New Statistical and Computational Results for Learning Junta Distributions

Lorenzo Beretta
University of California, Santa Cruz, CA, USA

Abstract

We study the problem of learning junta distributions on $\{\pm 1\}^n$, where a distribution is a k-junta if its probability mass function depends on a subset of at most k variables. We make two main contributions:

- We show that learning k-junta distributions is *computationally* equivalent to learning k-parity functions with noise (LPN), a landmark problem in computational learning theory.
- We design an algorithm for learning junta distributions whose statistical complexity is optimal, up to polylogarithmic factors. Computationally, our algorithm matches the complexity of previous (non-sample-optimal) algorithms.

Combined, our two contributions imply that our algorithm cannot be significantly improved, statistically or computationally, barring a breakthrough for LPN.

2012 ACM Subject Classification Theory of computation → Boolean function learning

Keywords and phrases Junta Distributions, Learning Parities with Noise

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.31

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2505.05819

Acknowledgements I would like to thank Caleb Koch for the many insightful conversations on juntas, and for introducing me to learning theory in the first place.

1 Introduction

We study the problem of learning k-junta distributions, as first introduced by Aliakbarpour, Blais, and Rubinfeld [1]. Given $n \in \mathbb{N}$ and $k \leqslant n$, a distribution $\mathcal{P}$ supported on $\{\pm 1\}^n$ is a k-junta distribution if its probability mass function $\mathcal{P}(x) = \mathbb{P}_{y\sim\mathcal{P}}[y = x]$ is a k-junta function, where a k-junta function on $\{\pm 1\}^n$ is a function that only depends on k out of n input variables[1]. The objective of the learning problem is to design algorithms that, given i.i.d. samples from an unknown k-junta distribution $\mathcal{P}$ over $\{\pm 1\}^n$, output the p.m.f. of a hypothesis distribution $\mathcal{D}$ that satisfies $||\mathcal{D} - \mathcal{P}||_{TV} \leqslant \varepsilon$.

In this work, we show that learning junta distributions is *computationally* as hard as learning parity functions with noise, a landmark problem in learning theory (Figure 1). We complement this result by designing a computationally-efficient algorithm for learning junta distributions that achieves almost optimal statistical complexity, improving over previous work (Table 1).

[1] More generally, [1] defines a k-junta distribution with respect to a fixed distribution $\mathcal{D}$. For $k \leqslant n$ and a fixed distribution $\mathcal{D}$ supported on $\{\pm 1\}^n$, a distribution $\mathcal{P}$ over $\{\pm 1\}^n$ is a k-junta distribution with respect to $\mathcal{D}$ if there exists a size-k set of coordinates $J \subseteq [n]$ such that, for every $x \in \{\pm 1\}^k$, the distributions $\mathcal{P}$ and $\mathcal{D}$ conditioned on coordinates in J being set according to x are equal. When $\mathcal{D}$ is the uniform distribution, the above definition is equivalent to the requirement that the probability mass function of $\mathcal{P}$ is a k-junta function.

1.1 Related Work

Feature selection and learning juntas

A key challenge in machine learning is learning target concepts despite the presence of irrelevant features [13]. As observed in [11], the task of PAC learning in the presence of irrelevant features can be formalized as the problem of learning the concept class of k-junta functions for $k \ll n$. Likewise, feature selection in the context of *distribution learning* is formalized by the task of learning junta distributions. Statistical and computational aspects of learning (and testing) juntas have been the subject of extensive work [36, 32, 34, 12, 35, 7, 4, 3, 14, 5, 26, 20, 10].

Learning junta functions

Computationally, learning junta functions is hard. In the *distribution-free* setting, learning k-juntas takes $n^{\Omega(k)}$ time, assuming randomized ETH [30]. Interestingly, characterizing the complexity of learning juntas with respect to the *uniform distribution* proved to be an elusive problem that resisted decades of attempts [32, 36, 8, 20, 9]. The fastest algorithm to date runs in time $\approx n^{0.6k}$, which is only moderately faster than brute-force [36, 2].

Learning parity functions with noise

Learning a function $f : \{\pm 1\}^n \to \{\pm 1\}$ under distribution $\mathcal{D}$ with noise rate $\eta < 1/2$ means recovering f while observing pairs $(x, f(x) \cdot b)$, where $x \sim \mathcal{D}$ and $b \in \{\pm 1\}$ is a Rademacher random variable with parameter η. A function f is a parity with relevant variables $J \subseteq [n]$ if $f(x) = \prod_{i \in J} x_i$. Learning parities with noise under the uniform distribution (LPN) is a cornerstone problem in computational learning theory, serving as a foundation for several hardness results (see Figure 1).

Feldman, Gopalan, Khot, and Ponnuswami first demonstrated the central role of LPN by computationally reducing several fundamental problems in learning theory to it [20, 21]. They showed that, under the uniform distribution, agnostically learning k-parity functions, learning k-junta functions (both with and without noise), and learning Boolean functions that admit a DNF representation of size 2^k all reduce to learning k-parities with noise. Furthermore, LPN can be reduced to numerous other learning problems: learning halfspaces [29] and ReLU functions [23] under the Gaussian distribution, learning submodular functions [22] and agnostically learning halfspaces under the uniform distribution [25].

LPN is widely believed to be computationally hard. Kearns established that LPN requires $\Omega(n^k)$ constant-noise queries in the statistical query (SQ) model [26, 6]. The learning with errors (LWE) problem, a generalization of LPN over $\mathbb{F}_q$, is widely conjectured to be hard and underpins modern lattice-based cryptography [33]. Moreover, LPN itself has been leveraged as a hardness assumption in cryptographic constructions [28].

The best known algorithm for sparse LPN ($k \ll n$) runs in time $\approx n^{0.8k}$ [36, 2], and neural networks trained with SGD encounter the same n^k computational barrier [6]. Finally, for $k \approx n$ the fastest known algorithm for LPN runs in time $2^{\frac{n}{\log n}}$, and extending this to $n^{\frac{k}{\log k}}$ for arbitrary k remains a major open problem [10].

Learning junta distributions

The problem of learning junta distributions (LJD) was introduced by Aliakbarpour, Blais, and Rubinfeld [1]. In their work, they observed that the *cover method*[2] gives an algorithm for LJD with (essentially) optimal sample complexity (see Table 1 for exact statistical and computational bounds). Computationally, the cover-method algorithm suffers from a doubly-exponential dependence on k. The main algorithmic contribution of [1] is to design an algorithm that achieves low sample complexity ($\approx 2^{2k} \log n$) while being computationally efficient ($\approx n^k$). While a computational cost of n^k does not look efficient at first sight, [1] proved that the problem of learning junta functions without noise (LJ) reduces to LJD, which rules out algorithms substantially faster than n^k, barring a breakthrough for LJ.

Chen, Jayaram, Levi and Waingarten studied the problem of learning junta distributions in a stronger model featuring *subcube conditioning* [15]. Subcube conditioning allows to sample x conditioned on $x_i = q_i$ for all $i \in I$, given $I \subseteq [n]$ and $q \in \{\pm 1\}^I$. The power of this stronger model makes the problem computationally easier; in fact, they show an algorithm running exponentially faster than previous work. Surprisingly, this stronger model does not make the problem any easier statistically, as the lower bounds of [15] match the statistical complexity achieved by the cover method.

Finally, Escudero Gutiérrez improved the algorithmic result of [1] statistically, shaving a factor 2^k while keeping the computational cost at $\approx n^k$ [19].

Truncated distributions

Learning junta distributions connects with an emerging line of work on testing distributions truncated by low-degree polynomials. A distribution is truncated by a function f if it is obtained by conditioning a ground distribution $x \sim \mathcal{D}$ on the value of $f(x)$. Recently, testing junta truncation [24], as well as low-degree polynomial truncation [16], have been considered. Apparently, a uniform distribution truncated by a junta function is a junta distribution, so junta truncation testing reduces to LJD. Moreover, juntas are low-degree polynomials, so low-degree polynomial truncation is a generalization of junta truncation.

1.2 Our Results

As stated above, [1] reduced the problem of learning k-junta functions without noise (LJ) to that of learning k-junta distributions (LJD). Since improving the computational complexity for LJ is a long-standing open problem, their reduction serves as a proof of *conditional hardness* for LJD. In this work, we strengthen this message and prove that LJD is computationally equivalent to learning k-parity functions with noise (LPN). See Figure 1 for a summary of these reductions.

▶ **Theorem 1.** *The computational complexity of these two problems is equal, up to* $\mathrm{poly}(2^k, n)$ *factors.*

- *Learning k-parity functions over $\{\pm 1\}^n$ with respect to the uniform distribution with noise rate $\eta = \frac{1}{2} - 2^{-O(k)}$*
- *Learning k-junta distributions over $\{\pm 1\}^n$ with sample access.*

Since LPN is a central problem in computational learning theory, we believe that Theorem 1 essentially closes the question about the computational complexity of LJD.

[2] See [17] for a general introduction to the method.

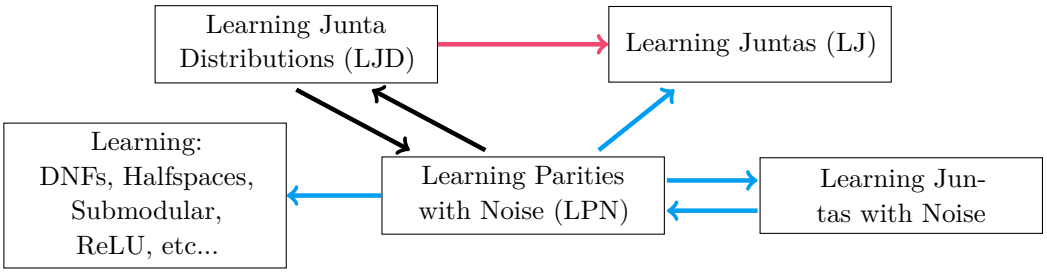

Figure 1 An arrow from A to B means that problem B reduces to problem A (i.e., A is harder than B). The blue arrows were proven in previous work, the red arrow was proven in [1] and the black arrows are proven in this work.

On the algorithmic side, we design a new algorithm for LJD which improves over [1] and [19]. In particular, [19] achieves a sample complexity of $\approx 2^k \cdot \log n$ and we improve it to $\approx 2^k + \log n$, which is a quadratic speedup for $k = \log \log n$. Moreover, our algorithm runs in time $\approx n^k$ which we expect to be hard to improve significantly, given Theorem 1. Finally, the sample complexity achieved by our algorithm matches the lower bound of [15] up to polylogarithmic factors. See Table 1 for an exhaustive history of the computational and statistical complexity of learning junta distributions.

▶ **Theorem 2.** *There exists an algorithm that properly learns k-junta distributions up to TV distance ε using $O(\frac{k}{\varepsilon^2}(2^k + \log n))$ samples and $O(\min(2^n, n^k) \cdot \frac{k}{\varepsilon^2}(2^k + \log n))$ running time.*

Table 1 Statistical and computational complexity of learning junta distributions.

Algorithms	Samples	Time
Cover Method	$O\left(\frac{1}{\varepsilon^2} \cdot (2^k \log \frac{1}{\varepsilon} + k \log n)\right)$	$O\left(\binom{n}{k} \cdot \varepsilon^{-\Omega(2^k)}\right)$
[1]	$O\left(\frac{k}{\varepsilon^4} \cdot 2^{2k} \log n\right)$	$\tilde{O}\left(\min(2^n, n^k) \cdot \frac{k}{\varepsilon^4} \cdot 2^{2k}\right)$
[19]	$O\left(\frac{k}{\varepsilon^2} \cdot 2^k \log n\right)$	$O\left(\min(2^n, n^k) \cdot \frac{k}{\varepsilon^2} \cdot 2^k \log n\right)$
This Work	$O\left(\frac{k}{\varepsilon^2}(2^k + \log n)\right)$	$O\left(\min(2^n, n^k) \cdot \frac{k}{\varepsilon^2}(2^k + \log n)\right)$
Lower Bounds		
[15]	$\Omega\left(\frac{1}{\varepsilon^2} \cdot \left(2^k + \log \binom{n}{k}\right)\right)$	

1.3 Technical Overview: Computational Complexity

In order to prove Theorem 1, we need to reduce learning parities with noise (LPN) to learning junta distributions (LJD) and vice versa.

