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 \(L_\infty\)-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 \(\log(n)\) 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


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.

Copyright 2004 ACM 1-58113-852-0/04/0006 ...$5.00.

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 Chernoby. 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 total order on the underlying domain and two elements \(x < y\) 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 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 $\Theta(\sqrt{n})$. 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 $[n] \times [t]$ runs in time $O(t^{n/2})$. In this case, we show a lower bound of $\Omega(t)$. 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. However, this is not the case, and 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 is not.

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 is 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.

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. 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 $O(n^{1/2})$ time. These ideas can be extended to higher dimensions with a sublinear running time of $O(n^{d/2})$; 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 distributions.
1. Related work

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 unimodality in the evaluation oracle model when an oracle access to the cumulative distribution is available to the algorithm. We obtain an $O(\log n)$ algorithm (Section 8).

2. PRELIMINARIES

We consider discrete probability distributions over $[n]$. Let $p = (p_1, \ldots, p_n)$ be such a distribution where $p_i \geq 0$, $\sum_i p_i = 1$. 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 $|p - q|$ to denote the $L_1$-distance and $|p - q|_\infty$ to denote the $L_\infty$-distance between two distributions. We call a distribution $p$ to be $e$-close in $L_1$-norm to a distribution $q$ if $|p - q| \leq e$. In particular, $p$ is $e$-close in $L_1$-norm if $|p - U_n| \leq e$ 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 $\max_i p_i \leq (1 + e) \min_i p_i$, then $|p|_1^2 \leq (1 + e^2)/n$. If $|p|_\infty \leq (1 + e^2)/n$, then $|p - U_n| \leq e$.

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 $e$-monotone in $L_1$-norm if there is a monotone distribution $q$ on $[n]$ such that $|p - q| \leq e$.

The notions of monotonicity and $e$-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 $p_i \geq \cdots \geq p_1 \geq p_{i+1} \geq \cdots \geq p_n$. A distribution $p$ on $[n]$ is said to be $e$-unimodal in $L_1$-norm if there is a unimodal distribution $q$ on $[n]$ such that $|p - q| \leq e$.

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 $i \leq j$, we (ab)use the interval notation $[i, j]$ to refer to the set $\{k \in \mathbb{Z} \mid i \leq k \leq j\}$. For a sample set $S$ and $i \in [n]$, occ$(i, S)$ denotes the number of times $i$ occurs in $S$; for $I \subseteq [n]$, occ$(I, S) \defeq \sum_{i \in I} \text{occ}(i, S)$. 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 $[1, n]$, $p(I)$ will denote $\sum_{i \in I} p_i$. For an interval $I = [i, i + k - 1]$, we use $I' = [i, i + k - 1]$ and $I'' = [i, i + k - 2]$ to denote its bisection. For a rectangle $R = I \times J \subseteq [n] \times [n]$ 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 as large 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 $p(I') \leq (1+\epsilon):p(I'')$, then $\sum_{i \in I} |p_i - \frac{w}{2k}| \leq \epsilon p(I)$.

Proof. We define $w \defeq p(I)$ and $\delta \defeq \left|p_i - \frac{w}{2k}\right|$. Let $j$ be the largest index in $I$ such that $p_j \geq \frac{w}{2k}$. First consider the case where $j \leq k$. Let $A_1 \defeq \sum_{i \in I, i \leq j} \delta_i$, $A_2 \defeq \sum_{i \in I, j \leq i} \delta_i$, $A_3 \defeq \sum_{i \in I, k \leq i \leq 2k} \delta_i$. We want to show that $A_1 + A_2 + A_3 \leq \epsilon w$. Note that $A_1 = A_3 = A_4$. By the assumption, we have

$\frac{A_1 - A_2 + \frac{w}{2}}{-A_3 + \frac{w}{2}} = \frac{p(I')}{p(I'')} \leq 1 + \epsilon$.

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

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 $p(I_1' \times \cdots \times I_d') \leq (1+\epsilon)p(I_1' \times \cdots \times I_d)$, then $p(I_1' \times \cdots \times I_d) \leq \epsilon w$.

