Electronic Colloquium on Computational Complexity, Report No. 121 (2024)

Approximating the Number of Relevant Variables in a Parity Implies Proper Learning

Nader H. Bshouty
Department of Computer Science, Technion, Israel.
bshouty@cs.technion.ac.il

George Haddad
Department of Computer Science, Technion, Israel.
haddadgeorge@campus.technion.ac.il

Abstract

Consider the model where we can access a parity function through random uniform labeled examples in the presence of random classification noise. In this paper, we show that approximating the number of relevant variables in the parity function is as hard as properly learning parities.

More specifically, let $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$, where $\gamma(x) \geq x$, be any strictly increasing function. In our first result, we show that from any polynomial-time algorithm that returns a γ-approximation, D (i.e., $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$), of the number of relevant variables $d(f)$ for any parity f, we can, in polynomial time, construct a solution to the long-standing open problem of polynomial-time learning $k(n)$-sparse parities (parities with $k(n) \leq n$ relevant variables), where $k(n) = \omega_n(1)$.

In our second result, we show that from any $T(n)$-time algorithm that, for any parity f, returns a γ-approximation of the number of relevant variables $d(f)$ of f, we can, in polynomial time, construct a $poly(\Gamma(n))T(\Gamma(n)^2)$-time algorithm that properly learns parities, where $\Gamma(x) = \gamma(\gamma(x))$.

If $T(\Gamma(n)^2) = \exp(o(n/\log n))$, this would resolve another long-standing open problem of properly learning parities in the presence of random classification noise in time $\exp(o(n/\log n))$.

ISSN 1433-8092

1 Introduction

The problem of PAC learning parity, with and without noise, and approximating its sparsity has been extensively studied in the literature. See [2, 4, 5, 6, 7, 8, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 32, 33, 35] and references therein.

In properly learning parities under the uniform distribution, the learner can observe labeled examples $\{(a_i, b_i)\}_i$, where $b_i = f(a_i)$, a_i are drawn independently from the uniform distribution, and f is the target parity. The goal is to return the target parity function f exactly.

In the random classification noise model with noise rate η, [1], each label b_i is independently flipped (misclassified) with probability η. The problem of learning parities with noise (LPN) is known to be computationally challenging. Some evidence of its hardness comes from the fact that it cannot be learned efficiently in the so-called statistical query (SQ) model [27] under the uniform distribution [9, 12]. LPN serves as the foundation for several cryptographic constructions, largely because its hardness in the presence of noise is assumed. See for example [10, 31].

While PAC learning of parities (and thus determining its sparsity) under the uniform distribution can be accomplished in polynomial time using Gaussian elimination, addressing this problem in the presence of random classification noise remains one of the most long-standing challenges in learning theory. The only known algorithm is that of Blum et al.[11], which runs in time $2^{O(n/\log n)}$, requires $2^{O(n/\log n)}$ labeled examples, and handles only a constant noise rate. This algorithm holds the record as the best-known solution for this problem. Finding a $2^{o(n/\log n)}$-time learning algorithm for parities or proving the impossibility of such an algorithm remains a significant and unresolved challenge.

When the number of relevant variables[1] k of the parity function f is known (f is called k-sparse parity), all the algorithms proposed in the literature run in time n^{ck} for some constant $c < 1$, [5, 7, 22, 33, 36]. Finding a polynomial-time algorithm for k-sparse parities for some $k = \omega(1)$, or proving the impossibility of such an algorithm, is another significant and unresolved challenge.

In a related vein, another challenging problem is determining or approximating the sparsity of the parity function, i.e., the number of relevant variables in the target function. This problem was studied in the PAC-learning model [34] under specific[2] distributions [2, 3, 4, 7, 16, 17, 18, 19, 20, 30, 35].

For the problem of determining the sparsity under any distribution and without noise, Downey et al. [18] and Bhattacharyya et al. [4] show that determining the sparsity k of parities is $W[1]$-hard. Bhattacharyya et al. [7] show that the time complexity is $\min(2^{\Theta(n)}, n^{\Theta(k)})$, assuming 3-SAT has no $2^{o(n)}$-time algorithm. For the problem of approximating the sparsity, Dumer et al. [19] showed that if RP$\neq$NP, then it is hard to approximate the sparsity within some constant factor $\gamma > 1$. See also [2, 16, 17, 30]. When the distribution is uniform, there is a polynomial-time algorithm that uses $O(n)$ labeled examples and learns parities using Gaussian elimination, thereby determining their sparsity.

In this paper, we pose the question: Can we approximate the sparsity of the parity function in polynomial time using random uniform labeled examples in the presence of random classification noise? We show that approximating the number of relevant variables in the parity function is as hard as properly learning parities.

[1] A variable is *relevant* in f if f depends on that variable.

[2] Some of the problems are introduced as follows: Given a matrix $M \in F_2^{m \times n}$, a vector $b \in \{0,1\}^m$, and an integer k. Deciding if there exists a weight k vector $x \in \{0,1\}^n$ such that $Mx = b$. This is equivalent to the decision problem when the distribution is uniform over the rows of M.

We first show the following.

Theorem 1. *Let $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$ be any strictly increasing function, where $\gamma(x) \geq x$. Consider a polynomial-time algorithm that, for any parity f, uses random uniform labeled examples of f in the presence of random classification noise and returns an integer D such that[3] $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$, where $d(f)$ is the number of relevant variables in f. One can, in polynomial time, construct an algorithm that runs in polynomial time, uses random uniform labeled examples in the presence of random classification noise, and learns $k(n)$-sparse parities for some[4] $k(n) = \omega_n(1)$.*

This would solve the long-standing open problem of polynomial-time learning k-sparse parities for some $k = \omega_n(1)$.

We then show that

Theorem 2. *From any $T(n)$-time algorithm that, for any parity $f : \{0,1\}^n \to \{0,1\}$, uses $Q(n)$ random uniform labeled examples of f in the presence of random classification noise and returns a γ-approximation of the number of relevant variables $d(f)$ of f, one can, in polynomial time, construct a poly$(\Gamma(n))T(\Gamma(n)^2)$-time algorithm that uses poly$(\Gamma(n))Q(\Gamma(n)^2)$ random uniform labeled examples in the presence of random classification noise and properly learns parities, where $\Gamma(x) = \gamma(\gamma(x))$.*

If $T(\Gamma(n)^2) = \exp(o(n/\log n))$, this would resolve another long-standing open problem of proper learning parities in the presence of random classification noise in time $\exp(o(n/\log n))$. This is applicable, for example, for any poly$(\cdot)$-approximation and $\exp(n^{1/c})$-time algorithm for some sufficiently large constant c. As well as to quasi-poly$(\cdot)$-approximation and $\exp(\exp((\log n)^{1/c}))$-time algorithm for some sufficiently large constant c.

In this paper, while the above discussions and the technique section have been primarily focused on parities, that is, linear functions over the binary field $\mathbb{F}_2$, the results we present in this paper are not limited to this specific case. We generalize our result to encompass any linear function over any finite field. This extension allows our results to be applicable to a broader range of linear systems beyond the binary paradigm, effectively widening their relevance in coding theory and cryptography.

1.1 Our Technique

In this section, we present the technique used in the paper to prove the results in Theorem 1 and 2.

For learning in the presence of random classification noise, when the noise rate $\eta = 1/2$, the labels will be randomly uniform, and learning is impossible. Therefore, we must assume that the learner knows some upper bound $\eta_b < 1/2$ for η [1].

1.2 Approximation Implies Learning k-Sparse Parities

In this section, we present two approaches that prove Theorem 1. The first is our method, and the second was suggested by an anonymous reviewer of RANDOM.

[3] See Section 1.4 for the justification of why we use this definition and not the standard definition $d(f) \leq D \leq \gamma(d(f))$.

[4] Throughout this paper, we also have $k = n - \omega(1)$. For $k = O(1)$ and $k = n - O(1)$, there are polynomial-time learning algorithms.

