Adaptivity Helps for Testing Juntas

Rocco A. Servedio[1], Li-Yang Tan[2], and John Wright[3]

1 **Columbia University**
New York, USA
rocco@cs.columbia.edu
2 **Simons Institute, UC Berkeley**
Berkeley, USA
liyang@cs.columbia.edu
3 **Carnegie Mellon University**
Pittsburgh, USA
jswright@cs.cmu.edu

Abstract

We give a new lower bound on the query complexity of any non-adaptive algorithm for testing whether an unknown Boolean function is a k-junta versus ε-far from every k-junta. Our lower bound is that any non-adaptive algorithm must make

$$\Omega\left(\frac{k \log k}{\varepsilon^c \log(\log(k)/\varepsilon^c)}\right)$$

queries for this testing problem, where c is any absolute constant < 1. For suitable values of ε this is asymptotically larger than the $O(k \log k + k/\varepsilon)$ query complexity of the best known adaptive algorithm [9] for testing juntas, and thus the new lower bound shows that adaptive algorithms are more powerful than non-adaptive algorithms for the junta testing problem.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Property testing, juntas, adaptivity

Digital Object Identifier 10.4230/LIPIcs.CCC.2015.264

1 Introduction

As popular and scientific interest in "big data" continues to build, the field of sublinear-time algorithms has received increasing research attention in recent years. The study of *property testing* is an important area within sublinear algorithms. At a high level, property testing algorithms are "ultra-fast" randomized algorithms which aim to (approximately) determine whether an unknown "massive object" has a particular property while inspecting only a tiny (sublinear, or in some cases even constant sized) portion of the object. Testing algorithms have by now been studied for many different types of mathematical objects; see e.g. [38, 39, 28] for some fairly recent surveys and overviews of contemporary property testing research.

In this work we shall consider property testing algorithms for Boolean functions, and in particular we study the question of testing whether an unknown Boolean function is a *k-junta*. Recall that a function f is a k-junta if it has at most k relevant variables, i.e. there exist k distinct indices $i_1, \ldots, i_k$ and a k-variable function $g : \{0,1\}^k \to \{0,1\}$ such that $f(x) = g(x_{i_1}, \ldots, x_{i_k})$ for all $x \in \{0,1\}^n$. A testing algorithm for k-juntas is given as input k and $\varepsilon > 0$, and is provided with black-box oracle access to an unknown and arbitrary $f : \{0,1\}^n \to \{0,1\}$. The algorithm must output "yes" with high probability (say at least $2/3$) if f is a k-junta, and must output "no" with high probability if f disagrees with every

30th Conference on Computational Complexity (CCC'15).
Editor: David Zuckerman; pp. 264–279

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

k-junta on at least an ε fraction of all possible inputs. The main goal in property testing is to obtain algorithms which make as few queries as possible to the unknown black-box function.

We motivate our work by observing that juntas are a very basic type of Boolean function whose study intersects many different areas within theoretical computer science. In complexity theory and cryptography, $k = O(1)$-juntas are precisely the Boolean functions computed by NC^0 circuits. Juntas arise naturally in settings where a small (unknown) set of features determines the label of a high-dimensional data point, and hence many researchers in learning theory have studied juntas across a wide range of different learning models, see e.g. [15, 22, 16, 30, 3, 37, 4, 2, 25, 42, 21]. Finally, the problem of testing whether an unknown Boolean function is a k-junta is one of the most thoroughly studied questions in Boolean function property testing. We briefly survey relevant previous work on testing juntas in the following subsection.

1.1 Prior work on testing juntas

Fischer et al. [26] were the first to explicitly consider the junta testing problem. Their influential paper gave several algorithms for testing k-juntas, the most efficient of which is a non-adaptive tester that makes $O(k^2(\log k)^2/\varepsilon)$ queries. This was improved by Blais [8] who gave a non-adaptive testing algorithm that uses only $O(k^{3/2}(\log k)^3/\varepsilon)$ queries; this result is still the most efficient known non-adaptive junta tester. Soon thereafter Blais [9] gave an *adaptive* junta testing algorithm that uses only $O(k \log k + k/\varepsilon)$ queries, which remains the most efficient known junta testing algorithm to date.

We note that ideas and and techniques from these junta testing algorithms have played an important role in a broad range of algorithmic results for other Boolean function property testing problems. These include efficient algorithms for testing various classes of functions, such as s-term DNF formulas, small Boolean circuits, and sparse $GF(2)$ polynomials, that are close to juntas but not actually juntas themselves (see e.g. [23, 29, 24, 19]), as well as algorithms for testing linear threshold functions [34] (which in general are not close to juntas). Junta testing is also closely related to the problem of Boolean function isomorphism testing, see e.g. [13, 14, 18, 1].

Lower bounds for testing k-juntas have also been intensively studied. The original [26] paper gave an $\Omega(\sqrt{k}/\log k)$ lower bound for nonadaptive algorithms that test whether an unknown function is a k-junta versus constant-far from every k-junta. Chockler and Gutfreund [20] simplified, strengthened and extended this lower bound by proving that even *adaptive* testers require $\Omega(k)$ queries to distinguish k-juntas from random functions on $k+1$ variables, which are easily seen to be constant-far from k-juntas. (We describe the construction and sketch the [20] argument in Section 1.3 below). Blais [8] was the first to give a lower bound that involves the distance parameter ε; he showed that for $\varepsilon \geq k/2^k$, any non-adaptive algorithm for ε-testing k-juntas must make $\Omega\left(\frac{k}{\varepsilon \log(k/\varepsilon)}\right)$ queries.

In recent years numerous other works have given junta testing lower bounds. In [11] Blais, Brody and Matulef established a connection between lower bounds in communication complexity and property testing lower bounds, and used this connection (together with known lower bounds on the communication complexity of the size-k set disjointness problem) to give a different proof of an $\Omega(k)$ lower bound for adaptively testing whether a function is a k-junta versus constant-far from every k-junta. More recently, Blais, Brody and Ghazi [10] gave new bounds on the communication complexity of the Hamming distance function, and used these bounds to give an alternate proof of the $\Omega(k)$ lower bound for adaptive junta testing algorithms via the [11] connection. Blais and Kane [12] studied the problem of

testing whether an n-variable Boolean function is a size-k parity function (as noted in [12], lower bounds for this problem give lower bounds for testing juntas), and via a geometric and Fourier-based analysis gave a $k - o(k)$ lower bound for adaptive algorithms and a $2k - O(1)$ lower bound for non-adaptive algorithms, again for ε constant. Buhrman et al. [17] combined the communication complexity based approach of [11] with an $\Omega(k \log k)$ lower bound for the one-way communication complexity of k-disjointness to obtain an $\Omega(k \log k)$ lower bound (for constant ε) for testing whether a function f is a size-k parity, and hence for testing whether f is a k-junta.

1.2 Our main result: Adaptivity helps for testing juntas

While the junta testing problem has been intensively studied, the results described above still leave a gap between the query complexity of the best *adaptive* algorithm [9] and the strongest known lower bounds for *non-adaptive* junta testing. The lower bounds of $\Omega\left(\frac{k}{\varepsilon \log(k/\varepsilon)}\right)$ from [8] and $\Omega(k \log k)$ (for ε constant) from [17] are incomparable, but neither of them is strong enough, for any setting of ε, to exceed the $O(k \log k + k/\varepsilon)$ upper bound from [9]. In [8] Blais asked as an open question "*Is there a gap between the query complexity of adaptive and non-adaptive algorithms for testing juntas?*" This question was reiterated in a 2010 survey article on testing juntas, in which Blais explicitly asked "*Does adaptivity help when testing k-juntas?*", referring to this as a "basic problem" [7].

Our main contribution in the present work is to give a better lower bound on non-adaptive junta testing algorithms which implies that the answer to the above questions is "yes." We prove the following:

▶ **Theorem 1.1.** *Let $\mathcal{A}$ be any non-adaptive algorithm which tests whether an unknown black-box $f : \{0,1\}^n \to \{0,1\}$ is a k-junta versus ε-far from every k-junta. Then for all ε satisfying $k^{-o_k(1)} \leq \varepsilon \leq o_k(1)$, algorithm $\mathcal{A}$ must make at least*

$$q = \frac{Ck \log k}{\varepsilon^c \log(\log(k)/\varepsilon^c)} \tag{1}$$

queries, where c is any absolute constant < 1 and $C > 0$ is an absolute constant.

For suitable choices of ε, such as $\varepsilon = 1/(\log k)$, the lower bound of Theorem 1.1 is asymptotically larger than the $O(k \log k + k/\varepsilon)$ upper bound of the [9] adaptive algorithm. Thus, together with the [9] upper bound, our lower bound gives an affirmative answer to the question posed in [8, 7]: adaptivity helps for testing k-juntas.[1]

It is interesting that while all of the recent junta testing lower bounds [11, 10, 17] employ the connection with communication complexity lower bounds that was established in [11], our proof of Theorem 1.1 does not follow this approach. Instead, we give a proof using Yao's classic minimax principle; however, our argument is somewhat involved, employing a new Boolean isoperimetric inequality and a very delicate application of a variant of McDiarmid's "method of bounded differences" that allows for a (low-probability) bad event. In the rest of this section we motivate and explain our approach at a high level before giving the full proof in the subsequent sections.

