Sublinear Algorithms for Testing Monotone and Unimodal Distributions

Tuğkan Batu[*]
batu@cs.utexas.edu
Department of Computer Sciences
University of Texas
Austin, TX 78712

Ravi Kumar
ravi@almaden.ibm.com
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120

Ronitt Rubinfeld[†]
ronitt@theory.lcs.mit.edu
Computer Science and Artificial Intelligence Laboratory, M.I.T.
Cambridge, MA 02139

ABSTRACT

The complexity of testing properties of monotone and unimodal distributions, when given access only to samples of the distribution, is investigated. Two kinds of sublinear-time algorithms—those for testing monotonicity and those that take advantage of monotonicity—are provided.

The first algorithm tests if a given distribution on $[n]$ is monotone or far away from any monotone distribution in L_1-norm; this algorithm uses $\tilde{O}(\sqrt{n})$ samples and is shown to be nearly optimal. The next algorithm, given a joint distribution on $[n] \times [n]$, tests if it is monotone or is far away from any monotone distribution in L_1-norm; this algorithm uses $\tilde{O}(n^{3/2})$ samples.

The problems of testing if two monotone distributions are close in L_1-norm and if two random variables with a monotone joint distribution are close to being independent in L_1-norm are also considered. Algorithms for these problems that use only $\mathrm{poly}(\log n)$ samples are presented. The closeness and independence testing algorithms for monotone distributions are significantly more efficient than the corresponding algorithms as well as the lower bounds for arbitrary distributions.

Some of the above results are also extended to unimodal distributions.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; G.3 [**Mathematics of Computing**]: Probability and Statistics

General Terms

Algorithms, Theory

Keywords

Sublinear algorithms, property testing, distribution testing, monotone and unimodal distributions

1. INTRODUCTION

Consider the following scenarios:

(1) Suppose one is studying the outbreak of a certain type of cancer and need to uncover any salient statistical properties of it that might hold. For example, it would be important to know if the probability of contracting the disease is monotone decreasing with the distance of one's home from Chernobyl. Once this is established, then one might want further information—such as if the distribution is close to the distribution of asthma. For obvious reasons, it is important to notice such trends using as few samples as possible.

(2) Suppose one is studying the performance of individuals in a standardized test. For example, it would be useful to know if age of the participant and score they obtain in the test are correlated at all. Furthermore, suppose that the distribution of the ages of the participants is normal and so is the distribution of the scores. Can one conclude that the distribution of the scores is independent of the distribution of the ages of the participants? Again, it is desirable to assess this using as few samples as possible.

In this paper we focus on two specific properties of distributions. The first is (decreasing) *monotonicity*, i.e., for some partial order on the underlying domain and two elements $x \prec y$ in the domain, the probability of x in the distribution is at least as big as the probability of y. The second is *unimodality*, which characterize distributions that have a single "peak."

There are several reasons to focus attention on the monotonicity and unimodality properties in the context of distributions. Many commonly studied distributions are either monotone or unimodal, or can be described as a combination of a small number of monotone distributions; familiar examples include Gaussian, Cauchy, exponential, and Zipf distributions. Moreover, tails of distributions occurring in natural phenomena are often monotone. The importance of such distributions motivates the problem of testing if a distribution is monotone/unimodal (Scenario (1)).

[*]The first author was partially supported by NSF Grant No. CCR-9912428 and a David and Lucile Packard Fellowship for Science and Engineering.
[†]Part of this work was done while the author was at NEC Laboratories America.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
STOC'04, June 13–15, 2004, Chicago, Illinois, USA.
Copyright 2004 ACM 1-58113-852-0/04/0006 ...$5.00.

The monotonicity property of distributions has been exploited in statistics, for example, in order to quickly generate random variables [5]. In [1], it has been shown that estimating the entropy of a distribution can be performed using exponentially fewer samples when the distribution is known to be monotone. This leads us to further investigate when one can exploit monotonicity/unimodality in getting more efficient algorithms for testing properties of distributions (Scenario (2)).

1.1 Summary of our results

We first focus on understanding the complexity of testing whether a distribution is monotone. Our main result is to show that the complexity of monotonicity testing for a distribution on $[n]$ is essentially the same (up to polylogarithmic factors) as that of testing uniformity, which is known to be $\tilde{\Theta}(\sqrt{n})$. We build on this basic algorithm to obtain a sublinear monotonicity testing algorithm for higher dimensions—for instance, the monotonicity testing algorithm for a distribution on $[n]\times[n]$ runs in time $\tilde{O}(n^{3/2})$. In this case, we show a lower bound of $\Omega(n)$. We next show that (as is the case with estimating the entropy) when distributions are known to be monotone, the tasks of testing if two distributions are close, or whether a joint distribution is independent, are (near-exponentially) easier than the general case.

Monotonicity testing. We begin by investigating algorithms that test if a distribution is monotone. It is tempting to construct an algorithm for testing monotonicity based on sampling: say, partition the domain into equal or unequal intervals, estimate the weight of the distribution in these intervals by sampling, and verify that the average weights are monotone. However, this naive approach fails. For instance, consider the distribution that is uniform on the even labeled domain points and zero on the odd labeled domain points. This distribution is far from any monotone distribution, but a test based purely on testing the monotonicity of weights of various partitions of the domain will be fooled.

The above example points to an intriguing relationship between the problems of testing monotonicity and testing closeness to uniformity in distributions. On one hand, the problem of testing monotonicity seems to be as hard as uniformity testing. We present a reduction showing that this is indeed the case, and thus monotonicity testing requires $\Omega(\sqrt{n})$ samples. On the other hand, could testing monotonicity be a much harder problem? One of our contributions is to show that, at least in the one-dimensional case, it cannot.

In the one-dimensional case, we reduce the problem of testing monotonicity to the problem of testing uniformity by showing how to recursively break up the domain of the distribution into a small number of *balanced intervals* (see Section 3), i.e., intervals for which the collision probability of the distribution is close to that of the uniform distribution. Since distributions that have low collision probability are known to be statistically close to uniform, as long as the average probability in each of the above intervals is monotone, the whole distribution must be close to monotone. Our techniques implicitly show that any monotone distribution can be approximated by a decomposition into a small (polylogarithmic in the size of the support) number of balanced intervals. We also show that this characterization is robust: it is not possible to decompose a distribution that is far from monotone into a small number of such balanced intervals. The biggest difficulty to overcome in showing this characterization is that a monotone distribution may be close to uniform on an interval, but still may not have a small enough collision probability, causing the algorithm to further subdivide the interval. A crucial fact that is used to upper bound the number of balanced intervals required to accurately represent monotone distributions is that the intervals can be linearly ordered such that the average weights of many consecutive intervals are substantially decreasing. We believe that this characterization of monotone distributions is interesting in its own right and might have other applications.[1]

Extending this approach to higher dimensions is tricky. The main reason is that the natural extension of intervals is to rectangles, which cannot be totally ordered according to the weights, but only partially ordered. Thus our crucial fact from the one-dimensional case does not give us a very strong bound on the number of rectangles in the decomposition. For those rectangles whose collision probability is not small enough to guarantee that their conditional distribution is close to uniform, we generalize the one-dimensional arguments in two new ways. First, we modify the recursive decomposition in such a way that rectangles that are "too far" from the origin are ignored. To argue that the error made by this truncation step is bounded, we look at a path decomposition of an appropriate partial order and upper bound both the maximum chain length and the total error contributed by any anti-chain. Second, rather than recursing, we perform a specialized test on *balanced rectangles* where the weight of the left half of the rectangle is almost the same as the right half. For such rectangles, we show that if the given distribution is monotone, then it is close to uniform on a large fraction of columns in a balanced rectangle. Thus, we would like to test monotonicity of these rectangles by testing uniformity of the columns. Unfortunately, existing uniformity tests may not pass distributions that are only guaranteed to be close to uniform. We overcome this barrier by showing how to use the one-dimensional monotonicity testing algorithm in order to give a specialized uniformity test. Finally, since the marginal distribution on the rows of the balanced rectangle is monotone, we invoke the characterization from the one-dimensional case to argue that the rows can be partitioned into intervals that are close to uniform. This induces a partitioning of the balanced rectangle into strips of columns where each strip is close to uniform. As in the one-dimensional case, we prove that if such a decomposition is possible, then it can be patched together into a monotone distribution. This approach yields a monotonicity testing algorithm that runs in $\tilde{O}(n^{3/2})$ time. These ideas can be extended to higher dimensions with a sublinear running time of $\tilde{O}(n^{d-1/2})$; a lower bound of $\Omega(n^{d/2})$ is shown.

