The Pennsylvania State University
The Graduate School
College of Engineering

UNATENESS TESTING

A Thesis in
Computer Science and Engineering
by
Roksana Baleshzar

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2017

The thesis of Roksana Baleshzar was reviewed and approved* by the following:

Sofya Raskhodnikova
Professor of Computer Science
Thesis Advisor, Chair of Committee

Paul Medvedev
Professor of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

We study the problem of testing Unateness. We solve this problem completely for real-valued functions by giving optimal testers for different domains and different type of testers.

A property tester is adaptive if its queries depend on the answers to its previous queries. Otherwise, it is nonadaptive. We prove that adaptivity helps for the unateness testing of real-valued functions. While it does not help for a large class of similar properties including monotonicity.

Table of Contents

Acknowledgments **v**

Chapter 1
Introduction **2**
1.1 Problem Definition 2
1.2 Previous Works 2
1.3 Our Work 3
1.4 Formal Statements and Technical Overview 4
1.4.1 Overview of Techniques 4
1.4.2 The nonadaptive lower bound 6

Chapter 2
Upper Bounds **8**
2.1 The Nonadaptive Tester over the Hypercube 8
2.2 The Adaptive Tester over the Hypercube 10
2.3 Extension to Hypergrids 11

Chapter 3
Lower Bounds **16**
3.1 The Lower Bound for Nonadaptive Testers over Hypercubes 16
3.1.1 Reduction to Comparison-Based Testers 16
3.1.2 The Hard Distributions 20
3.1.3 From Functions to Signed Graphs that are Hard to Distinguish 22
3.1.4 Proving Theorem 3.1.11: Good and Bad Events 24
3.1.5 Bounding the Probability of Bad Events: Proof of Theorem 3.1.16 . . 26

Chapter 4
Conclusion **29**
4.1 Conclusion and Open Directions 29

Acknowledgments

I would first like to thank my thesis adviser Sofya Raskhodnikova who helped me a lot whenever I ran into a trouble spot or had a question about my research or writing.

Chapter 1
Introduction

1.1 Problem Definition

We study the problem of testing whether a given real-valued function f on domain $[n]^d$, where $n, d \in \mathbb{N}$, is unate. A function $f : [n]^d \to \mathbb{R}$ is *unate* if for every coordinate $i \in [d]$, the function is either nonincreasing in the i^{th} coordinate or nondecreasing in the i^{th} coordinate. Monotone functions are special case of Unate functions, which are nondecreasing in all coordinates.

The domain $[n]^d$ is called a *hypergrid* and the special case $\{0,1\}^d$ is called a *hypercube*. The distance between two functions $f, g : [n]^d \to \mathbb{R}$ is equal to the fraction of points $x \in [n]^d$ where $f(x) \neq g(x)$. Given a parameter $\varepsilon \in (0,1)$, two functions f and g are ε-far from each other if the distance between f and g is at least ε. A function f is ε-far from a property P if it is ε-far from any function which has property P. A *property tester* **??** for a property P is a randomized algorithm which, given parameter $\varepsilon \in (0,1)$ and oracle access to the input function f, accepts f with probability $\frac{2}{3}$, if it has the property P, and rejects f with probability $\frac{2}{3}$, if it is ε-far from P. A testing algorithm for property P has 1-*sided error* if it always accepts all input functions that satisfy property P and 2-*sided error*, otherwise. A tester is *nonadaptive* if it makes all queries in advance, and *adaptive* if it can make queries after seeing answers to previous ones.

1.2 Previous Works

The problem of testing unateness was first considered by **?**. In their work, by extending their monotonicity tester, they obtain a nonadaptive, 1-sided error tester for unateness with query complexity $O(\frac{d^{3/2}}{\epsilon})$. **?** improved this upper bound by giving an adaptive unateness tester

with query complexity $O(\frac{d \log d}{\varepsilon})$.

The related properties of monotonicity, Lipschit **???** and bounded-derivative properties **?** has been studied extensively for different types of functions in the context of property testing. The problem of testing monotonicity of Boolean functions over the hypercube domain was first introduced by **?**. It was shown in **??** that monotonicity for hypercube domain can be tested with query complexity $O(\frac{d}{\epsilon})$. A monotonicity tester with better query complexity of $\tilde{O}(\frac{d^{7/8}}{\epsilon^{3/2}})$ was introduced by **?**. **?** modified the tester of **?** and improved the query complexity to $\tilde{O}(\frac{d^{5/6}}{\epsilon^4})$. Most recently, **?** improved the query complexity of the tester to $\tilde{O}(\frac{\sqrt{d}}{\epsilon^2})$. The lower bound for any nonadaptive one-sided error tester for monotonicity over the hypercube was proved to be $\Omega(\sqrt{d})$ by **?**. Also, **?** gave a lower bound of almost $\Omega(\sqrt{d})$ for any nonadaptive, two-sided error tester. Moreover, there is a lower bound of $\Omega(\min\{d, |R|^2\})$ over the real-valued functions by **?**. Most recently, **?** gave a lower bound of $\tilde{\Omega}(d^{\frac{1}{4}})$ for adaptive testers of Boolean functions over the hypercube. **?** proved that any adaptive, two-sided monotonicity tester for functions $f : [n]^d \to \mathbb{N}$ must make $\Omega(\frac{d \log n - \log \varepsilon^{-1}}{\varepsilon})$ queries.

1.3 Our Work

In this work, we improve previous results for unateness testing.

Specifically, we show that unateness of real-valued functions on hypercubes can be tested nonadaptively with $O(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon})$ queries and adaptively with $O(\frac{d}{\varepsilon})$ queries. More generally, we describe a $O(\frac{d}{\varepsilon} \cdot (\log \frac{d}{\varepsilon} + \log n))$-query nonadaptive tester and a $O(\frac{d \log n}{\varepsilon})$-query adaptive tester of unateness of real-valued functions over hypergrids.

In contrast to the state of knowledge for unateness testing, the complexity of testing monotonicity of real-valued functions over the hypercube and the hypergrid has been resolved. For constant distance parameter ε, it is known to be $\Theta(d \log n)$. Moreover, this bound holds for all *bounded-derivative* properties **?**, a large class that includes **b**-monotonicity and some properties quite different from monotonicity, such as the Lipschitz property. Amazingly, the upper bound for all these properties is achieved by the same simple and, in particular, nonadaptive, tester. Even though proving lower bounds for adaptive testers has been challenging in general, a line of work, starting from Fischer **?** and including **???**, has established that adaptivity does not help for this large class of properties. Since unateness is so closely related, it is natural to ask whether the same is true for testing unateness.

We answer this in the negative: we prove that any nonadaptive tester of real valued functions over the hypercube (for some constant distance parameter) must make $\Omega(d \log d)$

queries. More generally, it needs $\Omega(d(\log d + \log n))$ queries for the hypergrid domain. These lower bounds complement our algorithms, completing the picture for unateness testing of real-valued functions. From a property testing standpoint, our results establish that unateness is different from monotonicity and, more generally, any derivative-bounded property.

1.4 Formal Statements and Technical Overview

Our testers are summarized in the following theorem, stated for functions over the hypergrid domains. (Recall that the hypercube is a special case of the hypergrid with $n = 2$.)

Theorem 1.4.1. *Consider functions $f : [n]^d \to \mathbb{R}$ and a distance parameter $\varepsilon \in (0, 1/2)$.*

1. *There is a nonadaptive unateness tester that makes $O(\frac{d}{\varepsilon}(\log \frac{d}{\varepsilon} + \log n))$ queries*[1].
2. *There is an adaptive unateness tester that makes $O(\frac{d \log n}{\varepsilon})$ queries.*

Both testers have one-sided error.

Our main technical contribution is the proof that the extra $\Omega(\log d)$ is needed for nonadaptive testers. This result demonstrates a gap between adaptive and nonadaptive unateness testing.

Theorem 1.4.2. *Any nonadaptive unateness tester (even with two-sided error) for real-valued functions $f : \{0,1\}^d \to \mathbb{R}$ with distance parameter $\varepsilon = 1/8$ must make $\Omega(d \log d)$ queries.*

The lower bound for adaptive testers is an easy adaptation of the monotonicity lower bound in **?**. We state this theorem for completeness and prove it in Appendix 1.

Theorem 1.4.3. *Any unateness tester for functions $f : [n]^d \to \mathbb{R}$ with distance parameter $\varepsilon \in (0, 1/4)$ must make $\Omega\left(\frac{d \log n}{\varepsilon} - \frac{\log 1/\varepsilon}{\varepsilon}\right)$ queries.*

Theorems 1.4.2 and 1.4.3 directly imply that our nonadaptive tester is optimal for constant ε, even for the hypergrid domain. The details appear in Appendix 2.

1.4.1 Overview of Techniques

We first consider the hypercube domain. For each $i \in [d]$, an *i-edge* of the hypercube is a pair (x, y) of points in $\{0,1\}^d$, where $x_i = 0, y_i = 1$, and $x_j = y_j$ for all $j \in ([d] \setminus \{i\})$. Given an

[1] For many properties, when the domain is extended from the hypercube to the hypergrid, testers incur an extra multiplicative factor of $\log n$ in the query complexity. This is the case for our adaptive tester. However, note that the complexity of nonadaptive unateness testing (for constant ε) is $\Theta(d(\log d + \log n))$ rather than $\Theta(d \log d \log n)$.

input function $f : \{0,1\}^d \to \mathbb{R}$, we say an i-edge (x, y) is *increasing* if $f(x) < f(y)$, *decreasing* if $f(x) > f(y)$, and *constant* if $f(x) = f(y)$.

Our nonadaptive unateness tester on the hypercube uses the work investment strategy from **?** (also refer to Section 8.2.4 of Goldreich's book **?**) to "guess" a good dimension where to look for violations of unateness (specifically, both increasing and decreasing edges). For all $i \in [d]$, let α_i be the fraction of the i-edges that are decreasing, β_i be the fraction of the i-edges that are increasing, and $\mu_i = \min(\alpha_i, \beta_i)$. The dimension reduction theorem from **?** implies that if the input function is ε-far from unate, then the average of μ_i over all dimensions is at least $\frac{\varepsilon}{4d}$. If the tester knew which dimension had $\mu_i = \Omega(\varepsilon/d)$, it could detect a violation with high probability by querying the endpoints of $O(1/\mu_i) = O(d/\varepsilon)$ uniformly random edges. However, the tester does not know which μ_i is large and, intuitively, nonadaptively checks the following $\log d$ different scenarios, one for each $k \in [\log d]$: exactly 2^k different μ_i's are $\varepsilon/2^k$, and all others are 0. This leads to the query complexity of $O(\frac{d \log d}{\varepsilon})$.

With adaptivity, this search through $\log d$ different scenarios is not required. A pair of queries in each dimension detects influential coordinates (i.e., dimensions with many non-constant edges), and the algorithm focuses on finding violations among those coordinates. This leads to the query complexity of $O(d/\varepsilon)$, removing the $\log d$ factor.

