

Learning Parities in the Mistake-Bound Model

Harry Buhrman David García-Soriano Arie Matsliah
{buhrman, david, ariem}@cwi.nl
CWI Amsterdam

Abstract

We study the problem of learning parity functions that depend on at most k variables (k -parities) attribute-efficiently in the mistake-bound model. We design a simple, deterministic, polynomial-time algorithm for learning k -parities with mistake bound $O(n^{1-\frac{1}{k}})$. This is the first polynomial-time algorithm to learn $\omega(1)$ -parities in the mistake-bound model with mistake bound $o(n)$.

Using the standard conversion techniques from the mistake-bound model to the PAC model, our algorithm can also be used for learning k -parities in the PAC model. In particular, this implies a slight improvement over the results of Klivans and Servedio [KS04] for learning k -parities in the PAC model.

We also show that the $\tilde{O}(n^{k/2})$ time algorithm from [KS04] that PAC-learns k -parities with sample complexity $O(k \log n)$ can be extended to the mistake-bound model.

1 Introduction

The study of attribute-efficient learning was initiated in the on-line mistake-bound model, which was introduced by Littlestone in [Lit88]. In this model learning proceeds in rounds, where in each round the “teacher” provides an unlabelled example $x \in \{0, 1\}^n$, and the “learner” must predict the value $f(x)$ of the unknown target function f . Then the learner is given the true value of $f(x)$, according to which it can update its hypothesis. The *mistake bound* of the learner, with respect to a target function f , is the worst-case number of mistakes that it makes over all (arbitrary, possibly infinite) sequences of examples. The mistake bound on a concept class \mathcal{C} (of functions that map $\{0, 1\}^n$ to $\{0, 1\}$) is the maximum of the mistake bounds taken over all possible target functions $f \in \mathcal{C}$.

Given a concept class \mathcal{C} and a Boolean function $f \in \mathcal{C}$, let $\text{size}(f)$ denote the description length of f , under some reasonable encoding scheme. A learning algorithm A for \mathcal{C} in the mistake-bound model is *attribute-efficient* if the mistake bound of A on any $f \in \mathcal{C}$ is bounded by a uniform polynomial in $\text{size}(f)$. Similarly, an algorithm A for learning \mathcal{C} in Valiant’s PAC model is attribute-efficient if the sample size required by A to learn target function $f \in \mathcal{C}$ is polynomial in $\text{size}(f)$.

One of the long-standing open questions in both the mistake-bound and the PAC learning models is whether parities can be learned attribute-efficiently in polynomial time [Blu96].

There are several standard conversion techniques (see e.g. [Ang88, Lit89]) which can be used to transform any mistake-bound algorithm into a PAC learning algorithm. These transformations preserve the running time of the mistake-bound algorithm, and the sample size required by the PAC algorithm is equal to the mistake bound, up to constant factors that depend on its approximation and confidence parameters.

Theorem 1.1 ([Ang88, Lit89]) *Any algorithm A that learns \mathcal{C} in the mistake-bound model with mistake bound m and maximum running time per round t can be converted into an algorithm A' that learns \mathcal{C} in the PAC model using a sample set of size $O(\frac{1}{\epsilon}m + \frac{1}{\epsilon} \log \frac{1}{\delta})$ and running time $O(\frac{1}{\epsilon}mt \log m + \frac{t}{\epsilon} \log \frac{1}{\delta})$, where ϵ and δ are the approximation and confidence parameters of A' .*

These conversion techniques imply that positive results for mistake-bound learning, in particular those given in this paper, directly yield corresponding positive results for PAC learning. We mention here that for the other direction no such conversion is known. In fact, Blum [Blu94] proved that under widely held assumptions (namely, the existence of one-way functions) the mistake-bound model is strictly harder than the PAC model.