Monotone closeness and independence. We next consider the problem of testing whether two monotone distributions are close in L_1-norm—that is, to distinguish pairs of distri-

[1]We note that there is also an algorithm for partitioning a monotone distribution into intervals such that the conditional distribution is close to uniform in each interval [1]. However, the analysis of this algorithm makes strong use of the fact that the distribution is already known to be monotone. Thus, the algorithm that performs the partitioning can use simpler properties by which to make its decisions, and the analysis of the size of the partition is stronger, as well as significantly simpler.

butions that are identical from pairs of distributions that are far in L_1-norm. For this problem, we construct a test that uses only poly$(\log n)$ samples (Section 6.1). We also consider the problem of testing whether d random variables with a monotone joint distribution are close to independent—that is, to distinguish the case in which the distributions are independent from the case in which they are far in L_1-norm from any independent distribution. Once again, we construct a test that uses only poly$(d \log n)$ samples (Section 6.2). Here we make use of the work of [1], which allows us to decompose a known monotone distribution into a small number of uniform distributions.

Our monotone closeness testing algorithm should be contrasted against the $\Omega(n^{2/3})$ lower bound for testing closeness for arbitrary distributions [3]. Similarly, our monotone independence testing algorithm should be viewed in light of an $\Omega(n)$ lower bound for testing independence for arbitrary distributions [2]. Thus, the complexity of testing these properties of monotone distributions is near-exponentially smaller than that of testing the same properties of arbitrary distributions.

Unimodal distributions and other models. By suitably adapting the algorithms in the monotone case, we obtain algorithms for testing if a given distribution is unimodal and if two unimodal distributions are close in L_1-norm (Section 7). The sample complexities and the running times of these algorithms are almost the same as in the monotone case.

For comparison, we also consider the problem of testing monotonicity in the evaluation oracle model when an oracle access to the cumulative distribution is available to the algorithm. We obtain an $O(\log^2 n)$ algorithm (Section 8).

1.2 Related work

When no assumptions are made on the distributions, standard statistical tests, such as the χ^2-test and the straightforward use of Chernoff bounds in order to estimate various properties of the distribution, seem to require a number of samples that is superlinear in the domain size for the above tasks. However, there have been several recent works that achieve sublinear complexity for testing various properties of arbitrary distributions in the L_1-norm. From the work of [9], it can be seen that there is an $\tilde{O}(\sqrt{n})$-time algorithm to test if a given distribution is close to the uniform distribution; it is also known that this is almost optimal. This result was subsequently generalized in [2], where an algorithm using $\tilde{O}(\sqrt{n})$ samples was presented to test if a distribution is close to another, where the latter's probability distribution function is available as an advice to the algorithm.

In [3], it is shown that $\tilde{O}(n^{2/3})$ time is sufficient for distinguishing pairs of distributions that are close in L_1-norm from pairs of distributions that are far (this is also shown to be tight up to polylogarithmic factors); in contrast, it is also shown that one can approximate the distance in L_2-norm in time independent of n. In [2], it is shown that for a joint distribution of two variables over $[n] \times [m]$ (without loss of generality, assuming $n \geq m$), $\tilde{O}(n^{2/3}m^{1/3})$ time is sufficient for distinguishing the case when the two variables are independent from the case in which the joint distribution is far from any independent distribution (this is again shown to be tight up to polylogarithmic factors).

Finally, in [1], the number of samples needed to approximate the entropy is studied and for distributions with sufficiently high entropy, one can get a γ-multiplicative approximation of the entropy with $\tilde{O}(n^{1/\gamma^2})$ samples. In that paper, an $\Omega(n^{1/2\gamma^2})$ lower bound on the sample size was shown for approximating the entropy. However, it is also shown that for monotone distributions, only polylogarithmically many samples are needed in order to approximate the entropy. In fact, as we have already mentioned, we build on their ideas in our algorithms for testing closeness and independence of distributions that are known to be monotone.

Monotonicity, as a property on posets, has been extensively studied in the context of property testing [7, 4, 10, 6, 8]. In this setting, the model is the evaluation oracle model where the value of function at any point in the domain can be queried. In contrast, our result can be viewed as testing monotonicity property in the generation oracle model.

2. PRELIMINARIES

We consider discrete probability distributions over $[n]$. Let $\mathsf{p} = \langle p_1, \ldots, p_n \rangle$ be such a distribution where $p_i \geq 0, \sum_{i=1}^{n} p_i = 1$. We assume that all distributions are given via *generation oracles*: for distribution p over $[n]$, each invocation of the oracle supplies us with an element in $[n]$ distributed according to p and chosen independently of all previous oracle invocations. The parameters of interest are the number of samples and running time required by the algorithm. For simplicity, we will assume that n is a power of 2; this is without loss of generality.

We use $|\mathsf{p} - \mathsf{q}|$ to denote the L_1-distance[2] and $\|\mathsf{p} - \mathsf{q}\|$ to denote the L_2-distance between two distributions. We call a distribution p to be *ϵ-close* in L_1-norm to a distribution q if $|\mathsf{p} - \mathsf{q}| \leq \epsilon$. In particular, p is ϵ-close in L_1-norm to uniform if $|\mathsf{p} - U_n| \leq \epsilon$ where U_n is the uniform distribution on $[n]$. The following fact upper bounds the collision probability when the maximum and minimum probability values are not too far away from each other [3, 2].

LEMMA 1 ([3, 2]). *Let p be a distribution on $[n]$. If $\max_i p_i \leq (1+\epsilon) \cdot \min_i p_i$, then $\|\mathsf{p}\|^2 \leq (1+\epsilon^2)/n$. If $\|\mathsf{p}\|^2 \leq (1+\epsilon^2)/n$, then $|\mathsf{p} - U_n| \leq \epsilon$.*

We now formally define monotone and unimodal distributions. Unless otherwise specified, for this paper, monotone means monotone decreasing.

DEFINITION 2 (MONOTONE DISTRIBUTIONS). *A distribution p on $[n]$ is said to be* monotone *if $p_1 \geq \cdots \geq p_n$. A distribution p on $[n]$ is said to be* ϵ-monotone *in L_1-norm if there is a monotone distribution q on $[n]$ such that $|\mathsf{p} - \mathsf{q}| \leq \epsilon$.*

The notions of monotonicity and ϵ-monotonicity naturally extend to higher dimensions, when a partial order is imposed on the domain. For instance, in two dimensions, distribution p on $[n] \times [n]$ is monotone if $p_{i,j} \geq p_{i',j'}$ whenever $i \leq i'$ and $j \leq j'$.

DEFINITION 3 (UNIMODAL DISTRIBUTIONS). *A distribution p on $[n]$ is said to be* unimodal *if there exists an $i \in [n]$ such that $p_1 \leq \cdots \leq p_i \geq p_{i+1} \geq \cdots \geq p_n$. A distribution p on $[n]$ is said to be* ϵ-unimodal *in L_1-norm if there is a unimodal distribution q on $[n]$ such that $|\mathsf{p} - \mathsf{q}| \leq \epsilon$.*

[2]The commonly used *total variation distance* between distributions is defined to be half of the L_1-distance between distributions.

Notation. For $i, j \in \mathbb{Z}$ where $i \le j$, we (ab)use the interval notation $[i, j]$ to refer to the set $\{k \in \mathbb{Z} \mid i \le k \le j\}$. For a sample set S and $i \in [n]$, $\mathsf{occ}(i, S)$ denotes the number of times i occurs in S; for $I \subseteq [n]$, $\mathsf{occ}(I, S) \stackrel{\text{def}}{=} \sum_{i \in I} \mathsf{occ}(i, S)$. We also use S_I to denote the samples in S from the interval I. Given a function f defined over domain D, for $D' \subseteq D$, we use $f(D')$ to denote $\sum_{x \in D'} f(x)$. In particular, given a distribution p on $[n]$ and an interval I in $[1, n]$, $\mathsf{p}(I)$ will denote $\sum_{i \in I} p_i$. For an interval $I = [i, i + 2k - 1]$, we use $I^\ell = [i, i + k - 1]$ and $I^r = [i + k, i + 2k - 1]$ to denote its bisection. For a rectangle $K = I \times J \subseteq [n] \times [n]$ and $b, b' \in \{\ell, r\}$, we use $K^{b,b'}$ to denote the quadrant $I^b \times J^{b'}$.

3. BALANCED INTERVALS

A recurring technique in our algorithms in this paper is to reduce the complexity of the problem by partitioning the domain into subdomains where the conditional distribution is almost uniform. Weaker variants of this technique are implicit in some of the earlier work mentioned above.

Consider a monotone distribution p on $[n]$ and an interval in $[n]$. Intuitively, if the weight of p in the first half of an interval is nearly the same as its weight in the second half, then the conditional distribution of p over the interval must be close to uniform. The following lemma formalizes this intuition quantitatively.

