Attribute-efficient learning in query and mistake-bound models

Nader Bshouty*

Department of Computer Science

University of Calgary

Calgary, Alberta, CANADA

Lisa Hellerstein†

Department of Computer and Information Science

Polytechnic University

Brooklyn, NY, USA

*Partially supported by NSERC of Canada.

†Partially supported by NSF Grants CCR-9501660 and CCR-9210957.

Please send proofs to:

Lisa Hellerstein

Department of Computer and Information Science

Polytechnic University

5 Metrotech Center

Brooklyn, NY 11201

Proposed running head: Attribute-efficient learning

Abstract

We consider the problem of *attribute-efficient* learning in query and mistake-bound models. Attribute-efficient algorithms make a number of queries or mistakes that is polynomial in the number of relevant variables in the target function, but only sublinear in the number of irrelevant variables. We consider a variant of the membership query model in which the learning algorithm is given as input the number of relevant variables of the target function. We show that in this model, any projection and embedding closed class of functions (including parity) that can be learned in polynomial time can be learned attribute-efficiently in polynomial time. We show that this does not hold in the randomized membership query model. In the mistake-bound model, we consider the problem of learning attribute-efficiently using hypotheses that are formulas of small depth. Our results extend the work of Blum et al. [4] and Bshouty et al. [7].

1 Introduction

Consider the problem of learning an unknown Boolean function on n variables where n is large. Suppose that the output of this function is completely determined by the values of a fixed set of r of the n variables, where r is small. Thus our real task is to learn a function of r variables; our task is made more difficult because we don't know which of the n variables are the r relevant ones. Can we still learn "efficiently" even without this knowledge?

We consider this problem in the context of two standard models in computational learning theory, the *membership query* model, and the *mistake bound* model. We also consider the *equivalence query* model, which is closely related to the mistake-bound model. We define these models precisely in Section 2. In query models, the learning algorithm has access to oracles which answer questions about the unknown function. One goal in designing query algorithms is to minimize the number of queries asked. The mistake-bound model is an on-line model in which the learning algorithm is asked to predict the value of the hidden function on a sequence of input values. Following each prediction, the algorithm is told whether its prediction was correct. In designing mistake-bound algorithms, we would like to minimize the *mistake-bound* of the algorithm, which is the number of prediction mistakes it makes on a worst-case input sequence.

An *attribute-efficient* algorithm is one that performs "well" in the presence of irrelevant attributes (variables). More formally, let $I : \mathcal{N} \to \mathcal{N}$ be such that $I(n) = o(n)$, where $\mathcal{N}$ is the set of natural numbers. A mistake-bound algorithm is $I(n)$ *attribute-efficient* if its mistake-bound is $O(P(r,s)I(n))$ for some polynomial P, where s is the size of the target function, n is the number of variables on which the function is defined, and $r \leq n$ is the number of relevant variables. A query algorithm is $I(n)$ attribute-efficient if for some polynomial P, it makes $O(P(r,s)I(n))$

queries. Thus attribute-efficient algorithms make a number of mistakes or queries which has only a sublinear dependence on the number of irrelevant attributes in the target function.

Littlestone developed a polynomial time, $\log n$ attribute-efficient algorithm for learning threshold functions in the mistake-bound model [15]. Subsequently, Blum, Hellerstein, and Littlestone [4] considered the following general question: If a class of functions can be learned in polynomial time in a query or mistake-bound model, can it be learned by a polynomial-time $I(n)$ attribute-efficient algorithm in that model? (In particular, they considered the cases $I(n) = \log n$ and $I(n) = n^\alpha$ for $0 < \alpha < 1$.) This remains an open question in the mistake-bound model for all $I(n) = o(n)$. However, Blum et al. did show the following. In the membership and equivalence query model, if a projection and embedding closed (p.e.c.) class is learnable by a polynomial time algorithm, then it is learnable by a polynomial-time $\log n$ attribute-efficient algorithm. (Most natural classes are p.e.c. closed.) In the model of membership queries alone, this same implication holds for all classes containing only monotone functions, but does not hold for all p.e.c. classes—the class of parity functions is a counterexample. Parity functions can, however, be learned by a randomized $\log n$ attribute-efficient algorithm, with some probability of error. We consider questions raised by the above results.

