Machine Learning 26, 25–41 (1997)
© 1997 Kluwer Academic Publishers. Manufactured in The Netherlands.

Exact Learning of Formulas in Parallel

NADER H. BSHOUTY bshouty@cpsc.ucalgary.ca
Department of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4

Editor: Wolfgang Maass

Abstract. We investigate the parallel complexity of learning formulas from membership and equivalence queries. We show that many restricted classes of boolean functions cannot be efficiently learned in parallel with a polynomial number of processors.

Keywords: parallel learning, exact learning, membership and equivalence queries

1. Summary of results

In Angluin et al. (1993) present a polynomial time algorithm for learning (boolean) read-once formulas from membership and equivalence queries, and they pose the question of whether it is possible to obtain significant speed-up by using parallelism. This paper addresses this question as well as the more general question of when learning problems that can be solved sequentially in polynomial time can be solved quickly in parallel.

The model of learning that we consider is "exact" learning with membership and equivalence queries. We obtain both lower and upper bounds on the parallel complexity of several learning problems in this model.

We first show that classes over n variables that cannot be learned with poly$(\log n)$ equivalence queries cannot be learned efficiently in parallel from equivalence queries only. This result implies that all the interesting classes known from the literature are not efficiently learnable in parallel from equivalence queries only.

For parallel learning from membership and equivalence queries we show the following. With respect to read-once formulas, we show that boolean read-once formulas with n variables require at least $\Omega(n/\log n)$ parallel steps to learn using a polynomial number of processors.

We also show that $\Omega(n/\log n)$ parallel steps are required for other learning problems for which sequential polynomial time learning algorithms exist. These include the more restricted learning problems of read-once monotone formulas and read-once DNF formulas. Also, the lower bound holds for monotone DNF formulas and 2-DNF formulas (sequentially learnable in polynomial time by Angluin (1987).

All lower bounds hold even if the PRAM model is "information theoretic", that is, it allows unlimited computational power and the equivalence oracle accepts *any* boolean formula (not necessarily one in the class being learned). This latter property is significant because it implies that any of our lower bounds for a class C of formulas immediately

extends to any class of functions that contains C and to any learning from any class of hypothesis H (in particular, proper learning).

We also extend the Halving algorithm that learns any class C from $\log|C|$ equivalence queries (Littlestone, 1988) to a parallel halving algorithm with p processors that learns the class C from $O(\log|C|/\log p)$ parallel steps of membership and equivalence queries. For most of the above classes this bound is nearly tight.

A related work is by Vitter & Lin (1992) who investigate the parallel complexity of learning in the PAC model (Valiant, 1984).

This paper is organized as follows. Section 2 contains some preliminary definitions. Section 3 contains a lower bound for parallel learning from equivalence queries. Section 4 contains the upper bounds for parallel learning from membership and equivalence queries and Section 5 contains a lower bound technique for classes that are learnable from membership and equivalence queries. In Section 6 we give lower bounds for parallel learning classes of boolean functions and in Section 7 we give a lower bound for parallel learning monotone read-once formulas. Section 8 contains conclusion and open problems.

2. Preliminary definitions

The learning criterion we consider is *exact identification*. There is a formula f called the *target formula* which is a member of a class of formulas C defined over the variable set V. The goal of the learning algorithm is to halt and output a formula h from C that is logically equivalent to f. Note that the target f is an (arbitrary) unknown formula chosen from the target class C that is known to the learner.

In a *membership query*, the learning algorithm supplies an assignment a to the variables in V as input to a *membership oracle* and receives in return the value of $f(a)$.

In an *equivalence query*, the learning algorithm supplies any formula h as input to an *equivalence oracle* and the reply of the oracle is either "yes", signifying that h is equivalent to f, or a *counterexample* which is an assignment b such that $h(b) \neq f(b)$.

In the learning procedures given in this paper we assume that the algorithm has access to membership and equivalence oracles for a target formula f over a variable set $\{x_1, \ldots, x_n\}$. The functions in C are represented in some fixed representation $\mathcal{R} \subseteq \{0,1\}^*$. For any $f \in C$ let $s(f)$ be the minimal size of string in $\mathcal{R}$ that represents f. We say that a class of boolean functions C over n variables is *efficiently learnable* in parallel if there exists a parallel learning algorithm that learns any $f \in C$ with $\text{poly}(n, s(f))$ processors and $\text{poly}(\log n, \log s(f))$ time. This definition is similar to the definition of efficient parallel algorithms for solving problems in that the number of processors is polynomial in the input size and the time is poly(log) in the input size. In this paper all logarithms are base 2.

3. Lower bound for parallel learning from equivalence queries

In this section we give a lower bound for parallel learning from equivalence queries. We show that if the class of concepts C cannot be learned sequentially in $\text{poly}(\log n)$ queries then it is not efficiently learnable in parallel even with unlimited computational power. This shows that all the interesting classes considered in the literature that are efficiently

sequentially learnable from equivalence queries only and all the classes C with $\log|C| \geq \omega(\text{poly}(\log n))$ are not efficiently learnable in parallel.

Let C be a class of boolean functions and let e_C be the minimal number of equivalence queries needed to learn the class C from equivalence queries only. Let $E_C(p)$ be the minimal number of parallel equivalence queries steps needed to learn C with p processor. Our main theorem is stated next.

Theorem 1. *For any class of boolean functions C we have*

$$E_C(p) \geq \frac{e_C}{\lceil \log p \rceil}.$$

In particular, we have

$$E_C(poly(n)) \geq \Omega\left(\frac{e_C}{\log n}\right)$$

and if $e_C = \omega(poly(\log n))$ then there is no efficient parallel algorithm that learns C from equivalence queries only.

