Testing monotone high-dimensional distributions

Ronitt Rubinfeld
Computer Science and Artificial Intelligence Laboratory
MIT
Cambridge, MA
ronitt@theory.lcs.mit.edu

Rocco Servedio*
Dept. of Computer Science
Columbia University
New York, NY
rocco@cs.columbia.edu

February 16, 2005

Abstract

A *monotone distribution* P over a (partially) ordered domain assigns higher probability to y than to x if $y \geq x$ in the order. We study several natural problems concerning testing properties of monotone distributions over the n-dimensional Boolean cube, given access to random draws from the distribution being tested. We give a poly(n)-time algorithm for testing whether a monotone distribution is equivalent to or ϵ-far (in the L_1 norm) from the uniform distribution. A key ingredient of the algorithm is a generalization of a known isoperimetric inequality for the Boolean cube. We also introduce a method for proving lower bounds on various problems of testing monotone distributions over the n-dimensional Boolean cube, based on a new decomposition technique for monotone distributions. We use this method to show that our uniformity testing algorithm is optimal up to polylog(n) factors, and also to give exponential lower bounds on the complexity of several other problems, including testing whether a monotone distribution is identical to or ϵ-far from a fixed known monotone product distribution and approximating the entropy of an unknown monotone distribution.

1 Introduction

We study the complexity of testing several natural global properties of monotone probability distributions over large high-dimensional discrete domains. When no assumptions are made on the distributions, classical techniques, such as a naive use of Chernoff bounds or Chi-squared tests, require a number of samples that is at least linear in the size of the domain. In recent years, a number of algorithms for these testing problems have been designed which require a number of samples that is only sublinear in the size of the domain, while making no assumptions on the form of the distribution. For example, on arbitrary domains of size N, testing whether a distribution is close to uniform in statistical distance can be performed with only $\tilde{O}(\sqrt{N})$ samples [7, 2], and distinguishing whether two distributions are the same or far in statistical distance can be performed with $\tilde{O}(N^{2/3})$ samples [4]. Similar results have been obtained for testing whether a joint distribution is independent and estimating the entropy [2, 3]. Still, in many settings, where N may be extremely large, such sample complexities can be quite daunting. Unfortunately, one cannot do much better for general distributions, since known information theoretic lower bounds show that a sample complexity with a polynomial dependence on N is required for all of these problems.

*Supported in part by NSF CAREER award CCF-0347282. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

This leads us naturally to the question of whether there are interesting classes of distributions for which these testing problems are exponentially easier in terms of sample complexity. Recently a number of algorithms that use exponentially fewer samples on monotone and unimodal distributions over totally ordered domains have been devised [3, 5]. A distribution over a totally ordered discrete domain (w.l.o.g. $\{1, 2, \ldots, N\}$) is *monotone* if for all x, y in the domain such that $x \leq y$, we have that the probability assigned to x is at most the probability assigned to y. For monotone distributions on totally ordered domains, estimating the entropy, testing whether two distributions are close and testing whether a joint distribution over pairs (x, y) is close to independent can all be done in polylogarithmic time in the size of the domain [3, 5].

Our results. In this paper we give a detailed study of testing distributions that are monotone over a natural and important *partially* ordered domain, namely the Boolean cube $\{-1, 1\}^n$. Such distributions can be viewed as distributions in which the probability of an element depends monotonically on several different features of the element.

We investigate whether the monotonicity of distributions over such domains allows us to test various properties of these distributions with query complexity *polylogarithmic* in the domain size (i.e. polynomial in the dimension n), as it does in the case of totally ordered domains. We will refer to testing problems for which the answer is affirmative as *easy*, and we refer to all other testing problems as *hard*.

Our first result shows that there is an interesting property which is easy, that is, testing uniformity. More precisely, we show that it is possible to efficiently distinguish distributions that are uniform over the Boolean cube $\{-1, 1\}^n$ from monotone distributions that are far from uniform in statistical distance. Our algorithm uses $\tilde{O}(n)$ many draws from the distribution; we also give a lower bound which shows that this algorithm is essentially the best possible.

Somewhat surprisingly, we next exhibit closely related testing problems which are hard. For example, we show that any algorithm which distinguishes whether an unknown monotone distribution is equal to or very far from the product distribution $\mathcal{P}_{4/5}$ over $\{-1, 1\}^n$ must use $2^{\Omega(n)}$ samples.[1] (It is interesting to note that this exponential gap between the problems of testing whether a distribution is uniform or a product distribution is quite different from the behavior commonly found in other combinatorial and learning settings where the Boolean cube is studied. For example, in learning theory, most problems that have efficient learning algorithms under the uniform distribution also have efficient algorithms under product distributions, see e.g. [10, 6, 11].) We also give lower bounds with exponential dependence on n for other problems, such as estimating the entropy of a monotone distribution over $\{-1, 1\}^n$, and distinguishing whether a monotone distribution over $\{-1, 1\}^n$ is independent (i.e. a product distribution) or far from any independent distribution. All of these lower bounds are in contrast with the known results for monotone distributions over totally ordered domains [3, 5], where there exist algorithms that require only a polylogarithmic in the total domain size number of samples.

Finally, we study many of these testing problems in the *evaluation oracle* model, where one can query the probability that distribution p assigns to any domain element x.

Our techniques. In [3, 5], an efficient test is given for determining whether a monotone distribution over a totally ordered domain is uniform. The test estimates whether the weight of the largest half of the elements (according to the total order on the domain) is approximately $1/2$. In order to achieve our upper bound for testing whether a monotone distribution p on the cube is uniform, our test is somewhat different. Our test essentially estimates the expected value, according to a choice of $x = (x_1, \ldots, x_n) \in_p \{-1, 1\}^n$, of the sum of the x_i's, and rejects if the estimated value is too big.

[1] $\mathcal{P}_{4/5}$ independently picks every entry x_i of $x \equiv (x_1, \ldots, x_n) \in \{-1, 1\}^n$ to be 1 with probability $4/5$ and -1 otherwise.

In order to show correctness, we generalize a known isoperimetric inequality for the Boolean cube. In particular, we show that any monotone distribution which is far from uniform must have many pairs of neighboring nodes whose probabilities differ significantly. We use this to show that for any monotone distribution p that is ϵ-far from uniform in statistical distance, the expected value of the sum of the x_i's when x is chosen from p must be at least $\epsilon/2$.

For our negative results, we present a general technique for proving lower bounds on testing problems for monotone distributions, and apply this technique to construct a number of lower bounds. Our technique is based on describing a class of special monotone distributions, namely those with *subcube decompositions*. A monotone subcube distribution rooted at x is a distribution that is uniform over the set of all points y such that $y \geq x$. A monotone distribution has a subcube decomposition if can be expressed as a weighted average of a number of monotone subcube distributions. Not every monotone distribution has a subcube decomposition. Roughly speaking, our main claim essentially shows that it is hard to distinguish between a monotone distribution p which has a subcube decomposition, and a distribution q that is a weighted average of randomly chosen monotone subcube distributions from the decomposition of p. However, if q is a weighted average of a not too large subset of the distributions that make up p, it is easy to see that p and q are very different distributions. All our lower bounds use this approach with a suitable choice of p and q.

We note that the techniques used to achieve upper bound results for the problem of testing various properties of distributions over totally ordered domains [3, 5] essentially relied on showing that every monotone distribution p over a totally ordered domain could be approximated by a distribution q with a very concise description of the following form: the domain is partitioned into a polylogarithmic (in the size of the domain) number of contiguous segments, such that the distribution q is uniform over each segment. The testing algorithms efficiently find an approximation to this segmentation. However, in contrast with this situation for totally ordered domains, our strong negative results suggest that no such analogous decomposition of monotone distributions over the cube may exist.

Other related results. In [5], algorithms are given for testing whether a distribution over a totally ordered domain is close to monotone using $\tilde{\Theta}(\sqrt{N})$ draws. The proofs of [5] show that the complexity of testing monotonicity is linked to the complexity of testing uniformity. The algorithm was extended to domains of dimension 2 (i.e., each element is a member of $M \times M$ for $M^2 = N$), using $\tilde{O}(N^{3/4})$ queries.

Outline of paper. In the next section, we describe the necessary preliminaries. Our testing algorithm for uniformity is described in Section 3. In Section 4 we describe a general technique for obtaining lower bounds, and then use the technique in order to obtain lower bounds for several problems. Finally, in Section 5 we consider the query complexity of these problems in the evaluation oracle model.

2 Preliminaries

Throughout the paper we use $\{-1,1\}^n$ as our representation for the n-dimensional Boolean cube. For $x \in \{-1,1\}^n$ we write $\mathrm{bias}(x)$ to denote $\sum_{i=1}^n x_i$, and we write $\mathrm{ones}(x)$ to denote the number of indices i such that $x_i = 1$. (Of course we have $\mathrm{ones}(x) = \frac{\mathrm{bias}(x)+n}{2}$, but it is convenient to have both notations.) For $x, y \in \{-1,1\}^n$ we write $y \geq x$ if $y_i \geq x_i$ for all $i = 1, \ldots, n$. We refer to the $(n - \mathrm{ones}(x))$-dimensional subcube $\{y \in \{-1,1\}^n : y \geq x\}$ as the *monotone subcube rooted at* x, and we write $\mathcal{U}_x$ to denote the uniform distribution over this subcube, i.e. $\mathcal{U}_x(y) = 1/2^{n-\mathrm{ones}(x)}$ if $y \geq x$ and $\mathcal{U}_x(y) = 0$ if $y \not\geq x$. We refer to such a distribution $\mathcal{U}_x$ as a *uniform monotone subcube* *distribution.* A probability distribution p over $\{-1,1\}^n$ is *monotone* if $y \geq x$ implies $p(y) \geq p(x)$. It is clear that any convex combination of uniform monotone subcube distributions is a monotone distribution. We write $\mathcal{U}$ to denote the uniform distribution over all 2^n points of $\{-1,1\}^n$.