As noted by Blum et al., in the membership query model the learning algorithm is not given as input the number of relevant variables in the target function. The proof that parity functions cannot be learned by a $o(n)$ attribute-efficient algorithm relies on this fact. Consider a variation of the membership query model in which the learning algorithm *is* given as input the number of relevant variables in the target function, or a "reasonable" upper bound. (If the number of relevant variables is small, one can think of this extra input information as guaranteeing a certain simplicity in the target function.) We show (in Section 3) that in this model, the existence of a polynomial-time membership query algorithm does imply the existence of a polynomial-time $\log(n)$ attribute-efficient membership query algorithm for all p.e.c. classes. Thus knowing the number of relevant variables can significantly reduce the number of queries needed in the membership query model. This is in sharp contrast to the mistake-bound model, in which an algorithm that receives the number of relevant variables as input can be transformed into an algorithm that does not receive that information by using a standard doubling technique.

Our result in Section 3 consists of a method for transforming a polynomial-time membership query algorithm into a polynomial-time $\log(n)$ attribute-efficient membership query algorithm. The method is valid for all p.e.c. classes. Subsequent to the publication of a preliminary version of this paper [8], Uehara, Tsuchida, and Wegener obtained bounds on attribute-efficient learning of particular classes of p.e.c. functions [18].

The proof of our result in Section 3 is based on the construction of *coloring sets* (which are closely related to *splitters* [16]).

We give both randomized and deterministic constructions of such sets. Our deterministic construction is similar to a construction of Naor et al. [16] of (n, k, r) splitters of small size.

In Section 4 we consider randomized attribute-efficient algorithms in the membership query model. Since parity functions can be learned by a $\log n$ attribute-efficient randomized algorithm, we ask the following question: if a p.e.c. class of functions can be learned in polynomial time in the membership query model, can it always be learned by an attribute-efficient randomized algorithm? We show that the answer to this question is "no" by presenting a counterexample.

In Section 5, we consider problems of attribute-efficient learning in the (improper) equivalence query model (our results also apply to the mistake-bound model, which is closely related). Let C be a projection-closed class of Boolean functions. If computational time is not a concern, C can be learned by a $\log n$ attribute-efficient algorithm that is a simple variation of the halving algorithm [4]. (We assume here and throughout the paper that the number of functions in C of size s on n variables is at most $2^{\mathrm{poly}(n,s)}$—an assumption that holds for virtually every class of functions considered in the computational learning theory literature.) However, the hypotheses generated by the halving algorithm have extremely complex representations.

We ask whether attribute-efficient learning in the mistake-bound model can be achieved using hypotheses with simple representations. We show that the class of functions represented by single variable formulas, $f(x_1, \ldots, x_n) = x_i$ where $i \in \{1, \ldots, n\}$, is learnable by a $\log(n)$ attribute-efficient mistake-bound algorithm whose hypotheses are polynomial size boolean formulas of depth 3. We also show that polynomial size Boolean formulas of depth 2 are not sufficient for attribute-efficient learning of this class. Littlestone's WINNOW algorithm learns this class $\log n$ attribute-efficiently with hypotheses that are threshold functions [15]. Our techniques extend the results of Bshouty, Cleve, Kannan, and Tamon [7] on equivalence query learning with simple hypotheses.

2 Definitions

We use log to denote the logarithm base 2, and ln to denote the natural logarithm.

Let $V_n = \{x_1, x_2, \ldots, x_n\}$. Let $a \in \{0,1\}^{V_n}$ be an assignment to the variables in V_n. We consider a to be a function from V_n to $\{0,1\}$. If $x_i \in V_n$, and $z \in \{0,1\}$, then $a_{x_i \leftarrow z}$ is the assignment to the variables in V_n such that $a_{x_i \leftarrow z}(x_j) = a(x_j)$ for all $j \neq i$, and $a_{x_i \leftarrow z}(x_i) = z$. Throughout the paper, we assume that if C is a class of functions, and $f \in C$, then $f : \{0,1\}^{V_n} \rightarrow \{0,1\}$ for some natural number n.

A *Boolean formula* is a rooted tree whose internal nodes are each labeled with a member of the set $\{\wedge, \vee, \neg\}$, and whose leaves are each labeled with a variable.

A variable (or attribute) x_i is *relevant* for a function $f : \{0,1\}^{V_n} \rightarrow \{0,1\}$ if there is an assignment $a \in \{0,1\}^{V_n}$ such that $f(a) \neq f(a_{x_i \leftarrow \neg a(x_i)})$. If x_i is relevant, then f *depends on* x_i. Let C be a class of Boolean functions, each of which is defined on V_n for some n. C_n^r is the class of functions in C defined on V_n, such that at most r of the variables of V_n are relevant. The class $C_n = \cup_r C_n^r$ is the class of functions in C defined on V_n.

