Efficient Sample Extractors for Juntas with Applications

Sourav Chakraborty [1], David García-Soriano [2], and Arie Matsliah [3]

[1] Chennai Mathematical Institute, India.
[2] CWI Amsterdam, Netherlands.
[3] IBM Research and Technion, Haifa, Israel.

Abstract. We develop a query-efficient sample extractor for juntas, that is, a probabilistic algorithm that can simulate random samples from the core of a k-junta $f : \{0,1\}^n \to \{0,1\}$ given oracle access to a function $f' : \{0,1\}^n \to \{0,1\}$ that is only close to f. After a preprocessing step, which takes $\widetilde{O}(k)$ queries, generating each sample to the core of f takes only one query to f'.

We then plug in our sample extractor in the "testing by implicit learning" framework of Diakonikolas et al. [DLM$^+$07], improving the query complexity of testers for various Boolean function classes. In particular, for some of the classes considered in [DLM$^+$07], such as s-term DNF formulas, size-s decision trees, size-s Boolean formulas, s-sparse polynomials over $\mathbb{F}_2$, and size-s branching programs, the query complexity is reduced from $\widetilde{O}(s^4/\epsilon^2)$ to $\widetilde{O}(s/\epsilon^2)$. This shows that using the new sample extractor, testing by implicit learning can lead to testers having better query complexity than those tailored to a specific problem, such as the tester of Parnas et al. [PRS02] for the class of monotone s-term DNF formulas.

In terms of techniques, we extend the tools used in [CGM11] for testing function isomorphism to juntas. Specifically, while the original analysis in [CGM11] allowed query-efficient noisy sampling from the core of any k-junta f, the one presented here allows similar sampling from the core of the *closest* k-junta to f, even if f is not a k-junta but just close to being one. One of the observations leading to this extension is that the junta tester of Blais [Bla09], based on which the aforementioned sampling is achieved, enjoys a certain weak form of tolerance.

Keywords: property testing, sample extractors, implicit learning

1 Introduction

Suppose we wish to test for the property defined by a class $\mathcal{C}$ of Boolean functions over $\{0,1\}^n$; that is, we aim to distinguish the case $f \in \mathcal{C}$ from the case $\mathrm{dist}(f, \mathcal{C}) \geq \epsilon$. The class is parameterized by a "size" parameter s (e.g. the class of DNFs with s terms, or circuits of size s) and, as usual, our goal is to minimize the number of queries made to f. In particular we strive for query complexity independent of n whenever possible.

The main observation underlying the "testing by implicit learning" paradigm of Diakonikolas et al. [DLM$^+$07] (see also [Ser10], [DLM$^+$08], [GOS$^+$09]) is that a large number of interesting classes $\mathcal{C}$ can be well approximated by (relatively) small juntas also belonging to $\mathcal{C}$.

The prototypical example is obtained by taking for $\mathcal{C}$ the class of s-term DNFs. Let $\tau > 0$ be an approximation parameter (which for our purpose should be thought of as polynomial in ϵ/s). Any DNF term involving more than $\log(s/\tau)$ variables may be removed from f while affecting only a τ/s fraction of its values; hence, removing all of them results in an s-term DNF f' that is τ-close to f and *depends on only* $s\log(s/\tau)$ *variables* (equivalently, f' is a $s\log(s/\tau)$-junta). Let $\mathsf{Jun}_{[k]}$ denote the subset of (k-junta) functions $\{0,1\}^n \to \{0,1\}$ that depend only on the *first* k variables. Since the class $\mathcal{C}$ is isomorphism-invariant (closed under permutations of the variables), the foregoing observation can be rephrased as follows: for any $k \geq s\log(s/\tau)$, the subclass $\mathcal{C}_{[k]} \triangleq \mathcal{C} \cap \mathsf{Jun}_{[k]}$ is such that every $f \in \mathcal{C}$ is τ-close to being isomorphic to some $g \in \mathcal{C}_{[k]}$ (in short, $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \tau$).

On the other hand, for every f such that $\mathrm{dist}(f, \mathcal{C}) = \mathrm{distiso}(f, \mathcal{C}) \geq \epsilon$ it also holds that $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \geq \epsilon$, since $\mathcal{C}_{[k]} \subseteq \mathcal{C}$. Hence, to solve the original problem, all we need is to differentiate between the two cases (i) $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \tau$ and (ii) $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \geq \epsilon$.

Let us denote by f^* the k-junta that is closest to f; f^* can be identified with its *core*, i.e. the Boolean function $\mathsf{core}_k(f^*) : \{0,1\}^k \to \{0,1\}$ obtained from f^* by dropping its irrelevant variables. If we could somehow manage to get random samples of the form $(x, \mathsf{core}_k(f^*)(x)) \in \{0,1\}^k \times \{0,1\}$, we could use standard learning algorithms to identify an element $g \in \mathcal{C}_{[k]}$ which is close to being isomorphic to f^* (if any), which would essentially allow us to differentiate between the aforementioned cases. The number of such samples required for this is roughly logarithmic in $|\mathcal{C}_{[k]}|$; we elaborate on this later.[4] An important observation is that the size of $\mathcal{C}_{[k]} \triangleq \mathcal{C} \cap \mathsf{Jun}_{[k]}$ is usually very small, even compared to the size of $\mathsf{Jun}_{[k]}$, which is 2^{2^k}. For instance, it is not hard to see that for the case of s-term DNFs, the size of $\mathcal{C}_{[k]}$ is bounded by $(2k)^k$, which is exponential in k, rather than doubly exponential.

It is a surprising fact that such samples from the core of f^* can indeed be efficiently obtained (with some noise), even though f is the only function we have access to. Even having query access to f^* itself would not seem to help much at first glance, since the location of the relevant variables of f^* is unknown to us, and cannot be found without introducing a dependence of n in the query complexity. It is in this step that our approach departs from that of [DLM$^+$07]. We mention next the two main differences that, when combined together, lead to better query complexity bounds.

The first difference is in the junta-testing part; both algorithms start with a junta tester to identify k disjoint subsets of variables (blocks), such that every "influential" variable of the function f being tested lies in one of these blocks.

[4] Issues of computational efficiency are usually disregarded here; however see [DLM$^+$08].

While [DLM$^+$07] use the tolerant version of the junta-tester of Fischer et al. [FKR$^+$02], we switch to the query-efficient junta tester of Blais [Bla09]. To make this step possible, we have to show that the tester from [Bla09] is sufficiently tolerant (the level of tolerance of the tester determines how large τ can be, which in turn determines how small k can be). The second (and the main) difference is in sample extraction - the actual process that obtains samples from the core of f^*. While in [DLM$^+$07] sampling is achieved via independence tests[5], applied to each of the identified blocks separately (which requires $\Omega(k)$ queries to f per sample), we use ideas from [CGM11] instead. The algorithm presented in [CGM11, Section 7] accomplishes this task in the (strict) case $f = f^*$ by making just *one* query to f. The bulk of this work is a proof that, when f is close enough to f^*, it is still possible to obtain each such sample using only one query to f (an overview of the proof is given in Section 4.1).

Organization In Section 2 we give the notation necessary for the formal statement of our results, which is done in Section 3. In Section 4 we give the proofs. (Some tools used in Section 4 are similar to those that appear in [CGM11], up to small tailoring needed before being applicable. We reproduce them in the Appendix to make this submission self-contained.) In Section 5 we also prove new query-complexity lower bounds for several classes of functions.

2 Notation

For any permutation $\pi : [n] \to [n]$ and $x \in \{0,1\}^n$, we define $\pi(x)$ as the map on n-bit strings that sends $x = x_1 \dots x_n \in \{0,1\}^n$ to $\pi(x) \triangleq x_{\pi(1)} \dots x_{\pi(n)} \in \{0,1\}^n$. If $f : \{0,1\}^n \to \{0,1\}$, we also denote by f^π the function $f^\pi(x) \equiv f(\pi(x))$.

Given $x \in \{0,1\}^n$, $A \subseteq [n]$ and $y \in \{0,1\}^{|A|}$, we denote by $x_{A \leftarrow y}$ an input obtained by taking x and substituting its values in A with y (according to the natural ordering of $[n]$).

For a function $f : \{0,1\}^n \to \{0,1\}$ and a set $A \subseteq [n]$, the *influence*[6] of f on A is

$$Inf_f(A) \triangleq \Pr_{x \in \{0,1\}^n,\ y \in \{0,1\}^{|A|}} \left[f(x) \neq f(x_{A \leftarrow y}) \right].$$

Here and throughout this paper, $x \in S$ under the probability symbol means that an element x is chosen uniformly at random from a set S.

