Theoretical Computer Science 840 (2020) 249–256

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Improved learning of k-parities

Arnab Bhattacharyya [a], Ameet Gadekar [b,∗,1,2], Ninad Rajgopal [c,2]

[a] *Department of Computer Science & Automation, Indian Institute of Science, India*
[b] *Department of Computer Science, Aalto University, Finland*
[c] *Department of Computer Science, University of Oxford, United Kingdom*

A R T I C L E   I N F O

Article history:
Received 9 December 2018
Received in revised form 20 July 2020
Accepted 25 August 2020
Available online 27 August 2020

Keywords:
Learning sparse parities
Learning sparse parities with noise
Mistake bound model
PAC model
Learning k parities

A B S T R A C T

We consider the problem of learning k-parities in the online mistake-bound model: given a hidden vector $x \in \{0,1\}^n$ where the hamming weight of x is k and a sequence of "questions" $a_1, a_2, \cdots \in \{0,1\}^n$, where the algorithm must reply to each question with $\langle a_i, x\rangle \pmod 2$, what is the best trade-off between the number of mistakes made by the algorithm and its time complexity? We improve the previous best result of Buhrman et al. [3] by an $\exp(k)$ factor in the time complexity.
Next, we consider the problem of learning k-parities in the PAC model in the presence of random classification noise of rate $\eta \in (0, 1/2)$. Here, we observe that even in the presence of classification noise of non-trivial rate, it is possible to learn k-parities in time better than $\binom{n}{k/2}$, whereas the current best algorithm for learning noisy k-parities, due to Grigorescu et al. [9], inherently requires time $\binom{n}{k/2}$ even when the noise rate is polynomially small.

1. Introduction

By now, the "Parity Problem" of Blum, Kalai and Wasserman [4] has acquired widespread notoriety. The question is simple enough to be in our second sentence: in order to learn a hidden vector $x \in \{0,1\}^n$, what is the least number of random examples (a, ℓ) that need to be seen, where a is uniformly chosen from $\{0,1\}^n$ and $\ell = \sum_i a_i x_i \pmod 2$ with probability at least $1-\eta$? Information-theoretically, x can be recovered after only $O(n)$ examples, even if the noise rate η is close to $1/2$. But if we add the additional constraint that the running time of the learning algorithm be minimized, the barely sub-exponential running time of [4]'s algorithm, $2^{O(n/\log n)}$ still holds the record of being the fastest known for this problem (for any distribution)!

Learning parities with noise, abbreviated as LPN, is a central problem in theoretical computer science. It has incarnations in several different areas of computer science, including coding theory as the problem of decoding random binary linear codes and cryptography as the "learning with errors" problem that underlies lattice-based cryptosystems [14,7]. The learning with errors problem, often known as LWE, is in fact a generalization of LPN to finite fields of larger alphabets. Note that, LPN is a special case of LWE when the alphabet size is two.

∗ Corresponding author.
E-mail addresses: arnabb@iisc.ac.in (A. Bhattacharyya), ameet.gadekar@aalto.fi (A. Gadekar), ninad.rajgopal@cs.ox.ac.uk (N. Rajgopal).
[1] Partially supported by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 759557).
[2] Work partially completed while at IISc.

https://doi.org/10.1016/j.tcs.2020.08.025
0304-3975/© 2020 Elsevier B.V. All rights reserved.

In learning theory, the special case of the problem where the hidden vector x is known to be supported on a set of size k much smaller than n has great relevance. We refer to this problem as *learning k-parity with noise* or k-LPN. Feldman et al. [8] showed that learning k-juntas, as well as learning 2^k-term DNFs from uniformly random examples and variants of these problems in which the noise is adversarial instead of random, all reduce to the k-LPN problem. For the k-LPN problem, the current record is that of Grigorescu, Reyzin and Vempala [9] who showed a learning algorithm that succeeds with constant probability, takes $\binom{n}{k/2}^{1+(2\eta)^2+o(1)}$ time and uses $\frac{k\log n}{(1-2\eta)^2}\cdot\omega(1)$ samples. When the noise rate η is close to $1/2$, this running time is improved by an algorithm due to G. Valiant [16] that runs in time $n^{0.8k}\cdot\text{poly}(\frac{1}{1-2\eta})$. It is a wide open challenge to find a polynomial time algorithm for k-LPN for growing k or to prove a negative result.

Another outstanding challenge in machine learning is the problem of learning parities without noise in the "attribute-efficient" setting [6]. The algorithm is given access to a source of examples (a,ℓ) where a is chosen uniformly from $\{0,1\}^n$ and $\ell=\sum_i a_i x_i \pmod 2$ with no noise, and the question is to learn x using number of samples polynomial in the description length of x. Note that, the attribute-efficient setting is not an interesting question for general parities, as using poly(n) many samples is attribute-efficient and this is enough for Gaussian elimination. We focus on the case where x has sparsity $k\ll n$. Information-theoretically, of course, $O(k\log n)$ examples should be sufficient, as each linearly independent example reduces the number of consistent k-parities by a factor of 2. But the fastest known algorithm requiring $O(k\log n)$ samples runs in time $\tilde{O}(\binom{n}{k/2})$ [12], and it is open whether there exists a polynomial time algorithm for learning parities that is attribute-efficient, i.e. it uses poly$(k\log n)$ samples. Buhrman, García-Soriano and Matsliah [3] give the current best tradeoffs between the sample complexity and running time for learning parities in this noiseless setting and in fact, give an algorithm which achieves the current best running time for given sample complexity. Notice that with $O(n)$ samples, it is easy to learn the k-parity in polynomial time using Gaussian elimination.