Proof: Let $\mathcal{A}$ be an algorithm that learns the class C in parallel in $E_C(p)$ parallel steps. Each parallel step contains at most (w.l.o.g exactly) p equivalence queries. We change the algorithm $\mathcal{A}$ to a sequential algorithm $\mathcal{B}$ as follows. Algorithm $\mathcal{B}$ runs algorithm $\mathcal{A}$ until it asks p equivalence queries in one parallel step. Let $H = \{h_1, \ldots, h_p\}$ be the hypotheses of the equivalence queries. The sequential algorithm $\mathcal{B}$ will run the following procedure. In this procedure the function $\text{Maj}(L)$ is the majority function of all functions in L. That is,

$$\text{Maj}(L)(a) = \begin{cases} 0 & \text{more that } |L|/2 \text{ functions } f \in L \text{ satisfy } f(a) = 0, \\ 1 & \text{otherwise.} \end{cases}$$

Seq(H).

1. $L \leftarrow H$, $B \leftarrow \emptyset$.
2. Ask an equivalence queries with $h = \text{Maj}(L)$ and get a counterexample b. If the answer is YES then output h.
3. Eliminate from L all h_i that satisfy $h(b) \neq h_i(b)$.
4. $B \leftarrow B \cup \{b\}$
5. If $L = \emptyset$ then return B else goto 2.

We claim that the procedure **Seq**(H) asks $\lceil \log p \rceil$ equivalence queries and returns counterexamples for all $h_i \in H$. To show this notice that each time we ask an equivalence query the counterexample b will be a counterexample for at least half the hypotheses in L. This is because $\text{Maj}(L)(b) \neq f(b)$ and therefore at least half of the functions in L do not agree with f on b. Therefore the number of equivalence queries asked by the algorithm $\mathcal{B}$ is at most $t = E_C(p)\lceil \log p \rceil$. Since $t \geq e_C$ the result of the theorem follows. □

4. Upper bounds for parallel learning from membership and equivalence queries

In this section we generalize the sequential Halving algorithm for learning any class C from $\log|C|$ equivalence queries to a parallel Halving algorithm with p processors that learns any class C with $\log|C|/\log p$ parallel membership and equivalence queries. Here and throughout the paper when we say that the number of queries is t we always mean that it is $t + o(t)$. This, in particular, will be used to avoid (extra) floor and ceiling operations.

The model of computation (of the learner) will be any PRAM model with unlimited computational power. We call this model UPRAM. All the computations computed by the UPRAM are free and the parallel learning complexity will be the number of parallel membership and equivalence query steps. A UPRAM with p processors can ask a total number of p membership and equivalence queries in one step. The equivalence queries can be asked with any boolean function g. A UPRAM with one processor will be called a UPRAM. We define $\mathrm{ME}_C(p)$ to be the number of parallel steps needed to learn any formula from the class C in a UPRAM with p processors using membership and equivalence queries.

In the sequential Halving algorithm (Littlestone, 1988), the learning algorithm, at some stage, knows that the target $f \in C$ is consistent with some labeled assignments A. The learning algorithm then proceeds as follows. It defines the class $C(A)$ of all functions in C that are consistent with the target on the assignments A. If $C(A)$ contains one function h then h must be the target. In that case the algorithm outputs h and halts. Otherwise the algorithm asks an equivalence query with the hypothesis $\mathrm{Maj}(C(A))$, the majority of all the functions in $C(A)$. A counterexample a to this hypothesis will be added to A. Now it is easy to see that

$$|C(A \cup \{(a,\ f(a))\})| \ \le\ \frac{|C(A)|}{2}.$$

Therefore the worst case number of equivalence queries asked by the Halving learning algorithm is at most $\log|C|$.

Since each counterexample provides $n+1$ bit information the information theoretic lower bound for learning C is $\log|C|/(n+1)$ equivalence queries.

Before we present the parallel Halving algorithm we give the following definitions. For a class of functions $\hat{C}$, an integer r and an assignment a we say that a is an *r-assignment* with respect to $\hat{C}$ if more than $|\hat{C}| - |\hat{C}|/r$ of the functions in $\hat{C}$ have value 0 for a or more than $|\hat{C}| - |\hat{C}|/r$ of the functions in $\hat{C}$ have value 1 for a. The class $\hat{C}$ is called an r-class if every assignment is an r-assignment with respect to the class $\hat{C}$. An *r-strategy tree* for the class $\hat{C}$ is a tree whose internal nodes are labeled with pairs $(\tilde{C}, a)$ where $\tilde{C} \subset \hat{C}$, a is an assignment and the leaves of the tree are labeled with $(\tilde{C}, \mathrm{NIL})$ where $\tilde{C} \subset \hat{C}$. The node for class $\tilde{C} \subset \hat{C}$ is defined as follows: If $\tilde{C}$ is an r-class or $|\tilde{C}| \le |\hat{C}|/r$ then the node is a leaf labeled with $(\tilde{C}, \mathrm{NIL})$. Otherwise, let a be a witness that $\tilde{C}$ is not an r-class. Then the node is labeled $(\tilde{C}, a)$, its left child is an r-strategy tree for $\tilde{C}_0 = \{f \in \tilde{C} \mid f(a) = 0\}$ and its right child is an r-strategy tree for $\tilde{C}_1 = \{f \in \tilde{C} \mid f(a) = 1\}$. Notice that since $\tilde{C}_0 \cup \tilde{C}_1 = \tilde{C}$ and $\tilde{C}_0 \cap \tilde{C}_1 = \emptyset$ the set of classes in the leaves of the tree is a partition of $\tilde{C}$.

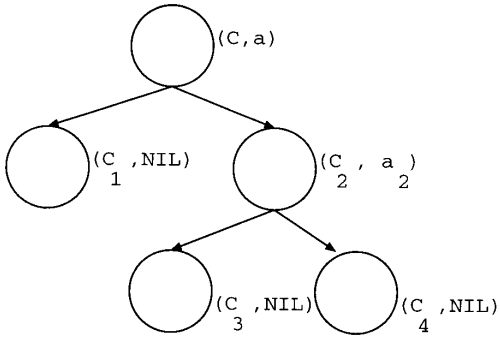

Figure 1. An example of a 3-strategy tree. The sets C_1, C_3 and C_4 are disjoint and $C = C_1 \cup C_3 \cup C_4$.

We first prove the following.