A set $S \subseteq [n]$ is *relevant* with respect to f if $Inf_f(S) \neq 0$; an index (variable) $i \in [n]$ is relevant if $\{i\}$ is. A *k-junta* is a function g that has at most k relevant variables; equivalently, there is $S \in \binom{[n]}{k}$ such that $Inf_g([n] \setminus S) = 0$.

[5] Loosely speaking, these tests try to extract the values of the relevant variables of f^* by querying f on several inputs that are slightly perturbed (see [FKR$^+$02] for details).

[6] When $|A| = 1$, this value is half that of the most common definition of influence of one variable; for consistency we stick to the previous definition instead in this case as well. It also appears in the literature under the alternate name of *variation*.

Jun_k denotes the class of k-juntas (on n variables), and for $A \subseteq [n]$, Jun_A denotes the class of juntas with all relevant variables in A. In addition, given a function $f : \{0,1\}^n \to \{0,1\}$, we denote by $f^* : \{0,1\}^n \to \{0,1\}$ the k-junta that is closest to f (if there are several k-juntas that are equally close, break ties using some arbitrarily fixed scheme). Clearly, if f is itself a k-junta then $f^* = f$.

Given a k-junta $f : \{0,1\}^n \to \{0,1\}$ we define $\mathsf{core}_k(f) : \{0,1\}^k \to \{0,1\}$ to be the restriction of f to its relevant variables (where the variables are placed according to the natural order). In case f has fewer than k relevant variables, $\mathsf{core}_k(f)$ is extended to a function $\{0,1\}^k \to \{0,1\}$ arbitrarily (by adding dummy variables).

Unless explicitly mentioned otherwise, $\mathcal{C}$ will always denote a class of functions $f : \{0,1\}^n \to \{0,1\}$ that is closed under permutation of variables; that is, for any f and permutation π of $[n]$, $f \in \mathcal{C}$ if and only if $f^\pi \in \mathcal{C}$. For any $k \in \mathbb{N}$, let $\mathcal{C}_{[k]}$ denote the subclass $\mathcal{C} \cap \mathsf{Jun}_{[k]}$. Note that since $\mathcal{C}$ is closed under permutations of variables, $\mathcal{C}_{[k]}$ is closed under permutations of the first k variables. With a slight abuse of notation, we may use $\mathsf{core}_k(\mathcal{C}_{[k]})$ to denote the class $\{\mathsf{core}_k(f) : f \in \mathcal{C}_{[k]}\}$ of k-variable functions.

3 Results

3.1 Upper bounds

The main tool we develop here is the following:

Theorem 1. *Let $\epsilon > 0, k \in \mathbb{N}$ and let $\mathcal{C}_{[k]} \subseteq \mathsf{Jun}_{[k]}$ be a class closed under permutations of the first k variables. Let $\theta_1(k,\epsilon) = (\epsilon/2400)^6/(10^{26}k^{10}) = \mathrm{poly}(\epsilon/k)$. There is a randomized algorithm A_1 that given ϵ, k and oracle access to a function $f : \{0,1\}^n \to \{0,1\}$ does the following:*

- *if $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \le \theta_1(k,\epsilon)$, A_1 accepts with probability at least $7/10$;*
- *if $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \ge \epsilon$, A_1 rejects with probability at least $7/10$;*
- *A_1 makes $O\left(\frac{k}{\epsilon} + k\log k + \frac{1+\log|\mathcal{C}_{[k]}|}{\epsilon^2}\right)$ queries to f.*

Coupled with the prior discussion on testing by implicit learning, Theorem 1 also implies:

Corollary 1. *Let $\epsilon > 0$ and let $\mathcal{C}$ be an isomorphism-invariant class of Boolean functions. In addition, let $k \in \mathbb{N}$ be such that for every $f \in \mathcal{C}$, $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \le \theta_1(k,\epsilon)$. Then there is an algorithm that makes*

$$O\left(\frac{k}{\epsilon} + k\log k + \frac{1+\log|\mathcal{C}_{[k]}|}{\epsilon^2}\right) = O\left(\frac{k\log k + \log|\mathcal{C}_{[k]}|}{\epsilon^2}\right)$$

queries and satisfies:

- *if $f \in \mathcal{C}$, it accepts with probability at least $7/10$;*
- *if $\mathrm{dist}(f, \mathcal{C}) \ge \epsilon$, it rejects with probability at least $7/10$.*

To minimize the query complexity, we would like to pick k as small as possible, subject to the requirement of the theorem. Let $k^\star(\mathcal{C},\tau)$ be the smallest $k \in \mathbb{N}$ such that for every $f \in \mathcal{C}$, $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \le \tau$; intuitively, this condition means that $\mathcal{C}$ is τ-approximated by $\mathcal{C}_{[k]}$. We take from [DLM+07] the bounds on $k^\star = k^\star(\mathcal{C},\tau)$ and $|\mathcal{C}_{[k^\star]}|$ for the following classes of functions:

	$\mathcal{C}$ (class)	$k^\star \triangleq k^\star(\mathcal{C},\tau) \le$	$\lvert\mathcal{C}_{[k^\star]}\rvert \le$
1	s-term DNFs	$s\log(s/\tau)$	$(2s\log(s/\tau))^{s\log(s/\tau))}$
2	size-s Boolean formulae	$s\log(s/\tau)$	$(2s\log(s/\tau))^{s\log(s/\tau)+s}$
3	size-s Boolean circuits	$s\log(s/\tau)$	2^{2s^2+4s}
4	s-sparse polynomials over $\mathbb{F}_2$	$s\log(s/\tau)$	$(2s\log(s/\tau))^{s\log(s/\tau))}$
5	size-s decision trees	s	$(8s)^s$
6	size-s branching programs	s	$s^s(s+1)^{2s}$
7	functions with Fourier degree at most d	$d2^d$	$2^{d^2 2^{2d}}$

These bounds hold for any approximation parameter $\tau \ge 0$. But to make Corollary 1 applicable, we need to pick τ and k such that the (circular) inequalities $\tau \le \theta_1(k,\epsilon)$ and $k \ge k^\star(\mathcal{C},\tau)$ are satisfied.

For items $5, 6, 7$ setting $\tau = 0$ does the job; the reason these bounds are independent of τ is the fact that the corresponding classes contain only functions that actually are $k^\star$-juntas (rather than functions that can be well approximated by $k^\star$-juntas).

For the first 4 items we can set $\tau = \theta_1(s,\epsilon)^2$. It is easy to verify that this satisfies the foregoing pair of inequalities. Furthermore, since $\theta_1(s,\epsilon)$ is polynomial in ϵ/s, we get $k = O(s(\log s + \log 1/\epsilon))$. Plugging in the resulting values into Corollary 1, we obtain the following query-complexity bounds:

Class	This work	[DLM+07], [PRS02]$^{(*)}$
s-term DNFs, size-s Boolean formulae, s-sparse polynomials over $\mathbb{F}_2$, size-s decision trees, size-s branching programs	$\widetilde{O}(s/\epsilon^2)$	$\widetilde{O}(s^4/\epsilon^2)$
size-s Boolean circuits	$\widetilde{O}(s^2/\epsilon^2)$	$\widetilde{O}(s^6/\epsilon^2)$
functions with Fourier degree at most d	$\widetilde{O}(2^{2d}/\epsilon^2)$	$\widetilde{O}(2^{6d}/\epsilon^2)$
s-term monotone DNFs	$\widetilde{O}(s/\epsilon^2)$	$\widetilde{O}(s^2/\epsilon)^*$

3.2 Lower bounds

For several classes of functions mentioned above we also improve the query-complexity lower bounds. In particular, for size-s Boolean formulae, size-s branching programs, size-s Boolean circuits and s-sparse polynomials over $GF(2)$ we prove lower bounds of $s^{\Omega(1)}$ queries, for functions with Fourier degree d there is a lower bound of $\Omega(d)$ queries, and for s-term DNFs and size-s decision trees we prove lower bounds of $\Omega(\log s)$ queries. We also reprove (and hopefully simplify) some known lower bounds for other classes of functions. For details see Section 5.

