THEORY OF COMPUTING, Volume 5 (2009), pp. 191–218
www.theoryofcomputing.org

Distribution-Free Testing Lower Bounds for Basic Boolean Functions

Dana Glasner* Rocco A. Servedio†

Received: September 18, 2008; published: October 17, 2009.

Abstract: In the *distribution-free* property testing model, the distance between functions is measured with respect to an arbitrary and unknown probability distribution $\mathcal{D}$ over the input domain. We consider distribution-free testing of several basic Boolean function classes over $\{0,1\}^n$, namely monotone conjunctions, general conjunctions, decision lists, and linear threshold functions. We prove that for each of these function classes, $\Omega((n/\log n)^{1/5})$ oracle calls are required for any distribution-free testing algorithm. Since each of these function classes is known to be distribution-free properly learnable (and hence testable) using $\Theta(n)$ oracle calls, our lower bounds are polynomially related to the best possible.

ACM Classification: F.2.2, G.3

AMS Classification: 68Q99, 68W20

Key words and phrases: property testing, distribution-free testing, decision list, conjunction, linear threshold function

1 Introduction

The field of property testing deals with algorithms that decide whether an input object is in a certain class or is far from being in the class after reading only a small fraction of the object. Property testing was formally introduced in [22] after significant prior work [5, 4] and has evolved into a rich field of study (see [3, 8, 11, 20, 21] for some surveys). A standard approach in property testing is to view the input to the testing algorithm as a function over some finite domain; the testing algorithm is required to distinguish functions that are in a certain class C from functions that are ε-far from being in class

*Supported in part by an FFSEAS Presidential Fellowship.
†Supported in part by NSF award CCF-0347282, by NSF award CCF-0523664, and by a Sloan Foundation Fellowship.

 DOI: 10.4086/toc.2009.v005a010

C. In the most commonly considered property testing scenario, a function f is ε-far from class C if f disagrees with every function g that is in class C on at least an ε fraction of the points in the input domain; equivalently, the distance between functions f and g is measured with respect to the uniform distribution over the domain. The testing algorithm "reads" f by adaptively querying a black-box oracle for f at points x of the algorithm's choosing (such oracle calls are often referred to as "membership queries" in computational learning theory). The main goal in designing property testing algorithms is to use as few queries as possible to distinguish the two types of functions; ideally the number of queries should depend only on ε and should be independent of the size of f's domain.

In recent years there has been considerable work in the standard "uniform distribution" property testing scenario on testing various natural Boolean function classes. Some classes for which uniform distribution testing results have been obtained are monotone functions [7, 10, 12]; Boolean literals, monotone conjunctions, general conjunctions and s-term monotone DNFs [19]; J-juntas [9]; parity functions (which are equivalent to degree-1 polynomials) [5]; degree-d polynomials [2]; decision lists, s-term DNFs, size-s decision trees and s-sparse polynomials [6]; and linear threshold functions [18].

Distribution-free property testing A natural generalization of the notion of property testing is to consider a broader notion of the distance between functions. Given a probability distribution $\mathcal{D}$ over the domain, we may define the distance between f and g as the probability that an input x drawn from $\mathcal{D}$ satisfies $f(x) \neq g(x)$; the "standard" notion of property testing described above corresponds to the case where $\mathcal{D}$ is the uniform distribution. *Distribution-free property testing* is the study of property testers in a setting where distance is measured with respect to a *fixed but unknown and arbitrary* probability distribution $\mathcal{D}$. Since the distribution $\mathcal{D}$ is unknown, in this scenario the testing algorithm is allowed to draw random samples from $\mathcal{D}$ in addition to querying a black-box oracle for the value of the function.

Distribution-free property testing is well-motivated by very similar models in computational learning theory (namely the model of distribution-free PAC learning with membership queries, which is closely related to the well-studied model of exact learning from equivalence and membership queries), and by the fact that in various settings the uniform distribution may not be the best way to measure distances. Distribution-free property testing has been considered by several authors [1, 13, 16, 14, 15]; we briefly describe some of the most relevant prior work below.

Goldreich *et al.* [13] introduced the model of distribution-free property testing, and observed that any *proper* distribution-free PAC learning algorithm (a learning algorithm for a class of functions that always outputs a hypothesis function that itself belongs to the class) can be used to directly obtain a distribution-free property testing algorithm. They also showed that several graph properties that have testing algorithms with query complexity independent of input size in the uniform-distribution model (such as bipartiteness, k-colorability, ρ-clique, ρ-cut and ρ-bisection) do not have distribution-free testing algorithms with query complexity independent of input size. In contrast, Halevy and Kushilevitz [14] gave a distribution-free algorithm for testing connectivity in sparse graphs that has $\mathrm{poly}(1/\varepsilon)$ query complexity independent of input size.

A range of positive and negative results have been established for distribution-free testing of Boolean functions over $\{0,1\}^n$. Halevy and Kushilevitz [15] showed that any distribution-free monotonicity testing algorithm over $\{0,1\}^n$ must make $2^{\Omega(n)}$ queries; this is in contrast with the uniform distribution setting, where monotonicity testing algorithms are known that have query complexity $\mathrm{poly}(n, 1/\varepsilon)$ [7, 10, 12]. On the other hand, [16] showed that for several important function classes over $\{0,1\}^n$ such as juntas, parities, low-degree polynomials and Boolean literals, there exist distribution-free testing algorithms with query complexity $\mathrm{poly}(1/\varepsilon)$ independent of n; these distribution-free results match the query bounds of uniform distribution testing algorithms for these classes.

To sum up, the current landscape of distribution-free property testing is intriguingly varied. For some testing problems (juntas, parities, Boolean literals, low-degree polynomials, connectivity in sparse graphs) the complexity of distribution-free testing is known to be essentially the same as the complexity of uniform-distribution testing; but for other natural testing problems (monotonicity, bipartiteness, k-colorability, ρ-clique, ρ-cut, ρ-bisection), distribution-free testing provably requires many more queries than uniform-distribution testing.

This work Our work is motivated by the fact that for many Boolean function classes over $\{0,1\}^n$ that are of fundamental interest, a very large gap exists between the query complexities of the best known distribution-free property testing algorithms (which typically follow trivially from learning algorithms and have query complexity $\Omega(n)$) and the best known uniform distribution property testing algorithms (which typically have query complexity $\mathrm{poly}(1/\varepsilon)$ independent of n). A natural goal is to try to close this gap, either by developing efficient distribution-free testing algorithms or by proving lower bounds for distribution-free testing for these classes.

We study distribution-free testability of several fundamental classes of Boolean functions that have been previously considered in the uniform distribution testing framework, and have been extensively studied in various distribution-free learning models. More precisely, we consider the following classes (in order of increasing generality): monotone conjunctions, arbitrary conjunctions, decision lists, and linear threshold functions. Each of these four classes is known to be testable in the uniform distribution setting using $\mathrm{poly}(1/\varepsilon)$ queries, independent of n (see [19] for monotone and general conjunctions, [6] for decision lists, and [18] for linear threshold functions). On the other hand, for each of these classes the most efficient known distribution-free testing algorithm is simply to use a proper learning algorithm. Using the fact that each of these classes has Vapnik-Chervonenkis dimension $\Theta(n)$, standard results in learning theory yield well-known algorithms that use $O(n/\varepsilon)$ random examples and no membership queries (see, e. g., Chapter 3 of [17]), and known results also imply that any learning algorithm must make $\Omega(n)$ oracle calls (see [23]).

Our main results are strong distribution-free lower bounds for testing each of these four function classes. As our first main result, we prove:

Theorem 1.1. *Let T be any algorithm which, given oracle access to an unknown function $f : \{0,1\}^n \to \{0,1\}$ and (sampling) oracle access to an unknown distribution $\mathcal{D}$ over $\{0,1\}^n$, outputs "yes" with probability at least $2/3$ if f is a monotone conjunction, and outputs "no" with probability at least $2/3$ if f is $1/6$-far from every decision list with respect to $\mathcal{D}$. Then T must make $\Omega((n/\log n)^{1/5})$ oracle calls in total.*

Since every monotone conjunction and conjunction can be expressed as a decision list, we have the following corollary:

Corollary 1.2. *Distribution-free testing of monotone conjunctions, conjunctions, and decision lists all require $\Omega((n/\log n)^{1/5})$ queries.*

Additionally, for the case of linear threshold functions we have:

Theorem 1.3. *Let T be any algorithm which, given oracle access to an unknown function $f : \{0,1\}^n \rightarrow \{0,1\}$ and (sampling) oracle access to an unknown distribution $\mathcal{D}$ over $\{0,1\}^n$, tests whether f is a linear threshold function or is $1/4$-far from every linear threshold function with respect to $\mathcal{D}$. Then T must make $\Omega((n/\log n)^{1/5})$ oracle calls in total.*

These results show that for these function classes, distribution-free testing is nearly as difficult (from a query perspective) as distribution-free learning, and is much more difficult than uniform-distribution testing. We remind the reader that unlike several models of learning, testing is not monotone with respect to inclusion: in other words, if C' is a subclass of C, an efficient algorithm for testing C does not immediately imply an efficient algorithm for testing C'. As an example of this, the empty class consisting of no functions is trivially testable, and the "complete" class consisting of all functions is also trivially testable, but many intermediate classes are not trivially testable.

Our Approach For simplicity, we discuss here only the construction for the lower bound for monotone conjunctions. The basic idea is to construct two distributions **YES** and **NO** over pairs $(h, \mathcal{D})$ where h is a Boolean function and $\mathcal{D}$ is a distribution over the boolean cube such that for every pair $(g, \mathcal{D}_g)$ in the support of **YES**, the function g is a monotone conjunction and for every pair $(f, \mathcal{D}_f)$ in the support of **NO**, the function f is far from every monotone conjunction with respect to the distribution $\mathcal{D}_f$. We then show that any algorithm that makes fewer than $\Omega((n/\log n)^{1/5})$ queries can only distinguish between a draw from the **YES**-distribution and the **NO**-distribution with probability at most $1/4$. Since a distribution-free tester must distinguish between the two with probability at least $2/3$, this implies that any distribution-free tester must make at least $\Omega((n/\log n)^{1/5})$ queries.

For both the **YES** and **NO** distributions, a draw from the distribution is obtained by first randomly choosing m triples of n-bit strings $(a^1, b^1, c^1), \ldots, (a^m, b^m, c^m)$. The $2m$ strings $a^1, b^1, a^2, b^2, \ldots, a^m, b^m$ each have (disjoint) sets of ℓ bits set to 0 and $n-\ell$ bits set to 1; each string c^i is the bitwise-and of a^i and b^i. (See the description of distribution $\mathcal{H}$ in Section 3.1.1 for a more detailed description.) A draw of $(g, \mathcal{D}_g)$ from **YES** is constructed in such a way that $g(a^i) = 0$, $g(b^i) = 1$ and $g(c^i) = 0$ for each i; the distribution $\mathcal{D}_g$ puts weight $2/(3m)$ on each b^i point and weight $1/(3m)$ on each c^i point. We define g over the rest of the points in the boolean cube so that g is a monotone conjunction. (See Section 3.1.2 for details.)

In contrast, a draw of $(f, \mathcal{D}_f)$ from **NO** is constructed such that $f(a^i) = 1$, $f(b^i) = 1$ but $f(c^i) = 0$, and the distribution $\mathcal{D}_f$ puts $1/(3m)$ weight on each a^i, b^i, or c^i point. (See Section 3.1.3 for details.) As noted in [19], any monotone conjunction h must satisfy $h(x) \wedge h(y) = h(x \wedge y)$ (where $x \wedge y$ denotes the bitwise AND of x and y) for all $x, y \in \{0,1\}^n$, and so no monotone conjunction h can satisfy $h(a^i) = h(b^i) = 1$ but $h(c^i) = 0$. Thus, every monotone conjunction must differ from f on at least one of a^i, b^i, c^i, for each of the m triples, and f is at least 1/3-far from every monotone conjunction.