It is relatively easy to extend (both adaptive and nonadaptive) testers from hypercubes to hypergrids by incurring an extra factor of $\log n$ in the query complexity. The role of i-edges is now played by *i-lines*. An i-line is a set of n domain points that differ only on coordinate i. The domain $[n]$ is called a line. Monotonicity on the line (a.k.a. sortedness) can be tested with $O(\frac{\log n}{\varepsilon})$ queries, using, for example, the classical *tree tester* from **?**. Instead of sampling a random i-edge, we sample a random i-line ℓ and run the tree tester on the restriction $f_{|\ell}$ of function f to the line ℓ. This is optimal for adaptive testers, but, interestingly, not for nonadaptive testers. We show that for each function f on the line that is ε-far from unateness, one of the two scenarios happen: (1) the tree tester is likely to find a violation of unateness; (2) function f is increasing (and also decreasing) on a constant fraction of pairs in $[n]$. This new angle on the classical tester allows us to replace the factor $(\log d)(\log n)$ with $\log d + \log n$ in the query complexity. Thus, the nonadaptive complexity becomes $O(d(\log d + \log n))$, which we show is optimal.

1.4.2 The nonadaptive lower bound

Our most significant finding is the $\log d$ gap in the query complexity between adaptive and nonadaptive testing of unateness. By previous work **??**, it suffices to prove lower bounds for *comparison-based* testers, i.e., testers that can only perform comparisons of the function values at queried points, but cannot use the values themselves. Our main technical contribution is the $\Omega(d \log d)$ lower bound for nonadaptive comparison-based testers of unateness on hypercube domains.

Intuitively, we wish to construct $K = \Theta(\log d)$ families of functions where, for each $k \in [K]$, functions in the k^{th} family have 2^k dimensions i with $\mu_i = \Theta(1/2^k)$, while $\mu_i = 0$ for all other dimensions. What makes the construction challenging is the existence of a *single, universal* nonadaptive $O(d)$-tester for all $\mathbf{b}$-monotonicity properties, proven in **?**. In other words, there is a single distribution on $O(d)$ queries that defines a nonadaptive property tester for $\mathbf{b}$-monotonicity, regardless of $\mathbf{b}$. Since unateness is the union of all $\mathbf{b}$-monotonicity properties, our construction must be able to fool such algorithms. Furthermore, nonadaptivity must be critical, since we obtained a $O(d)$-query adaptive tester for unateness.

Another obstacle is that once a tester finds a non-constant edge in each dimension, the problem reduces to testing $\mathbf{b}$-monotonicity for a vector $\mathbf{b}$ determined by the directions (increasing or decreasing) of the non-constant edges. That is, intuitively, most edges in our construction must be constant. This is one of the main technical challenges. The previous lower bound constructions for monotonicity testing **??** crucially used the fact that all edges in the hard functions were non-constant.

We briefly describe how we overcome the problems mentioned above. By Yao's minimax principle, it suffices to construct **Yes** and **No** distributions that a deterministic nonadaptive tester cannot distinguish. First, for some parameter m, we partition the hypercube into m subcubes based of the first $\log_2 m$ most significant coordinates. Both distributions, **Yes** and **No**, sample a uniform k from $[K]$, where $K = \Theta(\log d)$, and a set $R \subseteq [d]$ of cardinality 2^k. Furthermore, each subcube $j \in [m]$ selects an "action dimension" $r_j \in R$ uniformly at random. For both distributions, in any particular subcube j, the function value is completely determined by the coordinates *not in* R, and the random coordinate $r_j \in R$. Note that all the i-edges for $i \in (R \setminus \{r_j\})$ are constant. Within the subcube, the function is a linear function with exponentially increasing coefficients. In the **Yes** distribution, any two cubes j, j' with the same action dimension orient the edges in that dimension the same way (both increasing or both decreasing), while in the **No** distribution each cube decides on the orientation

independently. The former correlation maintains unateness while the latter independence creates distance to unateness. We prove that to distinguish the distributions, any comparison-based nonadaptive tester must find two distinct subcubes with the same action dimension r_j and, furthermore, make a specific query (in both) that reveals the coefficient of r_j. We show that, with $o(d \log d)$ queries, the probability of this event is negligible.

Chapter 2
Upper Bounds

In this section, we prove parts 1-2 of Theorem 1.4.1, starting from the hypercube domain.

Recall the definition of i-edges and i-lines from Section 1.4.1 and what it means for an edge to be increasing, decreasing, and constant.

The starting point for our algorithms is the dimension reduction theorem from **?**. It bounds the distance of $f : [n]^d \to \mathbb{R}$ to monotonicity in terms of average distances of restrictions of f to one-dimensional functions.

Theorem 2.0.1 (Dimension Reduction, Theorem 1.8 in **?**)**.** *Fix a bit vector $\mathbf{b} \in \{0,1\}^d$ and a function $f : [n]^d \to \mathbb{R}$ which is ε-far from $\mathbf{b}$-monotonicity. For all $i \in [d]$, let μ_i be the average distance of $f_{|\ell}$ to $\mathbf{b}_i$-monotonicity over all i-lines ℓ. Then,*

$$\sum_{i=1}^{d} \mu_i \geq \frac{\varepsilon}{4}.$$

For the special case of the hypercube domains, i-lines become i-edges, and the average distance μ_i to $\mathbf{b}_i$-monotonicity is the fraction of i-edges on which the function is not $\mathbf{b}_i$-monotone.

2.1 The Nonadaptive Tester over the Hypercube

We now describe Algorithm 1, the nonadaptive tester for unateness over the hypercubes.

It is evident that Algorithm 1 is a nonadaptive, one-sided error tester. Furthermore, its query complexity is $O\left(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\right)$. It suffices to prove the following.

Lemma 2.1.1. *If f is ε-far from unate, Algorithm 1 rejects with probability at least $2/3$.*

Algorithm 1: The Nonadaptive Unateness Tester over Hypercubes

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : \{0,1\}^d \to \mathbb{R}$.

1 **for** $r = 1$ *to* $\lceil 3 \log(4d/\varepsilon) \rceil$ **do**
2 **repeat** $s_r = \lceil \frac{16d \ln 4}{\varepsilon \cdot 2^r} \rceil$ **times**
3 Sample a dimension $i \in [d]$ uniformly at random.
4 Sample $3 \cdot 2^r$ i-edges uniformly and independently at random and **reject** if there exists an increasing edge and a decreasing edge among the sampled edges.
5 **accept**

Proof. Recall that α_i is the fraction of i-edges that are decreasing, β_i is the fraction of i-edges that are increasing and $\mu_i = \min(\alpha_i, \beta_i)$.

Define the d-dimensional bit vector $\mathbf{b}$ as follows: for each $i \in [d]$, let $\mathbf{b}_i = 0$ if $\alpha_i < \beta_i$ and $\mathbf{b}_i = 1$ otherwise. Observe that the average distance of f to $\mathbf{b}_i$-monotonicity over a random i-edge is precisely μ_i. Since f is ε-far from being unate, f is also ε-far from being $\mathbf{b}$-monotone. By Theorem 2.0.1, $\sum_{i \in [d]} \mu_i \geq \frac{\varepsilon}{4}$. Hence, $\mathbf{E}_{i \in [d]}[\mu_i] \geq \frac{\varepsilon}{4d}$. We now apply the work investment strategy due to Berman et al. **?** to get an upper bound on the probability that Algorithm 1 fails to reject.

Theorem 2.1.2 (**?**). *For a random variable $X \in [0,1]$ with $\mathbf{E}[X] \geq \mu$ for $\mu < \frac{1}{2}$, let $p_r = \Pr[X \geq 2^{-r}]$ and $\delta \in (0,1)$ be the desired error probability. Let $s_r = \frac{4 \ln 1/\delta}{\mu \cdot 2^r}$. Then,*

$$\prod_{r=1}^{\lceil 3 \log(1/\mu) \rceil} (1 - p_r)^{s_r} \leq \delta.$$

Consider running Algorithm 1 on a function f that is ε-far from unate. Let $X = \mu_i$ where i is sampled uniformly at random from $[d]$. Then $\mathbf{E}[X] \geq \frac{\varepsilon}{4d}$. Applying the work investment strategy (Theorem 2.1.2) on X with $\mu = \frac{\varepsilon}{4d}$, we get that the probability that, in some iteration, Step 3 samples a dimension i such that $\mu_i \geq 2^{-r}$ is at least $1 - \delta$. We set $\delta = 1/4$. Conditioned on sampling such a dimension, the probability that Step 4 fails to obtain an increasing edge and a decreasing edge among its $3 \cdot 2^r$ samples is at most $2\left(1 - 2^{-r}\right)^{3 \cdot 2^r} \leq 2e^{-3} < 1/9$, as the fraction of both increasing and decreasing edges in the dimension is at least 2^{-r}. Hence, the probability that Algorithm 1 rejects f is at least $\frac{3}{4} \cdot \frac{8}{9} = \frac{2}{3}$, which completes the proof of Lemma 2.1.1. □

2.2 The Adaptive Tester over the Hypercube

We now describe Algorithm 2, an adaptive tester for unateness over the hypercube domain with good expected query complexity. The final tester is obtained by repeating this tester and accepting if the number of queries exceeds a specified bound.

Algorithm 2: The Adaptive Unateness Tester over Hypercubes

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : \{0,1\}^d \to \mathbb{R}$.

1 **repeat** $10/\varepsilon$ **times**
2 **for** $i = 1$ *to* d **do**
3 Sample an i-edge e_i uniformly at random.
4 **if** e_i *is non-constant (i.e., increasing or decreasing)* **then**
5 Sample i-edges uniformly at random till we obtain a non-constant edge e_i'.
6 **reject** if one of the edges e_i, e_i' is increasing and the other is decreasing.
7 **accept**

Claim 2.2.1. *The expected number of queries made by Algorithm 2 is* $40d/\varepsilon$.

Proof. Consider one iteration of the **repeat**-loop in Step 1. We prove that the expected number of queries in this iteration is $4d$. The total number of queries in Step 3 is $2d$, as 2 points per dimension are queried. Let E_i be the event that edge e_i is non-constant and T_i be the random variable for the number of i-edges sampled in Step 5. Then $\mathbf{E}[T_i] = \frac{1}{\alpha_i+\beta_i} = \frac{1}{\Pr[E_i]}$. Therefore, the expected number of all edges sampled in Step 5 is $\sum_{i=1}^{d} \Pr[E_i] \cdot \mathbf{E}[T_i] = \sum_{i=1}^{d} \Pr[E_i] \cdot \frac{1}{\Pr[E_i]} = d$. Hence, the expected number of queries in Step 5 is $2d$. Since there are $10/\varepsilon$ iterations in Step 1, the expected number of queries in Algorithm 2 is $40d/\varepsilon$. □

Claim 2.2.2. *If f is ε-far from unate, Algorithm 2 accepts with probability at most* $1/6$.

