Electronic Colloquium on Computational Complexity, Report No. 105 (2016)

# Tolerant Junta Testing and the Connection to Submodular Optimization and Function Isomorphism

Eric Blais* Clément L. Canonne† Talya Eden‡ Amit Levi§ Dana Ron¶

July 13, 2016

**Abstract**

The function $f: \{-1,1\}^n \to \{-1,1\}$ is a $k$-junta if it depends on at most $k$ of its variables. We consider the problem of *tolerant* testing of $k$-juntas, where the testing algorithm must accept any function that is $\epsilon$-*close* to some $k$-junta and reject any function that is $\epsilon'$-far from every $k'$-junta for some $\epsilon' = O(\epsilon)$ and $k' = O(k)$.

Our first result is an algorithm that solves this problem with query complexity polynomial in $k$ and $1/\epsilon$. This result is obtained via a new polynomial-time approximation algorithm for *submodular function minimization* (SFM) under large cardinality constraints, which holds even when only given an approximate oracle access to the function.

Our second result considers the case where $k' = k$. We show how to obtain a smooth tradeoff between the amount of tolerance and the query complexity in this setting. Specifically, we design an algorithm that given $\rho \in (0, 1/2)$ accepts any function that is $\frac{\epsilon\rho}{8}$-close to some $k$-junta and rejects any function that is $\epsilon$-far from every $k$-junta. The query complexity of the algorithm is $O\big(\frac{k \log k}{\epsilon\rho(1-\rho)^k}\big)$.

Finally, we show how to apply the second result to the problem of tolerant isomorphism testing between two unknown Boolean functions $f$ and $g$. We give an algorithm for this problem whose query complexity only depends on the (unknown) smallest $k$ such that either $f$ or $g$ is *close* to being a $k$-junta.

*University of Waterloo. Email: eric.blais@uwaterloo.ca.

†Columbia University. Email: ccanonne@cs.columbia.edu. Research supported by NSF CCF-1115703 and NSF CCF-1319788.

‡School of EE, Tel Aviv University. Email: talyaa01@gmail.com. This research was partially supported by the Israel Science Foundation grant No. 671/13 and by a grant from the Blavatnik fund.

§University of Waterloo. Email: amit.levi@uwaterloo.ca.

¶School of EE, Tel Aviv University. Email: danaron@post.tau.ac.il. This research was partially supported by the Israel Science Foundation grant No. 671/13 and by a grant from the Blavatnik fund.



ISSN 1433-8092

# 1 Introduction

A function $f\colon \{-1,1\}^n \to \{-1,1\}$ is a *$k$-junta* if it depends on at most $k$ of its variables. Juntas are a central object of study in the analysis of Boolean functions, in particular since they are good approximators for many classes of (more complex) Boolean functions. In the context of learning, the study of juntas was introduced by Blum et al. [Blu94, BL97] to model the problem of learning in the presence of irrelevant attributes. Since then, juntas have been extensively studied both in computational learning theory (e.g., [MOS03, Val15]) and in applied machine learning (e.g., [JL10]).

Juntas have also been studied within the framework of property testing. Here the task is to design a randomized algorithm that, given query access to a function $f$, accepts $f$ if it is a $k$-junta and rejects $f$ if it is $\epsilon$-far from every $k$-junta (where by $\epsilon$-far we mean that $f$ must be modified in at least an $\epsilon$-fraction of its values in order to be made a $k$-junta). The algorithm should succeed with high constant probability, and should perform as few queries as possible. The problem of testing $k$-juntas was first addressed by Fischer et. [FKR$^+$04]. They designed an algorithm that queries the function on a number of inputs polynomial in $k$, and *independent of $n$*. A series of subsequent works essentially settled the optimal query complexity for this problem, establishing that $\tilde{\Theta}(k/\epsilon)$ queries are both necessary and sufficient for this problem [Bla08, Bla09, CG04, STW15].

The standard setting of property testing, however, is somewhat brittle, in that a testing algorithm is only guaranteed to accept functions that *exactly* satisfy the property. But what if one wishes to accept functions that are *close* to the desired property? To address this question, Parnas, Ron, and Rubinfeld introduced in [PRR06] a natural generalization of property testing, where the algorithm is required to be *tolerant*. Namely, a *tolerant property testing algorithm* is required to accept any function that is *close* to the property, and, as in the standard model, to reject any function that is far from the property.[1]

As observed in [PRR06], any standard testing algorithm whose queries are uniformly (but not necessarily independently) distributed, is inherently tolerant to some extent. However, for many problems, strengthening the tolerance requires applying different methods and devising new algorithms (see e.g., [GR05, PRR06, FN07, ACCL07, KS09, MR09, FR10, CGR13, BMR16]). Furthermore, there are some properties that have standard testers with sublinear query complexity, but for which any tolerant tester must perform a linear number of queries [FF06, Tel16].

The problem of tolerant testing of juntas was previously considered by Diakonikolas et al. [DLM$^+$07]. They applied the aforementioned observation from [PRR06] and showed that one of the junta testers from [FKR$^+$04] actually accepts functions that are $\mathrm{poly}(k, 1/\epsilon)$-close to $k$-juntas. Chakraborty et al. [CFGM12] observed that the analysis of the (standard) junta tester of Blais [Bla09] implicitly implies an $\exp(k/\epsilon)$ query complexity tolerant tester which accepts functions that are $\epsilon/C$-close to some $k$-junta (for some constant $C > 1$) and rejects functions that are $\epsilon$-far from every $k$-junta.

## 1.1 Our results

In this work, we study the question of tolerant testing of juntas from two different angles, and obtain

[1] Ideally, a tolerant testing algorithm should work for any given tolerance parameter $\epsilon' < \epsilon$ (that is, accept functions that are $\epsilon'$-close to having the property), and have complexity that depends on $\epsilon - \epsilon'$. However, in some cases the relation between $\epsilon'$ and $\epsilon$ may be more restricted (e.g., $\epsilon' = \epsilon/c$ for a constant $c$). A closely related notion considered in [PRR06] is that of *distance approximation* where the goal is to obtain an estimate of the distance that the object has to a property.

two algorithms with different (and incomparable) guarantees, Further, we show how to leverage one of these algorithms to get a tester for isomorphism between Boolean functions with "instance-by-instance" (defined below) query complexity. The first of our results is a $\mathrm{poly}(k, 1/\epsilon)$-query algorithm that accepts functions which are close to $k$-juntas and rejects functions which are far from every $4k$-junta.

**Theorem 1.1.** *There exists an algorithm that, given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$ and parameters $k \geq 1$ and $\epsilon \in (0,1)$, satisfies the following.*

- *If $f$ is $\epsilon/16$-close to some $k$-junta, then the algorithm accepts with high constant probability.*
- *If $f$ is $\epsilon$-far from every $4k$-junta, then the algorithm rejects with high constant probability.*

*The query complexity of the algorithm is* $\mathrm{poly}(k, \frac{1}{\epsilon})$ *.*

The algorithm referred to in the theorem can be seen as a relaxed version of a tolerant testing algorithm. Namely, the algorithm rejects functions that are $\epsilon$-far from every $4k$-junta rather than $\epsilon$-far from every $k$-junta. Similar relaxations have been considered both in the standard testing model (e.g., [PR02, KR98, KNOW14]) and in the tolerant testing model [PR02].

We next study the question of tolerant testing without the above relaxation. That is, when the tester is required to reject functions that are $\epsilon$-far from being a $k$-junta. We obtain a *smooth tradeoff* between the amount of tolerance and the query complexity. In particular, this tradeoff allows one to recover, as special cases, both the results of Fischer et al. [FKR+04] and (an improvement of) Chakraborty et al. [CFGM12].

**Theorem 1.2.** *There exists an algorithm that, given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$ and parameters $k \geq 1$, $\epsilon \in (0,1)$ and $\rho \in (0,1/2)$, satisfies the following.*

- *If $f$ is $\rho\epsilon/8$-close to some $k$-junta, then the algorithm accepts with high constant probability.*
- *If $f$ is $\epsilon$-far from every $k$-junta, then the algorithm rejects with high constant probability.*

*The query complexity of the algorithm is* $O\Big(\frac{k\log k}{\epsilon\rho(1-\rho)^k}\Big)$.

Finally, we show how the above results can be applied to the problem of *isomorphism testing*, which we recall next. Given query access to two unknown Boolean functions $f, g\colon \{-1,1\}^n \to \{-1,1\}$ and a parameter $\epsilon \in (0,1]$, one has to distinguish between (i) $f$ is equal to $g$ up to some relabeling of the input variables; and (ii) $\mathrm{dist}(f, g\circ\pi) > \epsilon$ for every such relabeling $\pi$. The worst-case complexity of this task is known, with $\tilde{\Theta}(2^{\frac{n}{2}})$ queries being necessary and sufficient [AB10, ABC+13].

However, is the exponential dependence on $n$ always necessary, or can we obtain better results for "simple" functions? Ideally we would like our testers to improve on this worst-case behavior, and instead have an *instance-specific* query complexity, depending only on some intrinsic parameter of the functions $f, g$ to be tested. This is the direction we pursue here. Let $k^* = k^*(f,g,\gamma)$ be the smallest $k$ such that either $f$ or $g$ is $\gamma$-close to being a $k$-junta. We show that it is possible to achieve a query complexity only depending on this (unknown) parameter, namely of the form $\tilde{O}(2^{k^*(f,g,O(\epsilon))/2})$.[2] Moreover, our algorithm offers a much stronger guarantee: namely, it allows *tolerant* isomorphism testing.

**Theorem 1.3** (Tolerant isomorphism testing)**.** *There exists an algorithm that, given query access to two functions $f, g\colon \{-1,1\}^n \to \{-1,1\}$ and parameter $\epsilon \in (0,1)$, satisfies the following, for some absolute constant $C \geq 1$.*

[2] It is worth noting that this parameter can be much lower that the actual number of relevant variables for either functions; for instance, there exist functions depending on *all* $n$ variables, yet that are $o(1)$-close to $O(1)$-juntas.

- *If $f$ and $g$ are $\frac{\epsilon}{C}$-close to isomorphic, then the algorithm accepts with high constant probability.*
- *If $f$ and $g$ are $\epsilon$-far from isomorphic, then the algorithm rejects with high constant probability.*

*The query complexity of the algorithm is $\tilde{O}(2^{\frac{k^*}{2}})$ with high-probability (and $\tilde{O}(2^{\frac{n}{2}})$ in the worst case), where $k^* = k^*(f, g, \frac{\epsilon}{C})$.*

The above statement is rather technical, and requires careful parsing. In particular, the parameter $k^*$ is crucially *not* provided as input to the algorithm: instead, it is discovered adaptively by invoking the tolerant tester of Theorem 1.2. This explains the high-probability bound on the query complexity: with some small probability, the algorithm may fail to retrieve the right value of $k^*$ – in which case it may use instead a larger value, possibly up to $n$.

*Remark* 1.4 (On the running time of our algorithms.). We note that, as in previous work on testing juntas, the query complexity depends only on $k$ and $1/\epsilon$ but the running time depends on $n$ (since even querying a single point in $\{-1, 1\}^n$ requires specifying $n$ bits).

### 1.2 Overview and techniques

The proofs of Theorems 1.1 and 1.2 both rely on the notion of the *influence* of a set of variables. Given a Boolean function $f\colon \{-1, 1\}^n \to \{-1, 1\}$ and a set $S \subseteq [n]$, the influence of a set $S$ (denoted $\mathbf{Inf}_f(S)$) is the probability that $f(x) \neq f(y)$ when $x$ and $y$ are selected uniformly subject to the constraint that for any $i \in \bar{S}$, $x_i = y_i$. The relation between the number of relevant variables and the influence of a set was utilized in previous works. For the sake of the discussion, we henceforth let $\mathcal{J}_k$ denote the set of all $k$-juntas.

Our starting point is similar to that of [FKR+04, Bla09]. We partition the $n$ variables into $m = O(k^2)$ parts, which allows us in a sense to remove the dependence on $n$. It is not hard to verify that if $f$ is close to $\mathcal{J}_k$, then there exists $k$ parts for which the following holds. If we denote by $S \subseteq [n]$ the union of variables in these $k$ parts, then the complement set $\bar{S}$ has small influence. On the other hand, Blais [Bla09] showed that if a function is far from $\mathcal{J}_k$, then a random partition into a sufficiently large number of parts ensures the following with high constant probability. For every union $S$ of $k$ parts, the complement set $\bar{S}$ will have large influence. The above gives rise to a $(2^{(1+(o(1))k\log k}/\epsilon)$-query complexity algorithm that distinguishes function $f$ that are $\frac{1}{3}\epsilon$-close to $\mathcal{J}_k$ from functions that are $\epsilon$-far from $\mathcal{J}_k$. The algorithm considers all unions $S \subseteq [n]$ of $k$ parts, estimates the influence of $\bar{S}$, and accepts if there exists a set with sufficiently small estimated influence. In order to obtain an algorithm with better query complexity, we consider two relaxations.

**Parameterized tolerant testing through submodular minimization.** In order to describe the algorithm referred to in Theorem 1.1, it will be useful to introduce the following function. For a Boolean function $f$ and a partition $\mathcal{I}$, we let $h\colon 2^{[m]} \to [0, 1]$ be defined as $h(T) \stackrel{\text{def}}{=} \mathbf{Inf}_f(\bigcup_{i\in T} I_i)$. The starting point of our approach is the observation that the exhaustive search algorithm described previously can be seen as performing a brute-force minimization of $h$, under a cardinality constraint. Indeed, it effectively goes over all sets $T \subseteq [m]$ of size $m - k$, estimates $h(T)$, and accepts if there exists a set $T$ for which the estimated value is sufficiently small. With this view, it is natural to ask whether this minimization cannot be performed more efficiently, by exploiting the structural properties of the function $h$: namely that, by the properties of the influence, $h$ is submodular. That is, for every two sets $T_1 \subseteq T_2$ and variable $i \notin T_1$, it holds that $h(T_1 \cup \{i\}) - h(T_1) \geq h(T_2 \cup \{i\}) - h(T_2)$. While it is possible to find the minimum value of a submodular function in polynomial time, if a cardinality constraint is introduced, then even finding an approximate minimum is hard [SF11]. In light of the hardness of the problem, we design an algorithm for the following *bi-criteria relaxation*. Given oracle access to a non-negative submodular function $h\colon 2^m \to \mathbb{R}$ and input parameters $\epsilon \in (0,1)$ and $k \in \mathbb{N}$, the algorithm distinguishes between the following cases:

- There exists a set $T$ such that $|T| \geq m-k$ and $h(T) \leq \epsilon$;
- For every set $T$ such that $|T| \geq m-2k$, $h(T) > 2\epsilon$.

Moreover, the algorithm can be adapted to the case where it is only granted access to an approximate oracle for $h$ (for a precise statement, see Theorem 4.1). This is critical in our setting, since $h(T) = \mathbf{Inf}_f(\bigcup_{i\in T} I_i)$, and we can only estimate the influence of sets of variables.

**Behind Theorem 1.2: biased influence and recycling queries.** The key idea behind our second approach is the following. The exhaustive search algorithm estimates the influence of every set $\bar{S}$ separately, by performing pairs of queries specifically designed for that set. Namely, it queries the value of the function on pairs of points that agree on the set $S$. If it was possible to use the same queries for estimating the influence of different sets, then we could reduce the query complexity. We show that this can be done if we consider a generalized notion of the influence. Given $\rho \in (0,1/2]$, the *$\rho$-biased influence* of a set $S \subseteq [n]$ (denoted as $\mathbf{Inf}_f^{\rho}(S)$), is the probability that $f(x) \neq f(y)$ where $x$ is drawn uniformly at random and $y$ is obtained from $x$ in the following way. For every $i \in \bar{S}$, $x_i = y_i$ and for every $i \in S$, $y_i = x_i$ with probability $1-\rho$ and $y_i = \neg x_i$ with probability $\rho$.[3]

Our main technical result is an algorithm which ensures that, for every $\epsilon, \gamma \in (0,1)$ and $\rho \in (0, \frac{1}{4}]$, with probability at least $1 - o(1)$ the following holds simultaneously for all $S \subseteq [n]$ such that $|S| > n-k$:

- if $\mathbf{Inf}_f^{\rho}(S) > 2\rho\epsilon$, than the estimate $\hat{\nu}_S$ of the $\rho$-biased influence of $S$ is within a multiplicative factor of $(1 \pm \gamma)$ of its true value;
- if $\mathbf{Inf}_f^{\rho}(S) < \rho\epsilon$, than the estimate $\hat{\nu}_S$ of the $\rho$-biased influence of $S$ does not exceed $(1+\gamma)\rho\epsilon$.

The query complexity of the algorithm is $O\Big(\frac{k \log n}{\gamma^2 \rho\epsilon(1-\rho)^k}\Big)$.

**Application to isomorphism testing: tolerant testing and noisy samplers.** The structure of our tolerant isomorphism testing algorithm is quite intuitive, and consists of two phases. In the first phase, we run a linear search on $k$, repeatedly invoking our tolerant tester to discover the smallest value $k$ satisfying $\min(\operatorname{dist}(f,\mathcal{J}_k), \operatorname{dist}(g,\mathcal{J}_k)) \leq \epsilon/C$. We note that a similar approach using a tester whose tolerance is only $\operatorname{poly}(\epsilon/k)$ might return a much larger value of $k$, since as $k$ increases the allowed tolerance decreases. In the second phase, we use this value of $k$ to tolerantly test isomorphism between $f$ and $g$. This phase, however, is not as straightforward as it seems: indeed, to achieve the desired query complexity, we would like to test isomorphism – for which we have known algorithms – between $f_k$ and $g_k$, that is, the $k$-juntas closest to $f$ and $g$ respectively.

Yet here, we face two issues: (i) we do not have query access to $f_k$ and $g_k$; (ii) even in the completeness case $f_k$ and $g_k$ *need not actually be isomorphic.* Indeed, $f$ and $g$ are only promised to be close to $k$-juntas, and close to isomorphic. Hence, the corresponding juntas are only guaranteed to be *close* to isomorphic.

[3] We note that when applying the $\rho$-biased influence on the set $[n]$, we actually get the stability of the function $f$ at $(1-2\rho)$. See Chapter 2.4 in [O'D14].