Claim 1. *The number of internal nodes in any r-strategy tree is at most $2r$.*

Proof: Suppose V are the nodes of the tree and let C_v, for all $v \in V$, be the class corresponding to node v. Let v be a leaf in the tree and let u_v be the parent node of v. Since u_v is not a leaf we must have $|C_{u_v}| > |C(A)|/r$ (otherwise u_v would be a leaf). Since the set of all C_{u_v} for all leaves v is a partition for $C(A)$, the number of u_v is at most r and therefore the number of leaves in the tree is at most $2r$. □

In the parallel Halving algorithm the learning algorithm, at some stage, knows that the target $f \in C$ is consistent with some labeled assignments A. The learning algorithm then proceeds as follows. Let $q = p/2$ where p is the number of processors. The algorithm will build a q-strategy tree for $C(A)$. The algorithm can do this because it has unlimited computational power. The algorithm then asks one parallel step of membership queries for all the assignments in the internal nodes of the tree. When the algorithm looks at the answers of the membership queries, they will lead it to some leaf v in the tree. If this leaf is a q-class leaf we ask one equivalence query with the hypothesis $\mathrm{Maj}(C_v)$ where C_v is the class in the leaf v. Since the class is q-class the counterexample guarantees a reduction in the size of the class $C(A)$ by a factor of q. If this leaf satisfies $|C_v| \leq |C(A)|/q$ then we do not need to ask the equivalence query. Therefore using at most two parallel steps we reduce the size of $C(A)$ by a factor of q.

Now, the number of membership queries asked in this step was the number of internal nodes in the strategy tree and by Claim 1 this is at most $2q = p$.

Since every two parallel steps reduce the size of the class by a factor of q, the number of parallel steps is

$$t = \frac{2 \log |C|}{\log q}$$

with $p = 2q$ processors. This implies the following. Note that the bound in the following theorem is $t + o(t)$.

Theorem 2. *The parallel Halving algorithm learns any class C with p processors and*

$$ME_C(p) \leq \frac{2\log|C|}{\log p}$$

parallel steps.

The bound in Theorem 2 can be reduced by a factor of 2 if we ask membership queries for all the internal nodes and equivalence queries with the majority of all of the classes in the leaves in one step (at the expence of more equivalence queries).

Notice also that the parallel algorithm in Theorem 2 can be regarded as $\log|C|/\log p$ pairs of parallel steps where in each pair the first component contains at most p membership queries and the second component contains at most one equivalence query. If we change this algorithm to a sequential algorithm we get an algorithm that uses

$$m = p\frac{\log|C|}{\log p}$$

membership queries and $\frac{\log|C|}{\log p}$ equivalence queries. Therefore, if $p \geq 2$ (and therefore $m \geq \log|C|$)

$$p = \frac{m}{\log|C|}\log p \geq \frac{m}{\log|C|}$$

and the number of equivalence queries in the algorithm is

$$\frac{\log|C|}{\log p} \leq \frac{\log|C|}{\log m - \log\log|C|}.$$

Let C be a class of boolean formulas. Let $e_C(m)$ be the number of equivalence queries needed to learn the functions in C using a URAM when only m membership queries are allowed to be asked by the learner. Obviously, $e_C(m') \geq e_C(m)$ for $m' \leq m$ and for classes $C \subseteq 2^X$ where $X = \{0,1\}^n$ we have $e_C(2^n) = 0$. The above gives the following upper bound for e_C. This upper bound is also proven in (Bshouty et al., 1993).

Lemma 1. *For any class C of boolean functions, if $m \geq \log|C|$ then we have*

$$e_C(m) \leq \frac{\log|C|}{\log m - \log\log|C|}.$$

5. Lower bounds for parallel learning from membership and equivalence queries

In this section we develop a new technique for proving lower bounds for parallel learning from membership and equivalence queries. The lower bounds are for the worst case complexity, i.e., a learner with p processors tries to learn a formula from some class C.

An adversary chooses the target function $f \in C$ and answers the queries of the learner. The learner can ask the adversary queries and the adversary provides the learner with true answers. The adversary can change the formula during the learning process but this formula should be consistent with all the answers of the queries already asked by the learner. The adversary will change the formula so that it will be hard for the learner to learn it.

We show that k independent equivalence queries can be simulated by $k-1$ membership queries and one equivalence query. We can therefore find lower bounds on the number of equivalence queries needed to learn the above classes in parallel.

Our main theorem is:

Theorem 3. *Let C be a class of boolean formulas. We have*

$$ME_C\left(\left\lfloor \frac{m}{e_C(m)} \right\rfloor\right) \geq e_C(m).$$

Proof: Let $\mathcal{A}$ be an UPRAM algorithm that learns C using $p = \lfloor m/e_C(m) \rfloor$ processors and $e < e_C(m)$ steps. We will show how to change this algorithm to a sequential algorithm in the URAM model that uses $m' \leq m$ membership queries and e equivalence queries. Since $e < e_C(m) \leq e_C(m')$, we get a contradiction and the result follows.

Suppose that the first step in $\mathcal{A}$ makes r equivalence queries with distinct boolean functions $g_1, \ldots, g_r$ and s membership queries with $r + s \leq p$. In the URAM model we ask the same membership queries sequentially. For the equivalence query we define $S \leftarrow \{g_1, \ldots, g_r\}$. We now show how to change the r equivalence queries to $r-1$ membership queries and one equivalence query. At stage $j < r$ the set S will contain $r - j + 1 \geq 2$ boolean functions. We choose any two g_{i_1} and g_{i_2} functions in S and find an assignment a such that $g_{i_1}(a) \neq g_{i_2}(a)$ (this can be done for free in the URAM model). We then use the membership query to find $f(a)$. Obviously, there must be a $j \in \{1, 2\}$ such that $f(a) \neq g_{i_j}(a)$ and we then provide the algorithm $\mathcal{A}$ with the counterexample a for g_{i_j} and define $S \leftarrow S \backslash \{g_{i_j}\}$. This process will stop after $r-1$ membership queries, i.e., when the set S contains one element. For this element we use the equivalence oracle. We repeat this procedure for each step.

This sequential algorithm uses at most $m' = \lfloor m/e_C(m) \rfloor e \leq m$ membership queries and $e < e_C(m)$ equivalence queries. □

The results of Lemma 1 with Theorems 2 and 3 provide the following.