Given two sets of variables V and V', an *embedding* of the domain $\{0,1\}^V$ in the domain $\{0,1\}^{V'}$ is an injective mapping $\phi : V \rightarrow V'$. Given $\phi : V \rightarrow V'$ and a Boolean function f defined on V, there is a natural way to define a corresponding function f' defined on V': for $b \in \{0,1\}^{V'}$, $f'(b) = f(a)$ where $a(x) = b(\phi(x))$ for all $x \in V$. Let $\phi(f)$ denote f'. The function $\phi(f)$ is an *embedding* of f.

A *partial assignment* p to V is a mapping from V to $\{0,1,*\}$. A variable x is *assigned by* p if $p(x) \neq *$. Let $W \subseteq V$ be the variables assigned by p. Let a be a partial assignment to some V' containing $V - W$ such that a assigns values to all variables in $V - W$. Then p/a is an assignment to V that agrees with p on all variables assigned by p, and agrees with a on the remainder.

Let p be a partial assignment to the variables of V that leaves exactly the variables $V' \subseteq V$ unassigned. Then for $f : \{0,1\}^V \rightarrow \{0,1\}$, f_p denotes the function over $\{0,1\}^{V'}$ defined by

$f_p(a) = f(p/a)$ for all $a \in \{0,1\}^{V'}$. We call f_p a *projection* of f. If $a \in \{0,1\}^{V''}$, where $V' \subset V''$, then $f_p(a)$ is defined to be $f_p(a^*)$, where a^* is the restriction of a to V'. If $x \in V$, we define $f_{x \leftarrow 1}$ to be equal to f_{p_1}, where p_1 is the partial assignment to V leaving all variables but x unassigned, and setting x to 1. The projection $f_{x \leftarrow 0}$ is defined analogously.

For each class C of functions, we assume there is an associated class of *representations* R containing at least one representation for every function in C, and a size function $\sigma : R \rightarrow \mathcal{N}$. The *size* of a function $f \in C$, denoted $|f|$, is the minimum, over all $\rho \in R$ representing f, of $\sigma(\rho)$. We also assume that if $\phi(f)$ is an embedding of f, then $|\phi(f)| = |f|$ (so the addition of irrelevant variables does not affect the size of a function), and if f_P is a projection of f, then $|f_P| \leq |f|$. Finally, we assume that if ρ is a representation of a function f defined on V_n, and $\phi(f)$ is an embedding of f into $V_{n'}$ where $n' \geq n$, then computing a representation of $\phi(f)$ from ρ takes time polynomial in n' and the size of ρ.

A class C is *embedding closed* if for all $k \leq n$, and for all embeddings ϕ of V_k into V_n, if $f \in C_k$, then $\phi(f) \in C_n$. C is *projection closed* if for all i, and for all partial assignments P to the variables in V_i, $\phi(f_P) \in C_{|W|}$, where W is the set of variables not assigned by P, and ϕ is an embedding of W into $V_{|W|}$.

In both the mistake-bound and membership query models of learning, the goal is to "learn" an unknown target function f taken from some fixed class of functions C. In the mistake-bound model [15], learning proceeds in a sequence of trials. In each trial the learning algorithm is given an assignment $a \in \{0,1\}^{V_n}$, where V_n are the variables over which the target f is defined. The algorithm then predicts the value of $f(a)$, and is told whether the prediction was correct. We count the number of trials in which the algorithm's prediction is incorrect (the number of *mistakes*). In this model, the mistake bound of an algorithm for learning C is $P(n, s)$ if for any target function $f \in C_n$ of size at most s, and any sequence of assignments, the algorithm makes at most $P(n, s)$ mistakes. Analogously, the algorithm makes $P(n, r, s)$ mistakes if for any $f \in C_n^r$ and any sequence of assignments, it makes at most $P(n, r, s)$ mistakes.

In the membership query model (cf. [1]), the learning algorithm is given as input n, where V_n is the set of variables on which the target f is defined. The algorithm has access to a *membership oracle* for f, which takes as input an assignment a, and returns $f(a)$. Queries to this oracle are called *membership queries.* The algorithm eventually halts and outputs a representation of f. The algorithm *learns C with $P(n, s)$ queries* if for any $f \in C_n$ of size at most s, it outputs a representation of f after making at most $P(n, s)$ queries. It learns C with $P(n, r, s)$ queries if for any $f \in C_n^r$ of size s, it outputs a representation of f after making at most $P(n, r, s)$ queries.