Addressing item (ii) relies on adapting the algorithm of [ABC+13], along with a careful and technical analysis of the distribution of the points it queries. (This analysis is also the key to providing the tolerance guarantees of our isomorphism tester.) We address item (i) as follows. Our algorithm builds on the ideas of Chakraborty et al. [CGM11], namely on their notion of a "noisy sampler". A noisy sampler is given query access to a function that is promised to be close to some $k$-junta and provides (almost) uniformly distributed samples labeled (approximately) according to this $k$-junta. While the [CGM11] noisy sampler works for functions that are $\operatorname{poly}(\epsilon/k)$-close to $\mathcal{J}_k$, we need a noisy sampler that works for functions that are only $\frac{\epsilon}{C}$-close to $\mathcal{J}_k$. To this end, we replace the weakly tolerant testing algorithm of [Bla09] used in the noisy sampler of [CGM11] with our tolerant testing algorithm. The query complexity of the resulting noisy sampler is indeed much higher than that of [CGM11]. However, this does not increase the overall query complexity of our tolerant isomorphism testing algorithm, as stated in Theorem 1.3.

### 1.3 Organization of the paper

After introducing the necessary notations and definitions in Section 2, we describe in Section 3 the common starting point of our algorithms – the reduction from $n$ variables to $O(k^2)$ parts. Section 4 then contains the details of the submodular minimization under cardinality constraint underlying Theorem 1.1, which is then implemented in Section 5 with an approximate submodular minimization primitive. We then turn in Section 6 to the proof of Theorem 1.2, before describing in Section 7 how to leverage it to obtain our instance-by-instance tolerant isomorphism testing result.

## 2 Preliminaries

### 2.1 Property testing, tolerance, and juntas

A *property* $\mathcal{P}$ of Boolean functions is a subset of all these functions, and we say a function $f$ *has the property* $\mathcal{P}$ if $f \in \mathcal{P}$. The distance between two functions $f, g \colon \{-1,1\}^n \to \{-1,1\}$ is defined as their (normalized) Hamming distance $\operatorname{dist}(f,g) \stackrel{\text{def}}{=} \Pr_x[\, f(x) \neq g(x)\,]$, where $x$ is drawn uniformly at random. Accordingly, for a function $f$ and a property $\mathcal{P}$ we define the distance from $f$ to $\mathcal{P}$ as $\operatorname{dist}(f,\mathcal{P}) \stackrel{\text{def}}{=} \min_{g\in\mathcal{P}} \operatorname{dist}(f,g)$. Given $\epsilon \geq 0$ and a property $\mathcal{P}$, we will say a function $f$ is $\epsilon$-*far* from $\mathcal{P}$ (resp. $\epsilon$-*close* to $\mathcal{P}$) if $\operatorname{dist}(f,\mathcal{P}) > \epsilon$ (resp. $\operatorname{dist}(f,\mathcal{P}) \leq \epsilon$).

We can now give a formal definition of a property testing algorithm.

**Definition 2.1.** A *testing algorithm for a property* $\mathcal{P}$ is a probabilistic algorithm that gets an input parameter $\epsilon$ and oracle access to a function $f \colon \{-1,1\}^n \to \{-1,1\}$. The algorithm should output a binary verdict that satisfies the following two conditions.

- If $f \in \mathcal{P}$ then the algorithm accepts $f$ with probability at least $2/3$.
- If $\operatorname{dist}(f,\mathcal{P}) > \epsilon$, then the algorithm rejects $f$ with probability at least $2/3$.

A testing algorithm has *one-sided error* if it accepts every object that satisfies the property with probability 1. Otherwise, it has *two-sided error*.

Next, we define the notion of *tolerant* testing algorithm, a testing algorithm that is also required to accept functions merely close to the property:

**Definition 2.2.** A *tolerant testing algorithm for a property* $\mathcal{P}$ is a probabilistic algorithm that gets two input parameters $\epsilon_1, \epsilon_2 \in [0,1]$ such that $\epsilon_1 < \epsilon_2$, and oracle access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$. The algorithm should output a binary verdict that satisfies the following two conditions.

- If $\operatorname{dist}(f, \mathcal{P}) \leq \epsilon_1$ then the algorithm accepts $f$ with probability at least $2/3$.
- If $\operatorname{dist}(f, \mathcal{P}) > \epsilon_2$, then the algorithm rejects $f$ with probability at least $2/3$.

In some cases one may want to consider a relaxation of the definition of tolerant testing to the following *tolerant testing of parameterized properties* .

**Definition 2.3** (Tolerant Testing of Parameterized Properties)**.** Let $\mathcal{P} = (\mathcal{P}_s)_{s\in\mathbb{N}}$ be a non-decreasing family of properties parameterized by $s \in \mathbb{N}$, i.e. such that $\mathcal{P}_s \subseteq \mathcal{P}_t$ whenever $s \leq t$; and $\sigma\colon \mathbb{N} \to \mathbb{N}$ be a non-decreasing mapping. A *$\sigma$-tolerant testing algorithm for* $\mathcal{P}$ is a probabilistic algorithm that gets three input parameters $s \in \mathbb{N}$ and $\epsilon_1, \epsilon_2 \in [0,1]$ such that $\epsilon_1 < \epsilon_2$, as well as oracle access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$. The algorithm should output a binary verdict that satisfies the following two conditions.

- If $\operatorname{dist}(f, \mathcal{P}_s) \leq \epsilon_1$ then the algorithm accepts $f$ with probability at least $2/3$.
- If $\operatorname{dist}\Big(f, \mathcal{P}_{\sigma(s)}\Big) > \epsilon_2$, then the algorithm rejects $f$ with probability at least $2/3$.

The main focus of this work will be the property of being a *junta*, that is a Boolean function that only truly depends on a (small) subset of its variables:

**Definition 2.4** (Juntas)**.** A Boolean function $f\colon \{\pm 1\}^n \to \{\pm 1\}$ is said to be a *$k$-junta* if there exists a set $J \subseteq [n]$ of size at most $k$, such that $f(x) = f(y)$ for every two assignments $x, y \in \{\pm 1\}^n$ that satisfy $x_i = y_i$ for every $i \in J$. We let $\mathcal{J}_k$ denote the set of all $k$-juntas (over $n$ variables).

**Notations.** Hereafter, we denote by log the binary logarithm, by $[n]$ the set of integers $\{1, \ldots, n\}$, and write $\mathcal{S}_n$ for the set of permutations of $[n]$. Given two disjoint sets $S, T \subseteq [n]$ and two partial assignments $x \in \{\pm 1\}^{|S|}$ and $y \in \{\pm 1\}^{|T|}$, we let $x \sqcup y \in \{\pm 1\}^{|S \cup T|}$ be the partial assignment whose $i$-th coordinate is $x_i$ if $i \in S$ and $y_i$ if $i \in T$. Finally, given a Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$ we write $\mathcal{O}_f$ for an oracle providing query access to $f$.

## 2.2 Fourier analysis over the hypercube

Throughout this work, our main object of interest are functions over the discrete cube $\{-1,1\}^n$, equipped with the uniform measure. Real-valued functions over the discrete cube can be expressed by their Fourier decomposition in the following way.

**Definition 2.5** (Characters and Fourier Coefficients)**.** Let $S \subseteq [n]$. The *character* $\chi_S$ is the function over $\{-1,1\}^n$, defined by $\chi_S(x) \stackrel{\text{def}}{=} \prod_{i\in S} x_i$. Given a function $f\colon \{-1,1\}^n \to \mathbb{R}$, its expansion as a linear combination of the characters

$$f(x) = \sum_{S\subseteq[n]} \hat{f}(S)\chi_S(x) \ ,$$

is called the *Fourier expansion* of $f$, and $\hat{f}(S)$ is the *Fourier coefficient* of $S$.

The set of all characters forms an orthonormal basis with respect to the following natural inner product.

$$\langle f, g \rangle \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{x \sim \{-1,1\}^n}[f(x)g(x)] \ .$$

A useful property of the characters is that for any $S \subseteq [n]$ and every $x, y \in \{-1,1\}^n$, $\chi_S(x \oplus y) = \chi_S(x)\chi_S(y)$.

A key notion in this work is the notion of *influence* of a set, which generalizes the standard notion of influence of a variable:

**Definition 2.6** (Set-influence)**.** For a Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$, the *set-influence* of a set $S \subseteq [n]$ is defined as

$$\mathbf{Inf}_f(S) = 2\Pr[\, f(x \sqcup u) \neq f(x \sqcup v)\,]$$

where $x \sim \{-1,1\}^{[n]\setminus S}$, and $u, v \sim \{-1,1\}^S$.

The following proposition establishes the Fourier representation of the set-influence.

**Fact 2.7.** *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ where $\{-1,1\}^n$ is equipped with the uniform measure. Then, for any set $S \subseteq [n]$,*

$$\mathbf{Inf}_f(S) = \sum_{T : S \cap T \neq \emptyset} \hat{f}(T)^2 \ .$$

# 3 From $n$ variables to $O(k^2)$ parts

In this section we build on techniques from [FKR+04, Bla09] and describe how to reduce the problem of testing closeness to a $k$-junta to testing closeness to a *$k$-part* junta (defined below). The advantage of doing so is that while the former question is about functions on $n$ variables, the latter *does no longer involve $n$ as a parameter*: only $k$ and $\epsilon$ now have a role to play. We start with a useful definition and two results that we shall require: that of *$k$-part juntas*, and their properties with regard to random partitions of the domain.

**Definition 3.1** (Partition juntas [Bla12, Definition 5.3], extended)**.** Let $\mathcal{I}$ be a partition of $[n]$, and $k \geq 1$. The function $f\colon \{-1,1\}^n \to \{-1,1\}$ is a *$k$-part junta with respect to $\mathcal{I}$* if the relevant coordinates in $f$ are all contained in at most $k$ parts of $\mathcal{I}$. Moreover,

(i) $f$ is said to *$\epsilon$-approximate being a $k$-part junta with respect to $\mathcal{I}$* if there exists a set $J$ formed by taking the union of $k$ parts in $\mathcal{I}$ satisfying $\mathbf{Inf}_f(\bar{J}) \leq 2\epsilon$.

(ii) Conversely, $f$ is said to *$\epsilon$-violate being a $k$-part junta with respect to $\mathcal{I}$* if for every set $J$ formed by taking the union of $k$ parts in $\mathcal{I}$, $\mathbf{Inf}_f(\bar{J}) > 2\epsilon$.

**Lemma 3.2** ([Bla12, Lemma 5.4])**.** *For $f\colon \{-1,1\}^n \to \{-1,1\}$ and $k \geq 1$, let $\alpha \stackrel{\text{def}}{=} \operatorname{dist}(f, \mathcal{J}_k)$. Also, let $\mathcal{I}$ be a random partition of $[n]$ with $s \stackrel{\text{def}}{=} 24k^2$ parts obtained by uniformly and independently assigning each coordinate to a part. With probability at least $5/6$ over the choice of the partition, $f$ $\frac{\alpha}{2}$-violates being a $k$-part junta with respect to $\mathcal{I}$.*

**Lemma 3.3.** *For $f\colon \{-1,1\}^n \to \{-1,1\}$ and $k \geq 1$, let $\alpha \stackrel{\text{def}}{=} \operatorname{dist}(f, \mathcal{J}_k)$ and let $\mathcal{I}$ be any partition of $[n]$ into $s \geq k$ parts. Then $f$ $2\alpha$-approximates being a $k$-part junta with respect to $\mathcal{I}$.*

**Proof:** Let $g \in \mathcal{J}_k$ be such that $\operatorname{dist}(f,g) = \operatorname{dist}(f,\mathcal{J}_k) = \alpha$. Let $I_{i_1},\dots,I_{i_\ell}$ be the $\ell \le k$ parts of $\mathcal{I}$ containing the relevant variables of $g$. Then, for any set $S \subset [s]$ of size $k$ such that $\{i_1,\dots,i_\ell\} \subseteq S$, we have that for $J \leftarrow \bigcup_{i\in S} I_i$, when drawing $x \sim \{-1,1\}^J$, and $u,v \sim \{-1,1\}^{\bar{J}}$ the following holds.

$$\begin{aligned}\mathbf{Inf}_f(\bar{J}) = 2\Pr[\, f(x \sqcup u) \neq f(x \sqcup v)\,] &\le 2\Pr[\, f(x \sqcup u) \neq g(x \sqcup u) \text{ or } f(x \sqcup v) \neq g(x \sqcup v)\,] \\ &\le 2\left(\Pr[\, f(x \sqcup u) \neq g(x \sqcup u)\,] + \Pr[\, f(x \sqcup v) \neq g(x \sqcup v)\,]\right) \le 2\,(\alpha+\alpha) = 4\alpha\ ,\end{aligned}$$

where the first inequality follows from observing that (as $g$ does not depend on variables in $\bar{J}$) one can only have $f(x \sqcup u) \neq f(x \sqcup v)$ if at $f$ disagrees with $g$ on at least one of the two points; and the third inequality holds since both $x \sqcup u$ and $x \sqcup v$ are uniformly distributed. ■

The above two lemmas suggest the following approach. We would like to distinguish functions that are $\epsilon'$-close to some $k$-junta and functions that are $\epsilon$-far from every $k'$-junta. Suppose we select a random partition of $[n]$ into $O(k^2)$ parts. Then with high probability over the choice of the partition, it is sufficient to distinguish between functions that $2\epsilon'$-approximate being a $k$-junta and functions that $\epsilon/2$-violate being a $k'$-part junta. Specifically, we get the proposition below, which we apply throughout this work:

**Proposition 3.4** (Reduction to part juntas)**.** *Let $\mathcal{T}$ be an algorithm that is given query access to a function $f : \{-1,1\}^n \to \{-1,1\}$, a partition $\mathcal{I} = (I_1,\dots,I_m)$ of $[n]$ into $m$ parts, and parameters $k \in \mathbb{N}$ and $\epsilon \in (0,1)$. Suppose that $\mathcal{T}$ performs $q(k,\epsilon,m)$ queries to $f$ and satisfies the following guarantees, for a pair of functions $\ell : (0,1) \times \mathbb{N} \to (0,1)$ and $\ell' : \mathbb{N} \to \mathbb{N}$.*

- *If $f$ $\epsilon'$-approximates being a $k$-part junta with respect to $\mathcal{I}$ and $\epsilon' \le \ell(\epsilon,k)$, then $\mathcal{T}$ returns accept with probability at least $5/6$;*
- *If $f$ $\epsilon$-violates being a $k'$-part junta with respect to $\mathcal{I}$ and $k' \ge \ell'(k)$, then $\mathcal{T}$ returns reject with probability at least $5/6$.*

*Then there exists an algorithm $\mathcal{T}'$, that given query access to $f$ and parameters $k \in \mathbb{N}$ and $\epsilon \in (0,1)$, satisfies the following.*

- *If $\operatorname{dist}(f,\mathcal{J}_k) \le \frac{\epsilon'}{2}$ and $\epsilon' \le \ell(\epsilon,k)$, then $\mathcal{T}'$ outputs accept with probability at least $2/3$;*
- *If $\operatorname{dist}(f,\mathcal{J}_{k'}) > 2\epsilon$ and $k' \ge \ell'(k)$, then $\mathcal{T}'$ outputs reject with probability at least $2/3$.*

*Moreover, the algorithm $\mathcal{T}'$ has query complexity $q(k,\epsilon,O(24k^2))$.*

**Proof of Proposition 3.4:** The algorithm $\mathcal{T}'$ first obtains a random partition $\mathcal{I}$ of $[n]$ into $m \stackrel{\text{def}}{=} 24k'^2$ parts by uniformly and independently assigning each coordinate to a part. $\mathcal{T}'$ then invokes $\mathcal{T}$ with parameters $\epsilon, k, m$ and the partition $\mathcal{I}$. By Lemma 3.2 and the choice of $m$, with probability at least $5/6$ the partition $\mathcal{I}$ is *good* in the following sense. For $\alpha = \operatorname{dist}(f,\mathcal{J}_{k'})$, it holds that $f$ $\frac{\alpha}{2}$-violates being a $k$ junta with respect to $\mathcal{I}$. Conditioned on $\mathcal{I}$ being good, and by Lemma 3.3, we are guaranteed that the following holds.

(i) If $\operatorname{dist}(f,\mathcal{J}_k) \le \frac{\epsilon'}{2}$, then $f$ $\epsilon'$-approximates being a $k$-part junta with respect to $\mathcal{I}$;

(ii) If $\operatorname{dist}(f,\mathcal{J}_{k'}) > 2\epsilon$, then $f$ $\epsilon$-violates being a $k'$-part junta with respect to $\mathcal{I}$.

Therefore, $\mathcal{T}$ will then answer as specified by the proposition with probability at least $5/6$, making $q(\epsilon,k,m)$ queries. Overall, $\mathcal{T}'$ is thus successful with probability at least $2/3$ by a union bound. ■

As an illustration of the above technique, and a warmup towards the (more involved) algorithms of the next sections, we show how to obtain an algorithm $\mathcal{T}'$ as specified in Proposition 3.4 with query complexity $2^{(1+o(1))k\log k}/\epsilon$. Given a partition $\mathcal{I}$ of $[n]$ into $m$ parts, $\mathcal{T}$ considers all $\binom{m}{k}$ sets of variables that result from taking the union of $k$ parts. For each such set $S$, it obtains an estimate $\widetilde{\mathbf{Inf}}_f(\bar{S})$ of the influence of $\bar{S}$, by performing $O\left(\frac{\log m}{\epsilon}\right)$ queries to $f$. $\mathcal{T}$ accepts if for at least one of the sets $S$, $\widetilde{\mathbf{Inf}}_f(\bar{S})$ is at most $\frac{3}{2}\epsilon$. Performing $O(\frac{\log m}{\epsilon})$ queries to the oracle for each set, ensures that the following holds with high constant probability. For every set $S$ such that $\mathbf{Inf}_f(S) \leq \frac{4}{3}\epsilon$, $\widetilde{\mathbf{Inf}}_f(S) \leq \frac{3}{2}\epsilon$ and for every set $S$ such that $\mathbf{Inf}_f(S) > 2\epsilon$, $\widetilde{\mathbf{Inf}}_f(S) > \frac{3}{2}\epsilon$. Hence, the algorithm $\mathcal{T}$ fulfills the requirements stated in Proposition 3.4, and it follows that:

- If $f$ is $\frac{1}{3}\epsilon$-close to some $k$-junta then $\mathcal{T}'$ accepts with probability at least $2/3$.
- If $f$ is $\epsilon$-far from every $k$-junta then $\mathcal{T}'$ rejects with probability at least $2/3$.