Corollary 2. *Let C be a class of boolean formulas. For $|C|^{1/2} \geq p \geq \log^5 |C|$ we have*

$$ME_C(p) \geq e_C\left(\frac{2p \log |C|}{\log p + \log\log |C|}\right)$$

and for $p < \log^5 |C|$ we have

$$ME_C(p) \geq e_C\left(\frac{\log^6 |C|}{3 \log\log |C|}\right).$$

Proof: Let $|C|^{1/2} \geq p \geq \log^5 |C|$ and

$$m = \frac{2p \log |C|}{\log p + \log\log |C|}.$$

Since $p \geq \log^5 |C|$, we have,

$$\begin{aligned}
\frac{\log\log |C|}{\log m} &\leq \frac{\log\log |C|}{\log(p \log |C|) - \log\log(p \log |C|)} \\
&= \frac{1}{\frac{\log(p \log |C|)}{\log\log |C|} - \frac{\log\log(p \log |C|)}{\log\log |C|}} \\
&\leq \frac{1}{5-1} = \frac{1}{4}.
\end{aligned}$$

By Lemma 1 we have,

$$\begin{aligned}
\frac{m}{e_C(m)} &\geq \frac{m(\log m - \log\log |C|)}{\log |C|} \\
&\geq \frac{m \log m}{\log |C|}\left(1 - \frac{\log\log |C|}{\log m}\right) \\
&\geq 2p(1 - o(1)) \times \frac{3}{4} \\
&\geq p.
\end{aligned}$$

Therefore, by Theorem 3, we have,

$$ME_C(p) \geq ME_C\left(\frac{m}{e_C(m)}\right) \geq e_C(m).$$

For $p < \log^5 |C|$ we have

$$ME_C(p) \geq ME_C(\log^5 |C|) \geq e_C\left(\frac{\log^6 |C|}{3 \log\log |C|}\right).$$ □

6. Parallel complexity of boolean functions

In this section we present lower and upper bounds for learning classes of boolean functions that are learnable from membership and equivalence queries. We show that even very restricted classes are not learnable in polylog time with a polynomial number of processors using a UPRAM and an unrestricted hypothesis space.

6.1. *Read-once DNF formulas*

In this section we study the complexity of parallel learning read-once DNF formulas. A read-once DNF formula is a disjunctive normal form formula where each variable appears at most one. We prove the following.

Result 1. *Let C be the class of read-once DNF over n variables. Then*

$$O\left(\frac{n\log n}{\log p}\right) \geq ME_C(p) \geq \Omega\left(\frac{n}{\log n + \log p}\right).$$

In particular, with a polynomial number of processors there is no efficient parallel algorithm for learning read-once DNF.

The latter negative result implies negative results for parallel learning read-once formulas and polynomial size DNF.

The upper bound of the result follows from Theorem 2 and the fact that the number of read-once DNF formulas is at most $n^{O(n)}$. We now prove the lower bound.

In this paper we use $x^c = x$ for $c = 1$ and $x^c = \bar{x}$ for $c = 0$. We will also use $v^{(i)}$ to denote the ith assignment and v_i to denote the ith bit of the assignment v. For instance, $v_j^{(i)}$ is the jth bit in the ith assignment.

We first prove the following.

Lemma 3. *Let C be the class of read-once DNF formulas. If $m \geq n$ then*

$$e_C(m) \geq \left\lfloor \frac{n}{2 + \lceil \log m \rceil} \right\rfloor.$$

Proof: In this proof we will define a read-once DNF that contains $O(n/\log m)$ terms each is of size $k > \log m$. Then we will show that as long as the number of membership queries is less than m the adversary can provide answers to the learner so that very little information can be found about the terms making an equivalence query is necessary. Then we show that the adversary can answer any equivalence query so that at most one term is revealed. This gives the $O(n/\log m)$ lower bound for the number of equivalence queries.

Let $k = 2 + \lceil \log m \rceil$ and $l = \lfloor n/k \rfloor$. We will prove the lower bound for learning the class

$$C' = \left\{ \left(x_1^{c_1} \wedge \cdots \wedge x_k^{c_k}\right) \vee \cdots \vee \left(x_{kl-k+1}^{c_{kl-k+1}} \wedge \cdots \wedge x_{kl}^{c_{kl}}\right) \,\middle|\, (c_1, \ldots, c_{kl}) \in \{0,1\}^{kl} \right\}.$$

The learner will try to learn a formula hidden by the adversary. The adversary will define l sets $S_1 = S_2 = \cdots = S_l = \{0,1\}^k$. The set S_i will be the possible values of $(c_{ki-k+1}, \ldots, c_{ki})$ in the ith term of the formula. The adversary will answer 0 for all the membership queries that the learner asks before the first equivalence query is asked. For

each membership query with assignment $a = (a_1, \ldots, a_n)$ the answer 0 will eliminate the elements

$$\bigcup_{i=1}^{l} \Big\{ \big(x_1^{c_1} \wedge \cdots \wedge x_k^{c_k}\big) \vee \cdots \vee \big(x_{ki-k+1}^{a_{ki-k+1}} \wedge \cdots \wedge x_{ki}^{a_{ki}}\big) \vee \cdots \vee \big(x_{kl-k+1}^{c_{kl-k+1}} \wedge \cdots \wedge x_{kl}^{c_{kl}}\big) \;\Big|\; (c_1, \ldots, c_{kl}) \in \{0,1\}^{kl} \Big\}$$

Therefore, the adversary will set

$$S_i \leftarrow S_i \backslash \{(a_{ki-k+1}, \ldots, a_{ki})\}, \quad i = 1, \ldots, l,$$

for each such assignment a. Since the learner can ask only m membership queries and each membership query eliminates at most one element from each set S_i and since the initial size of S_i is $2^k \geq 4m$, no one of the terms will be found. This implies that the learner should ask at least one equivalence query.