[1] We note in this context that several other natural Boolean function classes are known to exhibit a gap between the query complexity of adaptive versus non-adaptive testing algorithms. These include the class of signed majority functions [35, 40] and the class of read-once width-two OBDD [41]. In all three cases the adaptive tester which beats the best possible non-adaptive tester may be viewed as performing some sort of binary search.

1.3 The idea underlying our proof

Our approach is inspired by the lower bound of Chockler and Gutfreund [20] for adaptive algorithms, so we begin by briefly recalling their construction and analysis. Chockler and Gutfreund define two distributions $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$ over $(k+1)$-variable functions. A random $\boldsymbol{f}_{yes} \sim \mathcal{D}_{yes}$ is drawn by first choosing a random coordinate $\boldsymbol{i} \in [k+1]$ to be the irrelevant variable, and then choosing a random k-junta over the other k variables from $x_1, \ldots, x_{k+1}$. A random $\boldsymbol{f}_{no} \sim \mathcal{D}_{no}$ is drawn by choosing a random $(k+1)$-junta. Clearly every f in the support of $\mathcal{D}_{yes}$ is a k-junta, and it is easy to show (for k larger than an absolute constant) that almost every function in the support of $\mathcal{D}_{no}$ is constant-far from every k-junta.

Chockler and Gutfreund argue that any $k/6$-query *adaptive* algorithm $\mathcal{A}$ must have

$$\left| \mathop{\mathbf{Pr}}_{\boldsymbol{f}_{yes} \sim \mathcal{D}_{yes}} [\mathcal{A} \text{ accepts } \boldsymbol{f}_{yes}] - \mathop{\mathbf{Pr}}_{\boldsymbol{f}_{no} \sim \mathcal{D}_{no}} [\mathcal{A} \text{ accepts } \boldsymbol{f}_{no}] \right| \leq \frac{1}{6},$$

which gives their $\Omega(k)$ lower bound for adaptive algorithms. Their analysis shows that the only way an algorithm can get statistical evidence that the black-box f is a yes-function rather than a no-function is by querying a pair of inputs $x, y \in \{0,1\}^{k+1}$ that differ in precisely the coordinate $\boldsymbol{i} \in [k+1]$ that was chosen to be irrelevant in the selection of $\boldsymbol{f}_{yes} \sim \mathcal{D}_{yes}$ (they refer to such a pair of Hamming neighbors $x, x^{\oplus i}$ in $\{0,1\}^{k+1}$ as an *i-twin*). While we do not repeat their analysis here, for intution we observe that if x, y form a j-twin for $j \neq i$ then for both a random yes-function and a random no-function $f(x) = f(y)$ with probability exactly $1/2$, while if x, y form an i-twin then $f(x) = f(y)$ for a random yes-function with probability 1 while $f(x) = f(y)$ with probability $1/2$ for a random no-function. Since a set of t queries can contain i-twins for at most $t-1$ distinct coordinates, the $\Omega(k)$ lower bound follows by a "needle in a haystack" argument.

The starting point of our work is the simple observation that the analysis of the Chockler-Gutfreund construction is tight for adaptive algorithms: there is an adaptive algorithm that can distinguish a random $\boldsymbol{f}_{yes} \sim \mathcal{D}_{yes}$ from a random $\boldsymbol{f}_{no} \sim \mathcal{D}_{no}$ with $O(k)$ queries. This algorithm works as follows: for each successive coordinate $j = 1, \ldots, k+1$, it draws random j-twins until either (a) a j-twin $x, x^{\oplus j}$ is drawn for which $f(x) \neq f(x^{\oplus j})$, or (b) $10 \log(k+1)$ j-twins have been drawn and all had $f(x) = f(x^{\oplus j})$. If (b) holds for any $j \in [k+1]$ then halt and output "k-junta," and if (a) holds for every $j \in [k+1]$ halt and output "not a k-junta." Since the expected number of j-twins drawn for a coordinate $j \neq i$ is 2, a straightforward analysis establishes that this algorithm wvhp makes $O(k)$ queries and outputs the correct answer.

Intuitively, the above-described algorithm is only able to achieve $O(k)$ query complexity (an amortized $O(1)$ queries for each of the $k+1$ coordinates) because it is *adaptive* and hence can stop querying a given coordinate j once it receives a j-twin with $f(x) \neq f(x^{\oplus j})$. Since there are $k+1$ coordinates to consider, it is very likely that for some coordinate $j \neq i$, a collection of $\frac{1}{2} \log k$ randomly selected j-twins will all have $f(x) = f(x^{\oplus j})$ (in fact we expect this to happen for $\approx \sqrt{k}$ different coordinates). Since non-adaptive algorithms cannot "amortize" the coordinates along which they spend their queries, this suggests that (i) any *nonadaptive* algorithm will need to query $\Omega(\log k)$ j-twins for at least $\Omega(k)$ many choices of $j \in [k+1]$, and further raises the possibility that (ii) any non-adaptive algorithm for distinguishing $\mathcal{D}_{yes}$ from $\mathcal{D}_{no}$ may need $\Omega(k \log k)$ queries.

In fact, (i) above is correct but (ii) is not. While indeed a non-adaptive algorithm must "rule out" at least $\Omega(k)$ coordinates as not being irrelevant, and indeed $\Omega(\log k)$ j-twins must be queried to rule out a given coordinate j with confidence $1 - 1/\mathrm{poly}(k)$, it does not follow that $\Omega(k \log k)$ queries are required to rule out all coordinates. This is because a set of q

query points can induce $\omega(q)$ different twins – or, to put it in the more combinatorial terms that we use henceforth in the paper, a subset Q of vertices of the Boolean hypercube can induce $\omega(|Q|)$ hypercube edges.[2] Indeed, as observed by Frankl [27], there is a set Q of only $\Theta(k\frac{\log k}{\log\log k})$ points in $\{0,1\}^{k+1}$ that induces at least $\log(k+1)$ edges along each of the $k+1$ coordinates. This set S is as follows: letting $\ell = \log(2\log(k+1))$ (and assuming that ℓ and $\frac{k+1}{\ell}$ are integers), we partition $[k+1]$ into sets $A_1, \ldots, A_{(k+1)/\ell}$ of equal size ℓ each, and let Q be the union of the $\frac{k+1}{\ell}$ subcubes $C_1, \ldots, C_{(k+1)/\ell}$ where C_i consists of all 2^ℓ strings whose 1-coordinates are all contained in positions in A_i. It is easy to verify that the corresponding non-adaptive algorithm makes $\Theta(k\frac{\log k}{\log\log k})$ queries and successfully distinguishes $\boldsymbol{f}_{yes} \sim \mathcal{D}_{yes}$ from $\boldsymbol{f}_{no} \sim \mathcal{D}_{no}$.

It turns out that this is indeed an optimal query lower bound for non-adaptive algorithms that distinguish $\mathcal{D}_{yes}$ from $\mathcal{D}_{no}$, up to constant factors; this follows as a special case of our main result, taking ε to be constant. Our main result is proved by analyzing an ε-biased generalization of the Chockler-Gutfreund yes- and no-distributions; the distributions we consider are the same ones that Blais uses in [8] to establish his lower bound for non-adaptive algorithms. The analysis of [8] uses the edge-isoperimetric inequality of Harper [31], Bernstein [6], Lindsey [33], and Hart [32] and leads to a lower bound of $\Omega\left(\frac{k}{\varepsilon\log(k/\varepsilon)}\right)$ queries for non-adaptive algorithms. In contrast, we use a different edge-isoperimetric inequality, which may be viewed as an extension of Frankl's Theorem 4 in [27] (see Section 2.2). Our edge-isoperimetric inequality, which we state and prove in Section 2.2, implies that any set of vertices in $\{0,1\}^{k+1}$ that induces $\Omega(\log k)$ edges in each of $\Omega(k)$ distinct coordinates must be of size $\Theta(k\frac{\log k}{\log\log k})$.

Another significant difference between our approach and that of [8] is that while [8] essentially applies the Harper–Bernstein–Lindsey–Hart isoperimetric inequality via a union bound in a fairly straightforward way to obtain the $\Omega\left(\frac{k}{\varepsilon\log(k/\varepsilon)}\right)$ lower bound, our argument yielding a $\Omega\left(\frac{k\log k}{\varepsilon^c\log(\log(k)/\varepsilon^c)}\right)$ lower bound is significantly more involved. (The union bound approach of [8] would cost us at least a $\log k$ factor, which is more than we can afford to separate adaptive versus non-adaptive query complexity.) Instead, we use our edge-isoperimetric inequality in the context of a careful probabilistic analysis (to bound the variation distance between "yes-function" and "no-function" vectors of responses a la Yao's minimax method) which crucially relies on a variant of McDiarmid's "method of bounded differences" in which a low-probability "bad event" may take place [36].