We first show that with only random samples from the distribution, a tester cannot distinguish between a draw from the **YES** and **NO** distribution, with high probability. This is fairly easy and is argued in Section 3.3.1. Intuitively, in the **NO** case the a^i and b^i points cannot be told apart, so in both the **YES** case and the **NO** case $1/3$ of the draws "look like" c^i-type points and $2/3$ of the draws "look like" the other type(s).

The bulk of our work is to additionally show that queries "do not help" the tester. While there are various technical complications that need to be dealt with in the formal proof, the high-level idea is that an algorithm can only distinguish between a draw from the **YES**-distribution and **NO**-distribution if it manages to query certain types of points called "witnesses" (see Section 3.3). We show that witnesses are very hard to find when allowed only $o((n/\log n)^{1/5})$ queries since intuitively, a witness can only be found by random guessing (see the proof of Lemma 3.11).

Organization After giving preliminaries in Section 2, in Section 3 we present our construction of "yes" and "no" (function, distribution) pairs that are used in the lower bound for monotone conjunctions, conjunctions and decision lists as well as the proof of the lower bound. In Section 4 we describe a variant of the construction for linear threshold functions, and use it to prove the lower bound for linear threshold functions.

2 Preliminaries

Throughout the paper we deal with Boolean functions over n input variables.

Definition 2.1. Let $\mathcal{D}$ be a probability distribution over $\{0,1\}^n$. Given Boolean functions $f,g:\{0,1\}^n \to \{0,1\}$, the *distance between f and g with respect to $\mathcal{D}$* is defined by $\mathrm{dist}_{\mathcal{D}}(f,g) = \Pr_{x\sim\mathcal{D}}[f(x) \neq g(x)]$.

If C is a class of Boolean functions over $\{0,1\}^n$, we define the *distance between f and C with respect to $\mathcal{D}$* to be $\mathrm{dist}_{\mathcal{D}}(f,C) = \min_{g\in C} \mathrm{dist}_{\mathcal{D}}(f,g)$.

We say that f is *ε-far from C with respect to $\mathcal{D}$* if $\mathrm{dist}_{\mathcal{D}}(f,C) \geq \varepsilon$.

Now we can define the notion of a distribution-free tester for a class of functions C:

Definition 2.2. A *distribution-free tester for class C* is a probabilistic oracle machine T which takes as input a distance parameter $\varepsilon > 0$ and is given access to

- a *black-box oracle* to a fixed (but unknown and arbitrary) function $h:\{0,1\}^n \to \{0,1\}$ (when invoked with input x, the oracle returns the value $h(x)$); and
- a *sampling oracle* for a fixed (but unknown and arbitrary) distribution $\mathcal{D}$ over $\{0,1\}^n$ (each time it is invoked this oracle returns a pair $(x,h(x))$ where x is independently drawn from $\mathcal{D}$).

T must satisfy the following two conditions: for any $h:\{0,1\}^n \to \{0,1\}$ and any distribution $\mathcal{D}$,

- if h belongs to C, then $\Pr[T^{h,\mathcal{D}} = \text{Accept}] \geq \frac{2}{3}$; and
- if h is ε-far from C w.r.t. $\mathcal{D}$, then $\Pr[T^{h,\mathcal{D}} = \text{Accept}] \leq \frac{1}{3}$.

This definition allows the tester to be adaptive and to have two-sided error; this is of course the strongest version for proving lower bounds.

Notation and Terminology For a string $x \in \{0,1\}^n$ we write x_i to denote the i^{th} bit of x. For $x, y \in \{0,1\}^n$ we write $x \wedge y$ to denote the n-bit string z which is the bitwise AND of x and y, i. e., $z_i = x_i \wedge y_i$ for all i. The string $x \vee y$ is defined similarly to be the bitwise OR of x and y.

Recall that the *total variation distance*, or *statistical distance*, between two random variables X and Y that take values in a finite set S is

$$d_{TV}(X,Y) = \frac{1}{2}\sum_{\zeta \in S} |\Pr[X=\zeta] - \Pr[Y=\zeta]|\,.$$

The classes we consider For completeness we define here all the classes of functions that we will consider: these are (in order of increasing generality) monotone conjunctions, general conjunctions, decision lists, and linear threshold functions. We note that each of these function classes is quite basic and natural and has been studied intensively in fields such as computational learning theory.

Definition 2.3. The class MCONJ consists of all monotone conjunctions of any subset of Boolean variables from $x_1,\ldots,x_n$, i. e., all ANDs of (unnegated) Boolean variables.

Definition 2.4. The class CONJ consists of all conjunctions of any subset of Boolean literals over $\{0,1\}^n$ (a literal is a Boolean variable or the negation of a variable).

Definition 2.5. A *decision list* L of length k over the Boolean variables $x_1,\ldots,x_n$ is defined by a list of k pairs and a bit $(\ell_1,\beta_1),(\ell_2,\beta_2),\ldots,(\ell_k,\beta_k),\beta_{k+1}$ where each ℓ_i is a Boolean literal and each β_i is either 0 or 1. Given any $x \in \{0,1\}^n$, the value of $L(x)$ is β_i if i is the smallest index such that ℓ_i is made true by x; if no ℓ_i is true then $L(x) = \beta_{k+1}$. Let DL denote the class of all decision lists of arbitrary length $k \ge 0$ over $\{0,1\}^n$.

Definition 2.6. A *linear threshold function* is defined by a list of $n+1$ real values $w_1,\ldots,w_n,\theta$. The value of the function on input $x \in \{0,1\}^n$ is 1 if $w_1x_1+\cdots+w_nx_n \ge \theta$ and is 0 if $w_1x_1+\cdots+w_nx_n < \theta$. We write LTF to denote the class of all linear threshold functions over $\{0,1\}^n$.

It is well known and easy to see that MCONJ $\subsetneq$ CONJ $\subsetneq$ DL $\subsetneq$ LTF.

3 The lower bound for monotone conjunctions

In this section we will prove the following theorem:

Theorem 3.1. *Let $q = (1/20)(n/\log n)^{1/5}$. There exist two distributions,* **YES** *and* **NO***, over pairs $(h,\mathcal{D})$ where $h : \{0,1\}^n \to \{0,1\}$ is a Boolean function and $\mathcal{D}$ is a distribution over the domain $\{0,1\}^n$ that have the following properties:*

1. *For every pair $(g,\mathcal{D}_g)$ in the support of* **YES***, the function g is a monotone conjunction.*
2. *For every pair $(f,\mathcal{D}_f)$ in the support of* **NO***, the function f is $1/6$-far from* DL *with respect to $\mathcal{D}_f$.*

Moreover, for any probabilistic oracle algorithm T that, given a pair $(h,\mathcal{D})$, makes at most q black-box queries to h and samples $\mathcal{D}$ at most q times, we have

$$\left| \Pr_{(g,\mathcal{D}_g)\sim\mathbf{YES}}[T^{g,\mathcal{D}_g} = Accept] - \Pr_{(f,\mathcal{D}_f)\sim\mathbf{NO}}[T^{f,\mathcal{D}_f} = Accept] \right| \leq \frac{1}{4}.$$

Note that in the above theorem each probability is taken over the draw of the (function,distribution) pair from the appropriate distribution **YES** or **NO**, over the random draws from the distribution $\mathcal{D}_f$ or $\mathcal{D}_g$, and over any internal randomness of algorithm T.

A simple corollary of Theorem 3.1 gives us our main result:

Corollary 3.2. *Distribution-free testing of monotone conjunctions, conjunctions, and decision lists all require $\Omega(n/\log n)^{1/5}$ queries.*

Proof. Since MCONJ $\subset$ CONJ $\subset$ DL, properties (1) and (2) imply that any distribution-free tester for MCONJ, CONJ or DL that is run with distance parameter $\varepsilon = 1/6$ must accept a random pair $(g,\mathcal{D}_g)$ drawn from **YES** with probability at least $2/3$, and must accept a random pair $(f,\mathcal{D}_f)$ drawn from **NO** with probability at most $1/3$. Therefore, by Theorem 3.1, any such tester must make at least $q = \Omega(n/\log n)^{1/5}$ queries. □

3.1 The two distributions

We now define the two distributions, **YES** and **NO**, and prove that these distributions have properties (1) and (2) required by Theorem 3.1. Our constructions are parameterized by three values ℓ, m and s. As we will see the optimal setting (up to multiplicative constants) of these parameters for our purposes is

$$\ell = n^{2/5}(\log n)^{3/5}, \quad m = (n/\log n)^{2/5}, \quad s = \log n. \tag{3.1}$$

To keep the different roles of these parameters clear in our exposition we will present our constructions and analyses in terms of "ℓ," "m," and "s" as much as possible and only plug in the values from (3.1) toward the end of our analysis.

We first define the distribution $\mathcal{H}$ over tuples

$$(R, A_1, B_1, \ldots, A_m, B_m, \alpha_1, \ldots, \alpha_m),$$

where A_1, $B_1, \ldots, A_m, B_m$ are disjoint ℓ-element subsets of $[n]$, the set R equals $\bigcup_i (A_i \cup B_i)$, and α_i is an element of A_i for each i. It is useful to define this distribution since a draw from both the **YES** and **NO** distributions begins with a draw from $\mathcal{H}$.

3.1.1 The $\mathcal{H}$ distribution

A draw from the distribution $\mathcal{H}$ over $(R, A_1, B_1, \ldots, A_m, B_m, \alpha_1, \ldots, \alpha_m)$ tuples is obtained as follows:

- Let $R \subset [n]$ be a set of size $2\ell m$ selected uniformly at random. Randomly partition the set R into $2m$ subsets $A_1, B_1, \ldots, A_m, B_m$, each of size ℓ.

- For each $i = 1, \ldots, m$ let $\alpha(i)$ be an element chosen uniformly at random from the set A_i; we say that $\alpha(i)$ is a *representative* of A_i.

Given a draw $(R, A_1, B_1, \ldots, A_m, B_m, \alpha_1, \ldots, \alpha_m)$ from $\mathcal{H}$, we define the strings a^1, b^1, $c^1, \ldots, a^m$, b^m, c^m in the following way: Let $a^i \in \{0,1\}^n$ be the string whose j^{th} bit is 0 iff $j \in A_i$. The string b^i is defined similarly. The string c^i is defined to be $a^i \wedge b^i$. Similarly we define the set $C_i = A_i \cup B_i$. We sometimes refer to a^i, b^i, c^i as the "points of the i^{th} block."

Additionally, we define the conjunction g_1 to be the conjunction of all variables in $[n] \setminus R$.

3.1.2 The YES distribution

A draw from the distribution **YES** over $(g, \mathcal{D}_g)$ pairs is obtained as follows:

- Make a draw from the $\mathcal{H}$ distribution to obtain $(R, A_1, B_1, \ldots, A_m, B_m, \alpha_1, \ldots, \alpha_m)$.
- The distribution $\mathcal{D}_g$ puts weight $2/(3m)$ on b^i and puts weight $1/(3m)$ on c^i for each $i = 1, \ldots, m$.
- Define the function g_2 as follows: g_2 is a conjunction of length m formed by taking $g_2(x) = x_{\alpha(1)} \wedge \cdots \wedge x_{\alpha(m)}$, i. e., g_2 is an AND of the representatives from each of $A_1, \ldots, A_m$.
- The function g is defined as: $g = g_1 \wedge g_2$.

It is clear that for every $(g, \mathcal{D}_g)$ in the support of **YES**, the function g is a monotone conjunction that contains exactly $n - 2\ell m + m$ variables, so Property (1) of Theorem 3.1 indeed holds. We also note that for each i we have $g(a^i) = g(c^i) = 0$ and $g(b^i) = 1$; see Figure 1.