2 Our results and related work

In this paper we show the first non-trivial algorithm for learning parities in the mistake bound model. In order to state our results formally we need several definitions. With each vector $\tilde{f} \in \{0, 1\}^n$ we associate a parity function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ defined by $f(x) = \langle \tilde{f}, x \rangle \triangleq \sum_{i=1}^n \tilde{f}_i x_i \pmod{2}$ for all $x \in \{0, 1\}^n$. Learning a parity function can thus be thought of as learning the corresponding n -bit vector \tilde{f} . With a slight abuse of notation, from now on we will denote by f both the parity function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and its corresponding vector $\tilde{f} \in \{0, 1\}^n$; in particular we may e.g. write $|f|$ for the Hamming weight of \tilde{f} . We will also refrain from explicitly mentioning n throughout the paper. The concept class $\text{PAR}(k)$ is defined as the class of all parity functions of Hamming weight at most k . The description length of any function $f \in \text{PAR}(k)$ is $O(k \log n)$, and thus ideally we would like to have $\text{poly}(n)$ -time algorithms which learn $\text{PAR}(k)$ with a mistake bound (respectively sample size) of $\text{poly}(k \log n)$. This would correspond to attribute-efficient learning as defined above.

It is well known that, in exponential time, $\text{PAR}(k)$ can be learned attribute-efficiently in the mistake-bound model (and hence in the PAC model too). A simple algorithm with mistake bound at most $k \log n$ is the *halving algorithm*. It maintains a set $\mathcal{H} \subseteq \text{PAR}(k)$ of candidate parity functions, and given an example x , it predicts $\text{majority}\{h(x) : h \in \mathcal{H}\}$. Whenever a mistake is made, all “wrong” candidates (of which there are at least $|\mathcal{H}|/2$) are removed from \mathcal{H} . If initially the set \mathcal{H} was set to be $\text{PAR}(k)$, then after at most $\log |\text{PAR}(k)| \leq k \log n$ mistakes the function f is learned. The running time of the halving algorithm is dominated by the time needed to compute the predicate $\text{majority}\{h(x) : h \in \mathcal{H}\}$. A naive computation of this predicate requires $|\text{PAR}(k)| \geq \binom{n}{k}$ steps, and in fact the running time of all such known algorithms is super-polynomial for any $k = \omega(1)$.

On the other hand, with a mistake bound of n (respectively, a sample set of size $O(n)$), parities can be learned straightforwardly in polynomial time by checking, for each new example, whether it is a linear combination of the previous ones. We will call this the *trivial algorithm* (see also [Blu96]).

Despite the simplicity of these algorithms, no other methods for learning parities in the mistake-bound model were known prior to this work. In particular, it was unknown whether $\omega(1)$ -parities could be learned in polynomial time with $o(n)$ mistakes. Our main result (stated next) is the first step in this direction.

Theorem 2.1 (Main result) *Let $k, t : \mathbb{N} \rightarrow \mathbb{N}$ be two functions satisfying $k(n) \leq t(n) \leq n$.¹ For every $n \in \mathbb{N}$ (and the corresponding integers $k = k(n)$ and $t = t(n)$) there is a deterministic algorithm that learns $\text{PAR}(k)$ in the mistake-bound model, with mistake bound $k \lceil \frac{n}{t} \rceil + \lceil \log \binom{t}{k} \rceil$ and running time per round $O\left(\binom{t}{k} (kn/t)^2\right)$.*

Let us examine a few interesting values for the parameters in Theorem 2.1. For example, putting $k = \log n / \log \log n$ and $t = \log n$ yields mistake bound of $O(n / \log \log n)$ and running time per round $O(n^{2+o(1)})$. More generally, it is interesting to find out when $\text{PAR}(k)$ can be efficiently learned with $o(n)$ mistakes. It follows from the lower bound techniques described in [Lit88] that for $k = \Omega(n)$ it is impossible to learn $\text{PAR}(k)$ with sublinear mistake bound, even disregarding computational efficiency. (This is because the VC dimension of $\text{PAR}(k)$ is at least k). So we only need to consider the case $k = o(n)$. Recall that the running time of the halving algorithm is at least $\binom{n}{k}$, which is super-polynomial for any super-constant k , and is $2^{\Omega(k \log n)}$ for any positive $k = n^{1-\Omega(1)}$. In the following we show that, with appropriate parameters, our main theorem can be used to outperform the halving algorithm. Specifically,

- for any $k = o(\log n)$, $\text{PAR}(k)$ can be learned with $o(n)$ mistakes in polynomial time;
- for any $k = o(n)$, $\text{PAR}(k)$ can be learned with $o(n)$ mistakes in time $\approx 2^{O(k + \log n)}$.

