# Settling the Query Complexity of Non-adaptive Junta Testing

XI CHEN and ROCCO A. SERVEDIO, Columbia University, USA
LI-YANG TAN, Toyota Technological Institute, USA
ERIK WAINGARTEN and JINYU XIE, Columbia University, USA

We prove that any non-adaptive algorithm that tests whether an unknown Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a $k$-junta or $\epsilon$-far from every $k$-junta must make $\widetilde{\Omega}(k^{3/2}/\epsilon)$ many queries for a wide range of parameters $k$ and $\epsilon$. Our result dramatically improves previous lower bounds and is essentially optimal since there is a known non-adaptive junta tester which makes $\widetilde{O}(k^{3/2})/\epsilon$ queries. Combined with the known existence of an adaptive tester which makes $O(k \log k + k/\epsilon)$ queries, our result shows that adaptivity enables polynomial savings in query complexity for junta testing.

CCS Concepts: • **Theory of computation** → **Lower bounds and information complexity**;

Additional Key Words and Phrases: Property testing, juntas, adaptivity

**ACM Reference format:**
Xi Chen, Rocco A. Servedio, Li-Yang Tan, Erik Waingarten, and Jinyu Xie. 2018. Settling the Query Complexity of Non-adaptive Junta Testing. *J. ACM* 65, 6, Article 40 (November 2018), 18 pages.
https://doi.org/10.1145/3213772

## 1 INTRODUCTION

This article is concerned with the power of adaptivity in property testing, specifically property testing of Boolean functions. At a high level, a property tester for Boolean functions is a randomized algorithm, which, given black-box query access to an unknown and arbitrary Boolean function $f : \{0,1\}^n \to \{0,1\}$, aims to distinguish between the case that $f$ has some particular property of interest versus the case that $f$ is far in Hamming distance from every Boolean function satisfying the property. The main goals in the study of property testing algorithms are to develop testers that make *as few queries* as possible and to establish lower bounds matching these query-efficient algorithms. Property testing has by now been studied for many different types of Boolean functions, including linear functions and low-degree polynomials over $GF(2)$ [2, 6, 11], literals, conjunctions, $s$-term monotone, and non-monotone DNFs [18, 34], monotone and unate functions [3, 4, 13–16,

A preliminary version appeared in the proceedings of the 32nd Conference on Computational Complexity, 2017. This work is supported by NSF CCF-1703925, NSF CCF-1420349, NSF CCF-1563155, NSF CCF-1563122, and an NSF Graduate Research Fellowship under Grant No. DGE-16-44869.

Authors' addresses: X. Chen, R. A. Servedio, E. Waingarten, and J. Xie, Columbia University, 500 West 120 Street, New York, USA; emails: {xichen, rocco, eaw, jinyu}@cs.columbia.edu; L.-Y. Tan, Toyota Technological Institute, 6045 South Kenwood Ave, Chicago, Illinois, USA; email: liyang@cs.columbia.edu.

0004-5411/2018/11-ART40 $15.00
https://doi.org/10.1145/3213772

21, 25, 29, 30], various types of linear threshold functions [9, 32, 33], size-$s$ decision trees and $s$-sparse $GF(2)$ polynomials and parities [9, 10, 18], functions with sparse or low-degree Fourier spectrum [26], and much more. See, e.g., References [23, 24, 35, 36] for some fairly recent broad overviews of property testing research.

In this article, we consider the property of being a *$k$-junta*, which is one of the earliest and most extensively studied properties in the Boolean function property testing literature. Recall that $f$ 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 $x \in \{0,1\}^n$. Given $k = k(n) : \mathbb{N} \to \mathbb{N}$ and $\epsilon = \epsilon(n) : \mathbb{N} \to \mathbb{R}_{>0}$, we say an algorithm that has black-box access to an unknown and arbitrary $f : \{0,1\}^n \to \{0,1\}$ is an *$\epsilon$-tester* or *$\epsilon$-testing algorithm for $k$-juntas* if it accepts with probability at least 5/6 when $f$ is a $k(n)$-junta and rejects with probability at least 5/6 when $f$ is $\epsilon(n)$-far from all $k(n)$-juntas (meaning that $f$ disagrees with any $k(n)$-junta $g$ on at least $\epsilon(n) \cdot 2^n$ many inputs).

Property testers come in two flavors, adaptive and non-adaptive. An adaptive tester receives the value of $f$ on its $i$th query string before selecting its $(i+1)$-st query string, while a non-adaptive tester selects all of its query strings before receiving the value of $f$ on any of them. Note that non-adaptive testers can evaluate all of their queries in one parallel stage of execution, while this is in general not possible for adaptive testers. This means that if evaluating a query is very time-consuming, non-adaptive algorithms may sometimes be preferable to adaptive algorithms even if they require more queries. For this and other reasons, it is of interest to understand when, and to what extent, adaptive algorithms can use fewer queries than non-adaptive algorithms (see References [37, 38] for examples of property testing problems where indeed adaptive algorithms are provably more query-efficient than non-adaptive ones).

The query complexity of adaptive junta testing algorithms is at this point well understood. In Reference [17], Chockler and Gutfreund showed 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. Blais [8] gave an adaptive junta testing algorithm that uses only $O(k \log k + k/\epsilon)$ queries, which is optimal (for constant $\epsilon$) up to a multiplicative factor of $O(\log k)$.

Prior to the current work, the picture was significantly less clear for non-adaptive junta testing. In the first work on junta testing, Fischer et al. [20] gave a non-adaptive tester that makes $O(k^2(\log k)^2/\epsilon)$ queries. This was improved by Blais [7] with a non-adaptive tester that uses only $O(k^{3/2}(\log k)^3/\epsilon)$ queries. On the lower bounds side, Reference [7] also showed that for all $\epsilon \geq k/2^k$, any non-adaptive algorithm for $\epsilon$-testing $k$-juntas must make $\Omega\left(k/(\epsilon \log(k/\epsilon))\right)$ queries. Buhrman et al. [12] gave an $\Omega(k \log k)$ lower bound (for constant $\epsilon$) for non-adaptively testing whether a function $f$ is a size-$k$ parity; their argument also yields an $\Omega(k \log k)$ lower bound (for constant $\epsilon$) for non-adaptively $\epsilon$-testing $k$-juntas. More recently, Reference [40] obtained a new lower bound for non-adaptive junta testing that is incomparable to both the Reference [7] and Reference [12] lower bounds. They showed that for all $\epsilon : k^{-o_k(1)} \leq \epsilon \leq o_k(1)$, any non-adaptive $\epsilon$-tester for $k$-juntas must make

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

many queries, where $c$ is any absolute constant less than 1. For certain restricted values of $\epsilon$ such as $\epsilon = 1/\log k$, this lower bound is larger than the $O(k/\epsilon + k \log k)$ upper bound for Reference [8]'s adaptive algorithm, so the Reference [40] lower bound shows that in some restricted settings, adaptive junta testers can outperform non-adaptive ones. However, the difference in performance is quite small, at most a $o(\log k)$ factor. We further note that all of the lower bounds [7, 12, 40] are of the form $\widetilde{\Omega}(k)$[1] for constant $\epsilon$, and hence rather far from the $\widetilde{O}(k^{3/2})/\epsilon$ upper bound of Reference [7].

## 1.1 Our Results

The main result of the article is the following theorem:

Theorem 1. *Let $\alpha \in (0.5, 1)$ be an absolute constant. Let $k = k(n) : \mathbb{N} \to \mathbb{N}$ and $\epsilon = \epsilon(n) : \mathbb{N} \to \mathbb{R}_{>0}$ be two functions that satisfy $k(n) \le \alpha n$ and $2^{-n} \le \epsilon(n) \le 1/6$ for all sufficiently large $n$. Then any non-adaptive $\epsilon$-tester for $k$-juntas must make $\widetilde{\Omega}(k^{3/2}/\epsilon)$ many queries.*[2]

Together with the $\widetilde{O}(k^{3/2})/\epsilon$ non-adaptive upper bound from Reference [7], Theorem 1 settles the query complexity of non-adaptive junta testing up to poly-logarithmic factors.

## 1.2 High-Level Overview of Our Approach

Our lower bound approach differs significantly from previous work. Buhrman et al. [12] leveraged the connection between communication complexity lower bounds and property testing lower bounds that was established in the work of Reference [9] and applied an $\Omega(k \log k)$ lower bound on the one-way communication complexity of $k$-disjointness to establish their lower bound. Both References [7] and [40] are based on edge-isoperimetry results for the Boolean hypercube (the edge-isoperimetric inequality of Harper [27], Bernstein [5], Lindsey [31], and Hart [28] in the case of Reference [7], and a slight extension of a result of Frankl [22] in Reference [40]). In contrast, our lower bound argument takes a very different approach; it consists of a sequence of careful reductions, and employs an *upper* bound on the total variation distance between two Binomial distributions (see Claim 15).

Below we provide a high level overview of the proof of the lower bound given by Theorem 1. First, it is not difficult to show that Theorem 1 is a consequence of the following more specific lower bound for the case where $k = \alpha n$:

Theorem 2. *Let $\alpha \in (0.5, 1)$ be an absolute constant. Let $k = k(n) : \mathbb{N} \to \mathbb{N}$ and $\epsilon = \epsilon(n) : \mathbb{N} \to \mathbb{R}_{>0}$ be two functions that satisfy $k(n) = \alpha n$ and $2^{-(2\alpha-1)n/2} \le \epsilon(n) \le 1/6$ for sufficiently large $n$. Then any non-adaptive $\epsilon$-tester for $k$-juntas must make $\widetilde{\Omega}(n^{3/2}/\epsilon)$ many queries.*

See Appendix A for the proof that Theorem 2 implies Theorem 1.

We now provide a sketch of how Theorem 2 is proved. It may be convenient for the reader, on the first reading, to consider $\alpha = 3/4$ and to think of $\epsilon$ as being a small constant such as 0.01.

Fix a sufficiently large $n$. Let $k = \alpha n$ and $\epsilon = \epsilon(n)$ with $\epsilon$ satisfying the condition in Theorem 2. We proceed by Yao's principle and prove lower bounds for deterministic non-adaptive algorithms that receive inputs drawn from one of two probability distributions, $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$, over $n$-variable Boolean functions. The distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ are designed so that a Boolean function $f \leftarrow \mathcal{D}_{\text{yes}}$ is a $k$-junta with probability $1 - o(1)$ and $f \leftarrow \mathcal{D}_{\text{no}}$ is $\epsilon$-far from every $k$-junta with probability $1 - o(1)$. In Section 2, we define $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$, and establish the above properties. By Yao's principle, it then suffices to show that any $q$-query non-adaptive deterministic algorithm (i.e., any set of $q$ queries) that succeeds in distinguishing them must have $q = \widetilde{\Omega}(n^{3/2}/\epsilon)$.

This lower bound proof consists of two components:

(1) A reduction from a simple algorithmic task called Set-Size-Set-Queries (SSSQ for short), which we discuss informally later in this subsection and we define formally in Section 3.