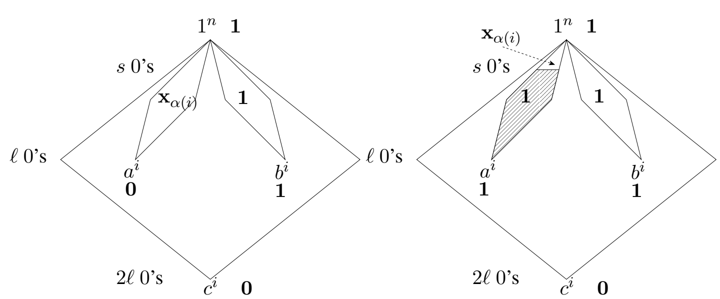

Figure 1: The left figure shows how a yes-function g labels c^i and the points above it (including a^i and b^i). Bold print indicate the label that g assigns. Note that every point above b^i is labeled **1** by g, and points above a^i are labeled according to $\mathbf{x}_{\boldsymbol{\alpha}(i)}$. The right figure shows how a no-function f labels c^i and the points above it (including a^i and b^i). Again, bold print indicates the label that f assigns. Note that every point above b^i is labeled **1** by f, and points above a^i with less than s 0's are labeled according to $\mathbf{x}_{\boldsymbol{\alpha}(i)}$. The i-special points for block i (see Section 3.2) are shaded and are labeled **1** by f.

3.1.3 The NO distribution

A draw from the distribution **NO** of $(f, \mathcal{D}_f)$ pairs is obtained as follows:

- Make a draw from the $\mathcal{H}$ distribution to obtain $(R, A_1, B_1, \ldots, A_m, B_m, \alpha_1, \ldots, \alpha_m)$.
- The distribution $\mathcal{D}_f$ is uniform over the $3m$ points $a^1, \ldots, c^m$.
- Define the function f' as follows: $f'(x) = 0$ if there exists some $i \in [m]$ such that both the following conditions hold:
  - $x_{\alpha(i)} = 0$ and
  - {fewer than s of the elements $j \in A_i$ have $x_j = 0$)} or $\{x_j = 0$ for some $j \in B_i\}$.
- The final function f is defined as $f = g_1 \wedge f'$.

See Figure 1 for a pictorial representation of the yes-function g and the no-function f.
With the definition of **NO** in place, we now establish Property (2) of Theorem 3.1:

Lemma 3.3. *For any pair $(f, \mathcal{D}_f)$ in the support of* **NO** *and any decision list h, the function f is at least $1/6$-far from h w.r.t. $\mathcal{D}_f$.*

Proof. Fix any $(f, \mathcal{D}_f)$ in the support of **NO** and any decision list

$$h = (\ell_1, \beta_1), (\ell_2, \beta_2), \ldots, (\ell_k, \beta_k), \beta_{k+1} .$$

We will show that at least one of the six points $a^1, b^1, c^1, a^2, b^2, c^2$ is labeled differently by h and f. Grouping all m blocks into pairs and applying the same argument to each pair gives the lemma.

Let $\mathrm{first}(a^1)$ be the index of the first literal, $\ell_{\mathrm{first}(a^1)}$, in h that is satisfied by the point a^1, so the value $h(a^1)$ equals $\beta_{\mathrm{first}(a^1)}$. Define $\mathrm{first}(b^1)$, $\mathrm{first}(c^1)$, $\mathrm{first}(a^2)$, $\mathrm{first}(b^2)$, and $\mathrm{first}(c^2)$ similarly. We will assume that h and f agree on all six points, i. e., that $\beta_{\mathrm{first}(a^1)} = \beta_{\mathrm{first}(b^1)} = \beta_{\mathrm{first}(a^2)} = \beta_{\mathrm{first}(b^2)} = 1$ and $\beta_{\mathrm{first}(c^1)} = \beta_{\mathrm{first}(c^2)} = 0$, and derive a contradiction.

We may suppose w.l.o.g. that $\mathrm{first}(a^1) = \min\{\mathrm{first}(a^1), \mathrm{first}(b^1), \mathrm{first}(a^2), \mathrm{first}(b^2)\}$. We now consider two cases depending on whether or not $\mathrm{first}(c^1) < \mathrm{first}(a^1)$. (Note that $\mathrm{first}(a^1)$ cannot equal $\mathrm{first}(c^1)$ since $f(a^1) = 1$ but $f(c^1) = 0$.)

Suppose first that $\mathrm{first}(c^1) < \mathrm{first}(a^1)$. No matter what literal $\ell_{\mathrm{first}(c^1)}$ is, since c^1 satisfies $\ell_{\mathrm{first}(c^1)}$ at least one of a^1, b^1 must satisfy it as well. But this means that $\min\{\mathrm{first}(a^1), \mathrm{first}(b^1)\} \le \mathrm{first}(c^1)$, which is impossible since $\mathrm{first}(c^1) < \mathrm{first}(a^1)$ and $\mathrm{first}(a^1) \le \min\{\mathrm{first}(a^1), \mathrm{first}(b^1)\}$.

Now suppose that $\mathrm{first}(a^1) < \mathrm{first}(c^1)$; then it must be the case that $\ell_{\mathrm{first}(a^1)}$ is a literal "x_j" for some $j \in B_1$. (The only other possibilities are that $\ell_{\mathrm{first}(a^1)}$ is "$\overline{x}_j$" for some $j \in A_i$ or is "x_j" for some $j \in ([n] \setminus C_1)$; in either case, this would imply that $f(c^1) = 1$, which does not hold.) Since $f(c^2) = 0$ and $(c^2)_j = 1$, it must be the case that $\mathrm{first}(c^2) < \mathrm{first}(a^1)$. But no matter what literal $\ell_{\mathrm{first}(c^2)}$ is, since c^2 satisfies it at least one of a^2 and b^2 must satisfy it as well. This means that

$$\min\{\mathrm{first}(a^2), \mathrm{first}(b^2)\} \le \mathrm{first}(c^2) < \mathrm{first}(a^1) \le \min\{\mathrm{first}(a^2), \mathrm{first}(b^2)\} ,$$

which is a contradiction. $\square$

In fact, we have a slightly stronger result for the case of testing monotone conjunctions. For any $(f,\mathcal{D}_f)$ drawn from **NO**, we have $f(a^i) = f(b^i) = 1$ and $f(c^i) = 0$ for each $i = 1,\ldots,m$. It is noted in [19] (and is easy to check) that any monotone conjunction h must satisfy $h(x) \wedge h(y) = h(x \wedge y)$ for all $x,y \in \{0,1\}^n$, and thus must satisfy $h(c^i) = h(a^i) \wedge h(b^i)$. Thus any monotone conjunction h must disagree with f on at least one of a_i, b_i, c_i for all i, and consequently f is $1/3$-far from any monotone conjunction with respect to $\mathcal{D}_f$.

Thus we have established properties (1) and (2) stated at the beginning of this section.

3.2 The idea

In this subsection we attempt to explain the main ideas of the proof of Theorem 3.1. Before we give the main ideas we first provide some preliminary intuition and useful terminology.

It is easy to see that in both the yes-case (when the (function, distribution) pair is drawn from **YES**) and the no-case (when the (function, distribution) pair is drawn from **NO**), any black-box query that sets any variable in $[n] \setminus R$ to 0 will get a 0 response. To give some preliminary intuition for our construction, let us explain here the role that the large conjunction g_1 (over $n-2\ell m$ variables) plays in both the **YES** and **NO** cases. The idea is that because of g_1, a testing algorithm that has obtained strings $z^1,\ldots,z^q$ from the distribution $\mathcal{D}$ will "gain nothing" by querying any string x that has any bit x_i set to 0 that was set to 1 in all of $z^1,\ldots,z^q$. This is because such a variable x_i will with very high probability (over a random choice of $(f,\mathcal{D}_f)$ from **NO** or a random choice of $(g,\mathcal{D}_g)$ from **YES**) be contained in $[n] \setminus R$, so in both the "yes" and "no" cases the query will yield an answer of 0 with very high probability. Consequently there is no point in making such a query in the first place. (We give a rigorous version of this argument in Section 3.3.2.)

The following terminology will be useful: we say that an input $x \in \{0,1\}^n$ is *i-special* if (at least s elements $j \in A_i$ have $x_j = 0$) and ($x_j = 1$ for all $j \in B_i$). Thus an equivalent way to define f' is that $f'(x) = g_2(x)$ unless $g_2(x) = 0$ (because some $x_{\alpha(i)} = 0$) and x is i-special for each i such that $x_{\alpha(i)} = 0$; in this case $f'(x) = 1$.

Here is some high-level intuition for the proof. If T could find a^i, b^i and c^i for some i then T would know which case it is in (yes versus no), because $h(a^i) \wedge h(b^i) = h(c^i)$ if and only if T is in the yes-case. Since T can only make $q \ll \sqrt{m}$ draws from $\mathcal{D}$, the birthday paradox tells us that with high probability the random sample that T draws contains at most one of a^i, b^i and c^i for each i. The c^i-type points (with $n-2\ell$ ones) are labeled negative in both the yes- and no- cases, so these "look the same" to T in both cases. Other than the c^i-type points, in the yes-case the distributions $\mathcal{D}_g$ put weight only on the b^i-type points (with $n-\ell$ ones) which are positively labeled, and in the no-case the distributions $\mathcal{D}_f$ put weight only on the a^i- and b^i-type points (with $n-\ell$ ones) which are also positively labeled. So the draws with $n-\ell$ ones "look the same" to T as well in both cases. So with high probability T cannot distinguish between yes-pairs and no-pairs on the basis of the first q random draws alone. (Corollary 3.5 formalizes this intuition.)

Of course, though, T can also make q queries. Perhaps T can identify a triple (a^i, b^i, c^i) through these queries, or perhaps T can otherwise determine which case it is in even without finding a triple? The crux of the proof is to show that in fact queries actually cannot help T much; we now sketch the idea.

Consider a single fixed block $i \in [m]$. If none of a^i, b^i or c^i are drawn in the initial sample, then by the argument at the start of this subsection, the tester will get no useful information about which case it is in from this block. By the birthday paradox we can assume that at most one of a^i, b^i and c^i is drawn in the initial sample; we consider the three cases in turn.

If b^i is drawn, then again by the argument at the start of this subsection all query points will have all the bits in A_i set to 1; such queries will "look the same" in both the yes- and no- cases as far as the i^{th} block is concerned.

If a^i is drawn (so we are in the no-case), then by the same argument all query points will have all the bits in B_i set to 1. Using the definition of f', as far as the i^{th} block is concerned with high probability it will "look like" the initial a^i point was a b^i-point from the yes-case. This is because the only way the tester can tell that it is in the no-case is if it manages to query a point which has fewer than s bits from A_i set to 0, but the representative $\alpha(i)$ is one of those bits. Such points are hard to find since $\alpha(i)$ is randomly selected from A_i. (See the "a-witness" case in the proof of Lemma 3.11.)

Finally, suppose that c^i is drawn. The only way a tester can distinguish between the yes- and no-cases is by finding an i-special point (or determining that no such point exists), but to find such a point it must make a query with at least s 0's in C_i, all of which lie in A_i. This is hard to do since the tester does not know how the elements of C_i are divided into the sets A_i and B_i. (See the "c-witness" case in the proof of Lemma 3.11.)

3.3 Proof of Theorem 3.1

Fix any probabilistic oracle algorithm T that makes at most q black-box queries to h and samples $\mathcal{D}$ at most q times. Without loss of generality we may assume that T first makes exactly q draws from distribution $\mathcal{D}$, and then makes exactly q (adaptive) queries to the black-box oracle for h.

It will be convenient for us to assume that algorithm T is actually given "extra information" on certain draws from the distribution $\mathcal{D}$. More precisely, we suppose that each time T calls the oracle for $\mathcal{D}$,

- If a "c^i-type" labeled example $(c^i, h(c^i))$ is generated by the oracle, algorithm T receives the triple $(c^i, h(c^i), \alpha(i))$ (recall that $\alpha(i)$ is the index of the variable from C_i that belongs to the conjunction g_2);
- If a "non-c^i-type" labeled example $(x, h(x))$ is generated by the oracle where $x \neq c^i$ for all $i = 1, \ldots, m$, algorithm T receives the triple $(x, h(x), 0)$. (Thus there is no "extra information" given on non-c^i points.)