Let g be the boolean function that is used for the first equivalence query. If $g \not\equiv 0$ then the adversary finds an assignment $b = (b_1, \ldots, b_n)$ such that $g(b) = 1$ and then provides b as a counterexample. In this case the equivalence query eliminates only one element in each set S_i, i.e., the adversary will set $S_i \leftarrow S_i \backslash \{(b_{ki-k+1}, \ldots, b_{ki})\}$ for $i = 1, \ldots, l$. If $g \equiv 0$ then the adversary provides any counterexample $(b_1, \ldots, b_n)$ where $(b_1, \ldots, b_k) \in S_1$ and sets

$$S_1 \leftarrow \{(b_1, \ldots, b_k)\}.$$

This is the case where the adversary has no other choice but to provide information about one of the terms.

After w equivalence queries the adversary assumes that the learner knows

$$S_1, S_2, \ldots, S_w$$

(and therefore the first w terms). Suppose $S_i = \{(c_{ki-k+1}, \ldots, c_{ki})\}$ for $i = 1, \ldots, w$, i.e., $x_{ki-k+1}^{c_{ki-k+1}} \wedge \cdots \wedge x_{ki}^{c_{ki}}$ is the ith term of the target formula for $i = 1, \ldots, w$. The adversary will act as follows. If the learner asks a membership query with $a = (a_1, \ldots, a_n)$ the adversary returns $b = (a_1^{c_1} \wedge \cdots \wedge a_k^{c_k}) \vee \cdots \vee (a_{kw-k+1}^{c_{kw-k+1}} \wedge \cdots \wedge a_{kw}^{c_{kw}})$. If $b = 1$ then no information is gained by the learner. If $b = 0$ then this assignment will eliminate at most one element in each set $S_i, i > w$. If the learner asks an equivalence query with a function g then we have the following cases.

Case I. If

$$g(\xi_1, \ldots, \xi_{sk-k}, c_{sk-k+1}, \ldots, c_{sk}, \xi_{sk+1}, \ldots, \xi_n) = 0$$

for some ξ_i and $s \leq w$ then the adversary returns the assignment

$$(\xi_1, \ldots, \xi_{sk-k}, c_{sk-k+1}, \ldots, c_{sk}, \xi_{sk+1}, \ldots, \xi_n).$$

In this case the learner can learn nothing about the other terms in the formula.

Case II. If $g(\xi_1, \ldots, \xi_n) = 1$ and $(\xi_1^{c_1} \wedge \cdots \wedge \xi_k^{c_k}) \vee \cdots \vee (\xi_{kw-k+1}^{c_{kw-k+1}} \wedge \cdots \wedge \xi_{kw}^{c_{kw}}) = 0$ for some ξ_i then the adversary will return $(\xi_1, \ldots, \xi_n)$ as a counterexample. In this case the counterexample eliminates at most one element from each S_i, $i > w$. The sets S_i will be updated by the adversary.

If Cases I and II do not apply then the following case does.

Case III. If $g(x_1, \ldots, x_n) \equiv (x_1^{c_1} \wedge \cdots \wedge x_k^{c_k}) \vee \cdots \vee (x_{kw-k+1}^{c_{kw-k+1}} \wedge \cdots \wedge x_{kw}^{c_{kw}})$ then the adversary provides the counterexample $(\bar{c}_1, \ldots, \bar{c}_{kw}, b_{kw+1}, \ldots, b_n)$ where $(b_{kw+1}, \ldots, b_{kw+k}) \in S_{w+1}$ and sets

$$S_{w+1} \leftarrow \{(b_{kw+1}, \ldots, b_{kw+k})\}.$$

In this case the adversary assumes that the learner knows now the first $w+1$ terms and continues as before. This implies that l is a lower bound for the number of equivalence queries used in the program. □

Proof of result 1: For C', defined in the proof of Lemma 3, we have $|C'| \leq 2^n$. By Corollary 2 and Lemma 3 we have

$$\begin{aligned} \mathrm{ME}_C(p) &\geq e_C\left(\left\lceil \frac{2p \log |C'|}{\log p + \log\log |C'|} \right\rceil\right) \\ &\geq \Omega(2p \log |C'|) \\ &\geq \Omega\left(\frac{n}{\log n + \log p}\right). \end{aligned}$$ □

6.2. *k-DNF formulas*

In this subsection we show that the class of 2-DNF (disjunctive normal forms where each term contains at most two literals) is not efficiently learnable in parallel using a polynomial number of processors.

Result 2. *Let C be the class of 2-DNF formulas over n variables. Then*

$$O\left(\frac{n^2}{\log p}\right) \geq ME_C(p) \geq \Omega\left(\frac{n}{\log n + \log p}\right).$$

In particular,

$$ME_C(poly(n)) = \Omega\left(\frac{n}{\log n}\right).$$

Proof: We prove the result for 2-CNF. The result then follows from duality. Consider the following boolean function

$$\begin{aligned} &(x_1 \equiv x_2 \equiv \cdots \equiv x_k) \\ &\quad = (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3) \wedge \cdots \wedge (\bar{x}_{k-1} \vee x_k) \wedge (x_{k-1} \vee \bar{x}_k). \end{aligned}$$

This function satisfies $(x_1 \equiv x_2 \equiv \cdots \equiv x_k) = 1$ if and only if $x_1 = x_2 = \cdots = x_k$. For a vector $\xi \in \{0,1\}^k$ we define

$$f_\xi(x_1, \ldots, x_k) = (x_{i_1} \equiv \cdots \equiv x_{i_r} \equiv 1) \wedge (x_{j_1} \equiv \cdots \equiv x_{j_s})$$

where $\{i_1, \ldots, i_r\} = \{i \mid \xi_i = 1\}$ and $\{j_1, \ldots, j_s\} = \{j \mid \xi_j = 0\}$. This formula is a 2-CNF and satisfies $f_\xi(x_1, \ldots, x_k) = 1$ if and only if $x = \xi$ or $x = (1, 1, \ldots, 1)$. The adversary will define the class

$$C = \{ f_{\xi^{(1)}}(x_1, \ldots, x_k) \wedge f_{\xi^{(2)}}(x_{k+1}, \ldots, x_{2k}) \wedge \cdots \\ \wedge f_{\xi^{(l)}}(x_{lk-k+1}, \ldots, x_{lk}) \mid \xi^{(1)}, \ldots, \xi^{(l)} \in \{0,1\}^k \}$$

where $k = \lceil \log 2n \rceil$ and $l = \lfloor n/k \rfloor$. After w equivalence queries the adversary assumes that $\xi^{(1)}, \ldots, \xi^{(w)}$ are known. The adversary also defines $S_{w+1}, \ldots, S_l$ of possible $\xi^{(w+1)}, \ldots, \xi^{(l)}$, respectively. As in the proof of Lemma 3 it can be shown that each membership query eliminates at most one element from exactly one S_i, $i > w$, and using equivalence queries the learner can learn at most one vector from $\xi^{(w+1)}, \ldots, \xi^{(l)}$. □