While the approach suggested by the anonymous reviewer is truly inspiring, we believe that our method offers significant value, is worth presenting in this paper, and may be useful for solving other problems.

1.2.1 First Approach

In this section, we will outline the technique for the binary field, though some essential details are omitted to provide a broader overview of the main concepts and approach. Additionally, proving the result for any field requires more careful treatment.

Let $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$ be any strictly increasing function such that for every[5] $x > 1$, $\gamma(x) > x$.

Let $\mathcal{A}$ be a polynomial-time randomized algorithm that γ-approximates the number of relevant variables $d(f)$ in a parity f, using random uniform labeled examples of f in the presence of random classification noise with any noise rate[6] $\eta \leq \eta_b$. Thus, for every parity f with $d(f)$ relevant variables, with probability at least $1 - \delta$ we have $\gamma^{-1}(d(f)) \leq \mathcal{A}(f) \leq \gamma(d(f))$. We will demonstrate how to construct a polynomial-time learning algorithm for $k(n)$-sparse parities, for some $k(n) = \omega_n(1)$. First, we will show how to find $k(n)$.

Let $\mathrm{Lin}(d)$ be the class of d-sparse parities. Assume for now that the noise rate is η_b. Later, we will show how to modify the algorithm to work for any unknown noise rate $\eta \leq \eta_b$.

We first use the algorithm $\mathcal{A}$ to construct a table that provides values which approximate

$$\Psi_{\mathcal{A}}(d) = \mathop{\mathbf{E}}_{(f,s)\sim_u \mathrm{Lin}(d)\times S(f)} [\mathcal{A}(f)]$$

with additive error of $1/poly(n)$, for every $d \in [n]$ and noise rate η_b. Here f is a d-sparse parity chosen uniformly at random from $\mathrm{Lin}(d)$, and s is a uniformly random string in $S(f)$ - the set of random bits used by the algorithm (for the randomness of the algorithm and the noise) for which the algorithm returns a correct answer, namely, returns D such that $\gamma^{-1}(d) \leq D \leq \gamma(d)$.

To approximate $\Psi_{\mathcal{A}}(d)$ for some $d \in [n]$, we iterate a polynomial number of times. At each iteration, we draw a random uniform $f \in \mathrm{Lin}(d)$ and run $\mathcal{A}$. For each labeled example requested by $\mathcal{A}$, we draw a random uniform $u \in \{0,1\}^n$ and compute $v = f(u)$. We then, with probability η_b, return[7] $(u, v+1)$ to $\mathcal{A}$, and, with probability $1 - \eta_b$, return (u, v). If the algorithm outputs an integer D such that $\gamma^{-1}(d) \leq D \leq \gamma(d)$, we retain that D. Otherwise, we repeat the process. Obviously, $\mathbf{E}[D] = \Psi_{\mathcal{A}}(d)$, and therefore, using Hoeffding's bound, such a table can be constructed in polynomial time.

Now, using the fact that γ is strictly increasing and $\gamma^{-1}(d) \leq \Psi_{\mathcal{A}}(d) \leq \gamma(d)$, and applying a basic averaging argument, we show that there exists a $k := k(n) = \omega_n(1)$ for which $\Psi_{\mathcal{A}}(k+1) - \Psi_{\mathcal{A}}(k-1) \geq 1/poly(n)$. We now show how to learn k-sparse parities with noise rate η_b in polynomial time and afterward for any $\eta \leq \eta_b$.

Suppose that the target function $f \in \mathrm{Lin}(k)$ can be accessed through random uniform labeled examples in the presence of random classification noise with noise rate η_b. We first show how to approximate $\Psi_{\mathcal{A}}(d(f(x) + x_i))$ for any $i \in [n]$ without knowing f. Recall that $d(f(x) + x_i)$ is the number of relevant variables in $f(x) + x_i$. The key idea here is that if (a, b) is a labeled example of f, then for a random uniform permutation ϕ, $((a_{\phi^{-1}(1)}, \ldots, a_{\phi^{-1}(n)}), b + a_i)$ is a labeled example of the function $f(x_{\phi(1)}, \ldots, x_{\phi(n)}) + x_{\phi(i)}$ which is a random and uniform drawn function

[5] We need this constraint to ensure that $\gamma^{-1}(x) < \gamma(x)$ for every $x > 1$.

[6] Here, η is not known to the algorithm, but η_b is known.

[7] Here, $+$ is exclusive or.

in $\mathrm{Lin}(d(f(x)+x_i))$. Therefore, using Hoeffding's Bound, we can approximate $\Psi_{\mathcal{A}}(d(f(x)+x_i))$ for every $i \in [n]$.

Now, x_i is relevant in $f(x)$ if and only if $f(x)+x_i \in \mathrm{Lin}(k-1)$ and then $\Psi_{\mathcal{A}}(d(f(x)+x_i)) = \Psi_{\mathcal{A}}(k-1)$. On the other hand, x_i is not relevant in $f(x)$ if and only if $f(x)+x_i \in \mathrm{Lin}(k+1)$, and then $\Psi_{\mathcal{A}}(d(f(x)+x_i)) = \Psi_{\mathcal{A}}(k+1)$. Since $\Psi_{\mathcal{A}}(k+1) - \Psi_{\mathcal{A}}(k-1) \geq 1/poly(n)$, these two cases are distinguishable in polynomial time. Consequently, we can differentiate between variables in f that are relevant and those that are not. This gives the learning algorithm to $\mathrm{Lin}(k)$ when the noise rate is η_b.

This algorithm runs in time $T = poly(n, 1/(1-2\eta_b))$. When η is not known, we can run the above procedure for all possible values $\eta^{(j)} = 1/2 - j/T^c$, where c is a sufficiently large constant, and $j \in [T^c/2] \cup \{1\}$. For each j, when the algorithm receives a labeled example (u, v), we magnify the error rate to η_b by drawing $\xi \in \{0,1\}$, which is equal to 1 with probability $(\eta_b - \eta^{(j)})/(1-2\eta^{(j)})$, and returning $(u, v+\xi)$ to the algorithm. This new labeled example has noise rate η_b. We collect all the $T^c/2+1$ hypotheses generated from the outputs and then employ a standard algorithm to select the one closest to the target [1]. The result follows because there exists a j such that[8] $|\eta^{(j)} - \eta| \leq 1/T^c$. Consequently, using the total variation distance, one of the hypotheses is the target.

In this paper, we extend our result to any linear function over any finite field $\mathbb{F}$. The approach used is similar to the case of parities (the binary field $\mathbb{F} = \{0,1\}$) with some technical but nontrivial modifications.

1.2.2 The Second Approach

This second approach was suggested by an anonymous reviewer of RANDOM, whose insightful comments and suggestions significantly improved this manuscript for the case of the binary field. For non-binary fields, this approach can identify the relevant variables of the function. We then use the approach developed in Lemma 6 and Lemma 7 to find the coefficients of the relevant variables.

Suppose there exists a randomized algorithm $\mathcal{A}(n)$ that runs in time $T(n)$ and γ-approximates the number of relevant variables in a parity function $f : \{0,1\}^n \to \{0,1\}$, using random uniform labeled examples of f in the presence of random classification noise with a noise rate $\eta \leq \eta_b$. Here, too, the algorithm needs to know an upper bound on η. We will explain the reasons for this below.

Let $k_1 = \omega_n(1)$ be an integer such that $k_2 = \gamma(\gamma(k_1)+1) = n - \omega_n(1)$. For any k_1-sparse parity f, the algorithm outputs $\mathcal{A}(f) \in [\gamma^{-1}(k_1), \gamma(k_1)]$, and for any k_2-sparse parity function g, it outputs $\mathcal{A}(g) \in [\gamma(k_1)+1, \gamma(\gamma(\gamma(k_1)+1))]$. Since the two intervals are disjoint, the algorithm can distinguish between k_1-sparse parities and k_2-sparse parities in polynomial time. Let $\mathcal{B}$ be the polynomial-time algorithm that distinguishes between them.