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 $j \in I_1' \times \cdots \times I_d'$, $p_j \geq w/R$. The argument is analogous when for all $j \in I_1' \times \cdots \times I_d'$, $p_j \leq w/R$. (Note that if there exists a $j \in I_1' \times \cdots \times I_d'$ such that $p_j < w/R$, then monotonicity implies that $p_j \leq w/R$ for all $j \in I_1' \times \cdots \times I_d'$.)

For each $b \in \{\ell, r\}^d$, let $A_b \defeq \{i \in I_1' \times \cdots \times I_d' \mid p_i \geq w/R\}$, and let $W_b \defeq \sum_{i \in A_b} (p_i - w/R)$. Finally, let $t$ be the $d$-dimensional all-1's vector, i.e., $t \defeq (1, \ldots, 1)$. Note that for any $b \in \{\ell, r\}^d$, $W_b \leq W_t$ by the monotonicity of $f$. Since $p(I_1' \times \cdots \times I_d') = W_t + (w/R)^d$ and $p(I_1' \times \cdots \times I_d') \leq w/R$, by the assumption in the lemma statement, we know that

$W_t + \left(\frac{w}{R}\right)^d \leq (1 + \epsilon)$.

Hence, $W_t \leq \epsilon w/R^d$. So, we can conclude

$\sum_{j \in I_1' \times \cdots \times I_d} |p_j - \frac{w}{R^d}| = 2 \sum_{b \in \{\ell, r\}^d} W_b \leq 2\epsilon/R$. □

4. MONOTONICITY IN ONE DIMENSION

We consider the problem of distinguishing monotone decreasing distributions from those that are not $\epsilon$-monotone in $L_1$-norm. We give an $O(\sqrt{n})$ 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 $\epsilon$-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 conditional distributions 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 $\ell$ be an integer and let $I_\ell = \{I_1, \ldots, I_\ell\}$ be a partition of $[n]$. A distribution $q$ on $[n]$ is called an $\ell$-flat distribution if it can be described by the pair $(w, I_\ell)$, with $w = \langle w_1, \ldots, w_\ell \rangle$ and $q_i = w_i / |I_i|$ for $i \in I_\ell$.

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 $\epsilon$-close to monotone if and only if it is $\epsilon$-close to a monotone flat distribution.

Lemma 7. An $\ell$-flat distribution $p$ described by $(w, I_\ell)$ is $\epsilon$-monotone if and only if $p$ is $\epsilon$-close to a monotone flat distribution.

Proof. It is clear that if $p$ is $\epsilon$-close to a monotone flat distribution, then $p$ is $\epsilon$-monotone. For the converse, let
q be a monotone distribution such that \( |p - q| \leq \epsilon \). Let \( w = \langle w_1, \ldots, w_d \rangle \) such that \( p_i = w_i / |I_i| \) if \( i \in I_j \). Define the monotone \( \ell \)-flat distribution \( q' \) that is described by the pair \((w', I_j)\) where \( w' = \langle w_1', \ldots, w_d' \rangle \) and \( w_j' = q(I_j), j \in [\ell] \). We now show that \( q' \) is \( \epsilon \)-close to \( p \). By the flatness property and triangle inequality,

\[
|p - q'| = \sum_{j \in [\ell]} |w_j - w_j'| = \sum_{j \in [\ell]} |w_j - q(I_j)|
\]

\[
= \sum_{j \in [\ell]} \left( \sum_{i \in I_j} \left| w_i - q(I_j) \right| \right) \leq \sum_{j \in [\ell]} \sum_{i \in I_j} \left| w_i - q(I_j) \right|
\]

\[
= \sum_{i \in [n]} |w_i - q_i| = |p - q| \leq \epsilon.
\]

Secondly, \( \epsilon \)-monotonicity of \( \ell \)-flat distributions can be tested in time polynomial in \( \ell \).

