L_p-Testing

Draft full version

Piotr Berman* Sofya Raskhodnikova† Grigory Yaroslavtsev‡

July 16, 2016

Abstract

We initiate a systematic study of sublinear algorithms for approximately testing properties of real-valued data with respect to L_p distances for $p \geq 1$. Such algorithms distinguish datasets which either have (or are close to having) a certain property from datasets which are far from having it with respect to L_p distance. For applications involving noisy real-valued data, using L_p distances allows algorithms to withstand noise of bounded L_p norm. While the classical property testing framework developed with respect to Hamming distance has been studied extensively, testing with respect to L_p distances has received little attention.

We use our framework to design simple and fast algorithms for classic problems, such as testing monotonicity, convexity and the Lipschitz property, and also distance approximation to monotonicity. In particular, for functions over the hypergrid domains $[n]^d$, the complexity of our algorithms for all these properties does not depend on the linear dimension n. This is impossible in the standard model. Most of our algorithms require minimal assumptions on the choice of sampled data: either uniform or easily samplable random queries suffice. We also show connections between the L_p-testing model and the standard framework of property testing with respect to Hamming distance. Some of our results improve existing bounds for Hamming distance.

1 Introduction

Property testing [54, 36] is a rigorous framework for approximate analysis of global properties of data, which can be performed given access to a small sample. For example, one can approximately verify whether a (possibly multidimensional) array of numbers is sorted by examining a carefully chosen subset of its elements [26, 37, 24]. Formally, a property testing problem can be stated by treating data as a function on the underlying domain. The datasets satisfying a property form a class of functions. E.g., the set of all sorted real-valued n-element datasets corresponds to the class of monotone functions of the form $f\colon [n] \to \mathbb{R}$, while monotone functions $f\colon [n]^d \to \mathbb{R}$ represent d-dimensional arrays with linear size n, sorted in each of the d dimensions[1]. The problem of testing sortedness can be formalized as follows. Let $\mathcal{M}$ be the class of all monotone functions. Given a function $f\colon [n]^d \to \mathbb{R}$ and a proximity parameter ε, we want to decide whether $f \in \mathcal{M}$ or f is at distance at least ε from any function in $\mathcal{M}$. The distance measure in the standard model is (relative) Hamming distance.

In this paper, we initiate a systematic study of properties of functions with respect to L_p distances, where $p > 0$. Property testing was originally introduced for analysis of algebraic properties of functions over finite fields such as linearity and low degree. Attention to these properties was motivated by applications

*Pennsylvania State University, USA; berman@cse.psu.edu.

†Pennsylvania State University, USA and Boston University, USA; sofya@cse.psu.edu.

‡Brown University, Institute for Computational and Experimental Research in Mathematics; grigory@grigory.us.

[1]We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

to Probabilistically Checkable Proofs. In such applications, Hamming distance between two functions is a natural choice because of its interpretation as the probability that the two functions differ on a random point from the domain. Also, many initial results in property testing focused on Boolean functions, for which L_p distances are the same for $p = 0, 1$ and, more generally, are related in a simple way for different values of p, so all choices of p lead to the same testing problems. Subsequently, property testing algorithms have been developed for multiple basic properties of functions over the reals (e.g., monotonicity, convexity, submodularity, the Lipschitz property, etc.). We study testing these properties w.r.t. L_p distances, providing different approximation guarantees, better suited for many applications with real-valued data.

L_p-testing. Let f be a real-valued function over a finite[2] domain D. For $p \geq 1$, the L_p-norm of f is $\|f\|_p = \left(\sum_{x \in D} |f(x)|^p\right)^{1/p}$. For $p = 0$, let $\|f\|_0 = \sum_{x \in D} |f(x)|^0$ be the number of non-zero values of f. Let $\mathbb{1}$ denote the function that evaluates to 1 on all $x \in D$. A *property* $\mathcal{P}$ is a set of functions over D. For real-valued functions $f : D \to [0, 1]$ and a property $\mathcal{P}$, we define relative L_p distance as follows[3]:

$$d_p(f, \mathcal{P}) = \inf_{g \in \mathcal{P}} \frac{\|f - g\|_p}{\|\mathbb{1}\|_p} = \inf_{g \in \mathcal{P}} \left(\mathbb{E}[|f - g|^p]\right)^{1/p},$$

where the first equality holds for $p \geq 0$ and the second for $p > 0$. The normalization by a factor $\|\mathbb{1}\|_p$ ensures that $d_p(f, \mathcal{P}) \in [0, 1]$. For $p \geq 0$, a function f is ε-far from a property $\mathcal{P}$ w.r.t. the L_p distance if $d_p(f, \mathcal{P}) \geq \varepsilon$. Otherwise, f is ε-close to $\mathcal{P}$.

Definition 1.1 (L_p-tester)**.** *An* L_p-tester *for a property $\mathcal{P}$ is a randomized algorithm that, given a proximity parameter $\varepsilon \in (0, 1)$ and oracle access to a function $f \colon \mathcal{D} \to [0, 1]$,*

1. *accepts with probability at least 2/3 if $f \in \mathcal{P}$;*
2. *rejects with probability at least 2/3 if $d_p(f, \mathcal{P}) \geq \varepsilon$.*

The corresponding algorithmic problem is called *L_p-testing*. Standard property testing corresponds to L_0-testing, which we also call *Hamming testing*.

Tolerant L_p-testing and L_p-distance approximation. An important motivation for measuring distances to properties of real-valued functions w.r.t. L_p metrics is noise-tolerance. In order to be able to withstand noise of bounded Hamming weight (small number of outliers) in the property testing framework, Parnas, Ron, and Rubinfeld [48] introduced *tolerant* property testing. One justification for L_p-testing is that in applications involving real-valued data, noise added to the function often has large Hamming weight, but bounded L_p-norm for some $p > 0$ (e.g., Brownian motion, white Gaussian noise, etc.). This leads us to the following definition, which generalizes tolerant testing [48].

Definition 1.2 (Tolerant L_p-tester)**.** *An $(\varepsilon_1, \varepsilon_2)$-tolerant$L_p$-tester for a property $\mathcal{P}$ is a randomized algorithm which, given $\varepsilon_1, \varepsilon_2 \in (0, 1)$, where $\varepsilon_1 < \varepsilon_2$, and oracle access to a function $f \colon \mathcal{D} \to [0, 1]$,*

1. *accepts with probability at least 2/3 if $d_p(f, \mathcal{P}) \leq \varepsilon_1$.*
2. *rejects with probability at least 2/3 $d_p(f, \mathcal{P}) \geq \varepsilon_2$.*

If the tester works for arbitrary $\varepsilon_1 < \varepsilon_2$, it is called fully tolerant.

[2]Some of our results apply to functions over infinite measurable domains, i.e., $\int_{\mathcal{D}} \mathbb{1} < \infty$, where $\mathbb{1}$ is an indicator function of the domain $\mathcal{D}$. This is why we use the name L_p rather than ℓ_p. An important example of such a domain is the hypercube $[0, 1]^d$ in $\mathbb{R}^d$.

[3]The definition of distance d_p can be extended to functions $f \colon D \to [a, b]$, where $a < b$, by changing the normalization factor to $\|\mathbb{1}\|_p \cdot (b - a)$. Our results hold for the more general range (if the algorithms are given bounds a and b), since for the properties we consider (monotonicity, the Lipschitz property and convexity), testing f reduces to testing $f' = \frac{f(x) - a}{b - a}$. For ease of presentation, we set the range to $[0, 1]$.

For example, a tolerant L_1-tester can ignore both uniform noise of bounded magnitude and noise of large magnitude concentrated on a small set of outliers.

A related computational task is *approximating the L_p distance to a property $\mathcal{P}$:* given oracle access to a function f, output an additive approximation to the relative L_p distance $d_p(f, \mathcal{P})$, which has the desired accuracy with probability at least 2/3. Distance approximation is equivalent to tolerant testing (up to small multiplicative factors in the running time) [48]. Both problems were studied extensively for monotonicity [48, 3, 55, 28] and convexity [27]. Despite significant progress, an optimal algorithm is not known even for monotonicity in one dimension. In contrast, for L_1-testing we are able to fully resolve this question for one-dimensional functions.

Connections with learning. Another compelling motivation for L_p-testing comes from learning theory [60]. It has been pointed out that property testing can be helpful in model selection. If the concept class for learning a target function is not known reliably in advance, one can first run more efficient property testing or distance approximation algorithms in order to check multiple candidate concept classes for the subsequent learning step. It is important that the approximation guarantees of the preprocessing step and the learning step be aligned. Because L_p distances are frequently used to measure error in PAC-learning of real-valued functions (see e.g. [41]), an L_p-testing algorithm is a natural fit for preprocessing in such applications, especially when noisy real-valued data is involved. We believe that investigating L_p-testing might be an important step in bridging the gap between existing property testing and learning models.

We note that other property testing models introduced to help bridge that gap (prominent examples include distribution-free testing [36, 39], and passive and active testing [5]) combine well with our distance measures. We leave the investigation of these models w.r.t. L_p distances for future work. We remark, however, that some of our algorithms only take uniform samples from the domain (or make queries according to another easily samplable distribution), so they can also be used in the passive testing model.

We also note that the well established connection between Hamming testing and learning [36, 52] naturally extends to L_p-testing, and we exploit it in our results on monotonicity and convexity (see Section 1.2). Namely, from the information-theoretic perspective, property testing is not harder than PAC-learning (up to a small additive factor), although computationally this holds only for proper learning. Tolerant testing and distance approximation are related in a similar way to agnostic learning [44, 48]. Thus, the goal of property testing is to design algorithms which have significantly lower complexity than the corresponding learning algorithms and even go beyond the lower bounds for learning.

Connections with approximation theory. Connections between learning and property testing are well established. Another closely related field is that of approximation theory. Computational tasks considered in that field (e.g., approximating a class of functions with L_p error) are similar to learning tasks. Basically, the only differences between approximation and learning tasks are that approximation algorithms are usually allowed to query the input function at points of their choice and are non-agnostic (i.e., they work only under the assumption that the input function is in a given class). Approximation theory is not known to imply any interesting results for Hamming property testing primarily because approximation results usually have L_p error with $p > 0$. But once the L_p metric is considered in property testing, the parallels between the computational tasks studied in sublinear algorithms and approximation theory become apparent. In Section 1.2.3, we exploit the connection to approximation theory to get an L_p-testing algorithm for convexity. We believe that a systematic investigation of L_p-testing is likely to yield deep and interesting connections between property testing and approximation theory.

Previous work related to L_p-testing. To the best of our knowledge no prior work systematically studies L_p-testing of properties of functions. The only explicitly stated L_p-testing result for $p > 0$ is the L_1-tester for submodular functions in a recent paper by Feldman and Vondrák [30]. It is a direct corollary of their junta approximation result w.r.t. L_1 distance. The upper bound on the query complexity of L_1-testing of submodularity in [30] is far from the known lower bound. For related classes of functions over the Boolean hypercube such as coverage, XOS and self-bounding functions, L_p-testing algorithms can be derived from

the learning algorithms of [30, 29]. In the Hamming testing model it is well-understood that for classes of functions, which can be approximated by juntas, learning-theoretic bounds can be significantly improved via a generic "testing by implicit learning" framework [22, 56]. It remains open whether the same is true for L_p-testing.

There are also property testing papers that work with L_1 distance, but consider different input access. Rademacher and Vempala [49] study property testing whether a given set S is convex, where ε-far means that there is no convex set K such that $vol(K \Delta S) \leq \varepsilon \cdot vol(S)$. In addition to oracle access, their algorithm can sample a random point form the set S. Finally, L_1 distance is also widely used in testing properties of distributions [7, 61, 59, 9, 21, 19].

1.1 Basic Relationships Between L_p-Testing Models

Our first goal is to establish relationships between Hamming, L_1 and L_p-testing for standard and tolerant models[4]. We stress that even though the basic relations between L_p-testing problems, summarized in Facts 1.1 and 1.2, are stated for query complexity of general algorithms, these facts still hold if the query complexity is defined for a restricted class of algorithms (e.g., nonadaptive and/or with one-sided error, defined next); they also hold for analogous time complexity measures.

Definition 1.3. *An algorithm is called* nonadaptive *if it makes all queries in advance, before receiving any responses; otherwise, it is called* adaptive. *A testing algorithm for property $\mathcal{P}$ has* one-sided error *if it always accepts all inputs in $\mathcal{P}$; otherwise, it has* 2-sided error.

L_p-testing. We denote the worst-case query complexity of L_p-testing for property $\mathcal{P}$ with proximity parameter ε by $Q_p(\mathcal{P}, \varepsilon)$. The following fact establishes basic relationships between L_p-testing problems and follows directly from the inequalities between L_p-norms. A generalization of this fact is proved in Section 5.1.

Fact 1.1. *Let $\varepsilon \in (0,1)$ and $\mathcal{P}$ be a property over any domain. Then*

$$1.\ Q_1(\mathcal{P}, \varepsilon) \leq Q_0(\mathcal{P}, \varepsilon); \quad 2.\ Q_1(\mathcal{P}, \varepsilon) \leq Q_p(\mathcal{P}, \varepsilon) \leq Q_1(\mathcal{P}, \varepsilon^p) \text{ for all } p \geq 1.$$

Moreover, if $\mathcal{P}$ is a property of Boolean functions then

$$1.\ Q_1(\mathcal{P}, \varepsilon) = Q_0(\mathcal{P}, \varepsilon); \quad 2.\ Q_p(\mathcal{P}, \varepsilon) = Q_1(\mathcal{P}, \varepsilon^p) \text{ for all } p \geq 1.$$

This fact has several implications. First, L_1-testing is no harder than standard Hamming testing (the first inequality). Hence, upper bounds on the query complexity of the latter are the baseline for the design of L_1-testing algorithms. As we will demonstrate, for the properties considered in this paper, L_1-testing has significantly smaller query complexity than Hamming testing. This fact also shows that the complexity of L_1 and L_2-testing problems is equivalent up to quadratic dependence on ε (the second and third inequalities). Finally, the equivalence of L_p-testing problems for Boolean functions (and, more generally, equivalence up to constant factors for functions with constant size range) implies that all lower bounds for such functions in the standard Hamming testing model are applicable to L_p-testing.

Tolerant L_p-testing. We denote the worst-case query complexity of $(\varepsilon_1, \varepsilon_2)$-tolerant L_p-testing of a property $\mathcal{P}$ by $Q_p(\mathcal{P}, \varepsilon_1, \varepsilon_2)$. Introducing tolerance complicates relationships between L_p-testing problems for different p as compared to the relationships in Fact 1.1. A generalization of this fact is proved in Section 5.2.

Fact 1.2. *Let $\varepsilon_1, \varepsilon_2 \in (0,1)$ such that $\varepsilon_1 < \varepsilon_2^p$ and $\mathcal{P}$ be a property over any domain. Then*

$$Q_1(\mathcal{P}, \varepsilon_1^p, \varepsilon_2) \leq Q_p(\mathcal{P}, \varepsilon_1, \varepsilon_2) \leq Q_1(\mathcal{P}, \varepsilon_1, \varepsilon_2^p).$$

Moreover, if $\mathcal{P}$ is a property of Boolean functions then

[4] In the rest of the paper, we consider L_p-testing and distance approximation only for $p = 0$ and $p \geq 1$, leaving the remaining cases for future work.

1. $Q_1(\mathcal{P}, \varepsilon_1, \varepsilon_2) = Q_0(\mathcal{P}, \varepsilon_1, \varepsilon_2)$; *2.* $Q_p(\mathcal{P}, \varepsilon_1, \varepsilon_2) = Q_1(\mathcal{P}, \varepsilon_1^p, \varepsilon_2^p)$ *for all* $p \geq 1$.

Facts 1.1-1.2 establish the key role of L_1-testing in understanding property testing w.r.t. L_p distances since results for L_2-testing follow with a minor loss in parameters. Moreover, in many cases, these results turn out to be optimal. Hence, in the rest of this paper we focus primarily on L_1-testing.

1.2 Our Results on Properties of Real-Valued Functions

We consider three properties of real-valued functions: monotonicity, the Lipschitz property and convexity. We focus on understanding the L_1 distance to these properties and obtain results for L_p-distance for $p > 1$ by applying Facts 1.1-1.2. Most of our algorithms are nonadaptive and/or have one-sided error (see Definition 1.3).

1.2.1 Monotonicity

Monotonicity is perhaps the most investigated property in the context of property testing and distance approximation. Monotonicity testing has been extensively studied, with specific focus on the hypergrid domains [37, 26, 24, 31, 6, 40, 2, 11, 13, 15, 16, 12, 17], as well as on general partially ordered domains [32, 10]. Distance approximation to monotone functions has also received significant attention [48, 3, 55, 28, 8].

Definition 1.4 (Monotone function)**.** *Let D be a (finite) domain equipped with a partial order $\preceq$. A function $f : D \to \mathbb{R}$ is* monotone *if $f(x) \leq f(y)$ for all $x, y \in D$ satisfying $x \preceq y$.*

An important specific domain D is a *d-dimensional hypergrid* $[n]^d$ equipped with the partial order $\preceq$, where $(x_1, \ldots, x_n) \preceq (y_1, \ldots, y_n)$ whenever $x_1 \leq y_1, \ldots, x_n \leq y_n$. The special case $[n]$ of the hypergrid is called a *line*, and the special case $[2]^d$ is a *hypercube*. These domains are interesting in their own right. For example, testing monotonicity on the line $[n]$ corresponds to testing whether a list of n numbers is sorted (in nondecreasing order). The L_1 distance to monotonicity on the line is the total change in the numbers required to make them sorted.

Characterization. We give a characterization of the L_1 distance to monotonicity in terms of the distance to monotonicity of Boolean functions (Lemma 2.1). The main idea in our characterization is that every function can be viewed as an integral over Boolean threshold functions. This view allows us to express the L_1 distance to monotonicity of a real-valued function $f \colon D \to [0, 1]$ as an integral over the L_1 distances to monotonicity of its Boolean threshold functions. We use this characterization to obtain reductions from the general monotonicity testing and distance approximation to the case of Boolean functions (Lemmas 2.2 and 2.3) that preserve query complexity and running time[5].

Recall that for Boolean functions, L_0 and L_1 distances are equal. Thus, our reductions allow us to capitalize on existing algorithms and lower bounds for (Hamming) testing of and distance approximation to monotonicity of Boolean functions. For example, for the case of the line domain $[n]$, it is folklore that monotonicity of Boolean functions can be tested nonadaptively and with 1-sided error in $O\left(\frac{1}{\varepsilon}\right)$ time. In contrast, for Hamming testing with the general range, one needs $\Omega\left(\frac{\log n}{\varepsilon} - \frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ queries even for adaptive 2-sided error testers [26, 31, 17]. Therefore, testing monotonicity on the line is a factor of $\log n$ faster w.r.t. the L_1 distance than Hamming distance. A comparison between the query complexity of L_p-testing monotonicity on the line, the hypercube, the hypergrid and general partially ordered (PO) domains for $p = 0$ and $p \geq 1$ is given in Table 1.1. The results for $p \geq 1$ for the line, the hypercube and general PO domains follow from the basic relationships between L_p-testing problems Fact 1.1, our reduction to the Boolean case

[5]Our reductions are stated for nonadaptive algorithms. Such reductions are useful because all known upper bounds for testing monotonicity can be achieved by nondaptive testers, with one exception: our adaptive bound for testing Boolean functions on constant-dimensional hypergrids from Section 2.3. We can get a reduction that works for adaptive algorithms by viewing L_1-testing monotonicity as a multi-input concatenation problem [33]. This reduction preserves the query complexity for the special class of *proximity-oblivious* testers [23], but incurs a loss of $O\left(\frac{1}{\varepsilon}\right)$ in general and, specifically, when applied to our adaptive tester. As this approach would not improve our results, we focus on reductions for nonadaptive algorithms.