Consider the algorithm $\mathcal{B}$ when it runs on random uniform examples with random uniform labels. Suppose that with probability p, the algorithm answers that the function is a k_1-sparse parity, and with probability $1-p$, it answers that it is a k_2-sparse parity. If $p > 1/2$, then with probability at least $1/2$, $\mathcal{B}$ can distinguish between k_2-sparse parities and random uniform examples with random uniform labels. Otherwise, with probability at least $1/2$, $\mathcal{B}$ can distinguish between k_1-sparse parities and random uniform examples with random uniform labels.

Suppose, without loss of generality, the latter holds. We now give an algorithm that finds the relevant variables when the target function is a k_1-parity function and, consequently, learns

[8] If $\eta_j = \eta + \epsilon$ then the magnified noise is $\eta_b + \lambda\epsilon$ where $\lambda = (\eta + \eta_b - 1 + \epsilon)/(1 - 2(\eta+\epsilon))$.

k_1-sparse parities. This algorithm is from [10].

For every $i \in [n]$, we run the algorithm $\mathcal{B}$ and change the i-th coordinate of each example to a random uniform element in $\{0,1\}$. If x_i is not a relevant variable of f, then the labeled examples are labeled examples of f, and the algorithm answers that it is a k_1-parity function. If x_i is a relevant variable of f, then it is easy to see that the new labeled examples are random uniform with random uniform labels, and the algorithm answers accordingly. This distinguishes between variables in f from those that are not in f.

In this method, too, we must know some upper bound on η. Otherwise, algorithms $\mathcal{A}$ and $\mathcal{B}$ would need to be Las Vegas algorithms, and we would not know when to stop the algorithm when dealing with random uniform examples with random uniform labels.

For the problem of finding the relevant variables of the target in other fields, the generalization of this to any field is straightforward.

1.3 Approximation Implies Learning Parities

In this section, we show how γ-approximation implies proper learning parities.

Let $\gamma(x)$ be any strictly increasing function. Suppose there exists a randomized algorithm $\mathcal{A}(n)$ that runs in time $T(n)$ and γ-approximates the number of relevant variables in a parity $f : \{0,1\}^n \to \{0,1\}$, using random uniform labeled examples of f in the presence of random classification noise.

Let $\Gamma(x) = \gamma(\gamma(x))$. As in Section 1.2.1, using a basic averaging argument, we show that there exists a sequence of integers $k_1 < k_2 < \cdots < k_t < \Gamma^{-1}(n)$, where for each i, $k_{i+1} < \Gamma(k_i)$ and $\Psi_{\mathcal{A}}(k_i + 1) - \Psi_{\mathcal{A}}(k_i - 1) > 1/poly(n)$. As before, we obtain algorithms that learn $\mathrm{Lin}(k_i)$ for each i.

Now, we show how to obtain a learning algorithm for d-sparse parities for every $d < \Gamma^{-1}(\sqrt{n})$. Given any $d < \Gamma^{-1}(\sqrt{n}))$, there exists a j such that $k_{j-1} < d \leq k_j$ where $k_0 = 0$. To learn d-sparse parities, we uniformly at random choose distinct $i_1, \ldots, i_{k_j - d} \in [n]$, run the algorithm for learning k_j-sparse parities and modify each labeled example (a, b) to $(a, b + a_{i_1} + \cdots + a_{i_{k_j-d}})$. If $g(x) = f(x) + x_{i_1} + \cdots + x_{i_{k_j-d}}$ is in $\mathrm{Lin}(k_j)$, then the algorithm w.h.p learns $g(x)$. We then show that because $d < \Gamma^{-1}(\sqrt{n}))$, with high probability, the variables $x_{i_1}, \ldots, x_{i_{k_j-d}}$ are not relevant variables in f. We can then conclude that, w.h.p, $g \in \mathrm{Lin}(k_j)$. Therefore, w.h.p., we can learn $g(x)$, and consequently, we can learn $f(x) = g(x) + x_{i_1} + \cdots + x_{i_{k_j-d}}$.

This provides a learning algorithm for d-sparse parities for any $d \leq \Gamma^{-1}(\sqrt{n})$. Recognizing that this applies to every n, we can regard f as a function over $N := \Gamma(n)^2$ variables by adding $\Gamma(n)^2 - n$ dummy variables and appending $\Gamma(n)^2 - n$ random uniform elements from $\{0,1\}$ to each a in the labeled example (a, b). By applying this construction to the algorithm $\mathcal{A}(N)$, we obtain a learning algorithm for d-sparse parity for any $d \leq \Gamma^{-1}(\sqrt{N}) = n$.

Now, the algorithm for learning parities can run all the learning algorithms for d-sparse parities for all $d \leq n$. It takes all the outputs and then employs a standard algorithm to select the one closest to the target [1]. See also Lemma 1.

1.4 Justification for the Use of the γ-Approximation Definition

In our approach, we define a γ-approximation of the number of relevant variables $d(f)$ in a parity function f such that $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$, instead of using the standard definition $d(f) \leq D \leq \gamma(d(f))$.

The key reason for this choice is that the latter definition loses its effectiveness when $d(f)$ approaches n, the number of variables. Specifically, if $d(f)$ is close to n, say $O(n)$, the condition $d(f) \leq D \leq \gamma(d(f))$, for $\gamma(n) = \omega(n)$, effectively reduces to $d(f) \leq D \leq n$. In this scenario, the value of γ becomes less significant because the approximation D would naturally fall within the trivial range of $d(f) = O(n)$ to n.

On the other hand, our chosen definition $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$ ensures that the approximation D always depends on the function γ. This definition retains its utility even when $d(f)$ is large, as $\gamma^{-1}(d(f))$ provides a lower bound that is influenced by γ, thereby maintaining the approximation's relevance and precision.

Thus, by using the γ-approximation definition $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$, we ensure a meaningful and consistent approximation of the number of relevant variables $d(f)$ across the entire range of possible values, preserving the value and impact of the function γ.

2 Definitions and Preliminaries

Let $\mathbb{F}$ be any finite field and $\mathbb{F}_q$ be the field with q elements. We define $\mathrm{Lin}(\mathbb{F})$ as the class of all linear functions over the field $\mathbb{F}$, i.e., functions $a \cdot x$ where $a \in \mathbb{F}^n$ and $x = (x_1, \ldots, x_n)$. A *d-sparse linear function* over $\mathbb{F}$ is a function in $\mathrm{Lin}(\mathbb{F})$ with d relevant variables. The class $\mathrm{Lin}(\mathbb{F}, d)$ is the class of all d-sparse linear functions over $\mathbb{F}$. When $\mathbb{F}$ is the binary field $\mathbb{F}_2 = \{0, 1\}$, the functions in $\mathrm{Lin}(\mathbb{F}_2)$ are called parity functions, and the functions in $\mathrm{Lin}(\mathbb{F}_2, d)$ are called d-sparse parities. We use the notation $\mathrm{Lin}_n(\mathbb{F})$ and $\mathrm{Lin}_n(\mathbb{F}, d)$ to emphasize the number of variables.

For $f \in \mathrm{Lin}(\mathbb{F})$, we denote by $d(f)$ the number of variables on which f depends. For a strictly increasing function $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$ such that $\gamma(x) > x$ for every x, we say that an algorithm $\mathcal{A}$ γ-approximates $d(f)$ in time $T = T(n)$ and $Q = Q(n)$ labeled examples if the algorithm runs in time T, uses Q labeled examples to f, and with probability at least $2/3$, returns an integer D such that $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$.

In proper learning $\mathrm{Lin}(\mathbb{F})$ under the uniform distribution, the learner can observe labeled examples (a, b) where $b = f(a)$ and $a \in \mathbb{F}^n$ are drawn independently and uniformly distributed over $\mathbb{F}^n$, with $f \in \mathrm{Lin}(\mathbb{F})$ being the target linear function. The goal is to (properly) exactly return the linear function f. In the random classification noise model with noise rate η, each label b is equal to $f(a)$ with probability $1 - \eta$ and is a random uniform element in $\mathbb{F} \backslash \{f(a)\}$ with probability η.