4 Proof of Theorem 1

4.1 Overview

A key component of our algorithm is the nearly optimal junta tester of [Bla09]. This is a test to distinguish k-juntas from functions that are ϵ-far from being one, and has perfect completeness, i.e., never rejects a k-junta (see Section 4.4 for a more detailed description). The tester is not guaranteed to accept functions that are, say, $\epsilon/10$ close to juntas. We observe, however, that it enjoys a certain weak form of *tolerance*; roughly speaking, $\theta_1(k, \epsilon)$ is a measure of the amount of tolerance of said tester, i.e. how close f must be to a k-junta in order to guarantee it will be accepted with high probability. This is Lemma 7 in Section 4.4.

Our algorithm begins by calling the junta tester with parameter k. If f is $\theta_1(k, \epsilon)$-close to being a k-junta, the aforementioned tolerance implies that f is not rejected. (Note however that f may be $\theta_1(k, \epsilon)$-far from any k-junta and still be accepted with high probability, as long as it is ϵ-close to some k-junta.) The tester also returns a set of k blocks (disjoint subsets of indices of the n variables) such that there is a k-junta h that is $O(\epsilon)$-close to f and has all its relevant variables in one of the k blocks, with no block containing more than one relevant variable. Such an h must be itself $O(\epsilon)$ close to f^* as well. Using these properties, we then obtain a noisy sampler for the core of f^*, which on each execution makes one query to f and outputs a pair $(x, a) \in \{0,1\}^k \times \{0,1\}$ such that $\mathsf{core}_k(f^*) = a$ with high probability.

Intuitively, the idea is that such samples may be obtained by making queries to f on certain strings $y \in \{0,1\}^n$ that are constant inside each of the blocks, so that we know the values that y sets on the (unknown) relevant variables of h (which is sufficiently close to both f and f^*). While such y's are far from being uniformly distributed, the approach can be shown to work most of the time. These samples are then used to test isomorphism between $\mathsf{core}_k(f^*)$ and the functions in $\mathcal{C}_{[k]}$; in this final step we allow a small, possibly correlated, fraction of the samples to be incorrectly labelled.

4.2 Main lemmas and proof of Theorem 1

We start with the notion of a noisy sampler.

Definition 1. *Let* $g : \{0,1\}^k \to \{0,1\}$ *be a function, and let* $\eta, \mu \in [0,1)$. *An* (η, μ)-noisy sampler for g *is a probabilistic algorithm* $\widetilde{g}$ *that on each execution outputs* $(x, a) \in \{0,1\}^k \times \{0,1\}$ *such that*

– *for all* $\alpha \in \{0,1\}^k$, $\Pr[x = \alpha] = \frac{1}{2^k}(1 \pm \mu)$;

– $\Pr[a = g(x)] \geq 1 - \eta$;
– *the pairs output on each execution are mutually independent.*

An η-noisy sampler is an $(\eta, 0)$-noisy sampler, i.e. one that on each execution picks a uniformly random x. [7]

Now assume that f is very close to a k-junta $g : \{0,1\}^n \to \{0,1\}$, and we have been given an η-noisy sampler for $\mathsf{core}_k(g) : \{0,1\}^k \to \{0,1\}$. Then we can use a variant of Occam's razor to test (tolerantly) whether g is close to some function from a given class $\mathcal{S}$:

Lemma 1. *There is an algorithm that given $\epsilon \in \mathbb{R}^+$, $k \in \mathbb{N}$, a set $\mathcal{S}$ of Boolean functions on $\{0,1\}^k$, and an η-noisy sampler $\widetilde{g}$ for some $g : \{0,1\}^k \to \{0,1\}$, where $\eta \leq \epsilon/100$, satisfies the following:*

– *if* $\operatorname{dist}(g, \mathcal{S}) < \epsilon/10$, *it accepts with probability at least* $9/10$;
– *if* $\operatorname{dist}(g, \mathcal{S}) > 9\epsilon/10$, *it rejects with probability at least* $9/10$;
– *it draws* $O\left(\frac{1+\log|\mathcal{S}|}{\epsilon^2}\right)$ *samples from* $\widetilde{g}$.

The proof appears in Appendix A.

Now is the time to state the main technical lemma.

Lemma 2 (Construction of efficient noisy samplers).

There are algorithms A_P, A_S (resp. preprocessor and sampler), both of which having oracle access to a function $f : \{0,1\}^n \to \{0,1\}$, and satisfying the following properties:

The preprocessor A_P takes $\epsilon > 0, k \in \mathbb{N}$ as inputs, makes $O(k/\epsilon + k \log k)$ queries to f and can either reject or accept and return a state $\alpha \in \{0,1\}^{\mathrm{poly}(n)}$. Assuming A_P accepted, the sampler A_S can be called on demand, with state α as an argument; in each call, A_S makes only one query to f and outputs a pair $(x, a) \in \{0,1\}^k \times \{0,1\}$.

On termination of the preprocessing stage A_P, all the following conditions are fulfilled with probability at least $4/5$:

– *If f is $\theta_1(k, \epsilon)$-close to a k-junta, A_P has accepted f;*
– *If f is $\epsilon/2400$-far from a k-junta, A_P has rejected f;*
– *If A_P has accepted, state α is such that, for some permutation $\pi : [k] \to [k]$, $A_S(\alpha)$ is an $\epsilon/100$-noisy sampler for $\mathsf{core}_k(f^*)^\pi$.*

The statement is somewhat technical and calls for careful reading. It is crucial that the last condition be satisfied with high probability for *any* f. When $\theta_1(k, \epsilon) < \operatorname{dist}(f, \mathsf{Jun}_k) < \epsilon/2400$, it might be the case that A_P always accepts f, always rejects f, or anything in between, but with high probability either f has been rejected *or* an $\epsilon/100$-noisy sampler for (a permutation of) $\mathsf{core}_k(f^*)$ has been constructed.

Assuming Lemmas 2 and 1 we can prove our main theorem.

[7] The reader familiar with [CGM11] should beware that the usage of the parameter μ here is slightly different from that of the similar definition thereof.

Proof (of Theorem 1). Let $\tau \triangleq \theta_1(k,\epsilon)$. Suppose first that $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \tau$. Then Lemma 2 says that, with probability at least 4/5, we can construct an $\epsilon/100$-noisy sampler for $\mathsf{core}_k(f^*)$. Since $\mathrm{dist}(f, f^*) \leq \tau$ and $\mathrm{dist}(f, \mathcal{C}_{[k]}) \leq \tau$, we actually obtain an $\epsilon/100$-noisy sampler for a function that is $2\tau < \epsilon/10$-close to the core of some $g \in \mathcal{C}_{[k]}$. Using this noisy sampler we may apply the algorithm from Lemma 1 with $\mathcal{S} = \mathsf{core}_k(\mathcal{C}_{[k]})$, which in turn will accept with probability at least 9/10. The overall acceptance probability in this case is at least 7/10 by the union bound.

Now consider the case $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \geq \epsilon$. There are two possible sub cases:

$\mathrm{dist}(f, \mathsf{Jun}_k) > \epsilon/2400$: In this case f is rejected with probability at least $4/5 > 7/10$.

$\mathrm{dist}(f, \mathsf{Jun}_k) \leq \epsilon/2400$: In this case, with probability at least 4/5, either f is rejected (in which case we are done), or an $\epsilon/100$-noisy sampler has been constructed for $\mathsf{core}_k(f^*)$. Since f^* is $\epsilon/2400$-close to f, by triangle inequality we have $\mathrm{dist}(\mathsf{core}_k(f^*), \mathsf{core}_k(\mathcal{C}_{[k]})) \geq \mathrm{distiso}(f, \mathcal{C}_{[k]}) - \mathrm{dist}(f, f^*) > 9\epsilon/10$, and hence with probability at least 9/10 the algorithm from Lemma 1 rejects. Thus the overall rejection probability in this case is at least 7/10 too.

The assertion about the number of queries is easily seen to be correct, as it is the sum of the number of queries made in the preprocessing stage by A_P, and the number of executions of the sampler A_S.

The rest of this section is devoted to the proof of Lemma 2.