Since $m = 24k^2$, the query complexity of the algorithm is $\binom{m}{k} \cdot O(\frac{\log m}{\epsilon}) = 2^{(1+o(1))k\log k}/\epsilon$.

# 4 Approximate submodular minimization under cardinality constraints

In this section we show how a certain bi-criteria approximate version of submodular minimization with a cardinality constraint can be reduced to approximate submodular minimization with no cardinality constraint. This reduction holds even when given *approximate oracle access* to the submodular function, and is meaningful when the cardinality constraint is sufficiently large. More precise details follows.

**Definition 4.1** (Approximate oracle)**.** Let $h\colon 2^{[m]} \to \mathbb{R}$ be a function. An *approximate oracle* for $h$, denoted $\mathcal{O}_h^{\pm}$, is a randomized algorithm which, for any input $S \subseteq [m]$ and parameters $\tau, \delta \in (0,1)$, returns a value $\tilde{h}(S)$ such that $|\tilde{h}(S) - h(S)| \leq \tau$ with probability at least $1-\delta$

**Definition 4.2** (Approximate submodular minimization algorithm)**.** Let $h\colon 2^{[m]} \to \mathbb{R}$ be a non-negative submodular function and let $\mathcal{O}_h^{\pm}$ denote an approximate oracle for $h$. An *approximate submodular function minimization* algorithm (ASFM) is an algorithm that, when given access to $\mathcal{O}_h^{\pm}$ and called with input parameters $\xi$ and $\delta$, returns a value $\nu$ such that $|\nu - \min_S\{h(S)\}| \leq \xi$ with probability at least $1-\delta$.

In Corollary 5.4 in Section 5 we establish the existence of such an algorithm. The running time of the algorithm is polynomial in $m$, logarithmic in the maximal value of the function and linear in the running time of the approximate oracle. We next present an algorithm for approximate submodular minimization under cardinality constraints.

**Algorithm 1** Submodular Minimization under Constraint$(\mathcal{O}_h^{\pm}, \epsilon, \xi, k)$

1: Let $h'(S) = h(S) - \frac{\epsilon}{k}|S|$ so that for every $\tau', \delta'$ $\mathcal{O}_{h'}^{\pm}(S, \tau', \delta') = \mathcal{O}_h^{\pm}(S, \tau', \delta') - \frac{\epsilon}{k}|S|$ .
2: Let $\nu$ be the returned value from invoking an ASFM algorithm with access to $\mathcal{O}_{h'}^{\pm}$ and parameters $\xi$ and $\delta$.
3: Accept if and only if $\nu \leq (1 - \frac{m-k}{k}) \cdot \epsilon + \xi$.

**Theorem 4.1.** *For a submodular function $h$, Algorithm 1 satisfies the following conditions:*

1. *If there exists a set $S$ such that $|S| \geq m-k$ and $h(S) \leq \epsilon$, then the algorithm accepts with probability at least $1-\delta$.*

2. *If for every set $S$ such that $|S| \geq m - 2(1+\frac{\xi}{\epsilon})k$, we have $h(S) > 2(\epsilon+\xi)$, then the algorithm rejects with probability at least $1-\delta$.*

*(Moreover, the second item can be strengthened so that it holds for functions $h$ which satisfy the following: (i) for every set $S$ such that $|S| \geq m-k$, $h(S) > 2\epsilon + 2\xi$ and (ii) for every set $S$ such that $|S| \geq m - 2(1+\frac{\xi}{\epsilon})k$, $h(S) > \epsilon + 2\xi$ . )*

**Proof:** By Definition 4.2, with probability at least $1-\delta$ the value $\nu$ defined in Step 2 of the algorithm satisfies

$$|\nu - \min_S\{h'(S)\}| \leq \xi \ . \tag{1}$$

We start with proving the first item. If there exists a set $S^*$ such that $|S^*| \geq m-k$, then by Equation (1) and the definition of $h'$,

$$\nu \leq \min_S h'(S) + \xi \leq h'(S^*) + \xi \leq h(S^*) - \frac{\epsilon}{k}|S^*| + \xi \leq \epsilon - \frac{m-k}{k}\epsilon + \xi = \left(1 - \frac{m-k}{k}\right)\epsilon + \xi \ .$$

Hence, the algorithm accepts.

We now prove that if the algorithm accepts (and conditioning on Equation (1) holding), then either (i) there exists a set $|S^*|$ such that $|S^*| \geq m-k$ and $h(S^*) \leq 2\epsilon+2\xi$ or (ii) $|S^*| \geq m-2(1+\frac{\xi}{\epsilon})k$ and $h(S^*) \leq \epsilon+2\xi$. We let $S^* \stackrel{\text{def}}{=} \text{argmin}_S\{h'(S)\}$, and divide the analysis into two cases, depending on $|S^*|$.

- If $|S^*| \geq m-k$, then since $\min_S\{h'(S)\} \leq \nu + \xi$ and $\nu \leq (1-\frac{m-k}{k})\epsilon + \xi$,

$$h(S^*) = h'(S^*) + \frac{\epsilon}{k}|S^*| \leq \nu + \xi + \frac{\epsilon}{k}\cdot m \leq \left(1 - \frac{m-k}{k}\right)\epsilon + 2\xi + \frac{\epsilon m}{k} = 2\epsilon + 2\xi,$$

  as claimed in the item (i).
- If $|S^*| \leq m-k$, then

$$h(S^*) = h'(S^*) + \frac{\epsilon}{k}|S^*| \leq \nu + \xi + \frac{\epsilon}{k}\cdot(m-k) \leq \left(1 - \frac{m-k}{k}\right)\epsilon + 2\xi + \frac{\epsilon(m-k)}{k} = \epsilon + 2\xi.$$

  Also, since for every set $S$, $h(S) \geq 0$ and $h'(S^*) \leq \nu + \xi$, it holds that

$$\frac{\epsilon}{k}|S^*| = h(S^*) - h'(S^*) \geq (\frac{m-k}{k} - 1)\epsilon - 2\xi.$$

  Therefore, $|S^*| \geq m - (2 + \frac{2\xi}{\epsilon})k$, and item (ii) holds.

■

# 5 Approximate submodular function minimization

In this section we use results from [LSW15] to obtain an approximate submodular minimization algorithm, as defined in Definition 4.2. This is done in three steps: (1) We use the known fact that the problem of finding the minimum of a submodular function $g$ can be reduced to finding the minimum of the Lovász extension for that function, denoted $\mathcal{L}_g$. (2) We then extend the results of [LSW15] (and specifically of Theorem 61) and provide a noisy separation oracle for $\mathcal{L}_g$ when only given approximate oracle access to the function $g$. (3) Finally, we apply Theorem 42 from [LSW15], which provides an algorithm that, when given access to a separation oracle for a function, returns an approximation to that function's minimum value.

We start with the following definition of the Lovász extension of a submodular function.

**Definition 5.1** (Lovász Extension)**.** Given a submodular function $g\colon 2^{[m]} \to \mathbb{R}$, the *Lovász extension* of $g$, is a function $\mathcal{L}_g\colon [0,1]^m \to \mathbb{R}$ which is defined for all $x \in [0,1]^m$ by

$$\mathcal{L}_g(x) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{t\sim[0,1]} [g(\{\ i:\ x_i \geq t\ \})] \ ,$$

where $t \sim [0,1]$ denotes that $t$ is drawn uniformly at random from $[0,1]$.

The following theorem is standard in combinatorial optimization (see e.g. [Bac13] and [GLS12, Sch02]) and contains useful properties of the Lovász extension.

**Theorem 5.1.** *The Lovász extension $\mathcal{L}_g$ of a submodular function $g\colon 2^{[m]} \to \mathbb{R}$ satisfies the following properties.*

1. *$\mathcal{L}_g$ is convex and* $\min_{x\in[0,1]^m}\{\mathcal{L}_g(x)\} = \min_{S\subseteq[m]}\{g(S)\}$.
2. *If $x_1 \geq \ldots \geq x_m$ , then*
$$\mathcal{L}_g(x) = \sum_{i=1}^{m} (g([i]) - g([i-1]))x_i \ .$$

By the first item of Theorem 5.1, in order to approximate the minimum value of a submodular function $g$, it suffices to approximate the minimum of its Lovász extension. As discussed at the start of the section, this is done by providing a *separation oracle* for $\mathcal{L}_g$.

**Definition 5.2** (Noisy Separation Oracle [LSW15, Definition 2])**.** Let $h$ be a convex function over $\mathbb{R}^m$ and let $\Omega$ be a convex set in $\mathbb{R}^m$. A *separation oracle for $h$* with respect to $\Omega$ is an algorithm that for an input $x \in \Omega$ and parameters $\eta, \gamma \geq 0$ satisfies the follows. It either asserts that $h(x) \leq \min_{y\in\Omega}\{h(y)\} + \eta$ or it outputs a halfspace $H \stackrel{\text{def}}{=} \{z : a^T z \leq a^T x + c\}$ such that

$$\{\ y \in \Omega :\ h(y) \leq h(x)\ \} \subset H \ ,$$

where $a \in [0,1]^m$, $a \neq 0$, and $c \leq \gamma\|a\|_2$.

In Theorem 61 in [LSW15] it is shown how to define a separation oracle for a function $g$ when given *exact* query access to $g$; we adapt the proof to the case where one is only granted access to an approximate oracle for $g$, and the resulting procedure has small failure probability.

**Algorithm 2** Separation Oracle $(\mathcal{O}_g^{\pm}, \bar{x}, \eta, \gamma, \delta)$

1: Assume without loss of generality that $\bar{x}_1 \geq \bar{x}_2 \geq \ldots \geq \bar{x}_m$ (otherwise re-index the coordinates).
2: Let $\tau = \min\{\eta/4m, \gamma/2m\}$.
3: For each $i$, let $\tilde{g}([i])$ be the returned value from invoking $\mathcal{O}_g^{\pm}$ on the set $[i]$ with parameters $\frac{\tau^2}{2}$ and $\frac{\delta}{m}$.
4: Define $\tilde{a} \in \mathbb{R}^m$ by $\tilde{a}_i \stackrel{\text{def}}{=} \tilde{g}([i]) - \tilde{g}([i-1])$ for each $i \in [m]$.
5: Let $\tilde{\mathcal{L}}_g(\bar{x}) \stackrel{\text{def}}{=} \tilde{a}^T \bar{x}$.
6: **if** for every $i \in [m]$, $|\tilde{a}_i| < \tau$ **then**
7:     **return** $\bar{x}$, which satisfies "$\mathcal{L}_g(\bar{x}) \leq \min_{y \in [0,1]^m}\{L_g(y)\} + \eta$".
8: **else**
9:     **return** the halfspace $H = \{z : \tilde{a}^T z \leq \tilde{\mathcal{L}}_g(\bar{x}) + 2\tau m \|a\|_2\}$ .
10: **end if**

**Lemma 5.3.** *Let $g\colon 2^{[m]} \to \mathbb{R}$ be a convex function, and let $\Phi_g(\cdot,\cdot)$ denote the running time of the approximate oracle for $g$. For every $x \in [0,1]^m$, $\eta, \gamma, \delta \in (0,1)$, with probability at least $1-\delta$, Algorithm 2 satisfies the guarantees of a separation oracle for $\mathcal{L}_g$ (with respect to $[0,1]^m$). The algorithm makes $m$ queries to $\mathcal{O}_g^{\pm}$ with parameters $\tau^2/2$ and $\delta/m$, where $\tau = \min\{\eta/4m, \gamma/2m\}$, and its running time is $m \cdot \Phi_g(\frac{\tau^2}{2}, \delta/m)$.*

In order to prove the above lemma we will use the following theorem from [LSW15].

**Theorem 5.2** ([LSW15, Theorem 61], restated)**.** *Let $g\colon 2^m \to \mathbb{R}$ be a submodular function. For every $x \in [0,1]^m$,*

$$\sum_{i=1}^{m} (g([i]) - g([i-1]))x_i \leq \mathcal{L}_g(x) \; .$$

**Proof of Lemma 5.3:** For every $i \in [m]$, let $a_i \stackrel{\text{def}}{=} g([i]) - g([i-1])$, and note that by a union bound over all $i \in [m]$, we have that with probability at least $1-\delta$, $\max_{i\in[m]}|g([i]) - \tilde{g}([i])| \leq \tau^2/2$. We henceforth condition on this, and observe that this implies that, for any $y \in [0,1]^m$,

$$|\tilde{a}^T y - a^T y| \leq 2m \cdot \frac{\tau^2}{2} = m\tau^2 \; . \tag{2}$$

We next consider two cases. Assume first that there exists an index $i \in [m]$ such that $|\tilde{a}_i| \geq \tau$. That is, assume that the condition in Step 6 does not hold. Then we prove that for every $y \in [0,1]^m$ such that $\mathcal{L}_g(y) \leq \mathcal{L}_g(\bar{x})$ it holds that $y \in H$, where $H$ is the halfspace defined in Step 9 of the algorithm.

By Theorem 5.2, we have that for any $y \in [0,1]^m$, $\sum_{i=1}^m a_i \cdot y_i \leq \mathcal{L}_g(y)$. Since $\mathcal{L}_g(y) \leq \mathcal{L}_g(\bar{x})$, we get that

$$\tilde{a}^T y \leq a^T y + m\tau^2 \leq \mathcal{L}_g(y) + m\tau^2 \leq \mathcal{L}_g(\bar{x}) + m\tau^2 \; . \tag{3}$$

By Theorem 5.1, together with the assumption that the coordinates of $\bar{x}$ are sorted,

$$\mathcal{L}_g(\bar{x}) = \sum_{i=1}^{m} a_i \cdot \bar{x}_i \leq \sum_{i=1}^{m} \tilde{a}_i \cdot \bar{x}_i + m\tau^2 = \tilde{\mathcal{L}}_g(\bar{x}) + m\tau^2. \tag{4}$$

Combining Equation (3) and Equation (4), and since there exists an $i$ such that $|\tilde{a}_i| \geq \tau$,

$$\tilde{a}^T y \leq \tilde{\mathcal{L}}_g(\bar{x}) + 2m\tau^2 \leq \tilde{\mathcal{L}}_g(\bar{x}) + 2m\tau\|\tilde{a}\|_2 \; .$$

This implies that $y$ is in $H$ and that for $b = \tilde{\mathcal{L}}_g(\bar{x})$ and $\gamma = 2\tau m$, $H$ fulfills the requirements of the halfspace defined in Definition 5.2.

Now consider the case that $|\tilde{a}_i| \leq \tau$ for all $i \in [m]$. it follows that for any $y \in [0,1]^m$, $-m\tau \leq \tilde{a}^T y \leq m\tau$. In particular, we have that $-m\tau \leq \tilde{\mathcal{L}}_g(\bar{x}) \leq m\tau$, which implies that for every $y \in [0,1]^m$,

$$\tilde{\mathcal{L}}_g(\bar{x}) - 2m\tau \leq -m\tau \leq \tilde{a}^T y \ .$$

Therefore, for every $y \in [0,1]^m$ we get

$$\tilde{\mathcal{L}}_g(\bar{x}) - 3m\tau \leq \tilde{a}^T y - m\tau \leq a^T y \leq \mathcal{L}_g(y) \ ,$$

where the second inequality follows from Equation (2), and the last inequality follows from Theorem 5.2. Hence, if we let $x^* = \arg\min\{\mathcal{L}_g(x)\}$, we have that

$$\tilde{\mathcal{L}}_g(\bar{x}) \leq \mathcal{L}_g(x^*) + 3m\tau \ .$$

By Equation (4), we have that $\mathcal{L}_g(\bar{x}) \leq \tilde{\mathcal{L}}_g(\bar{x}) + m\tau^2$. Hence, $\mathcal{L}_g(\bar{x}) \leq \mathcal{L}_g(x^*) + 3m\tau + m\tau^2 \leq \mathcal{L}_g(x^*) + 4m\tau$, and we get that $\bar{x}$ satisfies

$$\mathcal{L}_g(\bar{x}) \leq \min_{y \in [0,1]^m} \{\mathcal{L}_g(y)\} + \eta.$$

Therefore with probability at least $1-\delta$ the algorithm satisfies the conditions of a separation oracle with parameters $\eta$ and $\gamma$.

The algorithm performs $m$ queries to the approximate oracle for $g$ with parameters $\tau^2/2$ and $\delta/m$, where $\tau \stackrel{\text{def}}{=} \min\{\eta/4m, \gamma/2m\}$. Hence, the running time of the algorithm is $m\Phi(\frac{\tau^2}{2}, \frac{\delta}{m}) + m\log m$, as it also sorts the coordinates of $\bar{x}$ (in order to re-index the coordinates). ■

We can now use the separation oracle for $\mathcal{L}_g$ and apply the following theorem to get an approximate minimum of $\mathcal{L}_g$, which is also an approximate minimum of $g$.

**Theorem 5.3** ([LSW15, Theorem 42], restated)**.** *Let $h$ be a convex function on $\mathbb{R}^m$ and let $\Omega$ be a convex set with constant min-width[4] that contains a minimizer of $h$. Suppose we have a separation oracle for $h$ and that $\Omega$ is contained inside $B_\infty(R) \stackrel{\text{def}}{=} \{\, x : \ \|x\|_\infty \leq R \,\}$, where $R > 0$ is a constant. Then there is an algorithm which for any $0 < \alpha < 1$ and $\eta$ outputs $x \in \mathbb{R}^n$ such that*

$$h(x) - \min_{y \in \Omega}\{h(y)\} \leq \eta + \alpha \left( \max_{y \in \Omega}\{h(y)\} - \min_{y \in \Omega}\{h(y)\} \right) \ .$$

*In expectation, the algorithm performs $O(m \log(\frac{m}{\alpha}))$ calls to Algorithm 2, and has expected running time of*

$$O\left( m \cdot \mathrm{SO}(\eta,\gamma) \log\left(\frac{m}{\alpha}\right) + m^3 \log^{O(1)}\left(\frac{m}{\alpha}\right) \right) \ ,$$

*where $\gamma = \Theta\left(\frac{\alpha}{m^{3/2}}\right)$ and $\mathrm{SO}(\eta,\gamma)$ denotes the running time of the separation oracle when invoked with parameters $\eta$ and $\gamma$.*

[4] For a compact set $K \subseteq \mathbb{R}^m$, the min-width is defined as $\min_{a \in \mathbb{R}^m :\ \|a\|_2 = 1} \max_{x,y \in K} \langle a, x-y \rangle$. [LSW15, Definition 41]. In particular, it is not hard to see that the set $K = [0,1]^m \subseteq B_\infty(1)$ has unit min-width.