If p, q are two probability distributions over $\{-1,1\}^n$ we write $\|p-q\|_1$ to denote the L_1 distance $\sum_x |p(x) - q(x)|$ between p and q. We write $S \leftarrow (p)^t$ to indicate that S is the random variable which is the output of t independent draws from p (so S is a t-tuple of strings from $\{-1,1\}^n$).

Given a vector $\overline{\tau} = (\tau_1, \ldots, \tau_n)$, the *product distribution with parameter* $\overline{\tau}$, which we denote $\mathcal{P}_{\overline{\tau}}$, is the distribution over $\{-1,1\}^n$ where the i-th bit is chosen independently to be 1 with probability τ_i. For $\tau \in [0,1]$ we write $\mathcal{P}_\tau$ to denote the product distribution with $\tau_i = \tau$ for all $i = 1, \ldots, n$. Each string x has weight $\tau^{\text{ones}(x)}(1-\tau)^{n-\text{ones}(x)}$ under $\mathcal{P}_\tau$. Note that the distribution $\mathcal{P}_\tau$ is monotone iff $\frac{1}{2} \leq \tau \leq 1$, and that $\mathcal{P}_{1/2}$ is equivalent to $\mathcal{U}$.

A *generation oracle*, or simply *generator*, for a probability distribution $\mathcal{D}$ over $\{-1,1\}^n$ is an oracle which takes no input and, at each invocation, returns a random element x from $\{-1,1\}^n$ drawn according to $\mathcal{D}$ independently from all previous invocations of the oracle. An *evaluation oracle*, or simply *evaluator*, for a distribution $\mathcal{D}$ over $\{-1,1\}^n$ is an oracle which, when supplied with $x \in \{-1,1\}^n$, returns the probability weight $\mathcal{D}(x)$ which $\mathcal{D}$ assigns to x.

We will use the following version of the "data processing inequality:"

Fact 1 *Let X_1, X_2 be two random variables over the same domain. For any (possibly randomized) algorithm A, we have that $\|A(X_1) - A(X_2)\|_1 \leq \|X_1 - X_2\|_1$.*

We will typically use this fact in the following way: let S_1 and S_2 be random variables which denote samples of t draws taken from two distributions p_1 and p_2, and let A be an algorithm which, given a draw from S_i (where $i \in \{1,2\}$ is unknown to A), is supposed to output the correct value of i with high probability. If we know that $\|S_1 - S_2\|_1$ is small, then $|\Pr_{S_1 \leftarrow (p_1)^t}[A(S_1)$ outputs "1"$] - \Pr_{S_2 \leftarrow (p_2)^t}[A(S_2)$ outputs "1"$]|$ must be small as well by Fact 1, and thus A cannot succeed.

Finally, we will frequently use standard Chernoff bounds;

Fact 2 *[Additive Chernoff Bound] Let $X_1, \ldots, X_m$ be i.i.d. random variables which take values in the range $[-a, a]$. Let $\overline{\mu}$ denote $\frac{1}{m}\sum_{i=1}^m X_i$ and let μ denote $\mathbf{E}[\overline{\mu}]$. Then for all $\gamma > 0$ we have $\Pr[|\overline{\mu} - \mu| > \gamma] \leq 2\exp\left(-\frac{\gamma^2}{2a^2}m\right)$.*

Fact 3 *[Multiplicative Chernoff Bound] Let $X_1, \ldots, X_m$ be i.i.d. 0/1-valued random variables with $\mathbf{E}[X_i] = p$. Let X denote $\sum_{i=1}^m X_i$. Then for all $0 < \gamma < 1$ we have $\Pr[X \leq (1-\gamma)mp] \leq 2\exp\left(-\frac{\gamma^2\mu}{2}\right)$ and $\Pr[X \geq (1+\gamma)mp] \leq 2\exp\left(-\frac{\gamma^2\mu}{3}\right)$.*

3 A uniformity testing algorithm

In this section we give an efficient algorithm that can distinguish the uniform distribution over $\{-1,1\}^n$ from any monotone distribution which is far from uniform:

Theorem 4 *There is an efficient algorithm* `TestUniform` *(see Figure 1) which, given generator access to an unknown monotone distribution p over $\{-1,1\}^n$, makes $O(\frac{n}{\epsilon^2}\log\frac{n}{\epsilon})$ draws and satisfies the following properties: (i) If $p \equiv \mathcal{U}$ then* `TestUniform` *outputs "uniform" with probability at least $\frac{4}{5}$; (ii) If $\|p - \mathcal{U}\|_1 \geq \epsilon$ then* `TestUniform` *outputs "nonuniform" with probability at least $\frac{4}{5}$.*

The algorithm's analysis uses the following technical result, which can be viewed as a generalization of the well-known fact that any balanced partition of the hypercube $\{-1,1\}^n$ into red and blue nodes must have $\Omega(2^n)$ bichromatic edges.

Algorithm `TestUniform`

1. Draw a sample $S = x^1, \ldots, x^s$ of $s = \Theta(\frac{n}{\epsilon^2} \log \frac{n}{\epsilon})$ draws from the generator for p.
2. If any x^i in the sample has $|\text{bias}(x^i)| > \sqrt{2n \log(20s)}$, stop and output "nonuniform."
3. Let $\overline{\mu} = \frac{1}{s}\sum_{i=1}^{s} \text{bias}(x^i)$, the empirical estimate of $\mathbf{E}_p[\text{bias}(x)]$ obtained from S.
4. Output "uniform" if $\overline{\mu} \leq \frac{\epsilon}{4}$, and output "nonuniform" if $\overline{\mu} > \frac{\epsilon}{4}$.

Figure 1: Algorithm for testing whether an unknown monotone distribution over $\{-1,1\}^n$ is uniform or ϵ-far from uniform.

Theorem 5 *Let $\delta : \{-1,1\}^n \to \mathbf{R}$ be a monotone real-valued function with $\sum_x \delta(x) = 0$ and $\sum_x |\delta(x)| = \epsilon 2^n$. Then $\frac{1}{2^n}\sum_x \sum_{i=1}^n x_i \delta(x) \geq \epsilon$.*

Theorem 5 has the following easy corollary:

Corollary 6 *If p is a monotone distribution over $\{-1,1\}^n$ with $\|p - \mathcal{U}\|_1 \geq \epsilon$, then $\mathbf{E}_p[bias(x)] \geq \epsilon$.*

Proof: Define $\delta(x) = 2^n p(x) - 1$. Since $\|p - \mathcal{U}\|_1 = \sum_x |p(x) - \frac{1}{2^n}|$, it is easy to check that this function $\delta(x)$ satisfies the condition of Theorem 5. We thus have

$$\begin{aligned}
\mathbf{E}_p[\text{bias}(x)] &= \sum_x p(x) \sum_{i=1}^n x_i = \sum_x \sum_{i=1}^n \left(\frac{\delta(x)+1}{2^n}\right) x_i \\
&= \sum_x \sum_{i=1}^n \frac{\delta(x)}{2^n} x_i \geq \epsilon
\end{aligned}$$

where the inequality is by Theorem 5. ■

Proof of Theorem 5: Let $POS = \{x \in \{-1,1\}^n : \delta(x) \geq 0\}$, i.e. the positive inputs for δ, and let $NEG = \{-1,1\}^n \setminus POS$. Note that $\sum_{x \in POS} \delta(x) = \sum_{y \in NEG} |\delta(y)| = \frac{\epsilon}{2} 2^n$.

Given $x \in POS, y \in NEG$ we refer to $\Delta(x,y) := |\delta(x) - \delta(y)| = |\delta(x)| + |\delta(y)|$ as the *displacement* between x and y. If $z = (z_0 = x, z_1, \ldots, z_k = y)$ is a path between x and y along the edges of the cube, we refer to $dist_z(x,y) := \sum_{i=0}^{k-1} |\delta(z_i) - \delta(z_{i+1})|$ as the *z-distance* between x and y. For any path z between x and y we have $dist_z(x,y) \geq \Delta(x,y)$.

The total displacement between all pairs (x,y) with $x \in POS, y \in NEG$ is

$$\begin{aligned}
\sum_{x,y} \Delta(x,y) &= \sum_{x,y} |\delta(x)| + |\delta(y)| = \sum_{x,y} |\delta(x)| + \sum_{x,y} |\delta(y)| \\
&= \frac{|NEG|\epsilon}{2} \cdot 2^n + \frac{|POS|\epsilon}{2} \cdot 2^n = \frac{\epsilon}{2} \cdot 2^{2n}
\end{aligned}$$

Now for each pair $(x,y) \in POS \times NEG$, let $z_{x,y}$ be the canonical path from x to y, i.e. $z_{x,y}$ is the path starting from x where we scan through the bits from left to right flipping bits as required. Let e be any "downward-directed" edge of the Boolean cube, i.e. $e = (a1b, a(-1)b)$ where a, b are strings of $+1/-1$'s whose total length is $n-1$. The edge e occurs on at most 2^{n-1} canonical paths z_{v_1,v_2} (since if e is on the canonical path from v_1 to v_2 it must be the case that the prefix of v_2 is $a1$ and the suffix of v_1 is $(-1)b$).

Now let $DIST$ denote $\sum_{x\in POS, y\in NEG} dist_z(x,y)$ where each z in the above sum is the canonical path. Since each downward-directed edge $e=(u,v)$ lies on at most 2^{n-1} canonical paths, each such edge contributes at most $2^{n-1}|\delta(u)-\delta(v)|$ to $DIST$. Summing over all edges (u,v) in the downward-directed edge set E of $\{-1,1\}^n$, we have that

$$\sum_{(u,v)\in E} 2^{n-1}(|\delta(u)-\delta(v)|) \geq DIST \geq \sum_{x\in POS, y\in NEG} \Delta(x,y)$$

and thus we have $\sum_{(u,v)\in E} |\delta(u)-\delta(v)| \geq \epsilon \cdot 2^n$.