1.3.1 Reducing LPN to LJD

Let (x, y) be a sample-label input for LPN with ground truth $\chi_J(x) = \prod_{j \in J} x_j$. The sample $x \in \{\pm 1\}^n$ is distributed uniformly, whereas $y \in \{\pm 1\}$ is distributed as $\chi_J(x)$ flipped with probability $\eta < 1/2$.

We implement a filtering strategy that, given LPN samples (x, y), returns samples $z \sim \mathcal{D}$, where $\mathcal{D}$ is a junta distribution. Then, we show that learning $\mathcal{D}$ suffices to recover χ_J. We define $z \leftarrow x$ if $y = 1$ and re-sample (x, y) otherwise. This ensures that $\mathcal{D}$ is a mixture of

two distributions: (i) The uniform distribution supported on $\{w \in \{\pm 1\}^n \mid \chi_J(w) = 1\}$; (ii) The uniform distribution $\mathcal{U}_n$ over $\{\pm 1\}^n$. In this mixture (i) is weighted $\approx 1 - 2\eta$ and (ii) is weighted $\approx 2\eta$. Moreover, $\mathcal{D}$ is a k-junta distribution with relevant variables J.

By running an exponential search, we can assume that we know $1 - 2\eta$ up to a constant factor. We run our LJD learner on $\mathcal{D}$ with precision $\varepsilon \cdot (1 - 2\eta)$, and it returns $\mathcal{F}$ such that $||\mathcal{F} - \mathcal{D}||_1 \leqslant \varepsilon \cdot (1 - 2\eta)$. Now, define a boolean function $f : \{\pm 1\}^n \to \{\pm 1\}$ as $f(x) = 1$ iff $\mathcal{F}(x) > 1/2$. Our estimate $\mathcal{F}$ is precise enough so that f satisfies $\mathbb{E}_x [|f - \chi_J|] \leqslant \varepsilon$. Notice that since we have query access to the probability mass function of $\mathcal{F}$, we also have query access to f. In order to retrieve J from f, consider $f(w) \cdot f(w \oplus e_j)$ where $w \in \{\pm 1\}^n$ is uniformly distributed and $w \oplus e_j$ flips the j-th bit of w. By virtue of $\mathbb{E}_x [|f - \chi_J|] \leqslant \varepsilon$, we must have that $f(w) \cdot f(w \oplus e_j)$ has a majority of -1 for $j \in J$ and a majority of 1 for $j \notin J$. To conclude, we just test if $j \in J$ for all $j \in [n]$.

1.3.2 Reducing LJD to LPN

In order to reduce LJD to LPN we need to introduce an auxiliary problem: *learning noisy parity distributions* (LNPD). We say that a junta distribution is a parity distribution with relevant variables J if it coincides with the uniform distribution over $\{w \in \{\pm 1\}^n \mid \chi_J(w) = s\}$ for a fixed $s \in \{\pm 1\}$. We generalize this definition and say that a junta distribution is a noisy parity distribution if it is a mixture of a uniform distribution and a parity distribution. By symmetry, the only non-zero coefficients in the Fourier spectrum of (the p.m.f. of) a noisy parity distribution $\mathcal{D}$ with relevant variables J are $\widehat{\mathcal{D}}(\varnothing)$ and $\widehat{\mathcal{D}}(J)$. The reduction of LJD to LPN develops in two steps: First, we reduce LJD to LNPD; Second, we reduce LNPD to LPN.

Reducing LJD to LNPD

The Fourier spectrum of a junta distribution can have as many as 2^k non-zero coefficients, whereas a noisy parity distribution has at most two. The idea is to distill our input junta distribution $\mathcal{D}$ to $\mathcal{P}$ so that $\mathcal{P}$ is a noisy parity with relevant variables J such that $\widehat{\mathcal{P}}(J) = \widehat{\mathcal{D}}(J)$, for some fixed J. Then, use the LNPD algorithm to learn $\mathcal{P}$.

To perform such distillation, we use an idea first introduced in [20]: By carefully adding noise to the distribution $\mathcal{D}$ it is possible to zero all Fourier coefficients of $\mathcal{D}$ besides $\widehat{\mathcal{D}}(J)$ and $\widehat{\mathcal{D}}(\varnothing)$. Moreover, each $J \subseteq [n]$ with $\widehat{\mathcal{D}}(J) \neq 0$ has a large enough chance of being left intact. Thus, a coupon-collector style argument ensures that we learn all non-zero Fourier coefficients of $\mathcal{D}$ in $\approx k \cdot 2^k$ rounds.

Reducing LNPD to LPN

For ease of presentation, we show how to reduce learning (non-noisy) parity distributions to learning parity functions (without noise). The reduction carries over to the noisy case. Let $\mathcal{D}$ be a parity distribution over relevant variables J and such that $\chi_J(w) = 1$ for all $w \in \operatorname{supp}(\mathcal{D})$ (the case $\chi_J(w) = -1$ is symmetric). Let $z \sim \mathcal{D}$. Fix $j \in [n]$ and define (x, y) as follows: Flip the j-th bit of z with probability $1/2$ and define that to be x; define $y = -1$ if the j-th bit was flipped and $y = 1$ otherwise. If $j \in J$, then x is distributed uniformly over $\{\pm 1\}^n$ and $y = \chi_J(x)$. Else, if $j \notin J$ then y is distributed uniformly on $\{\pm 1\}$ and independently of x.

Then, we solve LPN on samples (x, y). If it turns out that y is a k-parity function of x then we apply the same technique used in the reduction from LPN to LJD above to retrieve S from f, and we find J. If $j \notin J$ then y is not going to be a parity function of x and we need to redefine $\mathcal{D}$ using some other $j \in [n]$. By scanning all $j \in [n]$, we are guaranteed that at some point $j \in J$ holds.

1.4 Technical Overview: Sample-Optimal Algorithm

Our algorithm learns JDN with fewer samples than previous work; in particular, it replaces the term $2^k \log n$ in the sample complexity of [19] with $2^k + \log n$ (see Table 1). The goal of this overview is to show how to make the term $\log n$ additive, rather than multiplicative.

Low-degree algorithms

Most previous literature on junta distributions, as well as our own algorithm, make use of the low-degree algorithm of Linial, Mansour, and Nisan [31]. In a nutshell, the low-degree algorithm estimates individual Fourier coefficients and outputs a hypothesis function consistent with the estimated Fourier coefficients. In [1], the authors pointed out that applying the low-degree algorithm to the junta distribution problem requires too many samples. Indeed, an accuracy of $\varepsilon \cdot 2^{-n/2}$ on individual Fourier coefficients is necessary to deliver an ε error in L_2-norm. To bypass this, Escudero introduced a thresholding of Fourier coefficients, thus ensuring that at most 2^k coefficients are estimated to be nonzero [19]. In this work, we simply apply the low-degree algorithm to distributions over a small domain $\{\pm 1\}^k$, thus the standard analysis from [31] suffices.

Our template algorithm

Consider the following template. We maintain a junta distribution $\mathcal{Q}$ as our "best guess" for $\mathcal{D}$, along with its set of relevant variables $N \subseteq [n]$ of size at most k. We maintain the invariant that $N \subseteq J$, where J is the true set of relevant variables. Given $S \supseteq N$, we can check if the marginals of $\mathcal{Q}$ and $\mathcal{D}$ with respect to the variables in S (resp. $\mathcal{Q}|_S$ and $\mathcal{D}|_S$) are close in total variation distance. In order to check the above condition, we draw samples from $\mathcal{D}$, project them onto the coordinates in S and feed these samples to a closeness tester.

If $||\mathcal{Q}|_S - \mathcal{D}|_S||_{TV} \leqslant \varepsilon$ for all $S \in \binom{[n]}{k}$ with $N \subseteq S$, then we can correctly output $\mathcal{Q}$. Else, if we find $S \in \binom{[n]}{k}$ with $N \subseteq S$ such that $||\mathcal{Q}|_S - \mathcal{D}|_S||_{TV} > \varepsilon$, then we learn the Fourier coefficients of $\mathcal{D}|_S$ using the low-degree algorithm. Then, we update the distribution $\mathcal{Q}$ so that it includes the newly recovered Fourier coefficients, and augment N accordingly. Each iteration of this scheme increases the size of N, so we are done after k iterations.

Sample complexity analysis

First, observe that while making multiple calls to an algorithm we can certainly re-use the same samples, and the price we pay is that we need to union bound over the error probability. The low-degree algorithm operates on distributions over $\{\pm 1\}^k$, so the cost of an individual call is negligible. Moreover, we make at most k calls to it, so a standard boosting of the success probability increases its sample complexity by at most $\log k$. On the other hand, we make up to $k \cdot n^k$ calls to the closeness tester. Each individual call costs at least $\tilde{\Omega}(2^k)$ by known lower bounds [37]. So, we cannot afford a standard boosting of the success probability, as it would introduce a multiplicative term $\log(n^k) = k \log n$.

To counter this, we develop a closeness tester that succeeds with probability $1 - \delta$ and uses only $s + \log 1/\delta$ samples, where s is the support size. The design of our tester uses the TV distance between the empirical probability mass function and the tested distribution as an estimator, borrowing ideas from [18].

1.5 Organization

In Section 2 we introduce notation. In Section 3 we prove the computational equivalence of learning junta distributions and learning parities with noise (Theorem 1). In Section 4 we show our new algorithm for the problem of learning junta distributions (Theorem 2).

2 Preliminaries

We denote by $\Delta(A)$ the set of probability distributions over the set A. Given a probability distribution $\mathcal{P} \in \Delta(\{\pm 1\}^n)$ and a set $S \subseteq [n]$ we denote by $\mathcal{P}|_S$ the marginal of $\mathcal{P}$ with respect to variables in S. We overload the notation and denote by $\mathcal{P}(\cdot)$ its probability mass function (PMF). We denote the uniform distribution over $\{\pm 1\}^n$ by $\mathcal{U}_n$. Whenever not specified, expectations are taken with respect to the uniform distribution.

Fourier transform

In this paper we use the Fourier transform over the boolean hypercube. For a subset $A \subseteq [n]$ we define the function $\chi_A : \{\pm 1\}^n \to \{\pm 1\}$ such that $\chi_A(x) = \prod_{i \in A} x_i$. Such functions form the Fourier basis of the vector space of functions $\{\pm 1\}^n \to \mathbb{R}$. Notice that the Fourier basis is orthonormal with respect to the dot product $\langle f, g \rangle = \mathbb{E}_x [f(x) \cdot g(x)]$.