When $\eta = 1 - 1/|\mathbb{F}|$, the label is a random uniform element of $\mathbb{F}$; hence, learning is impossible. Therefore, we must assume that the learner knows some upper bound $\eta_b < 1 - 1/|\mathbb{F}|$ for η [1]. When $\eta = \eta_b$, to distinguish between labeled examples with random uniform labels and the function $f(x) = 0$, we need at least $1/(1 - \eta_b - 1/|\mathbb{F}|)$ labeled examples. Therefore, a polynomial-time algorithm in this model is an algorithm that runs in time $poly(1/(1 - \eta_b - 1/|\mathbb{F}|)), n, 1/\delta)$ [1].

The following Lemma shows how to learn when the algorithm has unlimited computational power.

Lemma 1. *Let $C \subseteq \mathrm{Lin}(\mathbb{F})$. Then C is learnable under the uniform distribution in the random classification noise model in time $\tilde{O}(|C| \log(1/\delta)/(1-\eta_b - 1/|\mathbb{F}|)^2)$ from*

$$Q = \frac{\log \frac{|C|}{\delta}}{(1-\eta_b - 1/|\mathbb{F}|)^2}$$

labeled examples.

Proof. Let (a, b) be a labeled example and f be the target function. Then

$$\Pr[f(a) = b] = \eta \Pr[f(a) = b | b \neq f(a)] + (1-\eta) \Pr[f(a) = b | b = f(a)] = 1 - \eta \geq 1 - \eta_b.$$

If $g \neq f$ and $g \in \mathrm{Lin}(\mathbb{F})$ then

$$\Pr[g(a) = b] = \eta \Pr[g(a) = b | b \neq f(a)] + (1-\eta) \Pr[g(a) = b | b = f(a)] = \frac{1}{|\mathbb{F}|}.$$

The result now follows by applying Chernoff's bound to estimate $\Pr[g(a) = b]$ for all $g \in C$ with confidence of $1 - \delta/|C|$ and an additive error of $(1 - \eta_b - 1/|\mathbb{F}|)/4$. □

The following lemma shows that, in approximation algorithms, the dependency on δ is logarithmic. This is a well-known result. For completeness, a sketch of the proof is provided.

Lemma 2. *If there exists an algorithm $\mathcal{A}$ that runs in time $T(n)$, uses $Q(n)$ labeled examples to $f \in \mathrm{Lin}(\mathbb{F}, d)$ according to the uniform distribution in the presence of random classification noise and, with probability at least $2/3$, returns a γ-approximation of $d(f)$, then there is an algorithm that runs in time $O(T(n) \log(1/\delta))$, uses $O(Q(n) \log(1/\delta))$ labeled examples to $f \in \mathrm{Lin}(\mathbb{F}, d)$ according to the uniform distribution in the presence of random classification noise, and with probability at least $1 - \delta$, returns a γ-approximation of $d(f)$.*

Proof. We run $\mathcal{A}$, $O(\log(1/\delta))$ times and take the median of the outputs. The correctness of this algorithm follows from an application of Chernoff's bound. □

The same is true for learning.

Lemma 3. *If there exists an algorithm $\mathcal{A}$ that runs in time $T(n)$, uses $Q(n)$ labeled examples to $f \in \mathrm{Lin}(\mathbb{F}, d)$ according to the uniform distribution in the presence of random classification noise and, with probability at least $2/3$, properly learns the target f, then there is an algorithm that runs in time $O(T(n) \log(1/\delta))$, uses $O(Q(n) \log(1/\delta))$ labeled examples to $f \in \mathrm{Lin}(\mathbb{F}, d)$ according to the uniform distribution in the presence of random classification noise, and with probability at least $1 - \delta$, learns the target f.*

Proof. Since $\mathcal{A}$ properly learns f, we run the algorithm $O(\log(1/\delta))$ times and output the hypothesis that occurs most frequently in the output. □

3 Approximation vs. Learning

In this section, we prove the two results.

3.1 Approximation Implies Learning Some $\mathrm{Lin}(\mathbb{F}, k)$

In this section, we prove that approximating the number of relevant variables in the parity function implies polynomial-time properly learning $\mathrm{Lin}(\mathbb{F}, k(n))$ for some $k(n) = \omega_n(1)$.

We prove.

Theorem 3. *Let $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$ be any strictly increasing function where $\gamma(x) > x$ for every x. Let $\pi(n)$ be any function such that $\pi(n) = \omega_n(1)$. Consider any polynomial-time algorithm $\mathcal{A}'(n)$ that, for any linear function $f \in \mathrm{Lin}(\mathbb{F})$, uses random uniform labeled examples of f in the presence of random classification noise and, with probability at least $2/3$, returns a γ-approximation of the number of relevant variables $d(f)$ of f. From $\mathcal{A}'(n)$, one can, in polynomial time, construct a $poly(n, 1/(1-\eta_b-1/|\mathbb{F}|), \min(|\mathbb{F}|, 1/(1-\eta_b)^{\pi(n)}))$-time algorithm that properly learns $\mathrm{Lin}(\mathbb{F}, k(n))$ from a polynomial number of random uniform labeled examples in the presence of random classification noise for some $k(n) = \omega_n(1)$.*

We will assume for now that the noise rate $\eta = \eta_b$ is known. In Section 1.2.1, we showed how to handle unknown noise rates $\eta \leq \eta_b$. Recall that a polynomial-time algorithm in this model is an algorithm that runs in time $poly(1/(1-\eta_b-1/|\mathbb{F}|)), n, 1/\delta)$, [1]. In particular, the algorithm constructed in Theorem 3 runs in polynomial time for either

- Any η_b and fields of size[9] $|\mathbb{F}| = poly(n)$, or
- Any field $\mathbb{F}$ when $\eta_b \leq 1 - 1/|\mathbb{F}| - 1/\psi(n)$, where $\psi(n) = 2^{o(\log(n))}$.

Let $\mathcal{A}'(n, s, f)$ be any algorithm that uses random uniform labeled examples of $f \in \mathrm{Lin}_n(\mathbb{F})$ in the presence of random classification noise and, with probability at least $2/3$, returns a γ-approximation of the number of relevant variables, $d(f)$, of f. The new parameter s is added for the random bits used in the algorithm for its coin flips and the noise. First, we will use Lemma 2 to make the algorithm's success probability $1 - \delta'$ for a fixed, sufficient small δ' that depends on n and $|\mathbb{F}|$. For the proof of the Theorem in this section, $\delta' = 1/(|\mathbb{F}|n^7)$ suffices. By Lemma 2, this adds a factor of $O(\log n + \log|\mathbb{F}|)$ to the time and the number of labeled examples which will be swallowed by the $\tilde{O}(\cdot)$ in the final time and sample complexity. Second, we will modify the output of the algorithm to $\min(\gamma(D_f), n)$, where D_f is the output of the latter algorithm. Let the resulting algorithm be denoted as $\mathcal{A}$. We will denote the algorithm's output by $\mathcal{A}(n, s, f)$. Consequently, we will have that, with probability at least $1 - \delta'$,

$$d(f) \leq \mathcal{A}(n, s, f) \leq \Delta(d(f)) \leq n \tag{1}$$

where

$$\Delta(x) = \min(\gamma(\gamma(x)), n).$$

Let S_f be the set of all random strings s' for which $d(f) \leq \mathcal{A}(n, s', f) \leq \Delta(d(f))$; that is, it includes all the random strings that yield correct answers. Throughout this section and the next,

[9] This makes sense when we have a sequence of fields $\mathbb{F}_i$ such that $\mathbb{F}_i \subseteq \mathbb{F}_{i+1}$ and $|\mathbb{F}_n| = poly(n)$.

we say that $\mathcal{A}$ Δ-approximates $d(f)$. See (1). This should not be confused with the previous definition of γ-approximates $d(f)$. Here, we use the capital letter Δ to prevent any ambiguity.

