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 $O(n^{1-\frac{c}{k}})$, for any constant $c > 0$. This is the first polynomial-time algorithms that learns $\omega(1)$-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 on the results of Klivans and Servedio [KS04] for learning k-parities in the PAC model.

We also show that the $\widetilde{O}(n^{k/2})$ time algorithm from [KS04] that PAC-learns k-parities with optimal sample complexity 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 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 $\mathcal{C}$ is polynomial in $\max\{\text{size}(f) : f \in \mathcal{C}\}$. Similarly, algorithm A for learning $\mathcal{C}$ in Valiant's PAC model is attribute-efficient if the sample size required by A to learn $\mathcal{C}$ is polynomial in $\max\{\text{size}(f) : f \in \mathcal{C}\}$.

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].

Dagstuhl Seminar Proceedings 09421
Algebraic Methods in Computational Complexity
http://drops.dagstuhl.de/opus/volltexte/2010/2417

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 $O(\frac{1}{\epsilon}mt + \frac{t}{\epsilon}\log\frac{1}{\delta})$, 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. Every parity function $f : \{0,1\}^n \to \{0,1\}$ is associated with a vector $\tilde{f} \in \{0,1\}^n$ that for every $x \in \{0,1\}^n$ satisfies $f(x) = \langle \tilde{f}, x \rangle \triangleq \sum_{i=1}^{n} \tilde{f}_i x_i \bmod 2$. 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 $f : \{0,1\}^n \to \{0,1\}$ and its corresponding vector $\tilde{f} \in \{0,1\}^n$. We will also refrain from explicitly mentioning n throughout the paper. The concept class $\mathrm{PAR}(k)$ is defined as the class of all parity functions of Hamming weight at most k. The description length of any function $f \in \mathrm{PAR}(k)$ is $O(k \log n)$, and thus ideally we would like to have $\mathrm{poly}(n)$-time algorithms which learn $\mathrm{PAR}(k)$ with a mistake bound (respectively sample size) of $\mathrm{poly}(k \log n)$. This would correspond to attribute-efficient learning as defined above.

It is well known that, in exponential time, $\mathrm{PAR}(k)$ 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 $\mathcal{H} \subseteq \mathrm{PAR}(k)$ of candidate parity functions, and given an example x, it predicts $majority\{h(x) : h \in \mathcal{H}\}$. Whenever a mistake is made, all (at least $|\mathcal{H}|/2$) "wrong" candidates are removed from $\mathcal{H}$. If initially the set $\mathcal{H}$ was set to be $\mathrm{PAR}(k)$, then after at most $\log|\mathrm{PAR}(k)| \leq k \log n$ mistakes the function f is learned. The running time of the halving algorithm is dominated by computing the predicate $majority\{h(x) : h \in \mathcal{H}\}$. Currently the best known implementation for computing this predicate requires $|\mathrm{PAR}(k)| \geq \binom{n}{k}$ many steps, which is super-polynomial for any $k \in \omega(1)$.

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 can 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 $k, t : \mathbb{N} \to \mathbb{N}$ 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 $\mathrm{PAR}(k)$ in the mistake-bound model, with mistake bound $k\lceil \frac{n}{t} \rceil + \lceil \log \binom{t}{k} \rceil$ and running time per example $O\left(\binom{t}{k}(kn/t)^2\right)$.*

Let us examine a few interesting values for the parameters in Theorem 2.1, and see when $\mathrm{PAR}(k)$ 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 $\mathrm{PAR}(k)$ with sub-linear mistake bound, even disregarding computational efficiency. So we can only 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 $k = o(\log n)$, $\mathrm{PAR}(k)$ can be learned with $o(n)$ mistakes in polynomial time;
- for any $k = o(n)$, $\mathrm{PAR}(k)$ 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 $k \in O(\log n)$)
For any $k \in O(\log n)$ and $c \in \mathbb{N}$ define $t = t(n) = \lceil \frac{kn^{c/k}}{e} \rceil$. Then $\mathrm{PAR}(k)$ 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), $\mathrm{PAR}(k)$ can be learned deterministically in the PAC model with $O(n^{1-c/k})$ samples and running time $O(n^{3+c-(c+2)/k})$.

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

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

In addition to the corollaries above, observe that Theorem 2.1 with $t = \frac{n}{\log n}$ 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]

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

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

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 sub-linear size. (The second item in the following theorem is attributed to Dan Spielman.)

Theorem 2.4 ([KS04])

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

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. PAR(k) *can be learned in the PAC model with* $O(n^{1-1/k})$ *samples in time* $O(n^{4-3/k})$.
2. 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 Spielman's 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 optimal 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.

