymbol{x}_i$ are i.i.d. uniform from $\{-1,1\}^n$ (ii) $\boldsymbol{y}_i = f_{\text{random}}(\boldsymbol{x}_i)$.

First we argue that in Scenario II with probability at least $1 - \frac{2}{2^n}$ the function g given by the algorithm from Theorem 1 satisfies Equation (8), (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 (8). From the Hoeffding's bound, it follows that with probability at least[15] $1 - \frac{1}{2^n}$ over the choice of f_{random} it is the case that

$$\left| \Pr_{\boldsymbol{x}\sim\{-1,1\}^n}[f_{\text{random}}(\boldsymbol{x}) \neq f^*_{\text{mon}}(\boldsymbol{x})] - \Pr_{(\boldsymbol{x},\boldsymbol{y})\sim D_{\text{pairs}}}[f^*_{\text{mon}}(\boldsymbol{x}) \neq \boldsymbol{y}] \right| \leq \varepsilon. \quad (12)$$

Now, Theorem 1 implies that with probability at least $1 - \frac{1}{2^n}$

$$\Pr_{\boldsymbol{x}\sim\{-1,1\}^n}[g(\boldsymbol{x}) \neq f_{\text{random}}(\boldsymbol{x})] \leq \text{dist}_0(f_{\text{random}}, \text{mono}) + O(\varepsilon) \leq \Pr_{\boldsymbol{x}\sim\{-1,1\}^n}[f^*_{\text{mon}}(\boldsymbol{x}) \neq f_{\text{random}}(\boldsymbol{x})] + O(\varepsilon). \quad (13)$$

[15]Here we used that $\varepsilon \geq \frac{1}{\sqrt{n}\, poly \log n}$, because otherwise ε would be too small and we would be in the other case when the run-time of the algorithm in Theorem 1 exceeds $2^{0.1n}$. Also, we note that a much stronger bound can be deduced from the Hoeffding bound, but we only need a bound of $1 - \frac{1}{2^n}$.

Combining Equations 12 and 13 we we see that with probability at least $1 - \frac{2}{2^n}$, the function g given by the algorithm from Theorem 1 satisfies Equation (8) in Scenario II.

Finally, we argue that Equation (8) will be satisfied also in Scenario I with probability at least $1 - \frac{1}{2^{0.5n}}$ for sufficiently large n. Conditioned on the absence of sample pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ and $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ with $\boldsymbol{x}_i = \boldsymbol{x}_j$, 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 1 uses at most $2^{0.1n}$ samples. By taking a union bound over all pairs of samples, we bound the probability of such collision by $\frac{2^{0.2n}}{2^n} = 2^{-0.8n}$. Thus, information-theoretically, any algorithm can distinguish between Scenario I and Scenario II with an advantage of only at most $2^{-0.8n}$. In particular, this is true of the algorithm that checks whether Equation (8) applies. Thus, indeed Equation (8) will be satisfied also in Scenario I with probability at least $1 - \frac{2}{2^n} - \frac{1}{2^{0.8n}} \geq 1 - \frac{1}{2^{0.5n}}$, which finishes the proof of Theorem 5.

E. Proofs deferred from Section IV

Proof of Lemma IV.2. 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. □

Proof of Lemma IV.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.

- $f(y) \le m(x) \le m(y) \le f(x)$: then

$$\begin{aligned}&|m(x) - f(x)| + |m(y) - f(y)|\\ =&(f(x) - m(x)) + (m(y) - f(y))\\ \ge&(f(x) - m(y)) + (m(x) - f(y))\\ =&|g(y) - m(y)| + |g(x) - m(x)|.\end{aligned}$$

  The distance of this pair does not increase. The case of $m(x) \le f(y) \le f(x) \le m(y)$ is symmetric.
- $f(y) \le m(x) \le f(x) \le m(y)$: then

$$\begin{aligned}&|m(x) - f(x)| + |m(y) - f(y)|\\ =&(f(x) - m(x)) + (m(y) - f(y))\\ \ge&(m(x) - f(y)) + (m(y) - f(x))\\ =&|g(x) - m(x)| + |g(y) - m(y)|.\end{aligned}$$

  The distance of this pair does not increase. The case of $m(x) \le f(y) \le m(y) \le f(x)$ is symmetric.