**Corollary 5.4.** *Let $g\colon 2^{[m]} \to \mathbb{R}$ be a submodular function. There exists an algorithm that, when given access to $\mathcal{O}_g^{\pm}$, and for input parameters $\xi, \delta \in (0,1)$, returns with probability at least $9/10 - 2\delta$ a value $\nu \in \mathbb{R}$ such that $\nu \leq \min_{S\subseteq[m]}\{g(S)\} + \xi$.*

*The algorithm performs $m \log\left(\frac{mM}{\xi}\right)$ calls to $\mathcal{O}_g^{\pm}$ with parameters $\frac{\xi^2}{128m^5M^2}$ and $\frac{\delta}{Cm^2\log\left(\frac{mM}{\xi}\right)}$, where $M \stackrel{\text{def}}{=} \max\left\{2\max_{S\subseteq[m]}\{|g(S)|\}, \xi/2\right\}$ and $C > 0$ is an absolute constant. The running time of the algorithm is*

$$O\left(m^2 \cdot \Phi_g\left(\frac{\xi^2}{128m^5M^2}, \frac{\delta}{Cm^2\log\frac{mM}{\xi}}\right)\log\frac{mM}{\xi} + m^3\log^{O(1)}\frac{mM}{\xi}\right) ,$$

*where $\Phi_g$ is the running time of $\mathcal{O}_g^{\pm}$.*

**Proof:** We refer to the algorithm from Theorem 5.3 as the minimization algorithm and apply it to $\mathcal{L}_g$, with Algorithm 2 as a separation oracle. Once the minimization algorithm returns a point $x \in [0,1]^m$, we return the value $\nu = \mathcal{O}_{\mathcal{L}_g}^{\pm}(x, \xi/4, \delta)$.

Let $M' \stackrel{\text{def}}{=} 2\max_{S\subseteq[m]}\{|g(S)|\}$, and recall that $\mathcal{L}_g(x) = \mathop{\mathbf{E}}_{t\sim[0,1]}[g(\{\, i : x_i \geq t \,\})]$. Hence, $\max_{x\in[0,1]^m}\{\mathcal{L}_g(x)\} - \min_{x\in[0,1]^m}\{\mathcal{L}_g(x)\} \leq 2M'$. Setting $\alpha < \xi/(2M)$ and $\eta = \xi/4$ ensures that $0 < \alpha < 1$ and that

$$\eta + \alpha\left(\max_{x\in[0,1]^m}\{\mathcal{L}_g(x)\} - \min_{x\in[0,1]^m}\{\mathcal{L}_g(x)\}\right) \leq \eta + \alpha M' \leq 3\xi/4 \,. \tag{5}$$

The minimization algorithm invokes the separation oracle $C_1 \cdot m\log(m/\alpha) = C_1 \cdot m\log(mM/\xi)$ times in expectation, for some constant $C_1$. If at some points the number of calls to the separation oracle exceeds $10C_1 \cdot m\log(mM/\xi)$, then we halt and return $\mathsf{fail}$. By Markov's inequality this happens with probability at most $1/10$. Hence, every time the minimization algorithm calls the separation oracle with parameters $\eta$ and $\gamma$ we invoke Algorithm 2 with parameters $\eta, \gamma$ and $\delta' = \frac{\delta}{C_1m^2\log m}$. Therefore, with probability at least $1 - 1/10 - \delta$ all the calls to Algorithm 2 satisfy the guarantee of a separation oracle for $\mathcal{L}_g$ with parameters $\eta$ and $\gamma$. By Theorem 5.3 and Equation (5), with probability at least $9/10 - \delta$ the minimization algorithm returns a point $x$ such that

$$\mathcal{L}_g(x) - \min_{y\in[0,1]^m}\{\mathcal{L}_g(y)\} \leq \eta + \alpha\left(\max_{y\in[0,1]^m}\{\mathcal{L}_g(y)\} - \min_{y\in[0,1]^m}\{\mathcal{L}_g(y)\}\right) \leq \frac{3\xi}{4} \,,$$

and with probability at least $9/10 - 2\delta$ the value $\nu$ satisfies

$$\nu \leq \min_{y\in[0,1]^m}\{\mathcal{L}_g(y)\} + \xi \,,$$

as desired.

By the above settings and by Lemma 5.3 we get that $\tau = \frac{\xi}{8m^{5/2}M}$ so the running time of each invocation of the separation oracle is

$$m \cdot \Phi_f\left(\frac{\tau^2}{2}, \frac{\delta'}{m}\right) = m \cdot \Phi_g\left(\frac{\xi^2}{128^5M^2}, \frac{\delta}{C_1m^2\log\frac{mM}{\xi}}\right) \,.$$

Since the evaluation of $\nu$ in the final step is negligible in the running time of the minimization algorithm, we get that the overall time complexity is

$$O\left(m^2 \cdot \Phi_g\left(\frac{\xi^2}{128m^5M^2}, \frac{\delta}{C_1 m^2 \log\frac{mM}{\xi}}\right)\log\frac{mM}{\xi} + m^3\log^{O(1)}\frac{mM}{\xi}\right) .$$

■

**Corollary 5.5.** *There exists an algorithm that, when given query access to a function $f\colon\{-1,1\}^n \to \{-1,1\}$ and a partition $\mathcal{I} = (I_1,\dots,I_m)$ of $[n]$ into $m$ sets, as well as input parameters $k \in \mathbb{N}, \epsilon, \xi \in (0,1]$, satisfies the following. It has time and query complexity $\tilde{O}\left(\max\left(\frac{m^{12}}{\xi^4}, \frac{m^{16}\epsilon^4}{k^4\xi^4}\right)\right)$, and distinguishes with probability at least $5/6$ between the two following cases:*

1. *There exists a set $S \subseteq [m]$ such that $|S| \geq m-k$ and $h(S) \leq \epsilon$.*
2. *For every set $S$ such that $|S| \geq m - 2(1+\frac{\xi}{\epsilon})k$, $h(S) > 2(\epsilon+\xi)$*

*where $h\colon 2^m \to \mathbb{R}$ is defined as $h(S) \stackrel{\text{def}}{=} \mathbf{Inf}_f(\cup_{i\in S} I_i)$.*

*(Moreover, the second item can be strengthened so that it holds for functions $f$ which satisfy the following: (i) for every set $S$ such that $|S| \geq m-k$, $h(S) > 2\epsilon + 2\xi$ and (ii) for every set $S$ such that $|S| \geq m-2(1+\frac{\xi}{\epsilon})k$, $h(S) > \epsilon + 2\xi$.)*

**Proof:** We apply Corollary 5.4 to $h'\colon 2^{[m]} \to \mathbb{R}$, defined as in Algorithm 1 by $h'(S) \stackrel{\text{def}}{=} h(S) - \frac{\epsilon}{k}|S|$, with $\xi$, $M \stackrel{\text{def}}{=} \max\left(2\max(2, \frac{\epsilon m}{k}), \xi/2\right) = 4\max(1, \frac{\epsilon m}{2k})$, and $\delta \stackrel{\text{def}}{=} \frac{1}{30}$. In order to do so, we need to simulate a $(\tau', \delta')$-noisy oracle for $h$. Since $h(S) = \mathbf{Inf}_f(\cup_{i\in S} I_i)$, in order to estimate $h'(S)$ to an additive $\tau'$ with probability at least $1-\delta'$, it is sufficient to estimate $\mathbf{Inf}_f(\cup_{i\in S} I_i) \in [0,2]$ to an additive $\tau'$ with probability at least $1-\delta'$ (indeed, the additional term $\frac{\epsilon}{k}|S|$ can be computed exactly). By Chernoff bounds, this can be done with $\Phi_h(\tau',\delta') = O(\frac{1}{\tau'^2}\log\frac{1}{\delta'})$ queries to $f$.

This yields an approximate oracle $\mathcal{O}_h^{\pm}$, and therefore $\mathcal{O}_{h'}^{\pm}$ (with success probability $9/10-2\delta = 5/6$) which can be provided to the algorithm of Theorem 4.1, with query complexity

$$O\left(m^2 \cdot \Phi_h\left(\frac{\xi^2}{m^5M^2}, \frac{1}{m^2\log\frac{mM}{\xi}}\right)\log\frac{mM}{\xi} + m^3\log^{O(1)}\frac{mM}{\xi}\right)$$

which, given the above expression for $\Phi_h$, can be computed as follows.

- If $\epsilon < \frac{2k}{m}$, so that $M = 4$, this simplifies as

$$O\left(\frac{m^{12}}{\xi^4}\log^2\frac{m}{\xi}\right).$$

- If $\epsilon \geq \frac{2k}{m}$, which implies that $M = \frac{2\epsilon m}{k}$, this becomes

$$O\left(\frac{m^{16}\epsilon^4}{k^4\xi^4}\log^2 m\right).$$

as claimed. Observing that $h$ is indeed a non-negative submodular function (and that $h'$ is submodular as the sum of a submodular and a modular functions) allows us to conclude by Theorem 4.1. ■
In particular, setting $\xi = \epsilon$ we get the following:

**Corollary 5.6.** *There exists an algorithm that, given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$, a fixed partition $\mathcal{I}$ of $[n]$ into $m = O(k^2)$ parts, and parameters $k \geq 1$ and $\epsilon \in (0,1)$, satisfies the following. The query complexity of the algorithm is $\tilde{O}\left(\frac{k^{24}}{\epsilon^4} + k^{28}\right) = \mathrm{poly}(k, \frac{1}{\epsilon})$, and:*

1. *if $f$ $\frac{\epsilon}{2}$-approximates being a $k$-part junta with respect to $\mathcal{I}$, then the algorithm accepts with probability at least $\frac{5}{6}$;*
2. *if $f$ $2\epsilon$-violates being a $4k$-part junta with respect to $\mathcal{I}$, then the algorithm rejects with probability at least $\frac{5}{6}$.*

*(Moreover, the second item can be strengthened to "simultaneously $2\epsilon$-violates being a $k$-part junta and $\frac{3}{2}\epsilon$-violates being a $4k$-part junta.")*

**Proof:** Follows immediately from applying Corollary 5.5 with $\xi = \epsilon$. ■

The tolerant junta testing theorem (Theorem 1.1) follows immediately from the above, together with Proposition 3.4. With probability at least 5/6, a random partition of the variables in $m \stackrel{\text{def}}{=} 96k^2$ parts will have the right guarantees, reducing the problem to distinguishing between $\frac{\epsilon}{2}$-approximating being a $k$-part junta vs. $2\epsilon$-violating being a $4k$-part junta (with regard to this random partition). Overall, the result is therefore correct with probability at least 2/3 by a union bound.

## 6 A tradeoff between tolerance and query complexity

In this section, we show how to obtain a smooth tradeoff between the amount of tolerance and the query complexity. Formally, we prove Theorem 1.2, restated below.

**Theorem 1.2.** *There exists an algorithm that, given query access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$ and parameters $k \geq 1$, $\epsilon \in (0,1)$ and $\rho \in (0,1/2)$, satisfies the following.*

- *If $f$ is $\rho\epsilon/8$-close to some $k$-junta, then the algorithm accepts with high constant probability.*
- *If $f$ is $\epsilon$-far from every $k$-junta, then the algorithm rejects with high constant probability.*

*The query complexity of the algorithm is $O\left(\frac{k\log k}{\epsilon\rho(1-\rho)^k}\right)$.*

Before delving into the proof of the theorem, we discuss some of its consequences. Setting $\rho = \Omega(1)$, we obtain a tolerant tester that distinguishes between functions $O(\epsilon)$-close to $\mathcal{J}_k$ and functions $\epsilon$-far from $\mathcal{J}_k$, with query complexity $2^{O(k)}/\epsilon$ – thus matching (and even improving) the simple tester described in Section 3. At the other end of the spectrum, setting $\rho = O(1/k)$ yields a weakly tolerant tester that distinguishes $O(\epsilon/k)$-close to $\mathcal{J}_k$ from $\epsilon$-far from $\mathcal{J}_k$, but with query complexity $\tilde{O}(k^2/\epsilon)$ – qualitatively matching the guarantees provided by the junta tester of [FKR+04].

### 6.1 Biased influence of a function

For a set $S \subseteq [s]$, we let $\phi(S) \stackrel{\text{def}}{=} \cup_{i\in S} I_i$ denote the set of coordinates in parts with indices in $S$. For a given string $x \in \{-1,1\}^n$ and a set $S$ we denote by $x_S \in \{-1,1\}^{|S|}$ which results by restricting $x$ to the set $S$.

**Definition 6.1.** For $x \in \{-1,1\}^n$, $\rho \in (0,1)$, and set $S \subseteq [n]$ we denote by $\mu_\rho(x_S)$ the distribution over all strings of length $|S|$ which results from flipping the value of each coordinate in $x_S$ with probability $\rho$. When $\rho = \frac{1}{2}$, we write $z \sim x_S$ for $z \sim \mu_{1/2}(x_S)$.

The main definition of this section is a generalization of the definition of the influence of a set (Definition 2.6), which corresponds to the case $\rho = \frac{1}{2}$:

**Definition 6.2.** For any $\rho \in (0,1)$ we define the $\rho$*-biased influence* of a set $S \subseteq [n]$ as follows.

$$\mathbf{Inf}^{\rho}_f(S) \stackrel{\text{def}}{=} 2 \Pr_{\substack{x \sim \{-1,1\}^n \\ z \sim \mu_\rho(x_S)}} [\, f(x) \neq f(x_{\bar{S}} \sqcup z) \,]$$

**Lemma 6.3.** *For every $\rho \in (0,1)$ and every fixed $S \subseteq [n]$,*

$$\mathbf{Inf}^{\rho}_f(S) = \sum_{U\colon U \cap S \neq \emptyset} \hat{f}(U)^2 \left(1 - (1-2\rho)^{|U \cap S|}\right).$$

**Proof:** Unrolling the definition of $\mathbf{Inf}^{\rho}_f(S)$, we have

$$\begin{aligned}
\frac{1}{2}\mathbf{Inf}^{\rho}_f(S) &= \Pr_{\substack{x \sim \{-1,1\}^n \\ z \sim \mu_\rho(x_S)}} [\, f(x) \neq f(x_{\bar{S}} \sqcup z) \,] = \mathop{\mathbf{E}}_{x,z}\left[\mathbb{1}_{\{f(x) \neq f(x_{\bar{S}} \sqcup z)\}}\right] = \mathop{\mathbf{E}}_{x,z}\left[\frac{1 - f(x)f(x_{\bar{S}} \sqcup z)}{2}\right] \\
&= \frac{1}{2} - \frac{1}{2}\mathop{\mathbf{E}}_{x,z}[f(x)f(x_{\bar{S}} \sqcup z)] = \frac{1}{2} - \frac{1}{2}\sum_{U,T \subseteq [n]} \hat{f}(T)\hat{f}(U) \mathop{\mathbf{E}}_{x,z}[\chi_T(x)\chi_U(x_{\bar{S}} \sqcup z)].
\end{aligned}$$

We analyze each $\mathop{\mathbf{E}}_{x,z}[\chi_T(x)\chi_U(x_{\bar{S}} \sqcup z)]$ separately.

$$\begin{aligned}
\mathop{\mathbf{E}}_{x,z}[\chi_T(x)\chi_U(x_{\bar{S}} \sqcup z)] &= \mathop{\mathbf{E}}_{x}\mathop{\mathbf{E}}_{z}[\chi_T(x)\chi_{U\cap\bar{S}}(x)\chi_{U\cap S}(z)] = \mathop{\mathbf{E}}_{x}\left[\chi_T(x)\chi_{U\cap\bar{S}}(x)\mathop{\mathbf{E}}_{z}[\chi_{U\cap S}(z)]\right] \\
&= \mathop{\mathbf{E}}_{x}\left[\chi_T(x)\chi_{U\cap\bar{S}}(x)\prod_{i \in U\cap S}\mathop{\mathbf{E}}_{z_i}[z_i]\right] = \mathop{\mathbf{E}}_{x}\left[\chi_T(x)\chi_{U\cap\bar{S}}(x)\prod_{i \in U\cap S}((1-\rho)x_i - \rho x_i)\right] \\
&= \mathop{\mathbf{E}}_{x}\left[\chi_T(x)\chi_{U\cap\bar{S}}(x)\prod_{i \in U\cap S}(1-2\rho)x_i\right] = (1-2\rho)^{|U\cap S|}\mathop{\mathbf{E}}_{x}[\chi_T(x)\chi_{U\cap\bar{S}}(x)\chi_{U\cap S}(x)] \\
&= (1-2\rho)^{|U\cap S|}\mathop{\mathbf{E}}_{x}[\chi_T(x)\chi_U(x)] = (1-2\rho)^{|U\cap S|}\mathop{\mathbf{E}}_{x}[\chi_{T\oplus U}(x)].
\end{aligned}$$

Note that if $U \neq T$, then $\mathop{\mathbf{E}}_{x}[\chi_{T\oplus U}(x)] = 0$ and if $U = T$, $\mathop{\mathbf{E}}_{x}[\chi_{T\oplus U}(x)] = 1$. Therefore,

$$\begin{aligned}
\frac{1}{2}\mathbf{Inf}^{\rho}_f(S) &= \frac{1}{2} - \frac{1}{2}\sum_{U \subseteq [n]}\hat{f}(U)^2(1-2\rho)^{|U\cap S|} = \frac{1}{2} - \frac{1}{2}\sum_{U\colon U\cap S \neq \emptyset}\hat{f}(U)^2(1-2\rho)^{|U\cap S|} - \frac{1}{2}\sum_{U\colon U\cap S = \emptyset}\hat{f}(U)^2 \\
&= \frac{1}{2}\sum_{U\colon U\cap S \neq \emptyset}\hat{f}(U)^2 - \frac{1}{2}\sum_{U\colon U\cap S \neq \emptyset}\hat{f}(U)^2(1-2\rho)^{|U\cap S|} \\
&= \frac{1}{2}\sum_{U\colon U\cap S \neq \emptyset}\hat{f}(U)^2\left(1 - (1-2\rho)^{|U\cap S|}\right)
\end{aligned}$$

proving the lemma. ■

The next statement is a direct consequence of Lemma 6.3 and Fact 2.7.