Given a function $f : \{\pm 1\}^n \to \mathbb{R}$ we define its Fourier transform $\widehat{f} : 2^{[n]} \to \mathbb{R}$ as $\widehat{f}(A) = \mathbb{E}_x [f(x) \cdot \chi_A(x)]$. Thus, $f = \sum_{A \subseteq [n]} \widehat{f}(A) \cdot \chi_A$. Finally, we have

$$\mathop{\mathbb{E}}_{x \sim \mathcal{U}_n} \left[f^2(x)\right] = 2^{-n} \cdot ||f||_2^2 = \sum_{A \subseteq [n]} \widehat{f}(A)^2.$$

Learning functions and distributions

Fix a class of boolean functions $F \subseteq \{f : \{\pm 1\}^n \to \{\pm 1\}\}$. Consider an algorithm $\mathcal{A}$ that takes as input parameters $\varepsilon, \delta > 0$ and $\{(x_i, f(x_i))\}_{i \in [m]}$ where $x_1 \dots x_m \in \{\pm 1\}^n$ are i.i.d. samples from a distribution $\mathcal{D}$ and $f \in F$ is fixed. Moreover, $\mathcal{A}$ outputs a function $f_{\mathcal{A}}$ that satisfies $\mathbb{P}_{x \sim \mathcal{D}} [f(x) \neq f_{\mathcal{A}}(x)] \leqslant \varepsilon$ with probability at least $1 - \delta$. Then, we say that $\mathcal{A}$ learns F using $S(n)$ samples and $T(n)$ time under the distribution $\mathcal{D}$ if $\mathcal{A}$ takes $S(n) \cdot \mathrm{poly}(\varepsilon^{-1}, \delta^{-1})$ samples and runs in time $T(n) \cdot \mathrm{poly}(\varepsilon^{-1}, \delta^{-1})$. Moreover, we say that $\mathcal{A}$ learns F with noise of rate $\eta < 1/2$ if the label $f(x_i)$ in the samples $(x_i, f(x_i))$ is flipped with probability η.

Likewise, fix a class of distributions $P \subseteq \Delta(\{\pm 1\}^n)$. Consider an algorithm $\mathcal{A}$ that takes as input parameters $\varepsilon, \delta > 0$ and $x_1 \dots x_m$, where $x_1 \dots x_m \in \{\pm 1\}^n$ are i.i.d. samples from a fixed distribution $\mathcal{P} \in P$. Furthermore, $\mathcal{A}$ outputs a distribution $\mathcal{P}_{\mathcal{A}}$ that satisfies $||\mathcal{P} - \mathcal{P}_{\mathcal{A}}(x)||_{TV} \leqslant \varepsilon$ with probability at least $1 - \delta$. Then, we say that $\mathcal{A}$ learns P using $S(n)$ samples and $T(n)$ time if $\mathcal{A}$ takes $S(n) \cdot \mathrm{poly}(\varepsilon^{-1}, \delta^{-1})$ samples and runs in time $T(n) \cdot \mathrm{poly}(\varepsilon^{-1}, \delta^{-1})$.

In general, we can require an algorithm that learns a distribution to output either a *generator* or an *evaluator* for the learned distribution [27]. A generator is an algorithm that generate samples from the (approximate) target distribution, whereas an evaluator returns the value of the probability mass function on a given query point. In this work, we assume that a distribution learner returns evaluators.

Before delving into the proofs of Theorem 1 and Theorem 2, we prove two folklore lemmas that are going to be useful throughout the paper.

▶ **Lemma 3.** *Let $\mathcal{P}, \mathcal{Q} \in \Delta(\{\pm 1\}^n)$ be junta distributions with set of relevant variables $J \subseteq [n]$, then*

$$||\mathcal{P} - \mathcal{Q}||_1 = ||\mathcal{P}|_J - \mathcal{Q}|_J||_1.$$

Proof. For each $x \in \{\pm 1\}^J$, define $S_x = \{y \in \{\pm 1\}^n \mid y_j = x_j \text{ for all } j \in J\}$ and notice that $\mathcal{P}$ and $\mathcal{Q}$ are both constant on S_x.

$$\begin{aligned} &||\mathcal{P} - \mathcal{Q}||_1 = \\ &\sum_{x \in \{\pm 1\}^J} \sum_{y \in S_x} |\mathcal{P}(y) - \mathcal{Q}(y)| = \\ &\sum_{x \in \{\pm 1\}^J} \left| \sum_{y \in S_x} \mathcal{P}(y) - \mathcal{Q}(y) \right| = \\ &\sum_{x \in \{\pm 1\}^J} |\mathcal{P}|_J(x) - \mathcal{Q}|_J(x)| = \\ &||\mathcal{P}|_J - \mathcal{Q}|_J||_1. \end{aligned}$$

◀

▶ **Lemma 4.** *Given a distribution $\mathcal{P}$ over $\{\pm 1\}^n$ and a size-k set $S \subseteq [n]$ we have that for each $J \subseteq S$*

$$\widehat{\mathcal{P}|_S}(J) = 2^{n-k} \cdot \widehat{\mathcal{P}}(J).$$

Proof. For any distribution $\mathcal{D}$ over $\{\pm 1\}^n$ we have

$$\widehat{\mathcal{D}}(J) = 2^{-n} \cdot \left(\mathop{\mathbb{P}}_{x \sim \mathcal{D}} [\Pi_{i \in J} x_i = 1] - \mathop{\mathbb{P}}_{x \sim \mathcal{D}} [\Pi_{i \in J} x_i = -1] \right).$$

Applying this identity to both sides proves the desired result. ◀

3 The Complexity of Learning Junta Distributions

In this section we prove the following theorem.

▶ **Theorem 1.** *The computational complexity of these two problems is equal, up to* $\mathrm{poly}(2^k, n)$ *factors.*

- *Learning k-parity functions over $\{\pm 1\}^n$ with respect to the uniform distribution with noise rate $\eta = \frac{1}{2} - 2^{-O(k)}$*
- *Learning k-junta distributions over $\{\pm 1\}^n$ with sample access.*

In order to prove Theorem 1, we show mutual reductions between learning junta distributions (LJD) to learning parity functions with noise (LPN). In order to reduce LJD to LPN we need to introduce an auxiliary problem: *learning parity distributions with noise* (LPDN). In Section 3.1 we reduce LPDN to LPN (see Lemma 7). In Section 3.2 we reduce LJD to LPDN (see Lemma 10). Finally, in Section 3.3 we reduce LPN to LJD (see Lemma 17).

3.1 Reducing Noisy Parity Distributions to Parity Functions with Noise

First, we define noisy parity distributions.

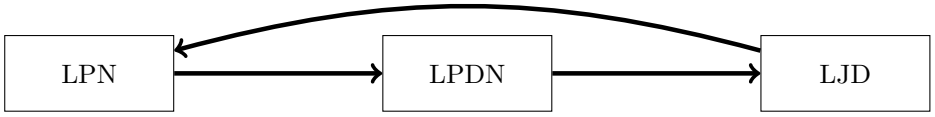

Figure 2 An arrow from A to B means that we reduce problem B to problem A.

▶ **Definition 5** (Noisy parity distribution). *We say that a distribution $\mathcal{D}$ over $\{\pm 1\}^n$ is a noisy parity distribution with relevant variables $J \subseteq [n]$ with noise rate $\eta < 1/2$ if*

$$\widehat{\mathcal{D}}(S) = \frac{1}{2^n} \cdot \begin{cases} 1 & \text{if } S = \varnothing \\ \pm(1-2\eta) & \text{if } S = J \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

▶ **Observation 6.** *A noisy parity distribution with noise rate $\eta = 0$ corresponds to the uniform distribution truncated by a parity function (i.e., $x \sim \mathcal{U}_n$ conditioned on either $\chi_J(x) = \pm 1$). We call such distribution a* parity distribution *and highlight that the PMF of a parity distribution is, indeed, a parity function (up to multiplying it by -1). Moreover, a noisy parity distribution over set J with noise rate $\eta > 0$ coincides with the mixture $(1-2\eta) \cdot \mathcal{P} + 2\eta \cdot \mathcal{U}_n$, where $\mathcal{P}$ is a parity distribution.*

The bulk of this section is spent proving the next lemma, which reduces learning noisy parity distributions to learning parity functions with noise.

▶ **Lemma 7.** *Let $\mathcal{A}$ be an algorithm that learns k-parity functions over $\{\pm 1\}^n$ under the uniform distribution with noise of rate $\eta < 1/2$ and uses $T\left(n, k, (1-2\eta)^{-1}\right)$ time. Then, there exists an algorithm $\mathcal{A}'$ that learns noisy k-parity distributions over $\{\pm 1\}^n$ with noise rate $\eta < 1/2$. Moreover, $\mathcal{A}'$ uses at most $\mathrm{poly}(2^k, n) \cdot T\left(n, k, (1-2\eta)^{-1}\right)$ time.*

Before delving into the proof of Lemma 7, we prove the following lemma.

▶ **Lemma 8.** *Fix $\varepsilon > 0$. Let $\mathcal{D}$ be a noisy parity distribution over $\{\pm 1\}^n$ with relevant variables $J \subseteq [n]$ and noise rate $\eta < 1/2$. Then, there exists an algorithm that takes as input s candidates sets $J_1 \dots J_s$ promised to satisfy $J \in \{J_1 \dots J_s\}$ and, without knowledge of η, estimates each $\widehat{\mathcal{D}}(J_i)$ up additive error $\pm\varepsilon \cdot \widehat{\mathcal{D}}(J)$ with probability $1-\delta$. Moreover, such algorithm finds J, and uses at most $O(s \cdot \varepsilon^{-2}(1-2\eta)^{-2} \log \delta^{-1})$ samples from $\mathcal{D}$ and time.*

Proof. By Definition 5, we have $\widehat{\mathcal{D}}(J) \in \{\pm 2^{-n} \cdot (1-2\eta)\}$. Assume $\widehat{\mathcal{D}}(J) > 0$ without loss of generality. We have $\mathbb{E}_{x\sim\mathcal{D}}\left[2^{-n} \cdot \chi_{J_i}(x)\right] = \widehat{\mathcal{D}}(J_i) = 2^{-n} \cdot (1-2\eta) \cdot [J_i = J]$ and $\mathbb{E}_{x\sim\mathcal{D}}\left[(2^{-n} \cdot \chi_{J_i}(x))^2\right] = 2^{-2n}$. Let $\eta \leqslant \bar{\eta} < 1/2$, $m = O(s \cdot \varepsilon^{-2}(1-2\bar{\eta})^{-2} \log \delta^{-1})$ and given i.i.d. samples $x_1 \dots x_m \sim \mathcal{D}$ define $\widehat{y}_i = \frac{1}{m} \cdot (\chi_{J_i}(x_1) + \dots + \chi_{J_i}(x_m))$ and $\tilde{y}_i = 2^{-n} \cdot \widehat{y}_i$. Standard concentration bounds imply that $\tilde{y}_i$ approximates $\widehat{\mathcal{D}}(J_i)$ up to an additive error $\pm\varepsilon \cdot (1-2\eta) \cdot 2^{-n}$ with probability at least $1 - \delta \cdot 2^{-s}$.

Since η is not known, we perform an exponential search over $\bar{\eta}$ by shrinking $1/2 - \bar{\eta}$ by half at each iteration. Define $Y = \{i \in [s] \mid J_i = J\}$ and $N = [s] \backslash Y$ (recall that Y and N are unknown to the algorithm). At each step, we have that, for $i \in Y$, $\widehat{y}_i = 1 - 2\eta \pm \varepsilon(1-2\bar{\eta})$, and, for $i \in N$, $\widehat{y}_i = \pm\varepsilon(1-2\bar{\eta})$ with high probability. Condition on this event. Then, we can halt whenever there exists an $i \in [s]$ such that $||\widehat{y}_i| - \varepsilon \cdot (1-2\bar{\eta})| \geqslant (1-2\bar{\eta})$ as this guarantees $\bar{\eta} \geqslant \eta$ with high probability. Notice that we halt as soon as $1 - 2\bar{\eta} < \frac{1}{2} \cdot (1-2\eta)$, which gives the desired complexity.

Once we have successfully approximated $\widehat{\mathcal{D}}(J_i)$ for all $i \in [s]$ up to an additive term $\pm\varepsilon \cdot (1-2\eta) \cdot 2^{-n} = \pm\varepsilon \cdot \widehat{\mathcal{D}}(J)$, we must have $J = J_i$ for all J_i satisfying $\tilde{y}_i > \varepsilon(1-2\bar{\eta}) \cdot 2^{-n}$. ◀

Finally, we prove Lemma 7.