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* $O(n^{\lceil k/2\rceil})$.

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 $majority\{h(x) : h \in \mathcal{H}\}$. The problem with this method is that for any $k = \omega(1)$, the initial set $\mathcal{H} = \mathrm{PAR}(k)$ 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 $N_1, N_2, \ldots \subseteq \{0,1\}^n$, 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 $\pi = C_1, \ldots, C_t$ be an arbitrary partition of $\{e_1, e_2 \ldots, e_n\}$ (the standard basis for $\{0,1\}^n$) into t equally (up to ± 1) sized parts. For every $s \in S$ we define the linear subspace $M_s = \mathrm{span}(U_s)$, 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 are in the parts that belong to 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

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

Proposition 3.1

1. *Every $f \in \{0,1\}^n$ with $|f| \leq k$ is contained in $\bigcup_{s \in S} M_s$; this follows from the fact that every set of k unit vectors is contained in the union of some d subsets $C_{i_1}, \ldots, C_{i_d}$ in the partition π, where $d \leq k$. Let $s \subseteq [t]$, $|s| = k$ be a set that contains $i_1, \ldots, i_d$. 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. *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 $N(x,z) \triangleq \{y \in N : \langle y, x \rangle = z \mod 2\}$. 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. Namely, XOR-ing x' (one-by-one) with the equations $y \in Lin^N$ that have more leading zeroes than x'. 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 $|N(x,z)| = |N|/2$.*

4. *Let $\{N_s : s \in S\}$ be a family of affine subspaces of $\{0,1\}^n$. The equality $|N_s(x,0)| + |N_s(x,1)| = |N_s|$ holds for any $s \in S$ and $x \in \{0,1\}^n$. This implies that for any $x \in \{0,1\}^n$ there exists $z \in \{0,1\}$ for which $\sum_{s \in S} |N_s(x,z)| \geq \frac{1}{2} \sum_{s \in S} |N_s|$.*

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 $s \in S$.

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 3 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 4 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 $0 < \left|\bigcup_{s \in S} N_s\right| \leq \sum_{s \in S} |N_s|$, 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. ■

4 Proof of Theorem 2.6

In this section we show how Spielman's algorithm from [KS04], which improves the running time of the halving algorithm to roughly $O(n^{k/2})$, extends to the mistake-bound model.

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 $|A| \in O(n^{k+1})$. 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-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 $x = x_1, x_2, \ldots, x_{m-1} \in \{0,1\}^{n \times m-1}$ with answers $y = y_1, y_2, \ldots, y_{m-1} \in \{0,1\}^{m-1}$, together with thew new example $x_m \in \{0,1\}^n$, and we are required to output our prediction for $f(x_m)$, where f is the unkown parity function. Let $a = y_1 y_2 \ldots y_{m-1} 0$ 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 $x_1, \ldots, x_{m-1}, x_m$; let $v_p = p(x_1)p(x_2)\ldots p(x_m) \in \{0,1\}^m$ their concatenation. Consider the multiset $V = \{v_p \mid p \in \text{PAR}(\lceil k/2 \rceil)\}$, and the multiset $W_a = \{v_p + a \mid v_p \in V\}$ (where the '+' 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 $V \cup W_a$ (corresponding 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. This means that there are exactly $r + s$ $\lceil k/2 \rceil$-parities $p_1, p_2, \ldots, p_r, q_1, \ldots, q_s$ such that $c = p_1(x) = p_2(x) = \ldots = p_r(x) = q_1(x) + a = q_2(x) + a = \ldots = q_s(x) + a$, where $p(x) \in \{0,1\}^m$ denotes the answers of parity p on all examples, including the new one.

Thus, there are exactly rs pairs of parities in $\mathrm{PAR}(\lceil k/2 \rceil)$ such that $p(x) + q(x) = a$ 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 ranges of equal elements in the sorted sequence $V \cup W_a$, we obtain the number of pairs $(p, q) \in A$ such that $p(x) + q(x) = a$. 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 $\mathrm{PAR}(\lceil k/2 \rceil)$ parities and compute their answers on x_m in $O\left(\binom{n}{\lceil k/2 \rceil}\right)$ time (a naive implementation would 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 sub-linear number of mistakes.

The mistake bound of our algorithm is still far from the optimal mistake bound that is achieved by the halving algorithm. It remains a major open problem 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. 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 *In Proceedings of the Workshop on On-Line Algorithms, Dagstuhl*, pages 306–325. Springer, 1996.

[KS04] Adam R. Klivans and Rocco A. Servedio. Toward attribute efficient learning of decision lists and parities. In *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.