1.4 Preliminaries

All logarithms are base 2 unless otherwise stated. We use boldface (e.g. $\boldsymbol{x}, \boldsymbol{y}$, and $\boldsymbol{f}$) to denote random variables. Given $S \subseteq \{0,1\}^n$, we write G_S to denote the subgraph of the Hamming graph induced by S. That is, $G_S = (S, E_S)$, where $(x,y) \in E_S$ iff $x, y \in S$ and $x = y^{\oplus i}$ (this is the string obtained by flipping y in the i-th coordinate) for some $i \in [n]$; we call such an edge (x,y) an *i-edge induced by S*.

[2] The edge-isoperimetric inequality of Harper [31], Bernstein [6], Lindsey [33], and Hart [32] gives a tight upper bound of $\frac{1}{2}|Q|\log|Q|$ edges. We return to this in Section 2.2 when we state and prove a different edge-isoperimetric inequality that we need for our proof.

2 Proof of Theorem 1.1

2.1 The "yes" and "no" distributions

As discussed in the introduction, we consider the same distributions $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$ that Blais used in [8] to establish his non-adaptive lower bound, which are biased generalizations of the yes- and no-distributions considered by Chockler and Gutfreund in [20]. A draw from $\mathcal{D}_{no}$ is an "ε-biased random $(k+1)$-junta" $\boldsymbol{f}_{no} : \{0,1\}^{k+1} \to \{0,1\}$, one which independently takes value 1 with probability ε on every string in $\{0,1\}^{k+1}$. A random $\boldsymbol{f}_{yes}$ from $\mathcal{D}_{yes}$ is drawn by first choosing a random coordinate $\boldsymbol{i} \in [k+1]$ to be irrelevant, and then choosing a random ε-biased random k-junta over the variables from $\{x_1, \dots, x_{k+1}\} \backslash \{x_{\boldsymbol{i}}\}$. Equivalently, $\mathcal{D}_{yes}$ is the uniform mixture of $\mathcal{D}_{yes}^{(1)}, \dots, \mathcal{D}_{yes}^{(k+1)}$, where a draw $\boldsymbol{f}_{yes}^{(i)}$ from $\mathcal{D}_{yes}^{(i)}$ is the random function $\boldsymbol{f}_{yes}^{(i)}(x) = \boldsymbol{f}_{no}(x^{i \leftarrow 1})$ for all $x \in \{0,1\}^n$, where $\boldsymbol{f}_{no} \sim \mathcal{D}_{no}$ and $x^{i \leftarrow 1}$ denotes the string $x \in \{0,1\}^{k+1}$ with its i-th bit set to 1. We see that $\mathcal{D}_{yes}$ is supported entirely on k-juntas (in particular, $\mathcal{D}_{yes}^{(i)}$ is supported entirely on functions that do not depend on the i-th coordinate), and a straightforward calculation shows $\mathcal{D}_{no}$ is supported almost entirely on functions that are $\Omega(\varepsilon)$-far from being a k-junta:

▶ **Lemma 2.1** (Lemma 4.2 of [8]). *When $6k/2^k < \varepsilon \le 1/2$ and $k \ge 3$, a function $\boldsymbol{f}_{no} : \{0,1\}^{k+1} \to \{0,1\}$ drawn from $\mathcal{D}_{no}$ is $(\varepsilon/6)$-far from being a k-junta with probability at least $11/12$.*

We note that these functions $\boldsymbol{f}_{yes}, \boldsymbol{f}_{no} : \{0,1\}^{k+1} \to \{0,1\}$ can be embedded in the full n-dimensional domain $\{0,1\}^n$ simply by defining $\mathbf{F}_{yes} : \{0,1\}^n \to \{0,1\}$ where $\mathbf{F}_{yes}(x) = \boldsymbol{f}_{yes}(x_{[k+1]})$ for all $x \in \{0,1\}^n$, where $x_{[k+1]}$ denotes the prefix substring $(x_1, \dots, x_{k+1}) \in \{0,1\}^{k+1}$ of x. Likewise, we may extend $\boldsymbol{f}_{no} : \{0,1\}^{k+1} \to \{0,1\}$ to $\mathbf{F}_{no} : \{0,1\}^n \to \{0,1\}$. In the rest of the paper we confine our discussion to the $\boldsymbol{f}_{yes}$ and $\boldsymbol{f}_{no}$ functions over $\{0,1\}^{k+1}$.

Fix any constant $c < 1$. Fix any query set $Q^* = \{v^{(1)}, \dots, v^{(q)}\} \subseteq \{0,1\}^{k+1}$ of cardinality q as specified in Equation (1) (we will specify the absolute constant C in Section 2.2 below). For now, we will let the ordering of the query strings $v^{(1)}, \dots, v^{(q)}$ be arbitrary, though we will later impose a carefully chosen particular ordering (see Proposition 2.13). By a standard application of Yao's minimax principle, to prove Theorem 1.1 it suffices to argue that $d_{\mathrm{TV}}(\boldsymbol{f}_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*)) \le 1/3$, where $\boldsymbol{f}_{yes}(Q^*)$ denotes the random "response vector" $(\boldsymbol{f}_{yes}(v^{(1)}), \dots, \boldsymbol{f}_{yes}(v^{(q)})) \in \{0,1\}^q$, likewise $\boldsymbol{f}_{no}(Q^*) = (\boldsymbol{f}_{no}(v^{(1)}), \dots, \boldsymbol{f}_{no}(v^{(q)})) \in \{0,1\}^q$, and $d_{\mathrm{TV}}(\cdot,\cdot)$ denotes the total variation distance (also known as statistical distance) between its two arguments.

2.2 A useful Boolean isoperimetric inequality

As discussed in the introduction, a key combinatorial lemma in Blais' $\tilde{\Omega}(k/\varepsilon)$ sharpening of the Chockler–Gutfreund $\Omega(k)$ lower bound is the classical edge-isoperimetric inequality of Harper, Bernstein, Lindsey, and Hart, which may be viewed as giving a lower bound on the cardinality of query sets in terms of the number of edges they induce.

▶ **Theorem 2.2** (Harper–Bernstein–Lindsey–Hart). *For all $S \subseteq \{0,1\}^n$, we have $|E_S| \le \frac{1}{2}|S| \log |S|$.*

We will need a variant of this inequality which takes into account the *directions* of the induced edges; in particular, it will be important for us that most directions have "not too few" induced edges in that direction.

▶ **Definition 2.3.** Let $S \subseteq \{0,1\}^n$. We say that S *m-saturates* direction $i \in [n]$ if S induces at least m many i-edges.

Motivated by our earlier discussion in Section 1.3, a "good" query set $Q \subseteq \{0,1\}^{k+1}$ for distinguishing between $\boldsymbol{f}_{yes} \sim \mathcal{D}_{yes}$ and $\boldsymbol{f}_{no} \sim \mathcal{D}_{no}$ is one which m-saturates most of the $k+1$ coordinates for a suitable choice of m (and of course we want Q to achieve this while being as small as possible). What kind of query sets Q are best suited to meet these two objectives? As an easy first observation, let Q_1 be an arbitrary query set such that G_{Q_1} has $q_{1,i}$ edges in each direction i. It is not difficult to show that there exists a query set Q_2, with $|Q_2| = |Q_1|$, such that (i) G_{Q_2} is a connected graph and (ii) G_{Q_2} has $q_{2,i} \geq q_{1,i}$ edges in each direction i. (Repeatedly translate connected components of Q_1 until they "come together" and only a single connected component is present; such translations cannot decrease the number of edges in any direction.)

In fact, a stronger statement than the above is true (and is not difficult to show): the "best" query set Q of a given size is of the form $g^{-1}(1)$ (or $g^{-1}(0)$) for some monotone Boolean function g. This is made precise through the following definition and fact:

▶ **Definition 2.4.** For each $i \in [n]$ the i-th down-shift operator κ_i acts on Boolean functions $g : \{0,1\}^n \to \{0,1\}$ as follows: $(\kappa_i g)(x) = g(x)$ if $g(x) = g(x^{\oplus i})$, and $(\kappa_i g)(x) = 1 - x_i$ otherwise.

▶ **Fact 2.5** (see e.g. [5])**.** *Let $S \subseteq \{0,1\}^n$ and $g : \{0,1\}^n \to \{0,1\}$ be its indicator function. Consider $S_{\text{shift}} := g_{\text{shift}}^{-1}(1) \subseteq \{0,1\}^n$, where $g_{\text{shift}} := \kappa_1 \cdots \kappa_n g$. Then $|S_{\text{shift}}| = |S|$ and S_{shift} is* downward closed*, meaning that for all $v' \preceq v$, if $v \in S_{\text{shift}}$ then $v' \in S_{\text{shift}}$. Furthermore, if G_S has q_i edges in direction i, then $G_{S_{\text{shift}}}$ has $q_{\text{shift},i} \geq q_i$ edges in direction i (hence if S m-saturates a direction i then so does S_{shift}).*

The following isoperimetric bound plays a key role in our arguments; it says that we need "many" vertices to m-saturate a large number of distinct directions.