Lemma 4. *Let $I \subseteq [n]$ be an interval of length $2k$ and let p be a monotone distribution on $[n]$. If $\mathsf{p}(I^\ell) \le (1+\epsilon) \cdot \mathsf{p}(I^r)$, then $\sum_{i \in I} \left| p_i - \frac{\mathsf{p}(I)}{2k} \right| \le \epsilon \mathsf{p}(I)$.*

Proof. We define $w \stackrel{\text{def}}{=} \mathsf{p}(I)$ and $\delta_i \stackrel{\text{def}}{=} \left|p_i - \frac{w}{2k}\right|$. Let j be the largest index in I such that $p_j \ge w/2k$. First consider the case when $j \le k$. Let $A_1 \stackrel{\text{def}}{=} \sum_{i \in I, i \le j} \delta_i$, $A_2 \stackrel{\text{def}}{=} \sum_{i \in I, j < i \le k} \delta_i$, $A_3 \stackrel{\text{def}}{=} \sum_{i \in I, k < i \le 2k} \delta_i$. We want to show that $A_1 + A_2 + A_3 \le \epsilon w$. Note that $A_1 = A_2 + A_3$. By the assumption, we have

$$\frac{A_1 - A_2 + w/2}{-A_3 + w/2} = \frac{\mathsf{p}(I^\ell)}{\mathsf{p}(I^r)} \le 1 + \epsilon.$$

By substituting $A_2 + A_3$ for A_1, we get $A_3 \le (\epsilon w)/(4 + 2\epsilon)$. By $\delta_{j+1} \le \delta_{j+2} \le \cdots \le \delta_{2k}$, $A_2 \le A_3$. Therefore, we have $A_1 + A_2 + A_3 \le \epsilon w$. The case $j > k$ is similar. $\square$

A weaker, but analogous result can be obtained for a monotone distribution on $[n]^d$.

Lemma 5. *Let $I_1, \ldots, I_d \subseteq [n]$ be intervals. Let p be a monotone distribution on $I_1 \times \cdots \times I_d$. If $\mathsf{p}(I_1^\ell \times \cdots \times I_d^\ell) \le (1 + \epsilon)\mathsf{p}(I_1^r \times \cdots \times I_d^r)$ and $w = \mathsf{p}(I_1 \times \cdots \times I_d)$, then*

$$\sum_{\mathsf{j} \in I_1 \times \cdots \times I_d} \left| p_{\mathsf{j}} - \frac{w}{\prod_{i \in [d]} |I_i|} \right| \le 2\epsilon w.$$

Proof. Let $R = \prod_{i \in [d]} |I_i|$, namely, the size of the d-dimensional rectangle. Without loss of generality, assume that for all $\mathsf{j} \in I_1^\ell \times \cdots \times I_d^\ell$, $p_{\mathsf{j}} \ge w/R$. The argument is analogous when for all $\mathsf{j} \in I_1^r \times \cdots \times I_d^r$, $p_{\mathsf{j}} \le w/R$. (Note that if there exists a $\mathsf{j} \in I_1^\ell \times \cdots \times I_d^\ell$ such that $p_{\mathsf{j}} < w/R$, then monotonicity implies that $p_{\mathsf{j}'} \le w/R$ for all $\mathsf{j}' \in I_1^r \times \cdots \times I_d^r$.)

For each $\mathsf{b} \in \{\ell, r\}^d$, let $A_{\mathsf{b}} \stackrel{\text{def}}{=} \{\mathsf{j} \in I_1^{b_1} \times \cdots \times I_d^{b_d} \mid p_{\mathsf{j}} \ge w/R\}$, and let $W_{\mathsf{b}} \stackrel{\text{def}}{=} \sum_{\mathsf{j} \in A_{\mathsf{b}}} (p_{\mathsf{j}} - w/R)$. Finally, let t be the d-dimensional all-ℓ's vector, i.e., $\mathsf{t} \stackrel{\text{def}}{=} \langle \ell, \ell, \ldots, \ell \rangle$. Note that for any $\mathsf{b} \in \{\ell, r\}^d$, $W_{\mathsf{b}} \le W_{\mathsf{t}}$ by the monotonicity of f.

Since $\mathsf{p}(I_1^\ell \times \cdots \times I_d^\ell) = W_{\mathsf{t}} + (w/2^d)$ and $\mathsf{p}(I_1^r \times \cdots \times I_d^r) \le w/2^d$, by the assumption in the lemma statement, we know that

$$\frac{W_{\mathsf{t}} + (w/2^d)}{(w/2^d)} \le (1 + \epsilon).$$

Hence, $W_{\mathsf{t}} \le (\epsilon w)/2^d$. So, we can conclude

$$\sum_{\mathsf{j} \in I_1 \times \cdots \times I_d} \left| p_{\mathsf{j}} - \frac{w}{R} \right| = 2 \sum_{\mathsf{b} \in \{\ell, r\}^d} W_{\mathsf{b}} \le 2^{d+1} \cdot W_{\mathsf{t}} \le 2\epsilon w. \quad \square$$

4. MONOTONICITY IN ONE DIMENSION

We consider the problem of distinguishing monotone decreasing distributions from those that are not ϵ-monotone in L_1-norm. We give an $\tilde{O}(\sqrt{n})$ algorithm that reduces the problem of testing if a distribution is close to monotone to the problem of testing if several distributions are close to uniform. Our reduction can be viewed as a structural decomposition of a monotone distribution into several uniform distributions. This reduction is robust in the sense that the resulting testing algorithm will pass monotone distributions and fail distributions that are not ϵ-monotone. We then essentially match this upper bound by showing that any algorithm for this problem requires $\Omega(\sqrt{n})$ samples, by reducing the problem of testing whether a distribution is close to uniform to monotonicity testing.

Our algorithm partitions the domain $[n]$ into a small (i.e., poly$(\log n)$) number of intervals, each of which has its weight distributed roughly evenly over the elements in the interval. The conditional distribution on such an interval is close to the uniform distribution such that each element in the interval has probability close to the average probability.

Once the desired partition is obtained, our algorithm then determines whether the uniform distributions in each of the intervals can be "patched" together to form a monotone distribution over the whole domain that is close to the original distribution. Since there are very few intervals, this latter task can be performed efficiently via linear programming.

Flat distributions. We define flat distributions, which are reminiscent of histograms.

Definition 6 (Flat distribution). *Let ℓ be an integer and let $\mathcal{I}_\ell = \langle I_1, \ldots, I_\ell \rangle$ be a partition of $[n]$. A distribution q on $[n]$ is called an ℓ-*flat *distribution if it can be described by the pair $(\mathsf{w}, \mathcal{I}_\ell)$, with $\mathsf{w} = \langle w_1, \ldots, w_\ell \rangle$ and $q_i = w_j / |I_j|$ for $i \in I_j$.*

Flat distributions are interesting to us for the following two reasons. Firstly, flatness is a robust property with respect to monotonicity, that is, a flat distribution is ϵ-close to monotone if and only if it is ϵ-close to a monotone flat distribution.

Lemma 7. *An ℓ-flat distribution p described by $(\mathsf{w}, \mathcal{I}_\ell)$ is ϵ-monotone if and only if p is ϵ-close to a monotone flat distribution.*

Proof. It is clear that if p is ϵ-close to a monotone flat distribution, then p is ϵ-monotone. For the converse, let

q be a monotone distribution such that $|\mathsf{p}-\mathsf{q}| \le \epsilon$. Let $\mathsf{w} = \langle w_1,\ldots,w_\ell\rangle$ such that $p_i = w_j/|I_j|$ if $i \in I_j$. Define the monotone ℓ-flat distribution q' that is described by the pair $(\mathsf{w}', \mathcal{I}_\ell)$ where $\mathsf{w}' = \langle w'_1,\ldots,w'_\ell\rangle$ and $w'_j = \mathsf{q}(I_j), j \in [\ell]$. We now show that q' is ϵ-close to p. By the flatness property and triangle inequality,

$$\begin{aligned}|\mathsf{p}-\mathsf{q}'| &= \sum_{j\in[\ell]} |w_j - w'_j| = \sum_{j\in[\ell]} |w_j - \mathsf{q}(I_j)| \\ &= \sum_{j\in[\ell]} \left|\sum_{i\in I_j} \frac{w_j}{|I_j|} - q_i\right| \le \sum_{j\in[\ell]}\sum_{i\in I_j} \left|\frac{w_j}{|I_j|} - q_i\right| \\ &= \sum_{i\in[n]} |p_i - q_i| = |\mathsf{p}-\mathsf{q}| \le \epsilon. \quad \square\end{aligned}$$

Secondly, ϵ-monotonicity of ℓ-flat distributions can be tested in time polynomial in ℓ.

Lemma 8. *There is an algorithm that outputs PASS if an ℓ-flat distribution described by $(\mathsf{w}, \mathcal{I}_\ell)$ is ϵ-close to a monotone flat distribution, and outputs FAIL otherwise. The running time of this algorithm is* $\mathrm{poly}(\ell)$.