**Corollary 6.1.** *For every $\rho \in [0, 1/2]$ and $S \subseteq [n]$,*

$$2\rho \cdot \mathbf{Inf}_f(S) \leq \mathbf{Inf}^{\rho}_f(S) \leq \mathbf{Inf}_f(S).$$

Finally, the last result of this section shows that one can view the $\rho$-biased influence of a set as the expected (regular) influence of a random "$\rho$-biased set:"

**Claim 6.2.** *For any $J \subseteq [\ell]$ and any $\rho \in (0, \frac{1}{2}]$,*

$$\mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(\bar{J})) = \mathop{\mathbf{E}}_{S\sim\varrho_\rho([n])}\left[\mathbf{Inf}_f(\phi(S)) \mid S \cap J = \emptyset\right],$$

*where a set $S \subseteq [\ell]$ is drawn from $\varrho_\rho([\ell])$ by including independently each $i \in S$ with probability $\rho$.*

**Proof:** For any fixed $J \subseteq [\ell]$,

$$\begin{aligned}
\mathop{\mathbf{E}}_{S\sim\varrho_\rho([\ell])}\left[\mathbf{Inf}_f(\phi(S)) \mid S \cap J = \emptyset\right] &= \mathop{\mathbf{E}}_{S\sim\varrho_\rho([\ell])\ \mid\ S\cap J=\emptyset}\left[\mathbf{Inf}_f(\phi(S))\right] \\
&= \mathop{\mathbf{E}}_{S\sim\varrho_\rho([\ell])\ \mid\ S\cap J=\emptyset}\left[2 \Pr_{\substack{x\sim\{-1,1\}^n \\ z\sim x_{\phi(S)}}}\left[\, f(x) \neq f(x_{\phi(\bar{S})} \sqcup z) \mid S \,\right]\right] \\
&= 2\sum_{S\subseteq[\ell]} \Pr_{\substack{x\sim\{-1,1\}^n \\ z\sim x_{\phi(S)}}}\left[\, f(x) \neq f(x_{\phi(\bar{S})} \sqcup z) \mid S \,\right] \cdot \Pr_{S\sim\varrho_\rho([\ell])|S\cap J=\emptyset}[\, S\,] \\
&= 2\sum_{S\subseteq[\ell]} \Pr_{\substack{x\sim\{-1,1\}^n \\ S\sim\varrho_\rho([\ell])|S\subseteq\overline{J} \\ z\sim x_{\phi(S)}}}\left[\, f(x) \neq f(x_{\phi(\bar{S})} \sqcup z) \,\right] \\
&= 2 \Pr_{\substack{x\sim\{-1,1\}^n \\ z\sim\mu_{\frac{\rho}{2}}(x_{\phi(\bar{J})})}}\left[\, f(x) \neq f(x_{\phi(J)} \sqcup z) \,\right] \\
&= \mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(\bar{J}))\ .
\end{aligned}$$

Where the second last equality stems from the following observation: the processes of (i) picking a set $S \sim \varrho_\rho([\ell])$ conditioned on $S \subseteq \overline{J}$, and then setting $z$ according to $\mu_{\frac{1}{2}}(x_{\phi(S)})$; and (ii) setting $z$ according to $\mu_{\frac{\rho}{2}}(x_{\phi(\bar{J})})$, are equivalent. ■

## 6.2 Approximation of the biased influence

We now describe and analyze an algorithm that enables one to simultaneously get good estimates of all $\rho$-biased influences of any given family of subsets, while amortizing the number of queries over all these sets. This algorithm will be in Section 6.3 the main building block of the tolerant junta tester of Theorem 1.2.

**Lemma 6.4.** *Let $\mathcal{I} = (I_1, \dots, I_\ell) \subseteq [n]$ be any fixed collection of pairwise disjoint sets. For every $0 < \epsilon < 1$ and $0 < \rho \leq 1/2$, Algorithm 3 satisfies that with probability at least $1 - o(1)$ the following holds simultaneously, for all $T \in \mathcal{I}$ such that $|T| > \ell - k$:*

1. *if $\mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(T)) > \rho\epsilon$, then the estimate $\hat{\nu}_T$ of the $\frac{\rho}{2}$-biased influence of $\phi(T)$ is within a multiplicative factor of $(1 \pm \gamma)$ of its true value;*
2. *if $\mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(T)) < \frac{\rho\epsilon}{2}$, then the estimate $\hat{\nu}_T$ of the $\frac{\rho}{2}$-biased influence of $\phi(T)$ does not exceed $(1 + \gamma)\frac{\rho\epsilon}{2}$.*

**Algorithm 3** Approximate $\rho/2$-biased influence $(\mathcal{O}_f, \rho, \epsilon, \gamma, k, \ell, \mathcal{I})$

```
1: Set m = C·k log ℓ / (γ²ερ(1−ρ)^k), where C ≥ 1 is an absolute constant,     ▷ C ≥ 256 ln 2 is sufficient.
2: for j = 1 to m do
3:     Create a set S_j ⊆ [ℓ] by including each i ∈ [ℓ] in S_j with probability ρ.
4:     Pick x ∈ {−1,1}^n uniformly at random, and let z ∼ x_{φ(S̄_j)} .
5:     Set y ← x_{φ(S_j)} ⊔ z.
6:     Set ϑ_{S_j} ← 𝟙_{f(x)≠f(y)} .
7: end for
8: for every J ⊆ [ℓ] of size at most k do
9:     Let L_J ⊆ [m] denote the set of indices j such that S_j ⊆ J̄
10:    Let ν̂_J̄ ← (1/|L_J|) Σ_{j∈L_J} ϑ_{S_j}     ▷ Estimate of Inf_f^{ρ/2}(φ(J̄))
11: end for
```

**Proof of Lemma 6.4:** We first claim that for every $0 < \epsilon < 1$, $0 < \rho \le 1/2$, and any fixed set $J \subseteq [\ell]$ of size at most $k$, with probability at least $1 - o(\ell^{-2k})$ there are at least $m' = \frac{1}{2}(1-\rho)^k m = \frac{Ck\log \ell}{2\gamma^2\epsilon\rho}$ indices $i_1, \ldots, i_{m'}$ such that for every $j \in [m']$, $S_{i_j} \subseteq \bar{J}$.

To see why, fix some $J \subseteq [\ell]$ of size at most $k$. For every $i \in [m]$, let $\mathbb{1}_{\{S_i \subseteq \bar{J}\}}$ be the indicator value which is equal to 1 if, and only if, $S_i \subseteq \bar{J}$. Then, for every $i \in [m]$, $\Pr\left[\mathbb{1}_{\{S_i \subseteq \bar{J}\}} = 1\right] = (1-\rho)^{|J|} \ge (1-\rho)^k$. By a Chernoff bound,

$$\Pr\left[\frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\{S_i \subseteq \bar{J}\}} < \frac{1}{2}\cdot(1-\rho)^k\right] \le e^{-\frac{m}{8}(1-\rho)^k} = e^{-\frac{C}{8}\frac{k\log\ell}{\epsilon\rho\gamma^2}} < 2^{-4k\log\ell}$$

for a suitable choice of $C \ge 1$. Therefore, by a union bound over all $\sum_{z=0}^{k}\binom{\ell}{z} = 2^{(2+o(1)k\log\ell}$ such sets the claim follows.

From there, we are guaranteed that for every set $J \subseteq [\ell]$ of size at most $k$, with probability $1 - o(\ell^{-2k})$ there are at least $m' = \frac{Ck\log\ell}{2\gamma^2\epsilon\rho}$ indices $j_1, \ldots, j_{m'}$ such that for every $t \in \{j_1, \ldots, j_{m'}\}$ it holds that $S_{j_t} \subseteq \bar{J}$. We hereafter condition on this holding for all such sets, which by a union bound over $\sum_{z=0}^{k}\binom{\ell}{z} \le 2^{2k\log\ell}$ of them holds with probability at least $1 - o(1)$. By Claim 6.2 it is the case that for any $J \subseteq [\ell]$ and any $\rho \le 1/2$

$$\mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(\bar{J})) = \mathop{\mathbf{E}}_{S\sim\varrho_\rho([\ell])|S\cap J=\emptyset}[\mathbf{Inf}_f(\phi(S))] = \mathop{\mathbf{E}}_{S,x,y}\left[\mathbb{1}_{f(x)\neq f(y)}\right]$$

where $S, x, y$ are chosen according to steps 3-6 of the algorithm. Fixing any such set $\bar{J}$ such that $\left|\bar{J}\right| > \ell - k$ and $\mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(\bar{J})) > \rho\epsilon$, by a Chernoff bound we get (where $L \ge m'$ is as in Step 10)

$$\Pr\left[\left|\frac{1}{L}\sum_{i=1}^{L}\vartheta_S - \mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(\bar{J}))\right| > \gamma\mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(\bar{J}))\right] \le 2e^{-\frac{L\gamma^2\epsilon\rho}{3}} \le 2e^{-\frac{m'\gamma^2\epsilon\rho}{3}} = 2e^{-\frac{Ck\log\ell}{64}} < 2^{-4k\log\ell}.$$

again for a suitable choice of the constant $C \ge 1$. By taking again a union bound we get that with probability at least $1 - o(1)$ all estimates of the $\frac{\rho}{2}$-biased influence of the sets $T$ such that $|T| > \ell - k$ and $\mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(T)) > \rho\epsilon$ are within a multiplicative factor $1 \pm \gamma$ of their true values.

In addition, for any set $J \subseteq [\ell]$ with $|J| \leq k$ such that $\mathbf{Inf}_f(\phi(\bar{J})) \leq \frac{\rho\epsilon}{2}$, similarly by a multiplicative Chernoff bound:

$$\Pr\left[\frac{1}{L}\sum_{i=1}^{L}\vartheta_S > (1+\gamma)\frac{\rho\epsilon}{2}\right] \leq e^{-\frac{\gamma^2}{3}\frac{\rho\epsilon}{2}L} \leq e^{-\frac{\gamma^2\rho\epsilon}{6}m'} = 2e^{-\frac{Ck}{12}\log\ell} < 2^{-4k\log\ell}.$$

as before, and we conclude again by a union bound over all subsets of size at most $k$. Overall, the conclusions above hold with probability at least $1-o(1)$, as claimed. ■

### 6.3 Tradeoff between tolerance and query complexity

We now describe how the algorithm from the previous section lets us easily derive the tolerant tester of Theorem 1.2.

**Algorithm 4** $\rho$-Tolerant Junta Tester $(\mathcal{O}_f, \epsilon, \rho, \epsilon, k)$

1: Create a random partition $\mathcal{I}$ of $\ell = 24k^2$ parts by uniformly and independently assigning each coordinate to a part.
2: Run Algorithm 3 with the partition $\mathcal{I}$, $\ell = 24k^2$ and $\gamma = 1/4$.
3: **if** there is a set $J \subseteq [\ell]$ of size at most $k$ such that $\hat{\nu}_{\bar{J}} \leq \frac{3\rho\epsilon}{4}$ **then**
4:     **return** accept.
5: **end if**
6: **return** reject.

**Proof of Theorem 1.2:** Given Proposition 3.4 it is sufficient to consider a partition $\mathcal{I}$ of size $\ell = 24k^2$ and show that Algorithm 4 distinguishes with probability at least $5/6$ between the following two cases.

1. $f$ $\frac{\rho\epsilon}{4}$-approximates being a $k$-part junta with respect to $\mathcal{I}$;
2. $f$ $\frac{\epsilon}{2}$-violates being a $k$-part junta with respect to $\mathcal{I}$

Suppose first there exists a set $J \subseteq [\ell]$ with $|J| \leq k$ such that $\mathbf{Inf}_f(\phi(\bar{J})) \leq \frac{\rho\epsilon}{2}$. By Corollary 6.1, $\mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(\bar{J})) \leq \frac{\rho\epsilon}{2}$ and thus, from Lemma 6.4, we know that with probability at least $1-o(1)$ our estimate $\hat{\nu}_{\bar{J}}$ at at most $(1+1/4)\frac{\epsilon\rho}{2} \leq \frac{3\epsilon\rho}{4}$. Therefore, Algorithm 4 will return accept when considering $J$.

Consider now the case where every set $J \subseteq [\ell]$ with $|J| \leq k$ is such that $\mathbf{Inf}_f(\phi(\bar{J})) > \epsilon$. Then, by Corollary 6.1 we have that $\mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(\bar{J})) \geq \rho\mathbf{Inf}_f(\phi(\bar{J})) > \rho\epsilon$. The approximation guarantee then applies: for every such $J \subseteq [\ell]$,

$$\hat{\nu}_{\bar{J}} \geq \frac{3}{4}\mathbf{Inf}_f^{\frac{\rho}{2}}(\phi(\bar{J})) \geq \frac{3}{4}\rho\mathbf{Inf}_f(\phi(\bar{J})) > \frac{3\rho\epsilon}{4}\ .$$

Thus, with probability at least $1-o(1)$, Algorithm 4 will reject $f$. ■

## 7 "Instance-by-instance" tolerant isomorphism testing

In this section, we show how the machinery developed in Section 6, and more precisely the algorithm from Theorem 1.2, can be leveraged to obtain *instance-by-instance tolerant isomorphism testing* between two unknown Boolean functions $f$ and $g$, as defined below.

We begin with some notation: for $f, g\colon \{-1,1\}^n \to \{-1,1\}$, we denote by $\operatorname{distiso}(f,g)$ the distance between $f$ and the closest isomorphism of $g$, that is $\operatorname{distiso}(f,g) \stackrel{\text{def}}{=} \min_{\pi\in\mathcal{S}_n} \operatorname{dist}(f, g\circ\pi)$. Given query access to two unknown Boolean functions $f, g\colon \{-1,1\}^n \to \{-1,1\}$ and a parameter $\epsilon \in (0,1]$, isomorphism testing then amounts to distinguishing between (i) $\operatorname{distiso}(f,g) = 0$; and (ii) $\operatorname{distiso}(f,g) > \epsilon$.[5]

Our result will be parameterized in terms of the *junta degree* of the unknown functions $f$ and $g$, formally defined below:

**Definition 7.1** (Junta degree)**.** Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a Boolean function, and $\gamma \in [0,1]$ a parameter. We define the *$\gamma$-junta degree of $f$* as the smallest integer $k$ such that $f$ is $\gamma$-close to being a $k$-junta, that is

$$k^*(f,\gamma) \stackrel{\text{def}}{=} \min\left\{\, k \in [n] : \ \operatorname{dist}(f, \mathcal{J}_k) \le \gamma \,\right\}.$$

Finally, we extend this definition to two functions $f, g$ by setting $k^*(f,g,\gamma) = \min(k^*(f,\gamma), k^*(g,\gamma))$.

With this terminology in hand, we can restate Theorem 1.3:

**Theorem 7.2** (Theorem 1.3, rephrased)**.** *There exist absolute constants $c \in (0,1)$, $\epsilon_0 \in (0,1]$ and a tolerant testing algorithm for isomorphism of two unknown functions $f$ and $g$ with the following guarantees. On inputs $\epsilon \in (0,\epsilon_0]$, $\delta \in (0,1]$, and query access to functions $f, g\colon \{-1,1\}^n \to \{-1,1\}$:*

- *if* $\operatorname{distiso}(f,g) \le c\epsilon$*, then it outputs* accept *with probability at least $1-\delta$;*
- *if* $\operatorname{distiso}(f,g) > \epsilon$*, then it outputs* reject *with probability at least $1-\delta$.*

*The query complexity of the algorithm satisfies the following, where $k^* = k^*(f,g,\frac{\rho c\epsilon}{8})$ is the $c\epsilon$-junta degree of $f$ and $g$:*

- *it is $\tilde{O}(2^{\frac{k^*}{2}}\log\frac{1}{\delta})$ with probability at least $1-\delta$;*
- *it is always at most $\tilde{O}(2^{\frac{n}{2}}\log\frac{1}{\delta})$.*

*Moreover, one can take $c = \frac{1}{2048}$, and $\epsilon_0 \stackrel{\text{def}}{=} \frac{4}{5}(5-2\sqrt{6}) \simeq 0.08$.*

## 7.1 Proof of Theorem 7.2

As described in Section 1.2, our algorithm first performs a linear search on $k$, invoking at each step the tolerant tester of Section 6 with parameter $\epsilon'$, to obtain (with high probability) a value $k^*$ such that $k^*(f,g,\epsilon') \le k^* \le k^*(f,g,\frac{\rho\epsilon'}{8})$. In the second stage, it calls a "noisy sampler" (defined below) to obtain uniformly random labeled samples from the "cores" of the $k^*$-juntas closest to $f$ and $g$, and robustly tests isomorphism between them. We accordingly divide this section in two, proving respectively these two statements:

**Lemma 7.3.** *There exists an algorithm (Algorithm 5) with the following guarantees. On inputs $\epsilon, \delta \in (0,1]$ and query access to $f, g\colon \{-1,1\}^n \to \{-1,1\}$, it returns a value $0 \le k \le n$, such that:*

- *with probability at least $1-\delta$, we have that:*
  - *(i) $k^*(f,g,\epsilon') \le k^* \le k^*(f,g,\frac{\rho\epsilon'}{8})$;*

[5] Phrased differently, this is testing the property $\mathcal{P} = \left\{ (f, f\circ\pi) : \ f \in 2^{2^n}, \pi \in \mathcal{S}_n \right\} \subseteq 2^{2^n} \times 2^{2^n}$.