**Lemma 8.** There is an algorithm that outputs \textsc{PASS} if an \( \ell \)-flat distribution described by \((w, I_\ell)\) is \( \epsilon \)-close to a monotone flat distribution, and outputs \textsc{FAIL} otherwise. The running time of this algorithm is \( \text{poly}(\ell) \).

**Proof.** For \( j \in [\ell] \), let \( k_j = |I_j| \) and let \( w = \langle w_1, \ldots, w_d \rangle \). First, consider the following mathematical program with variables \( y_j \) for \( i \in [\ell] \):

\[
\min \sum_{i \in [\ell]} y_i k_i - w_i \quad \text{s.t.} \quad \sum_{i \in [\ell]} y_i k_i = 1; \quad y_j \geq y_{j+1} \text{ for all } j \in [\ell - 1];
\]

\[
y_\ell \geq 0.
\]

The program above minimizes the distance between the given distribution and a monotone \( \ell \)-flat distribution. We now transform this into a linear program by introducing a new variable \( z_j \) to correspond to \( y_i k_i - w_i \) and by adding the constraints \( z_j \geq y_i k_i - w_i \) and \( -z_j \leq y_i k_i - w_i \), for all \( j \in [\ell] \). It is easy to see that this transformation preserves the value of the objective function. We have shown that a solution to this LP exists by solving LP in time \( \text{poly}(\ell) \), for example, using [11].

**The algorithm.** The collision count of \( S_i \), the samples in an interval \( I_i \), is defined to be \( \text{coll}(S_i) \equiv \sum_{j \in \text{occ}(S_i)} |I_j| \). 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:

\[
\left( \|q\|^2 - \frac{\epsilon^2}{32|I|^2} \right) \leq \frac{\text{coll}(S_i)}{|I|} \leq \left( \|q\|^2 + \frac{\epsilon^2}{32|I|^2} \right),
\]

with probability at least \( 1 - O(\log^{-2} n) \), provided that \( |S_i| = O(\epsilon^{-4} \sqrt{|I|} \log \log n) \).

Our algorithm will use collisions in the sample to determine a partition \( I_\ell \) 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 \( \epsilon \).

**Algorithm TestMonotonicity**

1. Obtain \( m \equiv O(\epsilon^{-4} \sqrt{n} \log n) \) samples \( S \) from \( p \).
2. Start with the interval \( I = [1, n] \), and bisect \( I \) in half recursively as long as:

\[
\text{coll}(S_i) \geq \left( \frac{(1 + \epsilon^2/32)}{|I|} \left| S_i \right| \right)
\]

\[
\text{and } |S_i| \geq \frac{m}{\log^2 n}.
\]

Abort the algorithm, and output FAIL if it performed more than \( O(\epsilon^{-1} \log^2 n) \) splits.

2. Let \( I_\ell = \{I_1, \ldots, I_{\ell} \} \) 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 \( O(\epsilon^{-2} \log^2 n) \).
4. Let \( \text{hist}(T, I_\ell) \) denote the \( \ell \)-flat distribution described by \((w, I_\ell)\) where \( w_i = \text{coll}(I_j) / |T| \).
5. Output PASS if \( \text{hist}(T, I_\ell) \) 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 \textsc{TestMonotonicity} outputs \textsc{PASS} when \( p \) is monotone and outputs \textsc{FAIL} when \( p \) is not \( \epsilon \)-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 \( \text{hist}(T, I_\ell) \) is a good approximation to \( p \), assuming Step 1 (succeeds), i.e., when \( \ell = O(\epsilon^{-1} \log^2 n) \). Consider the partition \( I_\ell \) obtained from Step 1. Call an interval \( I \) \textit{light} if \( \text{coll}(I, T) \leq m/\log^2 n \), 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 \( \sum_{i \in I} |p_i - q(I)|/|I| \leq \epsilon p(I)/4 \). The total weight of the light intervals is less than \( \epsilon/4 \) by the virtue of being light and the upper bound on \( \ell \). Hence, by summing over all intervals \( I \in I_\ell \), we get \( |\text{hist}(T, I_\ell) - p| \leq \epsilon/2 \).