□

Proof of Corollary IV.7. Let $f : \{-1,1\}^n \to [-1,1]$ be α-close to monotone in ℓ_1 distance. We call the algorithm HYPERCUBECORRECTOR(f, ε, r) with a random seed r of length $2^{O(\sqrt{n \log(1/\varepsilon)} \log n)}$. First we set the poset to be the truncated cube of width $\sqrt{2n \log 2/\varepsilon}$, which is a poset such that every element has at most $2^{O(\sqrt{n \log(1/\varepsilon)} \log n)}$ predecessors and successors. The representation of this poset (not its transitive closure) has size $poly(n, \log(1/\varepsilon))$. 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 n and $1/\varepsilon$. Since the Δ parameter for the truncated cube is $2^{O(\sqrt{n \log(1/\varepsilon)} \log n)}$, the h parameter is $O(\sqrt{n})$, and the N parameter is $< 2^n$, the worst-case running time and query complexity of this instance of k-CORRECTOR is $2^{O(\sqrt{n} \log n \log^{3/2}(1/\varepsilon))}$ by Lemma IV.6. Thus the representation size of the k-CORRECTOR instance is $2^{\tilde{O}(\sqrt{n} \log^{3/2}(1/\varepsilon))}$, and so the representation size of f'' is $2^{\tilde{O}(\sqrt{n} \log^{3/2}(1/\varepsilon))} \cdot s_f$. With the random seed of length $2^{O(\sqrt{n \log(1/\varepsilon)} \log n)} = poly(\Delta \log N)$, k-CORRECTOR succeeds with probability $N^{-10} \le 2^{-10n}$.

□

F. Proofs deferred from Section V

Proof of Lemma V.1. The proof of $\mathrm{dist}_1(f, \mathrm{mono}) \ge W/N$ is straightforward; for any edge (x, y), $x \prec y$ in the matching, any monotone function must have $g(y) \ge g(x)$ and thus $(f(x) - g(x)) + (g(y) - f(y)) \ge f(x) - f(y)$. 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, presented in [9]. 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 $V_\ge$ and $V_\le$. The duplicated copies of x will be denoted $x_\ge$ and $x_\le$. We define the bipartite graph $B_{f,g}$ to be the graph on $V_\ge \times V_\le$ 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 $\mathrm{viol}(f)$; 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 $|A| \le |N(A)|$.

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 $V_\le$, which makes it a neighbor of x.
b) If $x \in N(A)$, $y \in V_\le$, $x \prec y$, 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 $x \in A$, $y \in V_\le$, $x \prec y$, 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 $(x,y) \in M'$:

$$\begin{aligned}\sum_{(x,y)\in M'} & f(x) - f(y) \\ &= \sum_{(x,y)\in M'} f(x) - g(x) + g(y) - f(y) \\ &= \sum_{x \in V_> \cup V_<} |f(x) - g(x)| = N||f-g||_1.\end{aligned}$$

We will now find a matching M in $\mathrm{viol}(f)$ of equal weight. First replace each $x_\le$ and $x_\ge$ with x, obtaining an edge set in $\mathrm{viol}(f)$ 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 $(y, x_\le)$ and $(x_\ge, z)$, 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 $\mathrm{viol}(f)$ of weight equal to $N||f-g||_1$, which is equal to $N \cdot \mathrm{dist}_1(f, \mathrm{mono})$. □

Proof of Lemma V.2. Fix the random seed r and assume all calls to the algorithm of [30] using r succeed. Let M' be a maximum-weight matching over $\mathrm{viol}(f)$, 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 V-A) 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 $\delta_e \ge w_e/2$ 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 $\le \varepsilon$. We then have

$$\begin{aligned}w(M') &= w(M \cap M') + \sum_{e \in M' \setminus M} w_e \\ &\le w(M \cap M') + \sum_{e \in M' \setminus M} \max(2\delta_e, \varepsilon) \\ &\le w(M \cap M') + 2 \sum_{e \in M' \setminus M} \delta_e + \varepsilon N\end{aligned}$$

We claim that $\sum_{e \in M' \setminus M} \le 2 \cdot w(M \setminus M')$. 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,

$$\begin{aligned}w(M') &\le w(M \cap M') + 4 \sum_{e \in M \setminus M'} w_e + \varepsilon N \\ &\le 4 \cdot w(M) + \varepsilon N\end{aligned}$$

By Lemma V.1, $w(M') = N \cdot \mathrm{dist}_1(f, \mathrm{mono})$; therefore $w(M) \ge N(\frac{1}{4}\mathrm{dist}_1(f, \mathrm{mono}) - \varepsilon)$ as desired.