Proof. For $j \in [\ell]$, let $k_j = |I_j|$ and let $\mathsf{w} = \langle w_1,\ldots,w_\ell\rangle$. First, consider the following mathematical program with variables y_i for $i \in [l]$:

$$\begin{aligned}&\min \textstyle\sum_{j\in[\ell]} |y_j k_j - w_j| \qquad \text{s. t.}\\ &\textstyle\sum_{j\in[\ell]} y_j k_j = 1; \qquad y_j \ge y_{j+1} \text{ for all } j \in [\ell-1];\\ &y_\ell \ge 0.\end{aligned}$$

The program above minimizes the distance between the given distribution and a monotone ℓ-flat distribution. We now transform this into a linear program by introducing a new variable z_j to correspond to $|y_j k_j - w_j|$ and by adding the constraints $z_j \ge y_j k_j - w_j$ and $-z_j \le y_j k_j - w_j$, for all $j \in [\ell]$. It is easy to see that this transformation preserves the value of the objective function. The lemma follows by solving this LP in time $\mathrm{poly}(\ell)$, for example, using [11]. $\square$

The algorithm. The *collision count* of S_I, the samples in an interval I, is defined to be $\mathtt{coll}(S_I) \overset{\text{def}}{=} \sum_{i\in I} \binom{\mathtt{occ}(i,S_I)}{2}$. The following lemma relates the collision count to the L_2-norm [3, 2]:

Lemma 9 ([3, 2]). *Let I be an interval and q be the conditional distribution of p on I. Then,*

$$\left(\|\mathsf{q}\|^2 - \frac{\epsilon^2}{32|I|}\right) \le \frac{\mathtt{coll}(S_I)}{\binom{|S_I|}{2}} \le \left(\|\mathsf{q}\|^2 + \frac{\epsilon^2}{32|I|}\right),$$

with probability at least $1 - O(\log^{-3} n)$, provided that $|S_I| = \Omega(\epsilon^{-4}\sqrt{|I|}\log\log n)$.

Our algorithm will use collisions in the sample to determine a partition $\mathcal{I}_\ell$ of $[1,n]$. Since a low count of collisions in the sample suggests a nearly uniform distribution of the weight, the use of this statistic will result in a partition with close-to-uniform conditional distributions on each interval. After obtaining a partition $[1,n]$, the algorithm will check if these close-to-uniform conditional distributions can be patched together into a monotone flat distribution.

Now we describe our algorithm. The inputs are a generation oracle for p and an error parameter ϵ.

Algorithm TestMonotonicity

1. Obtain $m \overset{\text{def}}{=} O(\epsilon^{-4}\sqrt{n}\log n)$ samples S from p.

   Start with the interval $I = [1,n]$, and bisect I in half recursively as long as
   $$\mathtt{coll}(S_I) \ge \frac{(1+\epsilon^2/32)}{|I|}\binom{|S_I|}{2} \text{ and } |S_I| \ge \frac{m}{\log^3 n}.$$
   Abort the algorithm, and output FAIL if it performed more than $O(\epsilon^{-1}\log^2 n)$ splits.

2. Let $\mathcal{I}_\ell = \langle I_1,\ldots,I_\ell\rangle$ denote the partition of $[1,n]$ into intervals induced by the leaves of the recursion from the previous step.

3. Obtain an additional sample T of size $O(\epsilon^{-2}\log^4 n)$.

4. Let $\mathtt{hist}(T,\mathcal{I}_\ell)$ denote the ℓ-flat distribution described by $(\mathsf{w},\mathcal{I}_\ell)$ where $w_j = \mathtt{occ}(I_j,T)/|T|$.

5. Output PASS if $\mathtt{hist}(T,\mathcal{I}_\ell)$ is $\epsilon/2$-close to a monotone distribution (by using the algorithm from Lemma 8), otherwise output FAIL.

Thus, we obtain the following theorem.

Theorem 10. *Given access to a generation oracle for p over $[n]$, the algorithm* **TestMonotonicity** *outputs PASS when p is monotone and outputs FAIL when p is not ϵ-monotone, with probability at least $2/3$. The algorithm uses $O(\epsilon^{-4}\sqrt{n}\log n)$ samples and runs in time $O(\epsilon^{-4}\sqrt{n}\log^3 n)$.*

Proof. We first argue that $\mathtt{hist}(T,\mathcal{I}_\ell)$ is a good approximation to p, assuming Step (1) succeeds, i.e., when $\ell = O(\epsilon^{-1}\log^2 n)$. Consider the partition $\mathcal{I}_\ell$ obtained from Step (1). Call an interval I *light* if $\mathtt{occ}(I,T) \le m/\log^3 n$; call it *balanced* otherwise. By Lemma 9, the L_2-norm of the conditional distribution on a balanced interval I is at most $(1+\epsilon^2/16)/|I|$. Hence, by Lemma 1, we can claim that $\sum_{i\in I} |p_i - \mathsf{p}(I)/|I|| \le \epsilon\mathsf{p}(I)/4$. The total weight of the light intervals is less than $\epsilon/4$ by the virtue of being light and the upper bound on ℓ. Hence, by summing over all intervals $I \in \mathcal{I}_\ell$, we get $|\mathtt{hist}(T,\mathcal{I}_\ell) - \mathsf{p}| \le \epsilon/2$.

Suppose p is a monotone distribution. We show that the algorithm will output PASS with probability at least $2/3$. We first show that $\ell = O(\epsilon^{-1}\log^2 n)$, i.e., Step (1) will succeed. Using Lemma 9 and the union bound over all intervals, Step (1) will obtain a reliable estimate, as given by Lemma 9, for the collision probability (i.e., the square of the L_2-norm) with probability at least $2/3$. Now, fix a level and consider the internal nodes in this level of the recursive tree constructed in Step (1). For an interval that corresponds to one of these nodes, the ratio of the maximum probability to the minimum probability in the interval is at least $1+\epsilon/8$ by Lemma 1 and Lemma 9; by monotonicity of p, these extrema occur at the two ends of the interval. Since for each of these intervals, the maximum probability is at least n^{-2}, there are at most $O(\log_{1+\epsilon/8} n)$ internal nodes on any level. Therefore, the tree has $O(\epsilon^{-1}\log^2 n)$ internal nodes. Finally, since the tree is a complete binary tree, $\mathcal{I}_\ell$ contains $O(\epsilon^{-1}\log^2 n)$ intervals. Consequently, Step (1) will succeed and as we argued in the beginning of this proof, $|\mathtt{hist}(T,\mathcal{I}_\ell) - \mathsf{p}| \le \epsilon/2$. Since $\mathtt{hist}(T,\mathcal{I}_\ell)$ is ℓ-flat and is $\epsilon/2$-close to a monotone distribution, by Lemma 7, it is also

$\epsilon/2$-close to a monotone ℓ-flat distribution and so Step (5) will also succeed and the algorithm will output PASS.

Suppose the algorithm outputs PASS. Then, from Step (5), there is a monotone (in fact, flat) distribution q such that $|\mathtt{hist}(T, \mathcal{I}_\ell) - \mathsf{q}| \leq \epsilon/2$. Moreover, since Step (1) succeeded, $\ell = O(\epsilon^{-1}\log^2 n)$. Again, as we argued in the beginning of this proof, this implies that $|\mathtt{hist}(T, \mathcal{I}_\ell) - \mathsf{p}| \leq \epsilon/2$. By triangle inequality, $|\mathsf{p} - \mathsf{q}| \leq \epsilon$.

We also need to ensure that Step (5) can be implemented in polylogarithmic time. But, this is possible via Lemma 8 with $\ell = O(\epsilon^{-1}\log^2 n)$. Hence, the running time of the algorithm is dominated by the previous steps. $\square$

Lower bound. Next, we show that our upper bound is essentially optimal.

Theorem 11. *Let $\mathcal{A}$ be an algorithm that, given generation oracle access to an arbitrary distribution p over $[n]$, has the following behavior: if p is monotone, then $\mathcal{A}$ outputs PASS and if p is not ϵ-monotone in L_1-norm, then $\mathcal{A}$ outputs FAIL. Furthermore the error probability of $\mathcal{A}$ is at most 1/3. Then $\mathcal{A}$ requires $\Omega(\sqrt{n})$ samples.*

Proof. We reduce the problem of testing if a given distribution is close to uniform to testing if the distribution is close to monotone. Given a generation oracle, distinguishing uniform distributions from distributions that are not even ϵ-close to uniform in L_1-norm is known to require $\Omega(\sqrt{n})$ samples [2]. The desired lower bound for testing monotonicity will follow.

For simplicity, we assume p is a distribution on $[2n]$. Suppose there is an algorithm $\mathcal{A}$ to test if a given distribution is ϵ-monotone for $\epsilon < 1/16$. We use this algorithm as a black-box to construct a new algorithm $\mathcal{B}$ that tests if a given distribution is $(\epsilon' + \epsilon)$-close to uniform, where $\epsilon' = 4\epsilon/(1/2 - \epsilon) < 1$. Define a new generation oracle p^R that on invocation, invokes p and outputs i if p outputs $2n - i$. The algorithm $\mathcal{B}$ uses both p and p^R in the following manner: it runs $\mathcal{A}$ on both p and p^R and outputs PASS if and only if $\mathcal{A}$ outputs PASS on both p and p^R.