Let E_i denote the set of all downward-directed edges in the i-th direction of the cube, i.e. E_i is the set of all 2^{n-1} ordered pairs (u,v) where $u,v\in\{-1,1\}^n$ differ only in that $u_i=1$ while $v_i=-1$. Then $\sum_{(u,v)\in E} |\delta(u)-\delta(v)|$ equals

$$\begin{aligned}\sum_{i=1}^{n}\sum_{(u,v)\in E_i} |\delta(u)-\delta(v)| &= \sum_{i=1}^{n}\sum_{(u,v)\in E_i} (\delta(u)-\delta(v)) \\ &= \sum_{i=1}^{n}\sum_{x\in\{-1,1\}^n} x_i\delta(x)\end{aligned}$$

where the first equality follows from the monotonicity of δ. This proves the theorem. ■

We now give the proof of Theorem 4. The basic idea is to estimate bias(x) and use Corollary 6; however we must be careful to bound the variance in order for the algorithm to succeed with a small number of draws.

Proof:[of Theorem 4]

part (i): Since $p\equiv\mathcal{U}$, the true expected value $\mathbf{E}_p[\text{bias}(x)]$ is 0. By an additive Chernoff bound, for each fixed i the probability that $|\text{bias}(x^i)| > \gamma n$ is at most $2\exp(-\gamma^2 n/2)$. Taking $\gamma = \sqrt{2\log(20s)/n}$, this probability is at most $\frac{1}{10s}$, so a union bound over $i=1,\dots,s$ implies that we output "nonuniform" in step 2 with probability at most $\frac{1}{10}$.

If we reach step 3, then each value bias(x^i) is drawn from the binomial distribution (sum of n independent uniform ±1 values) conditioned on having each draw satisfy $|\text{bias}(x^i)| \leq \sqrt{2n\log(20s)}$. This conditional random variable has range $[-\sqrt{2n\log(20s)}, \sqrt{2n\log(20s)}]$ and has zero mean (by symmetry), so we may apply the additive Chernoff bound (Fact 2) with $a=\sqrt{2n\log(20s)}$, $\gamma=\frac{\epsilon}{4}$ and $m=s$, and we get that $\Pr[\overline{\mu} > \frac{\epsilon}{4}]$ is at most $2\exp(-\frac{\epsilon^2 s}{64n\log(20s)})$. It is easily verified that taking $s=\Theta(\frac{n}{\epsilon^2}\log\frac{n}{\epsilon})$ makes this bound at most $\frac{1}{10}$; so the overall probability that we do not output "uniform" is at most $\frac{1}{5}$.

part (ii): We consider two cases.

Case 1: $\Pr_p[|\text{bias}(x)| > \sqrt{2n\log(20s)}] \geq \frac{10}{s}$, i.e. p assigns probability at least $\frac{10}{s}$ to "unbalanced" strings. In this case, the probability that we do not output "nonuniform" in step 2 is at most $(1-\frac{10}{s})^s < \frac{1}{100}$.

Case 2: $\Pr_p[|\text{bias}(x)| > \sqrt{2n\log(20s)}] < \frac{10}{s}$, i.e. p assigns probability less than $\frac{10}{s}$ to unbalanced strings. Since our goal is to show that the algorithm outputs "uniform" with probability at most $\frac{1}{5}$, we may assume that the algorithm reaches step 3, i.e. that each sample value bias(x^i) which is obtained is conditioned on satisfying $|\text{bias}(x^i)| \leq \sqrt{2n\log(20s)}$. Since $\Pr_p[|\text{bias}(x)| > \sqrt{2n\log(20s)}] < \frac{10}{s}$ in this case and bias(x) always lies in $[-n,n]$, we have that the conditional expectation $\mathbf{E}_p[\text{bias}(x) \mid |\text{bias}(x)| \leq \sqrt{2n\log(20s)}]$ differs from the unconditional expectation

$\mathbf{E}_p[\text{bias}(x)]$ by at most $2n \cdot \frac{10}{s} = \Theta(\frac{\epsilon^2}{\log \frac{n}{\epsilon}})$ which is less than $\frac{\epsilon}{8}$ for n sufficiently large. Corollary 6 thus implies that the true value of the conditional expectation $\mathbf{E}_p[\text{bias}(x) \mid |\text{bias}(x)| \leq \sqrt{2n\log(20s)}]$ is at least $\frac{3\epsilon}{8}$. Thus, as in the proof of part (i) we may apply Fact 2 (now with $a = \sqrt{2n\log(20s)}$, $\gamma = \frac{\epsilon}{8}$, and $m = s$) and we get that $\Pr[\overline{\mu} < \frac{\epsilon}{4}]$ is at most $2\exp(-\frac{\epsilon^2 s}{256n\log(20s)})$. Taking $s = \Theta(\frac{n}{\epsilon^2}\log\frac{n}{\epsilon})$ makes this bound at most $\frac{1}{5}$, and we are done. ■

4 Lower bounds

In Section 4.1 we introduce a new general lower bound technique for testing monotone distributions given access to a generator. We then apply this technique on several different problems. Section 4.2 gives an $\Omega(n/\log^2(n))$ lower bound on testing whether a monotone distribution is uniform or far from uniform. This bound is optimal up to polylog(n) factors by the positive results of Section 3. Section 4.3 gives a $2^{\Omega(n)}$ lower bound on testing whether a monotone distribution is a particular known product distribution or is far from this distribution. Section 4.4 gives lower bounds on approximating the entropy of an unknown monotone distribution.

4.1 The lower bound technique Let p be a monotone distribution over $\{-1,1\}^n$. We say that a *monotone subcube decomposition of* p is a list (w_1, z^1), …, (w_M, z^M) with the following properties:

1. Each $w_i \geq 0$ and $\sum_{i=1}^M w_i = 1$.
2. Each $z^i \in \{-1,1\}^n$ and $z^i \neq z^j$ for $i \neq j$.
3. The distribution p is the w_i-weighted convex combination of the uniform monotone subcube distributions $\mathcal{U}_{z^i}$, i.e. $p(x) = \sum_{i=1}^M w_i \mathcal{U}_{z^i}(x)$ for each $x \in \{-1,1\}^n$.

Thus a draw from p can be simulated by the following two-stage process: (i) first choose an index $i \in \{1, 2, \ldots, M\}$ by picking each value i with probability w_i; and then (ii) make a draw from $\mathcal{U}_{z^i}$, i.e. pick a random point in the monotone subcube rooted at z^i.

Not every monotone distribution over $\{-1,1\}^n$ admits a monotone subcube decomposition (for example, one can easily show that the monotone distribution $p(-1,-1) = 0$, $p(-1,1) = p(1,-1) = p(1,1) = \frac{1}{3}$ has no such decomposition). In this section we show that given a monotone subcube decomposition of p, we can define a probability distribution $\mathcal{P}$ over distributions on $\{-1,1\}^n$ with the property that it is hard to distinguish a sample drawn from p from a sample drawn from q, where q is a distribution chosen randomly from $\mathcal{P}$. In the following subsections we will use this fact to obtain lower bounds on the sample complexity of various distribution testing problems.

So let p be a monotone distribution over $\{-1,1\}^n$ which is decomposable into $(w_1, z^1), \ldots, (w_M, z^M)$. Consider the following randomized procedure for constructing a probability distribution q: for $i = 1, \ldots, T$, let $a^i \in \{-1,1\}^n$ be obtained by independently choosing a^i to be z^j with probability w_j for $j = 1, \ldots, M$. Once $a^1, \ldots, a^T$ have been selected, the distribution q is defined as $q(x) = \sum_{i=1}^T \frac{1}{T}\mathcal{U}_{a^i}(x)$, i.e. q is a uniform mixture of the $\mathcal{U}_{a^i}$'s. (The value of T is unconstrained; Claim 7 below hold for any choice of T.) We write $\mathcal{D}(p, T)$ to denote the probability distribution over distributions q which results from the procedure described above.

Fix r to be any positive value which is at most $\sqrt{T}/10$. Let S_1 be a random variable which takes values from the set of all r-tuples of strings from $\{-1,1\}^n$, where a draw from S_1 is obtained by making r independent draws from p. Let S_2 be a random variable which takes values over the same domain, where a draw from S_2 is obtained by (i) first randomly selecting q from $\mathcal{D}(p, T)$ as described above, and then (ii) making r independent draws from q. The following claim is the key to all our generator lower bounds:

Claim 7 $\|S_1 - S_2\|_1 \leq \frac{1}{50}$.

Proof: We first note that since p is decomposable into $(w_1, z^1), \ldots, (w_M, z^M)$, we may view each string in S_1 as being obtained by independently (i) first choosing a $j \in \{1, \ldots, M\}$ where j is chosen with probability w_j; and then (ii) making a draw from $\mathcal{U}_{z^j}$.

On the other hand, once the strings $a^1, \ldots, a^T$ defining q have been chosen, we may view each string in S_2 as being obtained by independently (i) first choosing a uniform random value ℓ from $\{1, \ldots, T\}$; and then (ii) making a draw from $\mathcal{U}_{a^\ell}$. For $i = 1, \ldots, r$ let ℓ_i be the uniform value from $\{1, \ldots, T\}$ which is chosen in step (i) when the i-th string in S_2 is selected, i.e. ℓ_i is the index for which of the T uniform monotone subcube distributions $\mathcal{U}_{a^1}, \ldots, \mathcal{U}_{a^T}$ the i-th string is drawn from. A straightforward application of the Birthday Paradox shows that with probability at least $\frac{99}{100}$, all r indices $\ell_1, \ldots, \ell_r$ take distinct values from each other. Conditioned on all these indices being distinct, we may view the i-th string in S_2 as being obtained by independently (i) first choosing the string a^{ℓ_i} by taking it to be z^j with probability w_j; then (ii) making a draw from $U_{a^{\ell_i}}$. (We have independence across all i because all indices $\ell_1, \ldots, \ell_r$ are distinct from each other.) But this is precisely the same procedure which is used to obtain S_1. So conditioned on all indices $\ell_1, \ldots, \ell_r$ being distinct, the distribution of S_2 is identical to the distribution of S_1.