Suppose \( p \) is a monotone distribution. We show that the algorithm will output \textsc{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 interval 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 \( O(\log n \log n) \) 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, \( I_\ell \) contains \( O(\epsilon^{-1} \log^2 n) \) intervals. Consequently, Step 1 will succeed and as we argued in the beginning of this proof, \( |\text{hist}(T, I_\ell) - p| \leq \epsilon/2 \). Since \( \text{hist}(T, I_\ell) \) is \( \ell \)-flat and is \( \epsilon/2 \)-close to a monotone distribution, by Lemma 7, it is also
\[ e/2 \text{-close to a monotone } \ell \text{-flat distribution and so Step (5) will also succeed and the algorithm will output PASS.} \]

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

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

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

**Theorem 11.** Let \( A \) be an algorithm that given generation oracle access to an arbitrary distribution \( p \) over \([n]\), has the following behavior: if \( p \) is monotone, then \( A \) outputs \( \text{PASS} \) and if \( p \) is not \( \varepsilon \)-monotone in \( L_1 \)-norm, then \( A \) outputs \( \text{FAIL} \). Furthermore the error probability of \( A \) is at most \( 1/3 \). Then \( A \) 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 \( \varepsilon \)-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 \( A \) to test if a given distribution is \( \varepsilon \)-monotone for \( \varepsilon < 1/16 \). We use this algorithm as a black-box to construct a new algorithm \( B \) that tests if a given distribution is \( (\varepsilon' + \varepsilon) \)-close to uniform, where \( \varepsilon' = 4\varepsilon/(1-2/\varepsilon) < 1 \). Define a new generation oracle \( p^R \) that on invocation, \( p \) and \( p^R \) outputs \( 2n - i \). The algorithm \( B \) uses both \( p \) and \( p^R \) in the following manner: it runs \( A \) on both \( p \) and \( p^R \) and outputs \( \text{PASS} \) if and only if \( A \) outputs \( \text{PASS} \) on both \( p \) and \( p^R \).

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

Let \( F = f([1, n]) = \sum_{i=1}^{n} f_i \), \( F' = f([n+1, 2n]) = \sum_{i=n+1}^{2n} f_i \) and \( G = g([1, n]) = \sum_{i=1}^{n} g_i \). Since \( p \) 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, \( |F - G| \leq |F - f| + |f - g| \leq 2\varepsilon \) and in particular, \( |F - G| \leq 2\varepsilon \). Combining all of these, \( 1/2 \leq G \leq 2\varepsilon + 2\varepsilon \) and consequently, \( F' \geq 1/2 - 2\varepsilon \). We also have

\[
0 \leq F - F' = 2F - 1 \leq \left( \frac{1}{2} + 2\varepsilon \right) - 1 = \frac{4\varepsilon}{1 - 2\varepsilon} F' = \varepsilon' F'.
\]

Also, since \( f \) is a distribution, \( \sum_{i=1}^{2n} f_i = 1 \). By appealing to Lemma 4, we see that \( \sum_{i=1}^{n} [F_i - 1/(2n)] \leq \varepsilon' \), that is, \( f \) is \( \varepsilon' \)-close to uniform. \(\Box\)

**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 \( n^2 \) equal quadrants and recursing on each of the quadrants. In other words, the algorithm builds a quadrant tree \( T \) 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 \( v_L, v_U, v_R, v_T \), 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 in 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 \( b, c \) be constants such that \( b > 2c + 1 \) and \( c > a + 2 \). Let \( \delta = \varepsilon' \log n \). Let \( S \) be a sample of size \( O(n^{3/2} \text{polylog}(\log n, \varepsilon^{-1})) \) 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 \( \text{occ}(K, S)/|S| \leq 1/\log^a n \), then \( v \) is a leaf.
2. If \( \text{col}(S_K) \leq (1 + \varepsilon'^2/32) (|S|^2)/|K| \), then \( v \) is a leaf.
3. If the quadrant \( K \) is more than \( \log^b n \) away from the origin, then \( v \) is a leaf. Update \( W \leftarrow W + |\text{occ}(K, S)| \).
4. If $K$ is not already designated as a leaf, then do the following two steps for each of the following ordered pairs: $(K^{(i)}, K^{(r)})$, $(K^{(i)}, K^{(r)}, K^{(r)}, r)$, $(K^{(r)}, K^{(r)}, K^{(r)}), (K^{(r)}, K^{(r)}, r)$. We will illustrate the steps for $(K^{(i)}, K^{(r)})$.