It is clear that proving Theorem 3.1 for an arbitrary algorithm T that receives this extra information establishes the original theorem as stated (for algorithms that do not receive the extra information).

Following [15], we now define a *knowledge sequence* to precisely capture the notion of "what an algorithm learns from its queries." A knowledge sequence is a sequence of elements corresponding to the interactions that an algorithm has with each of the two oracles. The first q elements of a knowledge sequence are triples as described above; each corresponds to an independent draw from the distribution $\mathcal{D}$. The remaining elements of the knowledge sequence are input-output pairs corresponding to the algorithm's calls to the black-box oracle for h. (Recall that these later oracle calls are adaptive, i. e., each query point can depend on the answers received from previous oracle calls.)

Notation For any oracle algorithm ALG, let $\mathcal{P}^{ALG}_{yes}$ denote the distribution over knowledge sequences induced by running ALG on a pair $(g,\mathcal{D}_g)$ randomly drawn from **YES**. Similarly, let $\mathcal{P}^{ALG}_{no}$ denote the distribution over knowledge sequences induced by running ALG on a pair $(f,\mathcal{D}_f)$ randomly drawn from **NO**. Where there is no risk of confusion we will sometimes write $\mathcal{P}^{ALG}_{yes}$ to denote a random variable drawn from this distribution, and similarly for $\mathcal{P}^{ALG}_{no}$.

For $0 \le i \le q$ we write $\mathcal{P}^{ALG}_{yes,i}$ to denote the length-$(q+i)$ prefix of a knowledge sequence drawn from $\mathcal{P}^{ALG}_{yes}$, and similarly for $\mathcal{P}^{ALG}_{no,i}$.

We will prove Theorem 3.1 by showing that the statistical distance $d_{TV}(\mathcal{P}^T_{yes},\mathcal{P}^T_{no})$ between distributions $\mathcal{P}^T_{yes}$ and $\mathcal{P}^T_{no}$ is at most $1/4$.

3.3.1 Most sequences of draws are "clean" in both the yes- and no- cases

The main result of this subsection is Corollary 3.5; intuitively, this corollary shows that given only q draws from the distribution and no black-box queries, it is impossible to distinguish between the yes- and no- cases with high accuracy. This is achieved via the notion of a "clean" sequence of draws from the distribution, which we now explain.

Let $S = x^1,\dots,x^q$ be a sequence of q examples drawn from distribution $\mathcal{D}$, where $\mathcal{D}$ is either $\mathcal{D}_f$ for some $(f,\mathcal{D}_f) \sim$ **NO** or $\mathcal{D}_g$ for some $(g,\mathcal{D}_g) \sim$ **YES**. In either case there is a corresponding set of points $a^1,b^1,c^1,\dots,a^m,b^m,c^m$ as described in Section 3.1, corresponding to the initial draw from the $\mathcal{H}$ distribution. We say that S is *clean* if S does not hit any block $1,\dots,m$ twice, i. e., if the number of different blocks from $i = 1,\dots,m$ for which S contains some point a^i, b^i or c^i is exactly q.

We note that strictly speaking, cleanliness is not a property of a sequence S drawn from $\mathcal{D}_f$ or $\mathcal{D}_g$, but rather a property of the way the sequence S is generated (since it depends on the random choice of a^i, b^i, c^i, i. e., on the draw from $\mathcal{H}$). With this understood, for conciseness we shall speak of cleanliness simply as a property of a sequence S of draws from $\mathcal{D}_f$ or $\mathcal{D}_g$. We further note that while a draw from $\mathcal{P}^T_{no,0}$ is, strictly speaking, a sequence of q triples (as described at the beginning of Section 3.3), without risk of confusion we will simply write "$S \sim \mathcal{P}^T_{no,0}$" to indicate the corresponding sequence of draws from $\mathcal{D}_f$ (and likewise we write "$S \sim \mathcal{P}^T_{yes,0}$" to indicate the corresponding sequence of draws from $\mathcal{D}_g$).

With this notion and these conventions in place, we have the following claim and its easy corollary:

Claim 3.4. *We have*

$$\Pr_{S\sim\mathcal{P}^T_{yes,0}}[S \text{ is clean}] = \Pr_{S\sim\mathcal{P}^T_{no,0}}[S \text{ is clean}] \ge 1 - q^2/m\,.$$

Furthermore, the conditional random variables

$$(S \mid S \text{ is clean})_{S\sim\mathcal{P}^T_{yes,0}} \quad \text{and} \quad (S \mid S \text{ is clean})_{S\sim\mathcal{P}^T_{no,0}}$$

are identically distributed.

Proof. We first show that

$$\Pr_{S \sim \mathcal{P}^T_{yes,0}}[S \text{ is clean}] = \Pr_{S \sim \mathcal{P}^T_{no,0}}[S \text{ is clean}] \geq 1 - q^2/m\,.$$

Actually, we will prove something stronger, which is that regardless of which (g, D_g) is drawn from **YES**, the conditional probability of getting a clean sequence is at least $1 - q^2/m$. Fix any $(g, \mathcal{D}_g)$ in the support of **YES**, and consider the outcomes of $S \sim \mathcal{P}^T_{yes,0}$ corresponding to this $(g, \mathcal{D}_g)$ being drawn from **YES**. Since each independent draw from $\mathcal{D}_g$ hits each block $1, \ldots, m$ with probability $1/m$, the probability that $S \sim \mathcal{P}^T_{yes,0}$ is clean is

$$\prod_{i=1}^{q} \left(1 - \frac{i-1}{m}\right) \geq 1 - q^2/m\,.$$

The same argument shows that $\Pr_{S \sim \mathcal{P}^T_{no,0}}[S \text{ is clean}]$ also equals $\prod_{i=1}^{q} \left(1 - \frac{i-1}{m}\right)$.

Now we show that the conditional random variables are identically distributed. It is not difficult to see that for any $0 \leq j \leq q-1$, given any particular length-j prefix of a draw from $\mathcal{P}^T_{yes,0}$, conditioned on the draw from $\mathcal{P}^T_{yes,0}$ being clean, the $(j+1)$-st element of the draw from $\mathcal{P}^T_{yes,0}$ has

- a $2/3$ chance of being a triple $(x, 1, 0)$ where $x \in \{0,1\}^n$ has ℓ zeros and the locations of the ℓ zeros are selected uniformly at random (without replacement) from the set of those bit positions that had value 1 in all j of the previous draws;
- a $1/3$ chance of being a triple $(x, 0, \alpha)$ where x has 2ℓ zeros, the locations of the 2ℓ zeros are selected uniformly at random (without replacement) from the same set of bit positions described above, and α is an index drawn uniformly at random from the indices of the 2ℓ zeros in x.

It is also not difficult to see that given any particular length-j prefix of a draw from $\mathcal{P}^T_{no,0}$, conditioned on the draw from $\mathcal{P}^T_{no,0}$ being clean, the $(j+1)$-st element of the draw from $\mathcal{P}^T_{no,0}$ is distributed in the exact same way. □

Corollary 3.5. *The statistical distance $d_{TV}(\mathcal{P}^T_{yes,0}, \mathcal{P}^T_{no,0})$ is at most q^2/m.*

Proof. We can express the statistical distance between $\mathcal{P}^T_{yes,0}$ and $\mathcal{P}^T_{no,0}$ as

$$\frac{1}{2}\sum_{\zeta} \Bigg| \Pr_{S \sim \mathcal{P}^T_{yes,0}}[(S = \zeta)\ \&\ (S \text{ is clean})] + \Pr_{S \sim \mathcal{P}^T_{yes,0}}[(S = \zeta)\ \&\ (S \text{ not clean})]$$
$$- \Pr_{S \sim \mathcal{P}^T_{no,0}}[(S = \zeta)\ \&\ (S \text{ is clean})] - \Pr_{S \sim \mathcal{P}^T_{no,0}}[(S = \zeta)\ \&\ (S \text{ not clean})] \Bigg|\,.$$

By parts (1) and (2) of the claim, we have that

$$\Pr_{S \sim \mathcal{P}^T_{yes,0}}[(S = \zeta)\ \&\ (S \text{ is clean})] = \Pr_{S \sim \mathcal{P}^T_{no,0}}[(S = \zeta)\ \&\ (S \text{ is clean})]$$

for all ζ. Thus we can reexpress the statistical distance as

$$\frac{1}{2}\sum_{\zeta}\left|\Pr_{S\sim\mathcal{P}^T_{yes,0}}[(S=\zeta)\ \&\ (S\text{ not clean})]-\Pr_{S\sim\mathcal{P}^T_{no,0}}[(S=\zeta)\ \&\ (S\text{ not clean})]\right|.$$

This is at most

$$\frac{1}{2}\Big(\Pr_{S\sim\mathcal{P}^T_{yes,0}}[S\text{ not clean}]+\Pr_{S\sim\mathcal{P}^T_{no,0}}[S\text{ not clean}]\Big)$$

which is at most q^2/m by part (1) of the claim. □

3.3.2 Eliminating foolhardy queries

Let T^1 denote a modified version of algorithm T which works as follows: like T, it starts out by making q draws from the distribution. Let Q be the set of all indices i such that all q draws from the distribution have the i^{th} bit set to 1. We note that for both the yes-distribution and the no-distribution, we have that $[n]\setminus R$ is contained in Q. Moreover, if $i\in[m]$ is such that none of $\{a^i,b^i,c^i\}$ occur among the q draws, then C_i is also contained in Q.

We say that any query string $x\in\{0,1\}^n$ that has $x_j=0$ for some $j\in Q$ is *foolhardy*. After making its q draws from $\mathcal{D}$, algorithm T^1 simulates algorithm T for q black-box queries, except that for any foolhardy query that T makes, T^1 "fakes" the query in the following sense: it does not actually make the query but instead proceeds as T would proceed if it made the query and received the response 0.

Our goal in this subsection is to show that in both the yes- and no- cases, the executions of T and T^1 are statistically close. (Intuitively, this means that we can w.l.o.g. assume that the testing algorithm T does not make any foolhardy queries.) To analyze algorithm T^1 it will be useful to consider some other algorithms that are intermediate between T and T^1, which we now describe.

For each value $1\le k\le q$, let U_k denote the algorithm which works as follows: U_k first makes q draws from the distribution $\mathcal{D}$, then simulates algorithm T for k queries, except that for each of the first $k-1$ queries that T makes, if the query is foolhardy then U_k "fakes" the query as described above. Let U'_k denote the algorithm which works exactly like U_k, except that if the k^{th} query made by U_k is foolhardy then U'_k fakes that query as well. We have the following:

Lemma 3.6. *For all $k\in[q]$, the statistical distance*

$$d_{TV}\big((\mathcal{P}^{U_k}_{yes}\mid\mathcal{P}^{U_k}_{yes,0}\textit{ is clean}),(\mathcal{P}^{U'_k}_{yes}\mid\mathcal{P}^{U'_k}_{yes,0}\textit{ is clean})\big)$$

is at most $2\ell m/n$, and similarly

$$d_{TV}\big((\mathcal{P}^{U_k}_{no}\mid\mathcal{P}^{U_k}_{no,0}\textit{ is clean}),(\mathcal{P}^{U'_k}_{no}\mid\mathcal{P}^{U'_k}_{no,0}\textit{ is clean})\big)$$

is also at most $2\ell m/n$.

Proof. We consider the yes-case; the no-case follows by an essentially identical argument.

The executions of U_k and U'_k are identically distributed unless the k^{th} query string (which we denote z) is foolhardy and the black-box function g has $g(z)=1$. Consequently the variation distance

$$d_{TV}\big((\mathcal{P}^{U_k}_{yes} \mid \mathcal{P}^{U_k}_{yes,0} \text{ is clean}), (\mathcal{P}^{U'_k}_{yes} \mid \mathcal{P}^{U'_k}_{yes,0} \text{ is clean})\big)$$

is at most

$$\Pr[(z \text{ is foolhardy}) \,\&\, (g(z)=1)] \le \Pr[(g(z)=1) \mid (z \text{ is foolhardy})]\,,$$

where the probabilities are taken over a random draw of $(g,\mathcal{D}_g)$ from **YES** conditioned on $(g,\mathcal{D}_g)$ being consistent with the q draws from the distribution and with the first $k-1$ queries, and with the q draws from the distribution being clean.