▶ **Proposition 2.6.** *Let $S \subseteq \{0,1\}^n$ be a set of points that m-saturates at least ℓ directions. Then $|S| \geq \frac{m\ell}{\lfloor \log m + 1 \rfloor} = \Omega\Big(\frac{m\ell}{\log m}\Big)$.*

Proof. Let $\text{height}(S)$ denote the quantity $\max_{v \in S} \|v\|$, where $\|v\| = \sum_{i=1}^n v_i$ is the Hamming weight of $v \in \{0,1\}^n$. By Fact 2.5, we may restrict our attention to sets S that are downward closed. Let S^* be a downward-closed set of minimal size that m-saturates at least ℓ directions, and which has $\text{height}(S^*)$ as small as possible among all such minimal sets; for brevity we write h to denote $\text{height}(S^*)$. Note that we have the relationship

$$m\ell \leq |E_{S^*}| = \sum_{v \in S^*} \|v\| \leq h \cdot |S^*|, \tag{2}$$

and hence to prove a lower bound on the size of S^*, it suffices to show an upper bound on h, the height of S^*. Let v^* be a vertex in S with $\|v^*\| = h$, let $D_{v^*} = \{i \in [n] \colon v^*_i = 1\}$, and consider $S' = S^* \setminus \{v^*\}$. Since S^* is downward closed we have that G_{S^*} has at least 2^{h-1} edges in each direction $i \in D_{v^*}$. Deleting v^* from S^* removes exactly h induced edges, one from each direction $i \in D_{v^*}$, and so by the minimality of S^* it follows that $2^{h-1} - 1 < m$, or equivalently, $h \leq \lfloor \log m + 1 \rfloor$.This, with (2), completes the proof. ◀

▶ Remark. Proposition 2.6 recovers as a special case a classical result of Frankl (Theorem 4 of [27]), proved using the Kruskal–Katona theorem, giving a lower bound of $|S| = \Omega\big(\frac{mn}{\log m}\big)$ on the cardinality of any set $S \subseteq \{0,1\}^n$ which m-saturates all n directions. We note also

that the parameters of Proposition 2.6 are optimal up to a factor of 2. To see this, suppose $t := \log m + 1 \in \mathbb{N}$ and t divides ℓ. Let $A_1, \dots, A_{\ell/t}$ be a partition of $[\ell]$ into disjoint blocks of cardinality t, and for each $i \in [\ell/t]$, let $C_i = \{v \colon v_j = 0 \text{ for all } j \notin A_i\}$ be the t-dimensional subcube over the coordinates in $A_i \subseteq [\ell]$. Then $S := \bigcup C_i$ is a set of cardinality $|S| = (2m\ell/(\log m + 1)) - 1$ which m-saturates the first ℓ directions.

By Proposition 2.6, we can and shall assume that the query set $Q^* \subseteq \{0,1\}^{k+1}$ (which is of size at most $q \leq \frac{Ck \log k}{\varepsilon^c \log((\log k)/\varepsilon^c)}$, recall (1)) $(\log k)/\varepsilon^c$-saturates at most $0.1k$ directions. As noted earlier, by Yao's minimax principle, to prove Theorem 1.1 it remains to argue that $d_{\mathrm{TV}}(\boldsymbol{f}_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*)) \leq 1/3$.

2.3 Conditioning on unsaturated irrelevant coordinates, and bounding total variation by establishing concentration

In analyzing the random variable $\boldsymbol{f}_{yes}(Q^*)$, it will be helpful for us to condition on the event that Q^* only induces "a few" edges in the direction of the irrelevant coordinate $\boldsymbol{i} \in [k+1]$. Formally, let $U \subseteq [k+1]$ denote the directions that are not $((\log k)/\varepsilon^c)$-saturated by Q^*, and recall that $|U| \geq 0.9k$ by our assumption on the cardinality of Q^* along with Proposition 2.6. Let $\mathcal{D}'_{yes}$ denote the uniform mixture of $\mathcal{D}^{(i)}_{yes}$ for all $i \in U$ (i.e. $\mathcal{D}'_{yes}$ is $\mathcal{D}_{yes}$ conditioned on the irrelevant coordinate i being in U), and $\mathcal{D}''_{yes}$ denote the uniform mixture of $\mathcal{D}^{(i)}_{yes}$ for all $i \notin U$. In other words, $\mathcal{D}_{yes}$ is the mixture of $\mathcal{D}'_{yes}$ and $\mathcal{D}''_{yes}$ with mixing weights $1-\delta$ and δ respectively, where $\delta \leq 0.1$. We write $\boldsymbol{f}'_{yes}$ and $\boldsymbol{f}''_{yes}$ to denote draws from $\mathcal{D}'_{yes}$ and $\mathcal{D}''_{yes}$ respectively.

▶ **Lemma 2.7.** $d_{\mathrm{TV}}(\boldsymbol{f}_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*)) \leq d_{\mathrm{TV}}(\boldsymbol{f}'_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*)) + \delta$.

Proof. This holds by noting that $d_{\mathrm{TV}}(\boldsymbol{f}_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*))$ can be expressed as

$$\begin{aligned}
&\frac{1}{2} \sum_{y \in \{0,1\}^q} \left| (1-\delta)\,\mathbf{Pr}[\boldsymbol{f}'_{yes}(Q^*) = y] + \delta\,\mathbf{Pr}[\boldsymbol{f}''_{yes}(Q^*) = y] - \mathbf{Pr}[\boldsymbol{f}_{no}(Q^*) = y] \right| \\
\leq\ &\frac{1}{2} \sum_{y \in \{0,1\}^q} \left| \mathbf{Pr}[\boldsymbol{f}'_{yes}(Q^*) = y] - \mathbf{Pr}[\boldsymbol{f}_{no}(Q^*) = y] \right| \\
&\qquad + \delta\big(\mathbf{Pr}[\boldsymbol{f}'_{yes}(Q^*) = y] + \mathbf{Pr}[\boldsymbol{f}''_{yes}(Q^*) = y]\big) \\
=\ &d_{\mathrm{TV}}(\boldsymbol{f}'_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*)) + \delta.
\end{aligned}$$

◀

And so indeed, Lemma 2.7 reduces the task of proving $d_{\mathrm{TV}}(\boldsymbol{f}_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*)) \leq 1/3$ to that of showing

$$d_{\mathrm{TV}}(\boldsymbol{f}'_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*)) \leq (1/3) - 0.1, \tag{3}$$

which is what we will do. We begin by observing that the distribution of $\boldsymbol{f}_{no}(Q^*)$ is fairly easy to understand: for all $y \in \{0,1\}^q$, we have $\mathbf{Pr}\left[\boldsymbol{f}_{no}(Q^*) = y\right] = \varepsilon^{|y|}(1-\varepsilon)^{q-|y|} := \mathrm{wt}_\varepsilon(y)$. This motivates us to define the function $A : \{0,1\}^{Q^*} \to [0,1]$,

$$A(y) = \mathbf{Pr}\left[\boldsymbol{f}'_{yes}(Q^*) = y\right], \text{ and write } d_{\mathrm{TV}}(\boldsymbol{f}'_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*)) = \sum_{y \in \{0,1\}^{Q^*}} \frac{|A(y) - \mathrm{wt}_\varepsilon(y)|}{2}.$$

For the remainder of this proof, we will write $\boldsymbol{y} = (\boldsymbol{y}_1, \dots, \boldsymbol{y}_q)$ to denote a draw from $\{0,1\}^q_{(\varepsilon)}$, the ε-biased product distribution over $\{0,1\}^q$ where each coordinate is independently 1 with probability ε. It will be convenient to think of $\boldsymbol{y}$ as the values $\boldsymbol{f}_{no} \sim \mathcal{D}_{no}$ takes on the query

points in $Q^* \subseteq \{0,1\}^{k+1}$; in other words, $\boldsymbol{y}$ is distributed identically to $\boldsymbol{f}_{no}(Q^*)$. Writing $\tilde{A}(y) := A(y)/\mathrm{wt}_\varepsilon(y)$, we have

$$d_{\mathrm{TV}}(\boldsymbol{f}'_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*)) = \frac{1}{2} \sum_{y \in \{0,1\}^{Q^*}} \mathrm{wt}_\varepsilon(y) |\tilde{A}(y) - 1| = \frac{1}{2} \mathbf{E}\left[|\tilde{A}(\boldsymbol{y}) - 1|\right].$$

Since $1 = \sum_y A(y) = \sum_y \mathrm{wt}_\varepsilon(y) \cdot \tilde{A}(y) = \mathbf{E}[\tilde{A}(\boldsymbol{y})]$, it suffices for us to argue that the random variable $\tilde{A}(\boldsymbol{y})$ is concentrated around its expectation $\mathbf{E}[\tilde{A}(\boldsymbol{y})] = 1$:

▶ **Proposition 2.8.** $\mathbf{Pr}\left[\tilde{A}(\boldsymbol{y}) \in [0.9, 1.1]\right] \geq 0.9$.