The two items above are formalized next.

Corollary 2.2 (Case $k = O(\log n)$)

For any $\omega(1) \leq k = O(\log n)$ and $c \in \mathbb{N}$, define $t = t(n) = \lceil \frac{kn^{c/k}}{e} \rceil$ (for large enough n). Then $\text{PAR}(k)$ can be learned deterministically with mistake bound $O(n^{1-c/k})$ and running time per round $O(n^{c+2-2c/k})$. Consequently (see Theorem 1.1), $\text{PAR}(k)$ can be learned deterministically in the PAC model with $O(n^{1-c/k})$ samples and running time $\tilde{O}(n^{3+c-3c/k})$.

In particular, if $k = o(\log n)$ then the mistake bound (sample size) is $o(n)$.

Corollary 2.3 (Case $k = o(n)$) *For any $\omega(1) \leq k = o(n)$ let $t = t(n) \leq n$ be an arbitrary function with growth rate $\omega(k)$. Then $\text{PAR}(k)$ can be learned deterministically with mistake bound $O(kn/t + k \log \frac{t}{k}) \leq o(n)$, and total running time $2^{O(k \log \frac{t}{k} + \log n)}$. Consequently (see Theorem 1.1), $\text{PAR}(k)$ can be learned deterministically in the PAC model with $O(kn/t + k \log \frac{t}{k}) = o(n)$ samples and running time $2^{O(k \log \frac{t}{k} + \log n)}$.*

For example, if $t = k \log k$ then the running time in both cases is $2^{O(k \log \log k + \log n)}$.

¹Throughout this paper, we assume that the functions $k(n), t(n)$ are computable in $O(n^2)$ time.

In addition to the corollaries above, observe that Theorem 2.1 with $t = \frac{n}{\log(n/k)}$ gives the same mistake bound as the halving algorithm with slightly better running time. Similarly, we can obtain the features of the trivial algorithm by setting $k = t = n$.²

2.1 Learning parities in the PAC model

In the PAC model, Klivans and Servedio [KS04] were the first to show non-trivial algorithms for learning parities with sample sets of sublinear size. (They attribute the second item of the following theorem to Dan Spielman, although it seems to have previously appeared in the literature, e.g. [HB01]).

Theorem 2.4 ([KS04])

1. $\text{PAR}(k)$ can be learned in the PAC model with $O(n^{1-1/k} \log n)$ samples in time $O(n^4)$.
2. $\text{PAR}(k)$ can be learned in the PAC model with $O(k \log n)$ samples in time $\tilde{O}(n^{\lceil k/2 \rceil})$.

Since our main theorem holds in the harder mistake-bound model, using a standard conversion techniques (see Theorem 1.1) it also implies results similar to those in Theorem 2.4, even with improved parameters. In particular, from Corollary 2.2 (with $c = 1$) and Corollary 2.3 we get the following.

Theorem 2.5

1. $\text{PAR}(k)$ can be learned in the PAC model with $O(n^{1-1/k})$ samples in time $O(n^{4-3/k} \log n)$.
2. $\text{PAR}(k)$ can be learned in the PAC model with $o(n)$ samples in time $\approx 2^{O(k+\log n)}$.

In the first item, the number of samples required by our algorithm is improved by a factor of $\log n$, and the running time is improved by a factor of $n^{3/k}$. As for Item 2, our algorithm requires more than $O(k \log n)$ samples (although still $o(n)$), but its running time is reduced to $\approx 2^{O(k+\log n)}$, compared to the $2^{\Omega(k \log n)}$ time required by both the halving algorithm and the algorithm from Item 2 of Theorem 2.4. In addition to these features, our algorithms are *deterministic* whereas the algorithms from [KS04] are probabilistic.