A membership oracle for a function f can be used to simulate a membership oracle for any projection f_p of f, because for any assignment a to the variables of f_p, $f(p/a) = f_p(a)$.

The mistake-bound model is closely related to the *equivalence query model* defined in [1]. In this model, the algorithm asks queries of an *equivalence oracle*, which takes as input (the representation of) a hypothesis h from the class C. If h is equivalent to the target function f, then the oracle answers "yes." Otherwise, the oracle returns a *counterexample*, an assignment a such that $h(a) \neq f(a)$. In an extended definition of this model, the hypotheses h may come from a class other than C; in this case the equivalence queries are *improper.* Equivalence query algorithms receive n as input, ask queries, and eventually output a representation of the target f.

In the mistake-bound model, an algorithm is *polynomial time* if the worst case time spent in computation during each trial is polynomial in n and s. In the query models, a learning algorithm is polynomial time if its *total* running time is polynomial in n and s. We assume that each query is answered in constant time. (A more precise definition of the computational model can be found in [3]). Algorithms in any of the above models need time at least linear in n just to read or specify an assignment. Thus we do not hope to have algorithms whose time complexity has a sublinear dependence on n.

We note that in the infinite attribute model of Blum, it is possible to have algorithms whose time complexity is sublinear in n [5]. However, we do not consider that model here.

If there exists a polynomial-time mistake-bound algorithm for learning C that makes at most $Q(n, r, s)$ mistakes, then there exists a polynomial-time improper equivalence query algorithm for learning C that makes at most $Q(n, r, s) + 1$ queries. Conversely, if there exists a (proper or improper) polynomial-time equivalence query algorithm for learning C that makes at most $Q(n, r, s)$ queries, then there exists a mistake-bound algorithm for learning C that makes at most $Q(n, r, s)$ mistakes [15]. Mistake-bound algorithms implicitly use a hypothesis in predicting the value $f(a)$.

A randomized membership query algorithm for learning C is one that for any $f \in C$, with probability at least 2/3 outputs (a representation of) f. A randomized $I(n)$ attribute-efficient algorithm for learning C is one that for any $f \in C$ with n variables and at most r relevant variables, with probability at least 2/3 asks $O(\text{poly}(r, s)I(n))$ queries, and outputs f. It is polynomial time if with probability at least 2/3 it does all the above within time $\text{poly}(n, s)$. With probability at most 1/3 it runs for too long, asks too many queries, or outputs a wrong hypothesis.

3 Attribute-efficient learning when r is known

Let $\mathcal{A}$ be a polynomial-time algorithm that learns a p.e.c. class C with $Q(n,s)$ membership queries. We show that there exists a polynomial-time algorithm that learns C using $Q'(r,n,s) = \text{poly}(r,s,\log n)$ membership queries, given r as input, where r is an upper bound on the number of relevant variables of the target. The intuition behind our algorithm is as follows. We would like to partition the variables on which the target function is defined, such that each set in the partition contains at most one relevant variable, and such that the total number of sets in the partition is not much larger than r. Given such a partition, we could treat the variables in each set in the partition as a single, meta-variable, and learn the target function as if it were defined on these (relatively few) meta-variables. Finally, we could discover the at most one actual relevant variable in each set of the partition.

Unfortunately, we are not able to immediately generate such a partition. Instead, we construct a set of partitions, many of which have the desired property. From this set of partitions, we will find one with the desired property. Our construction of the partitions is based on the following definition.

Definition: A t *coloring set of* n *elements with* c *colors* is a set $A \subseteq [c]^n$ where $[c] = \{1,\ldots,c\}$ and $|A| = t$. For $0 \leq \mu < 1$, a t coloring set of n elements with c colors is called an (n,k,c,t,μ)-*coloring set* if for every $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ there are greater than $\mu|A|$ colorings $a \in A$ such that $|\{a_{i_1},\ldots,a_{i_k}\}| = k$. That is, for every k elements, a fraction greater than μ of the colorings in A color the k elements with different colors. (In the terminology of Uehara et al., an (n,k,c,t,μ)-*coloring set* is an (n,k,c,μ)-*splitter of size* t [18]). Where c and t are clear from the context, we refer to (n, k, μ)-coloring sets.