Our claimed bound on total variation distance (3) follows from Proposition 2.8 via the following calculation, where the penultimate inequality uses Proposition 2.8:

$$\begin{aligned} d_{\mathrm{TV}}(\boldsymbol{f}'_{yes}(Q^*), \boldsymbol{f}_{no}(Q^*)) &= \frac{1}{2} \sum_{y \in \{0,1\}^q} |A(y) - \mathrm{wt}_\varepsilon(y)| \\ &= \frac{1}{2}\Big(2 - 2 \sum_{y \in \{0,1\}^q} \min\{A(y), \mathrm{wt}_\varepsilon(y)\}\Big) \\ &\leq 1 - \sum_{\substack{y \in \{0,1\}^q \\ \tilde{A}(y) \in [0.9,1.1]}} \min\{A(y), \mathrm{wt}_\varepsilon(y)\} \\ &\leq 1 - \sum_{\substack{y \in \{0,1\}^q \\ \tilde{A}(y) \in [0.9,1.1]}} 0.9 \cdot \mathrm{wt}_\varepsilon(y) \;\leq\; 1 - (0.9)^2 \;<\; (1/3) - 0.1. \end{aligned}$$

2.4 Proof of Proposition 2.8

We will bound the probability that $\tilde{A}(\boldsymbol{y})$ deviates from its mean using the "method of averaged bounded differences" in which a rare "bad" event is allowed to take place:

▶ **Theorem 2.9** (special case of Theorem 3.7 of [36]). *Let $\tilde{A}$ be a function of $\{0,1\}$-valued random variables $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_q$ such that $\mathbf{E}[\tilde{A}(\boldsymbol{y})]$ is bounded. Let $\mathcal{B} \subseteq \{0,1\}^q$, and suppose that for all $b \in \{0,1\}^q \setminus \mathcal{B}$,*

$$\sum_{j \in [q]} \left(\mathbf{E}\left[\tilde{A}(b_1, \ldots, b_{j-1}, b_j, \boldsymbol{y}_{j+1}, \ldots, \boldsymbol{y}_q) - \tilde{A}(b_1, \ldots, b_{j-1}, \overline{b}_j, \boldsymbol{y}_{j+1}, \ldots, \boldsymbol{y}_q)\right]\right)^2 \leq \Delta. \quad (4)$$

Then for all $t \geq 0$, we have $\mathbf{Pr}\left[|\tilde{A}(\boldsymbol{y}) - \mathbf{E}[\tilde{A}(\boldsymbol{y})]| > t\right] \leq 2\exp\left(-2t^2/\Delta\right) + 2\,\mathbf{Pr}[\boldsymbol{y} \in \mathcal{B}]$.

We introduce some useful notation. Given a labelling $b = (b_1, \ldots, b_q) \in \{0,1\}^q$ of the query strings $v^{(1)}, \ldots, v^{(q)}$ in Q^*, we write $(b_j, \boldsymbol{y}_{j+1})$ to denote the hybrid string $(b_1, \ldots, b_{j-1}, b_j, \boldsymbol{y}_{j+1}, \ldots, \boldsymbol{y}_q)$ and likewise $(\overline{b}_j, \boldsymbol{y}_{j+1})$ to denote $(b_1, \ldots, b_{j-1}, \overline{b}_j, \boldsymbol{y}_{j+1}, \ldots, \boldsymbol{y}_q)$. We also write $\mathrm{diff}(b, j)$ to denote the difference $|\,\mathbf{E}[\tilde{A}(b_j, \boldsymbol{y}_{j+1}) - \tilde{A}(\overline{b}_j, \boldsymbol{y}_{j+1})]|$. This notational convention allows us to express the inequality (4) more succinctly as

$$\sum_{j \in [q]} \mathrm{diff}(b, j)^2 = \sum_{j \in [q]} \left(\mathbf{E}[\tilde{A}(b_j, \boldsymbol{y}_{j+1}) - \tilde{A}(\overline{b}_j, \boldsymbol{y}_{j+1})]\right)^2 \leq \Delta. \quad (5)$$

Furthermore, we write $\#_i 11(b)$ to denote the number of i-edges in G_{Q^*} whose endpoints are both labeled 1 by b, and likewise $\#_i 00(b)$ to denote the number of i-edges in G_{Q^*} whose endpoints are both labeled 0 by b. We write $\#_i 1(b)$ to denote the number of vertices in G_{Q^*}

that are labeled 1 by b and are not incident to an i-edge in G_{Q^*}, and likewise $\#_i 0(b)$. Finally, let $i\text{-bias}_\varepsilon(b)$ denote the quantity $\varepsilon^{-\#_i 11(b)} \cdot (1-\varepsilon)^{-\#_i 00(b)}$.

The following terminology will be useful: We say a labeling $y \in \{0,1\}^q$ of the query strings $v^{(1)}, \dots, v^{(q)} \in Q^*$ is *i-monochromatic* (abbreviated as "i-mono") if every i-edge in G_{Q^*} either has both endpoints labeled 1 by y, or has both endpoints labeled 0 by y. With this notation and terminology in place we may conveniently characterize $\tilde{A}(y)$ as follows:

▶ **Lemma 2.10.** $\tilde{A}(y) = \dfrac{1}{|U|} \displaystyle\sum_{i \in U} \mathbf{1}[y \textit{ is } i\textit{-mono}] \cdot i\text{-bias}_\varepsilon(y).$

Proof of Lemma 2.10. Fix a choice for the irrelevant coordinate $i \in U$. Conditioned on this, the probability that $\boldsymbol{f}'_{yes}(Q^*) = y$ is $\varepsilon^{\#_i 11(y)+\#_i 1(y)} \cdot (1-\varepsilon)^{\#_i 00(y)+\#_i 0(y)}$ if y is i-mono and is 0 otherwise. As $\mathrm{wt}_\varepsilon(y) = \varepsilon^{2\#_i 11(y)+\#_i 1(y)} \cdot (1-\varepsilon)^{2\#_i 00(y)+\#_i 0(y)}$ if y is i-mono, we have that $\tilde{A}(y)$ equals

$$\begin{aligned}
&\frac{1}{|U|} \sum_{i \in U} \mathbf{1}[y \text{ is } i\text{-mono}] \cdot \frac{1}{\mathrm{wt}_\varepsilon(y)} \cdot \mathbf{Pr}\left[\boldsymbol{f}'_{yes}(Q^*) = y \mid \boldsymbol{i} = i\right] \\
&= \frac{1}{|U|} \sum_{i \in U} \mathbf{1}[y \text{ is } i\text{-mono}] \cdot \frac{\varepsilon^{\#_i 11(y)+\#_i 1(y)} \cdot (1-\varepsilon)^{\#_i 00(y)+\#_i 0(y)}}{\varepsilon^{2\#_i 11(y)+\#_i 1(y)} \cdot (1-\varepsilon)^{2\#_i 00(y)+\#_i 0(y)}} \\
&= \frac{1}{|U|} \sum_{i \in U} \mathbf{1}[y \text{ is } i\text{-mono}] \cdot i\text{-bias}_\varepsilon(y).
\end{aligned}$$

◀

By Lemma 2.10, we have that $\mathrm{diff}(b,j)$ is at most $\frac{1}{|U|}$ times

$$\left| \sum_{i \in U} \underbrace{\mathop{\mathbf{E}}_{\boldsymbol{y}} \left[\mathbf{1}[(b_j, \boldsymbol{y}_{j+1}) \text{ is } i\text{-mono}]\, i\text{-bias}_\varepsilon(b_j, \boldsymbol{y}_{j+1}) - \mathbf{1}[(\bar{b}_j, \boldsymbol{y}_{j+1}) \text{ is } i\text{-mono}]\, i\text{-bias}_\varepsilon(\bar{b}_j, \boldsymbol{y}_{j+1}) \right]}_{(*)} \right|. \tag{6}$$

Fix $b \in \{0,1\}^{Q^*}$ and $j \in [q]$. We make a couple of observations about the quantity $(*)$ for a fixed coordinate $i \in U$ which will be useful later.

▶ **Observation 2.11.** *If $v^{(j)}$ is not incident to an i-edge within G_{Q^*}, then $(*) = 0$ pointwise for every possible outcome of $\boldsymbol{y}$.*

This is because the labeling of $v^{(j)}$ has no effect on either the monochromaticity of the i-th direction or the number of monochromatic i-edges, and hence $\mathbf{1}[(b_j, y_{j+1}) \text{ is } i\text{-mono}] = \mathbf{1}[(\bar{b}_j, y_{j+1}) \text{ is } i\text{-mono}]$ and $i\text{-bias}_\varepsilon(b_j, y_{j+1}) = i\text{-bias}_\varepsilon(\bar{b}_j, y_{j+1})$ for every possible outcome y of $\boldsymbol{y}$.