2.2 Extending the $\tilde{O}(n^{k/2})$ algorithm to the mistake-bound model

The second item in Theorem 2.4 brings down the running time of the halving algorithm to roughly $O(n^{k/2})$, while still using a sample set of the same size (up to constant factors). It is natural to ask whether such an improvement is attainable in the mistake-bound model too. Our main result does not directly imply such an improvement; however, using similar ideas it is possible to extend Item 2 of Theorem 2.4 to the mistake-bound model as well. Specifically, the following theorem is proved in Section 4.

²Usually k is a parameter of the concept class (and not of the algorithm), but it is clear that the class $\text{PAR}(n)$ contains $\text{PAR}(k)$ for all $k \leq n$.

Theorem 2.6 $\text{PAR}(k)$ can be learned in the mistake-bound model with mistake bound $O(k \log n)$ and maximum running time per round $O(n^{\lceil k/2 \rceil})$.

3 Proof of Theorem 2.1

The algorithm from Theorem 2.1 is based on an idea that was recently used by Alon, Panigrahy and Yekhanin, who gave elegant deterministic algorithms for approximating the Nearest Codeword and Remote Point problems (see [APY08] for details). First we outline the main idea in this algorithm, and then provide its formal description together with the proof.

3.1 Informal description of the algorithm

Recall that, in the halving algorithm, a set \mathcal{H} of candidate parity functions is maintained, and given an example x , the prediction of the learner is $\text{majority}\{h(x) : h \in \mathcal{H}\}$. The problem with this method is that for any $k = \omega(1)$, the initial set $\mathcal{H} = \text{PAR}(k)$ is of super-polynomial size, and we have no efficient algorithm to compute the majority vote.

In order to overcome this problem, we use a special set of affine spaces that enables a compact representation of (a superset of) the candidate parity functions, while at the same time enabling efficient approximation of their majority vote, for any example x . Specifically, our learning algorithm begins by obtaining a set of affine spaces $N_1, N_2, \dots \subseteq \{0, 1\}^n$, at least one of them containing the target parity function f . In every step of the the learning process, these sets of affine spaces are updated according to the response given by the teacher. The way these updates are performed guarantees:

- the running time is polynomial in n and linear in the number of affine spaces N_i ;
- after every mistake, some sets N_i get shrunk, so that the quantity $\sum_i |N_i|$ is at least halved (this is ensured by approximating the majority vote);
- the target function f is never removed from any N_i .

Since $\sum_i |N_i| \geq |\bigcup_i N_i|$, after at most a logarithmic (in $\sum_i |N_i|$) number of mistakes the target function f is the only element left in $\bigcup_i N_i$, and hence f is learned.

3.2 Formal description and proof of Theorem 2.1

Define $S \subseteq 2^{[t]}$ as $S = \{s \subseteq [t] : |s| = k\}$, hence $|S| = \binom{t}{k}$. Let $\pi = C_1, \dots, C_t$ be an arbitrary partition of $\{e_1, e_2, \dots, e_n\}$ (the standard basis of $\{0, 1\}^n$) into t equally (up to ± 1) sized parts. For every $s \in S$ we define the linear subspace $M_s = \text{span}(U_s)$, where U_s is a set of unit vectors defined as

$$U_s \triangleq \bigcup_{i \in s} C_i.$$

That is, M_s consists of all binary vectors whose non-zero entries belong to the parts that are indexed by the elements of s . Notice that for every $s \in S$, M_s is a span of at most $k \lceil \frac{n}{t} \rceil$ vectors, and hence

$$|M_s| \leq 2^{k \lceil n/t \rceil}.$$

Let $\ell = k \lceil \frac{n}{t} \rceil$. For every affine space $N \subseteq \{0, 1\}^\ell$, $x \in \{0, 1\}^\ell$ and $z \in \{0, 1\}$, we define the affine space $N(x, z) \triangleq \{y \in N : \langle y, x \rangle = z \pmod{2}\}$. Given $x \in \{0, 1\}^\ell$, $z \in \{0, 1\}$ and a representation for N as a system $Lin^N \in \{0, 1\}^{\ell \times (\ell+1)}$ of independent linear equations in triangular form, the corresponding representation of $N(x, z)$ (and the cardinality $|N(x, z)|$) can be computed in time $O(\ell^2)$. This is done by adding $x' = x \sqcup z \in \{0, 1\}^{\ell+1}$ to Lin^N and performing only one step of the Gaussian elimination procedure. Notice that this procedure has three possible outcomes:

- (i) x' is inconsistent with Lin^N , and hence $|N(x, z)| = 0$;
- (ii) x' is a linear combination of equations in Lin^N , and hence $|N(x, z)| = |N|$;
- (iii) x' is linearly independent of Lin^N , and hence $|N(x, z)| = |N|/2$.

Proposition 3.1

- 1. Every $f \in \{0, 1\}^n$ with $|f| \leq k$ is contained in $\bigcup_{s \in S} M_s$;
- 2. $|\bigcup_{s \in S} M_s| \leq \binom{t}{k} 2^{k \lceil n/t \rceil}$.
- 3. Let $\{N_s : s \in S\}$ be a family of affine subspaces of $\{0, 1\}^n$. For any $x \in \{0, 1\}^n$ there exists $z \in \{0, 1\}$ for which $\sum_{s \in S} |N_s(x, z)| \geq \frac{1}{2} \sum_{s \in S} |N_s|$.

Proof.

- 1. This follows from the fact that every set of k unit vectors is contained in the union of some $d \leq k$ subsets C_{i_1}, \dots, C_{i_d} in the partition π . Let $s \subseteq [t]$, $|s| = k$ be a set that contains i_1, \dots, i_d . Then $f \in M_s$.
- 2. $|\bigcup_{s \in S} M_s| \leq \sum_{s \in S} |M_s| \leq \binom{t}{k} 2^{k \lceil n/t \rceil}$.
- 3. This is a consequence of the equality $|N_s(x, 0)| + |N_s(x, 1)| = |N_s|$.

■

The learner proceeds as follows:

Initialization:

Obtain a system of equations describing each of the linear spaces M_s as defined above; and then initialize the affine spaces $N_s = M_s$ for all $s \in S$.

On example $x \in \{0, 1\}^n$:

Compute $n_0 = \sum_{s \in S} |N_s(x, 0)|$ and $n_1 = \sum_{s \in S} |N_s(x, 1)|$. Let $l' \in \{0, 1\}$ be a value that satisfies $n_{l'} \geq n_{1-l'}$. Output l' .

On answer $l = \langle f, x \rangle$:

Update $N_s := N_s(x, l)$ for each $s \in S$.

It might be helpful for the reader to think that each M_s runs an independent instance of the trivial algorithm of Section 2. Each instance assumes that all parity bits of f belong to the corresponding M_s , and N_s is the set of candidates (parities consistent with the answers to previous examples) left under this assumption. Some of these candidate sets will vanish as new values of f are learned, but at least one of them will always remain non-empty. The second step of the algorithm can be viewed as taking a weighted majority of all “surviving” instances, where the weight of an instance is proportional to the number of candidates left for it. Thus, whenever a new sample is, when restricted to a set M_s , linearly independent of the prior ones, we “penalize” the s -th instance by halving its weight; while if the new example is inconsistent we remove the s -th instance from consideration.

Proof of Theorem 2.1:

First notice that the invariant $f \in \bigcup_{s \in S} N_s$ holds at any stage of the learning algorithm. Initially it holds by Item 1 of Proposition 3.1, and every time the algorithm shrinks the sets N_s , only elements that are not equal to f are removed.

Since all the subspaces N_s contain vectors of Hamming weight at most $\ell = k \lceil \frac{n}{t} \rceil$, we can treat them as linear subspaces in $\{0, 1\}^\ell$ by truncating all their irrelevant coordinates. In addition, for any N_s , an example $x \in \{0, 1\}^n$ can be truncated to the corresponding ℓ -bit vector by removing all the irrelevant coordinates (with respect to N_s). Making this observation, the bound on the running time (per round) of the algorithm now follows from Item 2 of Proposition 3.1 and the fact that $|S| \leq \binom{t}{k}$.