Proof. First, we bound the probability that a violation of unateness is detected in some dimension $i \in [d]$ in one iteration of the **repeat**-loop. Consider the probability of finding a decreasing i-edge in Step 3, and an increasing i-edge in Step 5. The former is exactly α_i, and the latter is $\frac{\beta_i}{\alpha_i+\beta_i}$. Therefore, the probability we detect a violation from dimension i is $\frac{2\alpha_i\beta_i}{\alpha_i+\beta_i} \geq \min(\alpha_i, \beta_i) = \mu_i$. The probability that we fail to detect a violation in any of the d dimensions is at most $\prod_{i=1}^{d}(1-\mu_i) \leq \exp\Big(-\sum_{i=1}^{d}\mu_i\Big)$, which is at most $e^{-\varepsilon/4}$ by Theorem 2.0.1 (Dimension Reduction). By Taylor expansion of $e^{-\varepsilon/4}$, the probability of finding a violation in one iteration is at least $1 - e^{-\varepsilon/4} \geq \frac{\varepsilon}{4} - \frac{\varepsilon^2}{32} > \frac{\varepsilon}{5}$. The probability that Algorithm 2 does not reject in any iteration is at most $(1-\varepsilon/5)^{10/\varepsilon} < 1/6$. □

Proof of Theorem 1.4.1, Part 2 (for the special case of the hypercube domain). We run Algorithm 2, aborting and accepting if we ever make more than $240d/\varepsilon$ queries. By Markov's inequality, the probability of aborting is at most 1/6. By Claim 2.2.2, if f is ε-far from unate, Algorithm 2 accepts with probability at most 1/6. The theorem follows by a union bound. □

2.3 Extension to Hypergrids

We start by establishing terminology for lines and pairs. Consider a function $f : [n]^d \to \mathbb{R}$. Recall the definition of i-lines from Section 1.4.1. A pair of points that differ only in coordinate i is called an i-pair. An i-pair (x, y) with $x_i < y_i$ is called *increasing* if $f(x) < f(y)$, *decreasing* if $f(x) > f(y)$, and *constant* if $f(x) = f(y)$.

Algorithm 3: Tree Tester

input : Query access to a function $h : [n] \mapsto \mathbb{R}$.

1 Pick $x \in [n]$ uniformly at random.
2 Let $Q_x \subseteq [n]$ be the set of points visited in a binary search for x. Query h on all points in Q_x.
3 If there is an increasing pair in Q_x, set $\mathsf{dir} \leftarrow \{\uparrow\}$; otherwise, $\mathsf{dir} \leftarrow \emptyset$.
4 If there is a decreasing pair in Q_x, update $\mathsf{dir} \leftarrow \mathsf{dir} \cup \{\downarrow\}$.
5 **Return** dir.

The main tool for extending Algorithms 1 and 2 to work on hypergrids is the *tree tester*, designed by Ergun et al. **?** to test monotonicity of functions $h : [n] \to \mathbb{R}$. We modify the tree tester to return information about directions it observed instead of just accepting or rejecting. See Algorithm 3. We call a function $h : [n] \mapsto \mathbb{R}$ *antimonotone* if $f(x) \geq f(y)$ for all $x < y$. The following lemma summarizes the guarantee of the tree tester.

Lemma 2.3.1 (??). *If $h : [n] \mapsto \mathbb{R}$ is ε-far from monotone (respectively, antimonotone), then the output of Algorithm 3 on h contains $\downarrow$ (respectively, $\uparrow$) with probability at least ε.*

Our hypergrid testers are stated in Algorithms 4 and 5. Next, we explain how Lemma 2.3.1 and Theorem 2.0.1 are used in the analysis of the adaptive tester. For a dimension $i \in [d]$, let α_i and β_i denote the average distance of $f_{|\ell}$ to monotonicity and antimonotonicity, respectively, over all i-lines ℓ. Then $\mu_i := \min(\alpha_i, \beta_i)$ is the average fraction of points per i-line that needs to change to make f unate. Define the $\mathbf{b}$-vector with $\mathbf{b}_i = 0$ if $\alpha_i < \beta_i$, and $\mathbf{b}_i = 1$

Algorithm 4: The Adaptive Unateness Tester over Hypergrids

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : [n]^d \to \mathbb{R}$.

repeat $10/\varepsilon$ **times**
  for $i = 1$ *to* d **do**
    Sample an i-line ℓ_i uniformly at random.
    Let dir_i be the output of Algorithm 3 on $f_{|\ell_i}$.
    if $\mathsf{dir}_i \neq \emptyset$ **then**
      Sample i-lines uniformly at random and run Algorithm 3 on f restricted to each line until it returns a non-empty set. Call it dir'_i.
      If $\mathsf{dir}_i \cup \mathsf{dir}'_i = \{\uparrow, \downarrow\}$, **reject**.
accept

Algorithm 5: The Nonadaptive Unateness Tester over Hypergrids

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : [n]^d \to \mathbb{R}$.

repeat $220/\varepsilon$ **times**
  for $i = 1$ *to* d **do**
    Sample an i-line ℓ uniformly at random.
    Reject if Algorithm 3, on input $f_{|\ell}$, returns $\{\uparrow, \downarrow\}$.
for $r = 1$ *to* $\lceil 3 \log(200d/\varepsilon) \rceil$ **do**
  repeat $s_r = \lceil \frac{800d \ln 4}{\varepsilon \cdot 2^r} \rceil$ **times**
    Sample a dimension $i \in [d]$ uniformly at random.
    Sample $3 \cdot 2^r$ i-pairs uniformly and independently at random.
    If we find an increasing and a decreasing pair among the sampled pairs, **reject**.
accept

otherwise. By Theorem 2.0.1, if f is ε-far from unate, and thus ε-far from **b**-monotone, then $\sum_{i=1}^{d} \mu_i \geq \varepsilon/4$. By Lemma 2.3.1, the probability that the output of Algorithm 3 on $f_{|\ell}$ contains $\downarrow$ (respectively, $\uparrow$), where ℓ is a uniformly random i-line, is at least α_i (respectively, β_i). The rest of the analysis of Algorithm 4 is similar to that in the hypercube case.

Proof of Theorem 1.4.1, Part 2. The tester is Algorithm 4. As in the proof of Claim 2.2.1, the expected running time of Algorithm 4 is at most $(40d \log n)/\varepsilon$. The proof of Claim 2.2.2 carries over almost word-to-word. Fix dimension i. The probability that $\downarrow \in \mathsf{dir}_i$ in Step 4 is at least α_i. The probability that $\uparrow \in \mathsf{dir}'_i$ in Step 6 is at least $\frac{\beta_i}{\alpha_i + \beta_i}$. The rest of the calculation is identical to that of the proof of Claim 2.2.2. □

To analyze the nonadaptive tester, we prove Lemma 2.3.2, which demonstrates the power of the tree tester and may be of independent interest.

Lemma 2.3.2. *Consider a function $h : [n] \to \mathbb{R}$ which is ε-far from monotone (respectively, antimonotone). At least one of the following holds:*

1. $\Pr[$*Algorithm 3, on input h, returns* $\{\uparrow, \downarrow\}] \geq \varepsilon/25$.
2. $\Pr_{u,v\in[n]}[(u, v)$ *is a decreasing (respectively, increasing) pair*$] \geq \varepsilon/25$.

Proof. Let T be a balanced binary search tree consisting of elements in $[n]$, such that the set of points visited in a binary search for some $x \in [n]$ corresponds to a path from the root to the node containing x in T. Let Q_x denote the set of points visited in a binary search for $x \in [n]$. For $x, y \in [n]$, denote the least common ancestor of x and y by $\mathsf{lca}(x, y)$.

Let $W_{\uparrow\downarrow}$ be a set of points x such that Q_x contains both an increasing and a decreasing pair (with respect to h). If $|W_{\uparrow\downarrow}| \geq \frac{\varepsilon n}{10}$, then Case 1 of Lemma 2.3.2 holds. We may therefore assume that $|W_{\uparrow\downarrow}| < \frac{\varepsilon n}{10}$. Let $\mathcal{E}$ be the event that for any $u, v \in [n]$ such that $u < v$, the pair (u, v) is decreasing. We will prove that $\Pr[\mathcal{E}] \geq \varepsilon/25$.

Let $W_\downarrow$ be that set of points $x \in [n]$ such that Q_x contains a decreasing pair. Similarly, define the set $W_\uparrow$. Let W_c denote the set of points x such that $h_{|Q_x}$ is constant.

Claim 2.3.3 (?). *The function h restricted to the set $W_\uparrow \cup W_c$ is monotone.*

Proof. The proof is by contradiction. Suppose $x, y \in (W_\uparrow \cup W_c)$ such that $x < y$, but $h(x) > h(y)$. Consider $z = \mathsf{lca}(x, y)$. Either $h(x) > h(z)$ or $h(z) > h(y)$, contradicting the fact that $x, y \in W_\uparrow \cup W_c$. □

By symmetry, the function h restricted to the set $W_\downarrow \cup W_c$ is antimonotone.

A priori, points in $W_\uparrow$ and $W_\downarrow$ could be interspersed. The next claim shows that they are in different halves of the tree T.

Claim 2.3.4. *If $x \in W_\downarrow$ and $y \in W_\uparrow$, then $\mathsf{lca}(x, y)$ is the root of T (which is equal to $\lceil n/2 \rceil$).*

Proof. Suppose not. Let $z := \mathsf{lca}(x, y)$ and w be the parent of z. Consider the case where z is the left child of w, x lies in the left subtree of z and y lies in the right subtree of z. (All the other cases have analogous proofs.) Observe that all points in Q_y lie in the interval $[z, w]$. Both w and z are in Q_x as well as in Q_y. As $x \in W_\uparrow$ and $y \in W_\downarrow$, it must be the case that $h(w) = h(z)$. Since $y \notin W_{\uparrow\downarrow}$, for all $p \in Q_y$, we have $h(p) = h(w)$. This contradicts the fact that $y \in W_\uparrow$.

In all cases, we conclude that either $x \notin W_\downarrow$ or $y \notin W_\uparrow$. Thus, z cannot have a parent, and $z = \lceil n/2 \rceil$. □

Claim 2.3.5. *Let $g : [n] \mapsto \mathbb{R}$ be an antimonotone function and* $\mathsf{dist}(g, \text{constant})$ *denote the fraction of points that need to be changed so that g is a constant function. If g is antimonotone, and* $\mathsf{dist}(g, \text{constant}) \geq \rho$*, where $\rho \leq \frac{1}{2}$, then*

$$\Pr_{u,v \in [n]: u<v}[(u, v) \text{ is decreasing}] \geq \frac{\rho}{2}.$$

Proof. The probability that $g(u) \neq g(v)$ is at least $\rho(1-\rho)$ which is at least $\frac{\rho}{2}$ when $\rho \leq \frac{1}{2}$. Since g is antimonotone, (u, v) is a decreasing pair. □

Let L (respectively, R) be the set of points in $[n] \setminus W_{\uparrow\downarrow}$ in the left (respectively, right) subtree of the root. Define $\mu_L := |L|/n$; similarly, define μ_R. Observe that both μ_L and μ_R are at least $\frac{1}{2} - \frac{\varepsilon}{10}$. By Claims 2.3.3 and 2.3.4, $h_{|L}$ (and $h_{|R}$) is either monotone or antimonotone. Now, if any of these two functions were antimonotone and $\frac{\varepsilon}{2}$-far from being constant (w.l.o.g., assume $h_{|L}$ satisfies the condition), then by Claim 2.3.5, we would have

$$\Pr[\mathcal{E}] \geq \Pr_{u<v}[(u, v) \text{ is decreasing and } u, v \in L] \geq \frac{\varepsilon}{4} \cdot \left(\frac{1}{2} - \frac{\varepsilon}{10}\right)^2 \geq \frac{\varepsilon}{25}.$$

Assume that this doesn't occur. We have two cases.