▶ **Observation 2.12.** *If $v^{(j)}$ has an i-edge to $v^{(j')}$ within G_{Q^*} where $j' > j$, then again $(*) = 0$.*

(In the following equations we use the notation $(a_1, a_2, \boldsymbol{y}_{j+2})$ to denote the string $(b_j, \boldsymbol{y}_{j+1})$ except with the j-th bit set to a_1 and the j'-th bit set to a_2.) Observation 2.12 is true because

$$\begin{aligned}
\pm(*) &= \mathop{\mathbf{E}}_{\boldsymbol{y}} \left[\mathbf{1}[(1, \boldsymbol{y}_{j+1}) \text{ is } i\text{-mono}]\, i\text{-bias}_\varepsilon(1, \boldsymbol{y}_{j+1}) - \mathbf{1}[(0, \boldsymbol{y}_{j+1}) \text{ is } i\text{-mono}]\, i\text{-bias}_\varepsilon(0, \boldsymbol{y}_{j+1}) \right] \\
&= \mathop{\mathbf{E}}_{\boldsymbol{y}} \Big[\varepsilon \mathbf{1}[(1,1, \boldsymbol{y}_{j+2}) \text{ is } i\text{-mono}]\, i\text{-bias}_\varepsilon(1,1, \boldsymbol{y}_{j+2}) \\
&\qquad - (1-\varepsilon) \mathbf{1}[(0,0, \boldsymbol{y}_{j+2}) \text{ is } i\text{-mono}]\, i\text{-bias}_\varepsilon(0,0, \boldsymbol{y}_{j+2}) \Big],
\end{aligned}$$

and moreover, $\mathbf{1}[(1,1,y_{j+2}) \text{ is } i\text{-mono}] = \mathbf{1}[(0,0,y_{j+2}) \text{ is } i\text{-mono}]$ and $\varepsilon \cdot i\text{-bias}_\varepsilon(1,1,y_{j+2}) = (1-\varepsilon) \cdot i\text{-bias}_\varepsilon(0,0,y_{j+2})$ for every possible outcome y of $\boldsymbol{y}$.

2.4.1 Choosing an ordering

Given the preceding observations, we may rewrite (6) so that the sum is only over those directions $i \in U$ such that $v^{(j)}$ has an i-edge within G_{Q^*} to some $v^{(j')}$ where $j' < j$. A priori, there is no reason to believe that this rewriting will simplify the sum or significantly reduce the number of summands. However, the next proposition shows that by enforcing an appropriate ordering on the query set $Q^* = \{v^{(1)}, \ldots, v^{(q)}\}$, we can ensure that all but $\lfloor \log(q) \rfloor$ terms will drop out of (6).

▶ **Proposition 2.13.** *For every $S = \{v^{(1)}, \ldots, v^{(q)}\} \subseteq \{0,1\}^n$, there exists an ordering $v^{(1)} \prec v^{(2)} \prec \cdots \prec v^{(q)}$ such that every $v^{(i)}$ has at most $\lfloor \log q \rfloor$ many Hamming neighbors $v^{(j)}$ that precede it in the ordering.*

Proof. We proceed by induction on q, noting that the lemma trivially holds when $q = 1$. For the inductive step, we partition S into S_L and S_R, where

$$S_L = \{v \in S \colon \deg_S(v) \leq \lfloor \log q \rfloor\},$$
$$S_R = \{v \in S \colon \deg_S(v) > \lfloor \log q \rfloor\},$$

and $\deg_S(v)$ denotes the degree of v in G_S. By the edge-isoperimetric inequality (Theorem 2.2), we have that

$$\sum_{v \in S} \deg_S(v) = 2 \cdot |E_S| \leq q \log q,$$

and hence $|S_L| \geq 1$, or equivalently, $|S_R| \leq q - 1$. By our induction hypothesis applied to S_R, there exists an ordering of its vertices so that every vertex has at most $\lfloor \log S_R \rfloor \leq \lfloor \log q \rfloor$ Hamming neighbors that precede it in the ordering. Our ordering of S will be the vertices in S_R listed in this order given by the induction hypothesis, followed by the vertices in S_L listed in an arbitrary order. The proof is complete by recalling that $\deg_S(v) \leq \lfloor \log q \rfloor$ for every $v \in S_L$, and hence every $v \in S_L$ trivially has at most $\lfloor \log q \rfloor$ Hamming neighbors that precede it in the ordering. ◀

We will now assume that $Q^* = \{v^{(1)}, \ldots, v^{(q)}\}$ is sorted in the order given by Proposition 2.13. Since $|Q^*| = q = o(k^{1.1}) \ll k^2$ (recall (1) and the bounds on ε given in the conditions of Theorem 1.1) we have that there are fewer than $2 \log k$ such directions $i \in U$. Let $i^* \in U$ be the direction that maximizes $(*)$ in (6), and so $\mathrm{diff}(b, j)$ is at most (6), which in turn is at most $\frac{2 \log k}{0.9k}$ times

$$\left| \mathop{\mathbf{E}}_{\boldsymbol{y}} \Big[\underbrace{\mathbf{1}[(b_j, \boldsymbol{y}_{j+1}) \ i^*\text{-mono}]\, i^*\text{-bias}_\varepsilon(b_j, \boldsymbol{y}_{j+1})}_{(**)} - \underbrace{\mathbf{1}[(\bar{b}_j, \boldsymbol{y}_{j+1}) \ i^*\text{-mono}]\, i^*\text{-bias}_\varepsilon(\bar{b}_j, \boldsymbol{y}_{j+1})}_{(***)} \Big] \right|.$$

Since $v^{(j)}$ has an i^*-edge within G_{Q^*} to some $v^{(j')}$ where $j' < j$, it follows that either $\mathbf{E}[(**)] = 0$ or $\mathbf{E}[(***)] = 0$ (the former if $b_{j'} \neq b_j$, and the latter if $b_{j'} \neq \bar{b}_j$). We may assume w.l.o.g. that $\mathbf{E}[(***)] = 0$, and so

$$\mathrm{diff}(b, j) \leq \frac{2 \log k}{0.9k} \mathop{\mathbf{E}}_{\boldsymbol{y}} \Big[\mathbf{1}[(b_j, \boldsymbol{y}_{j+1}) \text{ is } i^*\text{-mono}] \cdot i^*\text{-bias}_\varepsilon(b_j, \boldsymbol{y}_{j+1}) \Big].$$

Next, we observe that the expectation above may be rewritten as

$$\begin{aligned} &\mathop{\mathbf{E}}_{\boldsymbol{y}} \Big[\mathbf{1}[(b_j, \boldsymbol{y}_{j+1}) \text{ is } i^*\text{-mono}] \cdot i^*\text{-bias}_\varepsilon(b_j, \boldsymbol{y}_{j+1}) \Big] \\ &= \prod_{i^*\text{-edges } e} \mathop{\mathbf{E}}_{\boldsymbol{y}} \Big[\mathbf{1}[e \text{ is mono w.r.t. } (b_j, \boldsymbol{y}_{j+1})] \cdot i^*\text{-bias}_\varepsilon((b_j, \boldsymbol{y}_{j+1})|_e) \Big], \end{aligned} \tag{7}$$

where

$$i^*\text{-bias}_\varepsilon((b_j, \boldsymbol{y}_{j+1})|_e) = \begin{cases} \varepsilon^{-1} & \text{if both endpoints of } e \text{ are labeled 1 by } (b_j, \boldsymbol{y}_{j+1}) \\ (1-\varepsilon)^{-1} & \text{if both endpoints of } e \text{ are labeled 0 by } (b_j, \boldsymbol{y}_{j+1}). \end{cases}$$

We claim that the expectation on the RHS of (7) is 1 unless $e = (v^{(\ell)}, v^{(r)})$, where $\ell < r \le j$. To see this, note that if $j < \ell < r$ then

$$\mathop{\mathbf{E}}_{\boldsymbol{y}}\Big[\mathbf{1}[e \text{ is mono w.r.t. } (b_j, \boldsymbol{y}_{j+1})] \cdot i^*\text{-bias}_\varepsilon((b_j, \boldsymbol{y}_{j+1})|_e)\Big] = \varepsilon^2 \cdot \frac{1}{\varepsilon} + (1-\varepsilon)^2 \cdot \frac{1}{1-\varepsilon} = 1,$$

and if $\ell \le j \le r$ then

$$\mathop{\mathbf{E}}_{\boldsymbol{y}}\Big[\mathbf{1}[e \text{ is mono w.r.t. } (b_j, \boldsymbol{y}_{j+1})] \cdot i^*\text{-bias}_\varepsilon((b_j, \boldsymbol{y}_{j+1})|_e)\Big] = \begin{cases} \varepsilon \cdot \varepsilon^{-1} & \text{if } b_\ell = 1 \\ (1-\varepsilon) \cdot (1-\varepsilon)^{-1} & \text{if } b_\ell = 0. \end{cases}$$

Since neither $\mathbf{1}[e \text{ is mono w.r.t. } (b_j, \boldsymbol{y}_{j+1})]$ nor $i^*\text{-bias}_\varepsilon((b_j, \boldsymbol{y}_{j+1})|_e)$ depend on $\boldsymbol{y}$ when $e = (v^{(\ell)}, v^{(r)})$ where $\ell < r \le j$, it follows that (7) may be simplified to be

$$(7) = \prod_{\substack{i^*\text{-edges } e \\ e=(v^{(\ell)}, v^{(r)}),\, \ell<r\le j}} \mathbf{1}[e \text{ is mono w.r.t. } b] \cdot i^*\text{-bias}_\varepsilon(b|_e).$$