Monotonicity		
Domain	Hamming Testing	L_p-Testing for $p \geq 1$
$[n]$	$O\left(\frac{\log n}{\varepsilon}\right)$ n.a. 1-s. [26]	$O\left(\frac{1}{\varepsilon^p}\right)$ n.a. 1-s. Lem. 2.2 + Fact 1.1
	$\Omega\left(\frac{\log n}{\varepsilon} - \frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ a. 2-s. [26, 31, 17]	$\Omega\left(\frac{1}{\varepsilon^p}\right)$ a. 2-s. Fact 1.1
$\{0,1\}^d$	$O\left(\frac{d}{\varepsilon}\right)$ n.a. 1-s. [24, 37]	$O\left(\frac{d^{5/6}}{\varepsilon^{5p/3}}\right)$ n.a. 1-s. [16] + Lem. 2.2 + Fact 1.1
	$\Omega\left(\frac{d}{\varepsilon}\right)$ a. 2-s. [11, 14]	$\Omega(\sqrt{d})$ n.a. 1-s., $\Omega(\log d)$ n.a. 2-s. [32] + Fact 1.1
$[n]^d$	$O\left(\frac{d\log n}{\varepsilon}\right)$ n.a. 1-s. [15]	$O\left(\frac{d}{\varepsilon^p}\log\frac{d}{\varepsilon^p}\right)$ n.a. 1-s. Thm. 1.3 + Fact 1.1
	$\Omega\left(\frac{d\log n}{\varepsilon} - \frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ a. 2-s. [17]	$\Omega\left(\frac{1}{\varepsilon^p}\log\frac{1}{\varepsilon^p}\right)$ n.a. 1-s. Thm. 2.14 + Fact 1.1
D	$O\left(\sqrt{\frac{\lvert D\rvert}{\varepsilon}}\right)$ n.a. 1-s. [32]	$O\left(\frac{\sqrt{\lvert D\rvert}}{\varepsilon^{\frac{p}{2}}}\right)$ n.a. 1-s. [32] + Lem. 2.2 + Fact 1.1
	$\lvert D\rvert^{\Omega\left(\frac{1}{\log\log\lvert D\rvert}\right)}$ n.a. 2-s. [32]	$\lvert D\rvert^{\Omega\left(\frac{1}{\log\log\lvert D\rvert}\right)}$ n.a. 2-s. [32] + Fact 1.1

Table 1.1: Query complexity of L_p-testing monotonicity of a function $f : \mathcal{D} \to [0,1]$ (a./n.a. = adaptive/nonadaptive, 1-s./2-s. = 1-sided error/2-sided error).

(Lemma 2.1) and the best known bounds on Boolean testers for these domains. For the hypergrid domains, in addition to Fact 1.1 and Lemma 2.1, the results in the table rely on our improvements to the bounds on the complexity of testing Boolean monotonicity on hypergrids, which we describe next.

L_1-testing on hypergrids and Levin's work investment strategy. One of our reductions (described above) shows that the nonadaptive complexity of L_1-testing monotonicity is the same for functions over $[0,1]$ and over $\{0,1\}$. Dodis et al. [24] gave a monotonicity tester of Boolean functions on $[n]^d$ that makes $O\left(\frac{d}{\varepsilon}\log^2\frac{d}{\varepsilon}\right)$ queries and runs in $O\left(\frac{d}{\varepsilon}\log^3\frac{d}{\varepsilon}\right)$ time. We obtain a tester with better query and time complexity.

Theorem 1.3. *Let $n, d \in \mathbb{N}$ and $\varepsilon \in (0,1)$. The time complexity of L_1-testing monotonicity of functions $f\colon [n]^d \to [0,1]$ with proximity parameter ε (nonadaptively and with one-sided error) is $O\left(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\right)$.*

The test in [24] is based on the dimension reduction (stated as Theorem 2.6 in Section 2.2) and Levin's work investment strategy [45], described in detail by Goldreich [33]. Our improvement in the upper bound on the query complexity stems from an improvement to Levin's strategy. (The additional improvement in running time comes from a more efficient check for violations of monotonicity among sampled points.) As described in [33], Levin's strategy has been applied in many different settings [45, 34], including testing connectedness of bounded-degree graphs in [35], testing connectedness of images in [50] and analyzing complexity of the concatenation problem [33]. Our improvement to Levin's strategy saves a logarithmic factor in the running time in these applications. Specifically, whether a graph of bounded degree is connected can be tested with $O\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ queries, whether an image represents a connected object can be tested with $O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)$ queries, and there is only an $O\left(\log\frac{1}{\varepsilon}\right)$ overhead in the query complexity of the concatenation of property testing instances as compared to solving a single instance.

Role of adaptivity. Researchers have repeatedly asked whether adaptivity is helpful in testing monotonicity. All previously known adaptive tests have been shown to have nonadaptive analogs with the same query

and time complexity. (See, e.g., [12] for a discussion of both points.) Yet, for some domains and ranges there is a large gap between adaptive and nonadaptive lower bounds. We exhibit the first monotonicity testing problem where adaptivity provably helps: we show that for functions of the form $f : [n]^2 \to \{0, 1\}$, monotonicity testing can be performed with $O\left(\frac{1}{\varepsilon}\right)$ queries with an adaptive 1-sided error algorithm, while every nonadaptive 1-sided error algorithm for this task requires $\Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ queries. Our upper bound of $O\left(\frac{1}{\varepsilon}\right)$ queries holds more generally: for any constant d. This upper bound is optimal because one needs $\Omega\left(\frac{1}{\varepsilon}\right)$ queries to test any nontrivial property, including monotonicity. Our lower bound shows that the tester from Theorem 1.3 is an optimal nonadaptive 1-sided error tester for hypergrids of constant dimension.

Our adaptive tester is based on an algorithm that partially learns the class of monotone Boolean functions over $[n]^d$. (The partial learning model is formalized in Definition 2.3. In particular, our partial learner implies a proper PAC learner under the uniform distribution with membership queries.) A straightforward transformation[6] gives a 1-sided error tester from the learner. For the special case of $d = 2$, the tester has the desired $O(\frac{1}{\varepsilon})$ query complexity. Our $O\left(\frac{1}{\varepsilon}\right)$-query tester for higher dimensions is more sophisticated: it uses our nonadaptive monotonicity tester (from Theorem 1.3) in conjunction with the learner. The idea is that the values previously deduced by the learner do not have to be queried, thus reducing the query complexity.

Our lower bound for nonadaptive testing is based on a hexagon packing. (See Figures 2.1 and 2.2).

Tolerant testing and distance approximation. We give L_1-distance approximation algorithms with additive error δ for monotonicity of functions on the line and the 2-dimensional grid. The query complexity for the line and the grid are $O(1/\delta^2)$ and $\tilde{O}(1/\delta^4)$, respectively. Our algorithm for the line is optimal. It implies a tolerant L_1-tester for monotonicity on the line with query complexity $O\big(\frac{\varepsilon_2}{(\varepsilon_2-\varepsilon_1)^2}\big)$ and, by Fact 1.2, for $p \geq 1$, a tolerant L_p-tester for this problem with query complexity $O\big(\frac{\varepsilon_2^p}{(\varepsilon_2^p-\varepsilon_1)^2}\big)$. The next theorem summarizes our results for L_1-distance approximation and tolerant L_1-testing.

Theorem 1.4. *Let $n \in \mathbb{N}$ and $\delta \in (0, 1)$. There is an algorithm that approximates the relative L_1-distance to monotonicity, $d_1(f, \mathcal{M})$, of functions $f\colon [n]^d \to [0, 1]$ with additive error δ with probability at least 2/3;*

1. *for $d = 1$, it makes $O\left(\max\left(\frac{d_{\mathcal{M}}(f)}{\delta^2}, \frac{1}{\delta}\right)\right)$ queries and runs in time ...*
2. *for $d = 2$, it makes $\tilde{O}(1/\delta^4)$ queries and runs in time ...*

Let $n \in \mathbb{N}$ and $\varepsilon_1, \varepsilon_2 \in (0, 1)$, where $\varepsilon_1 < \varepsilon_2$. There is an $(\varepsilon_1, \varepsilon_2)$-tolerant L_1-tester for monotonicity of functions $f\colon [n]^d \to [0, 1]$;

3. *for $d = 1$, it makes $O\big(\frac{\varepsilon_2}{(\varepsilon_2-\varepsilon_1)^2}\big)$ queries and runs in time ...*
4. *for $d = 2$, it makes $\tilde{O}(1/(\varepsilon_2 - \varepsilon_1)^4)$ queries and runs in time ...*

A crucial building block of our algorithms is the reduction from the general approximation problem to the special case of Boolean functions (Lemma 2.3). For the line, we further reduce the problem to approximating the longest correct bracket subsequence. Our distance approximation algorithm for Boolean functions improves on the $\tilde{O}(1/\delta^2)$-query algorithm of Fattal and Ron [28]. For $d = 2$, we apply our reduction to the algorithm of [28] for Boolean functions.

1.2.2 The c-Lipschitz properties

The c-Lipschitz properties are a subject of a recent wave of investigation [43, 4, 23, 15, 18], with a focus on hypergrid domains, due to their applications to differential privacy [25].

[6] Our transformation can be viewed as an analog of Proposition 3.1 in [36]. This proposition relates the query complexity of 2-sided error testing to the sample complexity of proper learning. Our transformation requires a stronger learner and yields a 1-sided error tester.

The c-Lipschitz property		
Domain	Hamming Testing	L_p-Testing for $p \geq 1$
$[n]^d$	$O\left(\frac{d \log n}{\epsilon}\right)$ n.a. 1-s. [15]	$O\left(\frac{d}{\varepsilon^p}\right)$ n.a. 1-s. Thm. 1.5 + Fact 1.1
	$\Omega\left(\frac{d \log n}{\varepsilon} - \frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ a. 2-s. [18]	$\Omega(d + \frac{1}{\varepsilon^p})$ a. 2-s. [43] + Fact 1.1

Table 1.2: Query complexity of L_p-testing the c-Lipschitz property of a function $f : \mathcal{D} \to [0,1]$ for $p \geq 1$ (a./n.a. = adaptive/nonadaptive, 1-s./2-s. = 1-sided error/2-sided error).

Definition 1.5 (c-Lipschitz function)**.** *Let (D, d_D) be a finite metric space, i.e., D is a finite set and $d_D : D \times D \to \mathbb{R}$ is a metric. Let* the Lipschitz constant *$c > 0$ be a real number. A function $f : D \to \mathbb{R}$ is* c-Lipschitz *if $|f(x) - f(y)| \leq c \cdot d_D(x, y)$ for all $x, y \in D$. If $c = 1$, such a function is called* Lipschitz*.*

The hypergrid, the hypercube and the line domains are defined as for monotonicity, except that instead of equipping them with the partial order, we equip them with the following metric: $d_D(x, y) = \|x - y\|_1$.

Characterization. We give a combinatorial characterization of the L_1 distance to the Lipschitz property (in Lemma 4.1). We show that it is equal to the weight of a maximum weight matching in the appropriately defined graph associated with the function. We note that a similar-looking near-characterization is known w.r.t. the Hamming distance, but the upper and lower bounds on the Hamming distance to the Lipschitz property are off by a factor of 2. Our characterization w.r.t. the L_1 distance is tight.

L_1-testing on hypergrids. We use our characterization to obtain a c-Lipschitz L_1-tester for functions over hypergrids that is faster by a factor of $\log n$ than the best possible Hamming tester. Known bounds on the query complexity of testing the c-Lipschitz property on hypergrids are summarized in Table 1.2.

Theorem 1.5. *Let $n, d \in \mathbb{N}$ and $\varepsilon, c \in (0, 1)$. The time complexity of L_1-testing the c-Lipschitz property of functions $f\colon [n]^d \to [0, 1]$ (nonadaptively and with 1-sided error) with proximity parameter ε is $O\left(\frac{d}{\varepsilon}\right)$.*

The running time of our tester has optimal dependence on dimension d. This follows from the $\Omega(d)$ lower bound on Hamming testing of the Lipschitz property of functions $f : \{0,1\}^d \to \{0,1,2\}$ in [43]. (This problem is equivalent to Hamming testing 1/2-Lipschitz property of functions $f : \{0,1\}^d \to \{0, 1/2, 1\}$, and for functions with this range, relative L_0 and L_1 distances are off by at most factor of 2.)

The running time of our tester does not depend on the Lipschitz constant c, but the algorithm itself does. The crux of designing the algorithm is understanding the number of pairs of points on the same line which do not obey the Lipschitz condition (called *violated pairs*), and selecting the right subset of pairs (depending on c) so that a constant fraction of them are violated by any function on the line that is ε-far from monotone. The analysis uses dimension reduction from [4], generalized to work for functions with range $\mathbb{R}$.

1.2.3 Convexity and Submodularity

We establish and exploit the connection of L_1-testing to approximation theory. Our results for testing convexity of functions over $[n]^d$ follow from this connection and are given in Appendix A. For $d = 1$, we get an optimal tester with query complexity $O(1/\epsilon)$ and for higher dimensions, query complexity is independent of the linear dimension n.

2 L_p-Testing Monotonicity

2.1 Characterization of L_1 Distance to Monotonicity

We characterize the L_1 distance to monotonicity in terms of the distance to monotonicity of Boolean functions. We use this characterization to obtain reductions from the general monotonicity testing and distance approximation to the case of Boolean functions, presented in Lemmas 2.2 and 2.3, respectively. The main idea in our characterization is that every function can be viewed as an integral over Boolean threshold functions, defined next.

Definition 2.1. *For a function $f\colon D \to [0,1]$ and $t \in [0,1]$, the* threshold function $f_{(t)}\colon D \to \{0,1\}$ *is:*

$$f_{(t)}(x) = \begin{cases} 1 & \text{if } f(x) \geq t; \\ 0 & \text{if } f(x) < t. \end{cases}$$

We can express a real-valued function $f\colon D \to [0,1]$ as an integral over its Boolean threshold functions:

$$f(x) = \int_0^{f(x)} dt = \int_0^1 f_{(t)}(x) dt.$$

The integrals above and all other integrals in this section are well defined because we are integrating over piecewise constant functions.

Let $L_1(f, \mathcal{M})$ denote the L_1 distance from f to the set of monotone functions, $\mathcal{M}$, and let $d_{\mathcal{M}}(f)$ be the relative version of this distance, i.e., $d_{\mathcal{M}}(f) = L_1(f, \mathcal{M})/|D|$ for functions $f : D \to [0,1]$.

Lemma 2.1 (Characterization of L_1 distance to monotone)**.** *For every function $f\colon D \to [0,1]$,*

$$d_{\mathcal{M}}(f) = \int_0^1 d_{\mathcal{M}}(f_{(t)})\, dt.$$

Proof. Since f and $f_{(t)}$ are functions over the same domain, it is enough to prove $L_1(f, \mathcal{M}) = \int_0^1 L_1(f_{(t)}, \mathcal{M})\, dt$. First, we prove that $L_1(f, \mathcal{M}) \leq \int_0^1 L_1(f_{(t)}, \mathcal{M})\, dt$. For all $t \in [0,1]$, let g_t be the closest monotone (Boolean) function to $f_{(t)}$. Define $g = \int_0^1 g_t\, dt$. Since g_t is monotone for all $t \in [0,1]$, function g is also monotone. Then

$$\begin{aligned} L_1(f, \mathcal{M}) &\leq \|f - g\|_1 = \Big\| \int_0^1 f_{(t)}\, dt - \int_0^1 g_t\, dt \Big\|_1 \\ &= \Big\| \int_0^1 (f_{(t)} - g_t)\, dt \Big\|_1 \\ &\leq \int_0^1 \|f_{(t)} - g_t\|_1\, dt = \int_0^1 L_1(f_{(t)}, \mathcal{M})\, dt. \end{aligned}$$

Next, we prove that $L_1(f, \mathcal{M}) \geq \int_0^1 L_1(f_{(t)}, \mathcal{M})\, dt$. Let g denote the closest monotone function to f in

L_1 distance. Then $g_{(t)}$ is monotone for all $t \in [0,1]$. We obtain:

$$\begin{aligned} L_1(f,\mathcal{M}) &= \|f-g\|_1 = \left\| \int_0^1 (f_{(t)} - g_{(t)})\, dt \right\|_1 \\ &= \sum_{x: f(x) \geq g(x)} \int_0^1 (f_{(t)}(x) - g_{(t)}(x))\, dt \\ &\quad + \sum_{x: f(x) < g(x)} \int_0^1 (g_{(t)}(x) - f_{(t)}(x))\, dt \\ &= \int_0^1 \Big(\sum_{x: f(x) \geq g(x)} (f_{(t)}(x) - g_{(t)}(x)) \\ &\quad + \sum_{x: f(x) < g(x)} (g_{(t)}(x) - f_{(t)}(x)) \Big)\, dt \\ &= \int_0^1 \|f_{(t)} - g_{(t)}\|_1\, dt \geq \int_0^1 L_1(f_{(t)}, \mathcal{M})\, dt. \end{aligned}$$

The last equality above holds because for all functions f, g and $x \in D$, the inequality $f(x) \geq g(x)$ holds iff for all thresholds $t \in [0,1]$, it holds that $f_{(t)}(x) \geq g_{(t)}(x)$. □

We use our characterization in reductions for L_1-testing (Lemma 2.2) and distance approximation (Lemma 2.3).

Lemma 2.2. *If T is a nonadaptive 1-sided error ε-test for monotonicity of functions $f : D \to \{0,1\}$ then it is also a nonadaptive 1-sided error ε-test w.r.t. the L_1 distance for monotonicity of functions $f : D \to [0,1]$.*

Proof. Observe that a 1-sided error nonadaptive test consists of querying f on a random (not necessarily uniformly random) set of points $Q \subseteq D$ and accepting iff f is monotone on Q. Such a test always accepts monotone functions. It remains to prove that if f is ε-far from monotone w.r.t. the L_1 distance, then T will reject with probability at least $2/3$.

Assume that $d_{\mathcal{M}}(f) \geq \varepsilon$. Then, it follows from Lemma 2.1 that there exists a threshold $t^* \in [0,1]$ such that $d_{\mathcal{M}}(f_{(t^*)}) \geq \varepsilon$. Recall that for Boolean functions, Hamming distance is the same as the L_1 distance. Since T is an ε-test for monotonicity of Boolean functions, for the random set Q selected by T, the restriction $f_{(t^*)}|_Q$ is not monotone with probability at least $2/3$. It is well known (see [32]) that for every function h, the restriction $h|_Q$ is not monotone iff there is a pair of points in Q *violated* by h, i.e., $x, y \in Q$ such that $x \prec y$ and $h(x) > h(y)$. That is, if $f_{(t^*)}|_Q$ is not monotone then $f_{(t^*)}(x) = 1$ and $f_{(t^*)}(y) = 0$ for some $x \prec y$, where $x, y \in Q$. But then this pair (x, y) is also violated by f, since $f(x) \geq t^* > f(y)$, and the restriction $f|_Q$ is not monotone. Thus, $f|_Q$ is not monotone with probability $2/3$, and the test T satisfies the requirements in the lemma. □

Lemma 2.3. *Let A be a nonadaptive algorithm that approximates $d_{\mathcal{M}}(f)$ for Boolean functions f over D with the following guarantee: there exist concave functions $\delta_A(), \sigma_A()$ such that for all input functions f, $|\mathbb{E}[A(f)] - d_{\mathcal{M}}(f)| \leq \delta_A(d_{\mathcal{M}}(f))$ and the standard deviation $\sigma(A(f)) \leq \sigma_A(d_{\mathcal{M}}(f))$.*

Suppose A makes q_A queries and runs in time t_A. Then there is a nonadaptive algorithm A' that approximates $d_{\mathcal{M}}(f)$ for functions $f : D \to [0,1]$ with the same guarantee, q_A queries and at most $q_A \cdot t_A$ time. Moreover, if A outputs $d_{\mathcal{M}}(f_{|Q})$ for a random (not necessarily uniformly random) set of queries Q, then so does A'. In this case, the running time of A' is $O(q_A \cdot \log q_A)$.

Proof. Let A' be the algorithm that on input f runs algorithm A on a Boolean function with the same domain as f, queries f on the q_A positions queried by A and uses the answers to these queries to compute and output $\int_0^1 A(f_{(t)})dt$. Observe that only values returned by the oracle for f in response to the queries need to be considered as thresholds in order to compute the integral, so the running time is at most $q_A \cdot t_A$.

Moreover, if $A(f) = d_{\mathcal{M}}(f_{|Q})$ for some random set Q of size q_A then, by Lemma 2.1, $A'(f) = \int_0^1 A(f_{(t)})dt = \int_0^1 d_{\mathcal{M}}((f_{(t)})_{|Q})dt = \int_0^1 d_{\mathcal{M}}((f_{|Q})_{(t)})dt = d_{\mathcal{M}}(f_{|Q})$. By Lemma B.1, this can be computed in time $O(q_A \cdot \log q_A)$. This lemma applies because the partially ordered set Q is a line.