Case 1. Both $h_{|L}$ and $h_{|R}$ are $\frac{\varepsilon}{2}$-close[1] to being constant. In this case, at least $(1 - \frac{\varepsilon}{2})|L|$ points of L evaluate to a constant C_1, and at least $(1 - \frac{\varepsilon}{2})|R|$ points of R evaluate to constant C_2. We must have $C_1 > C_2$, for otherwise, we can make h monotone by changing only $\frac{\varepsilon}{2} \cdot (|R| + |L|) + \frac{\varepsilon n}{10} < \varepsilon n$ points, which is a contradiction. Hence,

$$\Pr[\mathcal{E}] \geq \Pr_{u<v}[h(u) = C_1 \text{ and } h(v) = C_2] \geq \left(1 - \frac{\varepsilon}{2}\right)^2 \mu_L \mu_R > \frac{1}{4} \cdot \left(\frac{1}{2} - \frac{\varepsilon}{10}\right)^2 \geq \frac{\varepsilon}{25}.$$

Case 2. At least one of the functions is $\frac{\varepsilon}{2}$-far from being constant and is monotone. W.l.o.g., assume $h_{|L}$ satisfies this condition. Note that all points in L are only in $W_\uparrow \cup W_c$, and so, all points in R must be in $W_\downarrow \cup W_c$. This implies that $h_{|R}$ is antimonotone. (Note that a constant function is also antimonotone.) But then, $h_{|R}$ must be $\frac{\varepsilon}{2}$-close to being constant. Then at least $(1 - \frac{\varepsilon}{2})|R|$ points in R evaluate to a constant, say C. Let U denote the set of points in L whose values are strictly greater than C. Since $h_{|L}$ is monotone, we can make h monotone by deleting all points in $U, W_{\uparrow\downarrow}$, and the points in R that do not evaluate to C. The total number of points to be deleted is at most $|U| + \frac{\varepsilon n}{10} + \frac{\varepsilon n}{2}$, which must be at least εn,

[1] A function h is ε-close to a property $\mathcal{P}$ if it is sufficient to change at most ε-fraction of values in h to make it satisfy $\mathcal{P}$.

as h is ε-far from monotone. Hence, $|U| > \varepsilon n/3$. Therefore,

$$\Pr[\mathcal{E}] \geq \Pr_{u<v}[u \in U \text{ and } h(v) = C] \geq \frac{\varepsilon}{3} \cdot \left(1 - \frac{\varepsilon}{2}\right) \mu_R > \frac{\varepsilon}{25}.$$

This completes the proof of Lemma 2.3.2. □

We now analyze Algorithm 5. It is evident that it has one-sided error and makes $O(\frac{d}{\varepsilon}(\log n + \log \frac{d}{\varepsilon}))$ queries. It suffices to prove the following.

Theorem 2.3.6. *If $f : [n]^d \mapsto \mathbb{R}$ is ε-far from unate, then Algorithm 5 rejects with probability at least $2/3$.*

Proof. For any line ℓ, we define the following quantities.

- α_ℓ: the distance of $f_{|\ell}$ to monotonicity.
- β_ℓ: the distance of $f_{|\ell}$ to antimonotonicity.
- σ_ℓ: the probability that Algorithm 3, on input $f_{|\ell}$, returns $\{\uparrow, \downarrow\}$.
- δ_ℓ: the probability that a uniformly random pair in ℓ is decreasing.
- λ_ℓ: the probability that a uniformly random pair in ℓ is increasing.

Let L_i be the set of i-lines. By Theorem 2.0.1,

$$\frac{1}{n^{d-1}} \sum_{i=1}^{d} \min\left(\sum_{\ell \in L_i} \alpha_\ell, \sum_{\ell \in L_i} \beta_\ell\right) \geq \frac{\varepsilon}{4}.$$

By Lemma 2.3.2, for every line ℓ, we have $\sigma_\ell + \delta_\ell \geq \alpha_\ell/25$ and $\sigma_\ell + \lambda_\ell \geq \beta_\ell/25$. Also note,

$$\frac{1}{n^{d-1}} \sum_{i=1}^{d} \left[\sum_{\ell \in L_i} \sigma_\ell + \min\left(\sum_{\ell \in L_i} \delta_\ell, \sum_{\ell \in L_i} \lambda_\ell\right)\right] \geq \frac{1}{n^{d-1}} \sum_{i=1}^{d} \min\left(\sum_{\ell \in L_i} (\sigma_\ell + \delta_\ell), \sum_{\ell \in L_i} (\sigma_\ell + \lambda_\ell)\right)$$

Combining these bounds, we obtain that the LHS is at least $\varepsilon/100$. Note that the first term, which is equal to $\sum_{i=1}^{d} \mathbf{E}_{\ell \in L_i}[\sigma_\ell]$, is the expected number of times a single iteration of Steps 2-4 rejects. If this quantity is at least $\varepsilon/200$, then the tester rejects with probability at least $2/3$. If not, then we have $n^{-(d-1)} \sum_{i=1}^{d} \min(\sum_{\ell \in L_i} \delta_\ell, \sum_{\ell \in L_i} \lambda_\ell) \geq \varepsilon/200$. Using a calculation identical to that of the proof of Lemma 2.1.1, the probability that Step 9 rejects in some iteration is at least $2/3$. □

Chapter 3
Lower Bounds

3.1 The Lower Bound for Nonadaptive Testers over Hypercubes

In this section, we prove Theorem 1.4.2, which gives a lower bound for nonadaptive unateness testers for functions over the hypercube.

Fischer **?** showed that in order to prove lower bounds for a general class of properties on the line domain, it is sufficient to consider a special class of testers called *comparison-based testers*. The properties he looked at are called *order-based properties* (see Definition 3.1.2) and they include monotonicity and unateness. A tester is *comparison-based* if it bases its decisions only on the *order* of the function values at the points it queried, and not on the values themselves. Chakrabarty and Seshadhri **?** extended Fischer's proof to monotonicity on any partially-ordered domain. As we show in Section 3.1.1 below, Chakrabarty and Seshadhri's proof goes through for all order-based properties on partially-ordered domains. We include this proof for completeness, filling in the details needed to generalize the original proof.

Our main technical contribution is the construction of a distribution of functions $f : \{0,1\}^d \to \mathbb{R}$ on which every nonadaptive comparison-based tester must query $\Omega(d \log d)$ points to determine whether the sampled function is unate or far from unate. We describe this construction in Section 3.1.2 and show its correctness in Sections 3.1.3-3.1.5.

3.1.1 Reduction to Comparison-Based Testers

In this section, we prove that if there exists an ε-tester for an order-based property of functions over a partially-ordered domain, then there exists a comparison-based ε-tester for the same

property making the same number of queries. This is stated in Theorem 3.1.3. Before stating the theorem, we introduce several definitions.

Definition 3.1.1. *A (t, ε, δ)-tester for a property is a (2-sided error) ε-tester making at most t queries, that errs with probability at most δ.*

Definition 3.1.2 (Order-based property)**.** *For an arbitrary partial order D and an arbitrary total order R, a property $\mathcal{P}$ of functions $f : D \to R$ is* order-based *if, for all strictly increasing maps $\phi : R \to R$ and all functions f, we have $\mathsf{dist}(f, \mathcal{P}) = \mathsf{dist}(\phi \circ f, \mathcal{P})$.*

Specifically, unateness is an order-based property. The following theorem is an extension of Theorem 5 in **?** and Theorem 2.1 in **?**. In particular, Theorem 2.1 in **?** was proved with the assumption that the function values are distinct. We generalize the theorem by removing this assumption.

Theorem 3.1.3 (implicit in **??**)**.** *Let $\mathcal{P}$ be an order-based property of functions $f : D \to \mathbb{N}$. Suppose there exists a (t, ε, δ)-tester for $\mathcal{P}$. Then there exists a comparison-based $(t, \varepsilon, 2\delta)$-tester for $\mathcal{P}$.*

The rest of this section is devoted to proving Theorem 3.1.3. Our proof closely follows the proof of Theorem 2.1 in **?**. The proof has two parts. In the first part, we describe a reduction from a tester to a *discretized tester* and, in the second part, we describe a reduction from a discretized tester to a comparison-based tester.

Let $\mathcal{P}$ be a property of functions $f : D \to R$ for an arbitrary partial order D and an arbitrary total order $R \subseteq \mathbb{N}$. Let $\mathcal{T}$ be a (t, ε, δ)-tester for $\mathcal{P}$. First, we define a family of probability functions that completely characterizes $\mathcal{T}$. Fix some $s \in [t]$. Consider the point in time in an execution of the tester $\mathcal{T}$ on some input function f, where exactly s queries have been made. Suppose these queries are $x_1, x_2, \ldots, x_s \in D$ and the corresponding answers are $a_1 = f(x_1), a_2 = f(x_2), \ldots, a_s = f(x_s)$. Let *query vector* X be $(x_1, \ldots, x_s)$ and *answer vector* A be $(a_1, \ldots, a_s)$. The next action of the algorithm is either choosing the $(s+1)^{\text{th}}$ query from D or outputting *accept* or *reject*. For each action $y \in D \cup \{\texttt{accept}, \texttt{reject}\}$, let $p_X^y(A)$ denote the probability that $\mathcal{T}$ chooses action y after making queries X and receiving answers A. Since $p_X^y(A)$ is a probability distribution,

$$\forall s < t, \forall X \in D^s, \forall A \in R^s \sum_{y \in D \cup \{\texttt{accept}, \texttt{reject}\}} p_X^y(A) = 1.$$

Furthermore, the tester cannot make more than t queries, and so the action $(t+1)$ must be either `accept` or `reject`. Formally,

$$\forall X \in D^t, \forall A \in R^t \sum_{y \in \{\texttt{accept},\texttt{reject}\}} p_X^y(A) = 1.$$

Definition 3.1.4 (Discretized tester)**.** *A tester $\mathcal{T}$ is* discretized *if all $p_X^y(A)$-values associated with $\mathcal{T}$ come from the range $\left\{\frac{i}{K} : i \in \{0, 1, \ldots, K\}\right\}$ for some integer K.*

Chakrabarty and Seshadhri **?** proved that if there exists a (t, ε, δ)-monotonicity tester $\mathcal{T}$ for functions $f : D \to \mathbb{N}$, then there exists a discretized $(t, \varepsilon, 2\delta)$-monotonicity tester $\mathcal{T}'$ for the same class of functions. Both the statement and the proof in **?** hold not only for testers of monotonicity, but for testers of all properties of functions $f : D \to R$.

Lemma 3.1.5 (implicit in (**?**, Lemma 2.2))**.** *Suppose there exists a (t, ε, δ)-tester $\mathcal{T}$ for a property $\mathcal{P}$ of functions $f : D \to R$. Then, there exists a $(t, \varepsilon, 2\delta)$-discretized tester $\mathcal{T}'$ for $\mathcal{P}$.*

This completes the first part of the proof.

Next, we will show how to transform a discretized tester into a comparison-based tester. Intuitively, a tester is comparison-based if each query of the tester depends only on the ordering of the answers to the previous queries, not on the values themselves. We define a family of probability functions q in order to characterize comparison-based testers. The q-functions are defined in terms of p-functions, but, in their definition, we decouple the set of values that were received as answers from their positions in the answer vector. Let V represent the set $\{a_1, \ldots, a_s\}$ of answer values (without duplicates). Let r be the number of (distinct) values in V. Note that $r \leq s$. Suppose, V is $\{v_1, v_2, \ldots, v_r\}$ where $v_1, \ldots, v_r \in R$ and $v_1 < v_2 < \ldots < v_r$. Let ρ be the map from positions of values in the answer vector to their corresponding indices in V, that is, $\rho : [s] \to [r]$. Observe that ρ is surjective. The q-functions are defined as follows:

$$q_{X,\rho}^y(V) = p_X^y((v_{\rho(1)}, v_{\rho(2)}, \ldots, v_{\rho(s)})).$$

Let $R^{(r)}$ denote the set of all subsets of R of size r.