[1]We write $\widetilde{\Omega}(f)$ to denote $f/\text{polylog}(f)$ and $\widetilde{O}(f)$ for $f \cdot \text{polylog}(f)$.

[2]The precise lower bound is $\Omega(\frac{k^{3/2}}{\epsilon \log^3(k) \log^3(k/\epsilon)})$.

This reduction implies that the non-adaptive deterministic query complexity of distinguishing $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ is at least as large as that of SSSQ.

(2) A lower bound of $\widetilde{\Omega}(n^{3/2}/\epsilon)$ for the query complexity of SSSQ.

Having outlined the formal structure of our proof, let us give some intuition, which may hopefully be helpful in motivating our construction and reduction. Our yes-functions and no-functions have very similar structure to each other, but are constructed with slightly different parameter settings. The first step in drawing a random function from $\mathcal{D}_{\text{yes}}$ is choosing a uniform random subset $\mathbf{M}$ of $\Theta(n)$ "addressing" variables from $x_1, \ldots, x_n$. A random subset $\mathbf{A}$ of the complementary variables $\overline{\mathbf{M}}$ is also selected, and for each assignment to the variables in $\mathbf{M}$ (let us denote such an assignment by $i$), there is an independent random function $\boldsymbol{h}_i$ over a randomly selected subset $\mathbf{S}_i$ of the variables in $\mathbf{A}$. A random function from $\mathcal{D}_{\text{no}}$ is constructed in the same way, except that now the random subset $\mathbf{A}$ is chosen to be slightly larger than in the yes case. This disparity in the size of $\mathbf{A}$ between the two cases causes random functions from $\mathcal{D}_{\text{yes}}$ to almost always be $k$-juntas and random functions from $\mathcal{D}_{\text{no}}$ to almost always be far from $k$-juntas.

An intuitive explanation of why this construction is amenable to a lower bound for non-adaptive algorithms is as follows. Intuitively, for an algorithm to determine that it is interacting with (say) a random no-function rather than a random yes-function, it must determine that the subset $\mathbf{A}$ is larger than it should be in the yes case. Since the set $\mathbf{M}$ of $\Theta(n)$ many "addressing" variables is selected randomly, if a non-adaptive algorithm uses two query strings $x, x'$ that differ in more than a few coordinates, it is very likely that the random set $\mathbf{M}$ will contain a variable where $x$ and $x'$ differ, and therefore, $x$ and $x'$ will correspond to two different random functions $\boldsymbol{h}_i, \boldsymbol{h}_{i'}$. Hence, every pair of query strings $x, x'$ that correspond to the same $\boldsymbol{h}_i$ can differ only in a few coordinates with high probability. This phenomenon significantly limits the power of a non-adaptive algorithm distinguishing $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$, and allows us to reduce from the algorithmic task SSSQ at the price of only a small quantitative cost in query complexity, see Section 4.

At a high level, the SSSQ task involves distinguishing whether or not a hidden set (corresponding to $\mathbf{A}$) is "large." An algorithm for this task can only access certain random bits, whose biases are determined by the hidden set and whose exact distribution is inspired by the exact definition of the random functions $\boldsymbol{h}_i$ over the random subsets $\mathbf{S}_i$. The SSSQ problem is much easier to work with compared to the original problem of distinguishing $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$. In particular, we give a reduction from an even simpler algorithmic task called Set-Size-Element-Queries (SSEQ for short) to SSSQ (see Section 5.1) and the query complexity lower bound for SSSQ follows directly from the lower bound for SSEQ presented in Section 5.2. We hope that the SSSQ problem and/or the SSEQ problem may find other applications in lower bounds for query algorithms.

Let us give a high-level description of the SSEQ task to provide some intuition for how we prove a query lower bound on it. Roughly speaking, in this task an oracle holds an unknown and random subset $\mathbf{A}$ of $[m]$ (here $m = \Theta(n)$), which is either "small" (size roughly $m/2$) or "large" (size roughly $m/2 + \Theta(\sqrt{n} \cdot \log n)$), and the task is to determine whether $\mathbf{A}$ is small or large. The algorithm may repeatedly query the oracle by providing it, at the $j$th query, with an element $i_j \in [m]$; if $i_j \notin \mathbf{A}$ then the oracle responds 0 with probability 1, and if $i_j \in \mathbf{A}$ then the oracle responds 1 with probability $\epsilon/\sqrt{n}$ and 0 otherwise. Intuitively, the only way for an algorithm to determine that the unknown set $\mathbf{A}$ is (say) large, is to determine that the fraction of elements of $[m]$ that belong to $\mathbf{A}$ is $1/2 + \Theta(\log n/\sqrt{n})$ rather than $1/2$; this in turn intuitively requires sampling $\Omega(n/\log^2 n)$ many random elements of $[m]$ and for each one ascertaining with high confidence whether or not it belongs to $\mathbf{A}$. But the nature of the oracle access described above for SSEQ is such that for any given $i \in [m]$, at least $\Omega(\sqrt{n}/\epsilon)$ many repeated queries to the oracle on input $i$ are required to reach even a modest level of confidence as to whether or not $i \in \mathbf{A}$. As alluded to earlier, the formal argument establishing our lower bound on the query complexity of SSEQ relies on an upper bound on the total variation distance between two binomial distributions.

### 1.3 Organization and Notation

We start with the definitions of $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ as well as proofs of their properties in Section 2. We then introduce SSSQ in Section 3, and give a reduction from SSSQ to the problem of distinguishing $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ in Section 4. More formally, we show that any non-adaptive deterministic algorithm that distinguishes $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ can be used to solve SSSQ with only an $O(\log n)$ factor loss in the query complexity. Finally, we prove in Section 5 a lower bound for the query complexity of SSSQ. Theorem 2 then follows by combining this lower bound with the reduction in Section 4.

We use boldfaced letters such as $\boldsymbol{f}, \mathbf{A}, \mathbf{S}$ to denote random variables. Given a string $x \in \{0,1\}^n$ and $\ell \in [n]$, we write $x^{(\ell)}$ to denote the string obtained from $x$ by flipping the $\ell$th coordinate. An *edge* along the $\ell$th direction in $\{0,1\}^n$ is a pair $(x, y)$ of strings with $y = x^{(\ell)}$ , and we refer to the strings $x$ and $y$ in $\{0,1\}^n$ as the *vertices* of the edge $(x, y)$. We say an edge $(x, y)$ is *bichromatic with respect to a function $f$* (or simply $f$-bichromatic) if $f(x) \neq f(y)$. Given $x \in \{0,1\}^n$ and $S \subseteq [n]$, we use $x_{|S} \in \{0,1\}^S$ to denote the projection of $x$ on $S$.

## 2 THE $\mathcal{D}_{\text{yes}}$ AND $\mathcal{D}_{\text{no}}$ DISTRIBUTIONS

Let $\alpha \in (0.5, 1)$ be an absolute constant. Let $n$ be a sufficiently large integer, with $k = \alpha n$, and let $\epsilon$ be the distance parameter that satisfies

$$2^{-(2\alpha-1)n/2} \leq \epsilon \leq 1/6. \tag{1}$$

In this section, we describe a pair of probability distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ supported over Boolean functions $f : \{0,1\}^n \to \{0,1\}$. We then show that $\boldsymbol{f} \leftarrow \mathcal{D}_{\text{yes}}$ is a $k$-junta with probability $1 - o(1)$, and that $\boldsymbol{f} \leftarrow \mathcal{D}_{\text{no}}$ is $\epsilon$-far from being a $k$-junta with probability $1 - o(1)$.

We start with some parameters settings. Define

$$\delta \stackrel{\text{def}}{=} 1 - \alpha \in (0, 0.5), \qquad p \stackrel{\text{def}}{=} \frac{1}{2}, \qquad p_{\text{no}} \stackrel{\text{def}}{=} \frac{1}{2} + \frac{\log n}{\sqrt{n}},$$

$$m \stackrel{\text{def}}{=} 2\delta n + \delta\sqrt{n}\log n, \qquad t \stackrel{\text{def}}{=} n - m = (2\alpha - 1)n - \delta\sqrt{n}\log n, \qquad N \stackrel{\text{def}}{=} 2^t.$$

A function $\boldsymbol{f} \leftarrow \mathcal{D}_{\text{yes}}$ is drawn according to the following randomized procedure:

(1) Sample a random subset $\mathbf{M} \subset [n]$ of size $t$. Let $\Gamma = \Gamma_{\mathbf{M}} : \{0,1\}^n \to [N]$ be the function that maps $x \in \{0,1\}^n$ to the integer encoded by $x_{|\mathbf{M}}$ in binary plus one. Note that $|\overline{\mathbf{M}}| = n - t = m$.

(2) Sample an $\mathbf{A} \subseteq \overline{\mathbf{M}}$ by including each element of $\overline{\mathbf{M}}$ in $\mathbf{A}$ independently with probability $p$.

(3) Sample independently a sequence of $N$ random subsets $\mathbf{S} = (\mathbf{S}_i : i \in [N])$ of $\mathbf{A}$ as follows: for each $i \in [N]$, each element of $\mathbf{A}$ is included in $\mathbf{S}_i$ independently with probability $\epsilon/\sqrt{n}$. Next we sample a sequence of $N$ functions $\mathbf{H} = (\boldsymbol{h}_i : i \in [N])$, by letting $\boldsymbol{h}_i : \{0,1\}^n \to \{0,1\}$ be a random function over the coordinates in $\mathbf{S}_i$, i.e., we sample an unbiased bit $\boldsymbol{z}_i(b)$ for each string $b \in \{0,1\}^{\mathbf{S}_i}$ independently and set $\boldsymbol{h}_i(x) = \boldsymbol{z}_i(x_{|\mathbf{S}_i})$.