Now we prove the guarantee on the accuracy of A'. Consider any function $f : D \to [0,1]$. First, we show that $\mathbb{E}[A'(f)] \leq \delta_A + d_{\mathcal{M}}(f)$. We get:

$$\begin{aligned}\mathbb{E}[A'(f)] &= \mathbb{E}\left[\int_0^1 A(f_{(t)})\,dt\right] = \int_0^1 \mathbb{E}[A(f_{(t)})]\,dt \\ &\leq \int_0^1 [\delta_A(d_{\mathcal{M}}(f_{(t)})) + d_{\mathcal{M}}(f_{(t)})]\,dt \\ &\leq \delta_A\left(\int_0^1 d_{\mathcal{M}}(f_{(t)})\right)dt + \int_0^1 d_{\mathcal{M}}(f_{(t)})\,dt \\ &= \delta_A + d_{\mathcal{M}}(f)\end{aligned}$$

The first equality holds by definition of A, the second—by the linearity of expectation, the first inequality follows from the accuracy guarantee on A, the second inequality is by Jensen's inequality applied to $\delta_A()$, and the final equality holds by Lemma 2.1. Similarly, $\mathbb{E}[A'(f)] \geq d_{\mathcal{M}}(f) - \delta_A$. Putting these two inequalities together, we obtain $|\,\mathbb{E}[A'(f)] - d_{\mathcal{M}}(f)| \leq \delta_A$ for all f.

It remains to prove the bound on $\sigma(A'(f))$, the standard deviation of A'. We get:

$$\sigma(A'(f)) \leq \int_0^1 \sigma(A(f_{(t)}))\,dt \leq \int_0^1 \sigma_A(d_{\mathcal{M}}(f_{(t)}))\,dt \leq \sigma_A\left(\int_0^1 d_{\mathcal{M}}(f_{(t)})\right) = \sigma_A(d_{\mathcal{M}}(f)),$$

where the first inequality holds by subadditivity of standard deviation, the second one uses the bound on standard deviation of A from the assumption of the lemma, the third inequality is by Jensen's inequality and concavity of σ_A and the last equality is by Lemma 2.1. □

Using Lemma 2.3 we can derive L_1-distance approximation results from the corresponding results for Boolean functions given by Fattal and Ron [28].

Corollary 2.4. *There exists an L_1 distance approximation algorithm for functions $f\colon [n]^d \to [0,1]$ with additive error δ with query complexity:*

1. *$\tilde{O}(1/\delta^2)$ for $d = 1$.*
2. *$\tilde{O}(1/\delta^4)$ for $d = 2$.*

Proof. These two results follow from Lemma 2.3 together with a distance approximation algorithms of Fattal and Ron [28], who give a distance approximation algorithm for Boolean functions $f\colon [n]^d \to \{0,1\}$ with additive error δ and complexity $\tilde{O}(1/\delta^2)$ for $d = 1$ and $\tilde{O}(1/\delta^4)$ for $d = 2$. Bounds on expectation and variance that we need for the reduction can be derived form additive error bounds, as shown in Appendix D. □

The bound in the second part of Theorem 1.4 corresponds to the second part of the corollary above. As for the first part of this theorem, we prove a better bound using a different algorithm than that of [28] in Section 3.2.

2.2 Nonadaptive Monotonicity Tester for Boolean Functions on Hypergrids

Before proving Theorem 1.3, we state one of the building blocks—the probabilistic inequality that leads to an improvement on Levin's work investment strategy. To motivate the result, we summarize Goldreich's explanation [33] on how it can be used and instantiate it with our scenario.

Suppose an algorithm needs to find some "evidence" (e.g., a violation of monotonicity) using as little work as possible[7]. It can select an element (e.g., a line) according to some distribution $\mathcal{D}$ and invest some work in it to discover the evidence. All elements e have different quality $q(e) \in [0,1]$, not known in advance (e.g., the distance from the line to monotonicity). To extract evidence from element e (e.g., find a violated pair on the line), the algorithm must invest work which is higher when the quality $q(e)$ is lower (e.g., work inversely proportional[8] to $q(e)$). Suppose $\mathbb{E}_{e \leftarrow \mathcal{D}}[q(e)] \geq \alpha$. What is a good work investment strategy for the algorithm?

This question can be formalized as follows. The algorithm's strategy is the list $(s_1, \ldots, s_k)$ of k positive integers, indicating that the algorithm will invest work s_i in the ith element. In the ith step, the algorithm selects an element e_i according to its distribution. If s_i is at least the work needed for item e_i, the algorithm wins. If it does not win in k steps, it loses. The cost of the strategy is $\sum_{i=1}^{k} s_i$, the total work the algorithm decided to invest. What is the minimum cost of a strategy that guarantees that the algorithm wins with constant probability (specifically, say, probability at least 2/3)? A good strategy is suggested by the following probabilistic inequality (a strengthening of [33, Fact A.1]). In this inequality, X represents the random variable equal to $q(e)$ when e is selected according to $\mathcal{D}$.

Lemma 2.5. *Let X be a random variable that takes values in $[0,1]$. Suppose $\mathbb{E}[X] \geq \alpha$, where $\alpha \leq 1/2$, and let $t = \lceil 3 \log \frac{1}{\alpha} \rceil$. Let $\delta \in (0,1)$ be the desired probability of error. For all $j \in [t]$, let $p_j = \Pr[X \geq 2^{-j}]$ and $k_j = \frac{4 \ln 1/\delta}{2^j \alpha}$. Then*

$$\prod_{j=1}^{t} (1-p_j)^{k_j} \leq \delta.$$

That is, for each $j \in [t]$, the algorithm can set k_j of the s_i's to be equal to the work needed for elements with quality 2^{-j} (or higher). Then the probability that the algorithm loses in all rounds is at most δ. This strategy achieves a reduction by a factor of t for the number of s_i's that need to be devoted to elements of quality 2^{-j}, compared to Levin's strategy, as explained by Goldreich. For example, when the work is inversely proportional to the quality, k_j of the s_i's would be set to 2^j. Then the total cost of the strategy for constant δ is $\sum_{j \in [t]} O(1/(2^j \alpha)) \cdot 2^j = O\left(\frac{1}{\alpha} \log \frac{1}{\alpha}\right)$. This is a reduction by a factor of $\log \frac{1}{\alpha}$ in total work, compared to Levin's strategy. Lemma C.1 in the appendix shows that the improved strategy is optimal for this case.

Proof of Lemma 2.5. It is enough to prove that

$$\sum_{j=1}^{t} \frac{p_j}{2^j} \geq \frac{\alpha}{4}. \tag{1}$$

It implies the lemma since $\prod_{j=1}^{t} (1-p_j)^{k_j} \leq \prod_{j=1}^{t} e^{-p_j \cdot k_j} = e^{-\sum_{j=1}^{t} p_j \cdot k_j} = e^{-\sum_{j=1}^{t} \frac{p_j}{2^j \alpha} \cdot (4 \ln 1/\delta)} \leq e^{-\frac{1}{4} \cdot (4 \ln 1/\delta)} = \delta$. The first inequality in this calculation follows from the fact that $1 - x \leq e^{-x}$ and the last inequality follows from (1).

[7] Levin's work investment strategy, as described by Goldreich [33], deals with a *detection* problem. A similar idea was used by Chazelle, Rubinfeld, and Trevisan [20] to address an *estimation* problem: namely, estimating the number of connected components in a graph.

[8] The relationship between work and quality is different in our application to monotonicity, but this one is most frequently used. Also, in the application to monotonicity, there is no certainty that if you invest the specified amount of work, you will find a witness; rather, the probability of finding a witness increases with the amount of work you invest.

To prove (1), first, we bound the sum on the left-hand side of (1) with i going to ∞ instead of t:

$$\begin{aligned}\sum_{j=1}^{\infty} \frac{p_j}{2^j} &= \frac{1}{2}\sum_{j=1}^{\infty} 2^{-j+1} \Pr[X \geq 2^{-j}] \\ &\geq \frac{1}{2}\sum_{j=1}^{\infty} 2^{-j+1} \Pr\left[X \in \left(2^{-j}, 2^{-j+1}\right]\right] \\ &\geq \frac{1}{2}\,\mathbb{E}[X] \geq \frac{\alpha}{2}.\end{aligned}$$

The first inequality above holds since the interval $\left(2^{-j}, 2^{-j+1}\right]$ is contained in the interval $\left[2^{-j}, \infty\right)$. The second inequality follows from the definition of the expectation.

The terms corresponding to large i do not contribute much to the sum above:

$$\sum_{j=t+1}^{\infty} \frac{p_j}{2^j} \leq \sum_{j=t+1}^{\infty} \frac{1}{2^j} \leq \frac{2}{2^{t+1}} \leq \alpha^3.$$

Here, the first inequality holds because p_j are probabilities, and hence are at most 1. The last inequality follows from the definition of t.

Thus, $\sum_{j=1}^{t} \frac{p_j}{2^j} \geq \frac{\alpha}{2} - \alpha^3$, which is at least $\alpha/4$ for all $\alpha \leq 1/2$. This proves (1), and the lemma follows. □

Proof of Theorem 1.3. By Lemma 2.2, it suffices to prove Theorem 1.3 for the special case when the input function f is Boolean. Our monotonicity tester for functions $f : [n]^d \to \{0,1\}$, like that in [24], works by picking a random axis-parallel line, querying a few random points on it and checking if they violate monotonicity. The difference is in the choice of the number of lines and points to query and in a more efficient check for monotonicity violations.

We start by establishing notation for axis-parallel lines. For $i \in [d]$, let $e^i \in [n]^d$ be 1 on the ith coordinate and 0 on the remaining coordinates. Then for every dimension $i \in [d]$ and $\alpha \in [n]^d$ with $\alpha_i = 0$, the *line along dimension i with position α* is the set $\{\alpha + x_i \cdot e^i \mid x_i \in [n]\}$. Let $\mathcal{L}_{n,d}$ be the set of all dn^{d-1} axis-parallel lines in $[n]^d$. Let $d_{\mathcal{M}}(f)$ denote the (relative) distance to monotonicity of a function f.

Algorithm 1: Nonadaptive monotonicity tester of Boolean functions on hypergrids.

input : parameters n, d and ε; oracle access to $f\colon [n]^d \to \{0,1\}$.

for $j = 1$ **to** $\lceil 3\log(2d/\varepsilon)\rceil$ **do**
  repeat $\lceil (8\ln 9)d/(2^j\varepsilon)\rceil$ times:
    Sample a uniformly random line ℓ from $\mathcal{L}_{n,d}$.
    Query a set Q of $10 \cdot 2^j$ points selected uniformly at random from ℓ.
    Let x be the largest coordinate along ℓ of a point in Q whose f-value is 0.
    Let y be the smallest coordinate along ℓ of a point in Q whose f-value is 1.
    if $x > y$ **then reject**
accept

Consider the test presented in Algorithm 1. Clearly, its query complexity and running time are $O\left(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\right)$. It is nonadaptive and always accepts all monotone functions. It remains to show that functions that are ε-far from monotone are rejected with probability at least 2/3. We use the following two lemmas from [24] in our analysis of this case.

Lemma 2.6 ([24, Lemma 6(2)], Dimension Reduction for Boolean Range)**.** *For $f\colon [n]^d \to \{0,1\}$,*

$$\mathop{\mathbb{E}}_{\ell \leftarrow \mathcal{L}_{n,d}}[d_{\mathcal{M}}(f|_\ell)] \geq \frac{d_{\mathcal{M}}(f)}{2d}.$$

Lemma 2.7 ([24, Lemma 16])**.** *For $f\colon [n] \to \{0,1\}$ if $d_{\mathcal{M}}(f) = \gamma$ then for a random sample $Q \subseteq [n]$ of size k, the probability that $f|_Q$ is not monotone is at least $(1 - e^{-\gamma k/4})^2$.*

Consider the behavior of Algorithm 1 on an input function f that is ε-far from monotone. We apply our improvement to Levin's strategy (Lemma 2.5) with the random variable $X = d_{\mathcal{M}}(f|_\ell)$, where line ℓ is selected uniformly from $\mathcal{L}_{n,d}$, and with probability of error $\delta = 1/9$. By Lemma 2.6, the expectation $\mathbb{E}[X] \geq \frac{\varepsilon}{2d}$. Thus, in Lemma 2.5, parameter α is set to $\frac{\varepsilon}{2d}$, parameter t is set to $\lceil 3 \log \frac{2d}{\varepsilon} \rceil$, and for all $j \in [t]$, the parameter $k_j = \frac{(8 \ln 9) \cdot d}{2^j \varepsilon}$. By Lemma 2.5, with probability at least 8/9, in at least one iteration of Step 3, Algorithm 1 will sample a line ℓ satisfying $d_{\mathcal{M}}(f|_\ell) \geq 2^{-j}$. Conditioned on sampling such a line, by Lemma 2.7, the sample Q taken in Step 4 of Algorithm 1 results in a non-monotone restriction $f|_Q$ with probability at least 3/4. Finally, Step 6 of Algorithm 1 rejects iff $f|_Q$ is not monotone. Thus, for the index j given by Lemma 2.5, the probability that Algorithm 1 rejects f in the jth iteration of the **for** loop is at least $\frac{8}{9} \cdot \frac{3}{4} = \frac{2}{3}$, as required. □

2.3 Adaptive Monotonicity Tester for Boolean Functions on Hypergrids

Theorem 2.8. *The query complexity of ε-testing monotonicity of functions $f\colon [n]^d \to \{0,1\}$ (with one-sided error) is $O\big(d2^d \log^{d-1} \frac{1}{\varepsilon} + \frac{d^2 \log d}{\varepsilon}\big)$. Specifically, for constant d, the query complexity is $O\left(\frac{1}{\varepsilon}\right)$.*

Recall from Section 1.2.1 that our adaptive tester is based on an algorithm that learns the class of monotone Boolean functions over $[n]^d$ in the sense of Definition 2.3 below. The learner is presented in Section 2.3.1. Transformation from the learner to the tester is given in Lemma 2.13. This transformation, together with our learner, immediately implies Theorem 2.8 for the special case of $d = 2$. The general test is presented in Section 2.3.3.

2.3.1 Partial learner of monotone Boolean functions

Definition 2.2. *An* ε-partial function *g with domain D and range R is a function $g : D \to R \cup \{?\}$ that satisfies $\Pr_{x \in D}[g(x) = ?] \leq \varepsilon$. An ε-partial function g* agrees *with function f if $g(x) = f(x)$ for all x on which $g(x) \neq ?$. Given a function class $\mathcal{C}$, let $\mathcal{C}_\varepsilon$ denote the class of ε-partial functions, each of which agrees with some function in $\mathcal{C}$.*

Definition 2.3. *An* ε-partial learner *for a function class $\mathcal{C}$ is an algorithm that, given a parameter ε and oracle access to a function f, outputs a hypothesis $g \in \mathcal{C}_\varepsilon$ or fails. Moreover, if $f \in \mathcal{C}$ then it outputs g that agrees with f.*

Theorem 2.9. *Algorithm 2 is a $d/2^t$-partial learner for the class of monotone Boolean functions over $[n]^d$. It makes $O(d2^{t(d-1)+d})$ queries.*

Proof. Our learner (Algorithm 2) constructs a d-dimensional quadtree representation of (a partial function that agrees with) the input function f. Each node in the tree is marked 0, 1 or ?. The algorithm starts by constructing the root that holds the entire domain $[n]^d$ and is marked with ?. We use $\mathbf{1}_d$ to denote the d-dimensional vector of 1s. The domain can be thought of as a d-dimensional cube with the low and high corners at $\mathbf{1}_d$ and $n \cdot \mathbf{1}_d$, respectively. For the ease of presentation, we assume that n is a power of 2. At every stage, the learner constructs the next level of the tree by assigning 2^d children to each ? node of the previous level. The cubes corresponding to the children are obtained by cutting the cube of the parent through the middle in all dimensions. We record the cube corresponding to node v by storing its smallest and largest element in v[low] and v[hi], respectively. The algorithm queries the smallest and the largest element of each new square and marks the corresponding node of the quadtree with 0 or 1 if the function values on the points in the cube are determined by the answers. Namely, if f maps the largest element to 0, then the node is marked with 0; if f maps the smallest element to 1, then it is marked with 1. Otherwise, the new node is marked ?. The learner stops after constructing t tree levels.

The quadtree is a representation of a partial Boolean function g on $[n]^d$ in the following sense. Each point x in the domain is contained in exactly one leaf, and $g(x)$ is equal to the value the leaf is marked with.

Algorithm 2: Partial learner of monotone Boolean functions on hypergrids

input : parameters n, d and ε; depth t; oracle access to a monotone $f\colon [n]^d \to \{0,1\}$.
output: a d-dimensional quadtree representation of f.

1 Create quadtree root marked with ? and holding $[n]^d$, i.e., set root[low]$= \mathbf{1}_d$ and root[hi]$= n \cdot \mathbf{1}_d$.
   `// 1_d denotes the d-dimensional vector of 1s.`
2 **for** $j = 1$ **to** t **do**
   `// Create nodes of level j in the quadtree.`
3 Set $len = \frac{n}{2^j}$.
4 **foreach** leaf v of the quadtree marked with ? **do**
   `// Create 2^d children of v.`
5 **foreach** vector $\mathbf{k} \in \{0,1\}^d$ **do**
6 Create child $v_{\mathbf{k}}$ of v marked with ?.
7 Set $v_{\mathbf{k}}[\text{low}] = v[\text{low}] + len \cdot \mathbf{k}$ and $v_{\mathbf{k}}[\text{hi}] = v_{\mathbf{k}}[\text{low}] + (len - 1)\mathbf{1}_d$.
8 **if** $f(v_k[\text{hi}]) = 0$ **then** mark child v_k with 0.
9 **if** $f(v_k[\text{low}]) = 1$ **then** mark child v_k with 1.
10 **if** *more than* $d2^{j(d-1)}$ *nodes are marked* ? **then fail**.
11 **return** the quadtree

Lemma 2.10. *Algorithm 2 never fails if the input function f is monotone.*

Proof. We need to show that in the quadtree constructed by Algorithm 2 run on a monotone function f, for all j, at most $d2^{j(d-1)}$ nodes in level j are marked ?. Fix j. Consider the set $S = \{\frac{v[\text{hi}]}{n/2^j} \mid v \text{ is a node of level } j \text{ marked with ?}\}$.

Definition 2.4. *Let $x, y \in [n]^d$. We say that y* fully dominates *x if $x_i < y_i$ for all $i \in [d]$.*

Consider two nodes u and v at level j of the quadtree, such that v[hi] fully dominates u[hi]. Observe that at most one of u and v can be marked with ?. Indeed, if $f(u[\text{hi}]) = 0$ then u is marked with 0. Otherwise, $f(u[\text{hi}]) = 1$. By monotonicity of f, since $u[\text{hi}] \prec v[\text{low}]$, it follows that $f(v[\text{low}]) = 1$. That is, v is marked with 1. Thus, for no elements $x, y \in S$, element y fully dominates element x.

Finally, observe that $S \subseteq [m]^d$, where $m = 2^j$. The lemma follows from Claim 2.11 that bounds the number of elements in a set that satisfies the observed conditions. □

Claim 2.11. *Let $S \subseteq [m]^d$, such that for no elements $x, y \in S$, element y fully dominates element x. Then $|S| \le dm^{d-1}$.*

Proof. Let $A = \{a \in [m]^d \mid a_i = 1 \text{ for some } i \in [d]\}$. We can partition $[m]^d$ into chains $C_a = \{(a + c \cdot \mathbf{1}_d) \in [m]^d \mid c \in \mathbb{N}\}$, where $a \in A$. Note that for all $a \in A$ and all distinct $x, y \in C_a$, element y fully dominates element x. Thus, S can contain at most 1 element in each C_a. Consequently,

$$|S| \le |A| \le d \cdot m^{d-1}.$$

The last inequality holds because there are d ways to choose i such that $a_i = 1$ and at most m^{d-1} ways to set the remaining coordinates. □

Correctness of Algorithm 2. Each node at level t holds $1/2^{dt}$ fraction of the domain points. Because of the fail condition in Step 10, the fraction of the domain points that are associated with ? leaves of the quadtree is at most $\frac{d2^{t(d-1)}}{2^{td}} = \frac{d}{2^t}$.