Clearly, if p is uniform on $[2n]$, both p and p^R are monotone and $\mathcal{A}$ outputs PASS on both occasions. Conversely, suppose $\mathcal{A}$ outputs PASS for both p and p^R; then p and p^R must be ϵ-monotone. In this case, we show that p is $(\epsilon' + \epsilon)$-close to uniform. Let f be a monotone distribution on $[2n]$ such that $|\mathsf{p} - \mathsf{f}| \leq \epsilon$ and g^R be a monotone distribution on $[2n]$ such that $\left|\mathsf{p}^R - \mathsf{g}^R\right| \leq \epsilon$. By our construction of p^R, the function g defined by $g_i = g^R_{2n-i}$ is monotone non-decreasing and $|\mathsf{p} - \mathsf{g}| = \left|\mathsf{p}^R - \mathsf{g}^R\right| \leq \epsilon$. We conclude the proof by showing that f is ϵ'-close to uniform, which by the triangle inequality will show that p is $(\epsilon' + \epsilon)$-close to uniform.

Let $F = \mathsf{f}([1,n]) = \sum_{i=i}^n f_i$, $F' = 1 - F = \mathsf{f}([n+1, 2n]) = \sum_{i=n+1}^{2n} f_i$, and $G = \mathsf{g}([1,n]) = \sum_{i=1}^n g_i$. Since f is monotone non-increasing and g is monotone non-decreasing, we have $G \leq 1/2 \leq F$. On the other hand, by the triangle inequality, $|\mathsf{f} - \mathsf{g}| \leq |\mathsf{f} - \mathsf{p}| + |\mathsf{p} - \mathsf{g}| \leq 2\epsilon$ and in particular, $|F - G| \leq 2\epsilon$. Combining all of these, $1/2 \leq F \leq G + 2\epsilon \leq 1/2 + 2\epsilon$ and consequently, $F' \geq 1/2 - 2\epsilon$. We have

$$\begin{aligned} 0 \leq F - F' = 2F - 1 &\leq 2\left(\frac{1}{2} + 2\epsilon\right) - 1 \\ = \quad 4\epsilon &\leq \frac{4\epsilon}{1/2 - 2\epsilon} F' = \epsilon' F'. \end{aligned}$$

Also, since f is a distribution, $\sum_{i=1}^{2n} f_i = 1$. By appealing to Lemma 4, we see that $\sum_{i=1}^{2n} |f_i - 1/(2n)| \leq \epsilon'$, that is, f is ϵ'-close to uniform. $\square$

5. MONOTONICITY IN HIGHER DIMENSIONS

In this section we present an $\tilde{O}(n^{3/2})$ algorithm for testing monotonicity in two dimensions. These ideas can be used to give $\tilde{O}(n^{d-1/2})$-time algorithms for testing monotonicity in d dimensions. We also show a lower bound of $\Omega(n^{d/2})$ for testing monotonicity in d dimensions.

As we indicated in the Introduction, testing monotonicity in higher dimensions is trickier than the one-dimensional case because of the partial ordering on the domain. Similar to the one-dimensional case, we would like to recursively subdivide the domain until we can test for closeness to uniformity of each subdivision. In order to upper bound the running time and the error probability of the algorithm, we would need a bound on the number of subdivisions, as in the one-dimensional case. However, the partial ordering on the domain hinders the argument that bounds the number of subdivisions. To handle this problem, we make the following observations: (i) the quadrants that are "too far" from the origin can be discarded since they cannot contribute significant mass to a monotone distribution; (ii) if consecutive quadrants have similar weights, they can be further decomposed into almost-uniform partitions in one step.

Let the domain of the distribution be $[n] \times [n]$. We will think of the algorithm as partitioning the two dimensional space into four equal quadrants and recursing on each of the quadrants. In other words, the algorithm builds a quadtree T with the following semantics. The nodes in T will correspond to rectangles inside $[n] \times [n]$; the root of T corresponds to $[n] \times [n]$. For an internal node v that corresponds to a rectangle $K = I \times J$, the four children of v, labeled $v_{\ell,\ell}, v_{\ell,r}, v_{r,\ell}, v_{r,r}$ correspond to the four quadrants of K. Clearly, the depth of this tree is at most $\log n$.

For a given level of the tree, the *row* (resp. *column*) *distance* of a quadrant is the number of quadrants to the left (resp. bottom) of it to the origin. For a given level, the *distance* of a quadrant from the origin is the maximum of its row and column distances. We will derive a tree from T where each quadrant corresponds to a leaf in T can be further split, though not necessarily as quadrants. The size of this new tree will be essentially that of T and so we will deal with T for the rest of the discussion.

Let a be a suitably large constant and let b and c be constants such that $b > 2c + 1$ and $c > a + 2$. Let $\delta = O(\epsilon/\log^a n)$. Let S be a sample of size $O(n^{3/2} \cdot \mathrm{poly}(\log n, \epsilon^{-1}))$ and W be a global variable in the algorithm that keeps track of the number of samples ignored. We now describe how the algorithm recursively constructs the tree starting at the current node v, which corresponds to a rectangle $K = I \times J$. If a node is declared as a leaf, then we do not recurse on the node further.

Algorithm TestMonotonicity2D

```
1. If $\mathtt{occ}(K, S)/|S| \leq 1/\log^b n$, then $v$ is a leaf.

2. If $\mathtt{coll}(S_K) \leq (1 + \epsilon^2/32)\binom{|S_K|}{2}/|K|$, then $v$ is a leaf.

3. If the quadrant $K$ is more than $\log^c n$ away from the
   origin, then $v$ is a leaf. Update $W \leftarrow W + |\mathtt{occ}(K, S)|$.
```

4. If K is not already designated as a leaf, then do the following two steps for each of the following ordered pairs: $\langle K^{\ell,\ell}, K^{\ell,r}\rangle$, $\langle K^{\ell,\ell}, K^{r,\ell}\rangle$, $\langle K^{r,\ell}, K^{r,r}\rangle$, $\langle K^{\ell,r}, K^{r,r}\rangle$. We will illustrate the steps for $\langle K^{\ell,\ell}, K^{\ell,r}\rangle$.

   (4a) If $(1+\epsilon)\mathsf{occ}(K^{\ell,\ell}, S) < \mathsf{occ}(K^{\ell,r}, S)$, then output FAIL.

   (4b) If $\mathsf{occ}(K^{\ell,\ell}, S) \leq (1+\delta)\mathsf{occ}(K^{\ell,r}, S)$ then select $1/\delta$ many i's where probability of i is proportional to $\mathsf{p}(\{i\} \times J^r)$. For each i, output FAIL if the distribution along the i-th column $\{i\} \times J$ is not $(\epsilon/32)$-close to monotone or $\mathsf{p}(\{i\} \times J^\ell) > (1+\epsilon/32)\mathsf{p}(\{i\} \times J^r)$. Partition $I^\ell \times J$ into contiguous columns applying Step (1) in algorithm **TestMonotonicity** on domain I^ℓ and mark each set of columns as a leaf.

5. Recurse on the children that were not leaves in the previous step.

6. Output FAIL if $W > \epsilon|S|/8$.

7. Output FAIL if the partition of the domain induced by the leaves of the recursion is not $\epsilon/2$-close to a monotone distribution (This condition can be checked by a linear program formulation as in the one-dimensional algorithm), otherwise output PASS.

Running time. Note that the total number of nodes in this tree is $O(\log^{2c+1} n)$, which follows from the fact that at any fixed level of the tree, there are at most $\log^{2c} n$ internal nodes (from Step (3)). Thus, the sample complexity is dominated by the one-dimensional monotonicity testing in Step (4b) for poly$(\log n)$ columns each with weight at least $O(1/(n \log^b n))$. This entails $\tilde{O}(n^{3/2}/\epsilon^4)$ samples.

Also, it is easy to see that an LP-based algorithm can be designed to check if a given two-dimensional flat distribution is $\epsilon/2$-close to a two-dimensional monotone flat distribution in Step (7). Since the number of nodes in T is $\log^{2c+1} n$, the running time of this step will be overwhelmed by the other steps.

Proof overview. First note that the algorithm determines the rectangles not to be divided any further in Steps (1)–(4): such rectangles either have small weight (Step (1)), have almost uniform conditional distribution (Step (2)), are far from the origin (Step (3)), or can be further decomposed into almost-uniform partitions in one step (Step (4)). We show (Lemma 14) that the leaves designated by Step (3) have a negligible fraction of the total weight in a monotone distribution. We show that all these steps together ensure a small tree size and that the total weight of the leaves ignored by Steps (1) and (3) is negligible. Note that one cannot use the weight threshold from Step (1) both to upper bound the number of leaves and to simultaneously show that their total weight is negligible.