Let

$$\Psi_{\mathcal{A}}(d) = \mathop{\mathbf{E}}_{(f,s)\sim_u \mathrm{Lin}(\mathbb{F},d)\times S(f)} [\mathcal{A}(n,s,f)]$$

where $\sim_u$ indicates that f is chosen uniformly at random from $\mathrm{Lin}(\mathbb{F},d)$ and s uniformly at random from $S(f)$. Since $d \leq \mathcal{A}(n,s,f) \leq \Delta(d) \leq n$ for $s \in S(f)$ where $f \in \mathrm{Lin}(\mathbb{F},d)$, we have

$$d \leq \Psi_{\mathcal{A}}(d) \leq \Delta(d) \leq n. \tag{2}$$

We note here that $\Psi_{\mathcal{A}}(d)$ is independent of δ, as in $\mathcal{A}$, we set $\delta = \delta'$ for a fixed δ'. This is crucial for ensuring the correctness of the proof. Also, $\Psi_{\mathcal{A}}(d)$ depends on n. This will be essential only for the next result in the next section.

We first prove that the values of $\Psi_{\mathcal{A}}(d)$ for $d \in [n]$ can be approximated with high probability.

Lemma 4. *Let $0 < h < 1$. Let $\mathcal{A}$ be an algorithm that runs in time $T(n)$, uses $Q(n)$ labeled examples of $f \in \mathrm{Lin}(\mathbb{F})$ according to the uniform distribution in the presence of random classification noise, and, with probability at least $1-\delta'$, Δ-approximates $d(f)$. A table of real values $\Psi'_{\mathcal{A}}(d)$ for $1 \leq d \leq n$ can be constructed in time $\tilde{O}(n^3/h^2)T(n)\log(1/\delta)$, and without using any labeled examples. This table, with probability at least $1-\delta$, satisfies $|\Psi'_{\mathcal{A}}(d) - \Psi_{\mathcal{A}}(d)| \leq h$ for all $d \in [n]$.*

Proof. Define a random variable as the output D of the algorithm $\mathcal{A}$, obtained from running it on a uniformly random f from $\mathrm{Lin}(\mathbb{F},d)$, provided that the output lies within the interval $[d,\Delta(d)]$. The labeled examples of f can be generated by choosing a random uniform $u \in \{0,1\}^n$ and returning $(u, f(u)+e)$ to $\mathcal{A}$ where, with probability $1-\eta_b$, $e = 0$ and, with probability η_b, e is random uniform in $\mathbb{F}\backslash\{0\}$. Obviously, $\mathbf{E}[D] = \Psi_{\mathcal{A}}(d)$.

By Hoeffding's bound, to compute $\mathbf{E}[D]$ with an additive error h and a confidence probability of at least $1-\delta/(2n)$, we need to obtain $t = O((n^2/h^2)\log(n/\delta))$ values of D. Since the success probability of obtaining a value of D in the interval $[d,\Delta(d)]$ is $1-\delta' > 2/3$, we need to run the algorithm $O(t+\log(2n/\delta))$ times to acquire t values with a success probability at least $1-\delta/(2n)$. Therefore, the time complexity is $O((t+\log(2n/\delta))nT(n)) = O(tnT(n)) = \tilde{O}(n^3/h^2)T(n)\log(1/\delta)$. □

Our next result shows how to estimate $\Psi_{\mathcal{A}}(d(f))$ of the target f without knowing $d(f)$.

Lemma 5. *Let $0 < h < 1$ and $\tau = O((n^2/h^2)\log(1/\delta))$. Let $\mathcal{A}$ be an algorithm that runs in time $T(n)$, uses $Q(n)$ labeled examples of $f \in \mathrm{Lin}(\mathbb{F})$ according to the uniform distribution, in the presence of random classification noise, and, with probability at least $1-\delta'$, Δ-approximates $d(f)$. There is an algorithm $\mathcal{B}(n,h)$ that runs in time $T' = \tau T(n)$, uses $Q' = \tau Q(n)$ labeled examples of $f \in \mathrm{Lin}(\mathbb{F})$ according to the uniform distribution in the presence of random classification noise and, with probability at least $1-\delta/2-\tau\delta'$, returns ψ that satisfies $|\psi - \Psi_{\mathcal{A}}(d(f))| \leq h$.*

Proof. Suppose $v \in (\mathbb{F}\backslash\{0\})^n$ is chosen uniformly at random and $\phi : [n] \to [n]$ is a uniformly random permutation. If we run $\mathcal{A}$ with the target $f = \lambda_1 x_{i_1} + \cdots + \lambda_d x_{i_d}$, and for every labeled example $(a,b) \in \mathbb{F}^n \times \mathbb{F}$, we modify the labeled example to $((v_1^{-1}a_{\phi^{-1}(1)}, \ldots, v_n^{-1}a_{\phi^{-1}(n)}), b)$, then the new labeled examples remain uniform and consistent with the function $g(x) = \lambda_1 v_{\phi(i_1)} x_{\phi(i_1)} + \cdots + \lambda_d v_{\phi(i_d)} x_{\phi(i_d)}$. This function g is a uniformly random element of $\mathrm{Lin}(\mathbb{F},d)$. Using this fact, we show how to approximate $\Psi_{\mathcal{A}}(d)$.

To this end, let $\tau = O((n^2/h^2)\log(1/\delta))$. We iterate τ times, and at each iteration, we choose a random uniform $v \in (\mathbb{F}\backslash\{0\})^n$ and random uniform permutation $\phi : [n] \to [n]$. We request for $Q(n)$ labeled examples and modify each labeled example $(a, b) \in \mathbb{F}^n \times \mathbb{F}$ to $((v_1^{-1}a_{\phi^{-1}(1)}, \ldots, v_n^{-1}$ $a_{\phi^{-1}(n)}), b)$. We then run $\mathcal{A}$ on these labeled examples. Let D_i be the output of the i-th iteration. We then output $\psi' = \left(\sum_{i=1}^{\tau} D_i\right)/\tau$.

We now prove that, with probability at least $1 - \delta/2 - \tau\delta'$, we have $|\psi' - \Psi_{\mathcal{A}}(d(f))| \leq h$. Since $\mathcal{A}(n)$ runs τ times, with probability at least $1 - \tau\delta'$, all the seeds used by $\mathcal{A}$ are in $S(f)$ and $d(f) \leq D_i \leq \Delta(d(f))$. Also, since $\mathcal{A}(n)$ runs on a uniformly random function in $\mathrm{Lin}(\mathbb{F}, d)$, we have $\mathbf{E}[D_i] = \Psi_{\mathcal{A}}(d)$. By Hoeffding's bound, along with the fact that $D_i \leq \Delta(d(f)) \leq n$, we can conclude that, with probability at least $1 - \delta/2$, we have $|\psi' - \Psi_{\mathcal{A}}(d(f))| \leq h$. □

Notice that in Lemma 5, τ also depends on h. As h eventually will be $O(1/n)$ and $\delta' = 1/(|\mathbb{F}|n^7)$, the success probability $1 - \delta/2 - \tau\delta'$ will be $1 - o_n(1)$ for $\delta = 1/n$.

We now show that in any large enough sub-interval of $[0, n]$, there is k for which $\mathcal{A}$ can be used to learn $\mathrm{Lin}(\mathbb{F}, k)$.