If function f is monotone, the values on the remaining points are learned correctly because for each domain point x associated with a node that is marked 0 or 1, we have a witness that implies the corresponding value for $f(x)$ by monotonicity of f. Moreover, by Lemma 2.10, Algorithm 2 always outputs a quadtree in this case.

Complexity of Algorithm 2. The learner makes 2^{d+1} queries for each node marked with ?: two per child. Because of the fail condition in Step 10, there at most $\sum_{j=0}^{t} d2^{j(d-1)} \leq 2d \cdot 2^{t(d-1)}$ nodes marked ?, so it makes $O(d2^{t(d-1)+d})$ queries. This completes the proof of Theorem 2.9. □

Corollary 2.12. *There is an ε-partial learner for the class of monotone Boolean functions over $[n]^2$ that makes $O\left(\frac{1}{\varepsilon}\right)$ queries. Over $[n]^d$, there is a $\frac{d}{\log \frac{1}{\varepsilon}}$-partial learner that makes $O\big(d2^d \log^{d-1} \frac{1}{\varepsilon}\big)$ queries.*

Proof. To get the first statement, we run Algorithm 2 with $d = 2$ and depth $t = \lceil \log \frac{1}{\varepsilon} \rceil + 1$. For the second statement, we set $t = \lceil \log \log \frac{1}{\varepsilon} \rceil$. □

2.3.2 Optimal monotonicity tester for functions $[n]^2 \to \{0, 1\}$

For the special case of $d = 2$, we can easily get the tester claimed in Theorem 2.8 from the learner of Theorem 2.9 by applying the following reduction from testing to learning.

Lemma 2.13. *If there is an ε-partial learner for a function class $\mathcal{C}$ that makes $q(\varepsilon)$ queries then $\mathcal{C}$ can be ε-tested with 1-sided error with $q(\varepsilon/2) + O\left(\frac{1}{\varepsilon}\right)$ queries.*

Proof. To test if a function $f \in \mathcal{C}$, we run the learner with parameter $\varepsilon/2$. If it fails to output a hypothesis function g, we reject. Otherwise, we select $\lceil \frac{2 \ln 3}{\varepsilon} \rceil$ points $x \in D$ uniformly at random, and query both f and g on the sample. If we find a point x on which $g(x) \neq ?$, but $f(x) \neq g(x)$, we reject. Otherwise, we accept.

If a function $f \in \mathcal{C}$, it is accepted because the learner will output a hypothesis g which agrees with f. If f is ε-far from $\mathcal{C}$ then any $g \in \mathcal{C}_{\varepsilon/2}$ will differ from f on at least ε domain points, at most $\varepsilon/2$ of which can be mapped to ? by g. Thus, for at least an $\varepsilon/2$ fraction of the domain points $g(x) \neq ?$ and $g(x) \neq f(x)$. Such a point will be selected with probability at least 2/3. □

2.3.3 Optimal monotonicity tester for functions $f\colon [n]^d \to \{0, 1\}$

Proof of Theorem 2.8. Our tester is Algorithm 3.

Algorithm 3: Adaptive monotonicity tester of Boolean functions on hypergrids.

input : parameters $d \geq 3$, n and ε; oracle access to $f\colon [n]^d \to \{0, 1\}$.

1 Let g be the $\frac{d}{\log \frac{1}{\varepsilon}}$-partial function returned by the learner from Corollary 2.12 run on f.
2 **repeat** $\lceil \frac{2 \log 3}{\varepsilon} \rceil$ times:
3 Sample a uniform $x \in [n]^d$, query $f(x)$ and **reject** if $g(x) \neq ?$ and $g(x) \neq f(x)$.
4 Run Algorithm 1 with parameters $n, d, \varepsilon/2$ as follows:
5 **foreach** *point x queried by the algorithm* **do**
6 Query $f(x)$ only if $g(x) = ?$; otherwise, substitute $g(x)$ for $f(x)$.
7 **accept** if Algorithm 1 accepts, **reject** if it rejects.

Correctness. Algorithm 3 always accepts a monotone function because the learner produces a correct partial function g, there are no inconsistencies between g and f, and Algorithm 1 always accepts monotone functions. Now suppose f is ε-far from monotone. Let $h : [n]^d \to \{0, 1\}$ be the function defined as follows: $h(x) = f(x)$ if $g(x) = ?$, and $h(x) = g(x)$ otherwise. If $d_0(f, h) > \varepsilon/2$ then Step 2 of Algorithm 3 rejects with probability at least 2/3. Otherwise, h is $\varepsilon/2$-far from monotone. Consequently, it is rejected by Step 4 with probability at least 2/3.

Query complexity. By Corollary 2.12, the learning step uses $O\big(d2^d \log^{d-1} \frac{1}{\varepsilon}\big)$ queries. Step 2 (that checks whether g agrees with f) uses $O(1/\varepsilon)$ queries. It remains to analyze Step 4. Algorithm 1 queries $O(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon})$ points. Out of these, only the points mapped to ? by g are queried in Step 4. Since g is a $d/\log \frac{1}{\varepsilon}$-partial function, it maps at most $d/\log \frac{1}{\varepsilon}$ fraction of its domain to ?. The expected number of queries made by Step 4 is $O(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon})$ times the probability that the ith queried point x is mapped to ? by g. This

probability is at most $d/\log\frac{1}{\varepsilon}$, as all points in $[n]^d$ are equally likely to be the ith query (for all i). So, the expected query complexity is $O(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon})\cdot\frac{d}{\log\frac{1}{\varepsilon}} = O(\frac{d^2}{\varepsilon} + \frac{d^2\log d}{\varepsilon\log 1/\varepsilon}) = O(\frac{d^2\log d}{\varepsilon})$.

To get the same asymptotic guarantee in the worst case, we amplify the probability of success of Algorithm 3 to 5/6, run the new version and reject if it attempts to query more than 6 times the expected number of points. By Markov inequality, the algorithm runs into the limit on the number of queries with probability at most 1/6. The resulting algorithm always accepts monotone functions, rejects functions that are ε-far from monotone with probability at least 2/3 and has the claimed worst-case query complexity. □

2.4 A Lower Bound for Monotonicity Testing of Functions $f : [n]^2 \to \{0,1\}$

Theorem 2.14. *Every 1-sided error nonadaptive ε-test for monotonicity of functions $f : [n]^2 \to \{0,1\}$ must make $\Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ queries.*

Proof. By standard arguments, it suffices to specify a distribution on functions which are ε-far from monotone, such that every deterministic algorithm must make $\Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ queries to find a violated pair of domain points with probability at least 2/3 on inputs from this distribution. Our distribution is uniform over a *hard* set of functions, each of which corresponds to a hexagon. All pairs violated by a function in the set are contained in the corresponding hexagon. In this proof, we identify each point $(i,j) \in [n]^2$ in the domain of the input function with a point (x,y) with coordinates $x = i/n$ and $y = j/n$ in the Cartesian plane. We assume that n is large enough, so that the area of hexagons in $[0,1]^2$ we consider is a good approximation of the fraction of points of the form $(i/n, j/n)$ they contain.

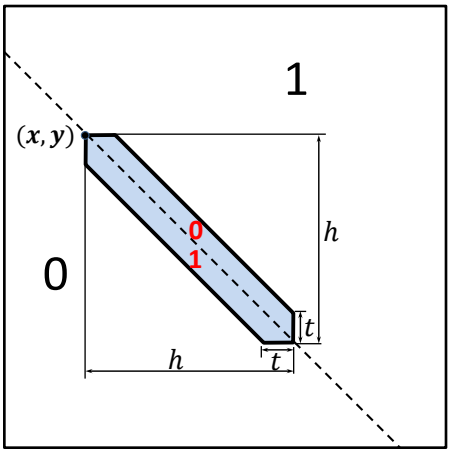

Figure 2.1: An illustration for Definition 2.5: a hexagon $H^{\mathbf{x},\mathbf{y}}_{t,h}$ and hexagon function $f^{\mathbf{x},\mathbf{y}}_{t,h}$.

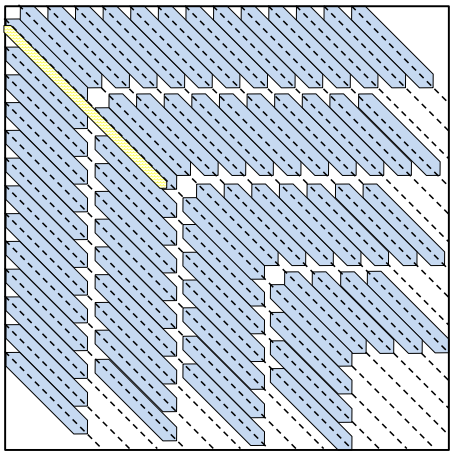

Figure 2.2: Packing of hexagons (one level of hexagons and a hexagon from the next level).

Definition 2.5. *As illustrated in Figure 2.1,* hexagon $H^{\mathbf{x},\mathbf{y}}_{t,h}$ *with the upper left corner* $(\mathbf{x},\mathbf{y})$*, thickness* t*, and height* h *consists of points* (x,y) *satisfying:*

$$\begin{aligned} \mathbf{x}+\mathbf{y}-t &< x+y < \mathbf{x}+\mathbf{y}+t, \\ \mathbf{x} &< x < \mathbf{x}+h, \\ \mathbf{y}-h &< y < \mathbf{y}. \end{aligned}$$

Now define the hexagon function $f^{\mathbf{x},\mathbf{y}}_{t,h} : [0,1]^2 \to \{0,1\}$*:*

$$f^{\mathbf{x},\mathbf{y}}_{t,h}(x,y) = \begin{cases} 1 - d_{\mathbf{x}+\mathbf{y}}(x,y) & \text{if } (x,y) \in H^{\mathbf{x},\mathbf{y}}_{t,h}, \\ d_{\mathbf{x}+\mathbf{y}}(x,y) & \text{otherwise,} \end{cases} \qquad \text{where } d_a(x,y) = \begin{cases} 1 & \text{if } x+y \geq a, \\ 0 & \text{otherwise.} \end{cases}$$

The reason we use hexagons as opposed to tilted rectangles in our construction is that we would like to make sure that $f^{\mathbf{x},\mathbf{y}}_{t,h}$ can only violate pairs of points contained in the corresponding hexagon $H^{\mathbf{x},\mathbf{y}}_{t,h}$.

Next, we specify the hexagon functions included in the hard set. The hard set is a union of *levels*, and each level is a union of *diagonals*. (See Figure 2.2.) Functions of the same level are defined by equal non-overlapping hexagons. Levels are indexed by integers starting from 0. Let $t_0 = \sqrt{\varepsilon}/\log\frac{1}{\varepsilon}$. In level i, hexagons have thickness $t_i = t_0 \cdot 2^{-i}$ and height $h_i = 2\varepsilon/t_i$. We include levels $i \geq 0$ with height $h_i \leq 1/2$.

Each level i is further subdivided into *diagonals* indexed by an integer $j \in (1, 1/t_i)$. In diagonal j, the coordinates of the upper left corners of the hexagons satisfy $\mathbf{x} + \mathbf{y} = (2j+1)t_i$. It remains to specify the functions that are contained in each diagonal. Intuitively, we pack the maximum possible number of hexagons into each diagonal, while leaving sufficient separation between them. If $\mathbf{x} + \mathbf{y} = (2j+1)t_i \leq 1$, we restrict $\mathbf{x}$ to integer multiples of $h_i + \sqrt{\varepsilon}$, and for $\mathbf{x} + \mathbf{y} = (2j+1)t_i > 1$, we restrict $1 - \mathbf{y}$ to integer multiples of $h_i + \sqrt{\varepsilon}$. In both cases, the projections of the hexagons of these functions onto an axis form disjoint intervals of length h_i that are separated by gaps of length $\sqrt{\varepsilon}$. Finally, only if a hexagon $H^{\mathbf{x},\mathbf{y}}_{t,h}$ is fully contained in $[0,1]^2$, the corresponding function $f^{\mathbf{x},\mathbf{y}}_{t,h}$ is included in the hard set.

Lemma 2.15. *The hard set contains $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ functions, each of which is ε-far from monotone.*

Proof. First, we argue that all functions in the hard set are ε-far from monotone. Define a bijection $b_a(x,y) = (a-y, a-x)$ (it is the reflection over the line $x+y=a$). For every $(x,y) \in H^{\mathbf{x},\mathbf{y}}_{t,h}$, the pair $((x,y), b_{\mathbf{x}+\mathbf{y}}(x,y))$ is violated by $f^{\mathbf{x},\mathbf{y}}_{t,h}$. Hence, the distance from $f^{\mathbf{x},\mathbf{y}}_{t,h}$ to monotone is half the fraction of points contained in the hexagon, but not on the line $x+y=a$. Ignoring low-order terms, it is one half of the area of the hexagon $H^{\mathbf{x},\mathbf{y}}_{t,h}$, that is, $h \cdot t - t^2/2$. In our construction, for all hexagons $h \cdot t = 2\varepsilon$ and $t < h$, so $h \cdot t - t^2/2 > \varepsilon$. Therefore, all functions in the hard set are ε-far from monotone.

Next, we give a lower bound on the number of functions in the hard set. The number of levels is $\frac{1}{2}\log\frac{1}{\varepsilon} + \Theta(\log\log\frac{1}{\varepsilon})$. The area of each hexagon is less than 4ε. We pack hexagons with gaps that are much smaller than the thickness of a hexagon, and the unused space in each diagonal is much smaller then the average length of a diagonal, so they occupy more than half the area of $[0,1]^2$. Thus, the number of functions in a level is at least $1/8\varepsilon$. Consequently, we have $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ functions in our hard set. □

Claim 2.16. *Each pair $((x,y),(x',y'))$ is violated by at most one function in the hard set.*

Proof. Function $f^{\mathbf{x},\mathbf{y}}_{t,h}$ violates only pairs which are in the hexagon $H^{\mathbf{x},\mathbf{y}}_{t,h}$. Functions of the same level cannot have violated pairs in common because their corresponding hexagons are disjoint.

Now we argue that functions from different levels cannot violate the same pair. The hexagons of functions in level i, diagonal j are subsets of the strip $\{(x,y) \mid 2jt_i < x+y < 2(j+1)t_i\}$. Each defining strip of level i is divided into two narrower strips in level $i+1$. Suppose two hexagons, H_1 and H_2, overlap, and H_1 has higher thickness (i.e., belongs to a lower level). Then the strip of H_2 is fully contained either in the lower or in the upper half of the strip of H_1. All the points in the overlap of H_1 and H_2 are assigned the same value by the function corresponding to H_1. Therefore, functions corresponding to different hexagons cannot violate the same pair of points. □

Lemma 2.17. *Every set Q of (queried) points contains violated pairs for at most $|Q|-1$ hard functions.*

Proof. Fix a set Q of queried points. Suppose it contains violated pairs for k different hard functions. Select one violated pair for each of k functions. Form an (undirected) *representative graph* on vertex set Q by adding edges corresponding to the k violated pairs. Each edge in the graph represents a hard function and is associated with its pair (i,j), where i is the level and j is the diagonal. We will prove that the representative graph is acyclic. It implies that the number of nodes $|Q| \geq k+1$.

Claim 2.18. *Consider a path P in the representative graph. Every edge in P is associated with a different pair (i,j), where i is the level and j is the diagonal.*

Proof. First, we show that every edge in the representative graph is short. Consider a pair $((x,y),(x',y'))$ violated by a hard function which belongs to some level i and a diagonal j of that level. Then the points in the pair must be comparable: $x \le x'$ and $y \le y'$. Both points must belong to the associated strip. Consequently, $2jt_i < x+y$ and $x'+y' < 2(j+1)t_i$. Therefore, $\|(x',y')-(x,y)\|_1 = x'-x+y'-y < 2(j+1)t_i - 2jt_i = 2t_i$.

Next, consider a path P in the representative graph. The L_1-length of a path, denoted $L_1(P)$, is the L_1 distance between its endpoints. We claim that if every edge on the path is associated with a different pair (i,j) then $L_1(P) < 2\sqrt{\varepsilon}$. Recall that strips associated with diagonals of the same level are disjoint. Each strip of level i contains two strips of level $(i+1)$ and is disjoint from other strips of level $(i+1)$. Consequently, P remains within one strip of level 0. Each strip of level 0 contains 2^i strips of level $i \ge 1$. Recall that there are fewer than $\log\frac{1}{\varepsilon}$ levels, and that an edge in level i has L_1-length at most $2t_i = 2t_0/2^i$. Each edge is associated with a different pair (i,j), implying $L_1(P) \le \sum_i 2^i \cdot (2t_0/2^i) < 2t_0 \log\frac{1}{\varepsilon} = 2\sqrt{\varepsilon}$.

Finally, recall that the L_1 distance between hexagons of the same level and diagonal is at least $2\sqrt{\varepsilon}$. Suppose for the sake of contradiction that the selection graph has a path that contains two edges associated with the same pair (i,j), and let P^* be the shortest such path. Let P be P^* with the last edge removed. Then every edge in P is associated with a different pair (i,j), so $L_1(P) < 2\sqrt{\varepsilon}$. But the first and the last edges of P^* represent different hard functions, so they must belong to different hexagons of the strip for level i, diagonal j. So, $L_1(P) > 2\sqrt{\varepsilon}$, a contradiction. □

Assume to the contrary that we have a cycle in the representative graph. Consider an edge (u,v) of the cycle that is associated with the lowest level i (i.e., the largest thickness t_i). This edge travels from one diagonal strip of level $i+1$ to another, say from S_{i+1} to S'_{i+1}. As we proceed further along the cycle, we stay within S'_{i+1} until we traverse another edge of level i (recall that the cycle has no edges of levels lower than i). But this edge is associated with the same pair (i,j) as (u,v), which contradicts Claim 2.18. Thus, the representative graph is acyclic, which completes the proof of the lemma. □

The theorem follows from Lemmas 2.15 and 2.17 because a deterministic test that succeeds on the hard distribution with probability at least 2/3, must query violated pairs for at least 2/3 of the hard functions. □

3 Approximation of L_1-Distance to Monotonicity

3.1 Approximation of L_1-Distance to Monotonicity for Boolean Functions

In this subsection we prove Theorem 3.1 which shows how to approximate L_1-distance to monotonicity for functions $f\colon [n] \to \{0,1\}$.

Theorem 3.1. *Let $f : [n] \to \{0,1\}$ and $s \in [n]$. Consider a random subset S of $[n]$ where each element of $[n]$ is chosen independently with probability s/n. Let $\tilde{\epsilon}(S) = L_1(f|_S, \mathcal{M})/s$. Then*

$$\begin{aligned} d_{\mathcal{M}}(f) - \sqrt{2d_{\mathcal{M}}(f)/s} \le \quad & \mathbb{E}[\tilde{\epsilon}(S)] && \le d_{\mathcal{M}}(f); \\ & \mathbb{V}\mathrm{ar}[\tilde{\epsilon}(S)] && = O\left(d_{\mathcal{M}}(f)/s\right). \end{aligned}$$

Proof. To simplify the presentation, in this proof we let $\epsilon_f = d_{\mathcal{M}}(f)$ and $\tilde{f} = f|_S$.

Let $Z(S) = L_1(\tilde{f}, \mathcal{M})$. To prove the theorem it suffices to show that there exist functions $X(S)$ and $Y(S)$ such that

1. $X(S) \le Z(S) \le Y(S)$;
2. $\mathbb{E}[X(S)] \ge \epsilon_f \cdot s - \sqrt{2\epsilon_f \cdot s}$ and $\mathbb{E}[Y(S)] = \epsilon_f \cdot s$;
3. $\mathbb{V}\mathrm{ar}[X(S)] = O\left(\epsilon_f \cdot s\right)$ and $\mathbb{V}\mathrm{ar}[Y(S)] \le \epsilon_f \cdot s$.

Then by linearity of expectation and items 1 and 2 above, $\epsilon_f \cdot s - \sqrt{2\epsilon_f \cdot s} \le \mathbb{E}[Z] \le \epsilon_f \cdot s$. The variance $\mathbb{V}\mathrm{ar}[Z(S)] = O(\epsilon_f \cdot s)$ by items 1-3 above and Lemma 3.7, proved at the end of Section 3.1, where the lemma is used with $\delta = O\left(\epsilon_f \cdot s\right)$. Then the stated bounds on the expectation and variance of $\tilde{\epsilon}(S) = Z(S)/s$ follow.