Proof of Lemma 7. Let $\mathcal{D}$ be the noisy parity distribution over the set $J \subseteq [n]$ with noise rate $\eta < 1/2$ that we would like to learn. By definition, $\mathcal{D}$ has only two non-zero Fourier coefficients: $\widehat{\mathcal{D}}(\varnothing)$ and $\widehat{\mathcal{D}}(J)$. We can assume $\widehat{\mathcal{D}}(J) > 0$, as the case $\widehat{\mathcal{D}}(J) < 0$ is symmetric. Since $\mathcal{D}$ is a probability distribution, we know that $\widehat{\mathcal{D}}(\varnothing) = 2^{-n}$. We are left to learn $\widehat{\mathcal{D}}(J)$. Once we find the set of relevant variables J, estimating $\widehat{\mathcal{D}}(J)$ is straightforward.

In order to find J, we proceed as follows. Suppose that we correctly guessed one element $j \in J$. Sample $x' \sim \mathcal{D}$, define $x \leftarrow x'$ and flip the j-th bit of x with probability $1/2$. Then, define the label y as $y = -1$ if we flipped x_j and $y = 1$ otherwise. Let $\mathcal{Q}$ be the distribution of (x, y) generated as above and let $\mathcal{Q}_x$ and $\mathcal{Q}_y$ be its marginals. We will prove that

(i) $\mathcal{Q}_x$ is the uniform distribution over $\{\pm 1\}^n$

(ii) Conditioned on x, y is distributed as $\chi_J(x)$ with noise η.

To prove point (i), we compute a generic coefficient $\widehat{\mathcal{Q}}_x(A)$. denote by $e_j \in \{\pm 1\}^n$ the j-th vector of the standard basis.

$$2^n \cdot \widehat{\mathcal{Q}}_x(A) = \sum_{z \in \{\pm 1\}^n} \mathcal{Q}_x(z) \cdot \chi_A(z) = \sum_{z \in \{\pm 1\}^n} \mathcal{D}(z) \cdot \frac{1}{2} \left(\chi_A(z) + \chi_A(z \oplus e_j)\right)$$

then, for $j \in A$ we have $\widehat{\mathcal{Q}}_x(A) = 0$, whereas for $j \notin A$ we have $\widehat{\mathcal{Q}}_x(A) = \widehat{\mathcal{D}}(A)$. Therefore, assuming $j \in J$, $\widehat{\mathcal{Q}}_x(A) = 0$ for all $A \neq \varnothing$.

To prove (ii) we note that, by Observation 6, $\mathcal{D}$ can be written as the mixture $(1 - 2\eta) \cdot \mathcal{P} + 2\eta \cdot \mathcal{U}_n$ where $\mathcal{P}$ is the parity distribution obtained as the uniform distribution over $\{\pm 1\}^n$ conditioned on $\chi_J(\cdot) = 1$. Equivalently, there exists an event $\mathcal{E}$ of probability 2η such that $\mathcal{D}$ conditioned on $\mathcal{E}$ equals $\mathcal{U}_n$ and $\mathcal{D}$ conditioned on $\neg\mathcal{E}$ equals $\mathcal{P}$. If we condition on $\mathcal{E}$, y is independent of x and uniform over $\{\pm 1\}$. If we condition on $\neg\mathcal{E}$, we have $y = \chi_J(x)$. Therefore, y is distributed as $\chi_J(x)$ with noise η.

Now, we feed our samples $(x, y) \sim \mathcal{Q}$ to the algorithm $\mathcal{A}$, which learns the k-parity χ_J in the presence of noise of rate η. By paying a $\mathrm{poly}(\varepsilon^{-1}, n)$ overhead, we can ensure that $\mathcal{A}$ learns χ_J up to ℓ_1-accuracy $\varepsilon \cdot 2^n > 0$ with probability $1 - \mathrm{poly}(n^{-1})$. Recall that $\mathcal{Q}$ depends on our guess $j \in [n]$. We repeat the procedure described above for all $j \in [n]$ and let $(g^j)_{j \in [n]}$ be the functions returned by $\mathcal{A}$. Let $\mathcal{E}$ be the event that, for all $j \in J$, $\mathcal{A}$ is correct. $\mathcal{E}$ happens with probability at least $1 - \mathrm{poly}(n^{-1})$. Condition on $\mathcal{E}$. Then, for each $j \in J$ we have $||g^j - \chi_J||_1 \leqslant \varepsilon \cdot 2^n$.

Now we show how to define a set of candidates $\{J_i\}_{i \in [n]}$ for J. We need a procedure FindCandidate(g) that returns a size-k subset of $[n]$ and such that FindCandidate(g) returns J whenever $||g - \chi_J||_1 \leqslant \varepsilon \cdot 2^n$. Then, we generate our set of candidates $\{J_i\}_{i \in [n]}$ as $J_i =$ FindCandidate(g^i). Lemma 9 describes how to implement FindCandidate. Finally, $(J_i)_{i \in [n]}$ is a collection of at most n candidate sets and we are promised that $J \in (J_i)_{i \in [n]}$, so we run the algorithm from Lemma 8.

▶ **Lemma 9** (Find Parity with Queries). *Assume that we have query access to a function $g : \{\pm 1\}^n \to \{\pm 1\}$ such that $||g - \chi_S||_1 \leqslant \varepsilon \cdot 2^n$ for some $S \subseteq [n]$. Then, we can find S using* $\mathrm{poly}(n)$ *queries and time.*

Proof. We construct our candidate I for S one element at a time. Initialize $I \leftarrow \varnothing$. We iterate over $\ell \in [n]$, and for each ℓ we sample $\mathrm{poly}(n)$ elements $w \sim \mathcal{U}_n$ and observe the distribution of $g(w) \cdot g(w \oplus e_\ell)$. If the majority of those is -1, then add ℓ to I; otherwise continue with the next $\ell \in [n]$.

Assuming $||g - \chi_S||_1 \leqslant \varepsilon \cdot 2^n$, we have $\mathbb{P}_{w\sim\mathcal{U}_n}\left[g(w) \neq \chi_S(w) \text{ or } g(w \oplus e_\ell) \neq \chi_S(w \oplus e_\ell)\right] \leqslant 2\varepsilon$. Thus, by standard concentration bounds we have that the majority of $g(w) \cdot g(w \oplus e_\ell)$ is -1 if and only if $\ell \in J$ with high probability. Conditioned on this event holding for each $\ell \in [n]$, we have $I \subseteq S$ and after ℓ rounds we have $S \cap [\ell] \subseteq I$. The described algorithm uses $\text{poly}(n)$ and queries and, with high probability, returns $I = S$. ◀

◀

3.2 Reducing Junta Distributions to Noisy Parity Distributions

In this section, we reduce the problem of learning junta distributions to that of learning noisy parity distributions. The main result of this section is the next lemma.

▶ **Lemma 10.** *Let $\mathcal{A}$ be an algorithm that learns a noisy k-parity distribution over $\{\pm 1\}^n$ with noise rate $\eta < 1/2$. Suppose that $\mathcal{A}$ uses $T\left(n, k, (1-2\eta)^{-1}\right)$ time. Then, there exists an algorithm $\mathcal{A}'$ that learns a k-junta distribution over $\{\pm 1\}^n$. Moreover, $\mathcal{A}'$ uses at most $\text{poly}(2^k, n) \cdot T\left(n, k, O(2^{k/2})\right)$ time.*

To prove Lemma 10, we mimic the techniques used in [20] to reduce learning junta functions with noise to learning parity functions with noise. Here is a summary of this technique. Let $\mathcal{D}$ be the junta distribution with relevant variables $J^\star$ that we would like to learn. We introduce some noise that, with large enough probability, zeros all Fourier coefficient of $\mathcal{D}$ besides $\widehat{\mathcal{D}}(\varnothing)$ and $\widehat{\mathcal{D}}(J)$ for some $J \subseteq J^\star$. Once all coefficients besides these two are zero, we are left exactly with a noisy parity distribution. As long as we manage to isolate all $J \subseteq J^\star$ such that $\widehat{\mathcal{D}}(J)$ is high enough, we can learn $\mathcal{D}$.

In order to implement this scheme we need the following lemmas that implement noise injection. Throughout the section we identify $\{0,1\}^n \sim 2^{[n]}$ so that Fourier coefficients are (also) indexed by boolean vectors. For $J \subseteq [n]$, denote by $\mathbb{1}_J \in \{0,1\}^n$ the vector such that $\mathbb{1}_J[i] = 1$ iff $i \in J$. We denote by $\oplus$ the sum modulo 2. We overload the notation $\mathcal{U}_n$ so that it represent the uniform distribution both on $\{\pm 1\}^n$ and $\{0,1\}^n$.

▶ **Lemma 11** (Essentially Lemma 1 in the full version of [20])**.** *Let $A \in \{0,1\}^{m\times n}$ and let $f : \{0,1\}^n \to \mathbb{R}$ be any function over the boolean hypercube. Then, for each $x \in \{0,1\}^n$ we have the identity*

$$f_{\mathcal{A}}(x) := \mathop{\mathbb{E}}_{p\sim\mathcal{U}_n}\left[f(x \oplus A^\top p)\right] = \sum_{\substack{a\in\{0,1\}^n \\ Aa = 0^m}} \widehat{f}(a)\chi_a(x).$$

Proof. First, notice that for each $a \in \{0,1\}^n$

$$\chi_a(A^\top p) = (-1)^{\langle a, A^\top p\rangle} = (-1)^{\langle Aa, p\rangle},$$

thus $\mathbb{E}_{p\sim\mathcal{U}_m}\left[\chi_a(A^\top p)\right] = 1$ if $Aa = 0$ and $\mathbb{E}_{p\sim\mathcal{U}_m}\left[\chi_a(A^\top p)\right] = 0$ otherwise.

$$\begin{aligned}
&\mathop{\mathbb{E}}_{p\sim\mathcal{U}_n}\left[f(x \oplus A^\top p)\right] = \\
&\sum_{a\in\{0,1\}^n} \widehat{f}(a) \cdot \mathop{\mathbb{E}}_{p\sim\mathcal{U}_m}\left[\chi_a\left(x \oplus A^\top p\right)\right] = \\
&\sum_{a\in\{0,1\}^n} \widehat{f}(a) \cdot \chi_a(x) \cdot \mathop{\mathbb{E}}_{p\sim\mathcal{U}_m}\left[\chi_a\left(A^\top p\right)\right] = \\
&\sum_{\substack{a\in\{0,1\}^n \\ Aa = 0^m}} \widehat{f}(a)\chi_a(x).
\end{aligned}$$

◀

▶ **Lemma 12.** *Let $\mathcal{D}$ be a k-junta distribution with relevant variables $J^\star \subseteq [n]$ and $m > k$. Sample $A \in \{0,1\}^{m\times n}$ uniformly at random. Define $\mathcal{D}_A(x) := \mathbb{E}_{p\sim\mathcal{U}_n}[\mathcal{D}(x \oplus A^\top p)]$ as in Lemma 11. Fix $J \subset J^\star$. Then, with probability at least $2^{-(m+1)}$ we have*

$$\widehat{\mathcal{D}}_A(I) = \begin{cases} \widehat{\mathcal{D}}(J), & \text{if } I = J \\ 2^{-n}, & \text{if } I = \varnothing \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Namely, $\mathcal{D}_A$ is a noisy parity distribution with relevant variables J.