Thus, with probability at least $\frac{99}{100}$ over the random choice of indices $\ell_1, \ldots, \ell_r$ in the construction of S_2, the distributions of S_2 and S_1 are identical. Even if the remaining $\frac{1}{100}$ probability mass for S_2 completely misses the support of S_1, we have that $\|S_1 - S_2\| \leq \frac{1}{50}$. ■

4.2 A lower bound for testing uniformity Our main result in this section is a $\Omega(n/\log^2 n)$ lower bound on the number of draws required to distinguish the uniform distribution over $\{-1,1\}^n$ even from monotone distributions which are quite far from uniform:

Theorem 8 *Any algorithm A which, given generator access to an unknown monotone distribution p over $\{-1,1\}^n$, determines correctly (with probability at least $4/5$) whether $p \equiv \mathcal{U}$ or $\|p - \mathcal{U}\|_1 > 2 - \frac{1}{n^9}$ must make $\Omega(n/\log^2 n)$ draws from the generator.*

(It will be clear from the proof of Theorem 8 that the $2 - \frac{1}{n^9}$ lower bound on $\|p - \mathcal{U}\|_1$ could in fact be taken to be $2 - \frac{1}{n^k}$ for any constant $k > 0$; we give the proof for $2 - \frac{1}{n^9}$ for simplicity.) This lower bound shows that in terms of the dependence on n, the $\Theta(n \log n)$ query algorithm given in Section 3 is optimal up to a polylog(n) factor. It is interesting to contrast this lower bound with the case of testing uniformity for monotone distributions over the domain $[N] = \{1, 2, \ldots, N\}$, where (as shown in [5]) there is an algorithm that makes $\Theta(1/\epsilon^2)$ queries independent of N.

The high-level idea of our lower bound for testing uniformity is as follows. We first show that the uniform distribution $\mathcal{U}$ cannot be distinguished from the product distribution $\mathcal{P}_{\tau'}$ over $\{-1,1\}^n$ with parameter $\tau' = \frac{1}{2} + \frac{12 \log n}{n}$ using fewer than $\Omega(n/\log^2 n)$ many draws. We then use the probabilistic method and the technique of Section 4.1 to show that there is a distribution q which is very far from $\mathcal{U}$, but which cannot be distinguished from $\mathcal{P}_{\tau'}$ using fewer than $\Omega(n/\log^2 n)$ many draws. Combining these bounds, it follows that q cannot be distinguished from $\mathcal{U}$ using fewer than $\Omega(n/\log^2 n)$ many draws.

Proof of Theorem 8: The proof is by contradiction; so suppose that A is an algorithm which makes $N = o(\frac{n}{\log^2 n})$ draws from the generator and satisfies $\Pr_{S \leftarrow (\mathcal{U})^N}[A(S) \text{ outputs "uniform"}] \geq \frac{4}{5}$. We will show that any such algorithm must also satisfy $\Pr_{S \leftarrow (q)^N}[A(S) \text{ outputs "uniform"}] \geq \frac{1}{2}$ for some monotone distribution q which satisfies $\|q - \mathcal{U}\|_1 \geq 2 - \frac{1}{n^9}$. This proves the theorem, since a correct algorithm would have $\Pr_{S \leftarrow (q)^N}[A(S) \text{ outputs "uniform"}] \leq \frac{1}{5}$ for all such distributions q.

Let $\tau' = \frac{1}{2} + \frac{12 \log n}{n}$. We have the following claim.

Claim 9 $\Pr_{S \leftarrow (\mathcal{P}_{\tau'})^N}[A(S)$ *outputs "uniform"*$] \geq \frac{79}{100}$.

Proof: [of Claim 9]

If $\Pr_{S \leftarrow (\mathcal{P}_{\tau'})^N}[A(S)$ outputs "uniform"$] < \frac{79}{100}$, then by running A $\Theta(1)$ many times we would have an algorithm which could distinguish with high success probability (say at least 9/10) between $\mathcal{U}$ and $\mathcal{P}_{\tau'}$ using $\Theta(N)$ many draws from the unknown generator. But as we show below, any algorithm which with probability at least $\frac{9}{10}$ can correctly determine whether an unknown generator is $\mathcal{U}$ or $\mathcal{P}_{\tau'}$ must make $\Omega(\frac{n}{\log^2 n})$ draws to the generator. ■

We show that the uniform distribution $\mathcal{U}$ and the product distribution $\mathcal{P}_\tau$ where $\tau = \frac{1}{2} + \epsilon$ are indistinguishable from fewer than $\Omega(\frac{1}{\epsilon^2 n})$ many draws:

Fact 10 *Any algorithm which with probability at least $\frac{9}{10}$ can correctly determine whether an unknown generator is $\mathcal{U}$ or is $\mathcal{P}_\tau$ must make $\Omega(\frac{1}{\epsilon^2 n})$ many draws to the generator.*

Proof: Suppose that A is an algorithm which, given an unknown coin of bias either $\frac{1}{2}$ or $\frac{1}{2} + \epsilon$, can output the right answer with probability at least $\frac{9}{10}$. It is well known that A must make at least $\Omega(\frac{1}{\epsilon^2})$ many coin tosses (see e.g. [4]). It is clear that T fair coin tosses can be converted into T/n draws from $\mathcal{U}$, and T biased coin tosses can be converted into T/n draws from $\mathcal{P}_\tau$, simply by grouping the tosses into strings of length n. Thus any distinguisher as described in the statement of Fact 10 must make $\Omega(\frac{1}{\epsilon^2 n})$ many draws from the generator, since otherwise it would yield a distinguisher for the coin problem which requires $o(\frac{1}{\epsilon^2})$ many coin tosses. ■

Now we use the lower bound technique of Section 4.1 to show that $\mathcal{P}_{\tau'}$ is indistinguishable from some distribution q which is far from $\mathcal{U}$. Given a value $\tau = \frac{1}{2} + \epsilon$ with $\epsilon \geq 0$, the following claim shows that making a draw from $\mathcal{P}_\tau$ is equivalent to the following two-step process: (i) first pick a random y from $\{-1,1\}^n$ according to the distribution $\mathcal{P}_{2\epsilon}$ (i.e. each y with $\text{ones}(y) = i$ is chosen with probability $(2\epsilon)^i(1-2\epsilon)^{n-i}$); and then (ii) pick a random x from $\mathcal{U}_y$.

Claim 11 *For any $\tau = \frac{1}{2} + \epsilon \in [\frac{1}{2}, 1]$ the product distribution $\mathcal{P}_\tau$ is decomposable as*

$$\mathcal{P}_\tau(x) = \sum_{i=0}^{n} \sum_{y \in \{-1,1\}^n:\ ones(y)=i} (2\epsilon)^i (1-2\epsilon)^i \cdot \mathcal{U}_y(x). \tag{1}$$

Proof: For $i = 0, 1, \ldots, n$ let α_i denote $(2\epsilon)^i(1-2\epsilon)^{n-i}$. We first note that $\sum_{i=0}^n \binom{n}{i}\alpha_i = 1$ and thus the right side of (1) is a convex combination of uniform monotone subcube distributions. Now fix an $x \in \{-1,1\}^n$ which has $\text{ones}(x) = k$. There are $\binom{k}{i}$ points $y \leq x$ which have $\text{ones}(y) = i$, and each such y contributes $\alpha_i \mathcal{U}_y(x) = (2\epsilon)^i (1-2\epsilon)^{n-i}/2^{n-i} = (2\epsilon)^i \left(\frac{1}{2} - \epsilon\right)^{n-i}$ to the probability of x under the right side of (1). We thus have that the probability of x under the right side of (1) is

$$\begin{aligned}\sum_{i=0}^{k} \binom{k}{i} (2\epsilon)^i (\tfrac{1}{2} - \epsilon)^{n-i} &= (\tfrac{1}{2} - \epsilon)^{n-k} \sum_{i=0}^{k} \binom{k}{i} (2\epsilon)^i (\tfrac{1}{2} - \epsilon)^{k-i} \\ &= (\tfrac{1}{2} - \epsilon)^{n-k} (\tfrac{1}{2} + \epsilon)^k\end{aligned}$$

and thus the right side of (1) is indeed a decomposition of the product distribution $\mathcal{P}_\tau$ as desired. ■

Now let q be a distribution over $\{-1,1\}^n$ which is drawn from $\mathcal{D}(\mathcal{P}_{\tau'}, n^2)$ as described in Section 4.1 (i.e. n^2 points $a^1, \ldots, a^{n^2}$ are independently selected from $\{-1,1\}^n$ where each point of weight i is chosen with probability $(2\epsilon)^i(1-2\epsilon)^{n-i}$, and q is the uniform convex combination of the corresponding uniform monotone subcube distributions). Let S_1 be the random variable defined by making N independent draws from $\mathcal{P}_{\tau'}$, and let S_2 be the random variable defined by (i) first selecting q randomly as described above, and then (ii) making N independent draws from q.