*(ii) the algorithm performs $O\Big(2^{\frac{k}{2}+o(k)} \cdot \frac{1}{\epsilon}\log\frac{1}{\delta}\Big)$ queries;*

- *the algorithm performs at most $O\Big(2^{\frac{n}{2}+o(n)} \cdot \frac{1}{\epsilon}\log\frac{1}{\delta}\Big)$ queries.*

**Proposition 7.4.** *There exists an algorithm (Algorithm 6) with query complexity $\tilde{O}\Big(\frac{2^{k/2}}{\epsilon}\Big)$ for testing of isomorphism of two unknown functions $f$ and $g$, under the premise that $f$ is close to $\mathcal{J}_k$. More precisely, there exist absolute constants $c > 0$ and $\epsilon_0 \in (0,1]$ such that, on inputs $k \in \mathbb{N}$, $\epsilon \in (0, \epsilon_0]$ and query access to functions $f, g\colon \{-1,1\}^n \to \{-1,1\}$, the algorithm has the following guarantees. Conditioned on $\operatorname{dist}(f, \mathcal{J}_k) \le c\epsilon$, it holds that:*

- *if $\operatorname{distiso}(f,g) \le c\epsilon$, then it outputs* accept *with probability at least $8/15$;*
- *if $\operatorname{distiso}(f,g) > \epsilon$, then it outputs* reject *with probability at least $8/15$.*

*Moreover, one can take $c = \frac{1}{2048}$, and $\epsilon_0 \stackrel{\text{def}}{=} \frac{4}{5}(5-2\sqrt{6}) \simeq 0.08$.*

Theorem 7.2 follows by the combination of Lemma 7.3 and Proposition 7.4.

**Proof of Theorem 7.2:** Let $\rho \stackrel{\text{def}}{=} 1 - \frac{1}{\sqrt{2}}$, and $\epsilon' = c\epsilon$. The algorithm proceeds as follows: it first invokes Algorithm 5 with inputs $f, g, \epsilon', \delta/2$, and gets by Lemma 7.3, a value $1 \le k^* \le n$ such that $k^*(f,g,\epsilon') \le k^* \le k^*(f,g,\frac{\rho}{8}\epsilon')$ with probability at least $1-\frac{\delta}{2}$. In particular, conditioning on this we are guaranteed that either $f$ or $g$ is $\epsilon'$-close to some $k^*$-junta (i.e., by our choice of $c$, one of the functions is $c\epsilon$-close to $\mathcal{J}_{k^*}$). It then calls Algorithm 6 with inputs $f, g, k^*, \epsilon$ independently $O(\log\frac{1}{\delta})$ times (for probability amplification from $8/15$ to $1-\frac{\delta}{2}$), and accepts if and only if the majority of these executions returned accept. The correctness of the algorithm follows from Proposition 7.4 and the bound on the query complexity follows from the bounds in Lemma 7.3 and Proposition 7.4. ■

### 7.1.1 Linear search: finding $k^*$.

Let $\mathcal{T}$ denote the algorithm of Theorem 1.2, with probability of success amplified by standard techniques to $1-\delta$ for any $\delta \in (0,1]$ (at the price of a factor $O\Big(\log\frac{1}{\delta}\Big)$ in its query complexity); and write $q_{\mathcal{T}}(k,\epsilon,\rho,\delta) = O\Big(\frac{k\log k}{\epsilon\rho(1-\rho)^k}\log\frac{1}{\delta}\Big)$ for its query complexity. Algorithm 5, given next, performs the linear search for $k^*$: we then analyze its correctness and query complexity.

**Algorithm 5** Junta Degree Finder($\mathcal{O}_f, \mathcal{O}_g, \epsilon', \delta$)

```
Set ρ ← 1 − 1/√2 and let T be the algorithm of Theorem 1.2.
for k = 0 to n do
    Call T on f with parameters k, ε', ρ, and 3δ/(2π²(k+1)²).
    Call T on g with parameters k, ε', ρ, and 3δ/(2π²(k+1)²).
    if either call to T returned accept then return k.
    end if
end for
return n
```

**Proof of Lemma 7.3:** By a union bound, all executions of $\mathcal{T}$ will be correct with probability at least $1 - 2\sum_{j=1}^{\infty}\frac{3\delta}{2\pi^2 j^2} = 1-\frac{\delta}{2}$. Conditioning on this, the tester will accept for some $k$ between $k^*(f,g,\epsilon')$ and $k^*(f,g,\rho\epsilon'/8)$. This is true since as long as we invoke $\mathcal{T}$ with values $k$ such that $f$

and $g$ are $\epsilon'$-far from $\mathcal{J}_k$, both invocations of $\mathcal{T}$ will reject. Therefore, once we accept, we have that either $f$ or $g$ is at least $\epsilon'$-close to $\mathcal{J}_k$. Hence, $k \geq k^*(f, g, \epsilon')$. Also, $\mathcal{T}$ is guaranteed to accept on some $k'$ whenever invoked on a function that is $\rho\epsilon'/8$-close to $\mathcal{J}_{k'}$. By definition, $k^*(f, g, \rho\epsilon'/8)$ is such a $k'$ for either $f$ or $g$; hence, $k \leq k^*(f, g, \rho\epsilon'/8)$.

In the case that all executions of $\mathcal{T}$ returned correctly, the query complexity is

$$q(\epsilon, f, g) = \sum_{k=0}^{k^*(f,g,\epsilon'')} 2q_{\mathcal{T}}\left(k, \epsilon', \rho, \frac{3\delta}{2\pi^2(k+1)^2}\right).$$

By the expression of $q_{\mathcal{T}}$, we get that $q(\epsilon, f, g)$ is upper bounded by

$$q(\epsilon, f, g) \leq \frac{O(1)}{\epsilon \cdot \rho} \sum_{k=1}^{k^\star} \frac{k \log k \log \frac{k}{\delta}}{(1-\rho)^k} \leq \frac{O(1)}{\epsilon} (k^\star \log k^\star)^2 2^{k^\star \log \frac{1}{1-\rho}} \log \frac{1}{\delta}$$

where $k^\star \stackrel{\text{def}}{=} k^*(f, g, \frac{\rho^2\epsilon}{64})$. In particular, from the choice of $\rho$, we get $q(\epsilon, f, g) \leq 2^{\frac{k^\star}{2} + o(k^*)} O\left(\frac{1}{\epsilon} \log \frac{1}{\delta}\right)$.

(If not all executions of the tester are successful, in the worst case the algorithm considers all possible values of $k$, before finally returning $n$. In this case, the query complexity is similarly bounded by $2^{\frac{n}{2} + o(n)} O\left(\frac{1}{\epsilon} \log \frac{1}{\delta}\right)$.)

■

#### 7.1.2 Noisy samplers and core juntas.

For a Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$ we denote by $f_k : \{-1,1\}^n \to \{-1,1\}$ the $k$-junta closest to $f$. That is, the function $h \in \mathcal{J}_k$ such that $\operatorname{dist}(f, h) = \operatorname{dist}(f, \mathcal{J}_k)$ (if this function is not unique, then we define $f_k$ to be the first according to lexicographic order). Moreover, following Chakraborty et al. [CGM11], for a $k$-junta $f \in \mathcal{J}_k$ (where we assume without loss of generality that $f$ depends on exactly $k$ variables) we define the *core* of $f$, as follows. The core of $f$, denoted $\mathsf{core}_f\colon \{-1,1\}^k \to \{-1,1\}$, is the restriction of $f$ to its relevant variables (where these variables are numbered according to the natural order); so that for some $i_1 \leq \cdots \leq i_k \in [n]$ we have

$$f(x) = \mathsf{core}_f(x_{i_1}, \ldots, x_{i_k})$$

for every $x \in \{-1,1\}^n$.

**Definition 7.5** ([CGM11, Definition 1])**.** Let $g\colon \{-1,1\}^k \to \{-1,1\}$ be a function and let $\eta, \mu \in [0,1)$. An $(\eta,\mu)$*-noisy sampler for $g$* is a probabilistic algorithm $\tilde{g}$ that on each execution outputs a pair $(x, a) \in \{-1,1\}^k \times \{-1,1\}$ such that

(i) For all $y \in \{-1,1\}^k$, $\Pr[\, x = y \,] \in \left[\frac{1-\mu}{2^k}, \frac{1+\mu}{2^k}\right]$;

(ii) $\Pr[\, a = g(x) \,] \geq 1 - \eta$;

(iii) the pairs output on different executions are mutually independent.

An $\eta$*-noisy sampler* is an $(\eta, 0)$-noisy sampler, i.e., one that on each execution selects a uniformly random $x \in \{-1,1\}^k$.

Chakraborty et al. [CGM11] show how to build an efficient $O(\epsilon)$-noisy sampler for $\mathsf{core}_{f_k}$, which is guaranteed to apply as long as $\mathrm{dist}(f, \mathcal{J}_k) = O(\epsilon^6/k^{10})$. In more detail, they first run a modified version of the junta tester from [Bla09], which, whenever it accepts, also returns some preprocessing information that enables one to build such a noisy sampler. Moreover, they show that this tester will indeed accept any function that is $O(\epsilon^6/k^{10})$-close to $k$-junta (in addition to rejecting those $\epsilon$-far from it), giving the above guarantee. Using instead (a small modification of) our tolerant tester from Section 6, we are able to extend their techniques to obtain the following – less efficient, but more robust – noisy sampler.

**Proposition 7.6** (Noisy sampler for close-to-junta functions)**.** *There are algorithms $\mathcal{A}_P, \mathcal{A}_S$ (respectively preprocessor and sampler), which both require oracle access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$, and satisfy the following properties.*

- *The preprocessor $\mathcal{A}_P$ takes $\epsilon' \in (0,1]$, $\rho \in (0,1/2]$, $k \in \mathbb{N}$ as inputs, makes $O\left(\frac{k \log \frac{k}{\epsilon'}}{\epsilon' \rho (1-\rho)^k}\right)$ queries to $f$, and either returns $\mathsf{fail}$ or a state $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$. The sampler $\mathcal{A}_S$ takes as input such a state $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$, makes a single query to $f$, and outputs a pair $(x,a) \in \{-1,1\}^k \times \{-1,1\}$. We say that a state $\sigma$ is $\gamma$-*good *if for some permutation $\pi \in \mathcal{S}_k$, $\mathcal{A}_S(\sigma)$ is a $\gamma$-noisy sampler for $\mathsf{core}_{f_k} \circ \pi$.*
- *$\mathcal{A}_P(\epsilon', \rho, k)$ fulfills the following conditions:*
  - *(i) If $\mathrm{dist}(f, \mathcal{J}_k) \leq \frac{\rho}{8}\epsilon'$, then with probability at least $4/5$, $\mathcal{A}_P$ returns a state $\sigma$ that is $8\epsilon'$-good.*
  - *(ii) If $\mathrm{dist}(f, \mathcal{J}_k) > \epsilon'$, then with probability at least $4/5$, $\mathcal{A}_P$ returns $\mathsf{fail}$.*
  - *(iii) If $\mathrm{dist}(f, \mathcal{J}_k) \leq \epsilon'$, then with probability at least $4/5$, $\mathcal{A}_p$ either returns $\mathsf{fail}$ or returns a state $\sigma$ that is $8\epsilon'$-good.*

The proof of Proposition 7.6 is deferred to Appendix A; indeed, it is almost identical to the proof of Proposition 4.16 in [CGM11], with small adaptations required to comply with the use of the tolerant tester from Section 6 instead of the tester from [AB10].

We note that the main difference between the guarantees of our noisy sampler and those of the noisy sampler in [CGM11, Lemma 2] lies in the set of functions for which the noisy sampler is required to return a good state. In our case, this set consists of functions that are *somewhat* close to $k$-juntas. In comparison, the construction from [CGM11] is more query-efficient (only $\tilde{O}(k/\epsilon)$ queries to $f$ in the preprocessing stage), but only guarantees the output of a noisy sampler for functions $f$ that are $O(\epsilon^6/k^{10})$-close to $\mathcal{J}_k$.

With these primitives in hand, we are almost ready to prove the main proposition of this subsection, Proposition 7.4. To state the algorithm (Algorithm 6) and proceed with its analysis, we will require the following definition:

**Definition 7.7** (Number of violating pairs $V_\pi$)**.** Given two sets $Q_1, Q_2 \subseteq \{-1,1\}^k \times \{-1,1\}$ and a permutation $\pi \in \mathcal{S}_k$ we say that pairs $(x, a_1) \in Q_1$ and $(y, a_2) \in Q_2$ are *violating with respect to* $\pi$, if $y = \pi(x)$ and $a_1 \neq a_2$. We denote the number of violating pairs with respect to $\pi$ by $V_\pi$.

**Algorithm 6** Tolerant isomorphism testing to an unknown $f$ such that $\mathrm{dist}(f, \mathcal{J}_k) \leq c\epsilon$ $(\mathcal{O}_f, \mathcal{O}_g, \epsilon, k)$

1: Let $\mathcal{A}_P, \mathcal{A}_S$ be as in Proposition 7.6, $\rho \leftarrow 1 - \frac{1}{\sqrt{2}}$, $\epsilon' \leftarrow \frac{\epsilon}{32}$, $\alpha \leftarrow 4c\epsilon$.

2: $s \leftarrow C\frac{2^{k/2}}{\epsilon}\sqrt{k \ln k}$, $t \leftarrow (3\alpha + 24\epsilon')\frac{s^2}{2^k}$ ▷ $C > 1$ is an absolute constant.

3: Run the preprocessor $\mathcal{A}_P$ on $f$ and $g$ with parameters $\epsilon' \stackrel{\text{def}}{=} \frac{\epsilon}{32}$, $\rho$, $k$. ▷ Failure probability $\frac{2}{5}$.

4: **if** either invocation of $\mathcal{A}_P$ returned fail **then**

5: **return** reject

6: **end if**

7: Using the $8\epsilon'$-noisy sampler $\mathcal{A}_S$ (called with the states returned on Step 3), construct "core" sets $Q_f, Q_g \subseteq \{-1,1\}^k \times \{-1,1\}$ each of size $s \leftarrow C\frac{2^{k/2}}{\epsilon}\sqrt{k \ln k}$.

8: **if** there exist $\pi \in \mathcal{S}_k$ such that $V_\pi \leq t$ **then**

9: **return** accept

10: **end if**

11: **return** reject

**Proof of Proposition 7.4:** The query complexity is the sum of the query complexities from Steps 3 and 7, i.e.,

$$O\left(\frac{k \log \frac{k}{\epsilon}}{\epsilon\rho(1-\rho)^k}\right) + 2s \cdot 1 = O\left(\frac{2^{k/2}}{\epsilon}k \log \frac{k}{\epsilon} + \frac{2^{k/2}}{\epsilon}\sqrt{k \ln k}\right) = O\left(\frac{2^{k/2}}{\epsilon}k \log \frac{k}{\epsilon}\right).$$

**Completeness.** Assume that $g$ is $c\epsilon$-close to isomorphic to $f$, which itself is $c\epsilon$-close to being a $k$-junta. Therefore, by the triangle inequality and by our choice of $c \leq \frac{1}{2048}$, $\mathrm{dist}(g, \mathcal{J}_k) \leq 2c\epsilon \leq \rho\epsilon'/8$ as well, so that with probability at least $3/5$ the algorithm does not output reject on Step 5 (we thereafter analyze this case). Moreover, by the triangle inequality there exists a permutation $\pi \in \mathcal{S}_n$ such that $\mathrm{dist}(f_k, g_k \circ \pi) \leq 2c\epsilon + 2c\epsilon = 4c\epsilon \stackrel{\text{def}}{=} \alpha$. In particular, this implies that there exists a permutation $\pi^* \in \mathcal{S}_k$ such that $\mathrm{dist}(\mathsf{core}_{f_k}, \mathsf{core}_{g_k} \circ \pi^*) \leq \alpha$. Let $T^* \subseteq \{-1,1\}^k$ be the disagreement set between $\mathsf{core}_{f_k}$ and $\mathsf{core}_{g_k} \circ \pi^*$: by the above $|T^*| \leq \alpha 2^k$.

Let $Q_f^s, Q_g^s \subseteq \{-1,1\}^k$ denote the sets resulting from taking the first element in each pair in $Q_f$ and $Q_g$ respectively. The size of the intersection $Z \stackrel{\text{def}}{=} \left|Q_f^s \cap T^*\right|$ is distributed as a hypergeometric random variable, namely $Z \sim \mathrm{HyperGeom}(s, 2^k, |T^*|)$, and conditioned on $Z$ we have $Z^* \stackrel{\text{def}}{=} \left|Q_f^s \cap Q_g^s \cap T^*\right| \sim \mathrm{HyperGeom}(s, 2^k, Z)$. In particular, we get

$$\mathbf{E}[Z] = \frac{s\,|T^*|}{2^k}, \quad \mathbf{E}[\,Z^* \mid Z\,] = \mathbf{E}\left[\,\left|Q_f^s \cap Q_g^s \cap T^*\right| \;\middle|\; |Q_f^s \cap T^*|\,\right] = \frac{sZ}{2^k}\ .$$

Let $\mathcal{A}_S^f$ denote the noisy sampler algorithm when invoked for $f$, and for every $x \in Q_f^s$ let $\mathcal{A}_S^f(x)$ denote the label given to $x$ by $\mathcal{A}_S^f$. Since $\mathcal{A}_S^f$ is an $8\epsilon'$-noisy sampler for $\mathsf{core}_{f_k}$, $\Pr[\mathcal{A}_S^f(x) \neq \mathsf{core}_{f_k}(x)] \leq 8\epsilon'$. An analogous statement holds for $g$. We let $N \stackrel{\text{def}}{=} |\{x \in Q_f^s \cap Q_g^s \ :\ \mathcal{A}_S^f(x) \neq \mathsf{core}_{f_k}(x) \text{ or } \mathcal{A}_S^g(x) \neq \mathsf{core}_{g_k}(x)\}|$ be the number of common samples incorrectly labelled by either noisy sampler, and observe that $N$ is dominated by a Binomial random variable $\tilde{N} \sim \mathrm{Bin}\left(\left|Q_f^s \cap Q_g^s\right|, 16\epsilon'\right)$.