We now bound the failure probability. When called with a random seed of length $poly(\log N, \log\log(1/\varepsilon))$ the algorithm of [30] 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 2, we may assume that $\log(1/\varepsilon) \le N$, and so the randomness complexity is $poly(\Delta, \log N)$. □

G. Proof of Claim A.2.

Let us first recall the statement of the claim:

Claim A.2. *For any positive integers n and d, real $\varepsilon, \delta \in (0,1)$, and any function $f : \{\pm 1\}^n \to [-1,1]$, let T be a collection of at least $n^{5d} \cdot \frac{100}{\varepsilon^2} \ln \frac{1}{\varepsilon} \ln \frac{1}{\delta}$ i.i.d. uniformly random elements of $\{\pm 1\}^n$. Then, with probability at least $1-\delta$*

$$\max_{\substack{\text{degree-}d\text{ polynomial } P \\ \text{with } \|P\|_2 \le 1}} \left| \|f - P\|_1 - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim T} [|f(\boldsymbol{x}) - P(\boldsymbol{x})|] \right| \le \varepsilon,$$

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 $\max_{\boldsymbol{x}\in\{\pm1\}^n} |P(\boldsymbol{x})| \leq n^d$. This implies, via the Hoeffding bound, that

$$\Pr_{\text{choice of } T}\left[\left|\|f-P\|_1 - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim T}[|f(\boldsymbol{x})-P(\boldsymbol{x})|]\right| > \frac{\varepsilon}{4}\right] \leq \exp\left(-\frac{\varepsilon^2}{32}\frac{|T|}{n^{2d}}\right).$$

We now move on to bounding the maximum over all degree-d polynomials P over $\{\pm1\}^n$ with $\|P\|_2 \leq 1$. We will need a collection $\mathcal{C}$ of degree d polynomials over $\{\pm1\}^n$, such that $|\mathcal{C}| \leq \exp\left(n^d \ln \frac{8n^d}{\varepsilon}\right)$ so for every degree d polynomial P with $\|P\|_2 \leq 1$ there is some element $P_{\text{closest}} \in \mathcal{C}$ for which it is the case that

$$\max_{\boldsymbol{x}\in\{\pm1\}^n} |P(\boldsymbol{x}) - P_{\text{closest}}(\boldsymbol{x})| \leq \frac{\varepsilon}{4}.$$

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 $\sum_{\substack{S\subset[n]\\|S|\leq d}} c_S\left(\chi_S(x)\right)$ with the coefficients c_S taking values in $[-1,+1]$ rounded to the nearest multiple of $\frac{\varepsilon}{8n^d}$, while discarding the polynomials whose L_2 norm is larger than 1. This way, since $\chi_S(x) \in \{\pm1\}$, when we round the coefficients of P to a multiple of $\frac{\varepsilon}{8n^d}$ the value at any $\boldsymbol{x} \in \{\pm1\}^n$ cannot change by more than $\frac{\varepsilon}{4}$, as there are at most n^d contributing monomials [16]. The total number of such polynomials is at most $\left(\frac{8n^d}{\varepsilon}\right)^{n^d} = e^{n^d \ln \frac{8n^d}{\varepsilon}}$.

Now, by taking a union bound on all elements of $\mathcal{C}$ we get

$$\Pr_{\text{choice of } T}\left[\max_{P\in\mathcal{C}}\left|\|f-P\|_1 - \left|\mathop{\mathbb{E}}_{\boldsymbol{x}\sim T}[|f(\boldsymbol{x})-P(\boldsymbol{x})|]\right| \leq \frac{\varepsilon}{2}\right] \geq 1-\exp\left(-\frac{\varepsilon^2}{32}\frac{|T|}{n^{2d}} + n^d \ln\frac{8n^d}{\varepsilon}\right)$$

Finally, if the above holds, by choosing a polynomial P_{closest} from $\mathcal{C}$ to minimize $\max_{\boldsymbol{x}\in\{\pm1\}^n} |P(\boldsymbol{x}) - P_{\text{closest}}(\boldsymbol{x})|$ we get that

$$\Pr_{\text{choice of } T}\left[\max_{\substack{\text{degree-}d \text{ polynomial } P \text{ over } \{\pm1\}^n\\ \text{with } \|P\|_2 \leq 1}}\left|\|f-P\|_1 - \left|\mathop{\mathbb{E}}_{\boldsymbol{x}\sim T}[|f(\boldsymbol{x})-P(\boldsymbol{x})|]\right| \leq \varepsilon\right] \geq 1-\exp\left(-\frac{\varepsilon^2}{8}\frac{|T|}{n^{2d}} + n^d \ln\frac{4n^d}{\varepsilon}\right).$$

Substituting $|T| \geq n^{5d}\frac{100}{\varepsilon^2}\ln\frac{1}{\varepsilon}\ln\frac{1}{\delta}$ we see that the above expression is at least $1-\delta$.

[16] To have $\|P_{\text{closest}}(\boldsymbol{x})\|_2 \leq \|P\|_2 \leq 1$ we should round to the closest multiple of $\frac{\varepsilon}{8n^d}$ that is smaller in the absolute value of the coefficient being rounded