We will define functions $X(S)$ and $Y(S)$ after we recall a characterization of $L_1(f, \mathcal{M})$ from [24, 32]. Let G_f be a violation graph of f, namely, a directed graph with vertex set $[n]$ whose edges correspond to violated pairs (x, y), i.e., pairs (x, y) for which $x < y$, but $f(x) > f(y)$. Let VC_f be an (arbitrary) minimum vertex cover of G_f and let MM_f be a maximum matching in G_f.

Lemma 3.2 ([24, 32])**.** *For a Boolean function f, the distance $L_1(f, \mathcal{M}) = |VC_f| = |MM_f|$.*

Definition 3.1 (Upper bound)**.** *The random variable $Y(S) = |VC_f \cap S|$.*

Lemma 3.3. *The random variable $Y(S)$ satisfies: $Z(S) \le Y(S)$, $\mathbb{E}[Y(S)] = \epsilon_f \cdot s$, and $\mathbb{V}\mathrm{ar}[Y(S)] \le \epsilon_f \cdot s$.*

Proof. First, we show that $Z(S) \le Y(S)$, that is, $L_1(\tilde{f}, \mathcal{M}) \le |\mathrm{VC}_f \cap S|$. By Lemma 3.2, $L_1(\tilde{f}, \mathcal{M}) = |\mathrm{VC}_{\tilde{f}}|$. Note that each pair violated by $\tilde{f}$ is also violated by f. Therefore, $\mathrm{VC}_f \cap S$ is a (not necessarily minimum) vertex cover for the violation graph $G_{\tilde{f}}$. Thus, $L_1(\tilde{f}, \mathcal{M}) = |\mathrm{VC}_{\tilde{f}}| \le |\mathrm{VC}_f \cap S|$, as desired.

To analyze the expectation and variance of $Y(S)$, recall from Lemma 3.2 that $|\mathrm{VC}_f| = L_1(f, \mathcal{M}) = \ell$. Each element of VC_f appears in S independently with probability s/n. Thus, the random variable $Y(S) = |\mathrm{VC}_f \cap S|$ follows the binomial distribution with mean $|\mathrm{VC}_f| \cdot s/n = \epsilon_f \cdot s$ and variance $|\mathrm{VC}_f| \cdot \frac{s}{n}(1 - \frac{s}{n}) \le \epsilon_f \cdot s$. This completes the proof of the lemma. □

Consider the maximum matching MM_f. In the rest of the proof, we denote $|\mathrm{MM}_f| = \epsilon_f \cdot n$ by ℓ. By definition, MM_f consists of ℓ edges of the form (a, b), where $a < b$, $f(a) = 1$ and $f(b) = 0$. Let $V(\mathrm{MM}_f)$ denote the endpoints of edges in MM_f. Let $a_1 < a_2 < \cdots < a_\ell$ be the vertices in $V(\mathrm{MM}_f)$ on which f evaluates to 1, sorted in increasing order, and $b_1 < b_2 < \cdots < b_\ell$ denote the vertices in $V(\mathrm{MM}_f)$ on which f evaluates to 0, also sorted in increasing order. Observe that $a_i < b_i$ for all $i \in [\ell]$ because each of $b_1, \ldots, b_i$ is matched to a smaller number in MM_f.

Definition 3.2 (Guaranteed edges, lower bound)**.** Guaranteed edges *are pairs (a_i, b_j) such that $a_i, b_j \in V(MM_f) \cap S$ and $i \le j$ for some $i, j \in [\ell]$. Let $\widetilde{MM}(S)$ denote a maximum matching that consists of guaranteed edges. The random variable $X(S) = |\widetilde{MM}(S)|$.*

Note that $\tilde{f}$ violates all guaranteed edges because $a_i < b_i < b_j$ for all $i < j$ and $f(a_i) = 1, f(b_j) = 0$.

Lemma 3.4. *The random variable $X(S)$ satisfies*

$$\textit{(a) } X(S) \le Z(S), \textit{ (b) } \mathbb{E}[X(S)] \ge \epsilon_f \cdot s - \sqrt{2\epsilon_f \cdot s} \textit{ and (c) } \mathbb{V}\mathrm{ar}[X(S)] \le c_2 \epsilon_f \cdot s.$$

Proof. Item (a) holds because by Lemma 3.2, $Z(S) = L_1(\tilde{f}, \mathcal{M}) = |\mathrm{MM}_{\tilde{f}}| \ge |\widetilde{\mathrm{MM}}(S)| = X(S)$. The inequality follows from the fact that $\widetilde{\mathrm{MM}}(S)$ is a maximum matching consisting of guaranteed edges, which are edges violated by $\tilde{f}$, while $\mathrm{MM}_{\tilde{f}}$ is a maximum matching consisting of any violated edges of $\tilde{f}$, not necessarily guaranteed edges.

Next, we analyze the distribution of $X(S)$. Let $X'(S) = |V(\mathrm{MM}_f) \cap S|$ and $U(S)$ be the number of elements of $V(\mathrm{MM}_f) \cap S$ that are left unmatched by the maximum matching $\widetilde{\mathrm{MM}}(S)$ that consists of guaranteed edges. Then $X(S) = [X'(S) - U(S)]/2$. The random variable $X'(S)$ has a similar distribution to $Y(S)$, i.e., $E[X'(S)] = 2\epsilon_f \cdot s$ and $\mathbb{V}\mathrm{ar}[X'(S)] < 2\epsilon_f \cdot s$.

To understand the distribution of $U(S)$, we express it using random variables that describe a one-dimensional random walk. For $i \in [\ell]$, we define

$$g_i = \begin{cases} 1 & \text{if } \{a_i, b_i\} \cap S = \{b_i\}; \\ -1 & \text{if } \{a_i, b_i\} \cap S = \{a_i\}; \\ 0 & \text{if } |\{a_i, b_i\} \cap S| \ne 1. \end{cases}$$

We also define the cumulative sums: $p_0 = 0$, $p_i = p_{i-1} + g_i$. We set $m(S) = \max_{i=0}^{\ell} p_i$, while $p(S) = p_\ell$, i.e., the largest and the last cumulative sums. In other words, $p(S)$ can be viewed as the position of the following random walk after ℓ steps: it starts at $p_0 = 0$, and at each step i, it goes up by one if $\{a_i, b_i\} \cap S = \{b_i\}$,

down by 1 if $\{a_i, b_i\} \cap S = \{a_i\}$, and stays in place otherwise. The random variable $m(S)$ is the maximum of this random walk. Note that $\Pr[g_i = 1] = \Pr[g_i = 0] = \frac{s}{n}(1 - \frac{s}{n})$. Next we express the random variable $U(S)$ in terms of the maximum value and the final position of this random walk.

Claim 3.5. *$U(S) \leq 2m(S) - p(S)$.*

Proof. We will construct a matching of guaranteed edges that leaves at most $2m(S) - p(S)$ elements of $V(\mathrm{MM}_f) \cap S$ unmatched. While constructing the matching, we perform operations on $g_1, \ldots, g_\ell$ that make the sequence shorter while the maximum and the final position of the corresponding walk remain the unchanged.

Suppose that $g_i = 0$ for some i. This means that either $a_i, b_i \in [n] - S$ or $a_i, b_i \in S$. We select the guaranteed edge (a_i, b_i) for our matching, and remove g_i from the sequence. This does not alter the maximum and the final position of the corresponding walk.

Suppose that for some i, there are two consecutive values $g_i = -1, g_{i+1} = 1$. Then $a_i, b_{i+1} \in S$. We add the guaranteed edge (a_i, b_{i+1}) to our matching and remove g_i, g_{i+1} from the sequence. Again, the maximum and the final position of the walk do not change.

Let ℓ' be the length of the sequence after the operations above cannot be applied anymore. Then all g_i's are 1 or -1 but a 1 cannot follow a -1. Thus, $g_i = 1$ for $i = 1, \ldots, m(S)$ and $g_i = -1$ for $i = m(S)+1, \ldots, \ell'$. That is, $p(S) = m(S) - (\ell' - m(S)) = 2m(S) - \ell'$, and hence $\ell' = 2m(S) - p(S)$. Clearly, the constructed matching leaves at most ℓ' members of $V(\mathrm{MM}_f) \cap S$ unmatched. So, in the maximum matching of guaranteed edges, the number of unmatched members of $V(\mathrm{MM}_f) \cap S$ is $U(S) \leq 2m(S) - p(S)$. □

Claim 3.6. *$\Pr[m(S) \geq z] \leq \Pr[|p(S)| \geq z]$ for all $z \in [\ell]$.*

Proof. Fix $z \in [\ell]$. Let E_i be the event that $p_i = z$ and $p_j < z$ for $j \in [i-1]$. The event "$m(S) \geq z$" is a disjoint union of the events E_i for $i \in [\ell]$. By symmetry, $\Pr[p(S) \geq z \mid E_i] \geq 1/2$. Thus,

$$\Pr[|p(S)| \geq z] = 2\Pr[p(S) \geq z] = 2\sum_{i=1}^{\ell} (\Pr[p(S) > z \mid E_i] \cdot \Pr[E_i]) \geq \sum_{i=1}^{\ell} \Pr[E_i] = \Pr[m(S) > z],$$

as required. □

Now we are ready to bound the expectation of $U(S)$. By Claims 3.5 and 3.6, the expectation of U is

$$\mathbb{E}[U(S)] \leq 2\mathbb{E}[m(S)] \leq 2\mathbb{E}[|p(S)|].$$

Recall that $p(S)$ is the sum of $\ell = \epsilon_f \cdot n$ independent zero-mean variables g_i that take values in $\{-1, 0, 1\}$, each of which satisfies $\Pr[g_i = 1] = \Pr[g_i = -1] \leq s/n$. Therefore, $\mathbb{E}[g_i^2] = \Pr[g_i^2 = 1] \leq 2s/n$. By Jensen's inequality,

$$\mathbb{E}^2[|p(S)|] \leq \mathbb{E}[(p(S))^2] = \mathbb{E}[(\sum_{i=1}^{\ell} g_i)^2] = \mathbb{E}[\sum_{i=1}^{\ell} g_i^2] \leq \ell \cdot 2s/n = 2\epsilon_f \cdot s.$$

Thus, $\mathbb{E}[U(S)] \leq 2\mathbb{E}[|p(S)|] \leq 2\sqrt{2\epsilon_f \cdot s}$. Recall that $X(S) = [X'(S) - U(S)]/2$ and $E[X'(S)] = 2\epsilon_f \cdot s$. Therefore, $\mathbb{E}[X(S)] = \mathbb{E}[X'(S)]/2 - \mathbb{E}[U(S)]/2 \geq \epsilon_f \cdot s - \sqrt{2\epsilon_f \cdot s}$, completing the proof of item (b).

Next, we bound the variance of $U(S)$:

$$\mathbb{V}\mathrm{ar}[U(S)] \leq \mathbb{E}[(U(S))^2] \leq 4\mathbb{E}[(m(S))^2] \leq 4\mathbb{E}[(p(S))^2] \leq 8\epsilon_f \cdot s.$$

Since $X(S) = [X'(S) - U(S)]/2$ and $\mathbb{V}\mathrm{ar}[X'(S)] \leq 2\epsilon_f \cdot s$, we get that $\mathbb{V}\mathrm{ar}[X(S)] = O(\epsilon_f \cdot s)$, completing the proof of item (c). □

This completes the proof of Theorem 3.1. □

The following lemma was used in the proof of Theorem 3.1.

Lemma 3.7 (Sandwich lemma)**.** *Let X, Y, Z be discrete random variables with means μ_X, μ_Y, μ_Z, satisfying $X \leq Z \leq Y$. Suppose there exists $\delta \geq 0$ such that*

$$\mu_Y - \mu_X \leq \sqrt{\delta}, \quad \mathbb{V}\mathrm{ar}[X] \leq \delta \quad \textit{and} \quad \mathbb{V}\mathrm{ar}[Y] \leq \delta.$$

Then $\mathbb{V}\mathrm{ar}[Z] \leq 3\delta$.

Proof. Consider random variables $X' = X - \mu_Z, Z' = Z - \mu_Z$, and $Y' = Y - \mu_Z$. Then $X' \leq Z' \leq Y'$ and the expectations of these random variables are $\mathbb{E}[X'] = \mu_X - \mu_Z$, $\mathbb{E}[Y'] = \mu_Y - \mu_Z$ and $\mathbb{E}[Z'] = 0$.

Let V^+ be the set of positive values and V^- be the set of negative values in the support of Z'. Then

$$\mathbb{V}\mathrm{ar}[Z] = \mathbb{V}\mathrm{ar}[Z'] = \mathbb{E}[(Z')^2] = \sum_{v \in V^+} \Pr[Z' = v] \cdot v^2 + \sum_{v \in V^-} \Pr[Z' = v] \cdot v^2 \leq \mathbb{E}[(Y')^2] + \mathbb{E}[(X')^2].$$

The inequality above holds because $X' \leq Z' \leq Y'$. Next, we rewrite the resulting expectations in terms of variances and squared means of the corresponding variables:

$$\begin{aligned} \mathbb{E}[(Y')^2] + \mathbb{E}[(X')^2] &= \mathbb{V}\mathrm{ar}[Y'] + \mu_{Y'}^2 + \mathbb{V}\mathrm{ar}[X'] + \mu_{X'}^2 \\ &= \mathbb{V}\mathrm{ar}[Y'] + \mathbb{V}\mathrm{ar}[X'] + (\mu_Y - \mu_Z)^2 + (\mu_Z - \mu_X)^2 \\ &\leq \mathbb{V}\mathrm{ar}[Y] + \mathbb{V}\mathrm{ar}[X] + (\mu_Y - \mu_X)^2 \\ &\leq 3\delta. \end{aligned}$$

The first inequality above holds because $a^2 + b^2 \leq (a+b)^2$ for all nonnegative a, b. The final inequality holds by the hypothesis in the statement of the lemma. □

Before we complete the proof of the first part of Theorem 1.4 we note that the results above can be extended to real-valued functions.

Corollary 3.8. *The guarantees of Theorem 3.1 also hold for real-valued functions $f\colon [n] \to [0,1]$.*

Proof. Note that both the additive error $\delta_A(d_{\mathcal{M}}(f))$ and standard deviation $\sigma_A(d_{\mathcal{M}}(f))$ in Theorem 3.1 are bounded by a concave function $O(\sqrt{d_{\mathcal{M}}(f)/s})$. Hence, we can apply Lemma 2.3 to extend Theorem 3.1 to real-valued functions. □

3.2 Adaptive distance estimation

Now we are ready to complete the proof of the first part of Theorem 1.4. The claimed bound follows from the analysis of the Algorithm 4, which uses adaptive queries to refine the approximation to the distance. The adaptive sampling technique we use is very natural and was used in a different context (join size estimation in databses) by Lipton and Naughton [46].

Algorithm 4: Adaptive distance estimation algorithm.

input : parameter δ; oracle access to $f\colon [n] \to [0,1]$.

1 Set $i = 1, s_1 = c_1/\delta$, $t_1 = \delta$
2 **repeat**
3 Pick a random sample $S_i \subseteq [n]$ of size s_i
4 **if** $d_{\mathcal{M}}(f|_{S_i}) < t_i$ **return** $d_{\mathcal{M}}(f|_{S_i})$
5 $s_{i+1} = 2s_i, t_{i+1} = 2t_i, i = i+1$

Lemma 3.9 (Theorem 1.4 for $d = 1$)**.** *For every function $f\colon [n] \to [0,1]$ Algorithm 4 returns a value d such that $|d - d_{\mathcal{M}}(f)| \leq \delta$ and queries $O\left(\max\left(\frac{d_{\mathcal{M}}(f)}{\delta^2}, \frac{1}{\delta}\right)\right)$ points with constant probability.*

Proof. **Query complexity.** Consider the first iteration i^* when $t_{i^*} = t_1 2^{i^*-1} > 2d_{\mathcal{M}}(f)$ and hence $s_{i^*} = s_1 2^{i^*-1}$. At this iteration we have $\Pr[d_{\mathcal{M}}(f|_{S_{i^*}}) \geq t_{i^*}] \leq \Pr[d_{\mathcal{M}}(f|_{S_{i^*}}) \geq 2d_{\mathcal{M}}(f)]$. From Corollary 3.8 by using Chebyshev inequality it follows that there exists a constant α such that for a random sample S of size s

$$\Pr\left[|d_{\mathcal{M}}(f|_S) - d_{\mathcal{M}}(f)| > \alpha c\sqrt{\frac{d_{\mathcal{M}}(f)}{s}}\right] \leq \frac{1}{c^2}. \tag{2}$$

Applying this to S_{i^*} and taking $c = \sqrt{d_{\mathcal{M}}(f)s_i}/\alpha$ we have:

$$\Pr[|d_{\mathcal{M}}(f|_{S_{i^*}}) - d_{\mathcal{M}}(f)| \geq d_{\mathcal{M}}(f)] \leq \frac{\alpha^2}{d_{\mathcal{M}}(f)s_{i^*}}$$

We have $d_{\mathcal{M}}(f)s_{i^*} = d_{\mathcal{M}}(f)s_1 2^{i^*-1} > \frac{2(d_{\mathcal{M}}(f))^2 s_1}{t_1} = \frac{2(d_{\mathcal{M}}(f))^2 c_1}{\delta^2}$.

Case 1. If $d_{\mathcal{M}}(f) > \delta/2$ then $d_{\mathcal{M}}(f)s_{i^*} > \frac{c_1}{2}$ and hence:

$$\Pr[|d_{\mathcal{M}}(f|_{S_{i^*}}) - d_{\mathcal{M}}(f)| \geq d_{\mathcal{M}}(f)] \leq \frac{2\alpha^2}{c_1}$$

Thus, for a large enough constant c_1 Algorithm 4 terminates after iteration i^* with constant probability.

Note that $t_{i^*} = t_1 2^{i^*-1} \leq 4d_{\mathcal{M}}(f)$ by the choice of i^* and so $i^* \leq \log_2\left(\frac{8d_{\mathcal{M}}(f)}{t_1}\right)$. The overall sample complexity in this case is thus $\sum_1^{i^*} s_i = \sum_1^{i^*} \frac{c_1 2^{i-1}}{\delta} \leq \frac{1}{\delta} c_1 2^{i^*} \leq \frac{8c_1 d_{\mathcal{M}}(f)}{\delta t_1} = O\left(\frac{d_{\mathcal{M}}(f)}{\delta^2}\right)$.

Case 2. If $d_{\mathcal{M}}(f) \leq \delta/2$ then $t_{i^*} = t_1 = \delta$ and thus the sample complexity is $s_1 = O(1/\delta)$.

Error probability. Let $t^* = d_{\mathcal{M}}(f)/2$. We want to claim that if $t_i < t^*$ then the probability of termination is small, while if $t_i \geq t^*$ then we have good enough error. First, let's analyze the termination probability for $t_i < t^*$. We have:

$$\Pr[\exists i \colon t_i < t^*, d_{\mathcal{M}}(f|_S) < t_i] \leq \sum_{i \colon t_i < t^*} \Pr[d_{\mathcal{M}}(f|_S) < t_i] = \sum_{i \colon t_i < t^*} \Pr[d_{\mathcal{M}}(f) - d_{\mathcal{M}}(f|_S) > d_{\mathcal{M}}(f) - t_i]$$

$$\leq \sum_{i \colon t_i < t^*} \Pr[|d_{\mathcal{M}}(f) - d_{\mathcal{M}}(f|_S)| > d_{\mathcal{M}}(f) - t_i] \leq \sum_{i \colon t_i < t^*} \Pr\left[|d_{\mathcal{M}}(f) - d_{\mathcal{M}}(f|_S)| > \frac{d_{\mathcal{M}}(f)}{2}\right].$$

If $t^* < \delta$ then the probability of early termination is zero so we can assume $d_{\mathcal{M}}(f) \geq 2\delta$. Taking $c = \frac{\sqrt{s_i d_{\mathcal{M}}(f)}}{2\alpha}$ in (2) we have that the probability of termination with $t_i < t^*$ is at most:

$$\sum_{i \colon t_i < t^*} \frac{4\alpha^2}{s_i d_{\mathcal{M}}(f)} = \frac{4\alpha^2}{d_{\mathcal{M}}(f)} \sum_{i \colon t_i < t^*} \frac{\delta}{c_1 2^i} \leq \frac{8\alpha^2\delta}{c_1 d_{\mathcal{M}}(f)} \leq \frac{4\alpha^2}{c_1}.$$

Now if $t_i \geq t^*$ we have $s_i \geq \frac{t^* s_1}{t_1} = \frac{d_{\mathcal{M}}(f)c_1}{2\delta^2}$ and thus by (2) taking $c = \frac{\sqrt{c_1/2}}{\alpha}$ we have that $\Pr[|d_{\mathcal{M}}(f) - d_{\mathcal{M}}(f|_S)| > \delta] \leq \frac{2\alpha^2}{c_1}$. □

4 L_1-Testing of the Lipschitz Property

4.1 Characterization of the L_1 distance to the Lipschitz property

In this section, we recall some basic definitions from [43, 24, 15] and present our characterization of the L_1 distance to the Lipschitz property.