Before we state our results formally, we give a quick and informal background here. For formal preliminaries, we refer to Section 2. Let PAR(k) denote the set of all k bit parity functions on n bit vectors. In this work, we consider two models of learning. First we consider the online mistake bound model, where in each round the learner has to guess the label of a given example. The mistake bound of a learner is the number of mistakes made by the learner in the worst case. Then, we consider the PAC model, where the examples are drawn randomly, and the learner has access to such examples and their labels. The learner should output a function that, with high probability, has high correlation with the target function. Finally, the k-LPN problem with noise rate η corresponds to the problem of PAC-learning PAR(k) under the uniform distribution, when the example source can mislabel examples with a rate $\eta\in(0,1/2)$.

1.1. Our results

We first study the noiseless setting and consider the mistake-bound model [13]. Our main technical result is an improved tradeoff between the sample complexity and runtime for learning parities.

We assume throughout, as in [3], that $k,t:\mathbb{N}\to\mathbb{N}$ are two functions such that they are constructible in quadratic time. We also use the shorthand notation $k=k(n)$ and $t=t(n)$ henceforth. Finally, we use $f(n)\ll g(n)$ to mean that $f(n)=o(g(n))$.

Theorem 1. *Let $k,t:\mathbb{N}\to\mathbb{N}$ be two functions such that $\log\log n\ll k(n)\ll t(n)\ll n$. Then for every $n\in\mathbb{N}$, there is an algorithm that learns PAR(k) in the mistake-bound model, with mistake bound at most $(1+o(1))\frac{kn}{t}+\log\binom{t}{k}$ and running time per round $e^{-k/4.01}\cdot\binom{t}{k}\cdot\tilde{O}\left((kn/t)^2\right)$.*

For comparison, let us quote the result of Buhrman et al.:

Theorem 2. *Let $k,t:\mathbb{N}\to\mathbb{N}$ be two functions satisfying $k(n)\le t(n)\le n$. For every $n\in\mathbb{N}$, there exists a deterministic algorithm that learns PAR(k) in the mistake-bound model, with the mistake bound $k\lceil\frac{n}{t}\rceil+\lceil\log\binom{t}{k}\rceil$ and running time per round $O\left(\binom{t}{k}(kn/t)^2\right)$.*

Thus, in the comparable regime, our Theorem 1 improves the runtime complexity of Theorem 2 by an exp(k) factor while its sample complexity remains the same upto constant factors. Note that as t approaches k, our algorithm requires $O(n)$ samples and takes poly(n) time which is the complexity of the Gaussian elimination approach. On the other hand, if $t=n/\log(n/k)$, our algorithm requires $O(k\log(n/k))$ samples and takes[3] $\exp(-k)\cdot\binom{n/k}{k}$ time (ignoring polynomial factors), compared to the trivial approach which explicitly keeps track of the subset of all the k-weight parities consistent with examples given so far and which requires $O(k\log(n/k))$ samples and takes $O(\binom{n}{k})$ time.

The mistake-bound model is stronger than the PAC model (in fact, strictly stronger assuming the existence of one-way functions [5]). As a consequence, we can get a PAC learning algorithm from the above theorem. There are standard conversion techniques which can be used to transform any mistake-bound algorithm into a PAC learning algorithm (over arbitrary distributions):

[3] By exp$(\cdot)$, we mean $2^{O(\cdot)}$.

Theorem 3 *([1,10,13]). Any algorithm $\mathcal{A}$ that learns a concept class $\mathcal{C}$ in the mistake-bound model with mistake bound m and running time t per round can be converted into an algorithm $\mathcal{A}'$ that PAC-learns $\mathcal{C}$ with sample complexity $O(\frac{1}{\varepsilon}m + \frac{1}{\varepsilon}\log\frac{1}{\delta})$, running time $O(\frac{1}{\varepsilon}mt + \frac{t}{\varepsilon}\log\frac{1}{\delta})$, approximation parameter ε, and confidence parameter δ.*

Using Theorem 3, we directly obtain the following corollary. In fact, since Theorem 3 produces a PAC-learner over any distribution, a statement of the form of Corollary 1 also holds for examples obtained from any distribution. Note that the poly(n) term in the running time below can be replaced by $O(n^4)$.

Corollary 1. *Let $k, t : \mathbb{N} \to \mathbb{N}$ be two functions satisfying $\log\log n \ll k(n) \ll t(n) \ll n$. For any $\delta > 0$, there is an algorithm that learns the concept class of k-parities on n variables with confidence parameter δ, using $O(kn/t + \log\binom{t}{k} + \log(1/\delta))$ uniformly random examples and $e^{-k/4.01}\binom{t}{k} \cdot \text{poly}(n) \cdot \log(1/\delta)$ running time.*[4]

We next examine the noisy setting. Here, our contribution is a simple, general observation that does not seem to have been explicitly made before. Let PAR(k) represent concept class of parities of Hamming weight k.