With this in hand, we can bound $\Pr[\, V_{\pi^*} > t\,]$ as follows (recall that $t = 3\alpha + 24\epsilon'$):

$$\begin{aligned}\Pr\left[\, V_{\pi^*} > (3\alpha + 24\epsilon')\frac{s^2}{2^k} \,\right] &\le \Pr\left[\, \left|Q_f^s \cap Q_g^s \cap T^*\right| > 3\alpha\frac{s^2}{2^k} \,\right] + \Pr\left[\, N > 24\epsilon'\frac{s^2}{2^k} \,\right] \\ &\le \Pr\left[\, \left|Q_f^s \cap Q_g^s \cap T^*\right| > 3\alpha\frac{s^2}{2^k} \,\right] + \Pr\left[\, \tilde{N} > 24\epsilon'\frac{s^2}{2^k} \,\right].\end{aligned}$$

Recall that $Z^* = \left|Q_f^s \cap Q_g^s \cap T^*\right|$. Since $\Pr\left[\, \left|Q_f^s \cap Q_g^s \cap T^*\right| > 3\alpha\frac{s^2}{2^k} \,\right]$ is maximized when $|T^*|$ is maximal, we assume without loss of generality that $|T^*| = \alpha 2^k$. We will handle each term separately.

$$\begin{aligned}\Pr\left[\, Z^* > \frac{3}{2}\cdot\alpha\frac{s^2}{2^k} \,\right] &= \Pr\left[\, Z^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \,\right] \\ &= \Pr\left[\, Z^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \;\middle|\; Z > \frac{5}{4}\frac{s\,|T^*|}{2^k} \,\right]\cdot \Pr\left[\, Z > \frac{5}{4}\frac{s\,|T^*|}{2^k} \,\right] \\ &\quad + \Pr\left[\, Z^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \;\middle|\; Z \le \frac{5}{4}\frac{s\,|T^*|}{2^k} \,\right]\cdot \Pr\left[\, Z \le \frac{5}{4}\frac{s\,|T^*|}{2^k} \,\right] \\ &\le \Pr\left[\, Z > \frac{5}{4}\frac{s\,|T^*|}{2^k} \,\right] + \Pr\left[\, Z^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \;\middle|\; Z \le \frac{5}{4}\frac{s\,|T^*|}{2^k} \,\right].\end{aligned}$$

We again bound the two terms separately. By the assumption that $|T^*| = \alpha 2^k$ and by the choice of $s$,[6]

$$\Pr\left[\, Z > \frac{5}{4}\frac{s\,|T^*|}{2^k} \,\right] = \Pr\left[\, Z > \frac{5}{4}\mathbf{E}[Z] \,\right] < \exp\left(-\frac{1}{2}\cdot\left(\frac{1}{4}\right)^2\cdot\frac{s\,|T^*|}{2^k}\right) < \frac{1}{30}.$$

As for the second term, since $\mathbf{E}[Z^*] = \frac{sZ}{2^k}$ and by the assumption on $T^*$ and the setting of $s$,

$$\begin{aligned}\Pr\left[\, Z^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \;\middle|\; Z < \frac{5}{4}\frac{s\,|T^*|}{2^k} \,\right] &\le \Pr\left[\, Z^* > \frac{3}{2}\frac{s^2\,|T^*|}{2^{2k}} \;\middle|\; Z = \frac{5}{4}\frac{s\,|T^*|}{2^k} \,\right] \\ &= \Pr\left[\, Z^* > \frac{6}{5}\mathbf{E}[Z^*] \;\middle|\; Z = \frac{5}{4}\frac{s\,|T^*|}{2^k} \,\right] \\ &< \exp\left(-\frac{1}{2}\cdot\left(\frac{1}{5}\right)^2\cdot\frac{s}{2^k}\frac{s\,|T^*|}{2^k}\right) < \frac{1}{30}\end{aligned}$$

for a sufficiently large constant $C$ in the definition of $s$.

[6] Where we use the fact that hypergeometric random variables enjoy the same concentration bounds as Binomial random variables – in particular, Chernoff bounds apply (see e.g. [AD11, Theorem 1.17]).

As for the second term, since $\mathbf{E}\Big[\tilde{N}\Big] = 16\epsilon' \left|Q_f^s \cap Q_g^s\right|$ we have,

$$\begin{aligned}
\Pr\left[\tilde{N} > 24\epsilon' \frac{s^2}{2^k}\right] &\leq \Pr\left[\tilde{N} > 24\epsilon' \frac{s^2}{2^k} \;\middle|\; \left|Q_f^s \cap Q_g^s\right| \leq \frac{5}{4}\frac{s^2}{2^k}\right] \cdot \Pr\left[\left|Q_f^s \cap Q_g^s\right| \leq \frac{5}{4}\frac{s^2}{2^k}\right] + \Pr\left[\left|Q_f^s \cap Q_g^s\right| > \frac{5}{4}\frac{s^2}{2^k}\right] \\
&\leq \Pr\left[\tilde{N} > 24\epsilon' \frac{s^2}{2^k} \;\middle|\; \left|Q_f^s \cap Q_g^s\right| = \frac{5}{4}\frac{s^2}{2^k}\right] + \Pr\left[\left|Q_f^s \cap Q_g^s\right| > \frac{5}{4}\frac{s^2}{2^k}\right] \\
&\leq \Pr\left[\tilde{N} > \frac{6}{5}\mathbf{E}\Big[\tilde{N}\Big] \;\middle|\; \left|Q_f^s \cap Q_g^s\right| = \frac{5}{4}\frac{s^2}{2^k}\right] + \Pr\left[\left|Q_f^s \cap Q_g^s\right| > \frac{5}{4}\frac{s^2}{2^k}\right] \\
&< \exp\left(-\frac{1}{2}\cdot\left(\frac{1}{6}\right)^2 \cdot \frac{16\epsilon' \cdot 5s^2}{4\cdot 2^k}\right) + \exp\left(-\frac{1}{2}\cdot\left(\frac{1}{4}\right)^2 \cdot \frac{s^2}{2^k}\right) \leq \frac{1}{15}.
\end{aligned}$$

(Actually $o(1)$.)

The algorithm will therefore reject with probability at most $\frac{2}{5} + \frac{1}{15} + \frac{1}{15} = \frac{7}{15}$.

**Soundness.** Assume that $\operatorname{dist}(f, \mathcal{J}_k) \leq c\epsilon$, and that $g$ is $\epsilon$-far from being isomorphic to $f$. Then one of the following must hold:

1. $\operatorname{dist}(g, \mathcal{J}_k) > \epsilon'$.
2. for all $\pi \in \mathcal{S}_k$, $\operatorname{dist}(\mathsf{core}_{f_k}, \mathsf{core}_{g_k} \circ\pi) > \epsilon - (\epsilon' + c\epsilon) > \epsilon - 2\epsilon'$.

If the first case holds, then the function will be rejected in Step 3 with probability at least $\frac{4}{5}$, and so the algorithm will reject as desired. We can therefore focus on the second case.

If the second case holds, either the tester rejects in Step 5 (and we are done) or it outputs a state which will be used to get the $8\epsilon'$-noisy sampler. Fix any $\pi \in \mathcal{S}_k$. Since $\operatorname{dist}(\mathsf{core}_{f_k}, \mathsf{core}_{g_k} \circ\pi) > (\epsilon - 2\epsilon')$, there are $m \stackrel{\text{def}}{=} m(\pi) \geq (\epsilon - 2\epsilon')2^k$ inputs $x \in \{-1,1\}^k$ such that $\mathsf{core}_{f_k}(x) \neq \mathsf{core}_{g_k} \circ\pi(x)$. Let $T = T(\pi) \subseteq \{-1,1\}^k$ denote the set of all such inputs (so that $|T| = m$).

We can make a similar argument as for the completeness case: we have that $\left|Q_f^s \cap T\right|$ is a random variable with hypergeometric distribution (of parameters $s$, $2^k$, and $|T|$). Conditioned on $\left|Q_f^s \cap T\right|$, we also have $\left|Q_f^s \cap Q_g^s \cap T\right| \sim \operatorname{HyperGeom}(s, 2^k, \left|Q_f^s \cap T\right|)$, so that

$$\mathbf{E}\Big[\left|Q_f^s \cap Q_g^s \cap T\right|\Big] = \mathbf{E}\Big[\mathbf{E}\Big[\left|Q_f^s \cap Q_g^s \cap T\right| \;\Big|\; \left|Q_f^s \cap T\right|\Big]\Big] = \mathbf{E}\left[\frac{s\left|Q_f^s \cap T\right|}{2^k}\right] = \frac{s^2\,|T|}{2^{2k}} \geq (\epsilon - 2\epsilon')\frac{s^2}{2^k} = 30\epsilon'\frac{s^2}{2^k}.$$

(Recall that our threshold was set to $t = (3\alpha + 24\epsilon')\frac{s^2}{2^k} \leq 27\epsilon'\frac{s^2}{2^k}$.) Moreover, each element $x \in Q_f^s \cap Q_g^s \cap T$ will contribute to $V_\pi$ with probability at least $(1-8\epsilon')^2 > \frac{24}{25}$ (since this is a lower bound on the probability that both $\mathcal{A}_S^f(x) = \mathsf{core}_{f_k}(x)$ and $\mathcal{A}_S^g(x) = \mathsf{core}_{g_k}(x)$). As before, we can therefore write, letting $Z \stackrel{\text{def}}{=} \left|Q_f^s \cap Q_g^s \cap T\right|$, and taking $|T|$ to be minimal so that $|T| = (\epsilon - 2\epsilon')2^k$,

$$\begin{aligned}
\Pr[\,V_\pi > t\,] &\geq \Pr\left[V_\pi > t \;\middle|\; Z \geq \frac{13}{12}t\right]\Pr\left[Z \geq \frac{13}{12}t\right] \\
&\geq (1 - e^{-\frac{1}{2}\left(\frac{1}{26}\right)^2\cdot\frac{24}{25}\cdot\frac{13t}{12}})\Pr\left[Z \geq \frac{13}{12}t\right] = \left(1 - e^{-\frac{t}{1300}}\right)\Pr\left[Z \geq \frac{13}{12}t\right] \quad \text{(Chernoff bound)}
\end{aligned}$$

so that it is sufficient to lower bound $\Pr\Big[Z \geq \frac{13}{12}t\Big]$. To do so, we will bound the probability of the two following events:

**E1:** $Y \stackrel{\text{def}}{=} \left|Q_f^s \cap T\right| < \frac{99}{100}\frac{s|T|}{2^k}$

**E2:** $Z = \left|Q_f^s \cap Q_g^s \cap T\right| < \frac{99}{100}\frac{s}{2^k}\left|Q_f^s \cap T\right|$, conditioning on $\left|Q_f^s \cap T\right| \geq \frac{99}{100}\frac{s|T|}{2^k}$.

This will be sufficient for us to conclude, as by our choice of $t$, we have $\left(\frac{99}{100}\right)^2 \frac{s|T|^2}{2^{2k}} > \frac{13}{12} \cdot 27\epsilon' \frac{s^2}{2^k} > \frac{13}{12}t$, and therefore, by a Chernoff bound

$$
\begin{aligned}
\Pr\left[Z < \frac{13}{12}t\right] &\leq \Pr\left[Z < \left(\frac{99}{100}\right)^2 \frac{s\,|T|^2}{2^{2k}}\right] \\
&\leq \Pr\left[Y < \frac{99}{100}\frac{s\,|T|}{2^k}\right] + \Pr\left[Z < \frac{99}{100}\frac{s}{2^k}Y \;\middle|\; Y \geq \frac{99}{100}\frac{s\,|T|}{2^k}\right]\Pr\left[Y \geq \frac{99}{100}\frac{s\,|T|}{2^k}\right] \\
&\leq \Pr\left[Y < \frac{99}{100}\frac{s\,|T|}{2^k}\right] + \Pr\left[Z < \frac{99}{100}\frac{s}{2^k}Y \;\middle|\; Y \geq \frac{99}{100}\frac{s\,|T|}{2^k}\right] \\
&< \exp\left(-\frac{1}{2}\cdot\left(\frac{1}{100}\right)^2 \cdot \frac{s|T|}{2^k}\right) + \Pr\left[Z < \frac{99}{100}\frac{sY}{2^k} \;\middle|\; Y = \frac{99}{100}\frac{s\,|T|}{2^k}\right] \\
&\leq \exp\left(-\frac{1}{2}\cdot\left(\frac{1}{100}\right)^2 \cdot \frac{s(\epsilon - 2\epsilon')\cdot 2^k}{2^k}\right) + \exp\left(-\frac{1}{2}\cdot\left(\frac{2}{100}\right)^2 \cdot \frac{s^2(\epsilon - 2\epsilon')\cdot 2^k}{2^{2k}}\right) \\
&< \exp(-\tau C^2 k \ln k),
\end{aligned}
$$

by the choice $s = C\frac{2^{k/2}}{\epsilon}\sqrt{k \ln k}$, and for some constant $\tau \in (0,1)$. Hence setting $C$ to a sufficiently large constant, the foregoing analysis implies that $\Pr[\,V_\pi \leq t\,] \leq e^{-\frac{t}{1300}} + e^{-\tau C^2 k \ln k} = e^{-\frac{12c+3/4}{1300}\epsilon s^2/2^k} + e^{-\tau C^2 k \ln k} \leq \frac{7}{15k^k}$. A union bound over all $k! < k^k$ permutations $\pi \in \mathcal{S}_k$ finally yields $\Pr[\,\exists\pi,\, V_\pi \leq t\,] \leq \frac{7}{15}$ as claimed.

■

# 8 Acknowledgements

The authors would like to thank Lap Chi Lau for useful discussions, and in particular his suggestion to look at [LSW15].

# References

[AB10] Noga Alon and Eric Blais. Testing Boolean function isomorphism. In Maria J. Serna, Ronen Shaltiel, Klaus Jansen, and José D. P. Rolim, editors, *APPROX–RANDOM 2010*, volume 6302 of *Lecture Notes in Computer Science*, pages 394–405. Springer, 2010. 1.1, 7.1.2

[ABC+13] Noga Alon, Eric Blais, Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Nearly tight bounds for testing function isomorphism. *SIAM J. Comput.*, 42(2):459–493, 2013. 1.1, 1.2

[ACCL07] N. Ailon, B. Chazelle, S. Comandur, and D. Liue. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31(3):371–383, 2007. 1

[AD11] Anne Auger and Benjamin Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. Series on theoretical computer science. World Scientific, 2011. 6

[Bac13] Francis R. Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145–373, 2013. 5

[BL97] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271, 1997. 1

[Bla08] Eric Blais. Improved bounds for testing juntas. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 317–330. Springer, 2008. 1

[Bla09] Eric Blais. Testing juntas nearly optimally. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 151–158. ACM, 2009. 1, 1.2, 1.2, 3, 7.1.2, A

[Bla12] Eric Blais. *Testing properties of Boolean functions*. PhD thesis, CMU, 2012. 3.1, 3.2, A, A

[Blu94] Avrim Blum. Relevant examples and relevant features: Thoughts from computational learning theory. In *AAAI Fall Symposium on 'Relevance*, volume 5, 1994. 1

[BMR16] Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Tolerant Testers of Image Properties. In *ICALP*, 2016. To appear. 1

[CFGM12] Sourav Chakraborty, Eldar Fischer, David GacíaSoriano, and Arie Matsliah. Junto-symmetric functions, hypergraph isomorphism and crunching. In *Computational Complexity (CCC), 2012 IEEE 27th Annual Conference on*, pages 148–158. IEEE, 2012. 1, 1.1

[CG04] Hana Chockler and Dan Gutfreund. A Lower Bound for Testing Juntas. *Information Processing Letters*, 90(6):301–305, 2004. 1

[CGM11] Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Efficient sample extractors for juntas with applications. In *ICALP 2011*, volume 6755 of *Lecture Notes in Computer Science*, pages 545–556. Springer, 2011. 1.2, 7.1.2, 7.5, 7.1.2, 7.1.2, A, A.1, A.2, A.3, A.4, A.5, A.6, A.7, A

[CGR13] Andrea Campagna, Alan Guo, and Ronitt Rubinfeld. Local reconstructors and tolerant testers for connectivity and diameter. In *APPROX-RANDOM*, volume 8096 of *Lecture Notes in Computer Science*, pages 411–424. Springer, 2013. 1

[DLM$^+$07] Ilias Diakonikolas, Homin K Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A Servedio, and Andrew Wan. Testing for concise representations. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 549–558. IEEE, 2007. 1

[FF06] Eldar Fischer and Lance Fortnow. Tolerant Versus Intolerant Testing for Boolean Properties. *Theory of Computing*, 2(9):173–183, 2006. 1

[FKR+04] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. *J. Comput. Syst. Sci.*, 68(4):753–787, 2004. 1, 1.1, 1.2, 3, 6

[FN07] Eldar Fischer and Ilan Newman. Testing versus estimation of graph properties. *SIAM J. Comput.*, 37(2):482–501, 2007. 1

[FR10] Shahar Fattal and Dana Ron. Approximating the distance to monotonicity in high dimensions. *ACM Transactions on Algorithms (TALG)*, 6(3):52, 2010. 1

[GLS12] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012. 5

[GR05] Venkatesan Guruswami and Atri Rudra. Tolerant locally testable codes. In *APPROX-RANDOM*, volume 3624 of *Lecture Notes in Computer Science*, pages 306–317. Springer, 2005. 1

[JL10] ZQ John Lu. The elements of statistical learning: data mining, inference, and prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 173(3):693–694, 2010. 1

[KNOW14] Pravesh Kothari, Amir Nayyeri, Ryan O'Donnell, and Chenggang Wu. Testing surface area. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1204–1214. Society for Industrial and Applied Mathematics, 2014. 1.1

[KR98] Michael Kearns and Dana Ron. Testing problems with sub-learning sample complexity. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 268–279. ACM, 1998. 1.1

[KS09] Swastik Kopparty and Shubhangi Saraf. Tolerant linearity testing and locally testable codes. In *APPROX-RANDOM*, volume 5687 of *Lecture Notes in Computer Science*, pages 601–614. Springer, 2009. 1