Lemma 1 *Suppose C is projection closed and there exists a polynomial-time algorithm A that learns C with membership queries. Then there exists a membership query algorithm that, given a membership query oracle for some $f \in C_r$, and a variable $x \in V_r$, decides in time polynomial in r whether f depends on x.*

Proof: Run two copies of algorithm A "in parallel" (by interleaving the steps of the two algorithms), to learn $f_{x \leftarrow 1}$ and $f_{x \leftarrow 0}$ respectively. In the copies, answer membership queries by simulating the oracles for $f_{x \leftarrow 1}$ and $f_{x \leftarrow 0}$ using the membership oracle for f.

If the answer to each membership query of A is identical in both copies, then they both must output the same final hypothesis. Thus $f_{x \leftarrow 1} = f_{x \leftarrow 0}$, and hence f does not depend on x. Otherwise, some membership query of A is answered differently in the two copies. This query is an assignment a such that $f_{x \leftarrow 1}(a) \neq f_{x \leftarrow 0}(a)$ and thus f depends on x. $\square$

Lemma 2 *Let C be a p.e.c. class that can be learned with membership queries in time polynomial in n and s using at most $Q(n, s)$ queries, where Q is non-decreasing in n and s. For $r < n$, let $c(n, r)$, $t(n, r)$, be functions such that an $(n, r, c(n, r), t(n, r), \frac{1}{2})$-coloring set can be constructed in time polynomial in n. Then there exists an algorithm for learning C with membership queries that runs in time polynomial in n and s, and makes $O(c(n, r)t(n, r)Q(c(n, r), s) + rQ(r, s) \log n)$ queries, if it is given as input r, where r is an upper bound on the number of relevant variables in the target.*

Proof: We describe and analyze the algorithm that takes r as input. Without loss of generality, assume that $r < n$ (if $r = n$, run the original algorithm that makes $Q(n, s)$ queries). Let $c = c(n, r)$ and $t = t(n, r)$. Let f be the target function.

1. On input r, construct an $(n, r, c, t, \frac{1}{2})$-coloring set. Let $a_1, \ldots, a_t \in [c]^n$ be the coloring set. Each coloring a_j in the coloring set colors the variables in $V_n = \{x_1, \ldots, x_n\}$ using c colors. The coloring a_j partitions the variables in V_n by color into c sets. Viewing each set as a meta-variable, there is a natural mapping τ_j from assignments to the c meta-variables to assignments to $x_1, \ldots, x_n$; the value assigned to a meta-variable is given to the variables in V_n that comprise that meta-variable. Formally, $\tau_j : \{0,1\}^{V_c} \to \{0,1\}^{V_n}$ such that for all $x_k \in V_n$, $x_i \in V_c$, and $b \in \{0,1\}^{V_c}$, $(\tau_j(b))(x_k) = b(x_i)$ iff $a_j[k] = i$. Now define $f_j : \{0,1\}^{V_c} \to \{0,1\}$ such that for all $b \in \{0,1\}^{V_c}$, $f_j(b) = f(\tau_j(b))$.

Recall that f depends on at most r variables, and hence is the embedding of a function f' of at most r variables into a larger domain of n variables. If the coloring a_j colors all relevant variables of f with different colors, then f_j is an embedding of f' into a domain of c variables. In this case, $f_j \in C$ and $|f_j| = |f| = s$. We call such a coloring *good*. If a_j does not color all relevant variables of f with different colors, then the number of relevant variables of f_j will be less than the number of relevant variables of f, f_j may or may not be a member of C, and $|f_j|$ may be greater than s.

2. For each of the t functions f_j, prepare to run the procedure of Lemma 1 on all variables in V_c to determine which of them are relevant to f_j. Run these t procedures "in parallel" (by interleaving their steps), simulating the oracle for f_j using the oracle for f in the obvious way. When more than half of these procedures terminate normally, abort the remainder of

the procedures.

3. Let $J = \{j|$ the procedure to find the relevant variables of f_j terminated normally$\}$. For $j \in J$, let W_j be the set of variables output by the procedure for f_j. Let $j^* \in J$ be such that $|W_{j^*}|$ is maximum.

If f_j is not in C, then W_j may not contain all the relevant variables of f_j (because the behavior of the learning algorithm for C is unpredictable given a membership oracle for a function not in C). However, whether or not f_j is in C, each variable x_i in W_j is relevant to f_j, because the procedure in Lemma 1 verifies that the projection of f_j setting x_i to 0 is not equal to the projection of f_j setting x_i to 1.

If a variable x_i is relevant in f_j, then the set of variables of f colored i by a_j must contain at least one relevant variable of f. If a_j is not a good coloring, then the number of variables in W_j is less than the number of relevant variables of f. Thus the number of variables in W_j is equal to the number of relevant variables of f iff a_j is a good coloring.