(4) Finally, $\boldsymbol{f} = \boldsymbol{f}_{\mathbf{M},\mathbf{A},\mathbf{H}} : \{0,1\}^n \to \{0,1\}$ is defined using $\mathbf{M}$, $\mathbf{A}$ and $\mathbf{H}$ as follows (note that we can skip $\mathbf{S}$ since the choice of $\mathbf{S}$ is included in the choice of $\mathbf{H}$):

$$\boldsymbol{f}(x) = \boldsymbol{h}_{\Gamma_{\mathbf{M}}(x)}(x), \quad \text{for each } x \in \{0,1\}^n.$$

In words, an input $x$ is assigned the value $\boldsymbol{f}(x)$ as follows: according to the coordinates of $x$ in the set $\mathbf{M}$ (which intuitively should be thought of as unknown), one of the $N$ functions $\boldsymbol{h}_i$ (each of which is, intuitively, a random function over an unknown subset $\mathbf{S}_i$

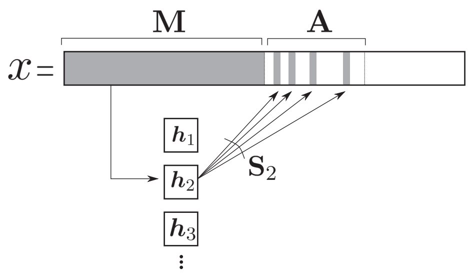

Fig. 1. An example of how an input $x \in \{0,1\}^n$ is evaluated by $f \sim \mathcal{D}_{\text{yes}}$ (or $\mathcal{D}_{\text{no}}$). The relevant variables of $x$ are shaded gray. The output $f(x)$ is computed in two steps. First, the input $x$ is indexed into one of $N$ functions $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N$ according to $\Gamma_{\mathbf{M}}(x) = x_{|\mathbf{M}} + 1$. Second, letting $i = \Gamma_{\mathbf{M}}(x)$, the output $f(x)$ is equal to $\boldsymbol{h}_i(x)$, which depends on the values of $x_{|\mathrm{S}_i}$ for a subset $\mathrm{S}_i \subset \mathbf{A}$.

of coordinates) is selected and evaluated on $x$'s coordinates in $\mathrm{S}_i$. For intuition, we note that both $\mathbf{M}$ and $\overline{\mathbf{M}}$ will always be of size $\Theta(n)$, the size of $\mathbf{A}$ will almost always be $\Theta(n)$, and for a given $i \in [N]$ the expected size of $\mathrm{S}_i$ will typically be $\Theta(\epsilon\sqrt{n})$ (though the size of $\mathrm{S}_i$ may not be as highly concentrated as the other sets when $\epsilon$ is tiny).

A function $f \leftarrow \mathcal{D}_{\text{no}}$ is generated using the same procedure except that $\mathbf{A}$ is a random subset of $\overline{\mathbf{M}}$ drawn by including each element of $\overline{\mathbf{M}}$ in $\mathbf{A}$ independently with probability $p_{\text{no}}$ (instead of $p$). See Figure 1 for an example of how an input $x \in \{0,1\}^n$ is evaluated by $f \sim \mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$.

## 2.1 Most Functions Drawn from $\mathcal{D}_{\text{yes}}$ are $k$-juntas

We first prove that $f \leftarrow \mathcal{D}_{\text{yes}}$ is a $k$-junta with probability $1 - o(1)$.

Lemma 3. *A function $f \leftarrow \mathcal{D}_{\text{yes}}$ is a $k$-junta with probability* $1 - o(1)$.

Proof. By the definition of $\mathcal{D}_{\text{yes}}$, all the relevant variables of $f \sim \mathcal{D}_{\text{yes}}$ belong to $\mathbf{M} \cup \mathbf{A}$. Note that $|\mathbf{M}| = t$. On the other hand, the expected size of $\mathbf{A}$ is $\delta n + \delta\sqrt{n}\log n/2$. By a Chernoff bound (see, e.g., Chapter 1 of Reference [19]), we have

$$|\mathbf{A}| \le \delta n + \frac{\delta\sqrt{n}\log n}{2} + \frac{\delta\sqrt{n}\log n}{4} < \delta n + \delta\sqrt{n}\log n$$

with probability $1 - o(1)$. When this happens, we have $|\mathbf{M} \cup \mathbf{A}| < \alpha n = k$. □

## 2.2 Most Functions Drawn from $\mathcal{D}_{\text{no}}$ are $\epsilon$-far from $k$-juntas

Next we prove that $f \leftarrow \mathcal{D}_{\text{no}}$ is $\epsilon$-far from any $k$-junta with probability $1 - o(1)$. The details of the argument are somewhat technical so we start by giving some high-level intuition, which is relatively simple. Since $p_{\text{no}} = p + \log(n)/\sqrt{n}$, a typical outcome of $\mathbf{A}$ drawn from $\mathcal{D}_{\text{no}}$ is slightly larger than a typical outcome drawn from $\mathcal{D}_{\text{yes}}$, and this difference causes almost every outcome of $|\mathbf{M} \cup \mathbf{A}|$ in $\mathcal{D}_{\text{no}}$ (with $\mathbf{M} \cup \mathbf{A}$ being the set of relevant variables for $f \leftarrow \mathcal{D}_{\text{no}}$) to be larger than $k$ by at least $9\sqrt{n}$. As a result, the relevant variables of any $k$-junta must miss either (a) at least one variable from $\mathbf{M}$, or (b) at least $9\sqrt{n}$ variables from $\mathbf{A}$. Missing even a single variable from $\mathbf{M}$ causes the $k$-junta to be far from $f$ (this is made precise in Claim 6 below). On the other hand, missing $9\sqrt{n}$ variables from $\mathbf{A}$ means that with probability $\Omega(\epsilon)$, at least one variable is missing from a typical $\mathrm{S}_i$ (recall that these are random $(\epsilon/\sqrt{n})$-dense subsets of $\mathbf{A}$). Because $\boldsymbol{h}_i$ is a random function over the variables in $\mathrm{S}_i$, missing even a single variable would lead to a constant fraction of error when $\boldsymbol{h}_i$ is the function determining the output of $f$.

Lemma 4. *A function $f \leftarrow \mathcal{D}_{\text{no}}$ is $\epsilon$-far from being a $k$-junta with probability* $1 - o(1)$.

Proof. Fix any subset $M \subset [n]$ of size $t$, and consider $\boldsymbol{f} = f_{M,\mathbf{A},\mathbf{H}}$ where $\mathbf{A}$ and $\mathbf{H}$ are sampled according to the procedure for $\mathcal{D}_{\text{no}}$. With probability $1 - o(1)$ over the choice of $\mathbf{A}$, we have

$$|\mathbf{A}| \geq p_{\text{no}}m - \frac{\delta\sqrt{n}\log n}{2} \geq \delta n + 2\delta\sqrt{n}\log n \quad \text{and} \quad |\mathbf{M} \cup \mathbf{A}| \geq k + \delta\sqrt{n}\log n. \tag{2}$$

Assume this is the case for the rest of the proof and fix any such set $A \subset \overline{M}$. It suffices to show that $\boldsymbol{f} = f_{M,A,\mathbf{H}}$ is $\epsilon$-far from any $k$-junta with probability $1 - o(1)$, where $\mathbf{H}$ is sampled according to the rest (steps 3 and 4) of the procedure for $\mathcal{D}_{\text{no}}$ (by sampling $\mathbf{S}_i$ from $A$ and then $\boldsymbol{h}_i$ over $\mathbf{S}_i$).

The plan for the rest of the proof is the following. For each $V \subset M \cup A$ of size $9\sqrt{n}$, we use $\mathbf{E}_V$ to denote the size of the *maximum* set of vertex-disjoint, $\boldsymbol{f}$-bichromatic edges along directions in $V$ only. We will prove the following claim:

Claim 5. *For each $V \subset M \cup A$ of size $9\sqrt{n}$, we have $\mathbf{E}_V \geq \epsilon 2^n$ with probability $1 - \exp(-2^{\Omega(n)})$ over the choice of $\mathbf{H}$.*

Note that when $\mathbf{E}_V \geq \epsilon 2^n$, we have $\text{dist}(\boldsymbol{f}, g) \geq \epsilon$ for every function $g$ that does not depend on any variable in $V$. This is because, for every $\boldsymbol{f}$-bichromatic edge $(x, x^{(\ell)})$ along a coordinate $\ell \in V$, we must have $\boldsymbol{f}(x) \neq \boldsymbol{f}(x^{(\ell)})$ since the edge is bichromatic but $g(x) = g(x^{(\ell)})$ as $g$ does not depend on the $\ell$th variable. As a result, $\boldsymbol{f}$ must disagree with $g$ on at least $\epsilon 2^n$ many points.

Assuming Claim 5 for now, we can apply a union bound over all

$$\binom{|M \cup A|}{9\sqrt{n}} \leq \binom{n}{9\sqrt{n}} \leq 2^{O(\sqrt{n}\log n)}$$

possible choices of $V \subset M \cup A$ to conclude that with probability $1 - o(1)$, $\boldsymbol{f} = f_{M,A,\mathbf{H}}$ is $\epsilon$-far from all functions that do not depend on at least $9\sqrt{n}$ variables in $M \cup A$. By Equation (2), this set includes all $k$-juntas. This concludes the proof of the Lemma 4 modulo the proof of Claim 5. □

In the rest of the section, we prove Claim 5 for a fixed subset $V \subset M \cup A$ of size $9\sqrt{n}$. We start with the simpler case when $V \cap M$ is nonempty.

Claim 6. *If $V \cap M \neq \emptyset$, then $\mathbf{E}_V \geq 2^n/5$ with probability $1 - \exp(-2^{\Omega(n)})$ over the choice of $\mathbf{H}$.*

Proof. Fix an $\ell \in V \cap M$; we will argue that with probability $1 - \exp(-2^{\Omega(n)})$ there are at least $2^n/5\boldsymbol{f}$-bichromatic edges along direction $\ell$. This suffices since such edges are clearly vertex-disjoint.

Observe that since $\ell \in M$, every $x \in \{0,1\}^n$ has $\Gamma(x) \neq \Gamma(x^{(\ell)})$. For each $b \in \{0,1\}^M$, let $X_b$ be the set of $x \in \{0,1\}^n$ with $x_{|M} = b$. We partition $\{0,1\}^n$ into $2^{t-1}$ pairs $X_b$ and $X_{b^{(\ell)}}$, where $b$ ranges over the $2^{t-1}$ strings in $\{0,1\}^M$ with $b_\ell = 0$. For each such pair, we use $\mathbf{D}_b$ to denote the number of $\boldsymbol{f}$-bichromatic edges between $X_b$ and $X_{b^{(\ell)}}$. We are interested in lower bounding $\sum_b \mathbf{D}_b$.

We will apply Hoeffding's inequality (see, e.g., Chapter 1 of Reference [19]). For this purpose we note that the $\mathbf{D}_b$'s are independent (since they depend on distinct $\boldsymbol{h}_i$'s), always lie between 0 and $2^m$, and each one has expectation $2^{m-1}$. The latter is because each edge $(x, x^{(\ell)})$ has $\boldsymbol{f}(x)$ and $\boldsymbol{f}(x^{(\ell)})$ drawn as two independent random bits, which is the case since $\Gamma(x) \neq \Gamma(x^{(\ell)})$. Thus, the expectation of $\sum_b \mathbf{D}_b$ is $2^{n-2}$. By Hoeffding's inequality, we have

$$\mathbf{Pr}\left[\left|\sum \mathbf{D}_b - 2^{n-2}\right| \geq \frac{2^n}{20}\right] \leq 2 \cdot \exp\left(-\frac{2(2^n/20)^2}{2^{t-1} \cdot 2^{2m}}\right) = \exp\left(-2^{\Omega(n)}\right)$$

since $t = \Omega(n)$. This finishes the proof of the claim. □

Now we may assume that $V \subset A$ (and $|V| = 9\sqrt{n}$). We use $\mathbf{I}$ to denote the set of $i \in [N]$ such that $\mathbf{S}_i \cap V \neq \emptyset$. The following claim shows that $\mathbf{I}$ is large with extremely high probability:

Claim 7. *We have $|\mathbf{I}| \geq 4.4\epsilon N$ with probability at least $1 - \exp(-2^{\Omega(n)})$ over the choice of $\mathbf{S}$.*

Proof. For each $i \in [N]$ we have (using $1 - x \le e^{-x}$ for all $x$ and $1 - x/2 \ge e^{-x}$ for $x \in [0, 1.5]$):

$$\mathbf{Pr}[i \in \mathbf{I}] = 1 - \left(1 - \frac{\epsilon}{\sqrt{n}}\right)^{9\sqrt{n}} \ge 1 - e^{-9\epsilon} \ge 4.5\epsilon,$$

since $\epsilon/\sqrt{n}$ is the probability of each element of $A$ being included in $\mathbf{S}_i$ and $\epsilon \le 1/6$ so $9\epsilon \le 1.5$.