Finally, we have to show that the number of mistakes that the learner makes is bounded by $k \lceil \frac{n}{t} \rceil + \lceil \log \binom{t}{k} \rceil$. Notice that by the definition of the output value l' and Item 3 of Proposition 3.1, every time the learner makes a mistake the quantity $\sum_{s \in S} |N_s|$ reduces by a factor of at least 2. Since at every step $0 < |\bigcup_{s \in S} N_s| \leq \sum_{s \in S} |N_s|$, and since initially we started with $\sum_{s \in S} |N_s| = \sum_{s \in S} |M_s| \leq \binom{t}{k} 2^{k \lceil n/t \rceil}$ (see Item 2 of Proposition 3.1), after at most $\log \left(\sum_{s \in S} |M_s| \right) \leq k \lceil \frac{n}{t} \rceil + \lceil \log \binom{t}{k} \rceil$ mistakes the size of $\bigcup_{s \in S} N_s$ will decrease to 1, which by the invariant above will imply that $\bigcup_{s \in S} N_s = \{f\}$, and the learner will no longer make any errors. ■

3.3 Optimality of the system of affine spaces

To recap, we have constructed a set \mathcal{A} of $m = \binom{t}{k}$ affine spaces of dimension $d = kn/t \leq \frac{n}{m^{1/k}}$ that together “cover” all vectors of weight $\leq k$, in that every such vector belongs to one of the elements of \mathcal{A} . The mistake bound we get is $\log \sum_{A \in \mathcal{A}} |A| = \log m + d \leq \log m + n/(m^{1/k})$. One may ask whether this value can be improved upon by finding a better system of affine spaces. This possibility can be ruled out:

Proposition 3.2 Let $0 < k \leq n/100$ and suppose \mathcal{A} is a collection of $m \leq \left(\frac{n}{100k}\right)^k$ affine spaces over $\{0, 1\}^n$ such that every $x \in \{0, 1\}^n$ with $|x| \leq k$ belongs to some $A \in \mathcal{A}$. Then

$$\log \left(\sum_{A \in \mathcal{A}} |A| \right) \geq \log m + \frac{n}{3m^{1/k}}.$$

Lemma 3.3 If $V \in \{0, 1\}^n$ is an affine subspace of dimension d , then the number of words in V of weight at most k is upper bounded by

$$\binom{d}{\leq k} \triangleq \binom{d}{0} + \binom{d}{1} + \dots + \binom{d}{k}.$$

Proof. As V is a d -dimensional affine subspace, there is a set $D \subseteq [n]$, $|D| = d$ such that the projection $\pi_D(x)$ of each element $x \in V$ on the coordinates of D uniquely determines x . Consider the set $V_k \triangleq \{x \in V \mid |x| \leq k\}$ and write $\pi_D(V_k) \triangleq \{\pi(x) \mid x \in V_k\}$; we have just seen that $|\pi_D(V_k)| = |V_k|$. But no element of $\pi_D(V_k)$ has weight greater than k by construction, hence $|\pi_D(V_k)| \leq \sum_{i=0}^k \binom{d}{i}$. ■

Proof of Proposition 3.2. Write $\mathcal{A} = \{A_1, \dots, A_m\}$ and $d_i \triangleq \dim A_i$, so $|A_i| = 2^{d_i}$. In order to cover all vectors of weight at most k , simple counting (along with the preceding lemma) tells us that

$$\sum_{i=1}^m \binom{d_i}{\leq k} \geq \binom{n}{\leq k}.$$

Suppose there are r subspaces of dimension larger than $2k$ and let $d_1, \dots, d_r > 2k \geq d_{r+1}, \dots, d_m$. For the small subspaces we can compute $\sum_{d_i \leq 2k} \binom{d_i}{\leq k} \leq m4^k$. This, along with our bound on m and the well-known inequality $(a/b)^b \leq \binom{a}{b} \leq (ae/b)^b$ implies that