When a rectangle K is divided, we would like to maintain that the weights of the consecutive quadrants in K are separated by a multiplicative factor, of at least $1+\delta$, in order to ensure a tree of polylogarithmic size at the end. Hence, when the weights of two consecutive quadrants, say, $\langle K^{\ell,\ell}, K^{\ell,r}\rangle$ in K are within $(1+\delta)$, these two quadrants are not recursively divided any further. In a monotone distribution we would expect that the individual columns in these two quadrants are roughly uniform. Step (4b) ensures such quadrants can be partitioned into $O(\log^2 n)$ subdivisions, each of which is close to uniform, using Lemma 13.

At the end of Step (7), we can derive a two-dimensional flat distribution, defined similar to the one-dimensional case, that is close to p. The leaves that are determined by Step (2) correspond to flat quadrants with the total mass induced by the sample. The conditional distribution is $\epsilon/4$-close to uniform for these rectangles. For the leaves that are determined by Step (4b), we split the rectangles one more level into groups of contiguous columns (or rows, depending on the orientation of the rectangle) to obtain $(\epsilon/4)$-uniform partitions. The total weight of all the other leaves is negligible.

First, we show that for a monotone distribution, we can assume that Step (4b) will not FAIL. We show that for a monotone distribution, if the two halves have roughly the same weight, then the conditional distributions on columns are close to uniform.

LEMMA 12. *Let $\epsilon, \sigma < 1/8$. Let distribution p over interval I be ϵ-monotone. Furthermore, let $\mathsf{p}(I^\ell) \leq (1+\sigma)\mathsf{p}(I^r)$. Then, p is $(4\epsilon + 2\sigma)$-close to uniform.*

PROOF. Let f be a monotone distribution such that $|\mathsf{p} - \mathsf{f}| \leq \epsilon$. Since $\mathsf{p}(I^\ell) - \mathsf{p}(I^r) \leq 2\sigma/3$ and $|\mathsf{p} - \mathsf{f}| \leq \epsilon$, $\mathsf{f}(I^\ell) \leq \mathsf{f}(I^r) + \epsilon + 2\sigma/3$. Thus, we get

$$\frac{\mathsf{f}(I^\ell)}{\mathsf{f}(I^r)} \leq 1 + \frac{\epsilon + 2\sigma/3}{\mathsf{f}(I^r)} \leq 1 + \frac{\epsilon + 2\sigma/3}{\frac{1-\epsilon-2\sigma/3}{2}} \leq 1 + 3\epsilon + 2\sigma.$$

Hence, by Lemma 4, f is $(3\epsilon + 2\sigma)$-close to uniform. So, by the triangle inequality, p is $(4\epsilon + 2\sigma)$-close to uniform. □

The next lemma shows that in monotone distributions, for those rectangles considered by Step (4b), most of the weight in the rectangle is distributed on columns with roughly uniform conditional distribution.

LEMMA 13. *Let $I \times J \subseteq [n] \times [n]$ and let p be a monotone distribution such that $\mathsf{p}(I \times J^\ell) \leq (1+\delta)\mathsf{p}(I \times J^r)$. Then, for any $\rho > 0$, $\Pr_{i\in I}\left[\mathsf{p}(\{i\} \times J^\ell) \geq (1+\rho\delta) \cdot \mathsf{p}(\{i\} \times J^r)\right] \leq 1/\rho$, where i is chosen with probability $\mathsf{p}(\{i\} \times J^r)/\mathsf{p}(I \times J^r)$.*

PROOF. Let $W = \mathsf{p}(I \times J^\ell), W' = \mathsf{p}(I \times J^r)$. We know that $W' \leq W \leq (1+\delta)W'$. Let $w_i = \mathsf{p}(\{i\} \times J^\ell)$ and $w'_i = \mathsf{p}(\{i\} \times J^r)$. We know that $w'_i \leq w_i$ for every $i \in I$. Let B be the set of i's such that $w_i \geq (1+\delta')w'_i$, for δ' to be chosen later. Then, from the definition of B and our assumptions, we have that

$$\sum_{i\in B}(1+\delta')w'_i + \sum_{i\notin B} w'_i \leq \sum_{i\in B} w_i + \sum_{i\notin B} w_i = W \leq (1+\delta)W'.$$

From this, it follows that $\sum_{i\in B} w'_i/W' \leq \delta/\delta'$. Setting $\delta' = \rho\delta$, we see that if i is picked proportional to w'_i, the probability that it is in B is at most $1/\rho$. □

We now bound the error introduced because of ignoring nodes that are too far away from the origin in Step (3).

LEMMA 14. *For a monotone distribution p, the total error accrued at any level of T because of Step (3) is at most $O(\epsilon^{-1}\log^{a-c+1} n)$.*

Proof. Consider a graph whose nodes are the internal nodes of the tree at level ℓ. In this graph, there is an edge between two nodes if the rectangles K_1 and K_2 corresponding to these nodes have an ordering relationship between them (according to the definition of monotonicity in two dimensions) and K_1 is one of the closest rectangles to K_2 on this level (i.e., either K_1 and K_2 are touching each other or none of the rectangles of the same size in between them survived until this level).

First, we claim that the maximum length of a path in this graph is $O(\delta^{-1} \log n)$, where recall that $\delta = O(\epsilon/\log^a n)$. Consider a path of length t. One in every three edges on this path have to be between two sibling nodes in the tree, because four nodes on this path of three edges can belong to at most three parents. Note that for each edge between two siblings along the path, the weight of the quadrants drops by at least a factor of $1+\delta$. This follows from the fact that these nodes are internal nodes and Step (4b) could not be applied to them. Hence, t is $O(\delta^{-1}\log n) = O(\epsilon^{-1}\log^{a+1} n)$.

Secondly, consider any set R of incomparable nodes, all at distance at least $\log^c n$ in this graph. Let v be a node in R. Interpreting v in the partial order, without loss of generality, let the "x-coordinate" of v be at least $\log^c n$. Let $w_1, \ldots, w_k$ be the set of nodes at level ℓ of a complete quad-tree with the same y-coordinate as v and a smaller x-coordinate than v, where $k \geq \log^c n$. We know by monotonicity that $\mathsf{p}(w_1) \geq \cdots \geq \mathsf{p}(w_k) \geq \mathsf{p}(v)$. Thus, $\mathsf{p}(v)$ as a fraction of $\sum_{i=1}^{k} \mathsf{p}(w_i)$ is at most $\log^{-c} n$. We count ignoring v as an error and charge this quantity to w_k. Now, we look at the charges each node gets. We claim that each node can get charged at most twice in level ℓ—once along x-direction and another along y-direction. This follows since R was chosen to be a set of incomparable nodes. Thus, as a fraction, the total weight of the nodes in S is at most $2\log^{-c} n$.

Now, the total error caused by Step (3) is upper bounded by the product of the maximum path length and the maximum weight of incomparable nodes. By the above two observations, this is at most $O(\epsilon^{-1}\log^{a-c+1} n)$. □

Thus, we obtain:

Theorem 15. *Given access to samples from a distribution* p *over* $[n] \times [n]$*, the algorithm* **TestMonotonicity2D** *outputs PASS when* p *is monotone and outputs FAIL when* p *is not* ϵ*-monotone, with probability at least* $2/3$*. Moreover, the algorithm runs in time* $O(n^{3/2} \cdot \text{poly}(\log n, \epsilon^{-1}))$*.*

Proof. First of all, by picking sample set S large enough, we can guarantee that the error probability for any of the operations (such as counting/comparing the number of occurrences, estimating collision probabilities, or performing one-dimensional monotonicity test, etc.) at each node in T is at most $\log^{-d} n$ for some constant $d > b > 2c+1$. Since the number of nodes in T is only $\log^{2c+1} n$, this will permit us to apply a union bound over all nodes in T to guarantee that no "bad event" happens.

Second, we also assume that the sampling error in estimating various parameters (such as number of occurrences of sample in a given quadrant, selecting i's in Step (4a), counting W, etc.) is ϵ' for some $\epsilon' \ll \epsilon$.

Note that the total error due to the nodes ignored in Step (1) is at most the number of nodes in the tree multiplied by $O(1/\log^b n)$, which is $O(\log^{-b+2c+1} n)$, and so is negligible when $b > 2c+1$.