Since the first $k-1$ queries do not involve any variables in Q (because foolhardy queries are faked for the first $k-1$ queries) and our analysis will only concern variables in Q, to analyze this conditional probability it is enough to consider $(g,\mathcal{D}_g)$ drawn from **YES** conditioned on $(g,\mathcal{D}_g)$ being consistent with the q draws from the distribution and with these q draws being clean. Suppose that these draws from the distribution yield r distinct c^i-type points (each with 2ℓ zeros) and $(q-r)$ distinct b^i-type points (each with ℓ zeros). Then after these draws, the algorithm "knows" $(r+q)\ell$ elements of R. Let Z denote the set of these $(r+q)\ell$ elements of R. The set R also contains $2\ell m-(r+q)\ell$ other "unknown" variables from among the $|Q| = n-(r+q)\ell$ variables in $[n]\setminus Z$.

We would like to find the probability, over random $(g,\mathcal{D}_g)$ drawn from **YES** consistent with the draws from the distribution, that $g(z)=1$ given that z is foolhardy. Since z is foolhardy there must be at least one index $j \in [n]\setminus Z$ such that $z_j=0$. So the desired probability is at most the probability that j belongs to R, since if $j \notin R$ the conjunction g_1 will evaluate to 0 on z. For a random $(g,\mathcal{D}_g)$ that is consistent with the draws from the distribution, the remaining $2\ell m-(r+q)\ell$ elements of $R\setminus Z$ are chosen randomly from the $n-(r+q)\ell$ elements of $[n]\setminus Z$. Consequently the probability that j belongs to $R\setminus Z$ is

$$\frac{2\ell m-(r+q)\ell}{n-(r+q)\ell} \le \frac{2\ell m}{n}\,,$$

and the lemma is proved. □

Now a hybrid argument using Lemma 3.6 lets us bound the statistical distance between the executions of T and T^1.

Lemma 3.7. *The statistical distance $d_{TV}(\mathcal{P}^{T^1}_{yes}, \mathcal{P}^{T}_{yes})$ is at most $2\ell mq/n + q^2/m$, and the same bound holds for $d_{TV}(\mathcal{P}^{T^1}_{no}, \mathcal{P}^{T}_{no})$.*

Proof. We prove the yes-case; the no-case follows by an identical argument.

By Claim 3.4, at the cost of q^2/m in $d_{TV}(\mathcal{P}^{T^1}_{yes}, \mathcal{P}^{T}_{yes})$ we may assume that the draws from the distribution are clean. So we henceforth in the proof always condition on the draws from the distribution being clean, and we will bound $d_{TV}(\mathcal{P}^{T^1}_{yes}, \mathcal{P}^{T}_{yes})$ by $2\ell mq/n$ under this conditioning on each argument to d_{TV}.

We use induction on i to show that $d_{TV}(\mathcal{P}^{T}_{yes,i}, \mathcal{P}^{T^1}_{yes,i})$ is at most $2\ell mi/n$ for all i. Once we have this, taking $i=q$ and recalling that $\mathcal{P}^{T}_{yes,q} = \mathcal{P}^{T}_{yes}$ and $\mathcal{P}^{T^1}_{yes,q} = \mathcal{P}^{T^1}_{yes}$ gives the desired bound.

The base case $i=0$ is clear since in this case no black-box queries are made by either T or T^1.

For the induction step we assume that $d_{TV}(\mathcal{P}^{T}_{yes,i}, \mathcal{P}^{T^1}_{yes,i}) \le 2\ell mi/n$, and we will show that

$$d_{TV}(\mathcal{P}^{T}_{yes,i+1}, \mathcal{P}^{T^1}_{yes,i+1}) \le \frac{2\ell m(i+1)}{n}\,.$$

We first note that the random variables $\mathcal{P}^{T^1}_{yes,i+1}$ and $\mathcal{P}^{U'_{i+1}}_{yes}$ are identically distributed, i. e., they have statistical distance zero. Lemma 3.6 now implies that $d_{TV}(\mathcal{P}^{T^1}_{yes,i+1}, \mathcal{P}^{U_{i+1}}_{yes})$ is at most $2\ell m/n$. Since

$$d_{TV}(\mathcal{P}^{T^1}_{yes,i+1}, \mathcal{P}^{T}_{yes,i+1}) \le d_{TV}(\mathcal{P}^{T^1}_{yes,i+1}, \mathcal{P}^{U_{i+1}}_{yes}) + d_{TV}(\mathcal{P}^{U_{i+1}}_{yes}, \mathcal{P}^{T}_{yes,i+1})\,,$$

it is enough to bound $d_{TV}(\mathcal{P}^{U_{i+1}}_{yes}, \mathcal{P}^{T}_{yes,i+1})$ by $2\ell mi/n$. But since the first $q+i$ elements of the sequence $\mathcal{P}^{U_{i+1}}_{yes}$ are distributed according to $\mathcal{P}^{T^1}_{i}$ (and the last element is obtained by performing the $(i+1)$-st query of T), we have that $d_{TV}(\mathcal{P}^{U_{i+1}}_{yes}, \mathcal{P}^{T}_{yes,i+1}) = d_{TV}(\mathcal{P}^{T^1}_{yes,i}, \mathcal{P}^{T}_{yes,i})$, and by the induction hypothesis this is at most $2\ell mi/n$. This concludes the proof. □

3.3.3 Bounding the probability of finding a witness

Let T^2 denote an algorithm that is a variant of T^1, modified as follows. T^2 simulates T^1 except that T^2 does not actually make queries on non-foolhardy strings. Instead T^2 simulates the answers to those queries "in the obvious way" that they should be answered if the target function were a yes-function and hence all of the draws from $\mathcal{D}$ that yielded strings with ℓ zeros were in fact b^i-type points. More precisely, assume that there are r distinct c^i-type points in the initial sequence of q draws from the distribution. Since for each c^i-type point the algorithm is given $\alpha(i)$, the algorithm "knows" r variables $x_{\alpha(i)}$ that are in the conjunction. The algorithm T^2 simulates an answer to a non-foolhardy query $x \in \{0,1\}^n$ with 0 if any of the r $x_{\alpha(i)}$ variables are set to 0 in x, and with 1 otherwise. Note that consequently T^2 does not actually make any black-box queries at all.

In this subsection we will show that in both the yes- and no- cases, the executions of T^1 and T^2 are statistically close; once we have this it is not difficult to complete the proof of Theorem 3.1. In the yes-case these distributions are in fact identical (Lemma 3.8), but in the no-case these distributions are not identical; we will argue that they are close using properties of the function f' from Section 3.1.3.

We first address the easier yes-case:

Lemma 3.8. *The statistical distance $d_{TV}(\mathcal{P}^{T^1}_{yes}, \mathcal{P}^{T^2}_{yes})$ is zero.*

Proof. We argue for every query, the two algorithms T^1 and T^2 receive (or simulate) the same answer. Fix any $1 \le i \le q$ and let z denote the i^{th} query made by T.

If z is a foolhardy query then both T^1 and T^2 simulate an answer of 0. So suppose that z is not a foolhardy query. Then any 0's that z contains must be in positions from points that were sampled in the first stage. Consequently the only variables that can be set to 0 that are in the conjunction g are the $x_{\alpha(i)}$ variables from the C_i sets corresponding to the c^i points in the draws. All the other variables that were assigned value 0 in some string that was drawn from the distribution are not in the conjunction, so setting them to 0 or 1 will not affect the value of $g(z)$. Therefore, the value $g(z)$ that T^1 receives in response to the query string z is 0 if any of the $x_{\alpha(i)}$ variables are set to 0 in z, and is 1 otherwise. This is exactly how T^2 simulates the answer to a non-foolhardy query string z as well. □

To handle the no-case, we introduce the notion of a "witness" that the black-box function is a no-function.

Definition 3.9. We say that a knowledge sequence contains a *witness* for $(f, \mathcal{D}_f)$ if elements $q+1, q+2, \ldots$ of the sequence (i. e., the black-box queries) contain either of the following:

1. A point $z \in \{0,1\}^n$ such that for some $1 \le i \le m$ for which a^i was sampled in the first q draws, the bit $z_{\alpha(i)}$ is 0 and fewer than s of the elements $j \in A_i$ have $z_j = 0$. We refer to such a point as an *a-witness for block i.*
2. A point $z \in \{0,1\}^n$ such that for some $1 \le i \le m$ for which c^i was sampled in the first q draws, z is i-special. We refer to such a point as a *c-witness for block i.*

The following lemma implies that it is enough to bound the probability that $\mathcal{P}^T_{no}$ contains a witness:

Lemma 3.10. *The statistical distance*

$$d_{TV}\big((\mathcal{P}^{T^1}_{no} \mid \mathcal{P}^{T^1}_{no} \textit{ does not contain a witness and } \mathcal{P}^{T^1}_{no,0} \textit{ is clean}),\\ (\mathcal{P}^{T^2}_{no} \mid \mathcal{P}^{T^2}_{no} \textit{ does not contain a witness and } \mathcal{P}^{T^2}_{no,0} \textit{ is clean})\big)$$

is zero.

Proof. Claim 3.4 implies that $(\mathcal{P}^{T^1}_{no,0} \mid \mathcal{P}^{T^1}_{no,0}$ is clean$)$ and $(\mathcal{P}^{T^2}_{no,0} \mid \mathcal{P}^{T^2}_{no,0}$ is clean$)$ are identically distributed. We show that if there is no witness, then for every query the algorithms T^1 and T^2 receive (or simulate) the same answer as each other; this gives the lemma. Fix any $1 \le i \le q$ and let z denote the i^{th} query.

If z is a foolhardy query, then both T^1 and T^2 simulate an answer to z with 0. So suppose that z is not a foolhardy query and not a witness. Then any 0's that z contains must be in positions from points that were sampled in the first stage.

First suppose that one of the $x_{\alpha(i)}$ variables from some c^i that was sampled is set to 0 in z. Since z is not a witness, either z has fewer than s zeros from A_i or some variable from B_i is set to 0 in z. So in this case we have $f(z) = g_2(z) = 0$.

Now suppose that none of the $x_{\alpha(i)}$ variables from the c^i's that were sampled are set to 0 in z. If no variable $x_{\alpha(i)}$ from any a^i that was sampled is set to 0 in z, then clearly $f(z) = g(z) = 1$. If any variable $x_{\alpha(i)}$ from an a^i that was sampled is set to 0 in z, then since z is not a witness there must be at least s elements of A_i set to 0 and every element of B_i set to 1 for each such $x_{\alpha(i)}$. Therefore, $f(z) = 1$.

Thus the value $f(z)$ that T^1 receives on query z is 0 if any of the $x_{\sigma(i)}$ variables from the c^i's that were sampled is set to 0 in z, and the value $f(z)$ is 1 otherwise. This is exactly how T^2 simulates its answer to the query z as well. □

Let us consider a sequence of algorithms that hybridize between T^1 and T^2, similar to the previous section. For each value $1 \le k \le q$, let V_k denote the algorithm which works as follows: V_k first makes q draws from the distribution $\mathcal{D}$, then simulates algorithm T^1 for k queries, except that each of the first

$k-1$ queries is faked (foolhardy queries are faked as described in the previous subsection, and non-foolhardy queries are faked as described at the start of this subsection). Thus algorithm V_k actually makes at most one query to the black-box oracle, the k^{th} one (if this is a foolhardy query then this one is faked as well). Let V'_k denote the algorithm which works exactly like V_k, except that if the k^{th} query made by V_k is non-foolhardy then V'_k fakes that query as well as described at the start of this subsection.