Definition 3.1.6 (Comparison-based tester)**.** *A tester $\mathcal{T}$ for an order-based property $\mathcal{P}$ is* comparison-based *for functions $f : D \to R$, if for all r, s satisfying $r \leq s \leq t$, and all $X \in D^s$, $y \in D \cup \{\texttt{accept}, \texttt{reject}\}$ and surjections $\rho : [s] \to [r]$, the function $q_{X,\rho}^y$ is constant on $R^{(r)}$. That is, for all $V, V' \in R^{(r)}$, we have $q_{X,\rho}^y(V) = q_{X,\rho}^y(V')$.*

To complete the proof of Theorem 3.1.3, we show that if there exists a discretized tester $\mathcal{T}$ for an order-based property $\mathcal{P}$ over the functions $f : D \to \mathbb{N}$, then there exists an infinite set $R \subseteq \mathbb{N}$ such that, for functions $f : D \to R$, the tester $\mathcal{T}$ is comparison-based. The existence of this infinite set R is proved using Ramsey theory arguments.

We introduce some Ramsey theory terminology. Consider an integer C, where $[C]$ represents a set of colors. For any positive integer i, a *finite coloring* of $\mathbb{N}^{(i)}$ is a function $\texttt{col}_i : \mathbb{N}^{(i)} \to [C]$. An infinite set $R \subseteq \mathbb{N}$ is *monochromatic* with respect to $\texttt{col}_i$ if for all i-sized subsets $V, V' \in R^{(i)}$, the color $\texttt{col}_i(V) = \texttt{col}_i(V')$. A *$k$-wise finite coloring of* $\mathbb{N}$ is a collection of k-colorings $\texttt{col}_1, \texttt{col}_2, \ldots, \texttt{col}_k$. Note that each coloring $\texttt{col}_1, \ldots, \texttt{col}_k$ is defined over subsets of different sizes. An infinite subset $R \subseteq \mathbb{N}$ is *k-wise monochromatic* with respect to $col_1, \ldots, col_k$ if R is monochromatic with respect to all $\texttt{col}_i$ for $i \in [k]$.

We use the following variant of Ramsey's theorem which was also used in **??**.

Theorem 3.1.7 (Theorem 2.3 in **?**)**.** *For any k-wise finite coloring of $\mathbb{N}$, there exists an infinite k-wise monochromatic subset $R \subseteq \mathbb{N}$.*

Proof of Theorem 3.1.3. Suppose there exists a (t, ε, δ)-tester for property $\mathcal{P}$ of functions $f : D \to \mathbb{N}$. By Lemma 3.1.5, there exists a $(t, \varepsilon, 2\delta)$-discretized tester $\mathcal{T}$ for $\mathcal{P}$. Special name for family of q functions? Let q be the family of probability functions that characterizes $\mathcal{T}$.

We define a t-wise finite coloring of $\mathbb{N}$. For each $r \in [t]$ and $V \in \mathbb{N}^{(r)}$, the color $\texttt{col}_r(V)$ is defined as a vector of probability values $q^y_{X,\rho}(V)$. The vector is indexed by (y, X, ρ) for each $y \in D \cup \{\texttt{accept}, \texttt{reject}\}, s$ satisfying $r \leq s \leq t$ and $X \in D^s$ and surjection $\rho : [r] \to [s]$. The value at the index (y, X, ρ) in $\texttt{col}_r(V)$ is equal to $q^y_{X,\rho}(V)$. Note that, there are finitely many possible values for y and X, and surjections ρ. So, the dimension of the vector $\texttt{col}_r(V)$ is finite. Furthermore, since the tester is discretized, the number of different values that the q-functions take is also finite. Hence, the range of $\texttt{col}_r$ is finite. Now, we have a t-wise finite coloring $\texttt{col}_1, \ldots, \texttt{col}_t$ of $\mathbb{N}$. By Theorem 3.1.7, there exists an infinite t-wise monochromatic set $R \subseteq \mathbb{N}$. Thus, for each $r \in [t]$ and $V, V' \in R^{(r)}$, we have $\texttt{col}_r(V) = \texttt{col}_r(V')$, implying that $q^y_{X,\rho}(V) = q^y_{X,\rho}(V')$ for all y, X, ρ. Thus, $\mathcal{T}$ is comparison-based for functions $f : D \to R$.

Consider a strictly monotone increasing map $\phi : \mathbb{N} \to R$. Given any function $f : D \to \mathbb{N}$, consider $\phi \circ f : D \to R$. Define an algorithm $\mathcal{T}'$, which on input f, runs $\mathcal{T}$ on $\phi \circ f$. Since $\mathcal{P}$ is order-based, $\text{dist}(f, \mathcal{P}) = \text{dist}(\phi \circ f, \mathcal{P})$. Hence, $\mathcal{T}'$ is a $(t, \varepsilon, 2\delta)$-tester for $\mathcal{P}$. Moreover, since the tester T' just runs T on a input $\phi \circ f : D \to R$ as a subroutine and $\mathcal{T}$ is comparison-based for that input, the tester $\mathcal{T}'$ is also comparison-based. □

3.1.2 The Hard Distributions

Our main lower bound theorem is stated next. Together with Theorem 3.1.3, it implies Theorem 1.4.2.

Theorem 3.1.8. *Any nonadaptive comparison-based unateness tester of functions* $f : \{0,1\}^d \to \mathbb{R}$ *must make* $\Omega(d \log d)$ *queries.*

The proof of Theorem 3.1.8 is presented in Sections 3.1.2-3.1.5 and forms the core technical content of this work.

By Theorem 3.1.3 and Yao's minimax principle **?**, it suffices to prove the lower bound for deterministic, nonadaptive, comparison-based testers over a known distribution of functions. It may be useful for the reader to recall the sketch of the main ideas given in Section 1.4.1. For convenience, assume d is a power of 2 and let $d' := d + \log_2 d$. We will focus on functions $h : \{0,1\}^{d'} \to \mathbb{R}$, and prove the lower bound of $\Omega(d \log d)$ for this class of functions, as $\Omega(d \log d) = \Omega(d' \log d')$.

We partition $\{0,1\}^{d'}$ into d subcubes based on the most significant $\log_2 d$ bits. Specifically, for $i \in [d]$, the i^{th} subcube is defined as

$$C_i := \{x \in \{0,1\}^{d'} : \mathsf{val}(x_{d'} x_{d'-1} \cdots x_{d+1}) = i - 1\},$$

where $\mathsf{val}(z) := \sum_{i=1}^{p} z_i 2^{i-1}$ denotes the integer equivalent of the binary string $z_p z_{p-1} \ldots z_1$.

Let $m = d$. We denote the set of indices of the subcube by $[m]$ and the set of dimensions by $[d]$. We use $i, j \in [m]$ to index subcubes, and $a, b \in [d]$ to index dimensions. We now define a series of random variables, where each subsequent variable may depend on the previous ones.

- k: a number picked uniformly at random from $\left[\frac{1}{2}\log_2 d\right]$.
- R: a uniformly random subset of $[d]$ of size 2^k.
- r_i: for each $i \in [m]$, r_i is picked from R uniformly and independently at random.
- α_b: for each $b \in [d]$, α_b is picked from $\{-1, +1\}$ uniformly and independently at random. (Note: α_b only needs to be defined for each $b \in R$. We define it over $[d]$ just so that it is independent of R.)
- β_i: for each $i \in [m]$, β_i is picked from $\{-1, +1\}$ uniformly and independently at random.

We denote the tuple $(k, R, \{r_i\})$ by $\boldsymbol{S}$, also referred to as the *shared randomness*. We use $\boldsymbol{T}$ to refer to the entire set of random variables $(k, R, \{r_i\}, \{\alpha_b\}, \{\beta_i\})$. Given $\boldsymbol{T}$, define the

functions

$$f_{\boldsymbol{T}}(x) := \sum_{b\in[d']\setminus R} x_b 3^b + \alpha_{r_i} x_{r_i} 3^{r_i},$$
$$g_{\boldsymbol{T}}(x) := \sum_{b\in[d']\setminus R} x_b 3^b + \beta_i x_{r_i} 3^{r_i}$$

where i is the subcube containing x, i.e., $i = \mathsf{val}(x_{d'} x_{d'-1} \cdots x_{d+1}) + 1$. The distributions **Yes** and **No** generate $f_{\boldsymbol{T}}$ and $g_{\boldsymbol{T}}$, respectively.

In all cases, the function restricted to any subcube C_i is linear. Consider some dimension $b \in R$. There can be several $i \in [m]$ such that $r_i = b$. For $f_{\boldsymbol{T}}$, in all of these subcubes, the coefficient of x_{r_i} has the same sign, namely α_{r_i}. For $g_{\boldsymbol{T}}$, the coefficient β_i is potentially different, as it depends on the actual subcube.
We write $f \sim \mathcal{D}$ to denote that f is sampled from distribution $\mathcal{D}$.

Claim 3.1.9. *Every function $f \sim$ **Yes** is unate.*

Proof. Fix some $f \in \mathsf{supp}(\mathbf{Yes})$. Since $f_{|C_i}$ is linear, it suffices to argue that, for any $b \in [d']$, the coefficient of x_b (when it is non-zero) has the same sign in all subcubes. When $b \in [d'] \setminus R$, the coefficient of x_b is always 3^b. If $b \in R$, then the coefficient is either 0 or $3^b \alpha_b$. □

Claim 3.1.10. *A function $g \sim$ **No** is $\frac{1}{8}$-far from unate with probability at least* 9/10.

Proof. Note that $|R| \le \sqrt{d}$. For any $r \in R$, let $A_r := \{i : r_i = r\}$, the set of subcube indices with $r_i = r$. Observe that $\mathbf{E}[|A_r|] \ge m/\sqrt{d} = \sqrt{d}$. By Chernoff bound and union bound, for all $r \in R$, we have $|A_r| \ge \sqrt{d}/2$ with probability at least $1 - d\exp(-\sqrt{d}/8)$.

Condition on the event that $|A_r| \ge \sqrt{d}/2$ for all $r \in R$. For each $i \in A_r$, there is a random choice of β_i. Partition A_r into A_r^+ and A_r^-, depending on whether β_i is $+1$ or -1, respectively. Again, by a Chernoff bound and union bound, for all $r \in R$, we have $\min(|A_r^+|, |A_r^-|) \ge |A_r|/4$ with probability at least $1 - d\exp(-\sqrt{d}/32)$. Thus, we can assume that the event $\min(|A_r^+|, |A_r^-|) \ge |A_r|/4$ holds with probability at least $1 - d(\exp(-\sqrt{d}/8) + \exp(-\sqrt{d}/32))$, which is at least 9/10, for large enough d and for any choice of k and R.