Using $\epsilon \ge 2^{-(2\alpha-1)n/2}$ from (1), we have $\mathbf{E}[|\mathbf{I}|] \ge 4.5\epsilon N = 2^{\Omega(n)}$. Since the $\mathbf{S}_i$'s are independent, a Chernoff bound implies that $|\mathbf{I}| \ge 4.4\epsilon N$ with probability $1 - \exp(-2^{\Omega(n)})$. □

By Claim 7, we fix $S_1, \ldots, S_N$ to be any sequence of subsets of $A$ that satisfy $|I| \ge 4.4\epsilon N$ in the rest of the proof, and it suffices to show that over the random choices of $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N$ (where each $\boldsymbol{h}_i$ is chosen to be a random function over $S_i$), $\mathbf{E}_V \ge \epsilon 2^n$ with probability at least $1 - \exp(-2^{\Omega(n)})$.

To this end we use $\rho(i)$ for each $i \in I$ to denote the first coordinate of $S_i$ in $V$, and $Z_i$ to denote the set of $x \in \{0,1\}^n$ with $\Gamma(x) = i$. Note that the $Z_i$'s are disjoint. We further partition each $Z_i$ into disjoint $Z_{i,b}$, $b \in \{0,1\}^{S_i}$, with $x \in Z_{i,b}$ iff $x \in Z_i$ and $x_{|S_i} = b$. For each $i \in I$ and $b \in \{0,1\}^{S_i}$ with $b_{\rho(i)} = 0$, we use $\mathbf{D}_{i,b}$ to denote the number of $\boldsymbol{f}$-bichromatic edges between $Z_{i,b}$ and $Z_{i,b^{(\rho(i))}}$ along the $\rho(i)$th direction. It is clear that such edges, over all $i$ and $b$, are vertex-disjoint and thus,

$$\mathbf{E}_V \ge \sum_{i \in I} \sum_{\substack{b \in \{0,1\}^{S_i} \\ b_{\rho(i)}=0}} \mathbf{D}_{i,b}. \tag{3}$$

We will apply Hoeffding's inequality. Note that $\mathbf{D}_{i,b}$ is $2^{m-|S_i|}$ with probability 1/2, and 0 with probability 1/2. Thus, the expectation of the RHS of Equation (3) is

$$\sum_{i \in I} 2^{|S_i|-1} \cdot 2^{m-|S_i|-1} = |I| \cdot 2^{m-2} \ge 1.1\epsilon 2^n,$$

using $|I| \ge 4.4\epsilon N$. Since all the $\mathbf{D}_{i,b}$'s are independent, by Hoeffding's inequality we have

$$\mathbf{Pr}\left[\,\left|\text{RHS of (3)} - |I| \cdot 2^{m-2}\right| \ge 0.01|I| \cdot 2^{m-2}\,\right] \le 2 \cdot \exp\left(-\frac{2(0.01|I| \cdot 2^{m-2})^2}{\sum_{i \in I} 2^{|S_i|-1} \cdot 2^{2(m-|S_i|)}}\right)$$
$$\le \exp\left(-2^{\Omega(n)}\right),$$

since $|I| \ge \Omega(\epsilon N) = 2^{\Omega(n)}$. Therefore, with probability $1 - \exp(2^{-\Omega(n)})$, we have

$$\mathbf{E}_V \ge 0.99 \cdot |I| \cdot 2^{m-2} > \epsilon 2^n.$$

This concludes the proof of Claim 5. □

## 3 THE SET-SIZE-SET-QUERIES (SSSQ) PROBLEM

We first introduce the SSSQ problem, which is an artificial problem that we use as a bridge to prove Theorem 2. We use the same parameters $p, p_{\text{no}}$, and $m$ from the definition of $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$, with $n$ being sufficiently large (so $m = \Omega(n)$ is also sufficiently large).

We start by defining $\mathcal{A}_{\text{yes}}$ and $\mathcal{A}_{\text{no}}$, two distributions over subsets of $[m]$: $\mathbf{A} \sim \mathcal{A}_{\text{yes}}$ is drawn by independently including each element of $[m]$ with probability $p$ and $\mathbf{A} \sim \mathcal{A}_{\text{no}}$ is drawn by independently including each element with probability $p_{\text{no}}$. In SSSQ, the algorithm needs to determine whether an unknown $A \subseteq [m]$ is drawn from $\mathcal{A}_{\text{yes}}$ or $\mathcal{A}_{\text{no}}$. (For intuition, to see that this task is reasonable, we observe here that a straightforward Chernoff bound shows that almost every outcome of $\mathbf{A} \sim \mathcal{A}_{\text{yes}}$ is larger than almost every outcome of $\mathbf{A} \sim \mathcal{A}_{\text{no}}$ by $\Omega(\sqrt{n}\log n)$.)

Let $A$ be a subset of $[m]$ that is hidden in an oracle. An algorithm accesses $A$ (to tell whether it is drawn from $\mathcal{A}_{\text{yes}}$ or $\mathcal{A}_{\text{no}}$) by interacting with the oracle in the following way: each time it calls the oracle, it does so by sending a subset of $[m]$ to the oracle. The oracle responds as follows: for each $j$ in the subset, it returns a bit that is 0 if $j \notin A$, and is 1 with probability $\epsilon/\sqrt{n}$ and 0 with probability $1 - \epsilon/\sqrt{n}$ if $j \in A$. The cost of such an oracle call is the size of the subset provided to the oracle.

More formally, a deterministic and non-adaptive algorithm $\textsc{Alg} = (g, T)$ for SSSQ accesses the set $A$ hidden in the oracle by submitting a list of queries $T = (T_1, \ldots, T_d)$, for some $d \geq 1$, where each $T_i \subseteq [m]$ is a set. (Thus, we call each $T_i$ a *set query*, as part of the name SSSQ.)

— Given $T$, the oracle returns a list of random vectors $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d)$, where $\boldsymbol{v}_i \in \{0,1\}^{T_i}$ and each bit $\boldsymbol{v}_{i,j}$ is independently distributed as follows: if $j \notin A$, then $\boldsymbol{v}_{i,j} = 0$, and if $j \in A$, then

$$\boldsymbol{v}_{i,j} = \begin{cases} 1 & \text{with probability } \epsilon/\sqrt{n} \\ 0 & \text{with probability } 1 - (\epsilon/\sqrt{n}). \end{cases} \tag{4}$$

Note that the random vectors in $\boldsymbol{v}$ depend on both $T$ and $A$.

— Given $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d)$, $\textsc{Alg}$ returns (deterministically) the value of $g(\boldsymbol{v}) \in \{\text{“yes,” “no”}\}$.

The performance of $\textsc{Alg} = (g, T)$ is measured by its *query complexity* and its *advantage*.

— The query complexity of $\textsc{Alg}$ is defined as $\sum_{i=1}^{d} |T_i|$, the total size of all the set queries. On the other hand, the advantage of $\textsc{Alg}$ is defined as

$$\Pr_{\mathbf{A} \sim \mathcal{A}_{\text{yes}}}\left[\textsc{Alg}(\mathbf{A}) = \text{“yes”}\right] - \Pr_{\mathbf{A} \sim \mathcal{A}_{\text{no}}}\left[\textsc{Alg}(\mathbf{A}) = \text{“yes”}\right].$$

*Remark 8.* In the definition above, $g$ is a deterministic map from all possible sequences of vectors returned by the oracle to “yes” or “no.” Considering only deterministic as opposed to randomized $g$ is without loss of generality since given any query sequence $T$, the highest possible advantage can always be achieved by a deterministic map $g$.

We prove the following lower bound for any deterministic, non-adaptive $\textsc{Alg}$ in Section 5.

Lemma 9. *Any deterministic, non-adaptive $\textsc{Alg}$ for* SSSQ *with advantage at least* $2/3$ *satisfies*

$$\sum_{i=1}^{d} |T_i| \geq \frac{n^{3/2}}{\epsilon \cdot \log^3 n \cdot \log^2(n/\epsilon)}.$$

## 4 REDUCING FROM SSSQ TO DISTINGUISHING $\mathcal{D}_{\text{yes}}$ AND $\mathcal{D}_{\text{no}}$

In this section, we reduce from SSSQ to the problem of distinguishing the pair of distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$. More precisely, let $\textsc{Alg}^* = (h, X)$ be a deterministic and nonadaptive algorithm that makes $q \leq (n/\epsilon)^2$ string queries[3] $X = (x_1, \ldots, x_q)$ to a hidden function $f$ drawn from either $\mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$, applies the (deterministic) map $h$ to return $h(f(x_1), \ldots, f(x_q)) \in \{\text{“yes,” “no”}\}$, and satisfies

$$\Pr_{f \sim \mathcal{D}_{\text{yes}}}\left[\textsc{Alg}^*(f) = \text{“yes”}\right] - \Pr_{f \sim \mathcal{D}_{\text{no}}}\left[\textsc{Alg}^*(f) = \text{“yes”}\right] \geq 3/4. \tag{5}$$

We show how to define from $\textsc{Alg}^* = (h, X)$ an algorithm $\textsc{Alg} = (g, T)$ for the problem SSSQ with query complexity at most $\tau \cdot q$ and advantage $2/3$, where $\tau = c_\alpha \cdot 5 \log(n/\epsilon)$ and

$$c_\alpha = -\frac{1}{\log(1.5 - \alpha)} > 0 \quad \text{with} \quad (1.5 - \alpha)^{c_\alpha} = 1/2$$

is a constant that depends on $\alpha$. Given this reduction it follows from Lemma 9 that $q \geq \widetilde{\Omega}(n^{3/2}/\epsilon)$. This finishes the proof of Theorem 2.

[3] Any algorithm that makes more than this many queries already fits the $\widetilde{\Omega}(n^{3/2}/\epsilon)$ lower bound we aim for.