Lemma 3.11. *For each value $1 \leq k \leq q$, the statistical distance*

$$d_{TV}\left((\mathcal{P}_{no}^{V_k} \mid \mathcal{P}_{no,0}^{V_k} \textit{ is clean}), (\mathcal{P}_{no}^{V'_k} \mid \mathcal{P}_{no,0}^{V'_k} \textit{ is clean})\right)$$

is at most $\max\{\frac{qs}{\ell}, \frac{q}{2^s}\}$. *(This equals $\frac{qs}{\ell}$ provided that $s2^s \geq \ell$, as is the case in our parameter settings given by* (3.1)*.)*

Proof. By Lemma 3.10, the executions of V_k and V'_k are identically distributed unless the k^{th} query string (which we denote z) is a witness for $(f, \mathcal{D}_f)$. Since neither V_k nor V'_k makes any black-box query prior to z, the variation distance between $\mathcal{P}_{no}^{V_k}$ and $\mathcal{P}_{no}^{V'_k}$ is at most $\Pr[z$ is a witness$]$, where the probability is taken over a random draw of $(f, \mathcal{D}_f)$ from **NO** conditioned on $(f, \mathcal{D}_f)$ being consistent with the q draws from the distribution and with those first q draws being clean. We bound the probability that z is a witness by considering both possibilities for z (an a-witness or a c-witness) in turn.

- We first bound the probability that z is an a-witness. So fix some $i \in [m]$ and let us suppose that a^i was sampled in the first stage of the algorithm. We will bound the probability that z is an a-witness for block i; once we have done this, a union bound over the (at most q) blocks such that a^i is sampled in the first stage gives a bound on the overall probability that z is an a-witness.

  Fix any possible outcome for z. In order for z to be an a-witness for block i, it must be the case that fewer than s of the ℓ elements in A_i are set to 0 in z, but the bit $z_{\alpha(i)}$ is set to 0. For a random choice of $(f, \mathcal{D}_f)$ as described above, since we are conditioning on the q draws from the distribution being clean, the only information that these q draws reveal about the index $\alpha(i)$ is that it is some member of the set A_i. Consequently for a random $(f, \mathcal{D}_f)$ as described above, each bit in A_i is equally likely to be chosen as $\alpha(i)$, so the probability that $\alpha(i)$ is chosen to be one of the at most s bits in A_i that are set to 0 in z is at most s/ℓ. Consequently the probability that z is an a-witness for block i is at most s/ℓ, and a union bound gives that the overall probability that z is an a-witness is at most qs/ℓ.

- Now we bound the probability that z is a c-witness. Fix some $i \in [m]$ and let us suppose that c^i was sampled in the first stage of the algorithm. We will bound the probability that z is a c-witness for block i and then use a union bound as above.

  Fix any possible outcome for z; let r denote the number of 0's that z has in the bit positions in C_i. In order for z to be a c-witness for block i it must be the case that z is i-special, i. e., $r \geq s$ and all r of these 0's in fact belong to A_i. For a random choice of $(f, \mathcal{D}_f)$ conditioned on being consistent with the q samples from the distribution and with those q samples being clean, the distribution over possible choices of A_i is uniform over all $\binom{2\ell}{\ell}$ possibilities for selecting a size-ℓ subset of C_i.

Consequently the probability that all r 0's belong to A_i is at most

$$\frac{\binom{2\ell-r}{\ell-r}}{\binom{2\ell}{\ell}} = \frac{\ell(\ell-1)\cdots(\ell-r+1)}{2\ell(2\ell-1)\cdots(2\ell-r+1)} < \frac{1}{2^r} \leq \frac{1}{2^s}.$$

So the probability that z is a c-witness for block i is at most $1/2^s$, and by a union bound the overall probability that z is a c-witness is at most $q/2^s$.

So the overall probability that z is a witness is at most $\max\{\frac{qs}{\ell}, \frac{q}{2^s}\}$. Using (3.1) we have that the maximum is qs/ℓ, and the lemma is proved. □

Now similar to Section 3.3.2, a hybrid argument using Lemma 3.11 lets us bound the statistical distance between the executions of T^1 and T^2. The proof of the following lemma is entirely similar to that of Lemma 3.7 so we omit it.

Lemma 3.12. *The statistical distance $d_{TV}(\mathcal{P}_{no}^{T^1}, \mathcal{P}_{no}^{T^2})$ is at most $q^2 s/\ell + q^2/m$.*

3.3.4 Putting the pieces together

At this stage, we have that T^2 is an algorithm that only makes draws from the distribution and makes no queries. It follows that the statistical distance $d_{TV}(\mathcal{P}_{yes}^{T^2}, \mathcal{P}_{no}^{T^2})$ is at most $d_{TV}(\mathcal{P}_{yes,0}^{T}, \mathcal{P}_{no,0}^{T})$. So we can bound $d_{TV}(\mathcal{P}_{yes}^{T}, \mathcal{P}_{no}^{T})$ as follows (we write "d" in place of "d_{TV}" for brevity):

$$\begin{aligned}
d_{TV}(\mathcal{P}_{yes}^{T}, \mathcal{P}_{no}^{T}) &\leq d(\mathcal{P}_{yes}^{T}, \mathcal{P}_{yes}^{T^1}) + d(\mathcal{P}_{yes}^{T^1}, \mathcal{P}_{yes}^{T^2}) + d(\mathcal{P}_{yes}^{T^2}, \mathcal{P}_{no}^{T^2}) + d(\mathcal{P}_{no}^{T^2}, \mathcal{P}_{no}^{T^1}) + d(\mathcal{P}_{no}^{T^1}, \mathcal{P}_{no}^{T}) \\
&\leq d(\mathcal{P}_{yes}^{T}, \mathcal{P}_{yes}^{T^1}) + d(\mathcal{P}_{yes}^{T^1}, \mathcal{P}_{yes}^{T^2}) + d(\mathcal{P}_{yes,0}^{T}, \mathcal{P}_{no,0}^{T}) + d(\mathcal{P}_{no}^{T^2}, \mathcal{P}_{no}^{T^1}) + d(\mathcal{P}_{no}^{T^1}, \mathcal{P}_{no}^{T}) \\
&\leq 4q^2/m + 4q\ell m/n + q^2 s/\ell
\end{aligned}$$

where the final bound follows by combining Corollary 3.5, Lemma 3.7, Lemma 3.8 and Lemma 3.12. Recalling the parameter settings $\ell = n^{2/5}(\log n)^{3/5}$, $m = (n/\log n)^{2/5}$, and $s = \log n$ from (3.1) and the fact that $q = (1/20)(n/\log n)^{1/5}$, this bound is less than $1/4$. This concludes the proof of Theorem 3.1.

4 A lower bound for linear threshold functions

The basic approach is similar to that of Section 3, and indeed several ingredients from the earlier proof can be directly reused; we focus our discussion on the points where the approaches differ. We shall prove the following:

Theorem 4.1. *Let $q = (1/20)(n/\log n)^{1/5}$. There exist two distributions,* **YES** *and* **NO**, *over pairs $(h, \mathcal{D})$ where $h : \{0,1\}^n \to \{0,1\}$ is a Boolean function and $\mathcal{D}$ is a distribution over the domain $\{0,1\}^n$ that have the following properties:*

1. *For every pair $(g, \mathcal{D}_g)$ in the support of* **YES**, *the function g is a linear threshold function.*

2. *For every pair $(f, \mathcal{D}_f)$ in the support of* **NO**, *the function f is $1/4$-far from* LTF *with respect to $\mathcal{D}_f$.*

Moreover, for any probabilistic oracle algorithm T that, given a pair $(h,\mathcal{D})$, makes at most q black-box queries to h and samples $\mathcal{D}$ at most q times, we have

$$\left|\Pr_{(g,\mathcal{D}_g)\sim\mathbf{YES}}[T^{g,\mathcal{D}_g}=Accept]-\Pr_{(f,\mathcal{D}_f)\sim\mathbf{NO}}[T^{f,\mathcal{D}_f}=Accept]\right|\leq\frac{1}{4}.$$

Since any distribution-free tester for LTF that is run with distance parameter $\varepsilon = 1/4$ must accept a random pair $(g,\mathcal{D}_g)$ drawn from **YES** with probability at least $2/3$, and must accept a random pair $(f,\mathcal{D}_f)$ drawn from **NO** with probability at most $1/3$, we have the following corollary of Theorem 4.1, which gives us our main result:

Corollary 4.2. *Distribution-free testing of linear threshold functions requires $\Omega(\frac{n}{\log n})^{1/5}$ queries.*

4.1 The two distributions for linear threshold functions

The construction from Section 3 is not suited for a lower bound on LTF (observe that for any $(f,\mathcal{D}_f)$ in the support of **NO** the function f is 0-far from the linear threshold function $x_1+\cdots+\cdots x_n \geq n-3\ell/2$ with respect to $\mathcal{D}_f$), so we need a different approach.

In the rest of this section we define two distributions **YES** and **NO** over pairs $(h,\mathcal{D})$ and prove that these distributions have properties (1) and (2) described in Theorem 4.1.

In Section 4.2 we use these distributions to prove a lower bound for LTF.

Before giving the precise construction, here is a very rough first intuition for how it works. Recall that in the earlier construction, we relied on the fact that no monotone conjunction h can satisfy $h(1,0) = h(0,1) = 1$ but $h(0,0) = 0$. For linear threshold functions, we will instead rely on the fact that no linear threshold function can satisfy $h(0,0) = h(1,1) = 0$ but $h(1,0) = h(0,1) = 1$.

4.1.1 The YES distribution

As in Section 3 our constructions are parameterized by values ℓ, m and s that are set according to Equation (3.1) in Section 3. A draw from the distribution **YES** over $(g,\mathcal{D}_g)$ pairs is obtained as follows:

- As before, make a draw from the $\mathcal{H}$ distribution to obtain $(R, A_1, B_1, \ldots, A_m, B_m, \alpha_1, \ldots, \alpha_m)$.
- The distribution $\mathcal{D}_g$ puts $1/4$ weight on the point 1^n, and puts weight $1/(2m)$ on b^i and $1/(4m)$ on c^i for all $i = 1,\ldots,m$.
- The function g is defined as follows:

  $g(x) = 1$ if and only if all three of the following conditions hold:

  1. $x_j = 1$ for all $j \in ([n] \setminus R)$;
  2. $x_j = 1$ for all $j = \alpha(1),\ldots,\alpha(m)$; and
  3. $\sum_{i=1}^{m}\sum_{k\in C_i,k\neq\alpha(i)} x_k \leq m(2\ell-1)-s$.

It is straightforward to show that an equivalent way to define g is as follows: $g(x)$ equals 1 if $u(x) \geq \theta$ and equals 0 if $u(x) < \theta$, where

$$u(x) = 10n^2 \sum_{j\in([n]\setminus R)} x_j + 5n\sum_{i=1}^{m} x_{\alpha(i)} - \sum_{i=1}^{m} \sum_{k\in C_i, k\neq\alpha(i)} x_k, \tag{4.1}$$

$$\theta = 10n^2(n-2\ell m) + 5nm - m(2\ell-1) + s. \tag{4.2}$$

Fix any $(g, \mathcal{D}_g)$ in the support of **YES**. It is clear that g is a linear threshold function. It is straightforward to check that

$$u(1^n) = 10n^2(n-2\ell m) + 5nm - m(2\ell-1),$$
$$u(c^i) = 10n^2(n-2\ell m) + 5n(m-1) - (m-1)(2\ell-1), \text{ and}$$
$$u(b^i) = 10n^2(n-2\ell m) + 5nm - m(2\ell-1) + \ell,$$

and consequently we have $g(1^n) = g(c^i) = 0$, $g(b^i) = 1$.