Suppose p is a monotone distribution. We show that the algorithm will output PASS with high probability. Since we assumed that the sampling is good enough, Step (4a) will never output FAIL for a monotone distribution. Coming to Step (4b), by our choice of parameters, at least $1 - 1/\Omega(\log^a n)$ fraction (by weight) of i's will be such that $w_i' \leq w_i \leq (1+\epsilon/64)w_i'$ where $w_i = \mathsf{p}(\{i\} \times J^\ell), w_i' = \mathsf{p}(\{i\} \times J^r)$ by Lemma 13. So, Step (4b) is not likely output FAIL. By Lemma 14, Step (6) will also not output FAIL. Finally, we show that the flat distribution obtained from the partition is $\epsilon/2$-monotone so that Step (7) does not output FAIL. The error due to Step (3) is the height of the tree multiplied by $O(\epsilon^{-1}\log^{a-c+1} n)$ (from Lemma 14), which is $O(\epsilon^{-1}\log^{a-c+2} n)$, and is negligible when $c > a+2$. The balanced rectangles from Step (4b) are divided into partitions each of which is $\epsilon/4$-close to uniform. The leaves designated by Step (2) also correspond to $(\epsilon/4)$-uniform rectangles. Hence, we see that the two-dimensional flat distribution is indeed $\epsilon/2$-close to p.

Suppose the algorithm outputs PASS. Since the sampling in Step (4b) is not likely to FAIL, it follows that the distributions restricted to i's are actually $\epsilon/32$-close to monotone and have the weights of the two halves within $(1+\epsilon/16)$ for at least $1 - 1/\Omega(\log^a n)$ fraction (by weight) of the i's. Hence, by Lemma 12, for those i's the distribution is $\epsilon/4$-close to uniform. If we replace columns for the rest of i's by uniform distributions, the total error resulting from this modification will be at most $\epsilon/4$. The total weight of the parts of the domain designated as leaves by steps (1) and (3) is at most $O(\log^{-b+2c+1} n) + |W|/|S| \leq \epsilon/4$. Hence, the two-dimensional flat distribution implied by the tree T is $\epsilon/2$-close to p. Finally, from the last step, there is a two-dimensional monotone flat distribution q that is $\epsilon/2$-close to the two-dimensional flat distribution implied by the tree T and the solution to the linear program constructed using T. By the triangle inequality, p and q are ϵ-close. □

Lower bound. By generalizing the lower argument in one-dimension from Section 4, we show that a lower bound on the sample complexity of testing monotonicity in higher dimensions. We reduce testing uniformity of a distribution to testing monotonicity of a distribution over tuples.

Theorem 16. *Let* $\mathcal{A}$ *be an algorithm that, given generation oracle access to an arbitrary distribution* p *over* $[n]^d$*, has the following behavior: if* p *is monotone, then* $\mathcal{A}$ *outputs PASS and if* p *is not* ϵ*-monotone in* L_1*-norm, then* $\mathcal{A}$ *outputs FAIL. Furthermore the error probability of* $\mathcal{A}$ *is at most 1/3. Then* $\mathcal{A}$ *requires* $\Omega(n^{d/2})$ *samples.*

6. CLOSENESS AND INDEPENDENCE

In this section we present efficient algorithms to test if two monotone distributions over $[n]$ are close in L_1-norm and if a monotone joint distribution is close in L_1-norm to being independent. Our algorithms run in time $O(\text{poly}(\log n))$, thereby going beyond the lower bounds for these problems in the general case [3, 2] by a near-exponential factor.

The main idea behind the algorithms is the observation underlying Lemma 4: if a monotone distribution p over $[n]$ is balanced, i.e., $\mathsf{p}([n/2])$ and $\mathsf{p}([n]\backslash[n/2])$ are close, then the distribution must be close to uniform. We require an efficient procedure that, given a monotone distribution, partitions the domain into a small number of intervals that are balanced. The next theorem from [1] comes to our rescue.

THEOREM 17 ([1]). *Let* p *be a monotone distribution on* $[n]$ *given via a generation oracle. There is a procedure* $\texttt{Partition}(\mathsf{p}, \epsilon, w)$ *that outputs a* $(k+1)$*-partition* $\mathcal{I} = \langle I_1, \ldots, I_k, J\rangle$ *of* $[n]$ *such that* I_j*'s are intervals,* $J \subseteq [n]$*, and with probability at least* $1 - o(1)$*, the following hold: (1)* $k = O(\epsilon^{-1}\log(n)\log\log n)$*; (2)* $\mathsf{p}(J) = o(wk)$*; (3) for* $j \in [k]$*,* $\mathsf{p}(I_j) > w$ *and* $\mathsf{p}(I_j^r) \leq \mathsf{p}(I_j^\ell) \leq (1+\epsilon)\cdot\mathsf{p}(I_j^r)$*. The procedure uses* $O(\epsilon^{-3}w^{-1}\log n)$ *samples from* p*.*

Notice that we could not have used Theorem 17 for testing monotonicity: `Partition` requires samples from a monotone distribution and the guarantee that it gives on the partition is weaker than the one we need for testing monotonicity.

6.1 Closeness of monotone distributions

In this section we present an algorithm to test if two monotone distributions are close. We use the algorithm in Theorem 17 to obtain a partition $\mathcal{I}_{\ell+1} = \langle I_1, \ldots, I_\ell, J\rangle$ of $[n]$. We then check if p and q are close in each of the intervals I_j and if $\mathsf{q}(J)$ is small. Here is a description of the algorithm.

Algorithm TestMonotoneCloseness

1. Let $\langle I_1, \ldots, I_k, J\rangle = \texttt{Partition}(\mathsf{p}, \epsilon, \log^{-2} n)$.
2. Obtain $m \stackrel{\text{def}}{=} O(\epsilon^{-3}\log^3 n)$ samples S^{p} and S^{q} from p and q respectively.
3. Output FAIL if $\mathsf{occ}(I_j^\ell, S^{\mathsf{q}}) > (1+2\epsilon)\cdot\mathsf{occ}(I_j^r, S^{\mathsf{q}})$ or if $|\mathsf{occ}(I_j, S^{\mathsf{p}}) - \mathsf{occ}(I_j, S^{\mathsf{q}})| \geq \epsilon\cdot\mathsf{occ}(I_j, S^{\mathsf{p}})$ for any $j \in [k]$, or if $\mathsf{occ}(J, S^{\mathsf{q}}) > \epsilon^{-1}m\log\log n/\log n$.

First, we show a simple consequence of Lemma 4:

LEMMA 18. *Let* p, q *be monotone distributions on* $[n]$ *and* $I \subseteq [n]$ *be an interval such that* $\mathsf{p}(I^\ell) \leq (1+\epsilon)\cdot\mathsf{p}(I^r)$ *and* $\mathsf{q}(I^\ell) \leq (1+\epsilon')\cdot\mathsf{q}(I^r)$*. Then,* $\sum_{i\in I}|p_i - q_i| \leq \epsilon\mathsf{p}(I) + \epsilon'\mathsf{q}(I) + |\mathsf{p}(I) - \mathsf{q}(I)|$*.*

PROOF. Let $w_1 = \mathsf{p}(I)$ and $w_2 = \mathsf{q}(I)$. Then, by the triangle inequality,

$$\begin{aligned}\sum_{i\in I}|p_i - q_i| &= \sum_{i\in I}\left|p_i + \frac{w_1 - w_1 + w_2 - w_2}{|I|} - q_i\right| \\ &\leq \sum_{i\in I}\left|p_i - \frac{w_1}{|I|}\right| + \sum_{i\in I}\left|\frac{w_2}{|I|} - q_i\right| + \sum_{i\in I}\frac{|w_1 - w_2|}{|I|} \\ &\leq \epsilon w_1 + \epsilon' w_2 + |w_1 - w_2|. \quad \square\end{aligned}$$

We now obtain

THEOREM 19. *Given generation oracle access to monotone distributions* p *and* q *over* $[n]$*, the algorithm* **TestMonotoneCloseness** *outputs PASS when* $\mathsf{p} = \mathsf{q}$ *and outputs FAIL when* $|\mathsf{p} - \mathsf{q}| \geq 9\epsilon$*, with probability at least* $2/3$*. Moreover, the algorithm runs in time* $O(\epsilon^{-3}\log^3 n)$*.*

PROOF. Suppose $\mathsf{p} = \mathsf{q}$. By Theorem 17, for each I_j, $\mathsf{p}(I_j^\ell) \leq (1+\epsilon)\cdot\mathsf{p}(I_j^r)$ with probability $1 - o(1)$. Moreover, since $\mathsf{q}(J) = o(\epsilon^{-1}\log\log n/\log n)$, with probability $1 - o(1)$, S^{q} will contain less than $\epsilon^{-1}m\log\log n/\log n$ samples from J. Therefore, Step (3) is not likely to output FAIL.

Suppose the algorithm outputs PASS. Then, for each interval I_j we know that $\mathsf{q}(I_j^\ell) \leq (1+4\epsilon)\cdot\mathsf{q}(I_j^r)$, and moreover, $|\mathsf{p}(I_j) - \mathsf{q}(I_j)| \leq 3\epsilon\cdot\mathsf{p}(I_j)$. Now, using Lemma 18, and the facts that $\mathsf{p}(J) = o(1)$ and $\mathsf{q}(J) = o(1)$, and summing over all $I_1, \ldots I_k$, we can see that $|\mathsf{p} - \mathsf{q}| \leq 9\epsilon$. $\square$

6.2 Independence of monotone joint distributions

In this section we consider monotone distributions on $[n]^d$, and the independence of the random variables defined by each component of the samples from these distributions. Our goal is to distinguish monotone independent distributions from monotone distributions that are far from any independent distribution.