We first give a sketch of the reduction and some intuition behind the proof. Fixing a subset $M$ of $[n]$ of size $t$, we use $X_1, \ldots, X_d$, for some $d \geq 1$, to denote a partition of the query strings $x_1, \ldots, x_q$ such that $x_i$ and $x_j$ belong to the same $X_\ell$ if and only if $(x_i)_{|M} = (x_j)_{|M}$. For each $\ell \in [d]$, we use $T_\ell$ to denote the set of indices $k \in \overline{M}$ such that $x_k \neq y_k$ for some strings $x, y \in X_\ell$. The first part of the proof shows that there exists an $M$ such that (1) $\textsc{Alg}^*$ can distinguish $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ conditioning on $\mathbf{M} = M$ (recall in both $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$, $\mathbf{M}$ is a subset of $[n]$ of size $t$ drawn uniformly at random), and (2) $\|x - y\|_1 \leq \tau$ for all $\ell \in [d]$ and $x, y \in X_\ell$, which in turn implies that $\sum_{\ell \in [d]} |T_\ell| \leq \tau \cdot q$. Indeed, we show that most draws from $\mathbf{M}$ satisfy both properties; the intuition behind (2) is that two query strings with a large Hamming distance would have different projections on $\mathbf{M}$ with high probability. Fixing such an $M$, we identify indices of $\overline{M}$ as those of $[m]$ in SSSQ (by picking an arbitrary bijection between them) and show that $T = (T_1, \ldots, T_d)$ can be used to obtain an algorithm $\textsc{Alg} = (g, T)$ for SSSQ, with query complexity at most $\tau \cdot q$, for some appropriate $g$. The intuition is that in SSSQ we receive intersections of $T_\ell$ with random subsets $\mathbf{S}_\ell$ drawn independently from the hidden subset $\mathbf{A}$, which can be used to simulate random functions $\boldsymbol{h}_\ell$ over $\mathbf{S}_\ell$ evaluated on strings in $X_\ell$.

We start the reduction with some notation. For a fixed $M$ of size $t$, we use $\mathcal{E}_{\text{yes}}(M)$ to denote the distribution of $\mathbf{A}$ and $\mathbf{H}$ sampled in the randomized procedure for $\mathcal{D}_{\text{yes}}$, conditioning on $\mathbf{M} = M$. We define $\mathcal{E}_{\text{no}}(M)$ similarly. Then conditioning on $\mathbf{M} = M$, $\boldsymbol{f} \sim \mathcal{D}_{\text{yes}}$ is distributed as $f_{M,\mathbf{A},\mathbf{H}}$ with $(\mathbf{A}, \mathbf{H}) \sim \mathcal{E}_{\text{yes}}(M)$ and $\boldsymbol{f} \sim \mathcal{D}_{\text{no}}$ is distributed as $f_{M,\mathbf{A},\mathbf{H}}$ with $(\mathbf{A}, \mathbf{H}) \sim \mathcal{E}_{\text{no}}(M)$. This allows us to rewrite Equation (5) as

$$\frac{1}{\binom{n}{t}} \cdot \sum_{M:|M|=t} \left( \Pr_{(\mathbf{A},\mathbf{H}) \sim \mathcal{E}_{\text{yes}}(M)} \left[ \textsc{Alg}^*(\boldsymbol{f}_{M,\mathbf{A},\mathbf{H}}) = \text{“yes”} \right] - \Pr_{(\mathbf{A},\mathbf{H}) \sim \mathcal{E}_{\text{no}}(M)} \left[ \textsc{Alg}^*(\boldsymbol{f}_{M,\mathbf{A},\mathbf{H}}) = \text{“yes”} \right] \right) \geq \frac{3}{4}.$$

We say $M \subset [n]$ is *good* if any two queries $x_i$ and $x_j$ in $X$ with Hamming distance $\|x_i - x_j\|_1 \geq \tau$ have different projections on $M$, i.e., $(x_i)_{|M} \neq (x_j)_{|M}$. We prove below that most $M$'s are good.

Claim 10. $\mathbf{Pr}_{\mathbf{M}}[\mathbf{M}$ *is not good*$] = o(1)$.

Proof. For each pair of strings $x_i$ and $x_j$ in $X$ with Hamming distance at least $\tau$, the probability of them having the same projection on $\mathbf{M}$ (drawn uniformly from all size-$t$ subsets) is at most

$$\frac{\binom{n-\tau}{t}}{\binom{n}{t}} = \frac{(n-\tau-t+1)\cdots(n-t)}{(n-\tau+1)\cdots n} \leq \left(1 - \frac{t}{n}\right)^\tau \leq \left(2(1-\alpha) + o(1)\right)^\tau < (1.5-\alpha)^\tau \leq O\left(\frac{\epsilon}{n}\right)^5,$$

by our choices of $c_\alpha$ and $\tau$. The claim follows by a union bound over at most $q^2 \leq (n/\epsilon)^4$ pairs. □

We can split the sum Equation (5) into two sums: the sum over good $M$ and the sum over bad $M$. By Claim 10, the contribution from the bad $M$ is at most $o(1)$, and thus we have that

$$\frac{1}{\binom{n}{t}} \cdot \sum_{\text{good } M} \left( \Pr_{(\mathbf{A},\mathbf{H}) \sim \mathcal{E}_{\text{yes}}(M)} \left[ \textsc{Alg}^*(\boldsymbol{f}_{M,\mathbf{A},\mathbf{H}}) = \text{“yes”} \right] - \Pr_{(\mathbf{A},\mathbf{H}) \sim \mathcal{E}_{\text{no}}(M)} \left[ \textsc{Alg}^*(\boldsymbol{f}_{M,\mathbf{A},\mathbf{H}}) = \text{“yes”} \right] \right)$$

is at least $3/4 - o(1)$. Thus, there must exist a good set $M \subset [n]$ of size $t$ with

$$\Pr_{(\mathbf{A},\mathbf{H}) \sim \mathcal{E}_{\text{yes}}(M)} \left[ \textsc{Alg}^*(\boldsymbol{f}_{M,\mathbf{A},\mathbf{H}}) = \text{“yes”} \right] - \Pr_{(\mathbf{A},\mathbf{H}) \sim \mathcal{E}_{\text{no}}(M)} \left[ \textsc{Alg}^*(\boldsymbol{f}_{M,\mathbf{A},\mathbf{H}}) = \text{“yes”} \right] \geq 2/3. \quad (6)$$

Fix such a good $M$. We use $\textsc{Alg}^* = (h, X)$ and $M$ to define an algorithm $\textsc{Alg} = (g, T)$ for SSSQ as follows (note that the algorithm $\textsc{Alg}$ below actually works over the universe $\overline{M}$ (of size $m$) instead of $[m]$ as in the original definition of SSSQ but this can be handled by picking any bijection between $\overline{M}$ and $[m]$; accordingly, $\mathbf{A} \sim \mathcal{A}_{\text{yes}}$ is drawn by including each element of $\overline{M}$ with probability $p$ and $\mathbf{A} \sim \mathcal{A}_{\text{no}}$ is drawn by including each element of $\overline{M}$ with probability $p_{\text{no}}$). We start with $T$:

(1) First we use $M$ to define an equivalence relation $\sim$ over the query set $X$, where $x_i \sim x_j$ if $(x_i)_{|M} = (x_j)_{|M}$. Let $X_1, \ldots, X_d$, $d \geq 1$, denote the equivalence classes of $X$, and let us write $\rho(\ell)$ for each $\ell \in [d]$ to denote the value $\Gamma(x) \in [N]$ that is shared by all strings $x \in X_\ell$.

(2) Next we define a sequence of subsets of $\overline{M}$, $T = (T_1, \ldots, T_d)$, as the set queries of Alg, where

$$T_\ell = \Big\{ i \in \overline{M} : \exists x, y \in X_\ell \text{ such that } x_i \neq y_i \Big\}. \tag{7}$$

To upper bound $|T_\ell|$, fixing an arbitrary string $x \in X_\ell$ and recalling that $M$ is good, we have that

$$|T_\ell| \leq \sum_{y \in X_\ell} \|x - y\|_1 \leq \sum_{y \in X_\ell} \tau = \tau \cdot |X_\ell|.$$

As a result, the query complexity of Alg (using $T$ as its set queries) is at most

$$\sum_{\ell=1}^{d} |T_\ell| \leq \tau \cdot \sum_{\ell=1}^{d} |X_\ell| \leq \tau \cdot q.$$

It remains to define $h$ and then prove that the advantage of Alg $= (g, T)$ for SSSQ is at least $2/3$. Indeed the $g$ that we define is a randomized map and we describe it as a randomized procedure below (by Remark 8, one can extract from $g$ a deterministic map that achieves the same advantage):

(1) Given $v_1, \ldots, v_d$, $v_\ell \in \{0,1\}^{T_\ell}$, as the strings returned by the oracle upon being given $T$, let

$$R_\ell = \Big\{ j \in T_\ell : v_{\ell,j} = 1 \Big\}. \tag{8}$$

For each $\ell \in [d]$, the procedure draws a random function $\boldsymbol{f}_\ell : \{0,1\}^{R_\ell} \to \{0,1\}$, by flipping $2^{|R_\ell|}$ many independent and unbiased random bits.

(2) Next, for each query $x \in X_\ell$, $\ell \in [d]$, we feed $\boldsymbol{f}_\ell(x_{|R_\ell})$ to $h$ as the bit that the oracle returns upon the query $x$. Finally the procedure returns the result ("yes" or "no") that $h$ returns.

In the rest of the proof we show that the advantage of Alg $= (g, T)$ is exactly the same as the LHS of Equation (6) and thus, is at least 2/3.