Theorem 4. *Given an algorithm $\mathcal{A}$ that learns PAR(k) over the uniform distribution with confidence parameter δ using $s(\delta)$ samples and running time $t(\delta)$, there is an algorithm $\mathcal{A}'$ that solves the k-LPN problem with noise rate $\eta \in (0, 1/3)$, using $O(s(\delta/2)\log(4/\delta))$ examples and running time $\exp(O(H(3\eta/2) \cdot s(\delta/2))) \cdot \log(4/\delta) \cdot (t(\delta/2) + s(\delta/2))$ and with confidence parameter δ.*

In the above, $H : [0, 1] \to [0, 1]$ denotes the binary entropy function $H(p) = p\log_2\frac{1}{p} + (1-p)\log_2\frac{1}{1-p}$. The main conceptual message carried by Theorem 4 is that improving the sample complexity for efficient learning of noiseless parity improves the running time for learning of noisy parity. For instance, if we use Spielman's algorithm as $\mathcal{A}$, reported in [12], that learns k-parity using $O(k\log n)$ samples and $O(\binom{n}{k/2})$ running time, we immediately get the following:

Corollary 2. *For any $\eta \in (0, 1/3)$ and constant confidence parameter, there is an algorithm for k-LPN with sample complexity $O(k\log n)$ and running time $O\left(\binom{n}{k/2}^{1+O(H(1.5\eta))}\right)$.*

For comparison, consider the current best result of [9]:

Theorem 5 *(Theorem 5 of [9]). For any $\varepsilon, \delta, \eta \in (0, 1/2)$, and distribution $\mathcal{D}$ over $\{0, 1\}^n$, the k-LPN problem over $\mathcal{D}$ with noise rate η can be solved using $\frac{k\log(n/\delta)\omega(1)}{\varepsilon^2(1-2\eta)^2}$ samples in time $\frac{\log 1/\delta}{\varepsilon^2(1-2\eta)^2} \cdot \binom{n}{k/2}^{1+(\frac{\eta}{\varepsilon+\eta-2\varepsilon\eta})^2+o(1)}$, where ε and δ are the approximation and confidence parameters respectively.*

This result has runtime $\binom{n}{k/2}^{1+4\eta^2+o(1)}$ and sample complexity $\omega(k\log n)$. In the regime under consideration, our algorithm's runtime has a worse exponent but an asymptotically better sample complexity.

The result of [9] requires $\binom{n}{k/2}$ time regardless of how small η is. We show via Theorem 4 and Corollary 1 for the uniform distribution, that it is possible to break the $\binom{n}{k/2}$ barrier when η is a small enough function of n.

Corollary 3. *Suppose $k(n) = n/f(n)$ for some function $f : \mathbb{N} \to \mathbb{N}$ for which $f(n) \ll n/\log\log n$, and suppose $\eta(n) = o(\frac{1}{((f(n))^\alpha \log n)})$ for some $\alpha \in [1/2, 1)$. Then, for constant confidence parameter, there exists an algorithm for k-LPN with noise rate η with running time $O\left(e^{-k/4.01+o(k)} \cdot \binom{n}{k}^{1-\alpha} \cdot \text{poly}(n)\right)$ and sample complexity $O(k(f(n))^\alpha)$.*

We note that because of the results of Feldman et al. [8], the above results for k-LPN also extend to the setting where the example source adversarially mislabels examples instead of randomly but with the same rate η.

1.2. Our techniques

We first give an algorithm to learn parities in the noiseless setting in the mistake bound model. We use the same approach as that of [3] (which was itself inspired by [2]). The idea is to consider a family $\mathcal{S}$ of subsets of $\{0, 1\}^n$ such that the hidden k-sparse vector is contained inside one of the elements of $\mathcal{S}$. We maintain this invariant throughout the algorithm. Now, each time an example comes, it specifies a halfspace H of $\{0, 1\}^n$ inside which the hidden vector is lying. So, we can update $\mathcal{S}$ by taking the intersection of each of its elements with H. If we can ensure that the set of points covered by the elements of $\mathcal{S}$ is decreasing by a constant factor at every round, then after $O(\log\sum_{S\in\mathcal{S}}|S|)$ examples, the

[4] The "4.01" can be replaced by any constant more than 4.

hidden vector is learned. The runtime is determined by the number of sets in $\mathcal{S}$ times the cost of taking the intersection of each set with a halfspace.