Denote the size of any subcube C_i by s. In $g_{\boldsymbol{T}}$, for all $i \in A_r^+$, all r-edges in C_i are increasing, whereas, for all $j \in A_r^-$, all r-edges in C_j are decreasing. To make $g_{\boldsymbol{T}}$ unate, all these edges must have the same direction (i.e., increasing or decreasing). This requires modifying at least $\frac{s}{2} \cdot \min(|A_r^+|, |A_r^-|) \ge \frac{s|A_r|}{8}$ values in $g_{\boldsymbol{T}}$. Summing over all r, we need to

change at least $\frac{s}{8}\sum_r |A_r|$ values. Since the A_r's partition the set of subcubes, this corresponds to at least a $\frac{1}{8}$-fraction of the domain. □

3.1.3 From Functions to Signed Graphs that are Hard to Distinguish

For convenience, denote $x \prec y$ if $\mathsf{val}(x) < \mathsf{val}(y)$. Note that $\prec$ forms a total ordering on $\{0,1\}^{d'}$. Given $x \prec y \in \{0,1\}^{d'}$ and a function $h : \{0,1\}^{d'} \to \mathbb{R}$, define $\mathsf{sgn}_h(x,y)$ to be 1 if $h(x) < h(y)$, 0 if $h(x) = h(y)$, and -1 if $h(x) > h(y)$.

Any deterministic, nonadaptive, comparison-based tester is defined as follows: It makes a set of queries Q and decides whether or not the input function h is unate depending on the $\binom{|Q|}{2}$-comparisons in Q. More precisely, for every pair $(x,y) \in Q \times Q$, $x \prec y$, we insert an edge labeled with $\mathsf{sgn}_h(x,y)$. Let this signed graph be called G_h^Q. Any nonadaptive, comparison-based algorithm can be described as a method to partition the universe of all signed graphs over Q into $\mathcal{G}_Y$ and $\mathcal{G}_N$. The algorithm accepts the function h iff $G_h^Q \in \mathcal{G}_Y$.

Let $\boldsymbol{G}_Y^Q$ be the distribution of the signed graphs G_h^Q when $h \sim \mathbf{Yes}$. Similarly, define $\boldsymbol{G}_N^Q$ when $h \sim \mathbf{No}$. Our main technical theorem is Theorem 3.1.11, which is proved in Section 3.1.4.

Theorem 3.1.11. *For small enough $\delta > 0$ and large enough d, if $|Q| \le \delta d \log d$, then $\|\boldsymbol{G}_Y^Q - \boldsymbol{G}_N^Q\|_{\mathrm{TV}} = O(\delta)$.*

We now prove that Theorem 3.1.11 implies Theorem 3.1.8, the main lower bound.

Proof of Theorem 3.1.8. Consider the distribution over functions where with probability 1/2, we sample from **Yes** and with the remaining probability we sample from **No**. By Theorem 3.1.3 and Yao's minimax principle, it suffices to prove that any deterministic, nonadaptive, comparison-based tester making at most $\delta d \log d$ queries (for small enough $\delta > 0$) errs with probability at least 1/3. Now, note that

$$\Pr[\text{error}] = \frac{1}{2} \cdot \Pr_{h \sim \mathbf{Yes}}[G_h^Q \in \mathcal{G}_N] + \frac{1}{2} \cdot \Pr_{h \sim \mathbf{No}}[G_h^Q \in \mathcal{G}_Y \text{ and } h \text{ is } \frac{1}{8}\text{-far from unate}].$$

By Theorem 3.1.11, the first term is at least $\frac{1}{2} \cdot \left(\Pr_{h \sim \mathbf{No}}[G_h^Q \in \mathcal{G}_N] - O(\delta)\right)$, and by Claim 3.1.10, the second term is at least $\frac{1}{2} \cdot \left(\Pr_{h \sim \mathbf{Yes}}[G_h^Q \in \mathcal{G}_Y] - O(\delta) - \frac{1}{10}\right)$. Summing them up, we get $\Pr[\text{error}] \ge \frac{1}{2} - O(\delta) - \frac{1}{20}$ which is at least $\frac{1}{3}$ for small enough δ. □

The proof of Theorem 3.1.11 is naturally tied to the behavior of sgn_h. Ideally, we would like to say that $\mathsf{sgn}_h(x,y)$ is almost identical regardless of whether $h \sim \mathbf{Yes}$ or $h \sim \mathbf{No}$.

Towards this, we determine exactly the set of pairs (x, y) that potentially differentiate **Yes** and **No**.

Claim 3.1.12. *For all $h \in \mathsf{supp}(\mathbf{Yes}) \cup \mathsf{supp}(\mathbf{No}), x \in C_i$ and $y \in C_j$ such that $i < j$, we have $\mathsf{sgn}_h(x, y) = 1$.*

Proof. For any h, we can write $h(x)$ as $\sum_{b>d} 3^b \cdot x_b + \sum_{b \le d} c_b(x) \cdot 3^b \cdot x_b$, where $c_b : \{0,1\}^{d'} \to \{-1, 0, +1\}$. Thus, $h(y) - h(x) = \sum_{b>d} 3^b (y_b - x_b) + \sum_{b \le d} 3^b (c_b(y) \cdot y_b - c_b(x) \cdot x_b)$. Recall that $x \in C_i, y \in C_j$, and $j > i$. Let q denote the most significant bit of difference between x and y. We have $q > d$, and $y_q = 1$ and $x_q = 0$. Note that for $b \le d$, $|c_b(y) \cdot y_b - c_b(x) \cdot x_b)| \le 2$. Thus, $h(y) - h(x) \ge 3^q - 2\sum_{b<q} 3^b > 0$. □

Thus, comparisons between points in different subcubes reveal no information about which distribution h was generated from. Therefore, the "interesting" pairs that can distinguish whether $h \sim \mathbf{Yes}$ or $h \sim \mathbf{No}$ must lie in the same subcube. The next claim shows a further criterion that is needed for a pair to be interesting. We first define another notation needed for the claim.

Definition 3.1.13. *For any setting of the shared randomness $\boldsymbol{S}$, subcube C_i, and points $x, y \in C_i$, we define $t^i_{\boldsymbol{S}}(x, y)$ to be the most significant coordinate of difference (between x, y) in $([d] \setminus R) \cup \{r_i\}$.*

Note that $\boldsymbol{S}$ determines R and $\{r_i\}$. For any $\boldsymbol{T}$ that extends $\boldsymbol{S}$, the restriction of both $f_{\boldsymbol{T}}$ and $g_{\boldsymbol{T}}$ to C_i is unaffected by coordinates in $R \setminus \{r_i\}$. Thus, $t^i_{\boldsymbol{S}}(x, y)$ is the first coordinate of difference that is influential in C_i.

Claim 3.1.14. *Fix some $\boldsymbol{S}$, subcube C_i, and points $x, y \in C_i$. Let $c = t^i_{\boldsymbol{S}}(x, y)$, and assume $x \prec y$. For any $\boldsymbol{T}$ that extends $\boldsymbol{S}$:*

- *If $c \neq r_i$, then $\mathsf{sgn}_{f_{\boldsymbol{T}}}(x, y) = \mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y) = 1$.*
- *If $c = r_i$, $\mathsf{sgn}_{f_{\boldsymbol{T}}}(x, y) = \alpha_c$ and $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y) = \beta_i$.*

Proof. Assume $x \in C_i$. Recall that $f_{\boldsymbol{T}}(x) = \sum_{b \in [d'] \setminus R} x_b 3^b + \alpha_{r_i} \cdot x_{r_i} 3^{r_i}$ and $g_{\boldsymbol{T}}(x) = \sum_{b \in [d'] \setminus R} x_b 3^b + \beta_i \cdot x_{r_i} 3^{r_i}$.

First, consider the case $c \neq r_i$. Thus, $c \notin R$. Observe that $x_b = y_b$, for all $b > c$ such that $b \notin R$. Furthermore, $x_c = 0$ and $y_c = 1$. Thus, $f_{\boldsymbol{T}}(y) - f_{\boldsymbol{T}}(x) > 3^c - \sum_{b<c} 3^b > 0$. An identical argument holds for $g_{\boldsymbol{T}}$.

Now, consider the case $c = r_i$. Thus, $f_{\boldsymbol{T}}(y) - f_{\boldsymbol{T}}(x) = \alpha_c 3^c + \sum_{b<c, b \notin R} (y_b - x_b) 3^b$. Using the same geometric series arguments as above, $\mathsf{sgn}_{f_{\boldsymbol{T}}}(x, y) = \alpha_c$. By an analogous argument, we can show that $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y) = \beta_i$. □

3.1.4 Proving Theorem 3.1.11: Good and Bad Events

For a given set of queries Q, we first identify certain "bad" values for $\boldsymbol{S}$, on which Q could potentially distinguish between $f_{\boldsymbol{T}}$ and $g_{\boldsymbol{T}}$ for any $\boldsymbol{T}$ that extends $\boldsymbol{S}$. We will prove that the probability of a bad $\boldsymbol{S}$ is small for a given Q. Furthermore, we show that Q cannot distinguish between $f_{\boldsymbol{T}}$ and $g_{\boldsymbol{T}}$ for any $\boldsymbol{T}$ that extends good $\boldsymbol{S}$. First, we set up some definitions.

Definition 3.1.15. *Given a pair (x, y), define $\mathsf{cap}(x, y)$ to be the 5 most significant coordinates*[1] *in which they differ. We say (x, y)* captures *these coordinates. For any set of points $S \subseteq \{0,1\}^{d'}$, define $\mathsf{cap}(S) := \bigcup_{x,y \in S} \mathsf{cap}(x, y)$ to be the coordinates captured by the set S.*

Fix any Q. We set $Q_i := Q \cap C_i$. We define two bad events for $\boldsymbol{S}$.

- Abort Event $\mathcal{A}$: There exist $x, y \in Q$ with $\mathsf{cap}(x, y) \subseteq R$.
- Collision Event $\mathcal{C}$: There exist $i, j \in [d]$ with $r_i = r_j$, such that $r_i \in \mathsf{cap}(Q_i)$ and $r_j \in \mathsf{cap}(Q_j)$.

If $\mathcal{A}$ does not occur, then for any pair (x, y), the sign $\mathsf{sgn}_h(x, y)$ is determined by $\mathsf{cap}(x, y)$ for any $h \in \mathsf{supp}(\mathbf{Yes}) \cup \mathsf{supp}(\mathbf{No})$. The heart of the analysis lies in Theorem 3.1.16, which states that the bad events happen rarely. Theorem 3.1.16 is proved in Section 3.1.5.

Theorem 3.1.16. *If $|Q| \leq \delta d \log d$, then $\Pr[\mathcal{A} \cup \mathcal{C}] = O(\delta)$.*

When neither the abort nor the collision events happen, we say $\boldsymbol{S}$ is good for Q. Next, we show that conditioned on a good $\boldsymbol{S}$, the set Q cannot distinguish $f \sim \mathbf{Yes}$ from $g \sim \mathbf{No}$.

Lemma 3.1.17. *For any signed graph G over Q,*

$$\Pr_{f \sim \mathbf{Yes}}[G_f^Q = G | \boldsymbol{S} \text{ is good}] = \Pr_{g \sim \mathbf{No}}[G_g^Q = G | \boldsymbol{S} \text{ is good}].$$

Proof. We first describe the high level ideas in the proof. As stated above, when the abort event does not happen, the sign $\mathsf{sgn}_h(x, y)$ is determined by $\mathsf{cap}(x, y)$ for any $h \in \mathsf{supp}(\mathbf{Yes}) \cup \mathsf{supp}(\mathbf{No})$. Furthermore, a pair (x, y) has a possibility of distinguishing (that is, the pair is interesting) only if $x, y \in C_i$ and $r_i \in \mathsf{cap}(x, y)$. Focus on such interesting pairs. For such a pair, both $\mathsf{sgn}_{f_{\boldsymbol{T}}}(x, y)$ and $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y)$ are equally likely to be $+1$ or -1. Therefore, to distinguish, we would need two interesting pairs, $(x, y) \in C_i$ and $(x', y') \in C_j$

[1] There is nothing special about the constant 5. It just needs to be sufficiently large.

with $i \neq j$. Note that, when $g \sim \mathbf{No}$, the signs $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y)$ and $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x', y')$ are independently set, whereas when $f \sim \mathbf{Yes}$, the signs are either the same when $r_i = r_j$, or independently set. But if the collision event does not occur, then $r_i \neq r_j$ for interesting pairs in different subcubes. Therefore, the probabilities are the same.