6.3. Other classes

In this subsection we give three bounds for parallel learning classes of boolean functions with a polynomial number of processors. The classes are k-term DNF formulas for $k = O(\log n)$ (DNF with at most k terms), monotone DNF formulas (DNF with no negated variables) and DNF formulas.

Result 3. *Let KDNF, MDNF and DNF be the classes of k-term DNF formulas for $k = O(\log n)$, the class of polynomial size monotone DNF formulas and the class of polynomial size DNF formulas, respectively. Then*

$$ME_{KDNF}(poly(n)) = k,$$
$$ME_{MDNF}(poly(n)) \geq poly(n),$$
$$ME_{DNF}(poly(n)) = poly(n).$$

Proof: All the results follow from (Bshouty et al., 1993). □

7. Parallel complexity of monotone read-once formulas

In this section we investigate the parallel complexity of monotone read-once formulas, MROF. It is known from (Angluin et al., 1993) that the class of MROF is learnable from membership queries only, i.e., $e_{\text{MROF}}(\text{poly}(n)) = 0$. Therefore the technique used so far that relies heavily on the lower bounds for e_C cannot be applied for this class. The negative results for MROF are based on the following.

Theorem 4. *Any UPRAM algorithm with p processors that learns a class C in t steps can be changed to a UPRAM algorithm with $p(p-1)/2$ processors that learns C in $2t$ steps where in each odd step (steps $1, 3, 5, \ldots$) the algorithm asks $p(p-1)/2$ membership queries and in each even step the algorithm asks one equivalence query.*

Proof: The proof is similar to the proof of Theorem 3. We have $\{g_1, \ldots, g_s\}$ boolean functions used for the equivalence query in some step and $p - s$ membership queries. For every $1 \le i < j \le s \le p$ we find an assignment $a^{(ij)}$ such that $g_i(a^{(ij)}) \ne g_j(a^{(ij)})$ and then use membership queries to find $f(a^{(ij)})$. It can be easily shown that this provides counterexamples for all g_i except at most one. For this one we use the equivalence query in the following step. The number of membership queries used for this step is

$$\frac{s(s-1)}{2} + (p-s) \le \frac{p(p-1)}{2}. \qquad \square$$

The proof of this result is based on Theorem 4.

Result 4. *Let C be the class of monotone read-once formulas over n variables. Then*

$$O\left(\frac{n \log n}{\log p}\right) \ge ME_C(p) \ge \Omega\left(\frac{n}{\log p}\right).$$

In particular,

$$ME_C(poly(n)) \ge \Omega\left(\frac{n}{\log n}\right).$$

Proof: The upper bound follows from Theorem 2 and the fact that the number of read-once formulas is bounded above by $n^{O(n)}$.

For the lower bound we use Theorem 4. The adversary defines the vectors $v^{(1)} = (x_1, \ldots, x_k), \ldots, v^{(l)} = (x_{lk-k+1}, \ldots, x_{lk})$ where $k = 2 + 2\lceil \log p \rceil$ and $l = \lfloor n/k \rfloor$. Notice that $lk \le n$ so the variables are all from $\{x_1, \ldots, x_n\}$. Define

$$f_{(\xi_1,\ldots,\xi_k)}(x_1, \ldots, x_k, y) = \left(\bigvee_{j \in \{i \le k \mid \xi_i = 0\}} x_j \right) \vee \left(\bigwedge_{j \in \{i \le k \mid \xi_i = 1\}} x_j \wedge y \right).$$

This function satisfies

$$f_{(\xi_1,\ldots,\xi_k)}(x_1, \ldots, x_k, y) = \begin{cases} 1 & \text{If } x_j = 1 \text{ for some } j \in \{i \le k \mid \xi_i = 0\}. \\ 0 & \text{If } x_j = 0 \text{ for all } j \in \{i \le k \mid \xi_i = 0\} \\ & \quad \text{and } x_j = 0 \text{ for some } j \in \{i \le k \mid \xi_i = 1\}. \\ y & \text{If } (x_1, \ldots, x_k) = (\xi_1, \ldots, \xi_k). \end{cases}$$

The adversary defines the set of monotone read-once formulas as follows.

$$C = \{ f_{\xi^{(1)}}(v^{(1)}, f_{\xi^{(2)}}(v^{(2)}, \ldots, f_{\xi^{(l)}}(v^{(l)}, 1))) \cdots) \mid \xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(l)} \in \{0,1\}^k \}.$$

Notice that

$$f_{\xi^{(1)}}(v^{(1)}, f_{\xi^{(2)}}(v^{(2)}, \ldots, f_{\xi^{(l)}}(v^{(l)}, 1)))\cdots)$$
$$= f_{\xi^{(r)}}(v^{(r)}, f_{\xi^{(r+1)}}(v^{(r+1)}, \ldots, f_{\xi^{(l)}}(v^{(l)}, 1)))\cdots) \qquad (\star)$$

if

$$\xi^{(1)} = v^{(1)} \cdots \xi^{(r-1)} = v^{(r-1)}.$$

Obviously C is a subset of monotone read-once formulas and learning a function in C is equivalent to learning $\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(l)}$.

We assume that the learner asks $p(p-1)/2$ membership queries in the odd parallel steps and one equivalence query in the even parallel steps.