One can think of the argument of Buhrman et al. [3] as essentially initializing $\mathcal{S}$ to be the set of all $\binom{n}{k}$ subspaces spanned by k standard basis vectors. The intersections of these subspaces with a halfspace can be computed efficiently by Gaussian elimination. Our idea is to reduce the number of sets in $\mathcal{S}$. Note that we can afford to make the size of each set in $\mathcal{S}$ larger by some factor C because this only increases the sample complexity by an additive $\log C$. Our approach is (essentially) to take $\mathcal{S}$ to be a random collection of subspaces spanned by αk standard basis vectors, where $\alpha > 1$ is a sufficiently large constant. We show that it is sufficient for the size of $\mathcal{S}$ to be smaller than $\binom{n/\alpha}{k}$ by a factor that is exponential in k, so that the running time is also improved by the same factor. Moreover, the sample complexity increases by only a lower-order additive term.

Our second main contribution is a reduction from noiseless parity learning to noisy parity learning. The algorithm is a simple exhaustive search which guesses the location of the mis-labelings, corrects those labels, applies the learner for noiseless parity and then verifies whether the output hypothesis matches the examples by drawing a few more samples. Surprisingly, this seemingly immediate algorithm allows us to devise the first algorithm which has a better running time than $\binom{n}{k/2}$ in the presence of a non-trivial amount of noise.

2. Preliminaries

Let $\mathrm{PAR}(k)$ be the class of all vectors $f \in \{0,1\}^n$ of Hamming weight k. So, $|\mathrm{PAR}(k)| = \binom{n}{k}$. With each vector $f \in \mathrm{PAR}(k)$, we associate a parity function $f : \{0,1\}^n \to \{0,1\}$ defined by $f(a) = \sum_{i=1}^{n} x_i a_i \pmod 2$.

Let $\mathcal{C}$ be a concept class of Boolean functions on n variables, such as $\mathrm{PAR}(k)$. We discuss two models of learning in this work. One is Littlestone's *online mistake bound* model [13]. Here, learning proceeds in a series of rounds, where in each round, the learner is given an unlabeled boolean example $a \in \{0,1\}^n$ by an oracle and must predict the value $f(a)$ of an unknown target function $f \in \mathcal{C}$. Once the learner predicts the value of $f(a)$, the true value of $f(a)$ is revealed to the learner by the oracle. The *mistake bound* of a learning algorithm is the worst-case number of mistakes that the algorithm makes over all sequences of examples and all possible target functions $f \in \mathcal{C}$.

The second model of learning we consider is Valiant's famous *PAC model* [15] of learning from random examples. Here, for an unknown target function $f \in \mathcal{C}$, the learner has access to a source of examples $(a, f(a))$ where a is chosen independently from a distribution $\mathcal{D}$ on $\{0,1\}^n$. A learning algorithm is said to PAC-learn $\mathcal{C}$ with *sample complexity* s, *running time* t, *approximation parameter* ε and *confidence parameter* δ if for all distributions $\mathcal{D}$ and all target functions $f \in \mathcal{C}$, the algorithm draws at most s samples from the example source, runs for time at most t and outputs a function f^* such that, with a probability at least $1-\delta$:

$$\Pr_{a \leftarrow \mathcal{D}}[f(a) \neq f^*(a)] < \varepsilon$$

Often in this paper (e.g., all of the Introduction), we consider PAC-learning over the uniform distribution, in which case $\mathcal{D}$ is fixed to be uniform on $\{0,1\}^n$. Notice that for learning $\mathrm{PAR}(k)$ over the uniform distribution, we can take $\varepsilon = \frac{1}{2}$ because any two distinct parities differ on half of $\{0,1\}^n$.

The k-LPN problem with noise rate η, mentioned in the introduction, corresponds to the problem of PAC-learning $\mathrm{PAR}(k)$ under the uniform distribution, when the example source can mislabel examples with a rate $\eta \in (0, 1/2)$. More generally, one can study the k-LPN *problem over* $\mathcal{D}$, an arbitrary distribution. In fact, Theorem 5 [9] shows their algorithm for any arbitrary distribution $\mathcal{D}$.

3. In the absence of noise

We re-state Theorem 1, which is the main result of this section.

Theorem 1. *Let $k, t : \mathbb{N} \to \mathbb{N}$ be two functions such that $\log\log n \ll k(n) \ll t(n) \ll n$. Then for every $n \in \mathbb{N}$, there is an algorithm that learns $\mathrm{PAR}(k)$ in the mistake-bound model, with mistake bound at most $(1+o(1))\frac{kn}{t} + \log\binom{t}{k}$ and running time per round $e^{-k/4.01} \cdot \binom{t}{k} \cdot \tilde{O}\left((kn/t)^2\right)$.*

For comparison, we quote the relevant result of [3] in the mistake-bound model.

Theorem 2 (recalled) [Theorem 2.1 of [3]]. *Let $k, t : \mathbb{N} \to \mathbb{N}$ be two functions such that $k(n) \le t(n) \le n$. Then for every $n \in \mathbb{N}$, there is a deterministic algorithm that learns $\mathrm{PAR}(k)$ in the mistake-bound model, with mistake bound at most $k\lceil \frac{n}{t} \rceil + \log\binom{t}{k}$ and running time per round $\binom{t}{k} \cdot O((kn/t)^2)$.*

Note that their mistake bound is better by a lower-order term which we do not see how to avoid in our setup. This slack is not enough though to recover Theorem 1 from Theorem 2: dividing t by C roughly multiplies the sample complexity by

C and divides the running time by C^k in [3]'s algorithm, whereas in our algorithm, dividing t by C roughly multiplies the sample complexity by C and divides the running time by $(1.28C)^k$.