Proof. By Lemma 11, we have

$$\mathcal{D}_{\mathcal{A}}(x) := \mathop{\mathbb{E}}_{p\sim\mathcal{U}_n}\left[\mathcal{D}(x \oplus A^\top p)\right] = \sum_{\substack{a\in\{0,1\}^n \\ Aa=0^m}} \widehat{\mathcal{D}}(a)\chi_a(x).$$

To realize Equation (2), we must have $A\mathbb{1}_J = 0^m$ and $A\mathbb{1}_I \neq 0^m$ for all $I \subseteq J^\star$ with $I \neq J$. Let $\mathcal{E}$ be such event. It is straightforward to verify that, for any $u, v \in \{0,1\}^m$ and any $M, N \subseteq [n]$ distinct, we have

$$\mathbb{P}\left[A \cdot \mathbb{1}_M = u \,|\, A \cdot \mathbb{1}_N = v\right] = 2^{-m}.$$

Thus, the probability of event $\mathcal{E}$ can be bounded as

$$\begin{aligned} &\mathbb{P}[\mathcal{E}] = \\ &\mathbb{P}\left[A \cdot \mathbb{1}_I \neq 0^m \text{ for all } I \subseteq J^\star \text{ with } I \neq J \,|\, A \cdot \mathbb{1}_J = 0^m\right] \cdot \mathbb{P}\left[A \cdot \mathbb{1}_J = 0^m\right] \geqslant \\ &(1 - 2^k \cdot 2^{-m}) \cdot 2^{-m} \geqslant 2^{-(m+1)}. \end{aligned}$$

◀

▶ **Lemma 13.** *Given a probability mass function $\mathcal{D}$ over $\{0,1\}^n$, and a matrix $A \in \{0,1\}_2^{m\times n}$ the function $\mathcal{D}_A(x) := \mathbb{E}_{p\sim\mathcal{U}_m}[\mathcal{D}(x \oplus A^\top p)]$ is a probability mass function. Moreover, we can simulate a sample from $\mathcal{D}_A$ by $y + A^\top q \sim \mathcal{D}_A$ where $y \sim \mathcal{D}$ and $q \sim \mathcal{U}_m$.*

Proof. For any fixed $p \in \{0,1\}^m$ we have that $x \mapsto \mathcal{D}(x \oplus A^\top p)$ is a probability mass function (PMF), moreover the set of PMFs is convex so $\mathcal{D}_A$ is also PMF. We have

$$\mathop{\mathbb{P}}_{y,z}\left[y \oplus A^\top q = z\right] = \mathop{\mathbb{P}}_{y,z}\left[y = z \oplus A^\top q\right] = \mathop{\mathbb{E}}_{q\sim\mathcal{U}_m}\left[\mathcal{D}(z \oplus A^\top q)\right] = \mathcal{D}_A(z).$$

◀

▶ **Lemma 14.** *Given two distributions $\mathcal{D}, \mathcal{P}$ over $\{\pm 1\}^n$ we have*

$$||\mathcal{D} - \mathcal{P}||_1 \leqslant 2^n \cdot \sqrt{\sum_{a\in\{0,1\}^n} \left(\widehat{\mathcal{D}}(a) - \widehat{\mathcal{P}}(a)\right)^2}.$$

Moreover, if $\mathcal{D}$ and $\mathcal{P}$ are k-junta distributions over the same set of relevant variables, we have

$$||\mathcal{P} - \mathcal{D}||_1 \leqslant 2^k \cdot \sqrt{\sum_{a\in\{0,1\}^k} \left(\widehat{\mathcal{D}|_J}(a) - \widehat{\mathcal{P}|_J}(a)\right)^2}.$$

Proof. First, we have

$$\begin{aligned}||\mathcal{D}-\mathcal{P}||_1 &= \sum_{x\in\{\pm1\}^n} |\mathcal{D}(x)-\mathcal{P}(x)| \\ &\leqslant 2^{n/2}\cdot\sqrt{\sum_{x\in\{\pm1\}^n}(\mathcal{D}(x)-\mathcal{P}(x))^2} \\ &= 2^n\cdot\sqrt{\mathop{\mathbb{E}}_{x\sim\mathcal{U}_n}[(\mathcal{D}(x)-\mathcal{P}(x))^2]} \\ &= 2^n\cdot\sqrt{\sum_{a\in\{0,1\}^n}(\widehat{\mathcal{D}}(a)-\widehat{\mathcal{P}}(a))^2}\end{aligned}$$

where the first inequality holds by Cauchy-Schwarz. Then, using Lemma 3 and Lemma 4 we obtain the second claim. ◀

▶ **Lemma 15** (Estimating a single Fourier coefficient)**.** *There exists an algorithm that takes as input $m = O(\varepsilon^{-2}\log\delta^{-1})$ i.i.d. samples from a distribution $\mathcal{D}$ over $\{\pm1\}^n$, $J\subseteq[n]$ and outputs z such that $z = \widehat{\mathcal{D}}(J)\pm\varepsilon\cdot2^{-n}$ with probability at least $1-\delta$.*

Proof. We have $\widehat{\mathcal{D}}(J) = \mathbb{E}_{x\sim\mathcal{D}}[2^{-n}\cdot\chi_J(x)]$ and $\mathbb{E}_{x\sim\mathcal{D}}\left[(2^{-n}\cdot\chi_J(x))^2\right] = 2^{-2n}$. The desiderata follows by standard concentration bounds applied to the average $z = \frac{1}{m}\cdot2^{-n}\cdot(\chi_J(x_1)+\cdots+\chi_J(x_m))$ where $x_1\ldots x_m$ are i.i.d. samples from $\mathcal{D}$. ◀

Finally, we prove Lemma 10.

Proof of Lemma 10. Here we design the algorithm $\mathcal{A}'$ that learns an arbitrary k-junta distribution $\mathcal{D}$ with set of relevant variables $J^\star$. denote by $\mathcal{J}$ the set of all $J\subseteq[n]$ such that $|\widehat{\mathcal{D}}(J)|\geqslant\varepsilon\cdot2^{-(n+k/2)}$. We have $\mathcal{J}\subseteq2^{J^\star}$, which implies $|\mathcal{J}|\leqslant2^k$. We will show a procedure FindHeavyFourier (Algorithm 1) such that the following holds.

▶ **Lemma 16.** *For each $J\in\mathcal{J}$, FindHeavyFourier (Algorithm 1) returns J with probability $\Omega(2^{-k})$. Moreover, whenever FindHeavyFourier returns (J,z) then both (i) $\widehat{\mathcal{D}}(J)\neq0$, and (ii) $z = \widehat{\mathcal{D}}(J)\pm\varepsilon\cdot2^{-(n+k/2)}$ hold with probability at least $1-\mathrm{poly}(2^{-k})$. Finally, FindHeavyFourier uses $\mathrm{poly}(2^k,\varepsilon^{-1})+T\left(n,k,O(2^{k/2})\right)$ time.*

We defer the proof of Lemma 16 and complete our reduction assuming Lemma 16. Let $\mathcal{I}$ be the set (not a multi-set) of sets J returned by FindHeavyFourier after re-running it $O(k2^{2k})$ times. By coupon collector, $\mathcal{J}\subseteq\mathcal{I}\subseteq2^{J^\star}$ with high probability. Additionally, for all $J\in\mathcal{I}$, z_J satisfies $z_J = \widehat{\mathcal{D}}(J)\pm\varepsilon\cdot2^{-(n+k/2)}$ with high probability (by union bound). Define $\mathcal{Z} := \sum_{J\in\mathcal{I}} z_J\cdot\chi_J$ and notice that $\mathcal{Z}$ is a junta with relevant variables $L := \bigcup\mathcal{I}\subseteq J^\star$. By Lemma 14 and Lemma 4 we have

$$\begin{aligned}||\mathcal{Z}-\mathcal{D}||_1 &= \\ ||\mathcal{Z}|_{J^\star}-\mathcal{D}|_{J^\star}||_1 &\leqslant \\ 2^k\cdot\sqrt{\sum_{J\subseteq J^\star}(\widehat{\mathcal{D}|_{J^\star}}(J)-\widehat{\mathcal{Z}}|_{J^\star}(J))^2} &\leqslant \\ 2^k\cdot\sqrt{\sum_{J\subseteq J^\star}2^{2(n-k)}\cdot(\widehat{\mathcal{D}}(J)-\widehat{\mathcal{Z}}(J))^2} &\leqslant \\ 2^k\cdot\sqrt{2^k\cdot2^{2(n-k)}\cdot\varepsilon^2\cdot2^{-2(n+k/2)}} &\leqslant \varepsilon.\end{aligned}$$

Finally, $\mathcal{Z}$ might not be a probability distribution. We would like to "round" $\mathcal{Z}$ to a probability distribution within ℓ_1-distance $O(\varepsilon)$ from $\mathcal{D}$. Since there exists a probability distribution $\mathcal{D}$ satisfying $||\mathcal{D} - \mathcal{Z}||_1 \leqslant \varepsilon$, then such rounding exists. Here we show how to implement such rounding efficiently.

Since $\mathcal{Z}$ is a junta with relevant variables $L \subseteq J^\star$, then we can simply round $\mathcal{Z}|_L$ and extend it to a rounding of $\mathcal{Z}$ by taking the product distribution with $\mathcal{U}_{n-\ell}$, where $\ell = |L|$. First, for all $x \in \{\pm 1\}^\ell$ such that $\mathcal{Z}|_L(x) < 0$ we set $\mathcal{Z}|_L(x) \leftarrow 0$. Then, we normalize all entries of $\mathcal{Z}|_L$ so that $\sum_{x \in \{\pm 1\}^\ell} \mathcal{Z}|_L(x) = 1$. It is straightforward to verify that both operations at most double the ℓ_1 distance between $\mathcal{Z}$ and $\mathcal{D}$. Hence, after rounding $\mathcal{Z}$ still satisfies $||\mathcal{Z} - \mathcal{D}||_1 = O(\varepsilon)$.

Complexity

We are left to bound the computational complexity of our reduction. We make $\mathrm{poly}(2^k)$ calls to FindHeavyFourier. Each call uses $\mathrm{poly}(2^k, \varepsilon^{-1}) + T\left(n, k, O(2^{k/2})\right)$ time. All other operations combined use at most $\mathrm{poly}(2^k, n)$ time.

Algorithm 1 FindHeavyFourier.

Input: Samples from $\mathcal{D} \in \Delta(\{\pm 1\}^n)$
Output: $J \in \mathcal{J}$ and $y_J = \widehat{\mathcal{D}}(J) \pm \varepsilon \cdot 2^{-(n+k/2)}$

Let $m = k + 1$
Sample a uniformly random matrix $A \in \{0,1\}_2^{m \times n}$
Sample $x_1 \dots x_S \sim \mathcal{D}$ i.i.d.
Sample $q_1 \dots q_S \sim \mathcal{U}_m$ i.i.d.
Let $\chi_\varnothing + y_J \cdot \chi_J$ be the noisy parity distribution returned by $\mathcal{A}(x_1 \oplus A^\top q_1, \dots, x_S \oplus A^\top q_S)$, where $\mathcal{A}$ is PAC if its input has noise of rate $1 - O(2^{-k/2})$.
Run the algorithm from Lemma 15 on $\mathcal{D}$, J and let z be the returned estimate. Set parameters so that $z = \widehat{\mathcal{D}}(J) \pm \frac{\varepsilon}{3} \cdot 2^{-(n+k/2)}$ with probability $\geqslant 1 - \mathrm{poly}(2^{-k})$.
If $|z| \geqslant \frac{2\varepsilon}{3} \cdot 2^{-(n+k/2)}$ **return** J and y_J

In the end, we show that FindHeavyFourier satisfies the properties stated in Lemma 16.

Proof of Lemma 16. Fix $J \in \mathcal{J}$. Define the event $\mathcal{E}_J$ that Equation (2) holds, which happens with probability at least $2^{-m-1} \geqslant 2^{-(k+2)}$ by Lemma 12. Conditioned on $\mathcal{E}_J$, $\mathcal{D}_A = \chi_\varnothing + \widehat{\mathcal{D}}(J) \cdot \chi_J$ is a noisy junta distribution with relevant variables J. Moreover, the samples $x_1 \oplus A^\top q_1 \dots x_S \oplus A^\top q_S$ are distributed according to $\mathcal{D}_\mathcal{A}$ (Lemma 13).