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1049–1065. IEEE, 2015. 5, 5.2, 5, 5, 5.2, 5.3, 4, 8

[MOS03] Elchanan Mossel, Ryan O'Donnell, and Rocco P Servedio. Learning juntas. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 206–212. ACM, 2003. 1

[MR09] Sharon Marko and Dana Ron. Approximating the distance to properties in bounded-degree and general sparse graphs. *ACM Trans. Algorithms*, 5(2), 2009. 1

[O'D14] Ryan O'Donnell. *Analysis of Boolean functions*. Cambridge University Press, 2014. 3

[PR02] Michal Parnas and Dana Ron. Testing the diameter of graphs. *Random Structures & Algorithms*, 20(2):165–183, 2002. 1.1

[PRR06] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006. 1, 1

[Sch02] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002. 5

[SF11] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM Journal on Computing*, 40(6):1715–1737, 2011. 1.2

[STW15] Rocco A. Servedio, Li-Yang Tan, and John Wright. Adaptivity helps for testing juntas. In *Conference on Computational Complexity*, volume 33 of *LIPIcs*, pages 264–279. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. 1

[Tel16] Roei Tell. A note on tolerant testing with one-sided error. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 23, page 32, 2016. 1

[Val15] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *Journal of the ACM (JACM)*, 62(2):13, 2015. 1

## A Proof of Proposition 7.6 (construction of a noisy sampler)

We provide in this appendix the proof of Proposition 7.6, restated below:

**Proposition 7.6** (Noisy sampler for close-to-junta functions)**.** *There are algorithms $\mathcal{A}_P, \mathcal{A}_S$ (respectively preprocessor and sampler), which both require oracle access to a function $f\colon \{-1,1\}^n \to \{-1,1\}$, and satisfy the following properties.*

- *The preprocessor $\mathcal{A}_P$ takes $\epsilon' \in (0,1]$, $\rho \in (0,1/2]$, $k \in \mathbb{N}$ as inputs, makes $O\Big(\frac{k\log\frac{k}{\epsilon'}}{\epsilon'\rho(1-\rho)^k}\Big)$ queries to $f$, and either returns* fail *or a state $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$. The sampler $\mathcal{A}_S$ takes as input such a state $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$, makes a single query to $f$, and outputs a pair $(x,a) \in \{-1,1\}^k \times \{-1,1\}$. We say that a state $\sigma$ is $\gamma$-*good *if for some permutation $\pi \in \mathcal{S}_k$, $\mathcal{A}_S(\sigma)$ is a $\gamma$-noisy sampler for $\mathsf{core}_{f_k} \circ \pi$.*
- *$\mathcal{A}_P(\epsilon', \rho, k)$ fulfills the following conditions:*
  - *(i) If $\operatorname{dist}(f, \mathcal{J}_k) \le \frac{\rho}{8}\epsilon'$, then with probability at least $4/5$, $\mathcal{A}_P$ returns a state $\sigma$ that is $8\epsilon'$-good.*
  - *(ii) If $\operatorname{dist}(f, \mathcal{J}_k) > \epsilon'$, then with probability at least $4/5$, $\mathcal{A}_P$ returns* fail.
  - *(iii) If $\operatorname{dist}(f, \mathcal{J}_k) \le \epsilon'$, then with probability at least $4/5$, $\mathcal{A}_p$ either returns* fail *or returns a state $\sigma$ that is $8\epsilon'$-good.*

We will very closely follow the argument from the full version of [CGM11] (Proposition 4.16),[7] adapting the corresponding parts in order to obtain our result. For completeness, we tried to make this appendix below self-contained, reproducing almost verbatim several parts of the proof from [CGM11].[8]

**Proof of Proposition 7.6:** In order to use our result from Section 6 *in lieu* of the junta tester from [Bla09], we first need to make a small modification to our algorithm. Specifically, in its first step our tester will now pick a random partition $\mathcal{I}$ of $[n]$ in $\ell \stackrel{\text{def}}{=} \frac{Ck^2}{\epsilon}$ parts instead of $24k^2$ (for some (small) absolute constant $C > 1$). It is easy to check that both Lemma 3.2 and Lemma 3.3 still hold (e.g., from the proof of [Bla12, Lemma 5.4]), now with probability at least 19/20. Moreover, our modified tolerant tester offers the same soundness and completeness guarantees as Theorem 1.2, at the price of a query complexity $O\Big(\frac{k\log(k/\epsilon)}{\epsilon\rho(1-\rho)^k}\Big)$ (instead of $O\Big(\frac{k\log k}{\epsilon\rho(1-\rho)^k}\Big)$). Moreover, in Step 4 of Algorithm 4, i.e. when the algorithm found a suitable set $J \subseteq [\ell]$ as a witness for accepting, we make the algorithm return $\mathcal{I}$ and the set $\mathcal{J} \stackrel{\text{def}}{=} \{I_j\}_{j\in J}$ along with the verdict accept.

We will also require the definitions of the distribution induced by a partition $\mathcal{I}$ and a subset $\mathcal{J} \subseteq \mathcal{I}$, and of such a couple $(\mathcal{I}, \mathcal{J})$ being *good* for a function:

**Definition A.1** ([CGM11, Definition 4.6])**.** For any partition $\mathcal{I} = (I_1, \dots, I_\ell)$ of $[n]$, and subset of parts $\mathcal{J} \subseteq \mathcal{I}$, we define a pair of distributions:

**The distribution $D_\mathcal{I}$ on $\{-1,1\}^n$.** An element $y \sim D_\mathcal{I}$ is sampled by

1. picking $z \in \{-1,1\}^\ell$ uniformly at random among all $\binom{\ell}{\ell/2}$ strings of weight $\frac{\ell}{2}$;

[7] The full version can be found at http://www.cs.technion.ac.il/~ariem/eseja.pdf.

[8] The reader may notice that Chakraborty et al. rely on a definition of set-influence that differs from ours by a factor 2; we propagated the changes through the argument.

2. setting $y_i = z_j$ for all $j \in [\ell]$ and $i \in I_j$.

**The distribution $D_{\mathcal{J}}$ on $\{-1,1\}^{|\mathcal{J}|}$.** An element $x \sim D_{\mathcal{J}}$ is sampled by

1. picking $y \sim D_{\mathcal{I}}$;
2. outputting $\mathsf{extract}_{(\mathcal{I},\mathcal{J})}(y)$, where $x = \mathsf{extract}_{(\mathcal{I},\mathcal{J})}(y)$ is defined as follows. For all $j \in [\ell]$ such that $I_j \in \mathcal{J}$:
   - if $I_j \neq \emptyset$, set $x_j = y_i$ (where $i \in I_j$);
   - if $I_j = \emptyset$, set $x_j$ to be a uniformly random bit.

**Lemma A.2** ([CGM11, Lemma 4.7]). *$D_{\mathcal{I}}$ and $D_{\mathcal{J}}$ as above satisfy the following.*

- *For all $a \in \{-1,1\}^n$, $\Pr_{\mathcal{I},y\sim D_{\mathcal{I}}}[\, y = a \,] = \frac{1}{2^n}$.*
- *Assume $\ell > 4\,|\mathcal{J}|^2$. For every $\mathcal{I}$ and $\mathcal{J} \subseteq \mathcal{I}$, the total variation distance between $D_{\mathcal{J}}$ and the uniform distribution on $\{-1,1\}^{|\mathcal{J}|}$ is bounded by $2\,|\mathcal{J}|^2/\ell$. Moreover, the $\ell_\infty$ distance between the two distributions is at most $4\,|\mathcal{J}|^2/(\ell 2^{|\mathcal{J}|})$.*

**Definition A.3** ([CGM11, Definition 4.8]). Given $(\mathcal{I},\mathcal{J})$ as above and oracle access to $f\colon \{-1,1\}^n \to \{-1,1\}$, we define a probabilistic algorithm $\mathsf{sampler}_{(\mathcal{I},\mathcal{J})}(f)$ that on each execution produces a pair $\langle x, a\rangle \in \{-1,1\}^{|\mathcal{J}|} \times \{-1,1\}$ as follows: first it picks a random $y \sim D_{\mathcal{I}}$, then it queries $f$ on $y$, computes $x = \mathsf{extract}_{(\mathcal{I},\mathcal{J})}(y)$ and outputs the pair $\langle x, f(y)\rangle$.

**Definition A.4** ([CGM11, Definition 4.9]). Given $\alpha > 0$, a function $f\colon \{-1,1\}^n \to \{-1,1\}$, a partition $\mathcal{I} = (I_1,\dots,I_\ell)$ of $[n]$ and a subset $\mathcal{J} \subseteq \mathcal{I}$ of $k$ parts, we call the pair $(\mathcal{I},\mathcal{J})$ *$\alpha$-good* (with respect to $f$) if there exists a $k$-junta $h \in \mathcal{J}_k$ such that the following conditions are satisfied:

1. Conditions on $h$:
   (a) Every relevant variable of $h$ is also a relevant variable of $f_k$;
   (b) $\operatorname{dist}(h, f_k) \leq \alpha$.
2. Conditions on $\mathcal{I}$:
   (a) For all $j \in [\ell]$, $I_j$ contains at most one variable of $\mathsf{core}_{f_k}$;
   (b) $\Pr_{y\sim D_{\mathcal{I}}}[\, f(y) \neq f_k(y) \,] \leq 10 \cdot \operatorname{dist}(f, f_k)$.
3. Condition on $\mathcal{J}$: the set $S \stackrel{\text{def}}{=} \bigcup_{I\in\mathcal{J}} I$ contains all relevant variables of $h$.

**Lemma A.5** ([CGM11, Lemma 4.10]). *Let $\alpha, f, \mathcal{I}, \mathcal{J}$ be as in the preceding definition. If the pair $(\mathcal{I},\mathcal{J})$ is $\alpha$-good (with respect to $f$), then $\mathsf{sampler}_{(\mathcal{I},\mathcal{J})}(f)$ (as per Definition A.3) is an $(\eta,\mu)$-noisy sampler for some permutation of $\mathsf{core}_{f_k}$, with $\eta \leq 2\alpha + \frac{4k^2}{\ell} + 10\cdot \operatorname{dist}(f, f_k)$ and $\mu \leq \frac{4k^2}{\ell}$.*

The last piece we shall need is the ability to convert an $(\eta,\mu)$-noisy sampler to a $(\eta',0)$-noisy sampler – that is, one whose samples are exactly uniformly distributed.

**Lemma A.6** ([CGM11, Lemma 4.4]). *Let $\tilde{g}$ be an $(\eta,\mu)$-noisy sampler for $g\colon \{-1,1\}^k \to \{-1,1\}$, that on each execution picks $x$ according to some fixed (and fully known) distribution $D$. Then $\tilde{g}$ can be turned into an $(\eta+\mu)$-noisy sampler $\tilde{g}_{\text{unif}}$ for $g$.*

With this in hand, we are ready to prove the main lemma:

**Lemma A.7** (Analogue of [CGM11, Proposition 4.16])**.** *The tester from Theorem 1.2, modified as above, as the following guarantees. It has query complexity $O\left(\frac{k\log(k/\epsilon)}{\epsilon\rho(1-\rho)^k}\right)$ and outputs, in case of acceptance, a partition $\mathcal{I}$ of $[n]$ in $\ell \stackrel{\text{def}}{=} O(k^2/\epsilon)$ parts along with a subset $\mathcal{J} \subseteq \mathcal{I}$ of $k$ parts such that for any $f$ the following conditions hold:*

- *if $\operatorname{dist}(f, \mathcal{J}_k) \leq \frac{\rho}{8}\epsilon$, the algorithm accepts with probability at least $9/10$;*
- *if $\operatorname{dist}(f, \mathcal{J}_k) > \epsilon$, the algorithm rejects with probability at least $9/10$;*
- *for any $f$, with probability at least $4/5$ either the algorithm rejects, or it outputs $\mathcal{J}$ such that the pair $(\mathcal{I}, \mathcal{J})$ is $(1+\frac{3}{2}\rho)\epsilon$-good (as per Definition A.4).*

*In particular, if $\operatorname{dist}(f, \mathcal{J}_k) \leq \frac{\rho}{8}\epsilon$, then with probability at least $4/5$ the algorithm outputs a set $\mathcal{J}$ such that $(\mathcal{I}, \mathcal{J})$ is $(1+\frac{3}{2}\rho)\epsilon$-good.*

**Proof of Lemma A.7:** The first two items follow from the analysis of the tester (Theorem 1.2) and the foregoing discussion; we thus turn to establishing the third item.

Called with parameters $k, \rho, \epsilon$, our algorithm, with probability at least $19/20$, either rejects or outputs a partition $\mathcal{I}$ of $[n]$ into $\ell = O(k^2)$ parts and set $\mathcal{J} \subseteq \mathcal{I}$ satisfying $\mathbf{Inf}_f(\cup_{I\in\mathcal{J}} I)) \leq \frac{\epsilon}{2}$. Let $R \subseteq [n]$ (with $|R| \leq k$) denote the set of relevant variables of $f_k$, and $V \supseteq R$ (with $|V| = k$) the set of relevant variables of $\mathsf{core}_{f_k}$. Assume that $\operatorname{dist}(f, \mathcal{J}_k) \leq \frac{\rho\epsilon}{2}$.[9] We then have:

- by the above, with probability at least $19/20$ the algorithm outputs a set $\mathcal{J} \subseteq \mathcal{I}$ (or more precisely $\mathcal{J}$, possibly extended with "dummy parts" to amount to $k$ parts) which satisfies
$$\mathbf{Inf}_f(\overline{\cup_{I\in\mathcal{J}} I})) \leq \frac{\epsilon}{2};$$
- since $\ell \gg k^2$, with probability at least $19/20$ all elements of $V$ fall in different parts of the partition $\mathcal{I}$;
- by Lemma A.2 and by Markov's inequality, with probability at least $9/10$ the partition $\mathcal{I}$ satisfies $\Pr_{y\sim\mathcal{D}_\mathcal{I}}[\, f(y) \neq f_k(y)\,] \leq 10 \cdot \operatorname{dist}(f, f_k)$.

So by a union bound, 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 $(1+\frac{3}{2}\rho)\epsilon$-good. Let $U \stackrel{\text{def}}{=} R \cap \left(\bigcup_{I\in\mathcal{J}} I\right)$ (informally, U is the subset of the relevant variables of $f_k$ that were successfully "discovered" by the tester). Since $\operatorname{dist}(f, \mathcal{J}_k) \leq \frac{\rho\epsilon}{2}$, we have $\mathbf{Inf}_f(\bar{V}) \leq 2\operatorname{dist}(f, \mathcal{J}_k) \leq \rho\epsilon$. By the subadditivity and monotonicity of influence we get

$$\mathbf{Inf}_f(\bar{U}) \leq \mathbf{Inf}_f(\bar{V}) + \mathbf{Inf}_f(V \setminus U) \leq \mathbf{Inf}_f(\bar{V}) + \mathbf{Inf}_f(\overline{\cup_{I\in\mathcal{J}} I})) \leq \rho\epsilon + \frac{\epsilon}{2}.$$

where the second inequality follows from $V \setminus U \subseteq \overline{\cup_{I\in\mathcal{J}} I}$. This means (see e.g. [Bla12, Lemma 2.21]) that there is a $k$-junta $h$ on $U$ such that $\operatorname{dist}(f, h) \leq 2(\rho\epsilon + \frac{\epsilon}{2})$, and by the triangle inequality $\operatorname{dist}(f_k, h) \leq 2(\rho\epsilon + \frac{\epsilon}{2}) + \frac{\rho\epsilon}{2} = (1+\frac{3}{2}\rho)\epsilon$. Based on this $h$, we can verify that the pair $(\mathcal{I}, \mathcal{J})$ is $(1+\frac{3}{2}\rho)\epsilon$-good by going over the conditions in Definition A.4. ■

**Concluding the proof of Proposition 7.6.** We conclude as in Section 4.6 of [CGM11], and start by describing how $\mathcal{A}_P$ and $\mathcal{A}_S$ operate. The preprocessor $\mathcal{A}_P$ starts by calling the tester $\mathcal{T}$ of Lemma A.7. Then, in case $\mathcal{T}$ accepted, $\mathcal{A}_P$ encodes in the state $\sigma$ the partition $\mathcal{I}$ and the subset

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

$\mathcal{J} \subseteq \mathcal{I}$ output by $\mathcal{T}$ (see Lemma A.7), along with the values of $k$ and $\epsilon$. The sampler $\mathcal{A}_S$, given $\sigma$, obtains a pair $\langle x, a \rangle \in \{-1,1\}^k \times \{-1,1\}$ by executing $\mathsf{sampler}_{(\mathcal{I},\mathcal{J})}(f)$ (from Definition A.3) once. Now we show how Proposition 7.6 follows from Lemma A.7. The first two items are immediate. As for the third item, notice that we only have to analyze the case where $\mathrm{dist}(f, f_k) \leq \frac{\rho\epsilon}{2}$ and $\mathcal{T}$ accepted; all other cases are taken care of by the first two items. By the third item in Lemma A.7, with probability at least 4/5 the pair $(\mathcal{I}, \mathcal{J})$ is $(1 + \frac{3}{2}\rho)\epsilon$-good. If so, by Lemma A.5 $\mathsf{sampler}_{(\mathcal{I},\mathcal{J})}(f)$ is an $(\eta, \mu)$-noisy sampler for some permutation of $\mathsf{core}_{f_k}$, where

$$\eta \leq 2(1 + \frac{3}{2}\rho)\epsilon + \frac{4k^2}{\ell} + 10 \cdot \mathrm{dist}(f, \mathcal{J}_k) \leq 2(1 + 4\rho)\epsilon + \frac{4k^2}{\ell}$$

and $\mu \leq \frac{4k^2}{\ell}$. This in turn implies by Lemma A.6 an $\eta'$-noisy sampler, for

$$\eta' = \eta + \mu \leq 2(1 + 4\rho)\epsilon + \frac{8k^2}{\ell} \leq 4(1 + 2\rho)\epsilon \leq 8\epsilon$$

as claimed. (Where we used that $\frac{8k^2}{\ell} \leq \epsilon$ by our choice of $\ell$.) ■



ECCC ISSN 1433-8092
http://eccc.hpi-web.de