Definition 4.1 (Violated pair, violation score, violation graph [43, 24, 15])**.** *Let $f : D \to \mathbb{R}$ be a function with a finite domain D, equipped with distance d_D, and let x, y be points in D. The pair (x, y) is* violated *by f if $|f(x) - f(y)| > d_D(x, y)$. The* violation score *of (x, y), denoted $\mathrm{vs}_f(x, y)$, is $|f(x) - f(y)| - d_D(x, y)$ if it is violated and 0 otherwise. The* violation graph *G_f of function f is the weighted undirected graph with vertex set D, which contains an edge (x, y) of weight $\mathrm{vs}_f(x, y)$ for each violated pair of points $x, y \in D$.*

Let $\mathcal{L}ip$ be the set of Lipschitz functions $f : D \to \mathbb{R}$. The following lemma characterizes $L_1(f, \mathcal{L}ip)$, the absolute L_1 distance from f to $\mathcal{L}ip$, in terms of matchings in G_f. The weight of a matching M is denoted by $\mathrm{vs}_f(M)$.

Lemma 4.1 (Characterization of L_1 distance to Lipschitz)**.** *Consider a function $f : D \to \mathbb{R}$, where D is a finite metric space. Let M be a maximum weight matching in G_f. Then $L_1(f, \mathcal{L}ip) = \mathrm{vs}_f(M)$.*

Proof. It has already been observed that $L_1(f, \mathcal{L}ip) \geq \mathrm{vs}_f(M)$ for any matching M in G_f [4, Lemma 2.3]. To show the second needed inequality, let g be a Lipschitz function closest to f, that is, satisfying $L_1(f, g) = L_1(f, \mathcal{L}ip)$. We will construct a matching M in G_f such that $L_1(f, g) = \mathrm{vs}_f(M)$. We start by constructing a matching M' of weight $L_1(f, g)$ in a bipartite graph related to G_f, and later transform it to the matching of the same weight in G_f.

Definition 4.2. *For each operation $op \in \{<, >, =\}$, define a point set $V_{op} = \{x \in D \mid f(x)\ op\ g(x)\}$.*

Definition 4.3 (bipartite graph BG_f)**.** *BG_f is a weighted bipartite graph. The nodes of BG_f are points in D, except that we make two copies, called $x_\leq$ and $x_\geq$, of every point $x \in V_=$. Nodes are partitioned into $V_\geq$ and $V_\leq$, where part $V_\geq = V_> \cup \{x_\geq \mid x \in V_=\}$ and, similarly, part $V_\leq = V_< \cup \{x_\leq \mid x \in V_=\}$. The set BE_f of edges of BG_f consists of pairs $(x, y) \in V_> \times V_\leq \cup V_\geq \times V_<$, such that*

$$g(x) - g(y) = d_D(x, y).$$

Metric d_D is extended to duplicates: $d_D(x_\leq, y) = d_D(x_\geq, y) = d_D(x, y)$, and the weights $\mathrm{vs}_f(x, y)$ are defined as before.

Observe that for every edge (x, y) in BG_f, by definition,

$$f(x) \geq g(x) > g(y) \geq f(y) \text{ and, moreover,} \tag{3}$$

$$\mathrm{vs}_f(x, y) = f(x) - f(y) - d_D(x, y) = f(x) - f(y) - (g(x) - g(y)) = |f(x) - g(x)| + |f(y) - g(y)|. \tag{4}$$

Later, we will show that BG_f contains a matching M' that matches every $x \in V_< \cup V_>$. By (4),

$$\begin{aligned} \mathrm{vs}_f(M') &= \sum_{(x,y)\in M'} \mathrm{vs}_f(x, y) = \sum_{x \text{ is matched in } M'} |f(x) - g(x)| \\ &= \sum_{x \in V_< \cup V_>} |f(x) - g(x)| = \|f - g\|_1 = L_1(f, \mathcal{L}ip). \end{aligned}$$

Next we show how to transform a matching M' in BG_f into a matching M in G_f, such that $\mathrm{vs}_f(M) = \mathrm{vs}_f(M')$. We first remove from M' all edges of weight 0. Then we replace each $x_\leq$ and $x_\geq$ with x. Finally, we handle the cases when both $x_\leq$ and $x_\geq$ were replaced with x. If this happens, M' contains matches $(y, x_\leq)$ and $(x_\geq, z)$, and we replace them with a single match (y, z). By (3), $f(y) > f(x) > f(z)$. Since (y, x) and (x, z) are violated pairs, $\mathrm{vs}_f(y, z) = \mathrm{vs}_f(y, x) + \mathrm{vs}_f(x, z)$. Thus, $\mathrm{vs}_f(M) = \mathrm{vs}_f(M')$, as claimed.

Now it suffices to show that BG_f contains a matching M' that matches every $x \in V_< \cup V_>$. By Hall's marriage theorem, it enough to show that whenever $A \subseteq V_<$ or $A \subseteq V_>$ we have $|A| \leq |N(A)|$, where $N(A)$ is the set of all neighbors of A in BG_f. Suppose to the contrary that the marriage condition does not hold for some set, and w.l.o.g. suppose it is a subset of $V_>$. Let $A \subset V_>$ be the largest set satisfying $|A| > |N(A)|$.

Claim 4.2. *If $x, y \in V_>$ and $g(x) - g(y) = d_D(x, y)$ then $N(y) \subseteq N(x)$.*

Proof. Suppose that $z \in N(y)$, i.e., $f(z) \leq g(z)$ and $g(y) - g(z) = d_D(y, z)$. Using the triangle inequality, we get

$$\begin{aligned} g(x) - g(z) &= [g(x) - g(y)] + [g(y) - g(z)] \\ &= d_D(x, y) + d_D(y, z) \geq d_D(x, z). \end{aligned}$$

Since g is Lipschitz, $g(x) - g(z) \leq d_D(x, z)$. Therefore, $g(x) - g(z) = d_D(x, z)$, and (x, z) is an edge of BG_f. □

Since A is the largest set that fails the marriage condition, if $x \in A$, $y \in V_>$ and $g(x) - g(y) = d_D(x, y)$ then $y \in A$. Similarly, we can argue that if $x \in N(A)$, $y \in V_\leq$ and $g(x) - g(y) = d_D(x, y)$ then $y \in N(A)$.

Thus, $g(x) - g(y) < d_D(x, y)$ for all $x \in A \cup N(A)$ and $y \notin A \cup N(A)$. Consequently, for some $\delta > 0$, if we increase $g(x)$ by δ for every $x \in A \cup N(A)$ then g remains Lipschitz. This decreases $L_1(f, g)$ by $\delta(|A| - |N(A)|$, a contradiction to g being the closest Lipschitz function to f in L_1 distance. □

4.2 c-Lipschitz Tester for Hypergrids

In this section, we present our c-Lipschitz test for functions on hypergrid domains and prove Theorem 1.5. Observe that testing the c-Lipschitz property of functions with range $[0, 1]$ is equivalent to testing the Lipschitz property of functions with range $[0, 1/c]$ over the same domain. To see this, first note that function f is c-Lipschitz iff function f/c is Lipschitz. Second, function f is ε-far from being c-Lipschitz iff f/c is ε-far from being Lipschitz, provided that the relative L_1 distance between functions scaled correctly: namely, for functions $f, g : D \to [0, r]$, we define $d_1(f, g) = \frac{\|f-g\|_1}{|D| \cdot r}$. Thus, we can restate Theorem 1.5 as follows.

Theorem 4.3 (Thm. 1.5 restated)**.** *Let $n, d \in \mathbb{N}$, $\varepsilon \in (0, 1), r \in [1, \infty)$. The time complexity of L_1-testing the Lipschitz property of functions $f \colon [n]^d \to [0, r]$ (nonadaptively and with 1-sided error) with proximity parameter ε is $O\left(\frac{d}{\varepsilon}\right)$.*

Next we present the test (Algorithm 5) with the complexity claimed in Theorem 4.3. To state the algorithm, we use the notation for axis-parallel lines established before Algorithm 1.

Algorithm 5: Nonadaptive tester of the Lipschitz property for functions on hypergrids.

input : parameters n, d and ε; oracle access to $f \colon [n]^d \to [0, r]$.

repeat $\lceil \frac{d \cdot 8 \ln 3}{\varepsilon} \rceil$ times:
  Sample a uniform line ℓ from $\mathcal{L}_{n,d}$. `// ` $\mathcal{L}_{n,d}$ `is the set of axis-parallel lines in` $[n]^d$.
  Let P_ℓ^r be the set of (unordered) pairs (x, y) of points from ℓ such that $\|x - y\|_1 \leq r$.
  Query a uniformly random pair of points from P_ℓ^{r-1}.
  if $|f(x) - f(y)| > \|x - y\|_1$ **then reject**
accept

Algorithm 5 is nonadaptive. It always accepts all Lipschitz functions. It remains to prove that a function that is ε-far from Lipschitz in L_1 distance is rejected with probability at least $2/3$. Since the algorithm is simply picking pairs (x, y) from a certain set and checking if the Lipschitz property is violated on the selected pairs, it suffices to show that a large fraction of the considered pairs (x, y) is violated by any function that is ε-far from Lipschitz. Observe that for each $\ell \in \mathcal{L}_{n,d}$, the number of pairs x, y on the line ℓ is $n(n-1)/2$. If $r < n$ then $|P_\ell^{r-1}| \leq n(r-1)$. I.e., $|P_\ell^{r-1}| \leq n \cdot \min\{n-1, r-1\}$. Note that $|f(x) - f(y)| > \|x - y\|_1$ implies that $\|x - y\|_1 \leq r - 1$. I.e., all violated pairs on the line ℓ are in P_ℓ^{r-1}. To complete the analysis of Algorithm 5, it suffices to prove the following lemma.

Lemma 4.4. *Let $\mathcal{P}_{n,d}$ bet the set of pairs $x, y \in [n]^d$, where x and y differ in exactly one coordinate. Consider a function $f : [n]^d \to [0, r]$ that is ε-far from Lipschitz w.r.t. the L_1 distance. Then the number of*

pairs in $\mathcal{P}_{n,d}$ violated by f is at least

$$\frac{\varepsilon n}{8d} \cdot \min\{n-1, r-1\} \cdot |\mathcal{L}_{n,d}|.$$

Section 4.3 is devoted to the analysis of the number of violated pairs for 1-dimensional functions. The proof of Lemma 4.4 is completed in Section 4.4.

4.3 Number of violated pairs on a line

Lemma 4.5. *Any function $f : [n] \to [0, r]$ violates at least $\frac{L_1(f,\mathcal{L}ip)}{4} \cdot \min\left\{1, \frac{n-1}{r-1}\right\}$ (unordered) pairs.*

Proof. Let M be a maximum weight matching in the violation graph G_f. Recall Lemma 4.1 that states that $L_1(f, \mathcal{L}ip)$ is the weight of M. We will show that for each edge (x, y) in M, there is a large number of pairs involving x or y violated by f.

Claim 4.6. *Let (x, y) be a pair violated by a function f over domain $[n]$ and let $v = \lceil \frac{\mathrm{vs}_f(x,y)}{2} \rceil - 1$. Then for all $z \in [x - v, y + v] \cap [n]$, function f violates at least one of the two pairs (x, z) and (y, z).*

Proof. W.l.o.g., suppose $x < y$. By definition of the violation score,

$$|f(x) - f(y)| = y - x + \mathrm{vs}_f(x, y) > y - x + 2v. \tag{5}$$

For the sake of contradiction, suppose there is a point $z \in [x - v, y + v] \cap [n]$, such that f violates neither (x, z) nor (y, z). Then

$$|f(y) - f(x)| \le |f(y) - f(z)| + |f(x) - f(z)| \le |y - z| + |x - z| \le |y + x - 2z|.$$

Since $z \ge x - v$, it follows that

$$y + x - 2z \le y + x - 2(x - v) = y - x + 2v.$$

Since $z \le y + v$, it follows that

$$2z - y - x \le 2(y + v) - y - x = y - x + 2v.$$

Therefore, $|f(y) - f(x)| \le |y + x - 2z| \le y - x + 2v$, which contradicts (5). □

Claim 4.7. *Let (x, y) be a pair violated by a function f over domain $[n]$. Then f violates at least*

$$\min\left\{\frac{\mathrm{vs}_f(x, y)}{2}, n - 1\right\}$$

(unordered) pairs in $\{x, y\} \times [n]$.

Proof. Define v as in Claim 4.6. If $y+v \le n$ then Claim 4.6 gives that for each $z \in \{y+1, \ldots, y+v\}$, function f violates at least one of the two pairs (x, z) and (y, z). Since (x, y) is also violated, it gives $v + 1 \ge \frac{\mathrm{vs}_f(x,y)}{2}$ violated pairs in $\{x, y\} \times [n]$. Similarly, if $x - v \ge 1$ then Claim 4.6 gives that for each $z \in \{x - v, \ldots, x - 1\}$, function f violates at least one of the two pairs (x, z) and (y, z), yielding at least $\frac{\mathrm{vs}_f(x,y)}{2}$ violated pairs in $\{x, y\} \times [n]$. Finally, if $y + v > n$ and $x - v < 1$ then Claim 4.6 gives that for each $z \in [n] \setminus \{x, y\}$, function f violates at least one of the two pairs (x, z) and (y, z). Together, with (x, y), it yields at least $n - 1$ violated pairs of the right form. □

Recall that M is a maximum weight matching in G_f. Claim 4.7 shows that each edge (x, y) in M contributes at least $\min\left\{\frac{\mathrm{vs}_f(x,y)}{2}, n-1\right\}$ violated pairs in $\{x, y\} \times [n]$. Since M is a matching, each violated pair can be contributed by at most two edges in M.

Let M_1 be the set of pairs in M with violation score at most $n-1$ and let $M_2 = M \setminus M_1$. By Claim 4.7, edges in M_1 contribute at least $\mathrm{vs}_f(M_1)/2$ violated pairs. Since the range of f is $[0, r]$, the violation score of a pair cannot exceed $r-1$. Therefore, there are at least $\frac{\mathrm{vs}_f(M_2)}{r-1}$ edges in M_2. By Claim 4.7, edges in M_2 contribute at least $\frac{\mathrm{vs}_f(M_2)}{r-1} \cdot (n-1)$ violated pairs. Since each violated pair is contributed at most twice, the total number of distinct violated pairs contributed by edges in M is at least

$$\frac{1}{2} \cdot \left(\frac{\mathrm{vs}_f(M_1)}{2} + \mathrm{vs}_f(M_2) \cdot \frac{n-1}{r-1} \right) = \frac{\mathrm{vs}_f(M)}{4} \cdot \min\left\{1, 2 \cdot \frac{n-1}{r-1}\right\} = \frac{L_1(f, \mathcal{L}ip)}{4} \cdot \min\left\{1, 2 \cdot \frac{n-1}{r-1}\right\}.$$

This completes the proof of the lemma. □

4.4 Analysis of Algorithm 5

We use the dimension reduction lemma below to bound the number of violated pairs on all axis-parallel lines $[n]^d$. This lemma is a direct generalization of the dimension reduction in [4] that was proved for functions whose values are multiples of integers. More precisely, the range is $\delta\mathbb{Z}$, i.e., δ multiples of integers, where $\delta \in (0, \infty)$. (In [4], this result is stated in terms of absolute distances, but for our purposes, we state it in terms of relative distances.)

Lemma 4.8 (Dimension Reduction)**.** *For all $f\colon [n]^d \to \mathbb{R}$,*

$$\mathop{\mathbb{E}}_{\ell \leftarrow \mathcal{L}_{n,d}} [d_{\mathcal{L}ip}(f|_\ell)] \geq \frac{d_{\mathcal{L}ip}(f)}{2d}.$$

Proof. We will use a proof by contradiction. Let $\mathcal{L}ip(D, R)$ denote the set of Lipschitz functions $f : D \to R$. Suppose that there is a function g that contradicts the lemma. In terms of the absolute L_1 distance, it means that $\sum_{\ell \in \mathcal{L}_{n,d}} [L_1(g|_\ell, \mathcal{L}ip([n], \mathbb{R}))] < L_1(g, \mathcal{L}ip([n]^d, \mathbb{R}))/2$. Let δ be sufficiently small, such that $1/\delta$ is a positive integer. (To be more precise in terms of the bound on δ, let $\Delta = L_1(g, \mathcal{L}ip([n]^d, \mathbb{R}))/2 - L_1(g|_\ell, \mathcal{L}ip([n], \mathbb{R}))]$. Then $\delta < \Delta/(2n \cdot |\mathcal{L}_{n,d}| + n^d/2)$.) We will construct a function $g' : [n]^d \to \delta\mathbb{Z}$ which contradicts the reduction dimension lemma in [4], that is, $\sum_{\ell \in \mathcal{L}_{n,d}} [L_1(g'|_\ell, \mathcal{L}ip([n], \delta\mathbb{Z}))] < L_1(g', \mathcal{L}ip([n]^d, \delta\mathbb{Z}))/2$.

Claim 4.9. *For every function f, define function f_δ as follows: for all x, set $f_\delta(x)$ to $f(x)$ rounded down to the nearest multiple of δ. Assuming that $1/\delta$ is an integer, if f is Lipschitz on a pair (x, y) then so is f_δ.*

Proof. Consider a pair (x, y), where $|f(y) - f(x)| \leq 1$, and w.l.o.g. assume that $f(y) \geq f(x)$. Note that $f_\delta(y) \leq f(y)$ because the f-values are rounded down to obtain f_δ-values. We get that $f_\delta(y) - 1 \leq f(y) - 1 \leq f(x)$. Since both $f_\delta(y)$ and 1 are integer multiples of δ, so is $f_\delta(y) - 1$. So, $f(x)$ gets rounded down to $f_\delta(y) - 1$ or a higher value, i.e., $f_\delta(x) \geq f_\delta(y) - 1$. It follows that f_δ is Lipschitz on (x, y). □

Let $g' = g_\delta$, defined as in Claim 4.9. Then

$$\begin{aligned} \sum_{\ell \in \mathcal{L}_{n,d}} [L_1(g'|_\ell, \mathcal{L}ip([n], \delta\mathbb{Z}))] &\leq \sum_{\ell \in \mathcal{L}_{n,d}} [L_1(g|_\ell, \mathcal{L}ip([n], \mathbb{R}))] + 2\delta n \cdot |\mathcal{L}_{n,d}| \\ &< L_1(g, \mathcal{L}ip([n]^d, \mathbb{R}))/2 - \delta n^d/2 \leq L_1(g', \mathcal{L}ip([n]^d, \delta\mathbb{Z}))/2. \end{aligned}$$

The first inequality above holds because for all ℓ, the function $g'|_\ell$ can be transformed into a Lipschitz function with range $\delta\mathbb{Z}$ by first changing it to $g|_\ell$ at the cost of at most δn in the L_1 distance, then changing $g|_\ell$ to the closest Lipschitz function—call it h—at the cost of $L_1(g|_\ell, \mathcal{L}ip([n], \mathbb{R}))$, and finally rounding it down to h_δ at the cost of δn. The second inequality above holds because δ is sufficiently small. The final inequality above holds because g can be transformed into a Lipschitz function by first changing it to g' at the cost of at most n^d in the L_1 distance and then changing it to the closest Lipschitz function.