Recalling that this expression (7) depends on both $b \in \{0,1\}^q$ and $j \in [q]$ (since i^* depends on j), we write $\mathrm{val}(b, j)$ to denote (7), i.e.

$$\mathrm{val}(b, j) := \prod_{\substack{i^*\text{-edges } e \\ e=(v^{(\ell)}, v^{(r)}),\, \ell<r\le j}} \mathbf{1}[e \text{ is mono w.r.t. } b] \cdot i^*\text{-bias}_\varepsilon(b|_e) \tag{8}$$

and hence we may write

$$\mathrm{diff}(b, j) \le \frac{2\log k}{0.9k} \cdot \mathrm{val}(b, j).$$

2.4.2 Bounding $\mathrm{val}(b, j)$ by bucketing

Our goal is to define a bad set $\mathcal{B} \subseteq \{0,1\}^q$ of small measure ($\mathbf{Pr}[\boldsymbol{y} \in \mathcal{B}] \le 0.01$ is sufficient, though our $\mathcal{B}$ will satisfy $\mathbf{Pr}[\boldsymbol{y} \in \mathcal{B}] = k^{-\Omega(1)}$) such that for all $b \notin \mathcal{B}$,

$$\sum_{j \in [q]} \mathrm{val}(b, j)^2 = O\big(k^{(\frac{25+c}{13})}\big). \tag{9}$$

This is sufficient since it implies that we may take $\Delta := 0.01$ and have that the LHS of (4) is at most

$$\sum_{j \in [q]} \mathrm{diff}(b, j)^2 \le \left(\frac{2\log k}{0.9k}\right)^2 \sum_{j \in [q]} \mathrm{val}(b, j)^2 = \frac{1}{k^{2-o(1)}} \cdot O\big(k^{(\frac{25+c}{13})}\big) \le \Delta = 0.01$$

for sufficiently large k. (This uses (5) along with the fact that $c < 1$.) Applying Theorem 2.9 with $t = 0.1$ would then complete the proof of Proposition 2.8, and hence Theorem 1.1.

To reason about $b \in \{0,1\}^q$ for which (9) does not hold, we group the q many summands on the LHS of (9) into $O(\log k)$ groups according to magnitude. Set $M := \big(\frac{23+c}{24}\big)\log k$,

and partition $[0,\infty)$ into $M+2$ intervals $I_0 = [0,1)$, $I_m = [2^{m-1}, 2^m)$ for all $m \in [M]$ and $I_{M+1} = [2^M, \infty)$. For each $b \in \{0,1\}^q$ and $m \in \{0,1,\dots,M+1\}$ we define

$$\text{bucket}(b,m) := \{j \in [q] \colon \text{val}(b,j) \in I_m\},$$
$$C(b,m) := \sum_{j \in \text{bucket}(b,m)} \text{val}(b,j)^2.$$

With this notation in hand, we may write

$$\sum_{j\in[q]} \text{val}(b,j)^2 = \sum_{m=0}^{M+1} C(b,m). \tag{10}$$

Next, for each $m \in \{0,1,\dots,M+1\}$ we define $\mathcal{B}_m \subseteq \{0,1\}^q$ to be

$$\mathcal{B}_m := \left\{b \in \{0,1\}^q \colon C(b,m) > k^{(\frac{23+c}{12})}\right\},$$

and finally $\mathcal{B} := \bigcup \mathcal{B}_m$. Certainly if $b \notin \mathcal{B}$ then by (10) we have that

$$\sum_{j\in[q]} \text{val}(b,j)^2 = \sum_{m=0}^{M+1} C(b,m) \le (M+2)\cdot k^{(\frac{23+c}{12})} = o(k^{(\frac{25+c}{13})}),$$

and so it suffices to prove the following proposition.

▶ **Proposition 2.14.** *For all $m \in \{0,1,\dots,M+1\}$, we have that $\mathbf{Pr}_{\boldsymbol{b}\sim\{0,1\}^q_{(\varepsilon)}}[\boldsymbol{b} \in \mathcal{B}_m] = k^{-\Omega(1)}$. (Consequently, $\mathbf{Pr}[\boldsymbol{b} \in \mathcal{B}] = k^{-\Omega(1)}$ by a union bound.)*

Proof. First note that for all $m \in \{0,1,\dots,M\}$, we have that

$$C(b,m) = \sum_{j \in \text{bucket}(b,m)} \text{val}(b,j)^2 < |\text{bucket}(b,m)| \cdot 2^{2m}. \tag{11}$$

Since $|\text{bucket}(b,m)| \le q = o(k^{1.1})$ for all m, we have that (11) $< k^{1.7}$ for $m \le 0.3\log k$, and hence recalling the definition of $\mathcal{B}_m$, we have $\mathbf{Pr}[\boldsymbol{b} \in \mathcal{B}_m] = 0$ for $m \le 0.3\log k$, so the proposition clearly holds for all such m. Hence we may assume that $m > 0.3\log k$; it will be convenient for us to write $m = \alpha \log k$ for some $\alpha \in (0.3, 1)$. Next, observe that in order for $C(b,m)$ to be greater than $k^{(\frac{23+c}{12})}$ it has to be the case that $|\text{bucket}(b,m)| \ge k^{(\frac{23+c}{12})} \cdot 2^{-2m} = k^{(\frac{23+c}{12})-2\alpha}$; this is trivially true for $m = M+1$ (since $k^{(\frac{23+c}{12})} \cdot 2^{-2m} = \frac{1}{4}$), and follows from (11) for $m \in \{0,1,\dots,M\}$. We will therefore focus on bounding the RHS of

$$\mathbf{Pr}\left[|\text{bucket}(\boldsymbol{b},m)| \ge k^{(\frac{23+c}{12})-2\alpha}\right] \le \mathbf{Pr}\left[|\{j \in [q] \colon \text{val}(\boldsymbol{b},j) \ge 2^{m-1} = \tfrac{1}{2}k^\alpha\}| \ge k^{(\frac{23+c}{12})-2\alpha}\right]$$

by $k^{-\Omega(1)}$. Consider the random variable $\text{val}(\boldsymbol{b},j)$ for a fixed $j \in [q]$. Let $E = E(j)$ denote the number of i^*-edges (where i^* depends on j), and note that $E \le \log(k)/\varepsilon^c$ since $i^* \in U$ is an unsaturated direction. Recalling (8), we may introduce independent random variables $\mathbf{X}_1^{(j)}, \dots, \mathbf{X}_E^{(j)}$ where

$$\mathbf{X}_\ell^{(j)} = \begin{cases} 0 & \text{with probability } 2\varepsilon(1-\varepsilon) \\ (1-\varepsilon)^{-1} & \text{with probability } (1-\varepsilon)^2 \\ \varepsilon^{-1} & \text{with probability } \varepsilon^2 \end{cases}$$

and note that $\text{val}(\boldsymbol{b},j)$ (where $\boldsymbol{b} \sim \{0,1\}^q_{(\varepsilon)}$) is distributed identically to $\prod_{\ell=1}^E \mathbf{X}_\ell^{(j)}$. Simplifying further, we introduce additional (mutually independent) random variables $\mathbf{Y}_1^{(j)}, \dots, \mathbf{Y}_E^{(j)}$, where each $\mathbf{Y}_\ell^{(j)}$ is coupled to $\mathbf{X}_\ell^{(j)}$ in the following way

$$\mathbf{Y}_\ell^{(j)} = \begin{cases} \mathbf{X}_\ell^{(j)} & \text{when } \mathbf{X}_\ell^{(j)} = \varepsilon^{-1} \\ 1 & \text{otherwise.} \end{cases}$$

Under such a coupling, we have that

$$\prod_{\ell=1}^{E} \mathbf{X}_\ell^{(j)} \le \left(\frac{1}{1-\varepsilon}\right)^E \prod_{\ell=1}^{E} \mathbf{Y}_\ell^{(j)}$$

with probability 1, where the factor $(1-\varepsilon)^{-E}$ is $\le k^{\nu(k)}$ for some function $\nu(k) = o_k(1)$, for all $\varepsilon = o_k(1)$ since $E \le \log(k)/\varepsilon^c$ (recall the bounds on ε given in the conditions of Theorem 1.1).

▶ **Claim 2.15.** $\mathbf{Pr}\left[\prod_{\ell=1}^{E} \mathbf{Y}_\ell^{(j)} \ge \frac{1}{2}k^{\alpha-\nu(k)}\right] = O(k^{-(\frac{4-c}{3})\alpha})$.