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

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

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


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 $\log^{c} n$ 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 $O(1/|n|\log^{3} n))$. This entails $\tilde{O}(n^{3/2/\epsilon^{3}})$ 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, $(K^{(i)}, K^{(r)})$ 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 $\epsilon$-monotone. Furthermore, let $p(I') \leq (1 + \sigma)p(I)$. Then $p$ is $(4\epsilon + 2\sigma)$-close to uniform.

**Proof.** Let $f$ be a monotone distribution such that $|p - f| \leq \epsilon$. Since $p(I') - p(I') \leq 2\epsilon/3$ and $|p - f| \leq \epsilon$, $f(I') = p(I') + 2\epsilon/3$. Thus, we get

$$\frac{f(I')}{f(I')} \leq 1 + \frac{2\epsilon/3}{f(I')} \leq 1 + \frac{2\epsilon/3}{1 - \frac{2\epsilon}{3}} \leq 1 + 3\epsilon + 2\sigma.$$

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 $p(I \times J) \leq (1 + \delta)p(I \times J')$. Then, for any $\rho > 0$, $\Pr_{i \in I}[p_{i}(I \times J') \geq (1 + \rho)]: p_{i}(I \times J)] \leq 1/\rho$, where $i$ is chosen with probability $p_{i}(I \times J)/p(I \times J')$.

**Proof.** Let $W = p(I \times J')$, $W' = p(I \times J)$. We know that $W' \leq W \leq (1 + \delta)W$. Let $w_{i} = p_{i}(I \times J)$ and $w'_{i} = p_{i}(I \times J')$. 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 $\delta$ 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 $\delta = \delta'$, 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 $O(\epsilon^{-1}\log^{2c+1} n)$.
tions the domain into a small number of intervals that are balanced. The next theorem from [1] comes to our rescue.

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

Notice that we could not have used Theorem 17 for testing monotonicity: \textbf{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 $\mathcal{I}_{d+1} = \{I_1, \ldots, I_k, J\}$ of $[n]$. We then check if $p$ and $q$ are close in each of the intervals $I_j$ and if $q(J)$ is small. Here is a description of the algorithm.

\textbf{Algorithm TestMonotoneCloseness}:

1. Let $\{I_1, \ldots, I_k, J\} = \text{Partition}(p, \epsilon, \log^{-2} n)$.
2. Obtain $m = O(e^{-3} \log^3 n)$ samples $S^p$ and $S^q$ from $p$ and $q$, respectively.
3. Output FAIL if oc$(I_j', S^p) > (1 + 2\epsilon) \cdot \text{oc}(I_j', S^q)$ or if $\lvert \text{oc}(I_j, S^p) - \text{oc}(I_j, S^q) \rvert \geq \epsilon \cdot \text{oc}(I_j, S^q)$ for any $j \in [k]$, or if $\text{oc}(I_j, S^q) > e^{-1} m \log n / \log n$.

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 $p(I') \leq (1 + \epsilon) \cdot p(I)$ and $q(I') \leq (1 + \epsilon') \cdot q(I)$. Then, $\sum_{i \in I} |p_i - q_i| \leq \epsilon \cdot (p(I) + \epsilon' q(I) + |p(I) - q(I)|).