Thus, g' contradicts the reduction dimension lemma in [4]. This completes the proof of Lemma 4.8. □

Proof of Lemma 4.4. Let $f : [n]^d \to [0, r]$ be ε-far from Lipschitz w.r.t. the L_1 distance. By Lemma 4.5, the number of pairs in $\mathcal{P}_{n,d}$ violated by f is at least

$$\begin{aligned}
\sum_{\ell \in \mathcal{L}_{n,d}} \frac{L_1(f|_\ell, \mathcal{L}ip)}{4} \cdot \min\left\{1, \frac{n-1}{r-1}\right\} &= \sum_{\ell \in \mathcal{L}_{n,d}} \frac{d_{\mathcal{L}ip}(f|_\ell) \cdot n \cdot (r-1)}{4} \cdot \min\left\{1, \frac{n-1}{r-1}\right\} \\
&= \mathop{\mathbb{E}}_{\ell \leftarrow \mathcal{L}_{n,d}} [d_{\mathcal{L}ip}(f|_\ell)] \cdot |\mathcal{L}_{n,d}| \cdot \frac{n}{4} \cdot \min\{r-1, n-1\} \\
&\geq \frac{d_{\mathcal{L}ip}(f)}{2d} \cdot |\mathcal{L}_{n,d}| \cdot \frac{n}{4} \cdot \min\{r-1, n-1\} \\
&\geq \frac{\varepsilon n}{8d} \cdot \min\{n-1, r-1\} \cdot |\mathcal{L}_{n,d}|.
\end{aligned}$$

The first inequality above follows from the dimension reduction lemma (Lemma 4.8). The second inequality holds because f is ε-far from Lipschitz w.r.t. L_1. □

5 Complexity of L_p-Testing Problems: Proofs of Facts 1.1 and 1.2

In this section, we prove more general versions of Facts 1.1 and 1.2, which establish basic relationships between L_p-testing problems.

Definitions of distances. Let (D, Σ, μ) be a measure space with finite measure $\mu(D)$ and let $p \in \{0\} \cup [1, \infty)$. For any real number a, we define

$$a^0 = \begin{cases} 1 \text{ if } a \neq 0; \\ 0 \text{ if } a = 0. \end{cases}$$

Consider the set of all measurable functions $h : D \to [-1, 1]$, where the integral $\int_D |h|^p \, d\mu$ is finite. Then the standard definition of the L_p-norm for $p \geq 1$ is

$$\|h\|_p = \big(\int_D |h|^p \, d\mu \big)^{1/p}.$$

We also define $\|h\|_0 = \int_D |h|^0 \, d\mu$. Hamming distance is not formally a norm; so L_0-norm is usually defined differently. We call Hamming distance L_0 for consistency with the special case of discrete domain D.

For all functions $f, g : D \to [0, 1]$ for which $\|f - g\|_p$ is well defined, we define the relative L_p distance between f and g as follows:

$$d_0(f, g) = \frac{\|f - g\|_0}{\mu(D)} \quad \text{and} \quad d_p(f, g) = \frac{\|f - g\|_p}{\mu(D)^{1/p}} \text{ for all } p \geq 1.$$

For a function $f : D \to [0, 1]$ and a set $\mathcal{P}$ of functions $g : D \to [0, 1]$, the relative L_p distance between f and $\mathcal{P}$ is

$$d_p(f, \mathcal{P}) = \inf_{g \in \mathcal{P}} d_p(f, g).$$

Basic relationships between L_p-norms and between d_p-distances. The following relationships between norms are standard. For all $h : D \to [-1, 1]$ and all $p \geq q \geq 1$,

$$1.\ \|h\|_1 \leq \|h\|_0; \quad 2.\ \|h\|_q \leq \mu(D)^{\frac{1}{q} - \frac{1}{p}} \|h\|_p; \quad 3.\ \|h\|_q^q \geq \|h\|_p^p.$$

The first and the third items hold because the range of h is $[-1, 1]$; the second item can be derived from Jensen's inequality. Moreover, if $h : D \to \{-1, 0, 1\}$, by definition,

$$\|h\|_0 = \|h\|_p^p \text{ for all } p \geq 1.$$

Consider $f, g : D \to [0,1]$. Recall that $d_0(f,g) = \frac{\|f-g\|_0}{\mu(D)}$ and $d_p(f,g) = \frac{\|f-g\|_p}{\mu(D)^{1/p}}$ for all $p \geq 1$. We set $h = f - g$ in the above relationships and immediately get that for all $p \geq q \geq 1$,

$$1.\ d_1(f,g) \leq d_0(f,g); \quad 2.\ d_q(f,g) \leq d_p(f,g); \quad 3.\ d_q^q(f,g) \geq d_p^p(f,g).$$

Moreover, if f and g are Boolean functions, we get

$$d_0(f,g) = d_p^p(f,g) \text{ for all } p \geq 1.$$

Analogous inequalities also hold for distances between a function and a set.

Observation 5.1. *For all $p \geq q \geq 1$, functions $f : D \to [0,1]$, and sets $\mathcal{P}$ of functions $D \to [0,1]$,*

$$1.\ d_1(f,P) \leq d_0(f,P); \quad 2.\ d_q(f,P) \leq d_p(f,P); \quad 3.\ d_q(f,P) \geq d_p^{p/q}(f,P).$$

Moreover, if f is a Boolean functions and $\mathcal{P}$ is a set of Boolean functions, we get

$$d_0(f,P) = d_p^p(f,P) \text{ for all } p \geq 1.$$

5.1 Relationships between complexity of L_p-testing problems

Recall that we denote the worst-case query complexity of L_p-testing for property $\mathcal{P}$ with proximity parameter ε by $Q_p(\mathcal{P}, \varepsilon)$.

Fact 5.2. *Let $\varepsilon \in (0,1)$ and $\mathcal{P}$ be a property over any domain. Then*

$$1.\ Q_1(\mathcal{P},\varepsilon) \leq Q_0(\mathcal{P},\varepsilon); \quad 2.\ Q_q(\mathcal{P},\varepsilon) \leq Q_p(\mathcal{P},\varepsilon) \leq Q_q(\mathcal{P},\varepsilon^{p/q}) \text{ for all } p \geq q \geq 1.$$

Moreover, if $\mathcal{P}$ is a property of Boolean functions then

$$Q_p(\mathcal{P},\varepsilon) = Q_0(\mathcal{P},\varepsilon^p) \text{ for all } p \geq 1.$$

Fact 5.2 follows from Observation 5.1 above and the following observation.

Observation 5.3. *Let f be a functions and $\mathcal{P}$ be a property of functions. Consider two distance measures d and d' that map a function and a set of functions to $[0,1]$. Suppose there is a monotone nondecreasing function $t : [0,1] \to [0,1]$ such that*

$$d(f,\mathcal{P}) \geq t(d'(f,\mathcal{P})) \text{ for all } f, \mathcal{P}.$$

Let $\varepsilon \in (0,1)$. Then a tester for property $\mathcal{P}$ w.r.t. distance d run with proximity parameter $t(\varepsilon)$ is also a tester for property $\mathcal{P}$ w.r.t. distance d' with proximity parameter ε.

Note that the time and query complexity of the tester are preserved in this no-op transformation from a tester w.r.t. d to a tester w.r.t. d'. Also, if a tester is nonadaptive or has one-sided error w.r.t. d, it will retain these features w.r.t. d'.

Proof. Let T be a tester for property $\mathcal{P}$ w.r.t. distance d with proximity parameter $t(\varepsilon)$. Then T

1. accepts functions $f \in \mathcal{P}$ with probability at least $2/3$, and
2. rejects functions f, where $d(f,\mathcal{P}) \geq t(\varepsilon)$, with probability at least $2/3$.

A tester for property $\mathcal{P}$ w.r.t. distance d' with proximity parameter ε, in addition to satisfying item 1 above, must also also reject functions f, where $d'(f,\mathcal{P}) \geq \varepsilon$, with probability at least $2/3$. For such a function f, the distance $d(f,\mathcal{P}) \geq t(d'(f,\mathcal{P})) \geq t(\varepsilon)$ by the inequality in the statement of Observation 5.3 and since t is monotone nondecreasing. Thus, by item 2 above, T rejects f with probability at least $2/3$, as required. □

Proof of Fact 5.2. Let $p_1, p_2 \in \{0\} \cup [1,\infty)$. By Observation 5.3, if there is a monotone nondecreasing function $t : [0,1] \to [0,1]$, such that $d_{p_1}(f,\mathcal{P}) \geq t(d_{p_2}(f,\mathcal{P}))$ for all $f, \mathcal{P}$, then $Q_{p_2}(\mathcal{P},\varepsilon) \leq Q_{p_1}(\mathcal{P}, t(\varepsilon))$. Specifically, if $d_{p_2}(f,\mathcal{P}) \leq d_{p_1}(f,\mathcal{P})$ for all $f, \mathcal{P}$ than $Q_{p_2}(\mathcal{P},\varepsilon) \leq Q_{p_1}(\mathcal{P},\varepsilon)$. Now all inequalities in Fact 5.2 follow from the corresponding inequalities in Observation 5.1. □

5.2 Relationships between complexity of tolerant L_p-testing problems

In this section, we state a prove a generalization of Fact 1.2.

Fact 5.4. *Consider $p \geq q \geq 1$. Let $\varepsilon_1, \varepsilon_2 \in (0,1)$, satisfying $\varepsilon_1 < \varepsilon_2^p$, and $\mathcal{P}$ be a property over any domain. Then*

$$Q_q(\mathcal{P}, \varepsilon_1^{p/q}, \varepsilon_2) \leq Q_p(\mathcal{P}, \varepsilon_1, \varepsilon_2) \leq Q_q(\mathcal{P}, \varepsilon_1, \varepsilon_2^{p/q}).$$

Moreover, if $\mathcal{P}$ is a property of Boolean functions then

$$Q_p(\mathcal{P}, \varepsilon_1, \varepsilon_2) = Q_0(\mathcal{P}, \varepsilon_1^p, \varepsilon_2^p) \text{ for all } p \geq 1.$$

Fact 5.4 follows from Observation 5.1 above and the following observation.

Observation 5.5. *Let f be a functions and $\mathcal{P}$ be a property of functions. Consider two distance measures d and d' that map a function and a set of functions to $[0,1]$. Suppose there are monotone nondecreasing functions $b, t : [0,1] \to [0,1]$ such that*

$$b(d'(f,\mathcal{P})) \geq d(f,\mathcal{P}) \geq t(d'(f,\mathcal{P})) \text{ for all } f, \mathcal{P}.$$

Let $\varepsilon_1, \varepsilon_2 \in (0,1)$, such that $b(\varepsilon_1) < t(\varepsilon_2)$. Then a tolerant tester for property $\mathcal{P}$ w.r.t. distance d run with parameters $b(\varepsilon_1), t(\varepsilon_2)$ is also a tolerant tester for property $\mathcal{P}$ w.r.t. distance d' with parameters $\varepsilon_1, \varepsilon_2$.

Proof. Let T be a tolerant tester for property $\mathcal{P}$ w.r.t. distance d with parameters $b(\varepsilon_1), t(\varepsilon_2)$. Then T

1. accepts functions f, where $d(f,\mathcal{P}) \leq b(\varepsilon_1)$, with probability at least 2/3, and
2. rejects functions f, where $d(f,\mathcal{P}) \geq t(\varepsilon_2)$, with probability at least 2/3.

A tolerant tester for property $\mathcal{P}$ w.r.t. distance d' with parameters $\varepsilon_1, \varepsilon_2$ must satisfy the following:

(Requirement 1): accept functions f, where $d'(f,\mathcal{P}) \leq \varepsilon_1$, with probability at least 2/3,

(Requirement 2): reject functions f, where $d'(f,\mathcal{P}) \geq \varepsilon_2$, with probability at least 2/3.

If f satisfies $d'(f,\mathcal{P}) \leq \varepsilon_1$ then the distance $d(f,\mathcal{P}) \leq b(d'(f,\mathcal{P})) \leq b(\varepsilon_1)$ by the inequality in the statement of Observation 5.5 and since b is monotone nondecreasing. Thus, by item 1 satisfied by T, it accepts f with probability at least 2/3, satisfying Requirement 1. If f satisfies $d'(f,\mathcal{P}) \geq \varepsilon_2$ then the distance $d(f,\mathcal{P}) \geq t(d'(f,\mathcal{P})) \geq t(\varepsilon_2)$ by the inequality in the statement of Observation 5.5 and since t is monotone nondecreasing. Thus, by item 2 satisfied by T, it rejects f with probability at least 2/3, satisfying Requirement 2. □

Proof of Fact 5.4. Let $p_1, p_2 \in \{0\} \cup [1,\infty)$. By Observation 5.5, if there are monotone nondecreasing functions $b, t : [0,1] \to [0,1]$, such that $b(d_{p_2}(f,\mathcal{P})) \geq d_{p_1}(f,\mathcal{P}) \geq t(d_{p_2}(f,\mathcal{P}))$ for all $f, \mathcal{P}$, then $Q_{p_2}(\mathcal{P},\varepsilon) \leq Q_{p_1}(\mathcal{P}, t(\varepsilon))$. Now all inequalities in Fact 5.4 follow from the corresponding inequalities in Observation 5.1. Specifically, the first inequality holds because $d_q^{q/p}(f,P) \geq d_p(f,P) \geq d_q(f,P)$. The second inequality holds because $d_p(f,P) \geq d_q(f,P) \geq d_p^{p/q}(f,P)$. The equality follows from the equality in Observation 5.1. □

6 Open Problems

This paper introduces testing properties of functions under L_p-distances. Among multiple open problems left we would like to highlight the following:

- (Adaptive L_1-tester for monotonicity) Our L_1-tester for monotonicity (Theorem 1.3) is nonadaptive and cannot be substantially improved without adaptivity (Theorem 2.14). However, we show that adaptivity helps for Boolean range (Theorem 2.8). Is there a better adaptive tester for L_1?

- (Better L_p-testers for $p > 1$) All our algorithms for L_p-testing for $p > 1$ were obtained directly from L_1-testers. Can one design better algorithms by working directly with L_p-distances for $p > 1$?
- (Approximating L_p-distance to monotonicity in high dimensions) We designed a distance approximation algorithm only for monotonicity of functions on $[n]$ and $[n]^2$ (Corollary 2.4) Designing tolerant L_p-testers and distance approximation algorithms for monotonicity of functions over higher-dimensional grids is an open question.
- (Tolerant testers for other properties) Designing tolerant L_p-testers for other properties is also open. The starting point here would be to design tolerant testers and distance approximation algorithms for the Lipschitz property and convexity of functions on a line ($d = 1$).
- (Testing convexity in high dimensions) The last open problem is designing a (non-tolerant) L_p-tester for convexity in high dimension with subexponential dependence on the dimension (see Table A.1. We expect this to be quite challenging (see also [49] for an exponential lower bound against the one-dimensional restriction technique in a related testing model).

Acknowledgements

We would like to thank Kenneth Clarkson, Jan Vondrak and Vitaly Feldman for helpful discussions and Adam Smith for comments on this document.

References

[1] *Mathematische Annalen*, 297(1), 1993. ISSN 0025-5831.

[2] N. Ailon and B. Chazelle. Information theory in property testing and monotonicity testing in higher dimension. *Inf. Comput.*, 204(11):1704–1717, 2006.

[3] N. Ailon, B. Chazelle, S. Comandur, and D. Liu. Estimating the distance to a monotone function. *Random Struct. Algorithms*, 31(3):371–383, 2007.

[4] P. Awasthi, M. Jha, M. Molinaro, and S. Raskhodnikova. Testing Lipschitz functions on hypergrid domains. In *APPROX-RANDOM*, pages 387–398, 2012.

[5] M.-F. Balcan, E. Blais, A. Blum, and L. Yang. Active property testing. In *FOCS*, pages 21–30, 2012.

[6] T. Batu, R. Rubinfeld, and P. White. Fast approximate PCPs for multidimensional bin-packing problems. *Inf. Comput.*, 196(1):42–56, 2005.

[7] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing closeness of discrete distributions. *J. ACM*, 60(1):4, 2013.

[8] S. Ben-Moshe, Y. Kanza, E. Fischer, A. Matsliah, M. Fischer, and C. Staelin. Detecting and exploiting near-sortedness for efficient relational query evaluation. In *ICDT*, pages 256–267, 2011.

[9] A. Bhattacharyya, E. Fischer, R. Rubinfeld, and P. Valiant. Testing monotonicity of distributions over general partial orders. In *ICS*, pages 239–252, 2011.

[10] A. Bhattacharyya, E. Grigorescu, K. Jung, S. Raskhodnikova, and D. P. Woodruff. Transitive-closure spanners. *SIAM J. Comput.*, 41(6):1380–1425, 2012.

[11] E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012.

[12] E. Blais, S. Raskhodnikova, and G. Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *CCC*, 2014.

[13] J. Briët, S. Chakraborty, D. García-Soriano, and A. Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.

[14] J. Brody and P. Hatami. Distance-sensitive property testing lower bounds. *CoRR*, abs/1304.6685, 2013.

[15] D. Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *STOC*, pages 419–428, 2013.

[16] D. Chakrabarty and C. Seshadhri. A $o(n)$ monotonicity tester for boolean functions over the hypercube. In *STOC*, pages 411–418, 2013.

[17] D. Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. In *APPROX-RANDOM*, pages 425–435, 2013.

[18] D. Chakrabarty, K. Dixit, M. Jha, and C. Seshadhri. Optimal lower bounds for Lipschitz testing via monotonicity. *Private communication*, 2013.

[19] S. Chan, I. Diakonikolas, P. Valiant, and G. Valiant. Optimal algorithms for testing closeness of discrete distributions. In *SODA*, pages 1193–1203, 2014.

[20] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, pages 1370–1379, 2005.

[21] C. Daskalakis, I. Diakonikolas, R. A. Servedio, G. Valiant, and P. Valiant. Testing k-modal distributions: Optimal algorithms via reductions. In *SODA*, pages 1833–1852, 2013.

[22] I. Diakonikolas, H. K. Lee, K. Matulef, K. Onak, R. Rubinfeld, R. A. Servedio, and A. Wan. Testing for concise representations. In *FOCS*, pages 549–558, 2007.

[23] K. Dixit, M. Jha, S. Raskhodnikova, and A. Thakurta. Testing the Lipschitz property over product distributions with applications to data privacy. In *TCC*, pages 418–436, 2013.

[24] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. In *RANDOM*, pages 97–108, 1999.

[25] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[26] F. Ergun, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *J. Comput. Syst. Sci.*, 60(3):717–751, 2000.

[27] S. Fattal and D. Ron. Approximating the distance to convexity. *Available at:* `http://www.eng.tau.ac.il/~danar/Public-pdf/app-conv.pdf`, 2007.

[28] S. Fattal and D. Ron. Approximating the distance to monotonicity in high dimensions. *ACM Transactions on Algorithms*, 6(3), 2010.

[29] V. Feldman and P. Kothari. Learning coverage functions. *CoRR*, abs/1304.2079, 2013.

[30] V. Feldman and J. Vondrák. Optimal bounds on approximation of submodular and XOS functions by juntas. In *FOCS*, pages 227–236, 2013.

[31] E. Fischer. On the strength of comparisons in property testing. *Inf. Comput.*, 189(1):107–116, 2004.

[32] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *STOC*, pages 474–483, 2002.

[33] O. Goldreich. On multiple input problems in property testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:67, 2013.

[34] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32, 1989.

[35] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

[36] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

[37] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

[38] P. M. Gruber. *Convex and discrete geometry*, volume 336. Springer Berlin, 2007.

[39] S. Halevy and E. Kushilevitz. Distribution-free property-testing. *SIAM J. Comput.*, 37(4):1107–1138, 2007.

[40] S. Halevy and E. Kushilevitz. Testing monotonicity over graph products. *Random Struct. Algorithms*, 33(1): 44–67, 2008.

[41] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inf. Comput.*, 100(1):78–150, 1992.

[42] A. Hinrichs, E. Novak, and H. Wozniakowski. The curse of dimensionality for the class of monotone functions and for the class of convex functions. *Journal of Approximation Theory*, 163(8):955–965, 2011.

[43] M. Jha and S. Raskhodnikova. Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM J. Comput.*, 42(2):700–731, 2013.