Let $\mathcal{E}_\mathcal{A}$ be the event that $\mathcal{A}$ is correct. Conditioned on $\mathcal{E}_J$, the noise rate of $\mathcal{D}_A$ is $\eta_J = \frac{1}{2} \cdot (1 - 2^n \cdot \widehat{\mathcal{D}}(J))$, thus $J \in \mathcal{J}$ implies $1 - 2\eta_J \geqslant \varepsilon \cdot 2^{-k/2}$. Since, conditioned on $\mathcal{E}_J$, the noise rate of $\mathcal{D}_A$ is appropriate we have $\mathbb{P}\left[\mathcal{E}_\mathcal{A} \,|\, \mathcal{E}_J\right] = \Omega(1)$.

From now on, condition on $\mathcal{E}_\mathcal{A} \cap \mathcal{E}_J$. Suppose that $\mathcal{A}$ returns the noisy parity $\chi_\varnothing + z \cdot \chi_J$. Since $J \in \mathcal{J}$, $|\widehat{\mathcal{D}}(J)| \geqslant \varepsilon \cdot 2^{-(n+k/2)}$ so $z = \widehat{\mathcal{D}}(J) \pm \frac{\varepsilon}{3} \cdot 2^{-(n+k/2)}$ implies $|z| \geqslant \frac{2\varepsilon}{3} \cdot 2^{-(n+k/2)}$. Thus, FindHeavyFourier returns (J, z) correctly with probability $\mathbb{P}\left[\mathcal{E}_\mathcal{A} \cap \mathcal{E}_J\right] = \Omega(2^{-k})$. Finally, points (i) and (ii) hold because of correctness of Lemma 8.

Now we analyze the running time of FindHeavyFourier. $\mathcal{A}$ uses $T\left(n, k, O(2^{k/2})\right)$ time. The algorithm from Lemma 8 uses $\mathrm{poly}(2^k, \varepsilon^{-1})$ time. All other operations combined use $\mathrm{poly}(n, 2^k)$ time. ◀

◀

3.3 Reducing Parity Functions with Noise to Junta Distributions

In this section we prove the next lemma.

▶ **Lemma 17.** *Let $\mathcal{A}$ be an algorithm that learns k-junta distributions over $\{\pm 1\}^n$. Suppose that $\mathcal{A}$ uses $T(n,k)$ time. Then, there exists an algorithm $\mathcal{A}'$ that learns k-parity functions on $\{\pm 1\}^n$ with noise rate $\eta < 1/2$ under the uniform distribution. Moreover, $\mathcal{A}'$ uses at most* $\mathrm{poly}(2^k, (1-2\eta)^{-1}) \cdot T(n,k)$ *time.*

Proof. Let χ_S be the parity function that we would like to learn. We observe samples of the form (x,y) where $x \sim \mathcal{U}_n$ and $y \sim \chi_S(x) \oplus \mathrm{Ber}(\eta)$, where $\mathrm{Ber}(\eta)$ is a Bernoulli random variable of parameter η. Namely, y equals $\chi_S(x)$ flipped with probability η. Equivalently, we can write

$$y = \begin{cases} \mathrm{Ber}(1/2), & \text{with probability } 2\eta \\ \chi_S(x), & \text{with probability } 1-2\eta. \end{cases}$$

We construct a sample z from a junta distribution $\mathcal{D}$ as follows. Given (x,y) sampled as above, if $y = 1$ set $z \leftarrow x$, otherwise sample a new independent copy of (x,y). We have $z \sim \mathcal{D}$ where $\mathcal{D}$ is defined as the convex combination

$$\mathcal{D} := \frac{1-2\eta}{1+2\eta} \cdot \mathcal{P} + \frac{4\eta}{1+2\eta} \cdot \mathcal{U}_n$$

and $\mathcal{P}$ is the uniform distribution over $\{x \in \{\pm 1\}^n \mid \chi_S(x) = 1\}$. In fact, if $y = \mathrm{Ber}(1/2)$ we have $z \sim U_n$ and if $y = \chi_S(x)$ then with probability $1/2$ $y = \chi_S(x) = 1$ and we have $z \sim \mathcal{P}$ and with probability $1/2$ we reject. The probability of *not* rejecting is $1 - \frac{1}{2} \cdot (1-2\eta)$, matching the coefficients above. Notice that the Fourier representation of $\widehat{\mathcal{D}}$ depends on whether: (i) $S \neq \varnothing$ or (ii)$S = \varnothing$. In case (i) we have

$$\widehat{\mathcal{D}}(A) = 2^{-n} \cdot \begin{cases} 1-2\eta, & \text{if } A = S \\ 2\eta, & \text{if } A = \varnothing \\ 0, & \text{otherwise.} \end{cases}$$

In case (ii) we have

$$\widehat{\mathcal{D}}(A) = 2^{-n} \cdot \begin{cases} 1, & \text{if } A = S = \varnothing \\ 0, & \text{otherwise.} \end{cases}$$

The algorithm

Here we describe our algorithm $\mathcal{A}'$. Assume that we have a procedure FindSet$(\bar{\eta})$ that takes as input a guess for the noise rate and returns a candidate $\tilde{S}$ for S. First, we design a procedure to certify $\tilde{S} = S$. Draw $m = \mathrm{poly}(2^k, (1-2\bar{\eta})^{-1})$ samples (x_i, y_i) where $x_i \sim \mathcal{U}_n$ and y_i equals $\chi_S(x_i)$ corrupted with noise of rate η. Define

$$\mathcal{I}_{\tilde{S}} := \frac{1}{m} \cdot \sum_{i \in [m]} \chi_{\tilde{S}}(x_i) \cdot y_i$$

Since $\mathbb{E}\left[(\chi_{\tilde{S}}(x_i) \cdot y_i)^2\right] = 1$, applying standard concentration inequalities we obtain, with high probability, $\mathcal{I}_{\tilde{S}} = \mathbb{E}\left[\chi_{\tilde{S}}(x_i) \cdot y_i\right] \pm o(1-2\bar{\eta})$. Moreover,

$$\mathbb{E}\left[\chi_{\tilde{S}}(x_i) \cdot y_i\right] = \begin{cases} 1-2\eta & \text{if } \tilde{S} = S \neq \varnothing \\ 1 & \text{if } \tilde{S} = S = \varnothing \\ 0 & \text{if } \tilde{S} \neq S. \end{cases}$$

Therefore whenever we observe $\mathcal{I}_{\tilde{S}} > 2 \cdot (1 - 2\bar{\eta})$, then we can certify that $\tilde{S} = S$. Moreover, if we have $1 - 2\bar{\eta} \leqslant \frac{1}{4} \cdot (1 - 2\eta)$ and $\tilde{S} = S$ then $\mathcal{I}_{\tilde{S}} > 2 \cdot (1 - 2\bar{\eta})$ and we can certify $\tilde{S} = S$. Now, we describe a procedure FindSet$(\bar{\eta})$ such that whenever $1 - 2\bar{\eta} \leqslant 1 - 2\eta$ returns a correct candidate $\tilde{S} = S$. These two ingredients together make it possible to perform an exponential search over the parameter $1 - 2\bar{\eta}$ by shrinking it in half at each round. As soon as $1 - 2\bar{\eta} \leqslant \frac{1}{4} \cdot (1 - 2\eta)$, FindSet$(\bar{\eta})$ returns a correct $\tilde{S} = S$ and $\mathcal{I}_{\tilde{S}} > 2(1 - 2\bar{\eta})$. Whenever it happens, we return $\tilde{S}$. Notice that the exponential search ensures that we obtain the right complexity in terms of $1 - 2\eta$.

Here we describe FindSet$(\bar{\eta})$. Recall that we only need FindSet to be correct if $1 - 2\bar{\eta} \leqslant 1 - 2\eta$. First, we run $\mathcal{A}$ with error parameter $\varepsilon \cdot (1 - 2\bar{\eta})$ on the samples $z \sim \mathcal{D}$ generated as described above. $\mathcal{A}$ returns $\mathcal{F}$ such that $||\mathcal{D} - \mathcal{F}||_1 \leqslant \varepsilon \cdot (1 - 2\bar{\eta})$. Recall that $\mathcal{A}$ returns an *evaluator*, so we have *query access* to $\mathcal{F}$.

Assume $S \neq \varnothing$. For each $x \in \{\pm 1\}^n$ we have $\mathcal{D}(x) = 2\eta \cdot 2^{-n} + (1 - 2\eta) \cdot 2^{-n+1} := \alpha$ if $\chi_S(x) = 1$ and $\mathcal{D}(x) = 2\eta \cdot 2^{-n} := \beta$ if $\chi_S(x) = -1$. Notice that $\beta - \alpha = (1 - 2\eta) \cdot 2^{-n+1}$. If we have $S = \varnothing$, then $\mathcal{D}(x) = 2^{-n}$ for all $x \in \{\pm 1\}$. We define a function $f : \{\pm 1\}^n \to \{\pm 1\}$ by rounding $\mathcal{F}$. If $\mathcal{F}(x)$ is closer to β than α then we set $f(x) = -1$, otherwise we set $f(x) = 1$.

A simple charging argument shows that whenever $f(x) \neq \chi_S(x)$ we can charge at least $(\beta - \alpha)/2$ to $||\mathcal{F} - \mathcal{D}||_1$. Therefore, $||\mathcal{F} - \mathcal{D}||_1 \leqslant \varepsilon \cdot (1 - 2\bar{\eta}) \leqslant \varepsilon \cdot (1 - 2\eta)$ implies $||f - \chi_S||_1 \leqslant \varepsilon \cdot 2^n$. Finally, we use the algorithm from Lemma 9 to find $\tilde{S}$.

Unfortunately, this rounding scheme does not work when $S = \varnothing$ because $\mathcal{D}(\cdot)$ is constant. To obviate this issue, we modify FindSet so that it returns two set at each round: $\tilde{S}$ and $\varnothing$. In this way, the run our certifying procedure both on $\tilde{S}$ and $\varnothing$. If either $S = \tilde{S}$ or $S = \varnothing$, the certifying procedures detects that $\mathcal{I}_{\tilde{S}}$ (respectively $\mathcal{I}_{\varnothing}$) is larger than $2 \cdot (1 - 2\bar{\eta})$.

The complexity of each step of our reduction is $\mathrm{poly}(n, (1 - 2\bar{\eta})^{-1}) \cdot T(n, k)$. Indeed, boosting the accuracy of $\mathcal{A}$ to $\varepsilon \cdot (1 - 2\bar{\eta})$ (in ℓ_1) costs $\mathrm{poly}(\varepsilon^{-1}, (1 - 2\bar{\eta})^{-1})$, the algorithm in Lemma 9 runs in time $\mathrm{poly}(n)$ and all other operations combined cost at most $\mathrm{poly}(n)$. Finally, our reduction uses time $\mathrm{poly}(n, (1 - 2\eta)^{-1}) \cdot T(n, k)$ thanks to our exponential search. ◀

4 Algorithm

A key ingredient of our algorithm is a tolerant tester for distribution identity that is efficient in the high-confidence regime. First, we develop the tester and then we apply it to learning junta distributions.

4.1 High-Confidence Tolerant Identity Tester

The goal of this section is to develop a sample-efficient tolerant identity tester for the high-probability regime. In particular, boosting off-the-shelf identity testers to failure probability δ incurs a multiplicative overhead of $\log \delta^{-1}$ on the number of samples. We improve over standard failure-probability boosting and obtain an algorithm that only pays an *additive* overhead of $\log \delta^{-1}$. To this end, we mimic the techniques used in [18].

▶ **Definition 18** (Tolerant Identity Tester). *We say that an algorithm T is an ℓ_1 $(\varepsilon_1, \varepsilon_2)$-tolerant identity tester for the distribution $\mathcal{P}$ with sample complexity m and failure probability δ if the following holds.*

For any distribution $\mathcal{D}$, let $x_1 \dots x_m$ be i.i.d. samples from $\mathcal{D}$.