For convenience, we use $\mathcal{V}_{\text{yes}}$ to denote the distribution of responses $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d)$ to $T$ when $\mathbf{A} \sim \mathcal{A}_{\text{yes}}$, and $\mathcal{V}_{\text{no}}$ to denote the distribution when $\mathbf{A} \sim \mathcal{A}_{\text{no}}$. Then the advantage of Alg is

$$\Pr_{\boldsymbol{v} \sim \mathcal{V}_{\text{yes}}} \Big[ g(\boldsymbol{v}) = \text{"yes"} \Big] - \Pr_{\boldsymbol{v} \sim \mathcal{V}_{\text{no}}} \Big[ g(\boldsymbol{v}) = \text{"yes"} \Big].$$

It suffices to show that

$$\Pr_{\boldsymbol{v} \sim \mathcal{V}_{\text{yes}}} \Big[ g(\boldsymbol{v}) = \text{"yes"} \Big] = \Pr_{(\mathbf{A},\mathbf{H}) \sim \mathcal{E}_{\text{yes}}(M)} \Big[ \text{Alg}^*(\boldsymbol{f}_{M,\mathbf{A},\mathbf{H}}) = \text{"yes"} \Big] \quad \text{and} \tag{9}$$

$$\Pr_{\boldsymbol{v} \sim \mathcal{V}_{\text{no}}} \Big[ g(\boldsymbol{v}) = \text{"yes"} \Big] = \Pr_{(\mathbf{A},\mathbf{H}) \sim \mathcal{E}_{\text{no}}(M)} \Big[ \text{Alg}^*(\boldsymbol{f}_{M,\mathbf{A},\mathbf{H}}) = \text{"yes"} \Big]. \tag{10}$$

We show Equation (9); the proof of Equation (10) is similar. From the definition of $\mathcal{V}_{\text{yes}}$ and $\mathcal{E}_{\text{yes}}(M)$, the distribution of $(\mathbf{R}_\ell : \ell \in [d])$ derived from $\boldsymbol{v} \sim \mathcal{V}_{\text{yes}}$ using Equation (8) is the same as the distribution of $(\mathbf{S}_{\rho(\ell)} \cap T_\ell : \ell \in [d])$: both are sampled by first drawing a random subset $\mathbf{A}$ of $\overline{M}$ and then drawing a random subset of $\mathbf{A} \cap T_\ell$ independently by including each element of $\mathbf{A} \cap T_\ell$ with the same probability $\epsilon/\sqrt{n}$ (recall, in particular, Equation (4) and step 3 of the randomized procedure specifying $\mathcal{D}_{\text{yes}}$ in Section 2). Since $\boldsymbol{f}_{M,\mathbf{A},\mathbf{H}}(x)$ for $x \in X_\ell$ is determined by a random Boolean function $\boldsymbol{h}_{\rho(\ell)}$ from $\{0,1\}^{\mathbf{S}_{\rho(\ell)}}$ to $\{0,1\}$, and since all the queries in $X_\ell$ only differ by coordinates in $T_\ell$, the distribution of the $q$ bits that $g$ feeds to $h$ when $\boldsymbol{v} \sim \mathcal{V}_{\text{yes}}$ is the same as the distribution of $(f(x) : x \in X)$ when $f \sim \mathcal{E}_{\text{yes}}(M)$. This finishes the proof of Equation (9), and concludes our reduction argument.

## 5 A LOWER BOUND ON THE NON-ADAPTIVE QUERY COMPLEXITY OF SSSQ

We will prove Lemma 9 by first giving a reduction from an even simpler algorithmic task, which we describe next in Section 5.1. We will then prove a lower bound for the simpler task in Section 5.2.

### 5.1 Set-Size-Element-Queries (SSEQ)

Recall the parameters $m, p, p_{\text{no}}$, and $\epsilon$ and the two distributions $\mathcal{A}_{\text{yes}}$ and $\mathcal{A}_{\text{no}}$ used in the definition of problem SSSQ. We now introduce a simpler algorithmic task called the SSEQ problem using the same parameters and distributions. As in the SSSQ problem, the goal is to distinguish between the case in which a hidden subset $A$ is drawn from $\mathcal{A}_{\text{yes}}$ or from $\mathcal{A}_{\text{no}}$.

Let $A$ be a subset of $[m]$ hidden in an oracle. An algorithm accesses the oracle to tell whether it is drawn from $\mathcal{A}_{\text{yes}}$ or $\mathcal{A}_{\text{no}}$. The difference between SSSQ and SSEQ is the way $A$ is accessed. In SSEQ, an algorithm $\text{Alg}' = (h, \ell)$ submits a vector $\ell = (\ell_1, \ldots, \ell_m)$ of nonnegative integers.

— On receiving $\ell$, the oracle returns a random response vector $\boldsymbol{b} \in \{0,1\}^m$, where each entry $\boldsymbol{b}_i$ is distributed independently as follows: if $i \notin A$ then $\boldsymbol{b}_i = 0$, and if $i \in A$, then

$$\boldsymbol{b}_i = \begin{cases} 1 & \text{with probability } \lambda(\ell_i) \\ 0 & \text{with probability } 1 - \lambda(\ell_i) \end{cases}, \qquad \text{where } \lambda(\ell_i) = 1 - \left(1 - \frac{\epsilon}{\sqrt{n}}\right)^{\ell_i}.$$

Equivalently, for each $i \in A$, the oracle independently flips $\ell_i$ coins, each of which is 1 with probability $\epsilon/\sqrt{n}$, and at the end returns $\boldsymbol{b}_i = 1$ to the algorithm if and only if at least one of the coins is 1. Thus, we refer to each $\ell_i$ as $\ell_i$ *element-queries* for the $i$th element.

— After receiving the vector $\boldsymbol{b}$ from the oracle, $\text{Alg}'$ returns the value $h(\boldsymbol{b}) \in \{\text{“yes,” “no”}\}$. Here $h$ is a deterministic map from $\{0,1\}^m$ to $\{\text{“yes,” “no”}\}$.

Similar to before, the performance of $\text{Alg}'$ is measured by its query complexity and its advantage:

— The query complexity of $\text{Alg}' = (h, \ell)$ is defined as $\|\ell\|_1 = \sum_{i=1}^m \ell_i$. For its advantage, we let $\mathcal{B}_{\text{yes}}$ denote the distribution of response vectors $\boldsymbol{b}$ to query $\ell$ when $\mathbf{A} \sim \mathcal{A}_{\text{yes}}$, and $\mathcal{B}_{\text{no}}$ denote the distribution when $\mathbf{A} \sim \mathcal{D}_{\text{no}}$. The advantage of $\text{Alg}' = (h, \ell)$ is then defined as

$$\Pr_{\boldsymbol{b} \sim \mathcal{B}_{\text{yes}}}\Big[h(\boldsymbol{b}) = \text{“yes”}\Big] - \Pr_{\boldsymbol{b} \sim \mathcal{B}_{\text{no}}}\Big[h(\boldsymbol{b}) = \text{“yes”}\Big].$$

*Remark 11.* It is worth pointing out (we will use it later) that the highest possible advantage over all deterministic maps $h$ is a monotonically non-decreasing function of the coordinates of $\ell$. To see this, let $A$ be the underlying set and let $\ell$ and $\ell'$ be two vectors with $\ell_i \le \ell'_i$ for every $i \in [m]$. Let $\boldsymbol{b}$ and $\boldsymbol{b}'$ be the random vectors returned by the oracle upon $\ell$ and $\ell'$. Then we can define $\boldsymbol{b}^*$ using $\boldsymbol{b}'$ as follows: $\boldsymbol{b}^*_i = 0$ if $\boldsymbol{b}'_i = 0$; otherwise when $\boldsymbol{b}'_i = 1$, we set

$$\boldsymbol{b}^*_i = \begin{cases} 1 & \text{with probability } \lambda(\ell_i)/\lambda(\ell'_i) \\ 0 & \text{with probability } 1 - \lambda(\ell_i)/\lambda(\ell'_i) \end{cases}.$$

One can easily verify that the distribution of $\boldsymbol{b}$ is exactly the same as the distribution of $\boldsymbol{b}^*$. Hence there is a randomized map $h'$ such that the advantage of $(h', \ell')$ is at least as large as the highest possible advantage achievable using $\ell$. The remark now follows by our earlier observation in Remark 8 that the highest possible advantage using $\ell'$ is always achieved by a deterministic $h'$.

The following lemma reduces the proof of Lemma 9 to proving a lower bound for SSEQ.

Lemma 12. *Given any deterministic and non-adaptive algorithm* Alg $= (g, T)$ *for* SSSQ*, there is a deterministic and non-adaptive algorithm* Alg$' = (h, \ell)$ *for* SSEQ *with the same query complexity as* Alg *and advantage at least as large as that of* Alg.

Proof. We show how to construct Alg$' = (h, \ell)$ from Alg $= (g, T)$, where $h$ is a randomized map, such that Alg$'$ has exactly the same query complexity and advantage as those of Alg. The lemma then follows from the observation we made earlier in Remark 8.

We define $\ell$ first. Given $T = (T_1, \ldots, T_d)$ for some $d \geq 1$, $\ell = (\ell_1, \ldots, \ell_m)$ is defined as

$$\ell_j = \Big|\{i \in [d] : j \in T_i\}\Big|.$$

So $\|\ell\|_1 = \sum_{i=1}^{d} |T_i|$. Recall from Section 4 that $\mathcal{V}_{\text{yes}}$ and $\mathcal{V}_{\text{no}}$ denote the distributions supported on responses $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d)$ to $T$ in SSSQ when $\mathbf{A} \sim \mathcal{A}_{\text{yes}}$ and $\mathbf{A} \sim \mathcal{A}_{\text{no}}$, respectively. To define $h$, we describe a randomized procedure $P$ that, given any $b \in \{0,1\}^m$, outputs a sequence of random vectors $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d)$, which simulates $\mathcal{V}_{\text{yes}}$ if $\boldsymbol{b} \sim \mathcal{B}_{\text{yes}}$ and $\mathcal{V}_{\text{no}}$ if $\boldsymbol{b} \sim \mathcal{B}_{\text{no}}$. In other words, we define $P$ below and prove the following claim:

Claim 13. *If* $\boldsymbol{b} \sim \mathcal{B}_{\text{yes}}$ *(or* $\mathcal{B}_{\text{no}}$*), then* $P(\boldsymbol{b})$ *is distributed the same as* $\mathcal{V}_{\text{yes}}$ *(or* $\mathcal{V}_{\text{no}}$*, respectively).*

Assuming Claim 13, we can set $h = g \circ P$ and the advantage of Alg$'$ would be the same as that of Alg. In the rest of the proof, we describe the randomized procedure $P$ and prove Claim 13.

Given $b \in \{0,1\}^m$, $P$ outputs a sequence of random vectors $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d)$ as follows:

—If $b_j = 0$, then for each $i \in [d]$ with $j \in T_i$, $P$ sets $\boldsymbol{v}_{i,j} = 0$.
—If $b_j = 1$ (this implies that $\ell_j > 0$ and $j \in T_i$ for some $i \in [d]$), $P$ sets $(\boldsymbol{v}_{i,j} : i \in [d], j \in T_i)$ to be a length-$r$, where $r = |\{i \in [d] : j \in T_i\}|$, binary string in which each bit is independently 1 with probability $\epsilon/\sqrt{n}$ and 0 with probability $1 - \epsilon/\sqrt{n}$, conditioned on its not being $0^r$.

Proof of Claim 13. It suffices to prove that, fixing any $A \subseteq [m]$ as the underlying set hidden in the oracle, the distribution of $\boldsymbol{v}$ is the same as that of $P(\boldsymbol{b})$. The claim then follows since in the definitions of both $\mathcal{B}_{\text{yes}}$ and $\mathcal{V}_{\text{yes}}$ (or $\mathcal{B}_{\text{no}}$ and $\mathcal{V}_{\text{no}}$), $A$ is drawn from $\mathcal{A}_{\text{yes}}$ (or $\mathcal{A}_{\text{no}}$, respectively).