Since $N < \frac{n}{10}$, by Claim 7 we have that $\|S_1 - S_2\|_1 \leq \frac{1}{50}$. Combining this with Fact 1, we have that $|\Pr_{S \leftarrow (\mathcal{P}_{\tau'})^N}[A(S) \text{ outputs “uniform]} - \Pr_{q \leftarrow \mathcal{D}(\mathcal{P}_{\tau'}, n^2),\ S \leftarrow (q)^N}[A(S) \text{ outputs “uniform]}| \leq \frac{1}{50}$, and combining this with Claim 9 we have that $\Pr_{q \leftarrow \mathcal{D}(\mathcal{P}_{\tau'}, n^2),\ S \leftarrow (q)^N}[A(S) \text{ outputs “uniform”}] \geq \frac{77}{100}$. This latter inequality immediately yields the following lemma:

Lemma 12 *With probability at least $\frac{54}{100}$ over the random selection of q from $\mathcal{D}(\mathcal{P}_{\tau'}, n^2)$ we have that $\Pr_{S \leftarrow (q)^N}[A(S) \text{ outputs “uniform”}] \geq \frac{1}{2}$.*

Now we use the following lemma which we prove shortly:

Lemma 13 *With probability $1 - o(1)$ over the random choice of q from $\mathcal{D}(\mathcal{P}_{\tau'}, n^2)$, we have that $\|q - \mathcal{U}\|_1 \geq 2 - \frac{2}{n^9}$.*

Combining Lemmas 12 and 13, we may conclude that there exists some distribution q in the support of $\mathcal{D}(\mathcal{P}_{\tau'}, n^2)$ which has both $\Pr_{S \leftarrow (q)^N}[A(S) \text{ outputs “uniform”}] \geq \frac{1}{2}$ and $\|q - \mathcal{U}\|_1 \geq 2 - \frac{2}{n^9}$. Since every distribution q in the support of $\mathcal{D}(\mathcal{P}_{\tau'}, n^2)$ is monotone, this proves Theorem 8. ■

Proof of Lemma 13: Let $\epsilon = \frac{12 \log n}{n}$. The distribution q is selected from $\mathcal{D}(\mathcal{P}_{\tau'}, n^2)$ by independently drawing n^2 many points $a^1, \ldots, a^{n^2}$ from the product distribution $\mathcal{P}_{2\epsilon}$ and taking $q = \sum_{i=1}^{n^2} \frac{1}{n^2} \mathcal{U}_{a^i}$. For any fixed i, the expected value of $\text{ones}(a^i)$ is $2\epsilon n = 24 \log n$, and the multiplicative Chernoff bound implies that $\Pr[\text{ones}(a^i) \leq 12 \log n]$ is at most $\exp(-3 \log n) < \frac{1}{n^3}$. A union bound across $i = 1, \ldots, n^2$ gives that with probability $1 - o(1)$, no a^i has $\text{ones}(a^i) < 12 \log n$. If each of the n^2 many a^i's has $\text{ones}(a^i) \geq 12 \log n$, then each $\mathcal{U}_{a^i}$ is supported on at most $2^n/n^{12}$ points, and thus the support of q contains at most $2^n/n^{10}$ points. It follows immediately from this that $\|q - \mathcal{U}\|_1$ must be at least $2 - \frac{2}{n^{10}}$, and the lemma is proved. (Lemma 13)■

In terms of ϵ, it is easy to see that at least $1/\epsilon^2$ many queries are required:

Theorem 14 *Any algorithm which determines correctly (with probability at least $4/5$) whether a monotone distribution p is equivalent to $\mathcal{U}$ or has $\|p - \mathcal{U}\|_1 > \epsilon$ must make at least $\Omega(1/\epsilon^2)$ draws from p.*

Proof: We assume without loss of generality that n is odd, so exactly half the points $x \in \{-1, 1\}^n$ have $|x| > 0$ and exactly half have $|x| < 0$. Let q be the distribution which puts weight $(1 + \frac{\epsilon}{2})/2^n$ on each x with $|x| > 0$ and puts weight $(1 - \frac{\epsilon}{2})/2^n$ on each x with $|x| < 0$. Consider the problem of determining whether p is equivalent to $\mathcal{U}$ or to q (with a guarantee that it is one of the two). If there were an $o(\frac{1}{\epsilon^2})$-query algorithm for this problem, then there would be an $o(\frac{1}{\epsilon^2})$-query algorithm to determine whether a coin is perfectly fair or has bias ϵ. (Given a coin to be tested, we can convert each “heads” to a string $x \in \{-1, 1\}^n$ chosen uniformly at random from all strings satisfying $|x| > 0$, and convert each “tails” to a string chosen uniformly at random from all strings satisfying $|x| < 0$.) But it is well known (see e.g. [4]) that any algorithm for the coin problem requires $\Omega(\frac{1}{\epsilon^2})$ many samples. ■

4.3 An exponential lower bound for testing equivalence to a fixed product distribution

In the previous sections we have seen that $\tilde{\Theta}(n)$ draws from a generator for a monotone distribution are necessary and sufficient to distinguish between the two cases of it being either equivalent to $\mathcal{U}$ or far from $\mathcal{U}$. In contrast, we now prove a $2^{\Omega(n)}$ lower bound on the number of draws which are required to distinguish between a generator for an unknown monotone distribution being either equivalent to $\mathcal{P}_{4/5}$ or far from $\mathcal{P}_{4/5}$:

Theorem 15 *Any algorithm A which, given generator access to an unknown monotone distribution p over $\{-1,1\}^n$, determines correctly (with probability at least $4/5$) whether $p \equiv \mathcal{P}_{4/5}$ or $\|p - \mathcal{P}_{4/5}\|_1 > 2 - \frac{1}{n}$ must make at least $2^{.16n}$ draws from the unknown generator.*

(Here too the bound $2 - \frac{1}{n}$ can be strengthened to $2 - \frac{1}{n^k}$ for any constant $k > 0$.) Theorem 15 is in sharp contrast with known results for a similar question of testing whether two unknown generators p and q for monotone distributions over $[N]$ are identical or have $\|p - q\|_1 \geq \epsilon$. In [5] an algorithm is given for this problem which runs in polylog$(N, \frac{1}{\epsilon})$ time steps. In contrast, here even though the generator q is known exactly and has a very simple structure, the problem is hard.

Proof of Theorem 15: The proof is again by contradiction; so suppose that A is an algorithm which makes $N < 2^{.16n}$ draws from the generator and satisfies $\Pr_{S \leftarrow (\mathcal{P}_{4/5})^N}[A(S)$ outputs "$\mathcal{P}_{4/5}$"$] \geq \frac{4}{5}$. By Claim 11 the distribution $\mathcal{P}_{4/5}$ is decomposable; as described in Section 4.1 we consider the distribution $\mathcal{D}(\mathcal{P}_{4/5}, T)$ over distributions q, where now we take $T = 100 \cdot 2^{.32n}$. Let S_1 be the random variable defined by making N independent draws from $\mathcal{P}_{4/5}$, and let S_2 be the random variable defined by (i) first selecting q randomly from $\mathcal{D}(\mathcal{P}_{4/5}, T)$ and then (ii) making N independent draws from q. Since $N < \sqrt{T}/10$, by Claim 7 and Fact 1 we have that $|\Pr_{q \leftarrow \mathcal{D}(\mathcal{P}_{4/5},T),\ S \leftarrow (q)^N}[A(S)$ outputs "$\mathcal{P}_{4/5}$"$] - \Pr_{S \leftarrow (\mathcal{P}_{4/5})^N}[A(S)$ outputs "$\mathcal{P}_{4/5}$"$]| \leq \frac{1}{50}$ so consequently $\Pr_{q \leftarrow \mathcal{D}(\mathcal{P}_{4/5},T),\ S \leftarrow (q)^N}[A(S)$ outputs "$\mathcal{P}_{4/5}$"$] \geq \frac{78}{100}$. Analogous to Lemma 12, from this we immediately obtain

Lemma 16 *With probability at least $\frac{56}{100}$ over the random selection of q from $\mathcal{D}(\mathcal{P}_{4/5}, T)$ we have that $\Pr_{S \leftarrow (q)^N}[A(S)$ outputs "$\mathcal{P}_{4/5}$"$] \geq \frac{1}{2}$.*

The following lemma now suffices to prove Theorem 15.

Lemma 17 *For $T = 100 \cdot 2^{.32n}$, with probability $1 - o(1)$ over the choice of q from $\mathcal{D}(\mathcal{P}_{4/5}, T)$ we have that $\|q - \mathcal{P}_{4/5}\|_1 \geq 2 - \frac{2}{n}$.*

Proof: Taking $\tau = \frac{4}{5}$ and $\epsilon = \frac{3}{10}$ in Claim 11, the distribution q is selected from $\mathcal{D}(\mathcal{P}_{4/5}, T)$ by independently drawing T points $a^1, \ldots, a^T$ from the product distribution $\mathcal{P}_{3/5}$ and taking $q = \sum_{i=1}^T \frac{1}{T}\mathcal{U}_{a^i}$. For any fixed i, since a^i is drawn from $\mathcal{P}_{3/5}$, using the multiplicative Chernoff bound we have that ones(a^i) lies outside the interval $[\frac{3n}{5} - c\sqrt{n \log n}, \frac{3n}{5} + c\sqrt{n \log n}]$ with probability at most $\frac{1}{2n^2}$, where c is some absolute constant. Consequently, the expected fraction of a^i's which have ones(a^i) outside this interval is at most $\frac{1}{2n^2}$; another multiplicative Chernoff bound gives that with probability $1 - o(1)$, at most a $\frac{1}{n^2}$ fraction of $a^1, \ldots, a^T$ have ones(a^i) outside of $[\frac{3n}{5} - c\sqrt{n \log n}, \frac{3n}{5} + c\sqrt{n \log n}]$. We will refer to the a^i's which have ones$(a^i) \in [\frac{3n}{5} - c\sqrt{n \log n}, \frac{3n}{5} + c\sqrt{n \log n}]$ as *good* a^i's, and to the other a^i's as *bad* a^i's. Thus, we may henceforth assume that at most a $\frac{1}{n^2}$ fraction of the probability weight assigned by q corresponds to bad a^i's. We now analyze how the remaining (at least) $1 - \frac{1}{n^2}$ weight assigned by q via the good a^i's is distributed relative to the weight distribution of $\mathcal{P}_{4/5}$; we will show that this $1 - \frac{1}{n^2}$ weight of q almost entirely misses the weight assigned by $\mathcal{P}_{4/5}$.