Now, we prove the lemma formally. Condition on a good $\boldsymbol{S}$. Note that the probability of the **Yes** distribution depends solely on $\{\alpha_b\}$ and that of the **No** distribution depends solely on $\{\beta_i\}$.

Consider any pair $(x, y) \in Q \times Q$ with $x \prec y$. We can classify it into three types: (i) x and y are in different subcubes, (ii) x and y are both in the same subcube C_i, and $t^i_{\boldsymbol{S}}(x, y) \neq r_i$, (iii) x and y are both in C_i, and $t^i_{\boldsymbol{S}}(x, y) = r_i$. For convenience, we refer to the third type as *interesting pairs*. Let $h \in \mathsf{supp}(\mathbf{Yes}|\boldsymbol{S}) \cup \mathsf{supp}(\mathbf{No}|\boldsymbol{S})$. For the first and second types of pairs, by Claim 3.1.12 and Claim 3.1.14, we have $\mathsf{sgn}_h(x, y) = 1$. For interesting pairs, by Claim 3.1.14, $\mathsf{sgn}_h(x, y)$ must have the same label for all pairs in $Q_i \times Q_i$. Thus, any G whose labels disagree with the above can never be G^Q_f or G^Q_g.

Fix a signed graph G. For any pair $(x, y) \in Q \times Q$, where $x \prec y$, let $w(x, y)$ be the label in G. Furthermore, for all interesting pairs within the same Q_i, the label $w(x, y)$ is the same and denoted by w_i. Let I denote the set of subcubes with interesting pairs. At this point, all of our discussion depends purely on $\boldsymbol{S}$ and involves no randomness.

Now we focus on $g \sim (\mathbf{No}|\boldsymbol{S})$.

$$\begin{aligned}
\Pr_{g\sim(\mathbf{No}|\boldsymbol{S})}[G^Q_g = G] &= \Pr\Bigg[\bigwedge_{i\in I}\bigwedge_{\substack{x,y\in Q_i\\ t^i_{\boldsymbol{S}}(x,y)=r_i}} (w(x, y) = \mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y))\Bigg] \\
&= \Pr\Bigg[\bigwedge_{i\in I}\bigwedge_{\substack{x,y\in Q_i\\ t^i_{\boldsymbol{S}}(x,y)=r_i}} (w(x, y) = \beta_i)\Bigg] \quad \text{(by Claim 3.1.14)} \\
&= \Pr\Bigg[\bigwedge_{i\in I}(w_i = \beta_i)\Bigg].
\end{aligned}$$

Observe that each β_i is chosen uniformly and independently at random from $\{-1, +1\}$, and so this probability is exactly $2^{-|I|}$.

The analogous expression for $f \sim (\mathbf{Yes}|\boldsymbol{S})$ yields

$$\Pr_{f\sim(\mathbf{Yes}|\boldsymbol{S})}[G^Q_f = G] = \Pr\Bigg[\bigwedge_{i\in I}(w_i = \alpha_{r_i})\Bigg].$$

Notice that if multiple r_i's are the same, then the individual events are not independent over

different subcubes. This is precisely what the abort and collision events capture. We formally argue below.

Consider an interesting pair $(x, y) \in Q_i \times Q_i$. Since the abort event $\mathcal{A}$ does not happen, $\mathtt{cap}(x, y) \not\subseteq R$. If $t^i_{\boldsymbol{S}}(x, y) = r_i \notin \mathtt{cap}(x, y)$, then there is a coordinate of $\overline{R}$ that is more significant than $t^i_{\boldsymbol{S}}(x, y)$. This contradicts the definition of the latter; so $r_i \in \mathtt{cap}(x, y) \subseteq \mathtt{cap}(Q_i)$. Equivalently, a subcube index $i \in I$ iff $r_i \in \mathtt{cap}(Q_i)$.

Since the collision event $\mathcal{C}$ does not happen, for any $j \in [m]$ such that $r_j = r_i$, we have $r_j \notin \mathtt{cap}(Q_j)$. Alternately, for any $i, i' \in I$, we have $r_i \neq r_{i'}$. Thus, $\Pr[\bigwedge_{i \in I}(w_i = \alpha_{r_i})] = \prod_{i \in I} \Pr[w_i = \alpha_{r_i}] = 2^{-|I|}$, completing the proof of the lemma. □

Now, we are armed to prove Theorem 3.1.11.

Proof of Theorem 3.1.11. Given any subset of signed graphs, $\mathcal{G}$, it suffices to upper bound

$$\left|\Pr_{f \sim \mathbf{Yes}}[G^Q_f \in \mathcal{G}] - \Pr_{f \sim \mathbf{No}}[G^Q_f \in \mathcal{G}]\right| \leq \sum_{\text{good } \boldsymbol{S}} \left|\Pr[\boldsymbol{S}] \cdot \left(\Pr_{f \sim \mathbf{Yes}}[G^Q_f \in \mathcal{G}|\boldsymbol{S}] - \Pr_{f \sim \mathbf{No}}[G^Q_f \in \mathcal{G}|\boldsymbol{S}]\right)\right| + \sum_{\text{bad } \boldsymbol{S}} \left|\Pr[\boldsymbol{S}] \cdot \left(\Pr_{f \sim \mathbf{Yes}}[G^Q_f \in \mathcal{G}|\boldsymbol{S}] - \Pr_{f \sim \mathbf{No}}[G^Q_f \in \mathcal{G}|\boldsymbol{S}]\right)\right|.$$

The first term of the RHS is 0 by Lemma 3.1.17. The second term is at most the probability of bad events, which is $O(\delta)$ by Theorem 3.1.16. □

3.1.5 Bounding the Probability of Bad Events: Proof of Theorem 3.1.16

We prove Theorem 3.1.16 by individually bounding $\Pr[\mathcal{A}]$ and $\Pr[\mathcal{C}]$.

Lemma 3.1.18. *If $|Q| \leq \delta d \log d$, then $\Pr[\mathcal{A}] \leq d^{-1/4}$.*

Proof. Fix any choice of k (in $\boldsymbol{S}$). For any pair of points $x, y \in Q$, we have $\Pr[\mathtt{cap}(x, y) \subseteq R] \leq (\frac{2^k}{d-5})^5$. Since $d - 5 \geq d/2$ for all $d \geq 10$ and $k \leq (\log_2 d)/2$, the probability is at most $32d^{-5/2}$. By a union bound, $\Pr[\mathcal{A}] \leq |Q \times Q| \cdot 32d^{-5/2} \leq d^{-1/4}$ for a large enough d. □

The most challenging part of this work is bounding the probability of the collision event, which forms the heart of the lower bound. We start by showing that, if each Q_i captures few coordinates, then the collision event has low probability. A critical point is the appearance of $d \log d$ in this bound.

Lemma 3.1.19. *If $\sum_i |\mathtt{cap}(Q_i)| \leq M$, then $\Pr[\mathcal{C}] = O\left(\frac{M}{d \log d}\right)$.*

Proof. For any $r \in [d]$, define $A_r := \{j : r \in \mathtt{cap}(Q_j)\}$ to be the set of indices of Q_j's that capture coordinate r. Let $a_r := |A_r|$. Define $n_\ell := |\{r : a_r \in (2^{\ell-1}, 2^\ell]\}|$. Observe that $\sum_{\ell \le \log_2 d} n_\ell 2^\ell \le 2 \sum_{r \in [d]} a_r \le 2M$.

Fix k. For $r \in [d]$, we say the event $\mathcal{C}_r$ occurs if (a) $r \in R$, and (b) there exists $i, j \in [d]$ such that $r_i = r_j = r$, and $r_i \in \mathtt{cap}(Q_i)$ and $r_j \in \mathtt{cap}(Q_j)$. By the union bound, $\Pr[\mathcal{C}|k] \le \sum_{r=1}^{d} \Pr[\mathcal{C}_r|k]$.

Now, we compute $\Pr[\mathcal{C}_r|k]$. Only sets Q_j's with $j \in A_r$ are of interest, since the others do not capture r. Event $\mathcal{C}_r$ occurs if at least two of these sets have $r_i = r_j = r$. Hence,

$$\begin{aligned}\Pr[\mathcal{C}_r|k] &= \Pr[r \in R] \cdot \Pr[\exists i, j \in A_r : r_i = r_j = r \mid r \in R] \\ &= \frac{2^k}{d} \cdot \sum_{c \ge 2} \binom{a_r}{c} \left(\frac{1}{2^k}\right)^c \left(1 - \frac{1}{2^k}\right)^{a_r - c}. \end{aligned} \tag{3.1}$$

A fixed r is in R with probability $\binom{d-1}{2^k-1}/\binom{d}{2^k} = \frac{2^k}{d}$. Given that $|R| = 2^k$, the probability that $r_i = r$ is precisely 2^{-k}.

If $a_r \ge \frac{2^k}{4}$, then we simply upper bound (3.1) by $\frac{2^k}{d}$. For $a_r < \frac{2^k}{4}$, we upper bound (3.1) by

$$\frac{2^k}{d}\left(1 - \frac{1}{2^k}\right)^{a_r} \sum_{c \ge 2} \left(a_r \cdot \frac{1}{2^k} \cdot \left(1 - \frac{1}{2^k}\right)^{-1}\right)^c \le \frac{2^k}{d} \sum_{c \ge 2} \left(\frac{a_r}{2^{k-1}}\right)^c \le \frac{8a_r^2}{2^k d}.$$

Summing over all r and grouping according to n_ℓ, we get

$$\Pr[\mathcal{C}|k] \le \sum_{r=1}^{d} \Pr[\mathcal{C}_r|k] \le \sum_{r : a_r \ge 2^{k-2}} \frac{2^k}{d} + \frac{8}{d} \sum_{r : a_r < 2^{k-2}} \frac{a_r^2}{2^k} \le \frac{2^k}{d} \sum_{\ell > k-2} n_\ell + \frac{8}{d} \sum_{\ell=1}^{k-2} n_\ell 2^{2\ell - k}.$$

Averaging over all k, we get

$$\begin{aligned}\Pr[\mathcal{C}] = \frac{2}{\log_2 d} \sum_{k=1}^{(\log_2 d)/2} \Pr[\mathcal{C}|k] \quad &\le \quad \frac{16}{d \log_2 d} \sum_{k=1}^{(\log_2 d)/2} \left(\sum_{\ell=1}^{k-2} n_\ell 2^{2\ell - k} + \sum_{\ell > k-2} n_\ell 2^k\right) \\ &= \frac{16}{d \log_2 d} \left(\sum_{\ell=1}^{(\log_2 d)/2} n_\ell \sum_{k \ge \ell+2} 2^{2\ell - k} + \sum_{\ell=1}^{\log_2 d} n_\ell \sum_{k < \ell+2} 2^k\right). \end{aligned} \tag{3.2}$$

Now, $\sum_{k \ge \ell+2} 2^{2\ell-k} \le 2^\ell$ and $\sum_{k < \ell+2} 2^k \le 4 \cdot 2^\ell$. Substituting, $\Pr[\mathcal{C}] \le \frac{80}{d \log_2 d} \sum_{\ell=1}^{\log_2 d} n_\ell 2^\ell \le \frac{160M}{d \log_2 d}$, proving the lemma. □

We are now left to bound $\sum_i |\mathtt{cap}(Q_i)|$. This is done by the following combinatorial lemma.