An easy but useful observation is that the marginal distributions of a monotone joint distribution are also monotone distributions. Based on this observation, we will use Theorem 17 to partition the domains of the marginal distribution into intervals. By Lemma 4, we know that the marginal distributions will be close to uniform on these intervals. Therefore, when the random variables defined by the joint distribution are independent, the conditional distributions on the "rectangles" formed by the cross product of the partitions will be close to uniform. Lemma 5 provides a means to check this condition.

For a rectangle, let the *midpoint* be the point that bisects the rectangle along each coordinate. Then we refer to the *top cube* (*bottom cube*) as the set of points in the rectangle that are smaller (larger) than the midpoint in each coordinate. Monotonicity ensures that each probability value in the top cube is greater than each of those in the bottom cube. The algorithm is:

Algorithm TestMonotoneIndependence

1. For each $i \in [d]$, apply `Partition` to the marginal distribution along the i-th dimension with $\epsilon_i = \epsilon/(32d)$ and $w = d^{-1}\log^{-2} n$ to obtain a partition of $[n]$ into $\mathcal{I}^{(i)} = \langle I_1^{(i)}, \ldots, I_{k_i}^{(i)}, J_i\rangle$.
2. For each d-dimensional rectangle $I_{i_1}^{(1)} \times I_{i_2}^{(2)} \times \cdots \times I_{i_d}^{(d)}$, output FAIL if the number of samples from the top cube is more than $(1+\epsilon/8)$ times that of the bottom cube.
3. Check that the distribution on the rectangles is $\epsilon/4$-close to the product of the marginal distributions on the rectangles.

THEOREM 20. *Given generation oracle access to monotone joint distribution* p *on* d*-tuples, the algorithm* **TestMonotoneIndependence** *outputs PASS if* p *induces* d *independent random variables and outputs FAIL if* p *has* L_1*-distance at least* ϵ *to any set of* d *independent variables, with probability at least* $2/3$*. Moreover, the algorithm uses* $O(\log^{(2d/3)+1} n)$ *samples and runs in time* $O(\log^d n)$*.*

PROOF. Suppose the joint distribution is independent. Then, for any d-dimensional rectangle that we check, the weight of the top cube is at most $(1+\epsilon/16)$ times that of the bottom cube, because in each marginal distribution, the top half of the interval has weight at most $(1+\epsilon/(32d))$ times that of the bottom half, and $(1+\epsilon/(32d))^d \leq (1+\epsilon/16)$.

Hence, after accounting for the sampling errors, all the rectangles in Step (2) will pass with high probability. The algorithm outputs PASS.

Now consider a distribution p that the algorithm outputs PASS. We know by Lemma 5 that the conditional distribution on each rectangle has L_1-distance at most $\epsilon/4$ to the uniform distribution. Let δ be the L_1-distance of p to the product of its marginal distributions. The total contribution of all the rectangles to δ will be at most $\epsilon/2$. Since, the total weight of the ignored parts of the domain, where at least one coordinate belongs to the corresponding J_i, is negligible, we can claim that $\delta \leq \epsilon$. Therefore, p has L_1-distance at most ϵ to a set of d independent variables on this domain.

The error probability is sum of the probabilities that Theorem 17 does not hold for any invocation of `Partition`. Therefore, the error probability is less than 1/3. The sample complexity of d invocations of the procedure `Partition` is $O(d^5\epsilon^{-3}\log^3 n)$. Step (3) can be accomplished by the algorithm to test if two distributions are close [3], which will entail $O(\log^{(2d/3)+1} n)$ samples. $\square$

7. UNIMODAL DISTRIBUTIONS

In this section we extend our results to unimodal distributions. We will only indicate the appropriate modifications/extensions needed for the unimodal case.

Testing unimodality. The outline of our algorithm for testing unimodality is be similar to our algorithm for testing monotonicity. After partitioning the domain $[n]$ into polylogarithmic number of intervals, each of which has close-to-uniform conditional distribution, the algorithm checks whether these intervals can be "patched" together to form a unimodal distribution. We will again use unimodal flat distributions as a tool. The analogs of Lemma 7 and Lemma 8 hold for the unimodal flat distributions. The only additional step in the proof of the latter is that since the maximum probability can occur in any one of the ℓ intervals, ℓ separate linear programs will be set up for each choice of the peak of the unimodal distribution. Thus, as before, we obtain an $\tilde{O}(\sqrt{n})$ algorithm for unimodality testing.

Testing closeness. The following is a unimodal analog of Lemma 4. It says that for a fine-enough partition, unimodality on balanced intervals implies close to uniformity.

LEMMA 21. *Let I be a interval, and let p be a unimodal distribution on $[n]$. Let $\ell = \lceil 1/\epsilon \rceil$, and $I_1, \ldots, I_\ell$ be a partition of I into equal-length subintervals. If, for all $j \in [\ell]$, $\frac{\mathsf{p}(I)}{(1+\epsilon)\ell} \leq \mathsf{p}(I_j) \leq \frac{(1+\epsilon)\mathsf{p}(I)}{\ell}$, then $\sum_{i \in I} \left| p_i - \frac{\mathsf{p}(I)}{|I|} \right| \leq \epsilon \mathsf{p}(I)$.*

We call an interval I to be $(1+\epsilon)$-*smooth* with respect to sample S if, for the ℓ-partition $\{I_1, \ldots, I_\ell\}$ of I where $\ell = \lceil 1/\epsilon \rceil$, $\frac{|S_I|}{(1+\epsilon)\ell} \leq |S_{I_j}| \leq \frac{(1+\epsilon)|S_I|}{\ell}$ for all j. The algorithm for testing closeness is similar to the monotone case, where we will use Theorem 17 to obtain a partition $\mathcal{I}_{k+1} = \langle I_1, \ldots, I_k, J \rangle$ of $[n]$, where each I_j is $(1+\epsilon)$-smooth.

8. THE CUMULATIVE ORACLE MODEL

It is instructive to compare the complexity of various tasks changes under different assumption on how the distributions are accessed. For example, suppose the only access to the distribution p is through a *cumulative evaluation oracle* P such that $P_i = \sum_{j=1}^{i} p_j$, and that the algorithm can access any P_i in one step. We show that in this model, monotonicity testing can be done in a simpler and more efficiently.

Note that from such an oracle, one can generate an element i with probability p_i in logarithmic time: generate a random $r \in [0,1]$ and output i such that $P_i \leq r$ by performing a binary search on P. We adapt the sorting spot-checker of [7] to obtain a sublinear algorithm for monotonicity in the cumulative oracle model.

THEOREM 22. *Given access to a cumulative oracle for distribution p over $[n]$, there is an algorithm that outputs PASS if p is monotone and outputs FAIL if p is not 2ϵ-monotone in L_1-norm, with probability at least $2/3$. The algorithm runs in time $O((1/\epsilon)(\log n + \log(1/\epsilon))\log n)$.*

Acknowledgments. The second author thanks Xin Guo and D. Sivakumar for many useful suggestions.

9. REFERENCES

[1] T. Batu, S. Dasgupta, R. Kumar, and R. Rubinfeld. The complexity of approximating the entropy. *Proc. 34th ACM Annual Symposium on Theory of Computing*, pages 678–687, 2002.

[2] T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld, and P. White. Testing random variables for independence and identity. *Proc. 42nd IEEE Annual Symposium on Foundations of Computer Science*, pages 442–451, 2001.

[3] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. *Proc. 41st IEEE Annual Symposium on Foundations of Computer Science*, pages 259–269, 2000.

[4] T. Batu, R. Rubinfeld, and P. White. Fast approximate PCPs for multidimensional bin-packing problems. Proc. 3rd International Workshop on Randomization and Approximation Techniques in Computer Science, pages 246–256, 1999.

[5] L. Devroye. Algorithms for generating discrete random variables with a given generating function or a given moment sequence. *SIAM J. on Scientific and Statistical Computing*, 12:107–126, 1991.

[6] Y. Dodis, O. Goldreich. E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. Proc. 3rd International Workshop on Randomization and Approximation Techniques in Computer Science, pages 97–108, 1999.

[7] F. Ergün, S. Kannan, R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *Journal of Computer and System Sciences,* 60(3):717–751, 2000.

[8] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. *Proc. 34th ACM Annual Symposium on Theory of Computing*, pages 474–483, 2002.

[9] O. Goldreich and D. Ron. On testing expansion in bounded degree graphs. *Electronic Colloquium on Computational Complexity*, TR00-020, 2000.

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

[11] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.