4.1.2 The NO distribution

A draw from the distribution **NO** of $(f, \mathcal{D}_f)$ pairs is obtained as follows:

- As in the yes-case, make a draw from the $\mathcal{H}$ distribution to obtain

$$(R, A_1, B_1, \ldots, A_m, B_m, \alpha_1, \ldots, \alpha_m).$$

- The distribution $\mathcal{D}_f$ puts weight $1/4$ on 1^n, puts $1/4$ weight uniformly over the m points $c^1, \ldots, c^m$, and puts $1/2$ weight uniformly over the $2m$ points $a^1, b^1, \ldots, a^m, b^m$.

- The function f is defined as follows. Given input x, let $J(x) \subseteq [m]$ be the set of those i such that x is i-special as defined in Section 3.2 (i. e., the i^{th} block of x has no zeros in B^i but has at least s zeros in A_i.)

  $f(x) = 1$ if and only if all three of the following conditions hold:

  1. $x_j = 1$ for all $j \in ([n] \setminus R)$;
  2. For all i such that x is *not* i-special, $x_{\alpha(i)} = 1$; and
  3. $|J(x)|(\ell-1) + \sum_{i\in J(x)} \sum_{k\in A_i} x_k + \sum_{i\in([m]\setminus J(x))} \sum_{k\in C_i, k\neq\alpha(i)} x_k \leq m(2\ell-1) - s$.

  It is straightforward to show that an equivalent way to define f is as follows: $f(x)$ equals 1 if $v(x) \geq \theta$ and equals 0 if $v(x) < \theta$. Here θ is defined as in (4.2) and $v(x)$ is defined as follows:

$$v(x) = 10n^2 \sum_{j\in([n]\setminus R)} x_j + 5n\left(|J(x)| + \sum_{i\in([m]\setminus J(x))} x_{\alpha(i)}\right) - |J(x)|(\ell-1) - \sum_{i\in J(x)} \sum_{k\in A_i} x_k - \sum_{i\in([m]\setminus J(x))} \sum_{k\in C_i, k\neq\alpha(i)} x_k. \tag{4.3}$$

Here is some intuition for the definition of f. Suppose that testing algorithm T manages to query an input string x which has $x_j = 1$ for all $j \in B_i$ but also has at least s bits in A_i set to 0. Then as we will see, it must be the case that the algorithm actually drew the point a^i in its sample from $\mathcal{D}_f$. So in order for T to be "fooled" into thinking that the function is a **YES** function, we want the contribution from the bits of A_i and B_i for this input to "look like" the function is a **YES** function for which the point a^i that was drawn from $\mathcal{D}_f$ is actually a point b^i drawn from $\mathcal{D}_g$. This is the rationale behind the definition of f; instead of computing $u(x)$ and comparing it with θ, we compute $v(x)$, which reverses the role of A_i and B_i bits on those blocks.

It is easy to see that in both the yes-case and the no-case, any black-box query that sets any variable in $[n] \setminus R$ to 0 will give a 0 response. As in the earlier construction, intuitively this will let us assume that any testing algorithm that has obtained strings $z^1, \ldots, z^q$ from the distribution $\mathcal{D}$ never queries any string x that has any bit x_i set to 0 that was set to 1 in all of $z^1, \ldots, z^q$.

Finally, it is easy to check that for any $(f, \mathcal{D}_f)$ drawn from **NO**, we have

$$\begin{aligned} v(1^n) &= 10n^2(n-2\ell m) + 5nm - m(2\ell-1)\,, \\ v(c^i) &= 10n^2(n-2\ell m) + 5n(m-1) - (m-1)(2\ell-1)\,, \text{ and} \\ v(a^i) = v(b^i) &= 10n^2(n-2\ell m) + 5nm - m(2\ell-1) + \ell\,. \end{aligned}$$

(Note that these values on 1^n, c^i and b^i are the same that the corresponding functions $u(x)$ would take in the yes-case.) Thus we have $f(1^n) = f(c^i) = 0$ and $f(a^i) = f(b^i) = 1$ for each $i = 1, \ldots, m$. It is easy to see that any linear threshold function must disagree with f on at least one of the four points $1^n, a^i, b^i, c^i$ for each i. Consequently f is at least $1/4$-far from any linear threshold function with respect to $\mathcal{D}_f$.

Thus we have established Properties (1) and (2) as stated in Theorem 4.1.

4.2 Proof of Theorem 4.1

As in Section 3.3, let T be any fixed oracle algorithm that makes exactly q draws from the distribution and then makes exactly q black-box queries. All notation such as $\mathcal{P}^T_{no,0}$ and the like is defined analogously to the definitions given in Section 3.3. We again assume that T is given "extra information" as described earlier when it draws c^i-type examples from the distribution (so the testing algorithm is assumed to actually receive triples instead of pairs, as in Section 3.3). Our definition of a knowledge sequence and of a "clean" sequence of draws are the same as before.

The following easy claim is an analogue of Claim 3.4:

Claim 4.3. *We have* $\Pr[\mathcal{P}^T_{yes,0}$ *is clean*$] = \Pr[\mathcal{P}^T_{no,0}$ *is clean*$] \geq 1 - q^2/m$. *Furthermore, the conditional random variables* $(\mathcal{P}^T_{yes,0} \mid \mathcal{P}^T_{yes,0}$ *is clean) and* $(\mathcal{P}^T_{no,0} \mid \mathcal{P}^T_{no,0}$ *is clean) are identically distributed.*

Proof. We first show that $\Pr[\mathcal{P}^T_{yes,0}$ is clean$] = \Pr[\mathcal{P}^T_{no,0}$ is clean$] \geq 1 - q^2/m$. Fix any $(g, \mathcal{D}_g)$ in the support of **YES**, and consider the outcomes of $\mathcal{P}^T_{yes,0}$ corresponding to this $(g, \mathcal{D}_g)$ being drawn from **YES**. Since each independent draw from $\mathcal{D}_g$ hits each block $1, \ldots, m$ with probability $3/4m$, the probability that $\mathcal{P}^T_{yes,0}$ is clean is

$$\prod_{i=1}^{q} \left(1 - \frac{3(i-1)}{4m}\right) \geq 1 - \frac{3q^2}{4m} \geq 1 - \frac{q^2}{m}\,.$$

The same argument shows that $\Pr[\mathcal{P}^T_{no,0}$ is clean$]$ also equals $\prod_{i=1}^{q}\left(1-\frac{3(i-1)}{4m}\right)$.

Now we show that the conditional random variables are identically distributed. It is not difficult to see that for any $0 \le j \le q-1$, given any particular length-j prefix of $\mathcal{P}^T_{yes,0}$, conditioned on $\mathcal{P}^T_{yes,0}$ being clean, the $(j+1)$-st element of $\mathcal{P}^T_{yes,0}$ has

- a $1/4$ chance of being the triple $(1^n, 0, 0)$;
- a $1/2$ chance of being a triple $(x, 1, 0)$ where $x \in \{0,1\}^n$ has ℓ zeros and the locations of the ℓ zeros are selected uniformly at random (without replacement) from the set of those bit positions that had value 1 in all j of the previous draws;
- a $1/4$ chance of being a triple $(x, 0, \alpha)$ where x has 2ℓ zeros, the locations of the 2ℓ zeros are selected uniformly at random (without replacement) from the same set of bit positions described above, and α is an index drawn uniformly at random from the indices of the 2ℓ zeros in x.

It is also not difficult to see that given any particular length-j prefix of $\mathcal{P}^T_{no,0}$, conditioned on $\mathcal{P}^T_{no,0}$ being clean, the $(j+1)$-st element of $\mathcal{P}^T_{no,0}$ is distributed in the exact same way. This proves the lemma. □

The proof of the following corollary is identical to the proof of Corollary 3.5:

Corollary 4.4. *The statistical distance $d_{TV}(\mathcal{P}^T_{yes,0}, \mathcal{P}^T_{no,0})$ is at most q^2/m.*

Foolhardy queries can be handled just as before. Let the algorithms T^1, U_k and U'_k be defined precisely as in Section 3.3.2. The arguments of Subsection 3.3.2 immediately yield:

Lemma 4.5. *For all $k \in [q]$, the statistical distance*

$$d_{TV}\big((\mathcal{P}^{U_k}_{yes} \mid \mathcal{P}^{U_k}_{yes,0} \textit{ is clean}), (\mathcal{P}^{U'_k}_{yes} \mid \mathcal{P}^{U'_k}_{yes,0} \textit{ is clean})\big)$$

is at most $2\ell m/n$, and similarly $d_{TV}\big((\mathcal{P}^{U_k}_{no} \mid \mathcal{P}^{U_k}_{no,0} \textit{ is clean}), (\mathcal{P}^{U'_k}_{no} \mid \mathcal{P}^{U'_k}_{no,0} \textit{ is clean})\big)$ is also at most $2\ell m/n$.

Lemma 4.6. *The statistical distance $d_{TV}(\mathcal{P}^{T^1}_{yes}, \mathcal{P}^T_{yes})$ is at most $2\ell mq/n + q^2/m$, and the same bound holds for $d_{TV}(\mathcal{P}^{T^1}_{no}, \mathcal{P}^T_{no})$.*

As we now describe, the details of how non-foolhardy queries are handled are different from Section 3.3.3.

Let T^2 denote an algorithm that is a variant of T^1, modified as follows. T^2 simulates T^1 except that T^2 does not actually make queries on non-foolhardy strings; instead T^2 simulates the answers to those queries "in the obvious way" that they should be answered if the target function were a yes-function and hence all of the draws from $\mathcal{D}$ that yielded strings with ℓ zeros were in fact b^i-type points. More precisely, assume that there are r distinct c^i-type points in the initial sequence of q draws from the distribution. Since for each c^i-type point the algorithm is given $\alpha(i)$, the algorithm "knows" r variables $x_{\alpha(i)}$. The algorithm T^2 simulates an answer to a non-foolhardy query $z \in \{0,1\}^n$ as follows: T^2 computes

$$u'(z) = 10n^2(n-2\ell m) + 5n(m - |I'|) - m(2\ell - 1) + |K \setminus I'|$$

where:

- K is the set of all variables x_i set to 0 in z, and
- I' is the set of "known" $x_{\alpha(i)}$ variables that are set to 0 in z

and answers 1 if $u'(z) \geq \theta$, and answers 0 otherwise.

Lemma 4.7. *The statistical distance $d_{TV}(\mathcal{P}^{T^1}_{yes}, \mathcal{P}^{T^2}_{yes})$ is zero.*

Proof. We argue that T^1 and T^2 answer all queries in exactly the same way. Fix any $1 \leq i \leq q$ and let z denote the i^{th} query made by T.

If z is a foolhardy query then both T^1 and T^2 answer z with 0. So suppose that z is not a foolhardy query. By inspection of (4.1), we can reexpress $u(z)$ as

$$u(z) = 10n^2(n - 2\ell m) + 5n(m - |I|) - m(2\ell - 1) + |K \setminus I|$$

where:

- K is the set of all variables x_i set to 0 in z, and
- I is the set of $x_{\alpha(i)}$ variables that are set to 0 in z

and $g(z)$ equals 1 if $u(z) \geq \theta$ and equals 0 if $u(z) < \theta$.

Since z is not foolhardy, the only zeros in z must be in positions from points that were sampled in the first stage. Consequently the only $x_{\alpha(i)}$ variables in I are the $x_{\alpha(i)}$ variables set to 0 in z from the C_i sets corresponding to the c^i points in the draws (these are the "known" $x_{\alpha(i)}$ variables). So $I = I'$ and $K \setminus I = K \setminus I'$.

Therefore, $u(z) = u'(z)$ and hence T^1's response is 0 if $u'(z) < \theta$, and is 1 otherwise. This is exactly how T^2 answers non-foolhardy queries as well. □

We define witnesses in exactly the same way as before:

Definition 4.8. We say that a knowledge sequence contains a *witness* for $(f, \mathcal{D}_f)$ if elements $q+1, \ldots$ of the sequence (the black-box queries) contain either of the following:

1. A point $z \in \{0,1\}^n$ such that for some $1 \leq i \leq m$ for which a^i was sampled in the first q draws, the bit $z_{\alpha(i)}$ is 0 but fewer than s of the elements $j \in A_i$ have $z_j = 0$. We refer to such a point as an *a-witness for block i.*
2. A point $z \in \{0,1\}^n$ such that for some $1 \leq i \leq m$ for which c^i was sampled in the first q draws, z is i-special. We refer to such a point as a *c-witness for block i.*

The following lemma is analogous to Lemma 3.10; it makes essential use of the way our no-functions f are defined in Section 4.1.2.