Because $a_1, \ldots, a_t \in [c]^n$ is an $(n, r, c, t, \frac{1}{2})$-coloring set, more than half the a_i are good colorings. Thus there exists at least one $j \in J$ such that a_j is a good coloring. It follows that a_{j^*} is a good coloring. Furthermore, for all good colorings a_i, learning f_i requires at most $Q(c, s)$ queries. Thus running the procedure of Lemma 1 to find the relevant variables of f_i requires at most $cQ(c, s)$ queries. Therefore, the t procedures will all terminate or be aborted by the time each makes at most $cQ(c, s)$ queries, for a total of at most $tcQ(c, s)$ queries.

4. Let $W_{j^*} = \{x_{i_1}, \ldots, x_{i_{r^*}}\}$. Let $r^* = |W_{j^*}|$. For each $x_{i_z} \in W_{j^*}$, let S_z be the set of variables of f colored i_z by a_{j^*}. Each set S_z contains exactly one relevant variable of f. Since r

is an upper bound on the number of relevant variables of f, $r^* \leq r$. The variables in $V_n - \{S_1 \cup \ldots \cup S_{r^*}\}$ are irrelevant to f. Let $\hat{f}$ be the projection of f setting the variables in $V_n - \{S_1 \cup \ldots \cup S_{r^*}\}$ to 0.

5. For each S_z, use the following binary search technique to find the relevant variable in S_z.

   - Initialize a set S to be equal to S_z. If $|S| = 1$, then the one variable in S is relevant and we are done. Otherwise, let U be a set containing half the variables of S. Let $\hat{f}_p$ be a projection of $\hat{f}$ formed by setting the variables in U to 0. Let $\gamma : \{0,1\}^{W_{j^*}} \rightarrow \{0,1\}^{S_1 \cup \ldots \cup S_{r^*}}$ such that for all $b \in \{0,1\}^{W_{j^*}}$, $\gamma(b)$ is defined as follows: for all $x_i \in S_1 \cup \ldots \cup S_{r^*}$, $(\gamma(b))(x_i) = b(x_{a_{j^*}[i]})$.
   - Let $g : \{0,1\}^{W_{j^*}} \rightarrow \{0,1\}$ such that for all $b \in \{0,1\}^{W_{j^*}}$, $g(b) = \hat{f}(\gamma(b))$, and let $g' : \{0,1\}^{W_{j^*}} \rightarrow \{0,1\}$ such that $g'(b) = \hat{f}_p(\gamma(b))$.
   - Run the membership query algorithm for C to learn g and g' (simulating the membership queries for g and g' with the oracle for f). If the answers to the corresponding queries in both algorithms are identical, then reset S to be $S - U$ and repeat the above procedure. Otherwise, reset S to be U.

6. Let R be the set of relevant variables of f obtained from all the sets S_z.

To see that the binary search procedure works, note that g depends on $r^* \leq r$ variables, g is a projection and embedding of f, and hence $g \in C$. If p does not set any relevant variables of f to 0, then $g = g'$, and $g' \in C$. In this case, the answers to corresponding queries in the membership query algorithms for learning g and g' will be the same. Otherwise, $\hat{f}_p$ depends on fewer than r^* variables, g' is a projection and embedding of $\hat{f}_p$, $g \neq g'$, and $g' \in C$. In this case, the corresponding answers to the queries will not be the same because the two algorithms must output different functions in C.

In performing the binary search procedure on a set S_z, we do $O(\log n)$ runs of the membership query algorithm for learning functions of r variables. So, we make $O(Q(r,s)\log n)$ queries to find the relevant variables in each set S_z. There are at most r sets S_z, so we make a total of $O(rQ(r,s)\log n)$ queries in finding R.

7. Set the irrelevant variables of f (those in $V_n - R$) to 0. This yields a projection $f' : \{0,1\}^R \rightarrow \{0,1\}$ of f such that f is an embedding of f'. We use the algorithm for C_r to learn f', and output the result.

Learning f' takes $Q(r,s)$ queries.

The total number of queries executed by the above algorithm is $O(ctQ(c,s) + rQ(r,s)\log n)$.

□

It remains to construct the coloring set. We first give a simple randomized construction of an $(n, r, O(r^2), O(r\log n), \mu)$-coloring set. With $\mu = \frac{1}{2}$, this construction yields a randomized algorithm.