- *If $||\mathcal{D} - \mathcal{P}||_1 \leqslant \varepsilon_1$ then $T(x_1 \dots x_m)$ returns "close" with probability at least $1 - \delta$;*
- *If $||\mathcal{D} - \mathcal{P}||_1 \geqslant \varepsilon_2$ then $T(x_1 \dots x_m)$ returns "far" with probability at least $1 - \delta$.*

Recall the "Bernstein" version of the standard bounded differences (McDiarmid) inequality.

▶ **Lemma 19** (Bernstein version of McDiarmid's inequality [38]). *Let $Y_1, \ldots, Y_m$ be independent random variables taking values in the set $\mathcal{Y}$. Let $f : \mathcal{Y}^m \to \mathbb{R}$ be a function of $y_1, \ldots, y_m$ so that for every $j \in [m]$ and $y_1, \ldots, y_m, y'_j \in \mathcal{Y}$, we have that:*

$$|f(y_1, \ldots, y_j, \ldots, y_m) - f(y_1, \ldots, y'_j, \ldots, y_m)| \leqslant B \ .$$

Then, we have:

$$\mathbb{P}\left[f(Y_1, \ldots, Y_m) - \mathbb{E}[f] \geqslant z\right] \leqslant \exp\left(\frac{-2z^2}{mB^2}\right) \ .$$

Now we can prove the main result of this section.

▶ **Lemma 20** (High-confidence tolerant identity tester). *For any $\alpha, \varepsilon, \delta > 0$ and any distribution $\mathcal{D}$ with support $[s]$, there exists an ℓ_1 $(\alpha, \alpha + \varepsilon)$-tolerant identity tester for $\mathcal{D}$ with sample complexity $O(\varepsilon^{-2}(s + \log \delta^{-1}))$.*

Proof. We define the same estimator used in [18] and we analyze its mean and concentration similarly to [18]. Let $m = O(\varepsilon^{-2}(s + \log \delta^{-1}))$ and $Y_1 \ldots Y_m$ be i.i.d. samples from $\mathcal{D}$. Let $X_1 \ldots X_s$ be the histograms induced by $Y_1 \ldots Y_m$, namely X_i is the number of $j \in [m]$ such that $Y_j = i$. Given a distribution $\mathcal{P}$ supported on $[s]$, define

$$S_{\mathcal{P}}(X) := \left|\left|\frac{X}{m} - \mathcal{P}\right|\right|_1 = \sum_{i \in [s]} \left|\frac{X_i}{m} - \mathcal{P}(i)\right| .$$

Let $\mathcal{P}$ be a probability distribution supported on $[s]$. In the following we prove that with probability $1 - \delta$ we have

$$S_{\mathcal{P}}(X) = ||\mathcal{P} - \mathcal{D}||_1 \pm \frac{\varepsilon}{2}.$$

Then, to obtain the desired tolerant tester, we return "close" if $S_{\mathcal{P}} \leqslant \alpha + \varepsilon/2$ and return "far" otherwise. By triangle inequality, we have

$$\left|\left|\frac{X}{m} - \mathcal{P}\right|\right|_1 = ||\mathcal{D} - \mathcal{P}||_1 \pm \left|\left|\frac{X}{m} - \mathcal{D}\right|\right|_1 .$$

So, we are left proving that $\mathbb{P}\left[S_{\mathcal{D}}(X) > \varepsilon/2\right] \leqslant \delta$. First, we bound the expectation of $S_{\mathcal{D}}(X)$.

$$\begin{aligned}
\mathbb{E}\left[S_{\mathcal{D}}(X)\right] &= \sum_{i \in [s]} \mathbb{E}\left[\left|\frac{X_i}{m} - \mathcal{D}(i)\right|\right] \\
&\leqslant \sum_{i \in [s]} \sqrt{\mathrm{Var}\left[\frac{X_i}{m} - \mathcal{D}(i)\right]} \\
&\leqslant \sum_{i \in [s]} \sqrt{\frac{m \cdot \mathcal{D}(i)}{m^2}} \\
&\leqslant \sum_{i \in [s]} \sqrt{\frac{m \cdot \frac{1}{s}}{m^2}} \qquad (3) \\
&= \sqrt{\frac{s}{m}} \leqslant \frac{\varepsilon}{4}.
\end{aligned}$$

Equation (3) holds because $\mathcal{D} \mapsto \sum_{i\in[s]} \sqrt{\mathcal{D}(i)}$ is a concave function of $\mathcal{D}$ and so it is minimized when $\mathcal{D}(i) = 1/s$ for all $i \in [s]$.

Then, we show that $S_{\mathcal{D}}(X)$ concentrates around its expectation. Notice that the map $(Y_1 \dots Y_m) \mapsto S_{\mathcal{D}}(X)$ has Lipschitz constant $B = 1/m$, thus applying Lemma 19 we obtain

$$\mathbb{P}\left[|S_{\mathcal{D}}(X) - \mathbb{E}\left[S_{\mathcal{D}}(X)\right]| > \frac{\varepsilon}{4}\right] \leqslant \exp\left(\frac{-2\cdot\left(\frac{\varepsilon}{4}\right)^2}{m\cdot B^2}\right) = \exp\left(\Omega\left(\varepsilon^2\cdot m\right)\right) \leqslant \delta.$$

Therefore, $\mathbb{P}\left[S_{\mathcal{D}}(X) > \varepsilon/2\right] \leqslant \delta$. ◀

4.2 An Algorithm for Learning Junta Distributions

In this section we prove Theorem 2, restated below.

▶ **Theorem 2.** *There exists an algorithm that properly learns k-junta distributions up to TV distance ε using $O(\frac{k}{\varepsilon^2}(2^k + \log n))$ samples and $O(\min(2^n, n^k)\cdot\frac{k}{\varepsilon^2}(2^k+\log n))$ running time.*

First, we prove a few technical lemmas.

▶ **Lemma 21** (Learning Fourier coefficients)**.** *There exists an algorithm that takes as input $O(\varepsilon^{-2}(k+\log\delta^{-1}))$ i.i.d. samples from a distribution $\mathcal{D}$ over $\{\pm1\}^k$ and outputs a distribution $\mathcal{Q}$ over $\{\pm1\}^k$ such that, for each $J \subseteq [k]$*

$$\widehat{\mathcal{Q}}(J) = \widehat{\mathcal{D}}(J) \pm \frac{\varepsilon}{2^k}.$$

Such an algorithm succeeds with probability at least $1-\delta$ and runs in time $\mathrm{poly}(2^k, \varepsilon, \log\delta^{-1})$.

Proof. Let $m = O(\varepsilon^{-2}2^k(k+\log\delta^{-1}))$ and $x_1 \dots x_m$ be i.i.d. samples from $\mathcal{D}$. For each $J \subseteq \{\pm1\}^k$, define $\widehat{\mathcal{P}}(J)$ as the J Fourier coefficient of the empirical distribution of the samples, namely

$$\widehat{\mathcal{P}}(J) := 2^{-k}\cdot\sum_{i\in[m]}\frac{1}{m}\cdot\chi_J(x_i).$$

For each $J \subseteq [k]$, we have $\mathbb{E}\left[\widehat{\mathcal{P}}(J)\right] = \widehat{\mathcal{D}}(J)$ and

$$\mathrm{Var}\left[\widehat{\mathcal{P}}(J)\right] = \frac{1}{m^2}\sum_{i\in[m]}\mathrm{Var}\left[2^{-k}\cdot\chi_J(x_i)\right] = \frac{2^{-2k}}{m}\cdot\mathrm{Var}\left[\chi_S(x_1)\right] \leqslant \frac{2^{-2k}}{m}.$$

Thus, by standard concentration bounds we have that, for each $J \subset [k]$,

$$\mathbb{P}\left[\left|\widehat{\mathcal{P}}(J) - \widehat{\mathcal{D}}(J)\right| > \varepsilon\cdot 2^{-k}\right] \leqslant 2^{-k}\cdot\delta.$$

By taking a union bound over all $J \subseteq [k]$ we obtain:

$$\mathbb{P}\left[\max_{J\subseteq[k]}\left|\widehat{\mathcal{P}}(J) - \widehat{\mathcal{D}}(J)\right| > \varepsilon\cdot 2^{-k}\right] \leqslant \delta. \tag{4}$$

The values $\widehat{\mathcal{P}}$ satisfy the desiderata, however we might have that $\mathcal{P}$, the inverse Fourier transform of $\widehat{\mathcal{P}}$, is not a probability distribution. On the other hand, conditioning on the previous step succeeding, we know that there exists a probability distribution $\mathcal{Q}$ such that $\left\|\widehat{\mathcal{P}} - \widehat{\mathcal{D}}\right\|_\infty \leqslant \varepsilon\cdot 2^{-k}$. We write a linear program that searches for a distribution $\mathcal{Q}$ such that

$$\left\|\widehat{\mathcal{P}} - \widehat{\mathcal{Q}}\right\|_\infty \leqslant \varepsilon \cdot 2^{-k}$$
$$\sum_{x\in\{\pm1\}^k} \mathcal{Q} = 1$$
$$\mathcal{Q}(x) \geqslant 0 \quad \text{for all} \quad x \in \{\pm1\}^k.$$

Since each coefficient of $\widehat{\mathcal{Q}}$ is a linear function of $\mathcal{Q}$ we can solve the above LP in time poly(2^k). Triangle inequality and Equation (4) imply our desiderata. ◀

▶ **Lemma 22.** *Let $\mathcal{Q}$ and $\mathcal{D}$ be distributions over $\{\pm1\}^k$ with $\max_{a\in\{\pm1\}^k} |\widehat{\mathcal{Q}}(a) - \widehat{\mathcal{D}}(a)| \leqslant \varepsilon \cdot 2^{-3k/2}$, then $||\mathcal{Q} - \mathcal{D}||_1 \leqslant \varepsilon$.*

Proof.

$$\begin{aligned}||\mathcal{Q} - \mathcal{D}||_1^2 &\leqslant 2^k \cdot ||\mathcal{Q} - \mathcal{D}||_2^2 \\ &= 2^{2k} \cdot \sum_{a\in\{0,1\}^k} (\widehat{\mathcal{Q}}(a) - \widehat{\mathcal{D}}(a))^2 \\ &\leqslant 2^{2k} \cdot \sum_{a\in\{0,1\}^k} \frac{\varepsilon^2}{2^{3k}} \\ &\leqslant \varepsilon^2.\end{aligned}$$

◀

▶ **Observation 23.** *Combining Lemma 22 and Lemma 21 we obtain the following. The algorithm from Lemma 21 run with parameter $\varepsilon' = \varepsilon \cdot 2^{-k/2}$ learns $\mathcal{D}$ in ℓ_1 norm up to additive error ε.*

Analysis of Algorithm 2

Here we prove Theorem 2. Namely, that Algorithm 2 correctly returns an approximation of $\mathcal{P}$ with high probability, and that it uses at most $O\left(\frac{k}{\varepsilon^2}(2^k + \log n)\right)$ samples and $O(\min(2^n, n^k) \cdot \frac{k}{\varepsilon^2}(2^k + \log n))$ time.

Denote with $J^\star$ the set of relevant variables of $\mathcal{P}$. Let $\mathcal{E}$ be the event that LearnFourier and Tester never fail throughout the execution of Algorithm 2.

▶ **Lemma 24.** $\mathbb{P}[\mathcal{E}] = o(1)$.