Lemma 4.9. *The statistical distance*

$$\begin{aligned} d_{TV}\big(&(\mathcal{P}^{T^1}_{no} \mid \mathcal{P}^{T^1}_{no} \textit{ does not contain a witness and } \mathcal{P}^{T^1}_{no,0} \textit{ is clean}), \\ &(\mathcal{P}^{T^2}_{no} \mid \mathcal{P}^{T^2}_{no} \textit{ does not contain a witness and } \mathcal{P}^{T^2}_{no,0} \textit{ is clean})\big) \end{aligned}$$

is zero.

Proof. As in the proof of Lemma 3.10, we show that if there is no witness then T^1 and T^2 answer all queries in exactly the same way. Fix any $1 \le i \le q$ and let z denote the i^{th} query.

If z is a foolhardy query then both T^1 and T^2 answer z with 0. So suppose that z is not a foolhardy query and not a witness. By inspection of (4.3), for any non-foolhardy point z we can express $v(z)$ as

$$v(z) = 10n^2(n - 2\ell m) + 5n(m - |L|) - m(2\ell - 1) + |K \setminus L|$$

where

- K is the set of all variables x_i set to 0 in z and
- L is the set of $x_{\alpha(i)}$ variables set to 0 in z such that $i \notin J(z)$ (i. e., z is not i-special).

Recall that:

$$u'(z) = 10n^2(n - 2\ell m) + 5n(m - |I'|) - m(2\ell - 1) + |K \setminus I'|$$

where

- K is the set of all variables x_i set to 0 in z and
- I' is the set of "known" $x_{\alpha(i)}$ variables that are set to 0 in z.

We will show that if z is not a witness and not foolhardy then $L = I'$. This implies that $v(z) = u'(z)$, so T^1 and T^2 will respond to such queries in exactly the same way.

First we show that $I' \subseteq L$. Fix any $x_{\alpha(i)}$ that belongs to I'; such a variable is set to 0 in z and is "known," so c^i must have been sampled in the first stage. Since z is not a witness, z must not be i-special, i. e., $i \notin J(z)$; so $x_{\alpha(i)}$ must belong to L.

Next we show that $L \subseteq I'$. Fix any $x_{\alpha(i)}$ that belongs to L; such a variable is set to 0 in z and $i \notin J(z)$. This means z is not i-special, so z either has a zero from B_i or fewer than s zeros from A_i. Since the initial draws from $\mathcal{D}$ were clean and z is not foolhardy, it cannot be the case that a^i was drawn in the sample, for if it were drawn then z could not have a zero from B_i and also could not have fewer than s zeros from A_i (since if it had fewer than s zeros from A_i then z would be an a-witness for block i, which contradicts the fact that z is not a witness). Since a^i was not drawn in the sample but the bit $\alpha(i)$ is set to 0 in z and z is not foolhardy, it must be the case that c^i was drawn in the sample. But this means that $x_{\alpha(i)}$ is "known," and consequently $x_{\alpha(i)}$ belongs to I.

So we have shown that $I' = L$, and the lemma is proved. □

From this point on the rest of the argument from Section 3.3 can be used without modification. We define hybrid algorithms V_k, V'_k exactly as in Section 3.3, and the exact proof of Lemma 3.11 now yields:

Lemma 4.10. *For each value $1 \le k \le q$, the statistical distance*

$$d_{TV}\big((\mathcal{P}_{no}^{V_k} \mid \mathcal{P}_{no,0}^{V_k} \text{ is clean}), (\mathcal{P}_{no}^{V'_k} \mid \mathcal{P}_{no,0}^{V'_k} \text{ is clean})\big)$$

is at most $\max\{\frac{qs}{\ell}, \frac{q}{2^s}\} = qs/\ell$.

Exactly as in Section 3.3, we obtain:

Lemma 4.11. *The statistical distance $d_{TV}(\mathcal{P}_{no}^{T^1}, \mathcal{P}_{no}^{T^2})$ is at most $q^2 s/\ell + q^2/m$.*

With all the pieces in place, the arguments from Section 3.3.4 go through unchanged to complete the proof of Theorem 4.1, and we are done.

5 Conclusion

We have given poly(n)-query lower bounds for distribution-free testing of various natural classes of Boolean functions. One obvious goal for future work is to decrease, or, ideally, eliminate the polynomial-factor gap which remains between our lower bounds and the best known upper bounds for these distribution-free testing problems. In particular, it would be quite interesting if improved distribution-free testing algorithms could be obtained which make fewer queries than the currently known testing algorithms (which are straightforward transformations of learning algorithms, as described in the Introduction).

Another natural goal for future work is to extend our techniques to obtain distribution-free lower bounds for richer classes of Boolean functions such as size-s decision trees, s-term DNF formulas, etc. These and several similar classes are known to be testable in the standard uniform distribution model with query complexity that depends on s but is independent of n [6]. In contrast, we suspect that for distribution-free testing, in analogy with the results of this paper the query complexity must depend (perhaps polynomially) on n.

Acknowledgements

We thank the anonymous referees for their detailed comments and suggestions which significantly improved the presentation of the paper.

References

[1] N. AILON AND B. CHAZELLE: Information theory in property testing and monotonicity testing in higher dimension. *Inform. and Comput.*, 204(11):1704–1717, 2006. [doi:10.1016/j.ic.2006.06.001]. 192

[2] N. ALON, T. KAUFMAN, M. KRIVELEVICH, S. LITSYN, AND D. RON: Testing Reed-Muller Codes. *IEEE Trans. Inform. Theory*, 51(11):4032–4039, 2005. [doi:10.1109/TIT.2005.856958]. 192

[3] N. ALON AND A. SHAPIRA: Homomorphisms in Graph Property Testing - A Survey. In *Topics in Discrete Mathematics*, volume 26 of *Algorithms and Combinatorics*, pp. 281–313. Springer, 2006. 191

[4] L. BABAI, L. FORTNOW, AND C. LUND: Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Comput. Complexity*, 1(1):3–40, 1991. [doi:10.1007/BF01200056]. 191

[5] M. BLUM, M. LUBY, AND R. RUBINFELD: Self-testing/correcting with applications to numerical problems. *J. Comput. System Sci.*, 47(3):549–595, 1993. [doi:10.1016/0022-0000(93)90044-W]. 191, 192

[6] I. DIAKONIKOLAS, H. LEE, K. MATULEF, K. ONAK, R. RUBINFELD, R. SERVEDIO, AND A. WAN: Testing for Concise Representations. In *Proc. 48th FOCS*, pp. 549–558. IEEE Comp. Soc. Press, 2007. [FOCS:10.1109/FOCS.2007.32]. 192, 193, 216

[7] Y. DODIS, O. GOLDREICH, E. LEHMAN, S. RASKHODNIKOVA, D. RON, AND A. SAMORODNITSKY: Improved Testing Algorithms for Monotonocity. In *Proc. 3rd Intern. Workshop on Randomization and Approximation Techniques in Comp. Sci. (RANDOM'99)*, volume 1671 of *Lecture Notes in Computer Science*, pp. 97–108. Springer, 1999. [doi:10.1007/b72324]. 192, 193

[8] E. FISCHER: The Art of Uninformed Decisions: A Primer to Property Testing. *Bull. Eur. Assoc. Theor. Comput. Sci.*, 75:97–126, 2001. 191

[9] E. FISCHER, G. KINDLER, D. RON, S. SAFRA, AND A. SAMORODNITSKY: Testing juntas. *J. Comput. System Sci.*, 68(4):753–787, 2004. [doi:10.1016/j.jcss.2003.11.004]. 192

[10] E. FISCHER, E. LEHMAN, I. NEWMAN, S. RASKHODNIKOVA, R. RUBINFELD, AND A. SAMORODNITSKY: Monotonicity Testing Over General Poset Domains. In *Proc. 34th STOC*, pp. 474–483, 2002. [STOC:10.1145/509907.509977]. 192, 193

[11] O. GOLDREICH: Combinatorial Property Testing – a survey. In *Randomization Methods in Algorithm Design*, pp. 45–60. AMS-DIMACS, 1998. 191

[12] O. GOLDREICH, S. GOLDWASSER, E. LEHMAN, D. RON, AND A. SAMORDINSKY: Testing Monotonicity. *Combinatorica*, 20(3):301–337, 2000. [doi:10.1007/s004930070011]. 192, 193

[13] O. GOLDREICH, S. GOLDWASSER, AND D. RON: Property testing and its connection to learning and approximation. *J. ACM*, 45:653–750, 1998. [JACM:285055.285060]. 192

[14] S. HALEVY AND E. KUSHILEVITZ: Distribution-Free Connectivity Testing for Sparse Graphs. *Algorithmica*, 51(1):24–48, 2004. [doi:10.1007/s00453-007-9054-1]. 192

[15] S. HALEVY AND E. KUSHILEVITZ: A Lower Bound for Distribution-Free Monotonicity Testing. In *Proc. 9th Intern. Workshop on Randomization and Approximation Techniques in Comp. Sci. (RANDOM'05)*, volume 3624 of *Lecture Notes in Computer Science*, pp. 330–341. Springer, 2005. [doi:10.1007/11538462_28]. 192, 201

[16] S. HALEVY AND E. KUSHILEVITZ: Distribution-Free Property Testing. *SIAM J. Comput.*, 37(4):1107–1138, 2007. [SICOMP:10.1137/050645804]. 192, 193

[17] M. KEARNS AND U. VAZIRANI: *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994. 193

[18] K. MATULEF, R. O'DONNELL, R. RUBINFELD, AND R. SERVEDIO: Testing Halfspaces. In *Proc. 19th Annual ACM-SIAM Symp. Dicrete Algorithms (SODA)*, pp. 256–264. ACM Press, 2009. [ACM:1496770.1496799]. 192, 193

[19] M. PARNAS, D. RON, AND A. SAMORODNITSKY: Testing Basic Boolean Formulae. *SIAM J. Discrete Math.*, 16(1):20–46, 2002. [doi:10.1137/S0895480101407444]. 192, 193, 194, 200

[20] D. RON: Property Testing (A Tutorial). In *Handbook of Randomized Computing*, volume 2, pp. 597–649. Kluwer, 2000. 191

[21] R. RUBINFELD: Sublinear time algorithms. In *Proc. Intern. Congress of Mathematicians*, volume 3, pp. 1095–1110. European Math. Soc., 2006. 191

[22] R. RUBINFELD AND M. SUDAN: Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. [doi:10.1137/S0097539793255151]. 191

[23] G. TURÁN: Lower bounds for PAC learning with queries. In *Proc. 6th Ann. Conf. Comput. Learning Theory*, pp. 384–391. ACM Press, 1993. [ACM:10.1145/168304.168382]. 193

AUTHORS

Dana Glasner
Department of Computer Science
Columbia University, New York, NY
dglasner@cs.columbia.edu

Rocco A. Servedio
Department of Computer Science
Columbia University, New York, NY
rocco@cs.columbia.edu
http://www.cs.columbia.edu/~rocco

ABOUT THE AUTHORS

DANA GLASNER is a Ph. D. candidate in the Department of Computer Science at Columbia University, supervised by Tal Malkin and Rocco Servedio. Her research interests include property testing and cryptography. She grew up in Pennsylvania and enjoys life in New York City.

ROCCO A. SERVEDIO is an associate professor in the Department of Computer Science at Columbia University. He graduated from Harvard University in 2001 where his Ph. D. was supervised by Leslie Valiant. He is interested in computational learning theory, computational complexity, and property testing, with the study of Boolean functions as an underlying theme tying these topics together. He enjoys spending time with his family and hopes to drink a glass of port with Herman Melville in the afterlife.