Proof of Claim 2.15. Set $t := (m - \nu(k)\log(k) - 2)/\log(1/\varepsilon) = ((\alpha - \nu(k))\log(k) - 2)/\log(1/\varepsilon)$, and so $\varepsilon^{-t} = \frac{1}{2}k^{\alpha-\nu(k)}$.

$$\begin{aligned}
\mathbf{Pr}\left[\prod_{\ell=1}^{E} \mathbf{Y}_\ell^{(j)} \ge \frac{k^{\alpha-\nu(k)}}{2}\right] &< \varepsilon^{2t}\binom{E}{t} \\
&\le \frac{4}{k^{2\alpha-\nu(k)}}\left(\frac{eE}{t}\right)^t \\
&\le \frac{4}{k^{2\alpha-\nu(k)}}\left(\frac{e\log(1/\varepsilon)}{0.3\cdot\varepsilon^c}\right)^t \\
&\le \frac{4}{k^{2\alpha-\nu(k)}}\cdot O\left(\varepsilon^{-(\frac{1+c}{2})t}\right) \\
&< \frac{4}{k^{2\alpha-\nu(k)}}\cdot O\left(k^{(\frac{1+c}{2})(\alpha-\nu(k))}\right) = O\left(\frac{1}{k^{(\frac{4-c}{3})\alpha}}\right),
\end{aligned}$$

where the third inequality uses the fact that $t > 0.3\log(k)/\log(1/\varepsilon)$ (recall our assumption that $\alpha > 0.3$), the fourth inequality uses the fact that $\varepsilon = o_k(1)$, and the last inequality uses the fact that $\nu(k) = o_k(1)$. ◀

By linearity of expectation, it follows that

$$\mathbf{E}\left[|\{j \in [q] \colon \mathrm{val}(\boldsymbol{b}, j) \ge \tfrac{1}{2}k^\alpha\}|\right] = O\big(q\cdot k^{-(\frac{4-c}{3})\alpha}\big) = O\big(k^{(\frac{7-c}{6})}\cdot k^{-(\frac{4-c}{3})\alpha}\big) = O(k^{(\frac{7-c}{6})-(\frac{4-c}{3})\alpha}),$$

where we have used the fact that $q = O(k^{1+\eta})$ (recall (1) and the bounds on ε given in the conditions of Theorem 1.1) for any fixed $\eta > 0$, and so by Markov's inequality, we conclude that

$$\mathbf{Pr}\left[|\{j \in [q] \colon \mathrm{val}(\boldsymbol{b}, j) \ge \tfrac{1}{2}k^\alpha\}| \ge k^{(\frac{23+c}{12})-2\alpha}\right] = O\big(k^{(\frac{7-c}{6})-(\frac{4-c}{3})\alpha-((\frac{23+c}{12})-2\alpha)}\big) = O(k^{(\frac{c-1}{12})})$$

for sufficiently large k. Because $c < 1$, this is $k^{-\Omega(1)}$, and therefore the proof of Proposition 2.14 is complete. ◀

Acknowledgements. RS and LYT are supported by NSF grants CCF-1115703 and CCF-1319788, and JW is supported by NSF grants CCF-1115703, CCF-1319788, CCF-0747250, and CCF-1116594, and a Simons Fellowship in Theoretical Computer Science. Some of this work was done while LYT was at MSR-SVC and at Columbia University, and while JW was visiting Columbia University.

References

1 Noga Alon, Eric Blais, Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Nearly tight bounds for testing function isomorphism. *SIAM J. Comput.*, 42(2):459–493, 2013.

2 J. Arpe and E. Mossel. Application of a generalization of Russo's formula to learning from multiple random oracles. *Combinatorics, Probability and Computing*, 19:183–199, 2010.

3 J. Arpe and R. Reischuk. Robust inference of relevant attributes. In *Proceedings of the Fourteenth International Conference on Algorithmic Learning Theory*, pages 99–113, 2003.

4 A. Atıcı and R. Servedio. Quantum algorithms for testing and learning juntas. *Quantum Information Processing*, 6(5):323–348, 2007.

5 M. Ben-Or and N. Linial. Collective coin flipping. In S. Micali, editor, *Randomness and Computation*, pages 91–115. Academic Press, 1990.

6 Arthur J. Bernstein. Maximally connected arrays on the n-cube. *SIAM J. Appl. Math.*, 15(6):1485–1489, 1967.

7 E. Blais. Testing juntas: A brief survey. In *Property Testing - Current Research and Surveys*, pages 32–40, 2010.

8 Eric Blais. Improved bounds for testing juntas. In *Proc. RANDOM*, pages 317–330, 2008.

9 Eric Blais. Testing juntas nearly optimally. In *Proceedings of STOC*, pages 151–158, 2009.

10 Eric Blais, Joshua Brody, and Badih Ghazi. The information complexity of hamming distance. In *RANDOM*, pages 462–486, 2014.

11 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. In *CCC*, pages 210–220, 2011.

12 Eric Blais and Daniel M. Kane. Tight bounds for testing k-linearity. In *RANDOM*, pages 435–446, 2012.

13 Eric Blais and Ryan O'Donnell. Lower bounds for testing function isomorphism. In *IEEE Conference on Computational Complexity*, pages 235–246, 2010.

14 Eric Blais, Amit Weinstein, and Yuichi Yoshida. Partially symmetric functions are efficiently isomorphism-testable. In *FOCS*, pages 551–560, 2012.

15 A. Blum. Relevant examples and relevant features: Thoughts from computational learning theory. in AAAI Fall Symposium on 'Relevance', 1994.

16 A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

17 Harry Buhrman, David García-Soriano, Arie Matsliah, and Ronald de Wolf. The non-adaptive query complexity of testing k-parities. *Chicago Journal of Theoretical Computer Science*, 2013, 2013.

18 Sourav Chakraborty, Eldar Fischer, David García-Soriano, and Arie Matsliah. Junto-symmetric functions, hypergraph isomorphism and crunching. In *CCC*, pages 148–158, 2012.

19 Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Efficient sample extractors for juntas with applications. In *ICALP*, pages 545–556. Springer, 2011.

20 H. Chockler and D. Gutfreund. A lower bound for testing juntas. *Information Processing Letters*, 90(6):301–305, 2004.

21 D. Dachman-Soled, V. Feldman, L.-Y. Tan, A. Wan, and K. Wimmer. Approximate resilience, monotonicity, and the complexity of agnostic learning. In *SODA*, pages 498–511, 2015.

22 A. Dhagat and L. Hellerstein. PAC learning with irrelevant attributes. In *Proceedings of the Thirty-Fifth Annual Symposium on Foundations of Computer Science*, pages 64–74, 1994.

23 I. Diakonikolas, H. Lee, K. Matulef, K. Onak, R. Rubinfeld, R. Servedio, and A. Wan. Testing for concise representations. In *Proc. 48th Ann. Symposium on Computer Science (FOCS)*, pages 549–558, 2007.

24 I. Diakonikolas, H.K. Lee, K. Matulef, R. Servedio, and A. Wan. Efficiently testing sparse GF(2) polynomials. *Algorithmica*, 61(3):580–605, 2011.

25 Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. On agnostic learning of parities, monomials, and halfspaces. *SIAM J. Comput.*, 39(2):606–645, 2009.

26 E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky. Testing juntas. *J. Computer & System Sciences*, 68(4):753–787, 2004.

27 Peter Frankl. On the trace of finite sets. *J. Comb. Theory, Ser. A*, 34(1):41–45, 1983.

28 O. Goldreich, editor. *Property Testing: Current Research and Surveys*. Springer, 2010. LNCS 6390.

29 P. Gopalan, R. O'Donnell, R. Servedio, A. Shpilka, and K. Wimmer. Testing Fourier dimensionality and sparsity. *SIAM J. on Computing*, 40(4):1075–1100, 2011.

30 D. Guijarro, J. Tarui, and T. Tsukiji. Finding relevant variables in the PAC model with membership queries. In *Proceedings of the Tenth International Conference on Algorithmic Learning Theory*, pages 313–322, 1999.

31 Larry H. Harper. Optimal assignments of numbers to vertices. *SIAM J. Appl. Math.*, 12(1):131–135, 1964.

32 Sergiu Hart. A note on the edges of the n-cube. *Disc. Math.*, 14:157–163, 1976.

33 J. H. Lindsey. Assignment of numbers to vertices. *Amer. Math. Monthly*, 71:508–516, 1964.

34 K. Matulef, R. O'Donnell, R. Rubinfeld, and R. Servedio. Testing halfspaces. *SIAM J. on Comput.*, 39(5):2004–2047, 2010.

35 Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A. Servedio. Testing ± 1-weight halfspace. In *APPROX-RANDOM*, pages 646–657, 2009.

36 Colin McDiarmid. Concentration. In *Probabilistic Methods for Algorithmic Discrete Mathematics*, pages 195–248, 1998.

37 E. Mossel, R. O'Donnell, and R. Servedio. Learning functions of k relevant variables. *Journal of Computer & System Sciences*, 69(3):421–434, 2004. Preliminary version in *Proc. STOC'03*.

38 D. Ron. Property Testing: A Learning Theory Perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008.

39 D. Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5:73–205, 2010.

40 D. Ron and R. Servedio. Exponentially improved algorithms and lower bounds for testing signed majorities. In *SODA*, pages 1319–1336, 2013.

41 D. Ron and G. Tsur. Testing computability by width-two OBDDs. *Theoretical Computer Science*, 420:64–79, 2012.

42 Gregory Valiant. Finding Correlations in Subquadratic Time, with Applications to Learning Parities and Juntas. In *FOCS*, pages 11–20, 2012.