4.3 Additional definitions and lemmas

Our first observation is that, using rejection sampling, one can easily obtain an exactly uniform sampler (as required in Lemma 1) from a slightly non-uniform sampler at the cost of a small increase in the error probability:

Lemma 3. *Let $\widetilde{g}$ be an (η, μ)-noisy sampler for $g : \{0,1\}^k \to \{0,1\}$, that on each execution picks x according to some fixed distribution D. Then $\widetilde{g}$ can be turned into an $(\eta + \mu)$-noisy sampler $\widetilde{g}_{uniform}$ for g.*

Proof. Write U to denote the uniform distribution on $\{0,1\}^k$. The new sampler $\widetilde{g}_{uniform}$ acts as follows: it first obtains a sample (x, a) from $\widetilde{g}$, and

(acceptance) with probability $p_x \triangleq \frac{\Pr_{y \sim U}[y=x]}{(1+\mu)\Pr_{z \sim D}[z=x]}$ it outputs (x, a);

(rejection) with probability $1 - p_x$ it picks uniformly random $z \in \{0,1\}^k$ and outputs $(z, 0)$.

(Note that $p_x \leq 1$ by definition of an (n, μ)-noisy sampler).

Let (x', a') denote the pairs output by $\widetilde{g}_{uniform}$. It is easy to verify that the overall acceptance probability is $\mathbb{E}_{x \sim D}[p_x] = 1/(1+\mu)$ and thus, conditioned on acceptance, x' is uniformly distributed. In the case of rejection (which occurs with probability $\mu/(1+\mu)$) it is uniform by definition; hence the overall distribution of x' is uniform too. Recalling that $\Pr[a \neq g(x)] \leq \eta$, we conclude that $\Pr[a' \neq g(x')] \leq \eta + \mu/(1+\mu) \leq \eta + \mu$.

We remark that the conversion made in Lemma 3 is only possible when the distribution D is known. However, this will be the case for the sampler that we construct here.

Throughout the rest of this section, a random partition $\mathcal{I} = I_1, \ldots, I_\ell$ of $[n]$ into ℓ sets is constructed by starting with ℓ empty sets, and then putting each coordinate $i \in [n]$ into one of the ℓ sets picked uniformly at random. Unless explicitly mentioned otherwise, $\mathcal{I}$ will always denote a random partition $\mathcal{I} = I_1, \ldots, I_\ell$ of $[n]$ into ℓ subsets, where ℓ is even; and $\mathcal{J} = J_1, \ldots, J_k$ will denote an (ordered) k-subset of $\mathcal{I}$ (meaning that there are $a_1, \ldots, a_k$ such that $J_i = I_{a_i}$ for all $i \in [k]$).

Definition 2 (Operators replicate and extract). *We call $y \in \{0,1\}^n$ $\mathcal{I}$-*blockwise constant *if the restriction of y on every set of $\mathcal{I}$ is constant; that is, if for all $i \in [\ell]$ and $j, j' \in I_i$, $y_j = y_{j'}$.*

- *Given $z \in \{0,1\}^\ell$, define $\mathsf{replicate}_{\mathcal{I}}(z)$ to be the $\mathcal{I}$-blockwise constant string $y \in \{0,1\}^n$ obtained by setting $y_j \leftarrow z_i$ for all $i \in \ell$ and $j \in I_i$.*
- *Given an $\mathcal{I}$-blockwise constant $y \in \{0,1\}^n$ and an ordered subset $\mathcal{J} = (J_1, \ldots, J_k)$ of $\mathcal{I}$ define $\mathsf{extract}_{\mathcal{I},\mathcal{J}}(y)$ to be the string $x \in \{0,1\}^k$ where for every $i \in [k]$: $x_i = y_j$ if $j \in J_i$; and x_i is a uniformly random bit if $J_i = \emptyset$.*

Definition 3 (Distributions $\mathcal{D}_{\mathcal{I}}$ and $\mathcal{D}_{\mathcal{J}}$). *For any $\mathcal{I}$ and $\mathcal{J} \subseteq \mathcal{I}$ as above, we define a pair of distributions:*

- *The distribution $\mathcal{D}_{\mathcal{I}}$ on $\{0,1\}^n$: A random $y \sim \mathcal{D}_{\mathcal{I}}$ is obtained by*
  1. *picking $z \in \{0,1\}^\ell$ uniformly at random among all $\binom{\ell}{\ell/2}$ strings of weight $\ell/2$;*
  2. *setting $y \leftarrow \mathsf{replicate}_{\mathcal{I}}(z)$.*
- *The distribution $\mathcal{D}_{\mathcal{J}}$ on $\{0,1\}^{|\mathcal{J}|}$: A random $x \sim \mathcal{D}_{\mathcal{J}}$ is obtained by*
  1. *picking $y \in \{0,1\}^n$ at random, according to $\mathcal{D}_{\mathcal{I}}$;*
  2. *setting $x \leftarrow \mathsf{extract}_{\mathcal{I},\mathcal{J}}(y)$.*

Lemma 4 (Properties of $\mathcal{D}_{\mathcal{I}}$ and $\mathcal{D}_{\mathcal{J}}$).

1. *For all $\alpha \in \{0,1\}^n$, $\Pr_{\mathcal{I}, y \sim \mathcal{D}_{\mathcal{I}}}[y = \alpha] = 1/2^n$;*
2. *Assume $\ell > 4|\mathcal{J}|^2$. For every $\mathcal{I}$ and $\mathcal{J} \subseteq \mathcal{I}$, the total variation distance between $\mathcal{D}_{\mathcal{J}}$ and the uniform distribution on $\{0,1\}^{|\mathcal{J}|}$ is bounded by $2|\mathcal{J}|^2/\ell$. Moreover, the L_∞ distance between the two distributions is at most $4|\mathcal{J}|^2/(\ell 2^{|\mathcal{J}|})$.*

The proof appears in Appendix B.

Definition 4 (Algorithm $\mathsf{sampler}_{\mathcal{I},\mathcal{J}}(f)$). *Given $\mathcal{I}, \mathcal{J}$ as above and oracle access to $f : \{0,1\}^n \to \{0,1\}$, we define a probabilistic algorithm $\mathsf{sampler}_{\mathcal{I},\mathcal{J}}(f)$, that on each execution produces a pair $(x, a) \in \{0,1\}^{|\mathcal{J}|} \times \{0,1\}$ as follows: first it picks a random $y \sim \mathcal{D}_{\mathcal{I}}$, then it queries f on y, and outputs the pair $(\mathsf{extract}_{\mathcal{I},\mathcal{J}}(y), f(y))$.*

Jumping ahead, we remark that the pair $\mathcal{I}, \mathcal{J}$ (along with the values of k, ϵ) will be the information encoded in state α referred to in Lemma 2. In order to ensure that the last condition there is satisfied, we need to impose certain conditions on $\mathcal{I}$ and $\mathcal{J}$.

Definition 5. *Given $\delta > 0$, a function $f : \{0,1\}^n \to \{0,1\}$, a partition $\mathcal{I} = I_1, \ldots, I_\ell$ of $[n]$ and a k-subset $\mathcal{J}$ of $\mathcal{I}$, we call the pair $(\mathcal{I}, \mathcal{J})$ δ-*good *(with respect to f) if there exists a k-junta $h : \{0,1\}^n \to \{0,1\}$ such that the following conditions are satisfied:*

1. *Conditions on h:*
   (a) *Every relevant variable of h is also a relevant variable of f^* (recall that f^* denotes the k-junta closest to f);*
   (b) $\mathrm{dist}(f^*, h) < \delta$.
2. *Conditions on $\mathcal{I}$:*
   (a) *For all $j \in [\ell]$, I_j contains at most one variable of* $\mathsf{core}_k(f^*)$; [8]
   (b) $\Pr_{y \sim \mathcal{D}_\mathcal{I}}[f(y) \neq f^*(y)] \leq 10 \cdot \mathrm{dist}(f, f^*)$;
3. *Conditions on $\mathcal{J}$:*
   (a) *The set $\bigcup_{I_j \in \mathcal{J}} I_j$ contains* all *relevant variables of h;*

Lemma 5. *Let $\delta, f, \mathcal{I}, \mathcal{J}$ be as in the preceding definition. If the pair $(\mathcal{I}, \mathcal{J})$ is δ-good (with respect to f), then* $\mathsf{sampler}_{\mathcal{I},\mathcal{J}}(f)$ *is an (η, μ)-noisy sampler for some permutation of* $\mathsf{core}_k(f^*)$, *with $\eta \leq 2\delta + 4k^2/\ell + 10 \cdot \mathrm{dist}(f, f^*)$ and $\mu \leq 4k^2/\ell$.*

The proof appears in Appendix C.