The multiplicative Chernoff bound implies that under the distribution $\mathcal{P}_{4/5}$, all but (at most) a $\frac{1}{n^2}$ fraction of probability weight is on points z which have $\text{ones}(z) \in [\frac{4n}{5} - c_2\sqrt{n \log n}, \frac{4n}{5} + c_2\sqrt{n \log n}]$ for some absolute constant c_2. Fix any integer value $r \in [\frac{4n}{5} - c_2\sqrt{n \log n}, \frac{4n}{5} + c_2\sqrt{n \log n}]$. It is clear that the distribution $\mathcal{P}_{4/5}$ puts equal weight on all points z with $\text{ones}(z) = r$. Now, for any i such that a^i is good, the support of $\mathcal{U}_{a^i}$ contains at most $2^{2n/5 + c\sqrt{n \log n}}$ points, and thus such an a^i contributes nonzero weight to at most $2^{2n/5 + c\sqrt{n \log n}}$ many points z which have $\text{ones}(z) = r$. Since there are at most T good a^i's, the total number of points $z \in \{-1, 1\}^n$ with $\text{ones}(z) = r$ which receive nonzero weight from any good a^i is at most $T 2^{2n/5 + c\sqrt{n \log n}} = 100 \cdot 2^{.72n + c\sqrt{n \log n}}$. Since there are $\binom{n}{r} > 2^{(H(r/n) - o(1))n}$ many points with $\text{ones}(z) = r$, where $H(x) = -x \log x - (1-x)\log(1-x)$ is the standard binary entropy function (see e.g. [8]), since $H(4/5) > .72$ we have that for each $r \in [\frac{4n}{5} - c_2\sqrt{n \log n}, \frac{4n}{5} + c_2\sqrt{n \log n}]$, all but an exponentially small fraction of the weight which $\mathcal{P}_{4/5}$ assigns to points of weight r goes to points z which receive zero weight from the good a^i's under q. Since all but at most $1/n^2$ weight under $\mathcal{P}_{4/5}$ is on such r's, and all but at most $1/n^2$ weight under q is assigned by good a^i's, this proves the lemma. ∎

We can use the approach of Theorem 15 to show that an algorithm given in [5] is close to optimal. In [5] a $O(\log^n m)$ time algorithm is given for determining whether a generator for an unknown monotone distribution over $\{1, \ldots, m\}^n$ is (i) a product of n independent distributions over $\{1, \ldots, m\}$, or (ii) is ϵ-far in L_1 norm from any such product distribution. The following theorem shows that any algorithm for this problem must indeed have an exponential dependence on the dimension n:

Theorem 18 *Let A be an algorithm which, given generator access to an unknown monotone distribution p over $\{-1, 1\}^n$, outputs "product" with probability at least $4/5$ if p is a product distribution, and outputs "not product" with probability at least $4/5$ if for every product distribution $\mathcal{P}_{\tau_1, \ldots, \tau_n}$ the distance $\|p - \mathcal{P}_{\tau_1, \ldots, \tau_n}\|_1$ is at least $\frac{1}{8n}$. Then A must make at least $2^{.16n}$ many queries.*

Proof: Suppose that A is an algorithm which makes $N < 2^{.16n}$ draws from the generator and, for any product distribution $\mathcal{P}$, satisfies

$$\Pr_{S \leftarrow (\mathcal{P})^N}[A(S) \text{ outputs "product"}] \geq \frac{4}{5}.$$

Then A must in particular output "product" with probability at least $4/5$ when run using a generator for $\mathcal{P}_{4/5}$. The proof now follows that of Theorem 15 up to Lemma 17. The following variant of Lemma 17 suffices to give Theorem 18:

Lemma 19 *For $T = 100 \cdot 2^{.32n}$, with probability $1 - o(1)$ over the choice of q from $\mathcal{D}(\mathcal{P}_{4/5}, T)$ we have that $\|q - \mathcal{P}\|_1 > \frac{1}{8n}$ for all product distributions $\mathcal{P}$.*

Proof of Lemma 19: Fix some $i \in \{1, 2, \ldots, n\}$. A Chernoff bound shows that with overwhelmingly high probability (much higher than $1 - \frac{1}{n^2}$) a randomly chosen q drawn from $\mathcal{D}(\mathcal{P}_{4/5}, T)$ will have $|\Pr_q[x_i = 1] - \frac{4}{5}| \leq \frac{1}{8n}$. Taking a union bound across all $i = 1, \ldots, n$ we have that with probability at least $1 - \frac{1}{n}$, a random q has $|\Pr_q[x_i = 1] - \frac{4}{5}| \leq \frac{1}{8n}$ for all i. Now, given any such distribution q, it is immediately clear that if $\mathcal{P}_{\tau_1, \ldots, \tau_n}$ has $|\tau_i - \frac{4}{5}| > \frac{1}{4n}$ for any τ_i, then we have $\|q - \mathcal{P}_{\tau_1, \ldots, \tau_n}\|_1 \geq \frac{1}{8n}$. Thus we need only consider product distributions $\mathcal{P}_{\tau_1, \ldots, \tau_n}$ with $|\tau_i - \frac{4}{5}| < \frac{1}{4n}$ for all i. We now use the following standard fact:

Fact 20 *Let $\mathcal{D}_1, \mathcal{D}_2$ be two distributions over X and $\mathcal{D}_3, \mathcal{D}_4$ be two distributions over Y. Then we have*

$$\|(\mathcal{D}_1 \times \mathcal{D}_3) - (\mathcal{D}_2 \times \mathcal{D}_4)\|_1 \leq \|\mathcal{D}_1 - \mathcal{D}_2\|_1 + \|\mathcal{D}_3 - \mathcal{D}_4\|_1.$$

Applying this fact repeatedly, we have that any distribution $\mathcal{P}_{\tau_1,\dots,\tau_n}$ with $|\tau_i - \frac{4}{5}| < \frac{1}{4n}$ for all i must satisfy $\|\mathcal{P}_{4/5} - \mathcal{P}_{\tau_1,\dots,\tau_n}\|_1 \leq \frac{1}{2}$. Now applying Lemma 17, we have that with probability $1 - o(1)$ a random q will satisfy $\|q - \mathcal{P}_{\tau_1,\dots,\tau_n}\|_1 \geq \frac{3}{2} - \frac{2}{n}$. This proves the lemma. ■

4.4 Lower bound for approximating entropy For p a distribution over a finite set, the *entropy* of p is $H(p) := \sum_x -p(x)\log p(x)$. For $h > 0$ we write $\mathcal{D}_h$ to denote the set of all monotone distributions over $\{-1,1\}^n$ which have entropy at least h. For $\gamma > 1$, we say that algorithm A *γ-approximates the entropy* of distributions in $\mathcal{D}_h$ if the following condition holds: given access to a generator p for any distribution in $\mathcal{D}_h$, with probability at least $4/5$ algorithm A outputs a value $A(p)$ such that $H(p)/\gamma \leq A(p) \leq \gamma H(p)$.

In [3] it was shown that for any constant $\gamma > 1$, given generator access to a monotone distribution over $[N]$ with entropy $\Omega(\gamma^{5/2}/(\log\sqrt{\gamma}+1)(\sqrt{\gamma}-1))$, it is possible to γ-approximate the entropy of the distribution using $O(\log^6 N)$ many draws from the generator. In contrast to this positive result, using the technique of Section 4.1 we can show that approximating the entropy of an unknown monotone distribution over $\{-1,1\}^n$ to within a fixed constant requires $2^{\Omega(n)}$ draws, even if the distribution is guaranteed to have entropy $\Omega(n)$:

Theorem 21 *Any algorithm that 1.16-approximates the entropy of any monotone distribution from $\mathcal{D}_{n/2}$ must make at least $\frac{1}{10}2^{n/20}$ many draws from the generator.*

A variant of our construction yields $\omega(1)$-inapproximability for algorithms which make $\Omega(2^{\sqrt{n}})$ generator draws, even if the distribution is guaranteed to have entropy $\Omega(\sqrt{n})$:

Theorem 22 *Any algorithm that $\Omega(\sqrt{\log n})$-approximates the entropy of any monotone distribution from $\mathcal{D}_{\sqrt{n}}$ must make at least $\frac{1}{10}2^{\sqrt{n}}$ many draws from the generator.*

We prove both of these results.

Proof:[of Theorems 21 and 22] Let p_{MAJ} be the distribution $p = \sum_{y:\ \mathrm{ones}(y)=n/2}\mathcal{U}_y$, i.e. p is a uniform convex combination of all $\binom{n}{n/2}$ uniform monotone subcube distributions rooted at points of weight $n/2$ (we assume throughout that n is even; the case where n is odd is entirely similar). We have:

Claim 23 $H(p_{MAJ}) > .81n$.

Proof: It is easy to see that for $i = n/2, n/2+1, \dots, n$ we have $\Pr_{x\leftarrow p_{MAJ}}[\mathrm{ones}(x) = i] = \binom{n/2}{i-n/2}/2^{n/2}$; we write $p^{(i)}_{MAJ}$ to denote this quantity. It is also clear that p_{MAJ} puts the same probability weight, which is $\binom{n/2}{i-n/2}/(2^{n/2}\cdot\binom{n}{i})$, on each of the $\binom{n}{i}$ strings x with $\mathrm{ones}(x) = i$. Now consider the contribution to $H(p_{MAJ}) = \sum_x -p_{MAJ}(x)\log p_{MAJ}(x)$ which comes from those x which have $\mathrm{ones}(x) = \frac{3n}{4}$. This contribution is

$$p^{(3n/4)}_{MAJ}\cdot\log\frac{2^{n/2}\binom{n}{3n/4}}{\binom{n/2}{n/4}} = p^{(3/4)}_{MAJ}\cdot(H(3/4)-o(1))n.$$

Thus the contribution to $H(p_{MAJ})$ from these strings is at least $p^{(3n/4)}_{MAJ}\cdot(.811n)$. Now for each i in $[3n/4-\sqrt{n\log n},\ 3n/4+\sqrt{n\log n}]$, a similar calculation shows that the contribution to $H(p_{MAJ})$ from those strings x with $\mathrm{ones}(x) = i$ is at least $p^{(i)}_{MAJ}\cdot(.8105n)$. Since only a $1-o(1)$ fraction of the weight of p_{MAJ} is on strings i with $i \notin [3n/4 - \sqrt{n \log n}, 3n/4 + \sqrt{n \log n}]$, we have that $H(p_{MAJ})$ is at least $(1 - o(1))(.8105n) > .81n$. ■

Now consider the distribution $\mathcal{D}(p_{MAJ}, T)$ over probability distributions q obtained from the decomposition of p_{MAJ} into $\binom{n}{n/2}$ uniform monotone subcube distributions. Using the techniques of the previous sections, we can show that if A is any algorithm which makes fewer than $\sqrt{T}/10$ draws from the generator, then there is some distribution q in the support of $\mathcal{D}(p_{MAJ}, T)$ which is indistinguishable from p_{MAJ} with high probability. However, it is easy to see that any distribution q which may be drawn from $\mathcal{D}(p_{MAJ}, T)$ puts nonzero weight on at most $T \cdot 2^{n/2}$ many points in $\{-1, 1\}^n$, and consequently we have $H(p_{MAJ}) \leq (n/2) + \log T$. We also have that q is a convex combination of distributions with entropy $n/2$, so we must have $n/2 \leq H(q) \leq (n/2) + \log T$.