Randomized Construction Let $\mu = 1 - \epsilon$ be a constant between 0 and 1. Given n and $r < n$, randomly uniformly choose $\lceil (8/\epsilon^2)(r\ln n + \ln(1/\delta))\rceil$ vectors (colorings) in $[\frac{r^2}{\epsilon}]^n$.

Lemma 3 *The randomized construction produces, with probability at least* $1-\delta$, *an* $(n, r, r^2/\epsilon, \lceil (8/\epsilon^2)(r\ln n + \ln(1/\delta))\rceil, 1-\epsilon)$*-coloring set for any constant* $0 < \epsilon < 1$.

Proof: The construction randomly colors n elements with c colors t times, where $c = \frac{1}{\epsilon}r^2$. and $t = \frac{8}{\epsilon^2}(r \ln n + \ln \frac{1}{\delta})$. The probability that the elements in a given set R of r elements are colored with different colors in the first coloring is

$$\frac{\binom{c}{r} r!}{c^r} \;=\; \left(1 - \frac{1}{c}\right)\left(1 - \frac{2}{c}\right)\cdots\left(1 - \frac{r-1}{c}\right) \;\geq\; 1 - \frac{r(r-1)}{2c}.$$

The inequality in the above follows by induction on r. Since $c = \frac{1}{\epsilon}r^2$, this probability is greater than $1 - \epsilon/2$. By a standard Chernoff bound (see e.g. [17], Theorem 4.2, p. 70), the probability that, for a given set R of r entries, the elements of R are colored differently in fewer than $(1-\epsilon)t$ of the t colorings is at most $e^{-\frac{\epsilon^2 t}{8(1-\epsilon/2)}}$. Therefore the probability that the colorings do not form an $(n, r, \frac{r^2}{\epsilon}, t, 1-\epsilon)$-coloring set is at most

$$\binom{n}{r} e^{-\frac{\epsilon^2 t}{8}}.$$

Because $t = \frac{8}{\epsilon^2}(r \ln n + \ln \frac{1}{\delta})$, the lemma follows.$\Box$

Combining the above construction with the proof of Lemma 2 yields a randomized algorithm. In particular, setting $\epsilon = \frac{1}{2}$ proves the following theorem.

Theorem 1 *Let C be a p.e.c. class of functions that is learnable by a polynomial-time membership query algorithm using at most $Q(n, s)$ queries, where Q is non-decreasing in n and s. Then there exists a polynomial-time randomized algorithm for learning C with membership queries using $O(r^3 Q(2r^2, s) \log n)$ membership queries, if it is given as input r, where r is an upper bound on the number of relevant variables in the target.*

If, in the randomized algorithm, input r equals the number of relevant variables, then because Q is bounded above by a polynomial (because the original algorithm runs in polynomial time), the algorithm is $\log n$ attribute-efficient. This also holds if r is bounded above by a polynomial in the number of relevant variables.

We now present a deterministic coloring construction which yields an algorithm that is deterministic. Like the randomized algorithm, it is $\log n$ attribute-efficient. However, it does have a higher query complexity in r and s than the randomized algorithm. Our deterministic construction is based on a class of error correcting codes developed by Alon et al. [2] that have asymptotically good low-rate. The construction is similar to a previous construction of Naor et al. [16] for (n, k, r) splitters of small size.

Deterministic Construction Given an alphabet Σ, a *code of length m over the alphabet Σ* is a set $C \subseteq \Sigma^m$. The elements of C are called *code-words.* For two code-words $u, v \in C$ the *distance* $d(u, v)$ is the number of indices i such that $u_i \neq v_i$. The *minimal relative distance* of a code C is

$$\min_{u,v\in C \text{ and } u\neq v} \frac{d(u,v)}{m}.$$

We now prove the following lemma.

Lemma 4 *For any constant $0 < \mu < 1$ there is an explicit construction of an $(n, r, r^2, O(r^6 \log r \log n), \mu)$-coloring set.*

Proof: Let $k = r/\sqrt{1-\mu}$. We use the code in [2]. This code is n code-words $v_1 \ldots, v_n$ of length $L = O(k^6 \log k \log n)$ over the alphabet $[k^2]$, with a minimal relative distance of at least $1 - 2/k^2$. Now define the coloring $A = \{a_1, \ldots, a_L\} \subseteq [r]^n$ where $a_{i,j} = v_{j,i}$. We now show that this coloring is an (n, r, r^2, L, μ)-coloring set.