Proof. The failure probability of Tester is set to n^{-2k} and we make at most $k \cdot n^k$ calls to it. The failure probability of LearnFourier is set to k^{-2} and we make at most $2k$ calls to it. ◀

▶ **Lemma 25.** *Conditioned on $\mathcal{E}$, we have $N \subseteq J^\star$ throughout the execution of Algorithm 2.*

Proof. We initialize $N \leftarrow \varnothing$ and the only step where we add elements to N is line 14. Conditioned on $\mathcal{E}$ we have $|\widehat{\mathcal{R}}(A) - \widehat{\mathcal{P}}|_S| \leqslant \varepsilon \cdot 2^{-3k/2}$, thus $|\mathcal{R}(A)| > \varepsilon \cdot 2^{-3k/2}$ implies $\widehat{\mathcal{P}}|_S(A) \neq 0$ and, in turn, $\widehat{\mathcal{P}}(A) \neq 0$. Hence, $A \subseteq J^\star$. ◀

▶ **Lemma 26.** *Conditioned on $\mathcal{E}$, let $\mathcal{Q}$ and N be defined as in line 15. Then,*

$$|\widehat{\mathcal{P}|_N}(A) - \widehat{\mathcal{Q}|_N}(A)| \leqslant \varepsilon \cdot 2^{-(|N|+k/2)}.$$

Proof. Notice that $\mathcal{P}|_N$ is supported on $\{\pm1\}^{|N|}$ and apply Lemma 21 with error parameter $\varepsilon' = \varepsilon \cdot 2^{-k/2}$. ◀

Algorithm 2 Junta Distribution Learner.

Input: $m = O\left(\frac{k}{\varepsilon^2}(2^k + \log n)\right)$ i.i.d. samples from a distribution $\mathcal{P}$ over $\{\pm 1\}^n$. Moreover, $\mathcal{P}$ is promised to be a junta distribution over an unknown size-k set $J^\star \subseteq [n]$.;
Output: The explicit description of a distribution $\mathcal{Q}$ over $\{\pm 1\}^n$ such that $||\mathcal{Q} - \mathcal{P}||_1 = O(\varepsilon)$;

Let TESTER be the $(2\varepsilon, 3\varepsilon)$-tolerant identity tester of Lemma 20 with failure probability n^{-2k}
Let LEARNFOURIER be the algorithm described in Lemma 21 with error parameter $\varepsilon' = \varepsilon \cdot 2^{-k/2}$ and failure probability k^{-2}
Initialize $N \leftarrow \varnothing$
Initialize $\mathcal{Q} \leftarrow \mathcal{U}_n$
for $i \in [k]$ **do**
  if *For all* $S \subseteq [n]$ *with* $S \supseteq N$ *and* $|S| = k$, TESTER$(\mathcal{P}|_S, \mathcal{Q}|_S)$ *returns "close"* **then**
    return $\mathcal{Q}$
  end
  Let S be such that TESTER$(\mathcal{P}|_S, \mathcal{Q}|_S)$ returns "far";
  $\mathcal{R} \leftarrow$ LEARNFOURIER$(\mathcal{P}|_S)$;
  For each $A \subseteq S$ with $|\widehat{\mathcal{R}}(A)| > \varepsilon \cdot 2^{-3k/2}$ do $N \leftarrow N \cup A$;
  Let $\mathcal{Q}$ be the junta distr. on variables N s.t. $\mathcal{Q}|_N \leftarrow$ LEARNFOURIER$(\mathcal{P}|_N)$;
end
return $\mathcal{Q}$

▶ **Lemma 27.** *Conditioned on $\mathcal{E}$, the size of N increases by at least one at line 14.*

Proof. Since TESTER$(\mathcal{P}|_S, \mathcal{Q}|_S)$ returned "far", we have that $||\mathcal{P}|_S - \mathcal{Q}|_S||_1 > 2\varepsilon$. Thus, by Lemma 22, there exists $A \subseteq S$ such that $|\widehat{\mathcal{P}|_S}(A) - \widehat{\mathcal{Q}|_S}(A)| > 2\varepsilon \cdot 2^{-3k/2}$. Moreover, for all $A \subseteq N$ we have

$$|\widehat{\mathcal{P}|_S}(A) - \widehat{\mathcal{Q}|_S}(A)| = 2^{|N|-k} \cdot |\widehat{\mathcal{P}|_N}(A) - \widehat{\mathcal{Q}|_N}(A)| \leqslant \varepsilon \cdot 2^{-3k/2}$$

where the first equality holds by Lemma 4 and the second inequality holds by Lemma 26. Thus, there must exist have $A \nsubseteq N$ such that

$$|\widehat{\mathcal{P}|_S}(A)| = |\widehat{\mathcal{P}|_S}(A) - \widehat{\mathcal{Q}|_S}(A)| > 2\varepsilon \cdot 2^{-3k/2}$$

where the first equality holds because $\mathcal{Q}$ is a junta with relevant variables N and $S \nsubseteq N$. Finally, since LEARNFOURIER$(\mathcal{P}|_S)$ returns $\mathcal{R}$ such that $\max_{B \subseteq S} |\widehat{\mathcal{P}|_S}(B) - \widehat{\mathcal{R}}(B)| \leqslant \varepsilon \cdot 2^{-3k/2}$ then we have $\widehat{\mathcal{R}}(A) > \varepsilon \cdot 2^{-3k/2}$. So we increase the size of N at line 14. ◀

▶ **Lemma 28.** *Conditioned on $\mathcal{E}$, Algorithm 2 returns $\mathcal{Q}$ such that $||\mathcal{Q} - \mathcal{P}||_1 \leqslant 2\varepsilon$.*

Proof. If $\mathcal{Q}$ is returned on line 10, then for $S = J^\star \supseteq N$ we obtain

$$|\mathcal{P} - \mathcal{Q}| = |\mathcal{P}|_{J^\star} - \mathcal{Q}|_{J^\star}| \leqslant 2\varepsilon$$

where the first inequality holds because $\mathcal{P}$ is a junta wrt $J^\star$ and $\mathcal{Q}$ is a junta wrt $N \subseteq J^\star$.

Suppose that $\mathcal{Q}$ is returned on line 17. Then, when executing line 17 we have $N \subseteq J^\star$ and $|N| = k$, thus $N = J^\star$. Finally, Lemma 26 applied on the last iteration (i.e., when $N = J^\star$), together with Lemma 22 and Lemma 3 imply

$$|\mathcal{P} - \mathcal{Q}| = |\mathcal{P}|_N - \mathcal{Q}|_N| \leqslant \varepsilon. \qquad \blacktriangleleft$$

References

1 Maryam Aliakbarpour, Eric Blais, and Ronitt Rubinfeld. Learning and testing junta distributions. In *Conference on Learning Theory*, pages 19–46. PMLR, 2016. URL: http://proceedings.mlr.press/v49/aliakbarpour16.html.

2 Josh Alman. An illuminating algorithm for the light bulb problem. *arXiv preprint arXiv:1810.06740*, 2018. arXiv:1810.06740.

3 Jan Arpe and Rüdiger Reischuk. Learning juntas in the presence of noise. *Theoretical computer science*, 384(1):2–21, 2007. doi:10.1016/J.TCS.2007.05.014.

4 Vikraman Arvind, Johannes Köbler, and Wolfgang Lindner. Parameterized learnability of juntas. *Theoretical computer science*, 410(47-49):4928–4936, 2009. doi:10.1016/J.TCS.2009.07.003.

5 Zongbo Bao and Penghui Yao. On testing and learning quantum junta channels. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 1064–1094. PMLR, 2023. URL: https://proceedings.mlr.press/v195/bao23b.html.

6 Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35:21750–21764, 2022.

7 Eric Blais. Improved bounds for testing juntas. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 317–330. Springer, 2008. doi:10.1007/978-3-540-85363-3_26.

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

9 Avrim Blum. Open problems-learning a function of r relevant variables. *Lecture Notes in Computer Science*, 2777:731–733, 2003.

10 Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003. doi:10.1145/792538.792543.

11 Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2):245–271, 1997. doi:10.1016/S0004-3702(97)00063-5.

12 Nader H Bshouty and Areej Costa. Exact learning of juntas from membership queries. *Theoretical Computer Science*, 742:82–97, 2018. doi:10.1016/J.TCS.2017.12.032.

13 Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & electrical engineering*, 40(1):16–28, 2014. doi:10.1016/J.COMPELECENG.2013.11.024.

14 Thomas Chen, Shivam Nadimpalli, and Henry Yuen. Testing and learning quantum juntas nearly optimally. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1163–1185. SIAM, 2023. doi:10.1137/1.9781611977554.CH43.

15 Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. Learning and testing junta distributions with sub cube conditioning. In *Conference on Learning Theory*, pages 1060–1113. PMLR, 2021. URL: http://proceedings.mlr.press/v134/chen21b.html.

16 Anindya De, Huan Li, Shivam Nadimpalli, and Rocco A Servedio. Detecting low-degree truncation. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1027–1038, 2024. doi:10.1145/3618260.3649633.

17 Ilias Diakonikolas. Learning structured distributions. *Handbook of Big Data*, 267:10–1201, 2016.

18 Ilias Diakonikolas, Themis Gouleakis, John Peebles, and Eric Price. Sample-optimal identity testing with high probability. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl – Leibniz Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.41.

19 Francisco Escudero-Gutiérrez. Learning junta distributions and quantum junta states, and qac 0 circuits. *arXiv preprint arXiv:2410.15822*, 2024. doi:10.48550/arXiv.2410.15822.

20 Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 563–574. IEEE, 2006. doi:10.1109/FOCS.2006.51.

21 Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. On agnostic learning of parities, monomials, and halfspaces. *SIAM Journal on Computing*, 39(2):606–645, 2009. doi:10.1137/070684914.

22 Vitaly Feldman, Pravesh Kothari, and Jan Vondrák. Representation, approximation and learning of submodular functions using low-rank decision trees. In *Conference on Learning Theory*, pages 711–740. PMLR, 2013. URL: http://proceedings.mlr.press/v30/Feldman13.html.

23 Surbhi Goel, Sushrut Karmalkar, and Adam Klivans. Time/accuracy tradeoffs for learning a relu with respect to gaussian marginals. *Advances in neural information processing systems*, 32, 2019.

24 William He and Shivam Nadimpalli. Testing junta truncation. *arXiv preprint arXiv:2308.13992*, 2023. doi:10.48550/arXiv.2308.13992.

25 Adam Tauman Kalai, Adam R Klivans, Yishay Mansour, and Rocco A Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008. doi:10.1137/060649057.

26 Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998. doi:10.1145/293347.293351.

27 Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E Schapire, and Linda Sellie. On the learnability of discrete distributions. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 273–282, 1994. doi:10.1145/195058.195155.

28 Eike Kiltz, Krzysztof Pietrzak, Daniele Venturi, David Cash, and Abhishek Jain. Efficient authentication from hard learning problems. *Journal of Cryptology*, 30(4):1238–1275, 2017. doi:10.1007/S00145-016-9247-3.

29 Adam Klivans and Pravesh Kothari. Embedding hard learning problems into gaussian space. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.793.

30 Caleb Koch, Carmen Strassle, and Li-Yang Tan. Superpolynomial lower bounds for decision tree learning and testing. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1962–1994. SIAM, 2023. doi:10.1137/1.9781611977554.CH75.

31 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM (JACM)*, 40(3):607–620, 1993. doi:10.1145/174130.174138.

32 Elchanan Mossel, Ryan O'Donnell, and Rocco A Servedio. Learning functions of k relevant variables. *Journal of Computer and System Sciences*, 69(3):421–434, 2004. doi:10.1016/J.JCSS.2004.04.002.

33 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009. doi:10.1145/1568318.1568324.

34 Mert Sağlam. Near log-convexity of measured heat in (discrete) time and consequences. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 967–978. IEEE, 2018.

35 Rocco A Servedio, Li-Yang Tan, and John Wright. Adaptivity helps for testing juntas. In *30th Conference on Computational Complexity (CCC 2015)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CCC.2015.264.

36 Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *Journal of the ACM (JACM)*, 62(2):1–45, 2015. doi:10.1145/2728167.

37 Gregory Valiant and Paul Valiant. Estimating the unseen: improved estimators for entropy and other properties. *Journal of the ACM (JACM)*, 64(6):1–41, 2017. doi:10.1145/3125643.

38 Yiming Ying. Mcdiarmid's inequalities of bernstein and bennett forms. *City University of Hong Kong*, page 318, 2004.