Thus, even with the guarantee that the unknown distribution has entropy at least $n/2$, it is hard to distinguish (with fewer than $\sqrt{T}/10$ draws) between $H(\cdot) > .81n$ and $H(\cdot) < (n/2) + \log T$. This gives rise to a range of lower bounds; if we take for example $T = 2^{.1n}$, then since $\sqrt{.81/.6} > 1.16$ we obtain Theorem 21.

A variant of this construction yields $\omega(1)$-inapproximability for algorithms which make poly(n) many draws from the generator. Let $p^\star$ be the distribution $p^\star = \sum_{y:\ \mathrm{ones}(y) = n - \sqrt{n}} \mathcal{U}_y$. An argument similar to that of Claim 23 shows that $H(p^\star) = \Omega(\sqrt{n} \log n)$. Taking $T = 2^{\sqrt{n}}$, we have that if A is any algorithm which makes fewer than $\frac{1}{10} 2^{\sqrt{n}/2}$ many draws from the generator, then there is some distribution q in the support of $\mathcal{D}(p^\star, 2^{\sqrt{n}})$ which is indistinguishable from $p^\star$. This q must have $\sqrt{n} \leq H(q) \leq \sqrt{n} + \log T = 2\sqrt{n}$, and we obtain Theorem 22. ■

5 The evaluator access model

In this section we study the complexity of the testing problems for monotone distributions over $\{-1, 1\}^n$ that we considered earlier, but now in the evaluator model rather than the generator model. We show that testing uniformity can be done with a single evaluator query, and that testing equivalence versus ϵ-distance from a known distribution can be done in $\Theta(1/\epsilon)$ evaluator queries independent of n. While these problems are substantially easier in the evaluator model than the generator model, we also show strong (superpolynomial in n) lower bounds on the query complexity of approximating entropy of monotone distributions on $\{-1, 1\}^n$ in the evaluator model, and on the query complexity of determining whether two unknown monotone distributions over $\{-1, 1\}^n$ (given as evaluators) are equivalent or far apart.

Previous authors have considered the evaluator model, and in particular have studied the problem of estimating the entropy [3]. For general distributions it was shown that the number of queries required to estimate the entropy is $\Omega(N)$ where N is the size of the domain. However, for monotone distributions over a totally ordered domain, it was shown in [3] that the entropy can be estimated to within a multiplicative factor of $\gamma > 1$ in time $O(\lceil 1/\log \gamma \rceil \log N)$ in the evaluator model.

Theorem 24 *There is a deterministic 1-query evaluator algorithm which correctly decides whether a monotone distribution p over $\{-1, 1\}^n$ is identical to $\mathcal{U}$.*

Proof: The algorithm queries $p(1^n)$ and outputs "uniform" if the result is $1/2^n$ and "not uniform" otherwise. If $p(1^n) \neq 1/2^n$ the distribution is not uniform. On the other hand, if $p(1^n) = 1/2^n$, then since p is monotone we must have $p(x) \leq 1/2^n$ for each $x \in \{-1, 1\}^n$. However, the total sum of all 2^n probabilities must be 1, and thus it must be the case that all domain elements have probability $1/2^n$. ■

Theorem 25 *Let q be a known distribution over $\{-1,1\}^n$ (not necessarily monotone). Given evaluator access to a distribution p over $\{-1,1\}^n$ (not necessarily monotone) and a parameter ϵ, there is an algorithm which makes $5/\epsilon$ evaluator queries and has the following behavior: (i) if $p \equiv q$ then the algorithm outputs "equal" with probability at least $4/5$; and (ii) if $\|p-q\|_1 > \epsilon$ then the algorithm outputs "not equal" with probability at least $4/5$.*

Proof: The testing algorithm chooses $s = 5/\epsilon$ points $x^1, \ldots, x^s \in \{-1,1\}^n$ independently according to q, and queries the evaluator oracle for p at each. If at any x_i it finds that $p(x^i) \neq q(x^i)$, the algorithm outputs "not equal", otherwise it outputs "equal".

It is clear that if $p \equiv q$, the algorithm will always output "equal". Say that $x \in \{-1,1\}^n$ is *good* if $p(x) = q(x)$. Let ρ be the probability that x is not good when chosen according to distribution q and ρ' be the probability that x is not good when chosen according to distribution p. Note that $\rho = \rho'$. To show that if $\|p-q\|_1 > \epsilon$ then the algorithm outputs "not equal" with probability at least $2/3$, we equivalently show the contrapositive, i.e., that if the algorithm outputs "equal" with probability greater than $1/3$ then $\|p-q\|_1 \leq \epsilon$. If the algorithm outputs "equal" with probability at least $1/3$, then $\rho \leq \epsilon/2$. (To see this, note that if $\rho > \epsilon/2$ then Pr[some x which is not good occurs in the s examples] $= 1-(1-\rho)^s > 1-(1-\epsilon/2)^{5/\epsilon}$ which is at least $2/3$ for $0 < \epsilon < 2$.) Thus $\|p-q\|_1 = \sum_{x \text{ good}} |p(x)-q(x)| + \sum_{x \text{ not good}} |p(x)-q(x)| \leq \sum_{x \text{ not good}} (p(x)+q(x)) \leq \rho + \rho' \leq \epsilon$. ■

5.1 Lower bounds in the evaluator model In this section, we prove the following three lower bounds.

Theorem 26 *Any evaluator algorithm that $\sqrt{n/\log n}$-approximates the entropy of any monotone distribution from $\mathcal{D}_{\log^2 n}$ must make $\Omega(n^{\log n})$ calls to the evaluator.*

Theorem 27 *Any algorithm in the evaluator model that 1.58-approximates the entropy of any monotone distribution from $\mathcal{D}_{n/10}$ must make $\Omega(2^{n/10})$ calls to the evaluator.*

Theorem 28 *Any algorithm that, given access to a pair of evaluators for two unknown monotone distributions p and q over $\{-1,1\}^n$, correctly distinguishes the case $p \equiv q$ from the case that $\|p-q\|_1 \geq 1/2$ with probability at least $4/5$ must make $2^{\Omega(\sqrt{n})}$ calls to the evaluator.*

In order to prove the above three theorems, we describe two families $\mathcal{P}$ and $\mathcal{Q}$ of monotone distributions over $\{-1,1\}^n$ that are hard to distinguish when given access to an evaluator oracle. Since distributions $p \in \mathcal{P}$ and $q \in \mathcal{Q}$ have very different entropies and are very far from each other in L_1 distance, this will give us lower bounds for both of those testing problems which are superpolynomial in n, the dimension of the cube. Thus, unlike the case of a totally ordered domain, these testing problems for monotone distributions over $\{-1,1\}^n$ cannot be performed with a number of queries that is polylogarithmic in the size of the domain.

Two families of distributions Let N denote $\sum_{i=0}^{b} \binom{n}{i}$, the number of points $x \in \{-1,1\}^n$ which have $\text{ones}(x) \geq n-b$. (b will be set later; it will always be set to a value between $\log n$ and $n/100$.)

Let ℓ be chosen as the least positive integer which satisfies $\sum_{i=0}^{\ell-b-1} \binom{\ell}{i} \geq N$, and let N' denote $\sum_{i=0}^{\ell-b-1} \binom{\ell}{i}$. Note that if $y \in \{-1,1\}^n$ has $\text{ones}(y) = n-\ell$, then there are precisely N' many points z in $\{-1,1\}^n$ such that $z \geq y$ and $\text{ones}(z) < n-b$. Given $y \in \{-1,1\}^n$ with $\text{ones}(y) = n-\ell$, the distribution p_y is defined as follows: p_y puts weight $1/(2N)$ on each of the N points x in $\{-1,1\}^n$

which have $\text{ones}(x) \geq n-b$, and puts weight $1/(2N')$ on each of the N' points $z \in \{-1,1\}^n$ which have $z \geq y$ and $\text{ones}(z) < n-b$. It is easy to see that p_y is a monotone distribution. The family $\mathcal{P}$ of distributions is $\mathcal{P} = \bigcup p_y$ where the union is over all y with $\text{ones}(y) = n - \ell$.

We now define the other family $\mathcal{Q}$. Given $y \in \{-1,1\}^n$ with $\text{ones}(y) = n/2$, we define the distribution q_y as follows: like the distributions in $\mathcal{P}$, q_y puts weight $1/(2N)$ on each of the N points x in $\{-1,1\}^n$ which have $\text{ones}(x) \geq n-b$. However, q_y distributes the remaining 1/2 weight equally over all $M \equiv \sum_{i=0}^{n/2-b-1} \binom{n/2}{i}$ many points z which satisfy $z \geq y$ and $\text{ones}(z) < n-b$. Note that since $M \gg N'$, each distribution in $\mathcal{Q}$ has much more entropy than each distribution in $\mathcal{P}$; we make this precise in the next claim.