Lemma 3.1.20. *Let V be a set of vectors over an arbitrary alphabet and any number of dimensions. For any natural number c and $x, y \in V$, let $\texttt{cap}_c(x, y)$ denote the (set of) first c coordinates at which x and y differ. Then $|\texttt{cap}_c(V)| \leq c(|V| - 1)$.*

Proof. We construct c different edge-colored graphs $G_1, \ldots, G_c$ over the vertex set V. For every coordinate $i \in \texttt{cap}_c(V)$, there must exist at least one pair of vectors x, y such that $i \in \texttt{cap}_c(x, y)$. Thinking of each $\texttt{cap}_c(x, y)$ as an ordered set, find a pair (x, y) where i appears "earliest" in $\texttt{cap}_c(x, y)$. Let the position of i in this $\texttt{cap}_c(x, y)$ be denoted by t. We add edge (x, y) to G_t, and color it i. Note that the same edge (x, y) cannot be added to G_t with multiple colors, and hence all G_t's are simple graphs. Furthermore, observe that each color is present only once over all G_t's.

We claim that each G_t is acyclic. Suppose not. Let there be a cycle C and let (x, y) be the edge in C with the smallest color i. Clearly, $x_i \neq y_i$ since $i \in \texttt{cap}_c(x, y)$. There must exist another edge (u, v) in C such that $u_i \neq v_i$. Furthermore, the color of (u, v) is $j > i$. Thus, j is the t^{th} entry in $\texttt{cap}_c(u, v)$. Note that $i \in \texttt{cap}_c(u, v)$ and must be the s^{th} entry for some $s < t$. But this means that the edge (u, v) colored i should be in G_s, contradicting the presence of $(x, y) \in G_t$. □

We wrap up the bound now.

Lemma 3.1.21. *If $|Q| \leq \delta d \log d$, then $\Pr[\mathcal{C}] = O(\delta)$.*

Proof. Lemma 3.1.20 applied to each Q_i, yields $\sum_i |\texttt{cap}(Q_i)| \leq 5|Q_i| = 5|Q|$. An application of Lemma 3.1.19 completes the proof. □

Chapter 4
Conclusion

4.1 Conclusion and Open Directions

In this work, we give the first algorithms for testing unateness of real-valued functions over the hypercube as well as the hypergrid domains. We also show that our algorithms are optimal by proving matching lower bounds, thus resolving the query complexity of testing unateness of real-valued functions. Our results demonstrate that, for real-valued functions, in contrast to monotonicity testing, adaptivity helps with testing unateness.

The query complexity of testing unateness of Boolean functions has not been completely resolved yet. Concurrent with our work, Chen et al. **?** proved a lower bound of $\Omega\left(\frac{d^{2/3}}{\log^3 d}\right)$ for adaptive unateness testers of Boolean functions over $\{0,1\}^d$. Subsequently, Chen et al. **?** gave an adaptive unateness tester with query complexity $\widetilde{O}\left(\frac{d^{3/4}}{\varepsilon^2}\right)$ for the same class of functions. It remains to be seen if this polynomial gap (in d) between the bounds can be closed further.

On nonadaptive unateness testing of Boolean functions over $\{0,1\}^d$, in a subsequent work, Baleshzar et al. **?** proved a lower bound of $\Omega\left(\frac{d}{\log d}\right)$ for one-sided error testers. Since Boolean functions are special cases of real-valued functions, our nonadaptive algorithm over the hypercube also works for Boolean functions. The query complexity of this algorithm is $O\left(\frac{d \log d}{\varepsilon}\right)$ which is currently the best known upper bound. An interesting open question is to determine if testers with two-sided error have better query complexity than testers with one-sided error in the nonadaptive setting.

Appendix
Missing Details from the Main Body

1 The Lower Bound for Adaptive Testers over Hypergrids

We show that every unateness tester for functions $f : [n]^d \mapsto \mathbb{R}$ requires $\Omega\left(\frac{d \log n}{\varepsilon} - \frac{\log 1/\varepsilon}{\varepsilon}\right)$ queries for $\varepsilon \in (0, 1/4)$ and prove Theorem 1.4.3.

Proof of Theorem 1.4.3. By Yao's minimax principle and the reduction to testing with comparison-based testers from **??** (stated for completeness in Theorem 3.1.3), it is sufficient to give a hard input distribution on which every deterministic comparison-based tester fails with probability more than 2/3. We use the hard distribution constructed by Chakrabarty and Seshadhri **?** to prove the same lower bound for testing monotonicity. Their distribution is a mixture of two distributions, **Yes** and **No**, on positive and negative instances, respectively. Positive instances for their problem are functions that are monotone and, therefore, unate; negative instances are functions that are ε-far from monotone. We show that their **No** distribution is supported on functions that are ε-far from unate, i.e., negative instances for our problem. Then the required lower bound for unateness follows from the fact that every deterministic comparison-based tester needs the stated number of queries to distinguish **Yes** and **No** distributions with high enough probability.

We start by describing the **Yes** and **No** distribution used in **?**. We will define them as distributions on functions over the hypercube domain. Next, we explain how to convert functions over hypercubes to functions over hypergrids.

Without loss of generality, assume n is a power of 2 and let $\ell := \log_2 n$. For any $z \in [n]$, let $bin(z)$ denote the binary representation of $z - 1$ as an ℓ-bit vector $(z_1, \ldots, z_\ell)$, where z_1 is the least significant bit.

We now describe the mapping used to convert functions on hypergrids to functions on

hypercubes. Let $\phi : [n]^d \to \{0,1\}^{d\ell}$ be the mapping that takes $y \in [n]^d$ to the concatenation of $bin(y_1), \ldots, bin(y_d)$. Any function $f : \{0,1\}^{d\ell} \mapsto \mathbb{R}$ can be easily converted into a function $\widetilde{f} : [n]^d \mapsto \mathbb{R}$, where $\widetilde{f}(y) := f(\phi(y))$.

Let $m := d\ell$. For $x \in \{0,1\}^m$, let $\texttt{val}(x) = \sum_{i=1}^{m} x_i 2^{i-1}$ denote the value of the binary number represented by vector x. For simplicity, assume $1/\varepsilon$ is a power of 2. Partition the set of points $x \in \{0,1\}^m$ according to the most significant $\log(1/2\varepsilon)$ dimensions. That is, for $k \in \{1, 2, \ldots, 1/2\varepsilon\}$, let

$$S_k := \{x : \texttt{val}(x) \in [(k-1) \cdot \varepsilon 2^{m+1}, k \cdot \varepsilon 2^{m+1} - 1]\}.$$

The hypercube is partitioned into $1/2\varepsilon$ sets S_k of equal size, and each S_k forms a subcube of dimension $m' = m - \log(1/\varepsilon) + 1$.

We now describe the **Yes** and **No** distributions for functions on hypercubes. The **Yes** distribution consists of a single function $f(x) = 2\texttt{val}(x)$. The **No** distribution is uniform over $m'/2\varepsilon$ functions $g_{j,k}$, where $j \in [m']$ and $k \in [1/2\varepsilon]$, defined as follows:

$$g_{j,k}(x) = \begin{cases} 2\texttt{val}(x) - 2^j - 1 & \text{if } x_j = 1 \text{ and } x \in S_k; \\ 2\texttt{val}(x), & \text{otherwise.} \end{cases}$$

To get the **Yes** and **No** distributions for the hypergrid, we convert f to $\widetilde{f}$ and each function $g_{j,k}$ to $\widetilde{g_{j,k}}$, using the transformation defined before.

Chakrabarty and Seshadhri **?** proved that f is monotone and each function $\widetilde{g_{j,k}}$ is ε-far from monotone. It remains to show that functions $\widetilde{g_{j,k}}$ are also ε-far from unate.

Claim 1.1. *Each function $\widetilde{g_{j,k}}$ is ε-far from unate.*

Proof. To prove that $\widetilde{g_{j,k}}$ is ε-far from unate, it suffices to show that there exists a dimension i, such that there are at least $\varepsilon 2^{d\ell}$ increasing i-pairs and at least $\varepsilon 2^{d\ell}$ decreasing i-pairs w.r.t. $\widetilde{g_{j,k}}$ and that all of these i-pairs are disjoint. Let $u, v \in [n]^d$ be two points such that $\phi(u)$ and $\phi(v)$ differ only in the j^{th} bit. Clearly, u and v form an i-pair, where $i = \lceil j/\ell \rceil$. Now, if $\phi(u), \phi(v) \in S_k$ and $u \prec v$, then $\widetilde{g_{j,k}}(v) = \widetilde{g_{j,k}}(u) - 1$. So, the i-pair (u, v) is decreasing. The total number of such i-pairs is $2^{d\ell - \log(1/2\varepsilon) - 1} = \varepsilon 2^{d\ell}$. If $\phi(u), \phi(v) \in S_{k'}$ where $k' \neq k$, then the i-pair (u, v) is increasing. Clearly, there are at least $\varepsilon 2^{d\ell}$ such i-pairs. All the i-pairs we mentioned are disjoint. Hence, $\widetilde{g_{j,k}}$ is ε-far from unate. □

This completes the proof of Theorem 1.4.3. □

2 The Lower Bound for Nonadaptive Testers over Hypergrids

The lower bound for nonadaptive testers over hypergrids follows from a combination of the lower bound for nonadaptive testers over hypercube and the lower bound for adaptive testers over hypergrids.

Theorem 2.1. *Any nonadaptive unateness tester (even with two-sided error) for real-values functions $f : [n]^d \mapsto \mathbb{R}$ must make $\Omega(d(\log n + \log d))$ queries.*

Proof. Fix $\varepsilon = 1/8$. The proof consists of two parts. The lower bound for adaptive testers is also a lower bound for nonadaptive tester, and so, the bound of $\Omega(d \log n)$ holds. Next, we extend the $\Omega(d \log d)$ lower bound for hypercubes. Assume n to be a power of 2. Define function $\psi : [n] \mapsto \{0, 1\}$ as $\psi(a) := \mathbb{1}[a > n/2]$ for $a \in [n]$. For $x = (x_1, x_2, \ldots, x_d) \in [n]^d$, define the mapping $\Psi : [n]^d \mapsto \{0, 1\}^d$ as $\Psi(x) := (\psi(x_1), \psi(x_2), \ldots, \psi(x_d))$. Any function $f : \{0, 1\}^d \mapsto \mathbb{R}$ can be extended to $\tilde{f} : [n]^d \mapsto \mathbb{R}$ using the mapping $\tilde{f}(x) = f(\Psi(x))$ for all $x \in [n]^d$. The proof of Theorem 3.1.8 goes through for hypergrids as well, and so we have an $\Omega(d \log d)$ lower bound. Combining the two lower bounds, we get a bound of $\Omega(d \cdot \max\{\log n, \log d\})$, which is asymptotically equal to $\Omega(d(\log n + \log d))$. □