$$\sum_{i=1}^r d_i^k \geq \left(\frac{n}{3}\right)^k.$$

The function $f : [e^{k-1}, \infty] \rightarrow \mathbb{R}^+$ defined by $f(x) = (\log x)^k$ is concave if $k > 0$, therefore Jensen's inequality applied to the sequence $\{2^{d_i}\}_{i \in [r]}$ furnishes

$$f\left(\frac{1}{r} \sum_{i=1}^r x_i\right) \geq \frac{1}{r} \sum_{i=1}^r f(x_i) \geq \frac{1}{r} \left(\frac{n}{3}\right)^k,$$

whence

$$\log \left(\sum_{d_i > 2k} 2^{d_i} \right) \geq \log r + \frac{n}{3r^{1/k}} \geq \log m + \frac{n}{3m^{1/k}},$$

where the last step made use of the fact that the function $g(r) = r + n/r^{1/k}$ is increasing in a range containing $[1, m]$. ■

4 Proof of Theorem 2.6

As mentioned previously, the possibility to improve the running time of the halving algorithm to roughly $O(n^{k/2})$ has been noted by several authors. In this section we explain how to extend the ideas to derive a mistake-bounded algorithm.

Theorem 2.6 *PAR(k) can be learned in the mistake-bound model with mistake bound $O(k \log n)$ and maximum running time per round $O(n^{\lceil k/2 \rceil})$.*

Proof. Let $A = \text{PAR}(\lceil k/2 \rceil) \times \text{PAR}(\lceil k/2 \rceil)$. Then $|A| = O(n^{k+1})$. We can associate each element $(p, q) \in A$ with the “parity-pair” $p \oplus q$; each parity $r \in \text{PAR}(k)$ will then correspond to several pairs in A , namely those such that $r(x) = p(x) \oplus q(x)$ for all $x \in \{0, 1\}^n$. Thus, we can view A as a multiset of $(k + 1)$ -parities (as well as a set of parity-pairs). The *answer* of a parity-pair (p, q) on x is defined as $p(x) \oplus q(x)$.

We will show that, given any input x , we can compute the majority vote of the answers of all parity-pairs in A that agree with all previous examples in $O(n^{\lceil k/2 \rceil})$ time, effectively simulating the halving algorithm over the multiset A . This implies that the number of mistakes will be bounded by $\log |A| = O(k \log n)$.

In order to compute this majority, it is enough to know how many parity-pairs in A are consistent with all the examples seen so far and would output 0 for the new example (and how many of them would output 1). Assume we have been given the examples $x = x_1, x_2, \dots, x_{m-1} \in \{0, 1\}^{n \times (m-1)}$ with answers $y = y_1, y_2, \dots, y_{m-1} \in \{0, 1\}^{m-1}$, together with the new example $x_m \in \{0, 1\}^n$, and we are required to output our prediction for $f(x_m)$, where f is the unknown parity function. Let $a \triangleq y_1 y_2 \dots y_{m-1} 0$ be the m -bit vector that contains the answers to all previous $m - 1$ examples and whose last entry is 0 (representing that we are trying to count how many consistent parity-pairs would answer 0 for x_m). Each parity $p \in \text{PAR}(\lceil k/2 \rceil)$ will give an answer for examples x_1, \dots, x_{m-1}, x_m ; let $p(x) \triangleq p(x_1)p(x_2) \dots p(x_m) \in \{0, 1\}^m$ be their concatenation. Consider the multisets $V \triangleq \{p(x) \mid p \in \text{PAR}(\lceil k/2 \rceil)\}$ and $W_a \triangleq \{p(x) \oplus a \mid p \in \text{PAR}(\lceil k/2 \rceil)\}$ (where \oplus denotes bitwise addition mod 2). Sort the multiset $V \cup W_a$ in, say, lexicographical order, keeping track of whether each vector comes from V or from W_a . For each range of (consecutive) equal elements in the sorted sequence $V \cup W_a$ (equal to some vector $c \in \{0, 1\}^m$), count how many of them are from V and how many are from W_a ; call these numbers r and s respectively. If we write $p(x) \in \{0, 1\}^m$ for the answers of parity p on all examples (including the new one), what this means is that there are $\lceil k/2 \rceil$ -parities $p_1, p_2, \dots, p_r, q_1, \dots, q_s$ such that $c = p_1(x) = p_2(x) = \dots = p_r(x) = q_1(x) \oplus a = q_2(x) \oplus a = \dots = q_s(x) \oplus a$, and p_1, \dots, p_r are distinct (as are q_1, \dots, q_s).