Claim 29 *(i) We have $N \leq (\frac{en}{b})^b < 2^{n/10}$, $\ell \leq b \log(\frac{en}{b}) + 1 < n/10$, and $M \geq \frac{1}{2}2^{n/2}$. (ii) The entropy of each $p_y \in \mathcal{P}$ is $H(p) \leq b\log(\frac{en}{b}) + 2$. (iii) The entropy of each $q_y \in \mathcal{Q}$ is $H(q) \geq n/4$. (iv) For all $p \in \mathcal{P}, q \in \mathcal{Q}$, $|p-q| \geq 1 - o(1)$.*

Proof: (i): Recall that $N = \sum_{i=0}^{b} \binom{n}{i} \leq (\frac{en}{b})^b$ (see e.g. Section 3.4 of [9]), and note that $\log((\frac{en}{b})^b) = b\log(\frac{en}{b})$ which is easily shown to be at most $\frac{9n}{100}$ since $b \leq n/100$. For $\ell' = \log((\frac{en}{b})^b)+1$ we have that $\sum_{i=0}^{\ell'-b-1} \binom{\ell'}{i} = 2^{\ell'} - \sum_{i=0}^{b} \binom{\ell'}{i} \geq 2N - \sum_{i=0}^{b} \binom{\ell'}{i} \geq 2N - \sum_{i=0}^{b} \binom{n}{i} = N$, so $\ell \leq \ell'$. The bound on M follows easily from $b \leq n/10$.

(ii): This follows from the above arguments which show that the support of p_y contains at most $2^{\ell'+1}$ many points.

(iii): This follows from the fact that at least half the weight of q_y is uniform over a set of at least $\frac{1}{2}2^{n/2}$ many points, and the other half of the weight is uniform over a set of at least two points.

(iv): This follows easily by comparing the support sizes of any $p_y \in \mathcal{P}$ and $q_y \in \mathcal{Q}$. ■

Difficulty of distinguishing $\mathcal{P}, \mathcal{Q}$ Members of $\mathcal{P}$ and $\mathcal{Q}$ have very different entropies; but randomly chosen $p \in \mathcal{P}$ and $q \in \mathcal{Q}$ are hard to tell apart given access only to an evaluator. To see why, first note that in both cases the value of an evaluator query at any point x with $\text{ones}(x) \geq n-b$ is exactly $1/(2N)$. Thus one can assume that the distinguishing algorithm only queries points z with $\text{ones}(z) < n-b$. Next, note that if the distinguishing algorithm does manage to find a point z with $\text{ones}(z) < n-b$ which has nonzero probability weight, then it is immediately evident whether the distribution is from $\mathcal{P}$ or $\mathcal{Q}$ based on whether the weight on z is large or small. If the unknown distribution is chosen at random from $\mathcal{P}$ or $\mathcal{Q}$, it is hard to find a point z with nonzero weight; but points with 0 weight do not give much information about whether the distribution is from $\mathcal{P}$ or $\mathcal{Q}$, and thus many queries are required. The following makes this intuition more precise.

Let $z : \{-1,1\}^n \to [0,1]$ be the function that assigns value $1/(2N)$ to each of the N points x with $\text{ones}(x) \geq n-b$, and assigns value 0 to all other points in $\{-1,1\}^n$. In the following, we show that randomly drawn elements of $\mathcal{P}$ and $\mathcal{Q}$ are hard to distinguish from z, and thus are hard to distinguish from each other.

Theorem 30 *Any algorithm that for all $a \in \mathcal{P} \bigcup \{z\}$ (respectively $\mathcal{Q} \bigcup \{z\}$), correctly distinguishes whether a is from $\mathcal{P}$ (respectively $\mathcal{Q}$) or $a = z$ with probability greater than 2/3 must make $\Omega((n/\ell)^b)$ (respectively $\Omega(2^b)$) evaluator queries to a.*

Proof: We give the proof for $\mathcal{P}$ and then sketch the modifications required for proving the result for $\mathcal{Q}$.

We use Yao's principle [12] which says the following: if there is a probability distribution $\mathcal{A}$ over the union of all positive and negative inputs which is such that any deterministic distinguishing algorithm making t queries is correct with probability less than 4/5 for an input chosen from $\mathcal{A}$, then one can conclude that t is a lower bound on the query complexity of any *randomized* distinguishing algorithm.

We define a distribution over the union of all positive and negative inputs as follows: the negative input z is assigned probability 1/2, and each member of $\mathcal{P}$ (the positive inputs) is assigned probability $1/(2|\mathcal{P}|)$.

Let A be a deterministic distinguishing algorithm which makes at most t queries when run on any evaluator from $\mathcal{P} \bigcup \{z\}$. By the discussion above we may assume that A never queries a string x with $\text{ones}(x) \geq n - b$, and also that A halts and outputs the correct answer if it ever receives a nonzero answer in response to a query. Thus we may view A as a binary decision tree of depth at most t, where each node represents a query to a particular string x (which satisfies $\text{ones}(x) < n - b$) and the two outgoing edges are labelled with "$= 0$" or "$\neq 0$". Each edge labelled "$\neq 0$" can be assumed to lead to a leaf at which A outputs "positive," i.e. $a \neq z$ (note that if the input is z the algorithm will never follow a "$\neq 0$" edge). Each leaf of A represents the end of a possible computation and is labelled "negative" ($a = z$) or "positive" ($a \neq z$) according to the output of A. Note that the number of leaves in the tree is only one more than the depth of the tree (because the "$\neq 0$" edges all lead to leaves as described above). The *error of* A, the sum of the probabilities of those inputs for which A outputs the wrong answer, is all incurred by inputs $a \in \mathcal{P}$ which always follow the "$= 0$" edge at each node.

If A is incorrect on z, then A is incorrect with probability at least 1/2; thus we may assume that A is correct on z. Hence A should output "negative" ($a = z$) if all t tested locations are equal to 0. Since inputs from $\mathcal{P}$ for which A does not find a nonzero location end up at the same leaf as z, they will be labelled incorrectly by A.

We now upper bound the number of inputs from $\mathcal{P}$ that follow the "$\neq 0$" labelled edge at any step in the algorithm in order to lower bound the number of leaves required for any A that is incorrect with probability at most 1/5 (and thus incorrect on at most 2/5 of inputs from $\mathcal{P}$). Consider a particular step in the algorithm at which a string $x \in \{-1, 1\}^n$ is queried. Let w be such that $\text{ones}(x) = n - w$. As noted above we can assume that $\ell \geq w > b$, since the query answers for locations of weight outside that range are known in advance. There are $\binom{n-w}{n-\ell} = \binom{n-w}{\ell-w}$ many strings y with $\text{ones}(y) = n - \ell$ and $y \leq x$; this number is maximized (for $b < w \leq \ell$) by taking $w = b + 1$. Since there are $\binom{n}{\ell}$ many strings y with $\text{ones}(y) = n - \ell$, in order for at least 3/5 of the inputs in $\mathcal{P}$ to reach a different leaf than z there must be at least $\frac{3}{5} \cdot \binom{n}{\ell} / \binom{n-b-1}{\ell-b-1} > \Theta(1) \cdot \binom{n}{\ell} / \binom{n-b}{\ell-b}$ many leaves (here the inequality holds because of our bound $\ell < n/10$ from Claim 29). Therefore the depth of the tree is at least $\Theta(1) \cdot \binom{n}{\ell} / \binom{n-b}{\ell-b}$, which is

$$\Theta(1) \frac{n(n-1) \cdots (n-b+1)}{\ell(\ell-1) \cdots (\ell-b+1)} > \Theta(1) \cdot (n/\ell)^b.$$

To show the result for $\mathcal{Q}$, consider a query on string x with $\text{ones}(x) = n - w$ where now $b < w \leq n/2$. There are $\binom{n-w}{n/2}$ many strings y with $\text{ones}(y) = n/2$ and $y \leq x$. Since there are $\binom{n}{n/2}$ many strings y with $\text{ones}(y) = n/2$, arguing as above we have that the depth of the tree must be at least $\Theta(1) \cdot \binom{n}{n/2} / \binom{n-b}{n/2-b} = \Omega(2^b)$. ■

Theorem 30 gives us the following corollary:

Corollary 31 *Any evaluator algorithm that for all $a \in \mathcal{P} \bigcup \mathcal{Q}$ correctly distinguishes whether a is from $\mathcal{P}$ or $\mathcal{Q}$ with probability greater than 4/5 must make $\Omega(2^b)$ queries.*

Taking $b = \log^2 n$ and $b = n/10$, by combining Claim 29 with Corollary 31 we get Theorems 26 and 27 respectively. Theorem 28 follows by combining part (iv) of Claim 29 with Theorem 30.

References

[1] D. Aldous. On the Markov chain simulation method for uniform combinatorial distributions and simulated annealing. *Probability in Engineering and Information Sciences*, 1:33–46, 1987.

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

[3] Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating entropy. In *ACM Symposium on Theory of Computing*, pages 678–687, 2002.

[4] Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing that distributions are close. In *IEEE Symposium on Foundations of Computer Science*, pages 259–269, 2000.

[5] Tugkan Batu, Ravi Kumar, and Ronitt Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *ACM Symposium on Theory of Computing*, pages 381–390, 2004.

[6] M. Furst, J. Jackson, and S. Smith. Improved learning of AC^0 functions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 317–325, 1991.

[7] O. Goldreich and D. Ron. On testing expansion in bounded-degree graphs. *Electronic Colloqium on Computational Complexity*, 7(20), 2000.

[8] W. C. Huffman and V. Pless. *Fundamentals of Error Correcting Codes.* Cambridge University Press, 2003.

[9] M. Kearns and U. Vazirani. *An introduction to computational learning theory.* MIT Press, Cambridge, MA, 1994.

[10] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *J. ACM*, 40(3):607–620, 1993.

[11] R. Servedio. On learning monotone DNF under product distributions. In *Proc. 14th ACM Conference on Computational Learning Theory*, pages 473–489, 2001.

[12] A. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Symposium on Foundations of Computer Science*, pages 222–227, 1977.