After $2w$ steps, the adversary assumes that the learner knows $\xi^{(1)}, \ldots, \xi^{(w)}$, two bits in $\xi^{(w+1)}$ but nothing about $\xi^{(w+2)}, \ldots, \xi^{(l)}$. We will show that $p(p-1)/2$ membership queries might help the learner to know $\xi^{(w+1)}$ but nothing about $\xi^{(w+2)}, \ldots, \xi^{(l)}$. Therefore the learner must ask an equivalence query. Then we will show that asking an equivalence query will help the learner to know at most two bits in $\xi^{(w+2)}$.

In step $2w+1$ the learner asks $m = p(p-1)/2$ membership queries. Let $Q = \{a^{(1)}, \ldots, a^{(m)}\}$ be these queried vectors. The adversary will look at the first kw entries of each vector $a^{(i)}$ and define

$$G = \left\{a^{(i)} \mid \left(a_1^{(i)}, \ldots, a_{kw}^{(i)}\right) \neq (\xi^{(1)}, \ldots, \xi^{(w)})\right\}$$

and

$$E = Q \backslash G.$$

Suppose that $\xi_{j_1}^{(w+1)}$ and $\xi_{j_2}^{(w+1)}$ are the bits that are known for the learner in $\xi^{(w+1)}$. The assignments in G are the assignments that have answers independent of $\xi^{(w+1)}, \ldots, \xi^{(l)}$ (see property $(\star)$). Therefore the assignments in G will give no information to the learner. The adversary will then look at the bits $(a_{kw+1}^{(i)}, \ldots, a_{kw+k}^{(i)})$ of each $a^{(i)} \in E$. Since at most two bits in $\xi^{(w+1)}$ are known to the learner and $2^{k-2} \geq p^2 > p(p-1)/2$ there exists $\eta \in \{0,1\}^k$ such that $\eta_{j1} = \xi_{j1}^{(w+1)}$, $\eta_{j2} = \xi_{j2}^{(w+1)}$ and for all $a^{(i)} \in E$, $(a_{kw+1}^{(i)}, \ldots, a_{kw+k}^{(i)}) \neq \eta$. The adversary then decides that $\xi^{(w+1)} = \eta$ and then for all $a^{(i)} \in E$ the value of $f(a^{(i)})$ will depend on $\xi^{(w+1)}$ but be independent on $\xi^{(w+2)}, \ldots, \xi^{(l)}$.

At step $2w+2$ we assume the learner knows $\xi^{(w+1)}$. We will show now that an equivalence query will reveal at most two bits in $\xi^{(w+2)}$. The learner asks one equivalence query with some function g. We have the following cases for g.

Case I.

$$g(x_1, \ldots, x_{lk}) \not\equiv f_{\xi^{(1)}}(v^{(1)}, f_{\xi^{(2)}}(v^{(2)}, \ldots, f_{\xi^{(w+1)}}(v^{(w+1)}, h)\cdots)$$

for any $h(v^{(w+2)}, \ldots, v^{(l)})$.

This case is equivalent to

$$(\forall h)(\exists u^{(1)}, \ldots, u^{(w+1)})$$
$$g(u^{(1)}, \ldots, u^{(w+1)}, x_{(w+1)k+1}, \ldots, x_{lk})$$
$$\not\equiv f_{\xi^{(1)}}(u^{(1)}, f_{\xi^{(2)}}(u^{(2)}, \ldots, f_{\xi^{(w+1)}}(u^{(w+1)}, h)\cdots)$$

Let $h' = g(\xi^{(1)}, \ldots, \xi^{(w+1)}, x_{(w+1)k+1}, \ldots, x_{lk})$. Then for $h = h'$ we have

$$(\exists z^{(1)}, \ldots, z^{(w+1)})$$
$$g(z^{(1)}, \ldots, z^{(w+1)}, x_{(w+1)k+1}, \ldots, x_{lk})$$
$$\not\equiv f_{\xi^{(1)}}(z^{(1)}, f_{\xi^{(2)}}(z^{(2)}, \ldots, f_{\xi^{(w+1)}}(z^{(w+1)}, h')\cdots).$$

If $(z^{(1)}, \ldots, z^{(w+1)}) = (\xi^{(1)}, \ldots, \xi^{(w+1)})$ then we get a contradiction. Therefore, for some $(z^{(1)}, \ldots, z^{(w+1)}) \neq (\xi^{(1)}, \ldots, \xi^{(w+1)})$ we have

$$g(z^{(1)}, \ldots, z^{(w+1)}, x_{(w+1)k+1}, \ldots, x_{lk}) \not\equiv f_{\xi^{(1)}}(z^{(1)}, f_{\xi^{(2)}}(z^{(2)}, \ldots, f_{\xi^{(w+1)}}(z^{(w+1)}, h')\cdots).$$

Now since $(z^{(1)}, \ldots, z^{(w+1)}) \neq (\xi^{(1)}, \ldots, \xi^{(w+1)})$ by the property in $(\star)$ we have $f_{\xi^{(1)}}(z^{(1)}, f_{\xi^{(2)}}(z^{(2)}, \ldots, f_{\xi^{(w+1)}}(z^{(w+1)}, h')\cdots) = \tau$ for some $\tau \in \{0, 1\}$. The adversary chooses some vector $a = (z^{(1)}, \ldots, z^{(w+1)}, x^{(0)}_{(w+1)k+1}, \ldots, x^{(0)}_{lk})$ such that $g(a) \neq \tau$ and returns the assignment a as a counterexample. In this case the learner can lean nothing about $\xi^{(w+2)}$.

Case II.

$$g(x_1, \ldots, x_{lk}) \equiv f_{\xi^{(1)}}(v^{(1)}, f_{\xi^{(2)}}(v^{(2)}, \ldots, f_{\xi^{(w+1)}}(v^{(w+1)}, h)\cdots)$$

for some $h(v^{(w+2)}, \ldots, v^{(l)})$.

Now we have the following subcases.