[44] M. J. Kearns, R. E. Schapire, and L. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3): 115–141, 1994.

[45] L. A. Levin. One-way functions and pseudorandom generators. In *STOC*, pages 363–365, 1985.

[46] R. J. Lipton and J. F. Naughton. Query size estimation by adaptive sampling. *J. Comput. Syst. Sci.*, 51(1): 18–25, 1995.

[47] M. Parnas, D. Ron, and R. Rubinfeld. On testing convexity and submodularity. *SIAM J. Comput.*, 32(5): 1158–1184, 2003.

[48] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006.

[49] L. Rademacher and S. Vempala. Testing geometric convexity. In *FSTTCS*, pages 469–480, 2004.

[50] S. Raskhodnikova. Approximate testing of visual properties. In *RANDOM-APPROX*, pages 370–381, 2003.

[51] S. Raskhodnikova and G. Yaroslavtsev. Learning pseudo-Boolean k-DNF and submodular functions. In *SODA*, pages 1356–1368, 2013.

[52] D. Ron. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning*, 1(3): 307–402, 2008.

[53] G. Rote. The convergence rate of the sandwich algorithm for approximating convex functions. *COMPUTING*, 48:337–361, 1992.

[54] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

[55] M. Saks and C. Seshadhri. Estimating the longest increasing sequence in polylogarithmic time. In *FOCS*, pages 458–467, 2010.

[56] R. A. Servedio. Testing by implicit learning: A brief survey. In *Property Testing*, pages 197–210, 2010.

[57] C. Seshadhri and J. Vondrák. Is submodularity testable? *Algorithmica*, 69(1):1–25, 2014.

[58] G. Sonnevend. An optimal sequential algorithm for the uniform approximation of convex functions on $[0,1]^2$. *Applied Mathematics and Optimization*, 10(1):127–142, 1983.

[59] G. Valiant and P. Valiant. The power of linear estimators. In *FOCS*, pages 403–412, 2011.

[60] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[61] P. Valiant. Testing symmetric properties of distributions. *SIAM J. Comput.*, 40(6):1927–1968, 2011.

Convexity, submodularity, Monge matrices

$\mathcal{D}$	Hamming Testing	L_p-testing
Convexity $[n]$	$\Theta\left(\frac{\log n}{\varepsilon}\right)$ [47]	$\Theta(\frac{1}{\varepsilon^p})$
Submodularity $\{0,1\}^d$	$\left(\frac{1}{\varepsilon}\right)^{O(\sqrt{d}\log d)}$ [57]	$2^{\tilde{O}(1/\varepsilon^p)} + poly(1/\varepsilon)\log d$ [30]
Monge $[n]^2$	$O\left(\frac{\log^2 n}{\varepsilon}\right)$ [47]	—
Convexity $[n]^d$	—	$O\left(\frac{c(d)}{\varepsilon^{p(d/2+1)}}\right)$

Table A.1: Query complexity of L_p-testing convexity/submodularity of a function $f : \mathcal{D} \to [0,1]$. For convexity $c(d)$ is a fixed function of d.

A L_1-Testing Convexity and Related properties via Learning

In this section we describe the connections between PAC-style learning models under L_p distances and L_p-testing and state the results, which can be obtained through this connection. For definitions of submodularity and Monge matrices see, e.g. [30, 47].

A.1 Testing via learning and approximation

A connection between PAC-style learning and Hamming property testing is well-known [36, 52]. We first give standard definitions of agnostic and proper PAC-style learning with L_p-error under the uniform distribution, including versions with and without queries [60, 44, 41], and then describe an analogous connection with L_p-testing.

Definition A.1 (PAC-style learning with L_p error)**.** *Let $\mathcal{P}$ be a class of functions $f\colon D \to [0,1]$. An algorithm $\mathcal{A}$* agnostically PAC-learns *$\mathcal{P}$ under the uniform distribution with L_p-error if for every $\varepsilon > 0$, given access to random examples labeled by f and drawn uniformly and independently from D, with probability at least $2/3$ it outputs a hypothesis h such that:*

$$\|f - h\|_p - \inf_{g\in\mathcal{P}} \|f - g\|_p \le \varepsilon.$$

If $\mathcal{A}$ has oracle access to f, we call it a PAC-style learning algorithm with queries*. For the promise problem when $f \in \mathcal{P}$ (i.e., $\inf_{g\in P} \|f - g\|_p = 0$) the agnostic quantifier is omitted. Algorithm $\mathcal{A}$ is* proper *if it always outputs $h \in \mathcal{P}$.*

If in the definition above the approximation guarantee holds with probability 1 instead of $2/3$ and $\mathcal{A}$ is a proper learner, which is allowed to make queries, then we call $\mathcal{A}$ an *approximation algorithm* with L_p error.

Lemma A.1. *If there exists an PAC-style learning algorithm $\mathcal{A}$ with L_p error for a class $\mathcal{P}$ with query complexity $q(\varepsilon, n)$ and running time $r(\varepsilon, n)$ then there exists an L_p-testing algorithm $\mathcal{B}$ with query complexity $q(\varepsilon, n) + O(\varepsilon^{-p})$. If the learning algorithm is also proper then the running time of the resulting testing algorithm is $r(\varepsilon, n) + O(\varepsilon^{-p})$. If the learning algorithm is agnostic, then there also exists a distance approximation algorithm with additive error ε with the same query complexity and running time, except for the additive term being $O(\varepsilon^{-2p})$ instead of $O(\varepsilon^{-p})$.*

If $\mathcal{A}$ only uses random examples then $\mathcal{B}$ also only uses random examples. If $\mathcal{A}$ is nonadaptive then $\mathcal{B}$ is also nonadaptive.

Our proof follows the lines of [52] except that we need to argue differently about sample complexity of estimating L_p distance between the concept output by a proper learner and a function f. This results in a different bound for L_2-testing as compared to that in [52], which considers L_0 testing. For L_1-testing the query complexity is the same as for L_0-testing. We also make the same observation as in [51] that for non-proper learners query complexity is the same as for proper learners, while the running time increases.

Proof of Lemma A.1. If the learning algorithm doesn't output a hypothesis, exceeds the upper bound on its running time or outputs a hypothesis that is not in $\mathcal{P}$ then we reject f. Otherwise, let $g \in \mathcal{P}$ be the hypothesis produced by a proper L_p-learning algorithm with precision $\varepsilon/3$ when it is given access to f. We use k independently chosen uniformly distributed samples $x_1, \ldots, x_k$ to estimate the distance between f and g as $d = k^{-\frac{1}{p}} \cdot \left(\sum_{i=1}^{k} |f(x_i) - g(x_i)|^p\right)^{\frac{1}{p}}$. If $d \leq 2\varepsilon/3$ then the algorithm accepts, otherwise it rejects.

Using multiplicative Chernoff bound (Theorem E.1) we show below that $k = O(1/\varepsilon)$ samples suffice for L_0, L_1 and L_2^2-testing and $k = O(1/\varepsilon^2)$ samples suffice for L_2-testing.

Indeed, if $f \in \mathcal{P}$ then $\|f - g\|_p \leq \varepsilon/3$ with probability $1 - \delta$ by the guarantee of a proper learning algorithm. We may assume that $\|f - g\|_p = \varepsilon/3$ since otherwise our bounds are only better. We have:

$$\begin{aligned}
&\Pr\left[d > \frac{2\varepsilon}{3}\right] \\
&= \Pr\left[k^{-\frac{q}{p}} \left(\sum_{i=1}^{k} |f(x_i) - g(x_i)|^p\right)^{\frac{1}{p}} > 2d_p(f,h)\right] \\
&= \Pr\left[\frac{1}{k}\left(\sum_{i=1}^{k} |f(x_i) - g(x_i)|^p\right) > 2^p dist_p^p(f,h)\right] \\
&< e^{-\frac{(2^p-1)^2(\varepsilon/3)^p k}{3}}
\end{aligned}$$

because $\chi_i = (f(x_i) - g(x_i))^p$ are i.i.d random variables with expectation $p = (dist_p^p(f,h)) = (\varepsilon/3)^p$. Thus, taking the number of samples to be $k = O(\varepsilon^{-p})$ suffices to have $\Pr[d > \frac{2\varepsilon}{3}] < 1/6$.

If f is ε-far from $\mathcal{P}$ then because $g \in \mathcal{P}$ (the learning algorithm is proper) we have $\|f - g\|_p \geq \varepsilon$. Same reasoning as above together with the second Chernoff bound (7) completes the proof for proper learners. If the learning algorithm is not proper, we can make it proper by finding the closest function h to g by going over all functions $h \in \mathcal{P}$. See Proposition A.1 in [51] for the details.

For the connection with agnostic learning note that by Chernoff bound a sample of size $O(\varepsilon^{-2p})$ suffices to estimate the distance between two functions with additive error ε under L_p distance. □

A.2 Convexity

Our results for L_p-testing convexity over hypergrids $[n]^d$ have query complexity independent of n, thus for convenience we use a continuous domain $[0,1]^d$ in this section instead. All approximation algorithms discussed in this section don't require any assumptions about smoothness of the function class, which is being approximated, and thus can be also used in the discrete setting. Stronger results are known for approximation under smoothness assumptions (see e.g. [38]).

A function $f\colon [0,1]^d \to [0,1]$ is convex if $\alpha f(x) + (1-\alpha)f(y) \geq f(\alpha x + (1-\alpha)y)$ for every $x, y \in [0,1]^d$ and $\alpha \in [0,1]$.

For the class of convex functions $f\colon [0,1]^d \to [0,1]$ the following approximation algorithms are known:

- For $d = 1$ the folklore nonadaptive proper "sandwich algorithm" with L_∞ error (and hence also L_1 error), which uses queries, has query complexity $O\left(\frac{1}{\sqrt{\varepsilon}}\right)$ (see, e.g., [53]).
- For $d = 2$ there exists an adaptive non-proper algorithm with L_∞ error, which uses queries and has query complexity $O\left(\frac{\log 1/\varepsilon}{\varepsilon}\right)$ [58] (see also [53] for a discussion).

- For every $d \geq 1$ there exists a nonadaptive proper algorithm with L_1 error which only uses random examples and has query complexity $O\left(\frac{c(d)}{\varepsilon^{d/2+1}}\right)$ for a fixed function c (see, e.g., [1, 49]).

Exponential in d lower bounds for deterministic approximation with L_p error follow from lower bounds for integration [42].

We remark that the algorithm in the last item above is stated in [1, 49] for d-dimensional convex bodies rather than functions and requires access to an oracle giving random samples from the body. The formulation that we use here follows by considering the plot above a convex function $f\colon [0,1]^d \to [0,1]$ as a $(d+1)$-dimensional convex body. To implement a random oracle it suffices to sample a random point $(x_1, \ldots, x_d, y) \in [0,1]^{d+1}$, return it if $f(x_1, \ldots, x_d) \leq y$ and repeat the process otherwise.

Using Lemma A.1 we get the following corollary.

Corollary A.2. *For $p \geq 1$ there exist L_p-testing algorithms for convexity of functions $f\colon [0,1]^d \to [0,1]$ with the following properties.*

- *Query complexity $O\left(\frac{1}{\varepsilon^p}\right)$ for $d = 1$ (nonadaptive, requires queries).*
- *Query complexity $O\left(\frac{\log 1/\varepsilon}{\varepsilon^p}\right)$ for $d = 2$ (adaptive, requires queries).*
- *Query complexity $O\left(\frac{c(d)}{\varepsilon^{p(d/2+1)}}\right)$ for $d \geq 1$ (nonadaptive, random queries suffice).*

While the algorithm for $d = 1$ matches the trivial $\Omega(1/\varepsilon^p)$ lower bound and the algorithm for $d = 2$ only loses a logarithmic factor, the complexity of testing convexity for $d \geq 3$ remains widely open. It is known, however, that probably the most natural approach to this problem which tests one-dimensional convexity of projection on a randomly chosen line fails to reduce the exponential dependence on d for a related problem of testing convexity of bodies [49].

B Computing L_1 distance to monotonicity

Lemma B.1. *Given $f\colon [n] \to [0,1]$, the relative L_1 distance to monotonicity, $d_{\mathcal{M}}(f)$, can be computed in $O(n \log n)$ time.*

Proof. By Lemma 2.1, $d_{\mathcal{M}}(f) = \int_0^1 d_{\mathcal{M}}(f_{(t)})\, dt$. Let $a_1 \leq \cdots \leq a_n$ be the sequence of values that the function f takes, sorted in nondecreasing order. Let $a_0 = 0, a_{n+1} = 1$. Clearly, for all $t \in (a_i, a_{i+1}]$, threshold functions $f_{(t)}$ are the same. Thus,

$$d_{\mathcal{M}}(f) = \sum_{i=0}^{n} (a_{i+1} - a_i) \cdot d_{\mathcal{M}}(f_{(a_i)}).$$

Thus, $d_{\mathcal{M}}(f_{(a_i)}) = \frac{n - 2m_{f_{(a_i)}} + s_{f_{(a_i)}}}{n}$. Thus, it suffices to compute values $m_{f_{(a_i)}}$ and $s_{f_{(a_i)}}$ for all $i \in [0,n]$. For $i = 0$ we have $m_{f_{(0)}} = 0$ and $s_{f_{(0)}} = -n$. Clearly, $s_{f_{(i)}} = s_{f_{(i-1)}} + 2$. The value $m_{f_{(a_i)}}$ can be also computed easily from $m_{f_{(a_{i-1})}}$ using the interval tree data structure.

Proposition B.2. *An interval tree data structure can be used to store a sequence $b_1, \ldots, b_n$, where $b_i \in [0,1]$, and support the following two operations, each in $O(\log n)$ time:*

1. *(Incremental update on an interval) Given $i \leq j$ and a number p, update b_k to $b_k + p$ for all $k \in [i,j]$.*
2. *(Maximum query on an interval) Given $i \leq j$, return $\max_{k \in [i,j]} b_k$.*

For $j \in [n]$, we define a sequence $b_j^i = \sum_{x=0}^{j} g_{f_{(a_i)}}(x)$. By definition, $m_{f_{(a_i)}} = \max_{j \in [n]} b_j^i$, and we can use the maximum query on the interval $[1,n]$ to determine this value. Given values b_j^i, the values b_j^{i+1} are defined as follows. Let x be such that $f(x) = a_{i+1}$. We have $b_j^{i+1} = b_j^i$ if $j < x$ and $b_j^{i+1} = b_j^i + 1$ if $j \geq x$. Thus the incremental update operation on an interval $[x,n]$ suffices to recompute values b_j^{i+1} from b_j^i. □

C Optimality of the improved Levin's work investment strategy

In this section, we give a result on the optimality of the work investment strategy explained in the beginning of Section 2.2.

Lemma C.1. *When the work needed for an element e is $1/q(e)$, the cost of the optimal work investment strategy is $\Theta\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$.*

After stating Lemma 2.5, we argued that the work investment strategy suggested by this lemma has cost $O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$. Here we show that there exists a distribution $\mathcal{D}$ for which every strategy costs $\Omega\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$. The lemma follows from the following claim.

Claim C.2. *Let $\epsilon \in [0, 1/16]$ be a parameter. There exists a distribution $\mathcal{D}$ with $\mathbb{E}_{X\sim\mathcal{D}}[X] = \epsilon$ and a constant $c > 0$ such that any algorithm which has $\sum_{i=1}^{k} s_i < c\frac{1}{\epsilon}\log\frac{1}{\epsilon}$ loses with probability at least $\frac{1}{2}$.*

Proof. Let $t = \log\frac{1}{\epsilon}$. Let $\mathcal{D}$ be described as follows. To pick $X \sim \mathcal{D}$, we do the following:

1. Pick j uniformly from $[t]$.
2. Let X be $1/2^j$ with probability $p = 2^j\epsilon$ and $X = 0$ otherwise.

Let $s_1, \ldots, s_k$ be the vector defining the strategy of the algorithm. We can assume that s_i's have been chosen in advance (i.e., the algorithm is nonadaptive) because the only information the algorithm gets during the game until it wins is that its attempts to win have been unsuccessful. We will also assume that each s_i is rounded up to the next power of two because this transformation can increase the sum of the numbers by at most a factor of 2. Because the probability of winning is monotonically increasing as a function of $(s_1, \ldots, s_k)$, it suffices to show the claim for this new sequence.

Let $n_j = \left|\{i | s_i = 2^j\}\right|$ for $j \in [t]$ so that we have $\sum_{j=0}^{t} n_j = k$. Now the probability of losing is given as:

$$\prod_{j\in[t]}\left(1 - \frac{1}{t}\sum_{\ell=1}^{j} 2^\ell\epsilon\right)^{n_j} \geq \prod_{j\in[t]}\left(1 - \frac{2^{j+1}\epsilon}{t}\right)^{n_j} \geq e^{-\frac{4\epsilon}{t}\sum_{j\in[t]} 2^j n_j}.$$

The second inequality uses that $1 - x \geq e^{-2x}$ for $x \in [0, 1/2]$, where we take $x = \frac{2^{j+1}\epsilon}{t}$. Indeed, $\frac{2^{j+1}\epsilon}{t} \leq 1/2$ for all j because the largest value of j is $t = \log 1/\epsilon$ and we have $\frac{2^{j+1}\epsilon}{t} = \frac{2}{\log\frac{1}{\epsilon}} \leq 1/2$ because $\epsilon \leq 1/16$.

Finally, note that $\sum_i s_i = \sum_j 2^j n_j$ and thus if $\sum_i s_i < \frac{\ln 2}{2}\frac{1}{\epsilon}\log\frac{1}{\epsilon}$ we have that the probability of not winning is at least $1/2$. □

D Auxiliary lemmas

Lemma D.1. *If $X \in [0, 1]$ is a random variable such that with probability at least $1 - \gamma$ it holds that:*

$$\epsilon - c_1\delta \leq X \leq \epsilon + c_2\delta,$$

for some values $\delta, \epsilon, c_1, c_2 > 0$ then if $\gamma = \delta^2$ then:

$$|\mathbb{E}[X] - \epsilon| = O(\delta)$$
$$\sigma[X] = O(\delta).$$

Proof. For the expectation we have:

$$\mathbb{E}[X] = \int_0^1 \Pr[X \geq x]dx = \int_0^{\epsilon+c_2\delta} \Pr[X \geq x]dx + \int_{\epsilon+c_2\delta}^1 \Pr[X \geq x]dx \leq \epsilon + c_2\delta + \gamma = \epsilon + O(\delta),$$

and also

$$\mathbb{E}[X] = \int_0^1 \Pr[X \geq x]dx = \int_0^{\epsilon - c_1\delta} \Pr[X \geq x]dx + \int_{\epsilon - c_1\delta}^1 \Pr[X \geq x]dx \geq (1-\gamma)(\epsilon - c_1\delta) = \epsilon - O(\delta).$$

For the variance:

$$Var[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \int_0^1 \Pr[(X - \mathbb{E}[X])^2 \geq x]dx \leq$$
$$\max\left(\int_0^1 \Pr[(X - \epsilon - c_2\delta - \gamma)^2 \geq x]dx, \int_0^1 \Pr[(X - (1-\gamma)(\epsilon - c_1\delta))^2 \geq x]dx\right)$$

We have:

$$\int_0^1 \Pr[(X - \epsilon - c_2\delta - \gamma)^2 \geq x]dx \leq (c_1\delta + c_2\delta + \gamma)^2 + \gamma$$

and the same bound holds for the second term as well. Thus for $\gamma = \delta^2$ we have $\sigma[X] \leq c\delta$ for some constant $c > 0$. □

E Standard Probabilistic Inequalities

Theorem E.1 (Multiplicative Chernoff Bound)**.** *Let $\chi_1, \ldots, \chi_m$ be m independent random variables, where $\chi_i \in [0,1]$ for all $i \in [m]$. Let $p = \frac{1}{m}\sum_i \mathbb{E}[\chi_i]$. Then for every $\gamma \in [0,1]$ the following bounds hold:*

$$\Pr\left[\frac{1}{m}\sum_{i=1}^m \chi_i > (1+\gamma)p\right] < e^{-\frac{\gamma^2 pm}{3}}; \tag{6}$$

$$\Pr\left[\frac{1}{m}\sum_{i=1}^m \chi_i < (1-\gamma)p\right] < e^{-\frac{\gamma^2 pm}{2}}. \tag{7}$$