Lemma 6. *Let $\mathcal{A}(n)$ be an algorithm that runs in time $T(n)$, uses $Q(n)$ labeled examples of $f \in \mathrm{Lin}(\mathbb{F})$ according to the uniform distribution in the presence of random classification noise, and, with probability at least $1-\delta'$, Δ-approximates $d(f)$. For every $1 \leq m \leq \min\{j|\Delta(j) = n\} = \gamma^{-1}(\gamma^{-1}(n))$ there exists $m \leq k \leq \Delta(m) + 1$ and*

1. *An algorithm that, for every $f \in \mathrm{Lin}(\mathbb{F}, k)$, with probability at least $1-\delta/8-2\tau n\delta'$, where $\tau = O(n^4\log(1/\delta))$, identifies the relevant variables of f from random uniform labeled examples in the presence of random classification noise. This algorithm runs in time $\tilde{O}(n^5)T(n)\log(1/\delta)$ and uses $\tilde{O}(n^4)Q(n)\log(1/\delta)$ labeled examples.*
2. *An algorithm that, with probability at least $1 - \delta/2 - |\mathbb{F}|kn\delta'$, properly learns $\mathrm{Lin}(\mathbb{F}, k)$, from random uniform labeled examples in the presence of random classification noise. This algorithm runs in time $\tilde{O}(|\mathbb{F}|kn^5)T(n)\log(1/\delta)$ and uses $\tilde{O}(n^4)Q(n)\log(|\mathbb{F}|/\delta)$ labeled examples.*

Such k can be found in time $\tilde{O}(n^5)T(n)\log(1/\delta)$.

Proof. We first prove the result when the field is not the binary field. Let m be any integer such that $1 \leq m \leq \min\{j|\Delta(j) = n\}$. Since by (2),

$$\begin{aligned}\sum_{i=m}^{\Delta(m)} \Psi_{\mathcal{A}}(i+1) - \Psi_{\mathcal{A}}(i) &= \Psi_{\mathcal{A}}\left(\Delta(m)+1\right) - \Psi_{\mathcal{A}}\left(m\right) \\ &\geq \Delta(m) + 1 - \Delta(m) = 1,\end{aligned}$$

there is k such that $m \leq k \leq \Delta(m)$ and

$$\Psi_{\mathcal{A}}(k+1) - \Psi_{\mathcal{A}}(k) \geq \frac{1}{\Delta(m) - m + 1} \geq \frac{1}{n}.$$

First, we find such k. By Lemma 4, taking $h = 1/(16n)$, with probability at least $1 - \delta/4$, we can find k such that $\Psi_{\mathcal{A}}(k+1) - \Psi_{\mathcal{A}}(k) \geq 7/(8n)$ in time $\tilde{O}(n^5)T(n)\log(1/\delta)$.

We now present an algorithm that learns $\mathrm{Lin}(\mathbb{F}, k)$. The algorithm uses the algorithm in Lemma 5 to approximate $\Psi_{\mathcal{A}}(d(f + x_i))$ and $\Psi_{\mathcal{A}}(d(f + \alpha x_i))$ for some $\alpha \in \mathbb{F}\backslash\{0, 1\}$ and all $i \in [n]$ with an additive error of $1/(8n)$. If x_i is not a relevant variable of the target function f, then

both $f + x_i$ and $f + \alpha x_i$ are in $\mathrm{Lin}(\mathbb{F}, k+1)$. Consequently, we obtain two values in the inteval $[\Psi_{\mathcal{A}}(k+1) - 1/(8n), \Psi_{\mathcal{A}}(k+1) + 1/(8n)]$. If x_i is a relevant variable in the function, then one of the functions, either $f + x_i$ or $f + \alpha x_i$ is in $\mathrm{Lin}(\mathbb{F}, k)$, and therefore, one of the values is in the inteval $[\Psi_{\mathcal{A}}(k) - 1/(8n), \Psi_{\mathcal{A}}(k) + 1/(8n)]$. Since $\Psi_{\mathcal{A}}(k) + 1/(8n) < \Psi_{\mathcal{A}}(k+1) - 1/(8n)$, the intervals are disjoint, and thus we can distinguish between the two cases.

By Lemma 5, with probability $1 - \delta/2 - \tau\delta'$, we can approximate each $\Psi_{\mathcal{A}}(d(f + x_i))$ (or $\Psi_{\mathcal{A}}(d(f + \alpha x_i))$) with an additive error $h = 1/(8n)$ in time $\tau T(n)$ and $\tau Q(n)$ labeled examples, where $\tau = O(n^4 \log(1/\delta))$. Taking $\delta/(8n)$ for δ, with probability $1 - \delta/8 - 2\tau n\delta'$, we can approximate all $\Psi_{\mathcal{A}}(d(f + x_i))$ and $\Psi_{\mathcal{A}}(d(f + \alpha x_i))$, $i \in [n]$ with an additive error $h = 1/(8n)$ in time $\tau' nT(n)$ and $\tau' Q(n)$ labeled examples where $\tau' = O(n^4 \log(n/\delta))$. This completes the proof of item 1 for the case where the field is not the binary field.

To prove item 2, suppose, without loss of generality, that $x_1, \ldots, x_k$ are the relevant variables in f. We approximate $\psi_{\alpha,i} := \Psi_{\mathcal{A}}(d(f - \alpha x_i + x_{k+1}))$ for all $\alpha \in \mathbb{F}$ and for every $i \in [n]$. The result follows from the fact that $\psi_{\alpha,i} = \Psi_{\mathcal{A}}(k)$ if and only if the coefficient of x_i is α. Otherwise, $\psi_{\alpha,i} = \Psi_{\mathcal{A}}(k+1)$. By Lemma 5, to approximate all $\psi_{\alpha,i}$, with success probability of $1 - \delta/4 - |\mathbb{F}|kn\delta'$, we need time $O(|\mathbb{F}|kn^5 T(n) \log(|\mathbb{F}|n/\delta))$ and $O(n^4 T(n) \log(|\mathbb{F}|n/\delta))$ labeled examples. This completes the proof of item 2 for the case where the field is not the binary field.

Similar to the approach described above, for the binary field, we can show that there exists a k such that $\Psi_{\mathcal{A}}(k+1) - \Psi_{\mathcal{A}}(k-1) \geq 1/n$. Then, use the algorithm described in Lemma 5 to approximate $\Psi_{\mathcal{A}}(d(f + x_i))$ for all $i \in [n]$. If x_i is not a relevant variable of the target function f, then $\Psi_{\mathcal{A}}(d(f + x_i)) \in [\Psi_{\mathcal{A}}(k+1) - 1/(8n), \Psi_{\mathcal{A}}(k+1) + 1/(8n)]$. If x_i is a relevant variable in f, then $\Psi_{\mathcal{A}}(d(f + x_i)) \in [\Psi_{\mathcal{A}}(k-1) - 1/(8n), \Psi_{\mathcal{A}}(k-1) + 1/(8n)]$. Since both intervals are disjoint, we obtain the desired result.[10] □

Notice that in item 2, the success probability $1 - \delta/2 - |\mathbb{F}|kn\delta'$, and the time complexity depends on $|\mathbb{F}|$. We now present an alternative algorithm for finding the coefficients of the linear function, given that the algorithm knows the relevant variables.

Lemma 7. *Let $\mathcal{A}$ be an algorithm that, for every $f \in \mathrm{Lin}(\mathbb{F}, k)$, runs in time T, uses Q random uniform labeled examples in the presence of random classification noise, and identifies the relevant variables of f. Then there is an algorithm that properly learns $\mathrm{Lin}(\mathbb{F}, k)$ in time $T + \tilde{O}((n^3/(1 - \eta_b)^k + n/(1 - \eta_b - 1/|\mathbb{F}|)^2) \log(1/\delta))$ and uses $Q + O(((1/(1 - \eta_b)^k + n/(1 - \eta_b - 1/|\mathbb{F}|)^2) \log(1/\delta))$ random uniform labeled examples in the presence of random classification noise.*

Proof. We run $\mathcal{A}$ to find the relevant variables. The algorithm that finds the coefficients iterates $O((1/(1-\eta_b)^k) \log(1/\delta))$ times. At each iteration, it requests k labeled examples and uses Gaussian elimination to find the coefficients. Then, it tests whether the output function matches the target. If not, it proceeds to the next iteration.

Since $\eta = \eta_b$, the probability that all the labeled examples are correct is $(1 - \eta_b)^k$. If $|\mathbb{F}| = q$, the probability that k random entries in the examples form a non-singular matrix is

$$\left(1 - \frac{1}{q^k}\right)\left(1 - \frac{1}{q^{k-1}}\right) \cdots \left(1 - \frac{1}{q}\right) \geq \frac{1}{4}.$$

[10] Another approach is to utilize the fact that $\Psi_{\mathcal{A}}(k) - \Psi_{\mathcal{A}}(k-1) \geq 1/n$, and for every i, approximate $\Psi_{\mathcal{A}}(g_i)$, where $g_i = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$, using all labeled examples (a, b) that satisfy $a_i = 0$.