3.1. The algorithm

Let $f \in \{0,1\}^n$ be the hidden vector of sparsity k that the learning algorithm is trying to learn. Let $e = \{e_1, e_2, \cdots, e_n\}$ be the set of standard basis of the vector space $\{0,1\}^n$.

Let α be a large constant we set later, and let $T = \alpha t$. Note that $T \ll n$. We define an arbitrary partition $\pi = C_1, C_2, \cdots, C_T$ on the set e into T parts, each of size at most $\lceil n/T \rceil$. Next, let $S_1, \ldots, S_m \subset [T]$ be m random subsets of $[T]$, each of size αk. We choose m to ensure the following:

Claim. *If $m = \tilde{O}\left(\frac{\binom{T}{\alpha k}}{\binom{T-k}{\alpha k-k}}\right)$, then with nonzero probability, for every set $A \subset [T]$ of size k, $A \subset S_i$ for some $i \in [m]$.*

Proof. This follows from the simple observation that for any fixed $i \in [m]$, $\mathbf{Pr}[A \subset S_i] = \binom{T-k}{\alpha k-k}/\binom{T}{\alpha k}$, and so,

$$\mathbf{Pr}[\forall i \in [m], A \not\subset S_i] = \left(1 - \binom{T-k}{\alpha k-k}\Big/\binom{T}{\alpha k}\right)^m \le e^{-m\binom{T-k}{\alpha k-k}/\binom{T}{\alpha k}}$$

Choosing $m = 2\frac{\binom{T}{\alpha k}}{\binom{T-k}{\alpha k-k}} \log\binom{T}{k}$ and applying the union bound finishes the proof. □

We fix some choice of $S_1, \ldots, S_m \subset [T]$ that satisfies the conclusion of above claim for what follows. In fact, the rest is exactly [3]'s algorithm, which we reproduce for completeness.

For every $i \in [m]$, let $M_i \subset \{0,1\}^n$ be the span of $\bigcup_{j \in S_i} C_j$. Note that $\left|\bigcup_{j \in S_i} C_j\right| \le \alpha k \lceil n/T \rceil \le \alpha k \cdot \left(\frac{n}{T} + 1\right) = \frac{kn}{t} + \alpha k = (1 + o(1))kn/t$, as $t \ll n$ and α is a constant. So, M_i is a linear subspace containing at most $2^{(1+o(1))kn/t}$ points.

Note that every $f \in \{0,1\}^n$ with $|f| = k$ is contained in some M_i. This is simply because every set of k standard basis vectors is contained in the union of at most k of the T parts in the partition π, and by Claim 3.1, every subset of $[T]$ of size k is contained in some S_i.

Initially, the unknown target vector f can be in any of the M_i's. Consider what happens when the learner sees an example $a \in \{0,1\}^n$ and a label $y \in \{0,1\}$. For $i \in [m]$, let $M_i(a, y) = \{v \in M_i : v(a) = y\}$. $M_i(a, y)$ may be of size 0, $|M_i|$ or $|M_i|/2$. Note that the size of $M_i(a, y)$ can be efficiently found using Gaussian elimination.

We are now ready to describe the algorithm:

- **Initialization:** The learning algorithm begins with a set of affine spaces $N_i, i \in [m]$ represented by a system of linear equations. Initialize the affine spaces $N_i = M_i$ for all $i \in [m]$.
- **On receiving an example** $a \in \{0,1\}^n$ **from the oracle:** Predict its label $\hat{y} \in \{0,1\}$ such that $\sum_{i \in [m]} |N_i(a, \hat{y})| \ge \sum_{i \in [m]} |N_i(a, 1 - \hat{y})|$.
- **On receiving the answer from the oracle** $y = f(a)$**:** Update N_i to $N_i(a, y)$ for each $i \in [m]$.
- **Termination:** The algorithm terminates when $|\cup_{i \in [m]} N_i| \leqslant 1$.

3.2. Analysis

Before we analyze the algorithm, we first establish a combinatorial claim that is the crux of our improvement:

Lemma 1. *If α is a large enough constant,*

$$\frac{\binom{T}{\alpha k}}{\binom{T-k}{\alpha k-k}} \le e^{-k/4.01} \cdot \binom{t}{k}$$

Proof.