As the lemma suggests, our next goal is to obtain a good pair $(\mathcal{I}, \mathcal{J})$. For this we need to prove that (a slight variation of) the junta tester from [Bla09] satisfies certain properties.

4.4 Junta testers, smoothness, and tolerance

Consider a property $\mathcal{P}$ of Boolean functions on $\{0,1\}^n$ and an ϵ-tester T for it that makes q queries and has success probability $1-\delta$. Let r denote a random seed (so that we can view the tester as a deterministic algorithm with an additional input r) and let $Q(f, r) \subseteq \{0,1\}^n$ be the set of queries it makes on input f and seed r. Define $Q(r) \triangleq \bigcup_f Q(f, r)$; this is the set of all possible queries T may make as f ranges over all possible functions, once r is fixed. We call $p \triangleq \max_r |Q(r)|$ the *non-adaptive complexity* of the tester. If $q = p$ then the tester is essentially non-adaptive; and clearly $p \leq 2^q$ holds for any tester of Boolean properties. We observe that for the junta tester of Blais [Bla09], p is in fact polynomially bounded in q. (Without loss of generality we assume that $Q(r)$ is never empty.)

[8] Note that this with 1a implies that every block I_j contains at most one relevant variable of h, since the variables of $\mathsf{core}_k(f^*)$ contain all relevant variables of f^*.

Definition 6. *A tester is* p-smooth *if its non-adaptive complexity is at most* p *and for all* $\alpha \in \{0,1\}^n$,

$$\Pr_{\substack{r \\ y \in Q(r)}}[y = \alpha] = \frac{1}{2^n}.$$

Notice that y is picked uniformly at random from the set $Q(r)$, regardless of the probability y would be queried by T on any particular f. In other words, we are picking one random query of the non-adaptive version of T that queries all of $Q(r)$ in bulk, and requiring that the resulting string be uniformly distributed.

Lemma 6. *Let* T *be a* p*-smooth tester for* $\mathcal{P}$ *that accepts every* $f \in \mathcal{P}$ *with probability at least* $1 - \delta$*. Then for every* $f : \{0,1\}^n \to \{0,1\}$, $\Pr[T$ *accepts* $f] \geq 1 - \delta - p \cdot \mathrm{dist}(f, \mathcal{P})$.

Proof. Choose any $f' \in \mathcal{P}$ and let $\Delta \triangleq \{y \in \{0,1\}^n : f(y) \neq f'(y)\}$. By the union bound, the probability (over r) that $Q(r)$ intersects Δ is at most $\mu \triangleq p \cdot \mathrm{dist}(f, f')$, and hence the probability is at least $1 - \mu$ that the tester reaches the same decision about f as it does about f'. But the probability that f' is rejected is at most δ, hence the claim follows.

Lemma 7. *The one-sided error junta tester* $\mathrm{T}_{[Bla09]}$ *from [Bla09] is* $p_7(k, 1/\epsilon)$*-smooth, where* $p_7(k, 1/\epsilon) \triangleq (10^{25} k^{10})/\epsilon^6$*. Thus, by Lemma 6, it accepts functions that are* $\theta_1(k, \epsilon)$*-close to* Jun_k *with probability at least* $9/10$ *(since* $10 \cdot \theta_1(k, \epsilon) \leq 1/p_7(k, 1/\epsilon)$*.) It also rejects functions that are* ϵ*-far from* Jun_k *with probability at least* $2/3$*, as proved in [Bla09].*

Before proving the lemma we need to briefly describe how $\mathrm{T}_{[\mathrm{Bla09}]}$ works. We refer to Algorithm 1 containing its pseudo-code. Given a random partition $\mathcal{I} = I_1, \ldots, I_\ell$ of $[n]$, it starts with an empty set $\mathcal{J}' = \emptyset$, and gradually adds to it the blocks I_i that have been found to contain a relevant variable as follows. For each of $O(k/\epsilon)$ rounds, it generates two random strings $x, y \in \{0,1\}^n$ and queries f on x and $x_{\bar{S}} y_S \triangleq x_{S \leftarrow y\restriction_S}$, where $S \triangleq [n] \setminus \bigcup_{I_i \in \mathcal{J}'} I_i$. (Picking x and y is the only place where randomness is used). If $f(x)$ turns out to be different from $f(x_{\bar{S}} y_S)$, we know that there is at least one relevant block in $\mathcal{I} \setminus \mathcal{J}'$ yet to be found. In this case, we can find a relevant block by performing a binary search on the $|\mathcal{I} \setminus \mathcal{J}'| + 1$ *hybrid* strings between x and $x_{\bar{S}} y_S$ obtained in the following way:

Let $I_{i_1}, \ldots, I_{i_m}$ be the blocks in $\mathcal{I} \setminus \mathcal{J}'$, where $1 \leq i_1 \leq \cdots \leq i_m \leq \ell$. The jth hybrid z_j has the same values as y on all indices in $I_{i_1} \cup \cdots \cup I_{i_j}$, and its values elsewhere are the same as those of x. In particular, $z_0 = x$ and $z_m = x_{\bar{S}} y_S$. If we know $a < b$ with $f(z_a) \neq f(z_b)$, then for any $a \leq m \leq b$ at least one of $f(z_a) \neq f(z_m)$ or $f(z_m) \neq f(z_b)$ must hold, so if $f(z_0) \neq f(z_m)$ we can use binary search to find a relevant block I_{i_j} after making at most $\log(m+1) \leq \log(\ell+1)$ queries to f on the hybrid strings.

If at some stage the tester discovers more than k relevant blocks then it rejects; otherwise it accepts and outputs a (possibly extended) set $\mathcal{J} \supseteq \mathcal{J}'$ of size k (see Algorithm 1).

Algorithm 1 ($\mathrm{T}_{[\mathrm{Bla09}]}(\mathrm{k}, \epsilon, \mathcal{I})$)

$\mathcal{J}' \leftarrow \emptyset$
for $i = 1$ **to** $\lceil 40(k+1)/\epsilon \rceil$ **do**
  $S \leftarrow [n] \setminus \bigcup_{I_i \in \mathcal{J}'} I_i$
  pick $x, y \in \{0,1\}^n$ uniformly at random
  if $f(x) \neq f(y_S x_{\bar{S}})$ **then**
    use binary search to find a block I_j containing a relevant variable
    $\mathcal{J}' \leftarrow \mathcal{J}' \cup \{I_j\}$
    if $|\mathcal{J}'| > k$, **reject** f
  end if
end for
extend (if needed) $\mathcal{J}'$ to a set $\mathcal{J}$ of size k, by adding to it $k - |\mathcal{J}'|$ arbitrary blocks from $\mathcal{I} \setminus \mathcal{J}'$
accept f and return $\mathcal{J}$

Remark 1. There are few minor differences between the original algorithm and the one presented here:

- The algorithm here has reduced probability of error; this can be easily achieved by increasing the partition size and number of iterations by a constant factor.
- The original algorithm constructs the random partition $\mathcal{I}$ by itself; here we treat $\mathcal{I}$ as an argument passed to the algorithm (for convenience).
- The original algorithm does not actually output the set $\mathcal{J}$, rather it identifies a set $\mathcal{J}'$ of at most k blocks containing relevant variables. Here $\mathrm{T}_{[\mathrm{Bla09}]}$ always returns a set $\mathcal{J}$ of size *exactly* k, by extending (if necessary) the set $\mathcal{J}'$ arbitrarily; as we show later, this extension will not affect the conditions claimed.

Proof (of Lemma 7). Note that once the randomness has been fixed, the number of possible queries that can be made in any given round is $|\mathcal{I} \setminus \mathcal{J}'| + 1 \leq \ell + 1$, so $|Q(r)| \leq 40\frac{k+1}{\epsilon}(\ell+1)$ (recall that ℓ is the number of blocks in partition $\mathcal{I}$). Also, any hybrid z_j of two uniformly random strings $x, y \in \{0,1\}^n$ is itself uniformly random. These two things together mean that the tester is $40\frac{k+1}{\epsilon}(\ell+1)$-smooth, and we can plug in the value $\ell = O(k^9/\epsilon^5)$ from [Bla09] (note that we modified slightly the constants therein in order to amplify the success probability).

4.5 Obtaining a good pair $(\mathcal{I}, \mathcal{J})$

We use the following lemma:

Lemma 8. *[FKR+02] For any $f : \{0,1\}^n \to \{0,1\}$ and $A \subseteq [n]$,* $\mathrm{dist}(f, \mathsf{Jun}_A) \leq Inf_f([n] \setminus A) \leq 2 \cdot \mathrm{dist}(f, \mathsf{Jun}_A)$.