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

$$\sum_{i \in I} |p_i - q_i| \leq \sum_{i \in I} \frac{|p_i - w_1 + w_1 - w_2 + w_2 - q_i|}{|I|} \leq \sum_{i \in I} |p_i - w_1|/|I| + \sum_{i \in I} |w_2 - q_i| + \sum_{i \in I} |w_1 - w_2|/|I| \leq \epsilon w_1 + \epsilon' w_2 + |w_1 - w_2|.$$

We now obtain

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

\textbf{Proof.} Suppose $p = q$. By Theorem 17, for each $I_j$, $p(I_j) \leq (1 + \epsilon) \cdot p(I_j')$ with probability at least $1 - \alpha(1)$. Moreover, since $q(J) = o(e^{-1} \log n / \log n)$, with probability $1 - \alpha(1)$, $S^p$ contains less than $\epsilon^{-1} m \log n / \log n$ samples from $J$. Therefore, Step (3) is not likely to output \textit{FAIL}.

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

### 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 independent monotone distributions from independent 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:

\textbf{Algorithm TestMonotoneIndependence}:

1. For each $i \in [d]$, apply \textbf{Partition} to the marginal distribution along the $i$-th dimension with $\epsilon_i = \epsilon/(32d)$ and $w = \epsilon^{-1} \log^{-2} n$ to obtain a partition of $[n]$ into $I_i^{(d)} = \{I_1^{(d)}, \ldots, I_k^{(d)}, J\}$.
2. For each $d$-dimensional rectangle $I_1^{(1)} \times I_2^{(1)} \times \cdots \times I_d^{(1)}$, output \textit{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 \textbf{TestMonotoneIndependence} outputs \textit{PASS} if $p$ induces $d$ independent random variables and outputs \textit{FAIL} if $p$ has $L_1$-distance at least $\epsilon$ to any set of $d$ independent variables, with probability at least $2/3$. Moreover, the algorithm uses $O((\log^{(d+1)/2} n)^2)$ samples and runs in time $O(\log^2 n)$.

\textbf{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 no larger than $(1 + \epsilon/(32d))$ times that of the bottom half. Thus, the algorithm outputs \textit{PASS}. If for any $d$-dimensional rectangle that we check, the weight of the top cube is at least $(1 + \epsilon/8)$ that of the bottom cube, the algorithm aborts and outputs \textit{FAIL}.

For rectangles that are far from independent, we use Theorem 17 to partition the domains of the marginal distributions 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.
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 $\delta$ be the $L_1$-distance of $p$ to the product of its marginal distributions. The total contribution of all the rectangles to $\delta$ 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 $\epsilon$ 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 $O(d^2 \epsilon^{-4} \log^2 n)$. Step (3) can be accomplished by the algorithm to test if two distributions are close [3], which will entail $O((\log n)^{d+1})$ samples. \qed

7. UNIMODAL DISTRIBUTIONS

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

Testing unimodality: The outline of our algorithm for testing unimodality is similar to our algorithm for testing monotonicity. After partitioning the domain $[n]$ into polynomially many 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 $\ell$ intervals, $\ell$ separate linear programs will be set up for each choice of the peak of the unimodal distribution. Thus, as before, we obtain an $O(\sqrt{n})$ 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 $\ell = [1/\epsilon]$, and $I_1, \ldots, I_\ell$ be a partition of $I$ into equal-length subintervals. If, for all $j \in [\ell]$, \[ \frac{p(I_j)}{(1+\epsilon)p(I_j)} \leq p(I_j) \leq \frac{p(I_j)}{1-\epsilon} \] then \[ \sum_{i \in I} |p_i - \frac{p(I)}{|I|}| \leq \epsilon p(I). \]

We call an interval $I$ to be $(1+\epsilon)$-smooth with respect to sample $S$ if, for the $\ell$-partition $\{I_1, \ldots, I_\ell\}$ of $I$ where $\ell = [1/\epsilon]$, \[ \frac{p(I_1)}{(1+\epsilon)p(I)} \leq |S_{I_1}| \leq \frac{p(I_1)}{1-\epsilon} \] for all $j$. The algorithm for testing closeness is similar to the monotone case, where we will use Theorem 17 to obtain a partition $I_{k+1} = \{I_1, \ldots, I_{k}, J\}$ 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 2-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