Consider a sequence $v$ of $d$ vectors $v_1, \ldots, v_d$ with $v_i \in \{0,1\}^{T_i}$ for each $i \in [d]$, and let

$$n_{j,1} = |\{i \in [d] : j \in T_i \text{ and } v_{i,j} = 1\}| \quad \text{and} \quad n_{j,0} = |\{i \in [d] : j \in T_i \text{ and } v_{i,j} = 0\}|,$$

for each $j \in [m]$. Then the $\boldsymbol{v}$ returned by the oracle (in SSSQ) is equal to $v$ with probability:

$$\mathbf{1}\Big[\forall j \notin A,\, n_{j,1} = 0\Big] \cdot \prod_{j \in A} \left(\frac{\epsilon}{\sqrt{n}}\right)^{n_{j,1}} \left(1 - \frac{\epsilon}{\sqrt{n}}\right)^{n_{j,0}}, \tag{11}$$

since all coordinates $\boldsymbol{v}_{i,j}$ are independent. (Here $\mathbf{1}$ denotes the indicator function, so $\mathbf{1}[E]$ is 1 if event $E$ holds and is 0 otherwise.) On the other hand, the probability of $P(\boldsymbol{b}) = v$ is

$$\mathbf{1}\Big[\forall j \notin A,\, n_{j,1} = 0\Big] \cdot \prod_{j \in A} \left(\mathbf{1}\Big[n_{j,0} = \ell_j\Big] \cdot \left(1 - \frac{\epsilon}{\sqrt{n}}\right)^{\ell_j} + \mathbf{1}\Big[n_{j,1} \geq 1\Big] \cdot \left(\frac{\epsilon}{\sqrt{n}}\right)^{n_{j,1}} \left(1 - \frac{\epsilon}{\sqrt{n}}\right)^{n_{j,0}}\right),$$

which is exactly the same as the probability of $\boldsymbol{v} = v$ in Equation (11). □

This finishes the proof of Lemma 12. □

## 5.2 A Lower Bound for SSEQ

We prove the following lower bound for SSEQ, from which Lemma 9 follows:

Lemma 14. *Any deterministic, non-adaptive* Alg$'$ *for* SSEQ *with advantage at least* $2/3$ *satisfies*

$$\|\ell\|_1 > s \overset{\text{def}}{=} \frac{n^{3/2}}{\epsilon \cdot \log^3 n \cdot \log^2(n/\epsilon)}.$$

Proof. Assume for contradiction that there is an algorithm $\text{Alg}' = (h, \ell)$ with $\|\ell\|_1 \le s$ and advantage at least $2/3$. Let $\ell^*$ be the vector obtained from $\ell$ by rounding each positive $\ell_i$ to the smallest power of 2 that is at least as large as $\ell_i$ (and taking $\ell_i^* = 0$ if $\ell_i = 0$). From Remark 11, there must be a map $h^*$ such that $(h^*, \ell^*)$ also has advantage at least $2/3$ but now we have (1) $\|\ell^*\|_1 \le 2s$ and (2) every positive entry of $\ell^*$ is a power of 2. Below we abuse notation and still use $\text{Alg}' = (h, \ell)$ to denote $(h^*, \ell^*)$: $\text{Alg}' = (h, \ell)$ satisfies $\|\ell\|_1 \le 2s$, every positive entry of $\ell$ is a power of 2, and has advantage at least $2/3$. We obtain a contradiction below by showing that any such $\ell$ can only have an advantage of $o(1)$.

Let $L = \lceil \log(2s) \rceil = O(\log(n/\epsilon))$. Given that $\|\ell\|_1 \le 2s$ we can partition $\{i \in [m] : \ell_i > 0\}$ into $L + 1$ bins $C_0, \ldots, C_L$, where bin $C_j$ contains those coordinates $i \in [m]$ with $\ell_i = 2^j$. We may make two further assumptions on $\text{Alg}' = (h, \ell)$ that will simplify the lower bound proof:

— We may reorder the entries in decreasing order and assume without loss of generality that

$$\ell = \left( \underbrace{2^L, \ldots, 2^L}_{c_L}, \underbrace{2^{L-1}, \ldots, 2^{L-1}}_{c_{L-1}}, \ldots, \underbrace{1, \ldots, 1}_{c_0}, 0, \ldots, 0 \right), \tag{12}$$

where $c_j = |C_j|$ satisfies $\sum_j c_j \cdot 2^j \le 2s$. This is without loss of generality since $\mathcal{A}_{\text{yes}}$ and $\mathcal{A}_{\text{no}}$ are symmetric in the coordinates (and so are $\mathcal{B}_{\text{yes}}$ and $\mathcal{B}_{\text{no}}$).

— For the same reason, we may assume that the map $h(b)$ depends only on the number of 1's of $b$ in each set $C_j$, which we refer to as the *summary* $S(b)$ of $b$:

$$S(b) \stackrel{\text{def}}{=} \left( \|b_{|C_L}\|_1, \|b_{|C_{L-1}}\|_1, \ldots, \|b_{|C_0}\|_1 \right) \in \mathbb{Z}_{\ge 0}^{L+1}.$$

To see that this is without loss of generality, consider a randomized procedure $P$ that, given $b \in \{0,1\}^m$, applies an independent random permutation over the entries of $C_j$ for each bin $j \in [0 : L]$. One can verify that the random map $h' = h \circ P$ only depends on the summary $S(b)$ of $b$ but achieves the same advantage as $h$.

Given a query $\ell$ as in (12), we define $\mathcal{S}_{\text{yes}}$ to be the distribution of $S(\boldsymbol{b})$ for $\boldsymbol{b} \sim \mathcal{B}_{\text{yes}}$ (recall that $\mathcal{B}_{\text{yes}}$ is the distribution of the vector $\boldsymbol{b}$ returned by the oracle upon the query $\ell$ when $\mathbf{A} \sim \mathcal{A}_{\text{yes}}$). Similarly, we define $\mathcal{S}_{\text{no}}$ as the distribution of $S(\boldsymbol{b})$ for $\boldsymbol{b} \sim \mathcal{B}_{\text{no}}$. As $h$ only depends on the summary the advantage is at most $d_{\text{TV}}(\mathcal{S}_{\text{yes}}, \mathcal{S}_{\text{no}})$, which we upper bound below by $o(1)$.

From the definition of $\mathcal{B}_{\text{yes}}$ (or $\mathcal{B}_{\text{no}}$, respectively) and the fact that $\mathcal{A}_{\text{yes}}$ (or $\mathcal{A}_{\text{no}}$, respectively) is symmetric over the $m$ coordinates, we have that the $L + 1$ entries of $\mathcal{S}_{\text{yes}}$ (of $\mathcal{S}_{\text{no}}$, respectively) are mutually independent, and that their entries for each $C_j$, $j \in [0 : L]$, are distributed as $\text{Bin}(c_j, p\lambda_j)$ (as $\text{Bin}(c_j, p_{\text{no}}\lambda_j)$, respectively), where we have $\lambda_j = 1 - (1 - (\epsilon/\sqrt{n}))^{2^j}$.

To prove that $d_{\text{TV}}(\mathcal{S}_{\text{yes}}, \mathcal{S}_{\text{no}}) = o(1)$ and achieve the desired contradiction, we will give upper bounds on the total variation distance between their $C_j$-entries for each $j \in \{0, \ldots, L\}$.

Claim 15. *For every $j \in [0 : L]$, we have $d_{\text{TV}}(\mathbf{X}, \mathbf{Y}) \le o(1/L)$, where $\mathbf{X} \sim \text{Bin}(c_j, p\lambda_j)$ and $\mathbf{Y} \sim \text{Bin}(c_j, p_{\text{no}}\lambda_j)$.*

We delay the proof of Claim 15, but assuming it we may simply apply the following well-known proposition to conclude that $d_{\text{TV}}(\mathcal{S}_{\text{yes}}, \mathcal{S}_{\text{no}}) = o(1)$.

Proposition 16 (Subadditivity of Total Variation Distance). *Let $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_k)$ and $\mathbf{Y} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_k)$ be two tuples of independent random variables. Then $d_{\text{TV}}(\mathbf{X}, \mathbf{Y}) \le \sum_{i=1}^k d_{\text{TV}}(\mathbf{X}_i, \mathbf{Y}_i)$.*

This gives us a contradiction and finishes the proof of Lemma 14. □

Below we prove Claim 15.

Proof of Claim 15. Consider any fixed $j \in [0:L]$. The claim is trivial when $c_j = 0$ so we assume below that $c_j > 0$.

Let $r_j = p\lambda_j$ and $x_j = \log n \cdot \lambda_j/\sqrt{n}$. Then $\mathbf{X} \sim \mathrm{Bin}(c_j, r_j)$ and $\mathbf{Y} \sim \mathrm{Bin}(c_j, r_j + x_j)$. As indicated in Equation (2.15) of Reference [1], Equation (15) of Reference [39] gives

$$d_{\mathrm{TV}}(\mathbf{X}, \mathbf{Y}) \le O\left(\frac{\tau_j(x_j)}{(1-\tau_j(x_j))^2}\right), \quad \text{where} \quad \tau_j(x_j) \overset{\text{def}}{=} x_j\sqrt{\frac{c_j+2}{2r_j(1-r_j)}}, \tag{13}$$

whenever $\tau_j(x_j) < 1$. Substituting for $x_j$ and $r_j$, we have (using $c_j \ge 1$, $r_j \le 1/2$ and $p = 1/2$)

$$\tau_j(x_j) = O\left(\frac{\log n \cdot \lambda_j}{\sqrt{n}} \cdot \sqrt{\frac{c_j}{r_j}}\right) = O\left(\log n \cdot \sqrt{\frac{\lambda_j \cdot c_j}{n}}\right) = O\left(\frac{1}{L} \cdot \sqrt{\frac{n^{1/2} \cdot \lambda_j}{2^j \cdot \epsilon \cdot \log n}}\right),$$

where the last inequality follows from

$$c_j \cdot 2^j \le 2s \le O\left(\frac{n^{3/2}}{\epsilon \cdot \log^3 n \cdot L^2}\right).$$

Finally, note that (using $1 - x > e^{-2x}$ for small positive $x$ and $1 - x \le e^{-x}$ for all $x$)

$$1 - \lambda_j = \left(1 - \frac{\epsilon}{\sqrt{n}}\right)^{2^j} \ge \left(e^{-2\epsilon/\sqrt{n}}\right)^{2^j} = e^{-2^{j+1}\epsilon/\sqrt{n}} \ge 1 - O(2^j\epsilon/\sqrt{n}),$$

so that $\frac{\sqrt{n}\cdot\lambda_j}{2^j\cdot\epsilon} = O(1)$. This implies $\tau_j(x_j) = o(1/L)$. The claim then follows from Equation (13). □

## APPENDIX

## A PROOF OF THEOREM 1 ASSUMING THEOREM 2

We prove the following claim in Appendix A.1.