$$\begin{aligned}\frac{1}{\binom{t}{k}} \cdot \frac{\binom{T}{\alpha k}}{\binom{T-k}{\alpha k-k}} &= \prod_{i=0}^{k-1} \frac{k-i}{t-i} \cdot \frac{T-i}{\alpha k-i} \\ &= \prod_{i=0}^{k-1} \frac{\alpha t-i}{\alpha k-i} \cdot \frac{k-i}{t-i} \\ &= \prod_{i=1}^{k-1} \left(1 - \frac{i\left(1-\frac{1}{\alpha}\right)\left(\frac{1}{k-i} - \frac{1}{t-i}\right)}{1 + \frac{i}{k-i}\left(1-\frac{1}{\alpha}\right)}\right)\end{aligned}$$

$$\leq \prod_{i=1}^{k-1}\left(1-\frac{0.999}{1+\frac{k-i}{i}\frac{\alpha}{\alpha-1}}\right)$$

where the equalities are routine calculation and the inequality is using that $k(n) \ll t(n)$. Each individual term in the product is strictly less than 1. So, the above is bounded by[5]:

$$\begin{aligned}&\leq \prod_{i=k/(2-\varepsilon)}^{k-1}\left(1-\frac{0.999}{1+\frac{k-i}{i}\frac{\alpha}{\alpha-1}}\right)\\ &\leq \left(1-\frac{0.999}{1+(1-\varepsilon)\frac{\alpha}{\alpha-1}}\right)^{\frac{1-\varepsilon}{2-\varepsilon}k}\\ &\leq \exp\left(-\lg e\cdot\frac{0.999(1-\varepsilon)}{(2-\varepsilon)(1+(1-\varepsilon)\frac{\alpha}{\alpha-1})}k\right)\leq e^{-k/4.01}\end{aligned}$$

for a small enough constant $\varepsilon > 0$ and large enough constant $\alpha > 1$. □

Proof of Theorem 1. Fix α to be a constant that makes the conclusion of Lemma 1 true.

We first check that the invariant is maintained throughout the algorithm that $f \in \cup_{i\in[m]} N_i$. This holds at initiation by the argument given earlier. After that, obviously, if $f \in N_i$, then $f \in N_i(a, f(a))$ for any $a \in \{0,1\}^n$, and so the invariant holds. Therefore, if the algorithm terminates, it will find the hidden vector f and return it as the solution. The rate of convergence is precisely captured by the number of mistakes learning algorithm makes, which we describe next.

Mistake Bound. Notice that when the algorithm begins, the sum of the sizes of all the affine spaces, $\sum_i |N_i| \leq \tilde{O}\left(\frac{\binom{T}{\alpha k}}{\binom{T-k}{\alpha k-k}}\right)2^{(1+o(1))kn/t}$. Now whenever the learner makes a mistake by predicting $\hat{y} \neq y$, the size of all affine spaces $\sum_i |N_i|$ reduces by a factor of at least 2. This is due to the definition of $\hat{y}$ and the fact that $|N_i(a,\hat{y})| + |N_i(a, 1-\hat{y})| = |N_i|$.

Hence, using Lemma 1, after at most

$$\log\left(\sum_i |N_i|\right) \leq \log\left[\tilde{O}\left(\frac{\binom{T}{\alpha k}}{\binom{T-k}{\alpha k-k}}\right)2^{(1+o(1))kn/t}\right] \leq (1+o(1))kn/t + \log\binom{t}{k} - \Omega(k) + O\left(\log\log\binom{t}{k}\right)$$

mistakes, the size of $\cup_{i\in[m]} N_i$ will decrease to 1, which by the invariant above will imply that $\cup_{i\in[m]} N_i = \{f\}$, and hence the learner makes no more mistakes. Since we assume $k \gg \log\log n$ and $t \ll n$, we can bound the number of mistakes by: $(1+o(1))kn/t + \log\binom{t}{k}$.

Running Time. We analyze the running time of the learner for each round. At each round, for a question $a \in \{0,1\}^n$, we need to compute $|N_i(a,0)|$ and $|N_i(a,1)|$ as well as store a representation of the updated N_i. Now, since for each N_i is spanned by at most $\ell = (1+o(1))kn/t$ basis vectors, we can treat each N_i as a linear subspace in $\{0,1\}^\ell$. $N_i(a,0)$ and $N_i(a,1)$ can be computed by performing one step of Gaussian elimination on a system of linear equations involving ℓ variables, which takes $O(\ell^2)$ time. Thus, the total running time is $O(m\ell^2)$, which using Lemma 1 is exactly the bound claimed in Theorem 1. □

4. In the presence of noise

Recall the k-LPN problem. In this section, we show a reduction from k-LPN to noiseless learning of PAR(k) and its applications.

4.1. The reduction

We focus on the case when the noise rate η is bounded by a constant less than $\frac{1}{3}$.

Theorem 4 (recalled). *Given an algorithm $\mathcal{A}$ that learns* PAR(*k*) *over the uniform distribution with confidence parameter δ using $s(\delta)$ samples and running time $t(\delta)$, there is an algorithm $\mathcal{A}'$ that solves the k-*LPN *problem with noise rate $\eta \in (0, 1/3)$, using $O(s(\delta/2)\log(4/\delta))$ examples and running time $\exp(O(H(3\eta/2)\cdot s(\delta/2)))\cdot\log(4/\delta)\cdot(t(\delta/2)+s(\delta/2))$ and with confidence parameter δ.*

[5] In the second last inequality, by $\exp(\cdot)$, we mean $e^{(\cdot)}$.