We also use the fact (see [FKR+02],[Bla09] for a proof) that influence is monotone and subadditive; namely, for all $f : \{0,1\}^n \to \{0,1\}$ and $A, B \subseteq [n]$, $Inf_f(A) \leq Inf_f(A \cup B) \leq Inf_f(A) + Inf_f(B)$.

In the following proposition we claim that the tester $\mathrm{T}_{[\mathrm{Bla09}]}$ satisfies several conditions that we need for obtaining the aforementioned sampler.

Proposition 1. *There is a tester* $\mathrm{T}_{[Bla09]}$ *for* Jun_k *with query complexity* $O(k \log k + k/\epsilon)$*, that takes a (random) partition* $\mathcal{I} = I_1, \ldots, I_\ell$ *of* $[n]$ *as input, where* $\ell = \Theta(k^9/\epsilon^5)$ *is even, and outputs (in case of acceptance) a* k*-subset* $\mathcal{J}$ *of* $\mathcal{I}$ *such that for any* f *the following conditions hold (the probabilities below are taken over the randomness of the tester and the construction of* $\mathcal{I}$*):*

- *if* f *is* $\theta_1(k, \epsilon)$ *close to* Jun_k*,* $\mathrm{T}_{[Bla09]}$ *accepts with probability at least* $9/10$*;*
- *if* f *is* $\epsilon/2400$*-far from* Jun_k*,* $\mathrm{T}_{[Bla09]}$ *rejects with probability at least* $9/10$*;*
- *for* any f*, with probability at least* $4/5$ *either* $\mathrm{T}_{[Bla09]}$ *rejects, or it outputs* $\mathcal{J}$ *such that the pair* $(\mathcal{I}, \mathcal{J})$ *is* $\epsilon/600$*-good (as per Definition 5).*

In particular, if $\mathrm{dist}(f, \mathsf{Jun}_k) \leq \theta_1(k, \epsilon)$*, then with probability at least* $4/5$ $\mathrm{T}_{[Bla09]}$ *outputs a set* $\mathcal{J}$ *such that* $(\mathcal{I}, \mathcal{J})$ *is* $\epsilon/600$*-good.*

Proof. By Lemma 7, the first two conditions are satisfied by the junta tester, called with a value of $\epsilon' = \epsilon/2400$; note that $10 \cdot \theta_1(k, \epsilon) = 1/p_7(k, \epsilon')$.

Let $\mathcal{J}' = (I_{s_1}, \ldots, I_{s_{|\mathcal{J}'|}})$ be the set output by the original algorithm $\mathrm{T}_{[\mathrm{Bla09}]}$ and let $S = \{s_1, \ldots, s_{|\mathcal{J}'|}\}$. Closer inspection of Algorithm 1 shows that, with probability at least 19/20,

(∗) either f is rejected or the set S satisfies $Inf_f\Big([n] \setminus (\bigcup_{j \in S} I_j)\Big) \leq \epsilon/4800.$

This is because the main loop of algorithm runs for $40(k+1)/\epsilon'$ rounds. Suppose that at any of these, the influence of the remaining blocks is always $\geq \epsilon'/2$. Since the expected number of rounds to find $k+1$ relevant blocks is at most $2(k+1)/\epsilon'$ in this case, it follows that with probability 19/20, a $(k+1)$-th relevant block is found and f is rejected.

Recall that when $|S| < k$ the set $\mathcal{J}'$ is extended by putting in it $k - |S|$ additional "dummy" blocks from $\mathcal{I} \backslash \mathcal{J}'$ (some of them possibly empty), obtaining a set $\mathcal{J}$ of size exactly k.

Now we go back to proving the third item. Let $R \in \binom{[n]}{\leq k}$ denote the set of the relevant variables of f^* (the closest k-junta to f), and let $V \in \binom{[n]}{k}$, $V \supseteq R$, denote the set of the variables of $\mathsf{core}_k(f^*)$. Assume that $\mathrm{dist}(f, \mathsf{Jun}_k) \leq \epsilon/2400$, [9] and $\mathrm{T}_{[\mathrm{Bla09}]}$ did not reject. In this case,

- by (∗), with probability at least 19/20 the set $\mathcal{J}$ satisfies

$$Inf_f\Big([n] \setminus (\bigcup_{I_j \in \mathcal{J}} I_j)\Big) \leq Inf_f\Big([n] \setminus (\bigcup_{j \in S} I_j)\Big) \leq \epsilon/4800;$$

- since $\ell \gg k^2$, with probability larger than 19/20 all elements of V fall into different blocks of the partition $\mathcal{I}$;

[9] For other f's the third item follows from the second item.

– by Lemma 4, $\Pr_{\mathcal{I}, y\sim\mathcal{D}_\mathcal{I}}\Big[f(y) = f^*(y)\Big] = \mathrm{dist}(f, f^*)$; hence by Markov's inequality, with probability at least $9/10$ the partition $\mathcal{I}$ satisfies $\Pr_{y\sim\mathcal{D}_\mathcal{I}}[f(y) \neq f^*(y)] \leq 10 \cdot \mathrm{dist}(f, f^*)$.

So with probability at least $4/5$, all three of these events occur. Now we show that conditioned on them, the pair $(\mathcal{I}, \mathcal{J})$ is $\epsilon/600$-good.

Let $U = R \cap (\bigcup_{I_j\in\mathcal{J}} I_j)$. Informally, U is the subset of the relevant variables of f^* that were successfully "discovered" by $\mathrm{T}_{[\mathrm{Bla09}]}$. Since $\mathrm{dist}(f, f^*) \leq \epsilon/2400$, we have $Inf_f([n] \setminus V) \leq \epsilon/1200$ (by Lemma 8). By the subadditivity and monotonicity of influence we get

$$Inf_f([n]\setminus U) \leq Inf_f([n]\setminus V) + Inf_f(V\setminus U) \leq Inf_f([n]\setminus V) + Inf_f\Big([n]\setminus(\bigcup_{I_j\in\mathcal{J}} I_j)\Big) \leq \epsilon/960,$$

where the second inequality follows from $V \setminus U \subseteq [n] \setminus (\bigcup_{I_j\in\mathcal{J}} I_j)$. This means, by Lemma 8, that there is a k-junta h in Jun_U satisfying $\mathrm{dist}(f, h) \leq \epsilon/960$, and by triangle inequality, $\mathrm{dist}(f^*, h) \leq \epsilon/2400 + \epsilon/960 < \epsilon/600$. Based on this h, we can verify that the pair $(\mathcal{I}, \mathcal{J})$ is $\epsilon/600$-good by going over the conditions in Definition 5.

We are finally ready to complete the proof of Lemma 2.

4.6 Proof of Lemma 2

We start by describing how A_P and A_S operate: The preprocessor A_P starts by constructing a random partition $\mathcal{I}$ and calling the junta tester $\mathrm{T}_{[\mathrm{Bla09}]}$. Then, in case $\mathrm{T}_{[\mathrm{Bla09}]}$ accepted, A_P encodes in the state α the partition $\mathcal{I}$ and the subset $\mathcal{J} \subseteq \mathcal{I}$ output by $\mathrm{T}_{[\mathrm{Bla09}]}$ (see Proposition 1), along with the values of k and ϵ. The sampler A_S, given α, obtains a pair $(x, a) \in \{0,1\}^k \times \{0,1\}$ by executing $\mathsf{sampler}_{\mathcal{I},\mathcal{J}}(f)$ (once).

Now we show how Lemma 2 follows from Proposition 1. The first two items are immediate. As for the third item, notice that we only have to analyze the case where $\mathrm{dist}(f, f^*) \leq \epsilon/2400$ and $\mathrm{T}_{[\mathrm{Bla09}]}$ accepted; all other cases are taken care of by the first two items. By the third item in Proposition 1, with probability at least $4/5$ the pair $(\mathcal{I}, \mathcal{J})$ is $\epsilon/600$-good. If so, by Lemma 5 $\mathsf{sampler}_{\mathcal{I},\mathcal{J}}(f)$ is an (η, μ)-noisy sampler for some permutation of $\mathsf{core}_k(f^*)$, with $\eta \leq \epsilon/300 + 4k^2/\ell + 10 \cdot \mathrm{dist}(f, f^*) \leq \epsilon/120 + 4k^2/\ell$ and $\mu \leq 4k^2/\ell$. The final step we apply is the conversion from Lemma 3, with which we obtain a $(\epsilon/120 + 4k^2/\ell + 4k^2/\ell) \leq (\epsilon/100)$-noisy sampler for some permutation of $\mathsf{core}_k(f^*)$. □

5 Lower bounds

In order to analyze how close to optimal our algorithms are, we show in this section lower bounds concerning most of the problems studied here. Some of the bounds were known from prior work; our exposition unifies most of them by using the notion of k-wise independent generators.