Therefore, after $t = O((1/(1-\eta_b)^k)\log(1/\delta))$ iterations, with probability at least $1-\delta/3$, at least one of the outputs is the target. By Lemma 1, learning the target from a set of t linear functions with a success probability of $1-\delta/3$ requires $\tilde{O}((1/(1-\eta_b-1/q)^2)(k+\log(1/\delta))$ labeled examples. $\square$

We are now ready to prove Theorem 3.

Proof. Let $\mathcal{A}'$ be an algorithm that runs in polynomial time, uses labeled examples according to the uniform distribution in the presence of random classification noise, and outputs $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$. Modify the algorithm to output $\min(\gamma(D), n)$. The algorithm now is a Δ-approximation algorithm, where $\Delta(x) = \min(\gamma(\gamma(x)), n)$. Let $m(n) = \gamma^{-1}(\gamma^{-1}(\pi(n)))$. Since $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$ is strictly increasing and is defined for all $\mathbb{R}^+$, we have $\gamma^{-1} : \mathbb{R}^+ \to \mathbb{R}^+$, is also strictly increasing, and $m(n) = \omega_n(1)$. Let $\delta' = 1/(|\mathbb{F}|n^7)$ and $\delta = 1/n$. By Lemma 6, item 1, there exists $m(n) \leq k(n) \leq \Delta(m(n)) = \pi(n)$, and an algorithm that, for every $f \in \mathrm{Lin}(\mathbb{F}, k(n))$, with probability at least $1 - o_n(1) > 2/3$, identifies the relevant variables of f from random uniform labeled examples in the presence of random classification noise. Also, this algorithm runs in polynomial time. Since $k(n) \leq \pi(n)$, by Lemma 7 and item 2 in Lemma 6, there is a $poly(n, 1/(1-\eta_b-1/|\mathbb{F}|), \min(|\mathbb{F}|, 1/(1-\eta_b)^{\pi(n)}))$ time learning algorithm for $\mathrm{Lin}(\mathbb{F}, k)$. Since $k(n) \geq m(n) = \omega_n(1)$, the result follows. $\square$

3.2 Approximation Implies Learning $\mathrm{Lin}(\mathbb{F})$

In this section, we prove.

Theorem 4. *Let $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$ be any strictly increasing function, where $\gamma(x) > x$ for every x. Let $\Gamma(x) := \gamma(\gamma(x))$. Consider any $T(n)$-time algorithm $\mathcal{A}'(n)$ that, for any linear function $f \in \mathrm{Lin}(\mathbb{F})$, uses $Q(n)$ random uniform labeled examples of f in the presence of random classification noise and, with probability at least $2/3$, returns a γ-approximation of the number of relevant variables $d(f)$ of f. From $\mathcal{A}'(n)$, one can, in polynomial time, construct a $\tilde{O}(|\mathbb{F}|\Gamma(n)^{12})T(O(\Gamma(n)^2))\log(1/\delta)$-time algorithm that properly learns $\mathrm{Lin}(\mathbb{F})$ from $\tilde{O}(\Gamma(n)^8)Q(O(\Gamma(n)^2))\log(|\mathbb{F}|/\delta)$ random uniform labeled examples in the presence of random classification noise.*

We first show that for every d satisfying $12\Gamma(d)^2 \leq n$, there exists a learning algorithm for $\mathrm{Lin}(\mathbb{F}, d)$.

Lemma 8. *Suppose that for every $1 \leq m \leq m' := \Gamma^{-1}(n)$, there exists a k such that $m \leq k \leq \Gamma(m)$, and an algorithm that runs in time $T(n)$ and, with probability at least $2/3$, properly learns $f \in \mathrm{Lin}(\mathbb{F}, k)$ under the uniform distribution in the presence of random classification noise, and uses $Q(n)$ labeled examples. Then, for every d such that $12\Gamma(d)^2 \leq n$, there is an algorithm that runs in time $O(T(n)\log(1/\delta))$ and properly learns $\mathrm{Lin}(\mathbb{F}, d)$ under the uniform distribution in the presence of random classification noise, using $O(Q(n)\log(1/\delta))$ labeled examples.*

Proof. Let d be an integer such that $12\Gamma(d)^2 < n$. Since $\Gamma(d) < n$, we have $d \leq m'$. Consequently, there exists k such that $d \leq k \leq \Gamma(d) \leq \Gamma(m') = n$, along with a proper learning algorithm $\mathcal{B}(n, k)$ for $\mathrm{Lin}(\mathbb{F}, k)$, that runs in time $T(n)$ and uses $Q(n)$ labeled examples.

We now present an algorithm for learning $\mathrm{Lin}(\mathbb{F}, d)$. We uniformly at random draw $k - d$ variables $x_{i_1}, \ldots, x_{i_{k-d}}$ and run the algorithm $\mathcal{B}(n, k)$. For each labeled example (a, b) of f, we feed $\mathcal{B}$ with the labeled example $(a, b + a_{i_1} + \cdots + a_{i_{k-d}})$. This modified labeled example serves as a

labeled example for the function $g = f + x_{i_1} + \cdots + x_{i_{k-d}}$. The probability that $g \in \mathrm{Lin}(\mathbb{F}, k)$ is the probability that none of the variables $x_{i_1}, \ldots, x_{i_{k-d}}$ are relevant in f. This probability is given by

$$\prod_{i=d}^{k} \left(1 - \frac{i}{n}\right) \geq 1 - \frac{k^2}{n} \geq 1 - \frac{\Gamma(d)^2}{n} \geq \frac{11}{12}.$$

Therefore, with probability at least $1 - (1/3 + 1/12) > 1/2$, algorithm $\mathcal{B}(n, k)$ learns g and thus learns f. By Lemma 3, the result follows. □

We now show how to construct a learning algorithm for $\mathrm{Lin}(\mathbb{F}, d)$ for every $d \leq n$.

Lemma 9. *Suppose that for every n and every d that satisfies $12\Gamma(d)^2 \leq n$, there is an algorithm $\mathcal{A}(n)$ that runs in time $T(n)$ and, with probability at least $2/3$, properly learns $f \in \mathrm{Lin}(\mathbb{F}, d)$ under the uniform distribution in the presence of random classification noise, using $Q(n)$ labeled examples. Let $N(n) = 12\Gamma(n)^2$. Then, for every n and every $d \leq n$, there is an algorithm that runs in time $T(N(n))$ and, with probability at least $2/3$, properly learns $f \in \mathrm{Lin}(\mathbb{F}, d)$ under the uniform distribution in the presence of random classification noise, using $Q(N(n))$ labeled examples.*

Proof. Let $N = N(n)$. Then for every $d \leq n$, we have $12\Gamma(d)^2 \leq 12\Gamma(n)^2 = N(n)$. We run $\mathcal{A}(N)$. Whenever the algorithm requests a labeled example, we draw a labeled example $(a, b) \in \mathbb{F}^n \times \mathbb{F}$, append $N - n$ random uniform entries to a, creating a'. We then provide (a', b) to $\mathcal{A}(N)$. The algorithm is effective for any d that satisfies $12\Gamma(d)^2 \leq N = 12\Gamma(n)^2$ and, thereby, covers all $d \leq n$. □

Theorem 4 now follows from Lemma 6, 8 and 9.

Acknowledgements: We would like to thank the anonymous reviewer of RANDOM for providing another approach for finding the relevant variables in the target function. We also extend our gratitude to the other reviewers for their useful comments and suggestions, which have greatly improved this manuscript.

References

[1] Dana Angluin and Philip D. Laird. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370, 1987. doi:10.1007/BF00116829.