Thus, there are exactly rs pairs of parities in $\text{PAR}(\lceil k/2 \rceil)$ such that $p(x) \oplus q(x) = a$ and $p(x) = c$. For each range of equal elements in the sorted sequence $V \cup W_a$, we will find a possible value of c . Summing rs over all these ranges we obtain the number of pairs $(p, q) \in A$ such that $p(x) \oplus q(x) = a$. We can compute this in linear time by making one pass over the sorted sequence. We can similarly compute the number of parity-pairs consistent with previous examples that output 1, and then predict the bit that agrees with the majority of consistent parity-pairs.

For the implementation, note that we can go through all $\text{PAR}(\lceil k/2 \rceil)$ parities and compute their answers on x_m in $O\left(\binom{n}{\lceil k/2 \rceil}\right)$ time (a naive implementation might give an additional factor of n , but this factor can be avoided with some care). Note also that before any example has been given, the sequence $V \cup W_a$ can be regarded as a multiset of empty vectors, and is thus sorted; and given a new example, if we keep the multiset $V \cup W_a$ corresponding to our answer from the previous round, we can update the sequence in $O(|V|)$ time by performing one step of radix-sort, since it is already sorted with respect to the first $m - 1$ bits and we only need to sort it with respect to the newly computed bit (the answer of the parity to x_m), which we can consider the most significant one.

Hence, the total running time per round is $O(|V|) = O\left(\binom{n}{\lceil k/2 \rceil}\right)$. ■

5 Concluding remarks

We developed new deterministic algorithms for learning parities in both the mistake-bound and the PAC models of learning. For the mistake-bound model we showed the first efficient algorithm that learns k parities for non-constant k while making a sublinear number of mistakes.

The mistake bound of our algorithm is still far from the value achieved by the halving algorithm. It remains a major open problem to determine whether parities can be learned attribute-efficiently in polynomial time. The halving algorithm is currently not efficient, but if $P = NP$ it can be converted into one that runs in polynomial time, and has approximately the same mistake bound. (This follows from the result of Stockmeyer [Sto83] that if one is provided with access to an NP oracle, then it is possible to use a randomized polynomial-time algorithm to approximate, to within a constant factor, the number of solutions to an NP predicate). Two possible lines of research remain open: either construct an efficient algorithm with improved mistake bound (ideally approaching the bounds of the halving algorithm), or show that the existence of such an algorithm is unlikely.

References

- [Ang88] Dana Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, 1988.
- [APY08] N. Alon, R. Panigrahy, and S. Yekhanin. Deterministic approximation algorithms for the nearest codeword problem. Technical Report TR08-065, Electronic Colloquium on Computational Complexity, 2008.
- [Blu94] Avrim L. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain. *SIAM J. Comput.*, 23(5):990–1000, 1994.
- [Blu96] Avrim Blum. On-line algorithms in machine learning. In *Proceedings of the Workshop on On-Line Algorithms, Dagstuhl*, pages 306–325. Springer, 1996.
- [HB01] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Appli-*

cation of Cryptology and Information Security, pages 52–66, London, UK, 2001. Springer-Verlag.

- [KS04] Adam R. Klivans and Rocco A. Servedio. Toward attribute efficient learning of decision lists and parities. In *Proceedings of COLT*, pages 234–248. MIT Press, 2004.
- [Lit88] Nick Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. In *Machine Learning*, pages 285–318, 1988.
- [Lit89] Nick Littlestone. From on-line to batch learning. In *COLT '89: Proceedings of the second annual workshop on Computational learning theory*, pages 269–284, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
- [Sto83] Larry Stockmeyer. The complexity of approximate counting. In *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 118–126, New York, NY, USA, 1983. ACM.