Definition 7. *Let $\mathcal{C}$ denote a class of functions $f : \mathbb{F} \to S$, where $\mathbb{F}$ is a field and S a finite set. We say that $\mathcal{C}$ can generate* k-wise independence *if there is a distribution $\mathcal{D}$ supported on elements of $\mathcal{C}$ such that the random variables $\{f(x)\}_{x\in\mathbb{F}}$ are k-wise independent and each of them is uniformly distributed on S, i.e.*

$$\Pr_{f\in\mathcal{D}}[f(x_1) = \alpha_1 \wedge f(x_2) = \alpha_2 \wedge \ldots \wedge f(x_k) = \alpha_k] = |S|^{-k}$$

for any k distinct $x_1, \ldots, x_k \in \mathbb{F}$ and any $\alpha_1, \ldots, \alpha_k \in S$.

We identify the set $\{0,1\}^n$ with the field with 2^n elements. Clearly the class of all boolean functions $f : \{0,1\}^n \to \{0,1\}$ can generate n-wise independence (here $\mathbb{F} = GF(2^n)$ and $S = \{0,1\}$).

For a less trivial example, take for $\mathcal{C}$ the class of all polynomials of degree $\leq k-1$ over $\mathbb{F}$. This class can generate k-wise independence, because any degree $\leq k-1$ univariate polynomial over a field can be interpolated from its values on any set of k distinct points, and the solution is unique. From this we can obtain a family of *boolean* functions on $\{0,1\}^n$ that generates k-wise independence in the following way: associate with each polynomial $p : GF(2^n) \to GF(2^n)$ of degree $k-1$ the function that, on input $x \in \{0,1\}^n$, returns the last bit of $p(x)$. (A different, slightly more efficient way, would be to work on a field of size roughly $2^n/n$). Clearly the resulting family can generate k-wise independence.

The following observation is just a restatement of Definition 7, coupled with Yao's principle:

Observation 2 *If $\mathcal{C}$ can generate k-wise independence under distribution $\mathcal{D}$, then at least $k+1$ adaptive queries are needed to distinguish, with probability $> 1/2$, between a function f drawn from $\mathcal{D}$ and a uniformly random $f : \mathbb{F} \to S$.*

We say class $\mathcal{C}$ is *far from uniform* if a uniformly random function $f : \mathbb{F} \to S$ is (say) 1/10-far from every element of $\mathcal{C}$ with probability larger than 2/3. It follows that if $\mathcal{C}$ is both far from uniform and capable of generating k-wise independence, then more than k queries are necessary for testing membership in it (with distance parameter 1/10). From our second example we see that the latter condition holds for any class $\mathcal{C}$ that can evaluate any polynomial of degree $< k$ over $\mathbb{F}_{2^n}$, therefore we obtain:

Observation 3 *If $\mathcal{C}$ is far from uniform and contains all functions computed by polynomials of degree $< k$ over $GF(2^n)$, then testing membership in $\mathcal{C}$ requires $> k$ queries.*

By a result of Healy and Viola [HV06], field arithmetic over $GF(2^n)$ is in TC^0. In particular it is possible to evaluate any t-term polynomial $p \in GF(2^n)[x]$ with a circuit built up from threshold gates having constant depth and size $\text{poly}(n,t)$, which is $\text{poly}(n)$ if $t = \text{poly}(n)$. It is known that TC^0 is contained in NC^1. It is also known that NC^1 corresponds precisely to the set of Boolean formulas of polinomial size, and also to the set of functions computed by width-5 branching programs of polynomial size [Bar89]. Summarizing, the last bit of polynomial functions over $GF(2^n)$ of degree n (and therefore $n+1$ terms) can be computed by boolean formulae of size n^c and branching programs of size n^c for some c.

Proposition 2. *The following lower bounds hold for the respective testing problems (for size parameter up to some polynomial of* n*):*

1. *size-*s *boolean formulae, branching programs and boolean circuits:* $\mathrm{poly}(s)$*. (The bound for circuits appeared in [CGM11]).*
2. *functions with Fourier degree* d*:* $\Omega(d)$*.*
3. s*-sparse polynomials over* $GF(2)$*:* $\Omega(\sqrt{s})$*.*
4. s*-term DNFs, size-*s *decision trees:* $\Omega(\log s)$*.*

Proof. (sketch). For items 1 and 3 we follow the approach outlined above. The fact that each of the respective classes is far from uniform is easy to verify, so we skip the proof of that part.

1. Let c be as before and $m = 2s^{1/c} \leq n$. From the above, we can find a distribution over boolean functions $f : \{0,1\}^m \to \{0,1\}$ computed by boolean formulae or branching programs of size s, which is both far from uniform and can generate $m/2 = s^{1/c}$ independence. (One can also extend these functions to $f' : \{0,1\}^n \to \{0,1\}$ by padding without an increase in the distance from uniform, size parameter, or hardness of testing). Boolean formulae are a special case of Boolean circuits, hence a lower bound of $s^{1/c}$ follows for testing any of these clases.
2. See [CGM11].
3. This follows from the recent work of Goldreich [Gol10] showing an $\Omega(\sqrt{s})$ lower bound for distinguishing s-parities from random parities, as any s-parity is in particular an s-sparse polynomial. (It appears that his techniques can be improved further to yield an $\Omega(s/(\log s \log\log s))$ lower bound).
4. Any function can be computed by DNFs and decision trees of size 2^n. It is known (see [CGM11] for a precise statement and its proof) that there are k-juntas, and hence DNFs of size $s = 2^k$, whose random permutations look uniformy random to any tester making fewer than $\delta k = \delta \log s$ queries for some constant $\delta > 0$. Provided $s \leq 2^{n/2}$, the class of random permutations of any such junta is far from uniform, hence the result.

Remark 2. The lower bound for circuits appeared in [CGM11], but the argument given here is more direct. From independent work of Blais, Brody and Matulef [BBM11] follows a stronger lower bound of $\Omega(s)$ queries for s-sparse polynomials. They also obtain the $\Omega(\log s)$ lower bounds for DNFS and decision trees.

Acknowledgement

We thank Noga Alon, Eric Blais and Eldar Fischer for very useful discussions.

References

[Bar89] David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *J. Comput. Syst. Sci.*, 38:150–164, February 1989.

[BBM11] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Personal communication*, 2011.

[Bla09] Eric Blais. Testing juntas nearly optimally. In *Proc. ACM symposium on the Theory of computing*, pages 151–158, New York, NY, USA, 2009. ACM.

[CGM11] Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Nearly tight bounds for testing function isomorphism. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

[DLM+07] Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A. Servedio, and Andrew Wan. Testing for concise representations. *Proc. IEEE Symposium on Foundations of Computer Science*, 0:549–558, 2007.

[DLM+08] Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Rocco A. Servedio, and Andrew Wan. Efficiently testing sparse GF(2) polynomials. In *ICALP '08: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I*, pages 502–514, Berlin, Heidelberg, 2008. Springer-Verlag.

[FKR+02] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. In *FOCS*, pages 103–112, 2002.

[Gol10] Oded Goldreich. On testing computability by small width obdds. In *APPROX-RANDOM*, pages 574–587, 2010.

[GOS+09] Parikshit Gopalan, Ryan O'Donnell, Rocco A. Servedio, Amir Shpilka, and Karl Wimmer. Testing fourier dimensionality and sparsity. In *ICALP*, pages 500–512, 2009.

[HV06] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In Bruno Durand and Wolfgang Thomas, editors, *STACS 2006*, volume 3884 of *Lecture Notes in Computer Science*, pages 672–683. Springer Berlin / Heidelberg, 2006.

[PRS02] Michal Parnas, Dana Ron, and Alex Samorodnitsky. Testing basic boolean formulae. *SIAM J. Discrete Math.*, 16(1):20–46, 2002.

[Ser10] Rocco A. Servedio. Testing by implicit learning: a brief survey. 2010.