1. $h(\eta_{(w+2)k+1}, \ldots, \eta_{lk})$ for some $(\eta_{(w+2)k+1}, \ldots, \eta_{(w+2)k+k}) \neq (0, \ldots, 0)$. Then the adversary returns

   $$a = (\xi^{(1)}, \ldots, \xi^{(w+1)}, \eta_{(w+2)k+1}, \ldots, \eta_n)$$

   as a counterexample (here $g(a) = 0$). Let $\eta_{(w+1)k+h} \neq 0$ for $1 \leq h \leq k$. The adversary will also assume that $\xi_h^{(w+2)} = 0$. This bit makes sure that $f(a) = 1$ where f is the target function.
2. $h(0, \ldots, 0, \eta_{(w+3)k+1}, \ldots, \eta_{lk}) = 1$ for some η_i. Then the adversary returns

   $$a = (\xi^{(1)}, \ldots, \xi^{(w+1)}, 0, \ldots, 0, \eta_{(w+3)k+1}, \ldots, \eta_n)$$

   as a counterexample (here $g(a) = 1$). The adversary will also assume that $\xi_1^{(w+2)} = 1$. This bit makes sure that $f(a) = 0$.

3. If $h(x_{(w+2)k+1}, \ldots, x_{lk}) = x_{(w+2)k+1} \vee \cdots \vee x_{(w+2)k+k}$, then the adversary returns

$$a = (\xi^{(1)}, \ldots, \xi^{(w+1)}, 1, 0, \ldots, 0)$$

as a counterexample (here $g(a) = 1$) and defines $\xi_1^{(w+2)} = \xi_2^{(w+2)} = 1$ to make sure that $f(a) = 0$.

In case II the adversary's counterexample depends only on $\xi^{(w+1)}$ and two bits in $\xi^{(w+2)}$. Therefore to learn the function the learner needs at least l parallel steps. □

8. Conclusion and open problems

We showed that any class of boolean functions that cannot be learned with poly$(\log n)$ equivalence queries cannot be learned efficiently in parallel. In other words, if a class is efficiently learnable in parallel with a polynomial number of processors then it is efficiently learnable in parallel with one processor. This shows that we cannot speed up learning algorithms that use only equivalence queries. On the other hand, many efficient parallel learning algorithms are known for the PAC-learning model (Vitter & Lin, 1992). It would be interesting to prove that certain classes can be efficiently probabilisticly exactly learned using counterexamples. A class C is probabilisticly exactly learnable (PEC-learnable) according to a distribution D if it is exactly learnable with D-equivalence queries. Here the answer of the D-equivalence query with a hypothesis h is a counterexample a that is chosen according to the distribution D_A where D_A is the projection of the distribution D on the set of all counterexamples $A = \{x \mid f(x) \neq h(x)\}$.

We then give a technique for proving lower bounds for parallel exact learning using membership and equivalence query. This technique is based on the fact that any two independent equivalence queries can be changed to one equivalence query and one membership query. This implies that a lower bound for the number of equivalence queries in any learning algorithm that uses a polynomial number of membership queries is a lower bound on the number of parallel steps of the algorithm. We then give lower bounds for the number of equivalence queries needed to learn certain classes showing that they are not efficiently learnable in parallel. These classes include read-once DNF, k-DNF, k-term DNF, monotone DNF and any class that contains one of them.

This approach cannot be used for monotone read-once formulas because this class is learnable from membership queries only. Here we developed a second technique to show that parallel membership queries and equivalence queries cannot speed up the learnability of a subclass of monotone read-once formulas by more than an $O(\log n)$ factor. We then show an $n/\log n$ lower bound for the parallel steps needed to learn this class.

An interesting open problem is to show parallel PEC-learnability of the above classes. Also, finding a parallel PAC-learning algorithm with membership queries for read-once formulas and automatas would be interesting. An efficient parallel PAC-learning algorithm with membership queries for decision trees can be found in Bshouty (1996).

Acknowledgments

Research supported in part by WSERC of Canada. A preliminary version of this paper appears in (Bshouty & Cleve, 1992).

References

Angluin, D. (1987). Queries and concept learning. *Machine Learning*, 2(4):319–342.

Angluin, D., Hellerstein, L., & Karpinski, M. (1993). Learning read-once formulas with queries. *Journal of ACM*, 40(1):185–210.

Balcázar, J.L., Díaz, J., Gavaldà, R., & Watanabe, O. (1994). An optimal parallel algorithm for learning DFA. In *The 1994 Workshop on Computational Learning Theory* (pp. 208–217).

Bshouty, N.H. (1996). Toward the learnability of DNF formulae. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (pp. 131–140).

Bshouty, N.H., & Cleve, R. (1992). On the exact learning of formulas in parallel. *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science* (pp. 24–27).

Bshouty, N.H., Goldman, S., Hancock, T., & Matar, S. (1993). Asking questions to minimize errors. In *The 1993 Workshop on Computational Learning Theory* (pp. 41–50).

Bshouty, N.H., Hancock, T.R., & Hellerstein, L. (1995). Learning boolean read-once formulas over generalized basis. *Journal of Computer and System Sciences*, 50(3):521–542.

Bshouty, N.H., Hancock, T.R., Hellerstein, L., & Karpinski, M. (1994). An algorithm to learn read-once threshold formulas and transformations between learning models. *Computational Complexity*, 4:37–61.

Hancock, T. (1990). Identifying μ-formula decision trees with queries. In *The 1990 Workshop on Computational Learning Theory* (pp. 23–37).

Hancock, T., & Hellerstein, L. (1991). Learning read-once formulas over fields and extended bases. In *The 1991 Workshop on Computational Learning Theory* (pp. 326–336).

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear threshold algorithm. *Machine Learning*, 2(4):285–318.

Maass, W., & Turán, G. (1990). On the complexity of learning from counterexamples and membership queries. In *Proceedings of the 31st Symposium on Foundations of Computer Science* (pp. 203–210).

Maass, W., & Turán, G. (1992). Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9:104–145.

Maass, W., & Turán, G. (1994). Algorithms and lower bounds for on-line learning of geometrical concepts, *Machine Learning*, 14:251–202.

Valiant, L.G. (1984). A theory of the learnable. *Communications of the ACM*, 27:1134–1142.

Vitter, J.S., & Lin, J. (1992). Learning in parallel. *Information and Computation* (pp. 179–202).

Received May 23, 1995
Accepted July 3, 1996
Final Manuscript July 22, 1996