[2] Per Austrin and Subhash Khot. A simple deterministic reduction for the gap minimum distance of code problem. *IEEE Trans. Inf. Theory*, 60(10):6636–6645, 2014. doi:10.1109/TIT.2014.2340869.

[3] Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *J. ACM*, 68(3):16:1–16:40, 2021. doi:10.1145/3444942.

[4] Arnab Bhattacharyya, Ameet Gadekar, Suprovat Ghoshal, and Rishi Saket. On the hardness of learning sparse parities. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*,

volume 57 of *LIPIcs*, pages 11:1–11:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. URL: https://doi.org/10.4230/LIPIcs.ESA.2016.11, doi:10.4230/LIPICS.ESA.2016.11.

[5] Arnab Bhattacharyya, Ameet Gadekar, and Ninad Rajgopal. On learning k-parities with and without noise. *CoRR*, abs/1502.05375, 2015. URL: http://arxiv.org/abs/1502.05375, arXiv:1502.05375.

[6] Arnab Bhattacharyya, Ameet Gadekar, and Ninad Rajgopal. Improved learning of k-parities. *Theor. Comput. Sci.*, 840:249–256, 2020. URL: https://doi.org/10.1016/j.tcs.2020.08.025, doi:10.1016/J.TCS.2020.08.025.

[7] Arnab Bhattacharyya, Piotr Indyk, David P. Woodruff, and Ning Xie. The complexity of linear dependence problems in vector spaces. In Bernard Chazelle, editor, *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 496–508. Tsinghua University Press, 2011. URL: http://conference.iiis.tsinghua.edu.cn/ICS2011/content/papers/33.html.

[8] Avrim Blum. On-line algorithms in machine learning. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms, The State of the Art (the book grow out of a Dagstuhl Seminar, June 1996)*, volume 1442 of *Lecture Notes in Computer Science*, pages 306–325. Springer, 1996. URL: https://doi.org/10.1007/BFb0029575, doi:10.1007/BFB0029575.

[9] Avrim Blum, Merrick L. Furst, Jeffrey C. Jackson, Michael J. Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using fourier analysis. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 253–262, 1994. doi:10.1145/195058.195147.

[10] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993. doi:10.1007/3-540-48329-2_24.

[11] Avrim Blum, Lisa Hellerstein, and Nick Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. *J. Comput. Syst. Sci.*, 50(1):32–40, 1995. URL: https://doi.org/10.1006/jcss.1995.1004, doi:10.1006/JCSS.1995.1004.

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

[13] Nader H. Bshouty. Exact learning from an honest teacher that answers membership queries. *Theor. Comput. Sci.*, 733:4–43, 2018. URL: https://doi.org/10.1016/j.tcs.2018.04.034, doi:10.1016/J.TCS.2018.04.034.

[14] Nader H. Bshouty and Lisa Hellerstein. Attribute-efficient learning in query and mistake-bound models. *J. Comput. Syst. Sci.*, 56(3):310–319, 1998. URL: https://doi.org/10.1006/jcss.1998.1571, doi:10.1006/JCSS.1998.1571.

[15] Harry Buhrman, David García-Soriano, and Arie Matsliah. Learning parities in the mistake-bound model. *Inf. Process. Lett.*, 111(1):16–21, 2010. URL: https://doi.org/10.1016/j.ipl.2010.10.009, doi:10.1016/J.IPL.2010.10.009.

[16] Qi Cheng and Daqing Wan. Complexity of decoding positive-rate primitive reed-solomon codes. *IEEE Trans. Inf. Theory*, 56(10):5217–5222, 2010. doi:10.1109/TIT.2010.2060234.

[17] Qi Cheng and Daqing Wan. A deterministic reduction for the gap minimum distance problem. *IEEE Trans. Inf. Theory*, 58(11):6935–6941, 2012. doi:10.1109/TIT.2012.2209198.

[18] Rodney G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM J. Comput.*, 29(2):545–570, 1999. doi:10.1137/S0097539797323571.

[19] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Trans. Inf. Theory*, 49(1):22–37, 2003. doi:10.1109/TIT.2002.806118.

[20] Vitaly Feldman. Attribute-efficient and non-adaptive learning of parities and DNF expressions. *J. Mach. Learn. Res.*, 8:1431–1460, 2007. URL: https://dl.acm.org/doi/10.5555/1314498.1314547, doi:10.5555/1314498.1314547.

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

[22] Elena Grigorescu, Lev Reyzin, and Santosh S. Vempala. On noise-tolerant learning of sparse parities and related problems. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann, editors, *Algorithmic Learning Theory - 22nd International Conference, ALT 2011, Espoo, Finland, October 5-7, 2011. Proceedings*, volume 6925 of *Lecture Notes in Computer Science*, pages 413–424. Springer, 2011. doi:10.1007/978-3-642-24412-4_32.

[23] David Guijarro, Víctor Lavín, and Vijay Raghavan. Exact learning when irrelevant variables abound. *Inf. Process. Lett.*, 70(5):233–239, 1999. doi:10.1016/S0020-0190(99)00063-0.

[24] David Guijarro, Jun Tarui, and Tatsuie Tsukiji. Finding relevant variables in PAC model with membership queries. In Osamu Watanabe and Takashi Yokomori, editors, *Algorithmic Learning Theory, 10th International Conference, ALT '99, Tokyo, Japan, December 6-8, 1999, Proceedings*, volume 1720 of *Lecture Notes in Computer Science*, page 313. Springer, 1999. doi:10.1007/3-540-46769-6_26.

[25] Thomas Hofmeister. An application of codes to attribute-efficient learning. In Paul Fischer and Hans Ulrich Simon, editors, *Computational Learning Theory, 4th European Conference, EuroCOLT '99, Nordkirchen, Germany, March 29-31, 1999, Proceedings*, volume 1572 of *Lecture Notes in Computer Science*, pages 101–110. Springer, 1999. doi:10.1007/3-540-49097-3_9.

[26] Adam Tauman Kalai, Yishay Mansour, and Elad Verbin. On agnostic boosting and parity learning. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 629–638. ACM, 2008. doi:10.1145/1374376.1374466.

[27] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998. doi:10.1145/293347.293351.

[28] Adam R. Klivans and Rocco A. Servedio. Toward attribute efficient learning of decision lists and parities. In John Shawe-Taylor and Yoram Singer, editors, *Learning Theory, 17th Annual Conference on Learning Theory, COLT 2004, Banff, Canada, July 1-4, 2004, Proceedings*, volume 3120 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 2004. doi:10.1007/978-3-540-27819-1_16.

[29] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th International-Workshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, pages 378–389, 2005. doi:10.1007/11538462_32.

[30] Daniele Micciancio. Locally dense codes. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 90–97. IEEE Computer Society, 2014. doi:10.1109/CCC.2014.17.

[31] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *CoRR*, abs/2401.03703, 2024. URL: https://doi.org/10.48550/arXiv.2401.03703, arXiv:2401.03703, doi:10.48550/ARXIV.2401.03703.

[32] Ryuhei Uehara, Kensei Tsuchida, and Ingo Wegener. Optimal attribute-efficient learning of disjunction, parity and threshold functions. In Shai Ben-David, editor, *Computational Learning Theory, Third European Conference, EuroCOLT '97, Jerusalem, Israel, March 17-19, 1997, Proceedings*, volume 1208 of *Lecture Notes in Computer Science*, pages 171–184. Springer, 1997. doi:10.1007/3-540-62685-9_15.

[33] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13:1–13:45, 2015. doi:10.1145/2728167.

[34] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. doi:10.1145/1968.1972.

[35] Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Inf. Theory*, 43(6):1757–1766, 1997. doi:10.1109/18.641542.

[36] Di Yan, Yu Yu, Hanlin Liu, Shuoyao Zhao, and Jiang Zhang. An improved algorithm for learning sparse parities in the presence of noise. *Theor. Comput. Sci.*, 873:76–86, 2021. URL: https://doi.org/10.1016/j.tcs.2021.04.026, doi:10.1016/J.TCS.2021.04.026.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il