As mentioned in Section 1 the idea is very simple: The algorithm searches exhaustively the potential location of the mis-labelings. Next, for each guess of mis-labelings, it first corrects those labels and then applies our learner for noiseless parity. Finally, it verifies whether the output hypothesis matches the examples by drawing a few more samples. Formally, let $\mathcal{A}(\delta)$ be a PAC-learning algorithm over the uniform distribution for PAR(k) of length n with confidence parameter δ that draws $s(\delta)$ examples and runs in time $t(\delta)$. Below is our algorithm Noisy for k-LPN. Here, H denotes the binary entropy function $p \mapsto p\log_2(1/p) + (1-p)\log_2(1/(1-p))$.

Algorithm Noisy(δ, η).

```
1: $\mathcal{X} = \phi$
2: for $\rho = 1$ to $3\log(4/\delta)$ do
3:     Draw $s' = s(\delta/2)$ random samples $(a_1, \ell_1), \ldots, (a_{s'}, \ell_{s'}) \in \{0,1\}^n \times \{0,1\}$.
4:     for all $S \subset [s'], |S| \le \frac{3}{2}\eta s'$ do
5:         for $i \in [s']$ do
6:             if $i \in S$ then $\tilde{\ell}_i \leftarrow 1 - \ell_i$
7:             else $\tilde{\ell}_i \leftarrow \ell_i$
8:             end if
9:         end for
10:        $x_S \leftarrow \mathcal{A}(\delta/2)$ applied to examples $(a_1, \tilde{\ell}_1), \ldots, (a_{s'}, \tilde{\ell}_{s'})$.
11:        $\mathcal{X} = \mathcal{X} \cup x_S$
12:    end for
13: end for
14: Draw $s'' = 600\big(s' \cdot H(3\eta/2) + \log\big(\frac{24\log(4/\delta)}{\delta}\big)\big)$ random samples $(b_1, m_1), \ldots, (b_{s''}, m_{s''}) \in \{0,1\}^n \times \{0,1\}$
15: $x_{S^*} \leftarrow \arg\max_{x_S \in \mathcal{X}} |\{i \in [s''] : \langle b_i, x_S\rangle = m_i\}|$
16: return $x_{S^*}$
```

Proof of Theorem 4.

Lemma 2. *The sample complexity of* Noisy *is* $s' \cdot 3\log(4/\delta) + s'' = O(s(\delta/2)\log(4/\delta))$.

Proof. Immediate. □

Lemma 3. *The running time of* Noisy *is* $2^{O(H(3\eta/2)s(\delta/2))} \cdot \log(4/\delta) \cdot (t(\delta/2) + s(\delta/2))$.

Proof. We use the standard estimate $\sum_{i=0}^{\alpha x}\binom{x}{i} \le 2^{H(\alpha)x}$ for $\alpha \le \frac{1}{2}$. The bound is then immediate. □

Lemma 4. *If x is the hidden vector and x^* is output by* Noisy(δ)*, then with probability at least $1-\delta$, $x^* = x$.*

Proof. Let $T = \{i \in [s'] : \langle a_i, x\rangle \ne \ell_i\}$ be the subset of the s' samples drawn in line 3 that are mislabeled by the example source. By Markov's inequality:

$$\mathbf{Pr}[|T| > 3\eta s'/2] \le 2/3$$

Thus, if we repeat step 3, $3\log(4/\delta)$ times, then with probability at least $1-\delta/4$, it is true that $|T| \le 3\eta s'/2$ in some round. If $|T| \le 3\eta s'/2$, we have with probability at least $1-\delta/2$, $x_T = x$. Thus, for any $i \in [s'']$, $\mathbf{Pr}_{b_i}[\langle x_T, b_i\rangle \ne m_i] \le \eta$. On the other hand, for all $x_S \ne x_T$, $\mathbf{Pr}_{b_i}[\langle x_S, b_i\rangle \ne \langle x, b_i\rangle] = 1/2$, and so $\mathbf{Pr}_{b_i}[\langle x_S, b_i\rangle \ne m_i] = 1/2$ as the noise is random. Again, using Chernoff bounds,

$$Pr[\exists S \ne T \text{ s.t.} |\{i \in [s''] : \langle b_i, x_S\rangle \ne m_i\}| \le 5s''/12] \le 2^{H(3\eta/2)s'} \cdot 3\log(4/\delta) \cdot e^{-s''/450} < \frac{\delta}{8}$$

On the other hand, for x_T itself, $\mathbf{Pr}[|\{i \in [s''] : \langle b_i, x_T\rangle \ne m_i\}| > 5s''/12] < \frac{\delta}{8}$ by a similar use of Chernoff bounds. So, in all, with probability at least $1-\delta$, x_T will be returned in step 15. □ □

When the noise rate η is more than 1/3, a similar reduction can be given by adjusting the parameters accordingly. For example, for any $\eta < 1/2$, performing the for loop of step 2, $\frac{\log(4/\delta)}{(1-2\eta)}$ times, with $|S| \le s'/2$ in step 4, we would still have that with probability at least $1-\delta/4$, it is true that $|T| \le s'/2$. Further, when the distribution is arbitrary, $\mathcal{A}$ is invoked with a smaller approximation parameter than the one given to Noisy so that the filtering step in line 14 works.