Appendix

A Proof of Lemma 1

Consider the tester described in Algorithm 2. It is clear that it uses $O\left(\frac{1+\log|\mathcal{S}|}{\epsilon^2}\right)$

Algorithm 2 (tests if $g \in \mathcal{S}$)

---

let $q \leftarrow \frac{c+\log|\mathcal{S}|}{\epsilon^2}$, where c is a constant to be specified later
obtain q independent samples $(x^1, a^1), \ldots, (x^q, a^q)$ from $\widetilde{g}$
accept if and only if there exists a function $f \in \mathcal{S}$ such that $\left|\left\{i \in [q] : f(x^i) \neq a^i\right\}\right| < \epsilon q/2$.

---

samples.

Let $f \in \mathcal{S}$, $\delta_f = \mathrm{dist}(f, g)$ and let $\Delta_f \subseteq \{0,1\}^k$, $|\Delta_f| = \delta_f 2^k$, be the set of inputs on which f and g disagree. Since the x's are independent and uniformly distributed random variables, we have that $\Pr_x[x \in \Delta_f] = \delta_f$. Using Chernoff bounds (additive form) we obtain

$$\Pr\left[\left|\left|\{i \in [q] : f(x^i) \neq g(x^i)\}\right| - \delta_f q\right| > \epsilon q/10\right] = 2^{-\Omega(\epsilon^2 q)},$$

which is less than $\frac{1}{20|\mathcal{S}|}$ for a sufficiently large constant c. Therefore, with probability at least $19/20$,

$$|\{i \in [q] : f(x^i) \neq g(x^i)\}| = \delta_f q \pm \frac{\epsilon q}{10}$$

holds for all $f \in \mathcal{S}$. To relate this to the fraction of samples (x, a) for which $f(x) \neq a$, we use Markov's inequality:

$$\Pr\left[|\{i \in [q] : a^i \neq g(x^i)\}| \geq \epsilon q/5\right] \leq \Pr\left[|\{i \in [q] : a^i \neq g(x^i)\}| \geq 20\eta q\right] \leq 1/20.$$

Therefore, with probability at least $9/10$,

$$|\{i \in [q] : f(x^i) \neq a^i\}| = \delta_f q \pm 3\epsilon q/10$$

for all $f \in \mathcal{S}$.

The result follows, since if $\mathrm{dist}(g, \mathcal{S}) < \epsilon/10$ then there is $f \in \mathcal{S}$ such that $\delta_f q + 3\epsilon q/10 < \epsilon q/2$; and if $\mathrm{dist}(g, \mathcal{S}) > 9\epsilon/10$ then for all $f \in \mathcal{S}$, $\delta_f q - 3\epsilon q/10 > \epsilon q/2$. □

B Proof of Lemma 4

Item 1 Each choice of $z \in \{0,1\}^\ell$, $|z| = \ell/2$, in Definition 3 splits $\mathcal{I}$ into two equally-sized sets: $\mathcal{I}^0$ and $\mathcal{I}^1$; and the bits corresponding to indices in $\mathcal{I}^b$ (where $b \in \{0,1\}$) are set to b in the construction of y. For each index $i \in [n]$, the block it is assigned to is chosen independently at random from $\mathcal{I}$, and therefore falls within $\mathcal{I}^0$ (or $\mathcal{I}^1$) with probability $1/2$, independently of other $j \in [n]$. (This actually shows that the first item of the lemma still holds if z is an arbitrarily fixed string of weight $\ell/2$, rather than a randomly chosen one).

Item 2 Let $k = |\mathcal{J}|$. Let us prove the claim on the L_∞ distance, which implies the other one. We may assume that all sets J_i in $\mathcal{J}$ are non-empty; having empty sets can only decrease the distance to uniform. Let $w \in \{0,1\}^k$. The choice of $y \sim \mathcal{D}_\mathcal{I}$, in the process of obtaining $x \sim \mathcal{D}_\mathcal{J}$, is independent of $\mathcal{J}$; thus, for every $i \in [k]$ we have

$$\Pr_{x\sim\mathcal{D}_\mathcal{J}}[x_i = w_i \mid x_j = w_j \ \forall j < i] \le \frac{\ell/2}{\ell - k} < \frac{1}{2} + \frac{k}{\ell},$$

and

$$\Pr_{x\sim\mathcal{D}_\mathcal{J}}[x_i = w_i \mid x_j = w_j \ \forall j < i] \ge \frac{\ell/2 - k}{\ell - k} > \frac{1}{2} - \frac{k}{\ell}.$$

Using the inequalities $1 - my \le (1-y)^m$ for all $y < 1, m \in \mathbb{N}$ and $(1+y)^m \le e^{my} \le 1 + 2my$ for all $m \ge 0, 0 \le my \le 1/2$, we conclude

$$\Pr_{x\sim\mathcal{D}_\mathcal{J}}[x = w] = \left(\frac{1}{2} \pm \frac{k}{\ell}\right)^k = \frac{1}{2^k}\left(1 \pm \frac{4k^2}{\ell}\right).$$

whereas a truly uniform distribution U should satisfy $\Pr_{x\sim U}[x = w] = 1/2^k$. $\square$

C Proof of Lemma 5

By item 2b in Definition 5, it suffices to prove that

$$\Pr_{y\sim\mathcal{D}_\mathcal{I}}[f^*(y) \ne \mathsf{core}_k(f^*)^\pi(\mathsf{extract}_{\mathcal{I},\mathcal{J}}(y))] < 2\delta + 4k^2/\ell$$

for some π.

Let h be the k-junta witnessing the fact that the pair $(\mathcal{I},\mathcal{J})$ is δ-good. Let $V \subseteq [n]$ be the set of k variables of $\mathsf{core}_k(f^*)$. (Recall that V may actually be a superset of the relevant variables of f^*.) Let $\mathcal{J}' \triangleq \{I_j \in \mathcal{I} : I_j \cap V \ne \emptyset\}$ be an ordered subset respecting the order of $\mathcal{J}$, and let π be the permutation whose inverse maps the i-th relevant variable of f^* (in the standard order) to the index of the element of $\mathcal{J}'$ in which it is contained. We assume without loss of generality that π is the identity map.

It follows from Definition 5 that $|\mathcal{J}'| = |V| = k$, since each block in $\mathcal{I}$ contains at most one variable of $\mathsf{core}_k(f^*)$. For any $\mathcal{I}$-uniform $y \in \{0,1\}^n$, let

$x \triangleq \mathsf{extract}_{I,\mathcal{J}}(y)$ and $x' \triangleq \mathsf{extract}_{I,\mathcal{J}'}(y)$ denote the k-bit strings corresponding to $\mathcal{J}$ and $\mathcal{J}'$. By definitions, we have the equalities

(1) $f^*(y) = \mathsf{core}_k(f^*)(x')$,

(2) $\mathsf{core}_k(h)(x) = \mathsf{core}_k(h)(x')$.

The first equality is by Definition 2, and the second one follows from items 1a and 3a in Definition 5. From item 1b we also have

(3) $\Pr_{r\in\{0,1\}^k}[\mathsf{core}_k(f^*)(r) \neq \mathsf{core}_k(h)(r)] < \delta$,

where r is picked uniformly at random. However, by the second item of Lemma 4, the distribution $\mathcal{D}_{\mathcal{J}}$ is $2k^2/\ell$ close to uniform; combining this with (3) we also get

(4) $\Pr_{y\sim\mathcal{D}_{\mathcal{I}}}[\mathsf{core}_k(f^*)(x) \neq \mathsf{core}_k(h)(x)] < \delta + 2k^2/\ell$.

Likewise, we have

(5) $\Pr_{y\sim\mathcal{D}_{\mathcal{I}}}[\mathsf{core}_k(f^*)(x') \neq \mathsf{core}_k(h)(x')] < \delta + 2k^2/\ell$,

thus, using $(2, 4, 5)$ and the union bound we get

(6) $\Pr_{y\sim\mathcal{D}_{\mathcal{I}}}[\mathsf{core}_k(f^*)(x') \neq \mathsf{core}_k(f^*)(x)] < 2\delta + 4k^2/\ell$.

Combining (1) and (6) we conclude that

$$\Pr_{y\sim\mathcal{D}_{\mathcal{I}}}[f^*(y) \neq \mathsf{core}_k(f^*)(x)] < 2\delta + 4k^2/\ell,$$

and the claim follows. □