Claim 17. *Let $\epsilon(n)$ be a function that satisfies $2^{-n} \le \epsilon(n) \le 1/5$ for sufficiently large $n$. Then any non-adaptive algorithm that accepts the all-0 function with probability at least $5/6$ and rejects every function that is $\epsilon$-far from $(n-1)$-juntas with probability at least $5/6$ must make $\Omega(1/\epsilon)$ queries.*

Next let $k(n)$ and $\epsilon(n)$ be the pair of functions from the statement of Theorem 1. We consider a sufficiently large $n$ (letting $k = k(n)$ and $\epsilon = \epsilon(n)$ below) and separate the proof into two cases:

$$2^{-(2\alpha-1)k/(2\alpha)} \le \epsilon \le 1/6 \quad \text{and} \quad 2^{-n} \le \epsilon < 2^{-(2\alpha-1)k/(2\alpha)}.$$

For the first case, if $k = O(1)$ then the bound we aim for is simply $\widetilde{\Omega}(1/\epsilon)$, which follows trivially from Claim 17 (since $k \le \alpha n < n - 1$ and the all-0 function is a $k$-junta). Otherwise, we combine the following reduction with Theorem 2: any $\epsilon$-tester for $k$-juntas over $n$-variable functions can be used to obtain an $\epsilon$-tester for $k$-juntas over $(k/\alpha)$-variable functions. This can be done by adding $n - k/\alpha$ dummy variables to any $(k/\alpha)$-variable function to make the number of variables $n$ (as $k \le \alpha n$). The lower bound then follows from Theorem 2 since $\alpha$ is a constant. For the second case, the lower bound claimed in Theorem 1 is $\widetilde{\Omega}(1/\epsilon)$, which follows again from Claim 17. This concludes the proof of Theorem 1 given Theorem 2 and Claim 17.

### A.1 Proof of Claim 17

Let $C$ be a sufficiently large constant. We prove Claim 17 by considering two cases:

$$\epsilon \ge \frac{C\log n}{2^n} \quad \text{and} \quad \epsilon < \frac{C\log n}{2^n}.$$

For the first case of $2^n\epsilon \geq C \log n$, we use $\mathcal{D}_1$ to denote the following distribution over $n$-variable Boolean functions: to draw $\boldsymbol{g} \sim \mathcal{D}_1$, independently for each $x \in \{0,1\}^n$ the value of $\boldsymbol{g}(x)$ is set to 0 with probability $1 - 3\epsilon$ (recall that $\epsilon \leq 1/5$) and 1 with probability $3\epsilon$.

We prove the following lemma for the distribution $\mathcal{D}_1$:

Lemma 18. *With probability at least* $1 - o(1)$, $\boldsymbol{g} \sim \mathcal{D}_1$ *is* $\epsilon$*-far from every* $(n-1)$*-junta.*

Proof. Note that every $(n-1)$-junta is such that for some $i \in [n]$, the function does not depend on the $i$-th variable; we refer to such a function as a type-$i$ junta. An easy lower bound for the distance from a function $g$ to all type-$i$ juntas is the number of $g$-bichromatic edges $(x, x^{(i)})$ divided by $2^n$. When $\boldsymbol{g} \sim \mathcal{D}_1$ each edge $(x, x^{(i)})$ is independently $\boldsymbol{g}$-bichromatic with probability $6\epsilon(1-3\epsilon) \geq 12\epsilon/5$ (as $\epsilon \leq 1/5$). Thus when $2^n\epsilon \geq C \log n$, the expected number of such edges is at least

$$2^{n-1} \cdot (12\epsilon/5) \geq (6/5) \cdot 2^n\epsilon \geq (6/5) \cdot C \log n.$$

Using a Chernoff bound, the probability of having fewer than $2^n\epsilon$ bichromatic edges along direction $i$ is at most $1/n^2$ when $C$ is sufficiently large. The lemma follows from a union bound over $i$. $\square$

As a result, when $2^n\epsilon \geq C \log n$, if $\mathcal{A}$ is a non-adaptive algorithm with the property described in Claim 17, then $\mathcal{A}$ must satisfy

$$\mathbf{Pr}\Big[\mathcal{A} \text{ accepts the all-0 function}\Big] - \mathop{\mathbf{Pr}}_{\boldsymbol{g} \sim \mathcal{D}_1}[\mathcal{A} \text{ accepts } \boldsymbol{g}] \geq 2/3 - o(1).$$

But any such non-adaptive algorithm must make $\Omega(1/\epsilon)$ queries as otherwise with high probability all of its queries to $\boldsymbol{g} \sim \mathcal{D}_1$ would be answered 0, and hence its behavior would be the same as if it were running on the all-0 function.

Finally, we work on the case when $1 \leq 2^n\epsilon = O(\log n)$. The proof is the same except that we let $\boldsymbol{g}$ be drawn from $\mathcal{D}_2$, which we define to be the distribution where all entries of $\boldsymbol{g} \sim \mathcal{D}_2$ are 0 except for exactly $2^n\epsilon$ of them picked uniformly at random. The claim follows from the following lemma:

Lemma 19. *With probability at least* $1 - o(1)$, $\boldsymbol{g} \sim \mathcal{D}_2$ *is* $\epsilon$*-far from every* $(n-1)$*-junta.*

Proof. This follows from the observation that, with probability $1 - o(1)$, no two points picked form an edge. When this occurs, we have $2^n\epsilon$ bichromatic edges along the $i$th direction for all $i$. $\square$

## REFERENCES

[1] José A. Adell and Pedro Jodrá. 2006. Exact Kolmogorov and total variation distances between some familiar discrete distributions. *J. Inequal. Appl.* 2006, 1 (2006), 1–8.

[2] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron. 2005. Testing Reed-Muller codes. *IEEE Trans. Inform. Theor.* 51, 11 (2005), 4032–4039.

[3] Roksana Baleshzar, Meiram Murzabulatov, Ramesh Krishnan S. Pallavoor, and Sofya Raskhodnikova. 2016. Testing unateness of real-valued functions. *CoRR* abs/1608.07652 (2016).

[4] A. Belovs and E. Blais. 2016. A polynomial lower bound for testing monotonicity. In *Proceedings of the 48th ACM Symposium on Theory of Computing (STOC'16).* ACM, 1021–1032.

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

[6] Arnab Bhattacharyya, Swastik Kopparty, Grant Schoenebeck, Madhu Sudan, and David Zuckerman. 2010. Optimal testing of Reed-Muller codes. In *51st Annual IEEE Symposium on Foundations of Computer Science (FOCS'10).* IEEE Computer Society, 488–497.

[7] Eric Blais. 2008. Improved bounds for testing juntas. In *Proceedings of the 12th International Workshop on Randomization (RANDOM'08).* Springer, 317–330.

[8] Eric Blais. 2009. Testing juntas nearly optimally. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC'09).* ACM, 151–158.

[9] Eric Blais, Joshua Brody, and Kevin Matulef. 2011. Property testing lower bounds via communication complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity (CCC'11).* IEEE Computer Society, 210–220.

[10] Eric Blais and Daniel M. Kane. 2012. Tight bounds for testing $k$-linearity. In *Proceedings of the 16th International Workshop on Randomization (RANDOM'12)*. Springer, 435–446.

[11] M. Blum, M. Luby, and R. Rubinfeld. 1993. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.* 47, 3 (1993), 549–595.

[12] Harry Buhrman, David García-Soriano, Arie Matsliah, and Ronald de Wolf. 2013. The non-adaptive query complexity of testing k-parities. *Chicago J. Theor. Comput. Sci.* Article 06 (2013), 1–11.

[13] Deeparnab Chakrabarty and C. Seshadhri. 2013. A $o(n)$ monotonicity tester for boolean functions over the hypercube. In *Proceedings of the 45th ACM Symposium on Theory of Computing (STOC'13)*. ACM, 411–418.

[14] Deeparnab Chakrabarty and C. Seshadhri. 2016. A $\widetilde{O}(n)$ non-adaptive tester for unateness. *CoRR* abs/1608.06980 (2016).

[15] Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. 2015. Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. In *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC'15)*. ACM, 519–528.

[16] X. Chen, R. Servedio, and L.-Y. Tan. 2014. New algorithms and lower bounds for testing monotonicity. In *Proceedings of the 55th IEEE Symposium on Foundations of Computer Science (FOCS'14)*. IEEE Computer Society, 286–295.

[17] H. Chockler and D. Gutfreund. 2004. A lower bound for testing juntas. *Inform. Process. Lett.* 90, 6 (2004), 301–305.

[18] I. Diakonikolas, H. Lee, K. Matulef, K. Onak, R. Rubinfeld, R. Servedio, and A. Wan. 2007. Testing for concise representations. In *Proceedings of the 48th Annual Symposium on Computer Science (FOCS'07)*. IEEE Computer Society, 549–558.

[19] D. Dubhashi and A. Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, Cambridge.

[20] E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky. 2004. Testing juntas. *J. Comput. Syst. Sci.* 68, 4 (2004), 753–787.

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

[22] Peter Frankl. 1983. On the trace of finite sets. *J. Comb. Theor. Ser. A* 34, 1 (1983), 41–45.

[23] O. Goldreich (Ed.). 2010. *Property Testing: Current Research and Surveys*. LNCS 6390. Springer.

[24] Oded Goldreich. 2017. *Introduction to Property Testing*. Cambridge University Press.

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

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

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

[28] Sergiu Hart. 1976. A note on the edges of the $n$-cube. *Disc. Math.* 14, 2 (1976), 157–163.

[29] Subhash Khot, Dor Minzer, and Muli Safra. 2015. On monotonicity testing and Boolean isoperimetric type theorems. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS'15)*. IEEE Computer Society, 52–58.

[30] Subhash Khot and Igor Shinkar. 2016. An O(n) queries adaptive tester for unateness. In *Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 37:1–37:7.

[31] J. H. Lindsey. 1964. Assignment of numbers to vertices. *Amer. Math. Monthly* 71, 5 (1964), 508–516.

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

[33] Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A. Servedio. 2009. Testing ±1-weight halfspace. In *Proceedings of the 13th International Workshop on Randomization (RANDOM'09)*. Springer, 646–657.

[34] M. Parnas, D. Ron, and A. Samorodnitsky. 2002. Testing basic boolean formulae. *SIAM J. Disc. Math.* 16, 1 (2002), 20–46.

[35] D. Ron. 2008. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning* 1, 3 (2008), 307–402.

[36] D. Ron. 2010. Algorithmic and analysis techniques in property testing. *Found. Trends Theor. Comput. Sci.* 5, 2 (2010), 73–205.

[37] D. Ron and R. Servedio. 2013. Exponentially improved algorithms and lower bounds for testing signed majorities. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'13)*. SIAM, 1319–1336.

[38] Dana Ron and Gilad Tsur. 2011. *Testing Computability by Width-Two OBDDs*. Technical Report 11(041). Electronic Colloquium on Computational Complexity (ECCC). available at http://eccc.hpi-web.de/report/2011/041/.

[39] B. Roos. 2000. Binomial approximation to the Poisson binomial distribution: The Krawtchouk expansion. *Theory Probab. Appl.* 45, 2 (2000), 328–344.

[40] Rocco Servedio, Li-Yang Tan, and John Wright. 2015. Adaptivity helps for testing juntas. In *Proceedings of the 30th Conference on Computational Complexity (CCC'15)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 264–279.

Received September 2017; revised April 2018; accepted July 2018