4.2. Applications

An immediate application of Theorem 4 is obtained by letting $\mathcal{A}$ be the current fastest known attribute-efficient algorithm for learning PAR(k), the algorithm due to Spielman[6] [12] that requires $O(k\log n)$ samples and takes $O(\binom{n}{k/2})$ time (for constant confidence parameter δ). (We ignore the confidence parameter in this section for simplicity.)

Corollary 2 (recalled). *For any $\eta \in (0, 1/3)$ and constant confidence parameter, there is an algorithm for k-LPN with sample complexity $O(k\log n)$ and running time $O\left(\binom{n}{k/2}^{1+O(H(1.5\eta))}\right)$.*

Proof. Immediate from Theorem 4. $\square$

Our next application of Theorem 4 uses our improved PAR(k) learning algorithm from Section 3.

Corollary 3 (recalled). *Suppose $k(n) = n/f(n)$ for some function $f : \mathbb{N} \to \mathbb{N}$ for which $f(n) \ll n/\log\log n$, and suppose $\eta(n) = o(1/((f(n))^{\alpha}\log n))$ for some $\alpha \in [1/2, 1)$. Then, for constant confidence parameter, there exists an algorithm for k-LPN with noise rate η with running time $O\left(e^{-k/4.01+o(k)} \cdot \binom{n}{k}^{1-\alpha} \cdot \mathrm{poly}(n)\right)$ and sample complexity $O(k(f(n))^{\alpha})$.*

Proof. Let $\mathcal{A}$ be the algorithm of Corollary 1 with $t(n) = \lceil n/(f(n))^{\alpha}\rceil$. The running time of $\mathcal{A}$ is $e^{-k/4.01} \cdot \binom{n}{k}^{1-\alpha} \cdot \mathrm{poly}(n)$ and its sample complexity is $O(k \cdot f(n))^{\alpha})$. Now, applying Theorem 4, we see that since $H(1.5\eta) = o((f(n))^{-\alpha})$, the running time for Noisy is only a $2^{o(k)}$ factor times the running time of $\mathcal{A}$. This yields our desired result. $\square$

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Dana Angluin, Queries and concept learning, Mach. Learn. 2 (4) (1988) 319–342.
[2] Noga Alon, Rina Panigrahy, Sergey Yekhanin, Deterministic approximation algorithms for the nearest codeword problem, in: Irit Dinur, Klaus Jansen, Joseph Naor, José Rolim (Eds.), Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, in: Lecture Notes in Computer Science, vol. 5687, Springer, Berlin, Heidelberg, 2009, pp. 339–351.
[3] Harry Buhrman, David García-Soriano, Arie Matsliah, Learning parities in the mistake-bound model, Inf. Process. Lett. 111 (1) (2010) 16–21.
[4] Avrim Blum, Adam Kalai, Hal Wasserman, Noise tolerant learning, the parity problem, and the statistical query model, J. ACM 50 (4) (2003) 506–519.
[5] Avrim Blum, Separating distribution-free and mistake-bound learning models over the Boolean domain, SIAM J. Comput. 23 (5) (1994) 990–1000.
[6] Avrim Blum, On-line algorithms in machine learning, in: Workshop on On-Line Algorithms, Dagstuhl, Springer, 1996, pp. 305–325.
[7] Zvika Brakerski, Vinod Vaikuntanathan, Efficient fully homomorphic encryption from (standard) LWE, in: Proc. 52nd Annual IEEE Symposium on Foundations of Computer Science, 2011, pp. 97–106.
[8] Vitaly Feldman, Parikshit Gopalan, Subhash Khot, Ashok Kumar Ponnuswami, On agnostic learning of parities, monomials, and halfspaces, SIAM J. Comput. 39 (2) (2009) 606–645.
[9] Elena Grigorescu, Lev Reyzin, Santosh Vempala, On noise-tolerant learning of sparse parities and related problems, in: Algorithmic Learning Theory, Springer, 2011, pp. 413–424.
[10] David Haussler, Space efficient learning algorithms, Technical Report UCSC-CRL-88-2, University of California at Santa Cruz, 1988.
[11] Nicholas J. Hopper, Manuel Blum, Secure human identification protocols, in: Advances in Cryptology–ASIACRYPT 2001, Springer, 2001, pp. 52–66.
[12] Adam R. Klivans, Rocco A. Servedio, Toward attribute efficient learning of decision lists and parities, J. Mach. Learn. Res. 7 (2006) 587–602.
[13] Nick Littlestone, From on-line to batch learning, in: Proc. 2nd Annual ACM Workshop on Computational Learning Theory, 1989, pp. 269–284.
[14] Oded Regev, On lattices, learning with errors, random linear codes, and cryptography, J. ACM 56 (6) (2009) 1–40.
[15] Leslie G. Valiant, A theory of the learnable, Commun. ACM 27 (11) (1984) 1134–1142.
[16] Gregory Valiant, Finding correlations in subquadratic time, with applications to learning parities and juntas, in: Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science, IEEE, 2012, pp. 11–20.

[6] Though a similar algorithm was also proposed by Hopper and Blum [11].