Let $1 \leq i_1 < \cdots < i_r \leq n$. Since for all $1 \leq j_1 \leq j_2 \leq k$, v_{j_1} and v_{j_2} have relative distance at least $1 - 2/k^2$, there are at least

$$\left(1 - \frac{2}{k^2}\binom{r}{2}\right) L \geq \mu L$$

indices l such that $v_{i_1,l}, \ldots, v_{i_r,l}$ are distinct. This implies that there exist at least μL colorings a_l such that $|\{a_{l,i_1}, \ldots, a_{l,i_r}\}| = r$. □

Combining this construction with Lemma 2 gives the following theorem.

Theorem 2 *Let C be a class of functions that is learnable by a polynomial-time membership query algorithm that makes at most $Q(n,s)$ queries, where Q is non-decreasing in n and s. Then there exists a polynomial-time algorithm for learning C with membership queries that makes $O(r^8 \log r Q(r^2, s) \log n)$ membership queries, if it is given as input r, where r is an upper bound on the number of relevant variables in the target.*

A deterministic construction in an earlier version of this paper relies on a number-theoretic coloring technique [8]. This technique is a variation of one used in constructing small, constant depth, circuits for symmetric functions [12]. A similar technique was used earlier in constructing short monotone formulas for threshold-k functions [14]. (Constructions producing shorter formulas appeared in e.g. [9], but seem to have no relation to our learning problems.) The number of queries in the algorithm obtained from this construction is upper bounded by $\tilde{O}(r^6 Q(r^2, s) \log^{1+\epsilon} n)$ for any $\epsilon > 0$ (here we use $\tilde{O}$ to denote the complexity after removing factors polynomial in $\log r$ and $\log s$). Although this construction is not $\log n$ attribute-efficient, it gives a better complexity when $r = \Omega(\{log^{1+\epsilon} n)$, for any epsilon.

4 Negative result for attribute-efficient randomized learning

Let C consist of the contant functions 0 and 1, and all functions represented by terms (conjunctions of literals) containing at most one negated variable. This class is p.e.c. The class C is learnable with $O(n)$ membership queries as follows. Ask a membership query with the assignment $\bar{1}$ setting all variables to 1. If $f(\bar{1}) = 1$ then the term is monotone, and can be learned with n queries. If the term is not monotone then ask n membership queries on all assignments $\bar{e}_i$, $i = 1, \ldots, n$ where $\bar{e}_i$ is the $\bar{1}$ assignment but with x_i set to 0. Use those assignments to find the negated variable. The other monotone variables in the term can then be learned with n membership queries. We show that C cannot be learned by a randomized polynomial-time $o(n)$ attribute-efficient algorithm.

Theorem 3 *There is no $o(n)$ attribute-efficient randomized membership query algorithm for learning C.*

Proof:

For the purpose of contradiction, suppose that there exists an $o(n)$ attribute-efficient randomized membership query algorithm for learning C. Then there exists a randomized membership query algorithm $\mathcal{A}$ for learning C such that for any function $f \in C_n^r$ (for large enough n), A asks at most $n/2$ queries and, with probability at least $2/3$, outputs f.

Without loss of generality, we can assume that $\mathcal{A}$ asks exactly $n/2$ queries and always outputs a hypothesis: We can prevent $\mathcal{A}$ from asking more than $n/2$ queries by making it halt, and we can prevent $\mathcal{A}$ from asking fewer than $n/2$ queries by having it ask its last query repeatedly. We can also force $\mathcal{A}$ to output an arbitrary hypothesis rather than no hypothesis.

Let $\mathcal{T}$ be the set of *full terms* T with one negated variable. A full term is one that contains n literals, one for each variable. It is satisfied by a unique assignment. For $T \in \mathcal{T}$, let $S^T = \{a \in \{0,1\}^{V_n} | T(a) = 0\}$.

For $T \in \mathcal{T}$, let $a_1, \ldots, a_i$ be a sequence of queries such that $a_1, \ldots, a_{i-1} \in S^T$, and $a_i \notin S^T$. Note that the answers to queries $a_1, \ldots, a_{i-1}$ are the same whether the target is T or $f = 0$. Therefore, the probability that $\mathcal{A}$ asks a sequence of $n/2$ queries that begins with the prefix $a_1, \ldots, a_i$ is the same whether the target is T or $f = 0$.

Similarly, for target $T \in \mathcal{T}$, the probability that $\mathcal{A}$ asks a sequence of queries $a_1, \ldots, a_{n/2} \in S^T$ and then outputs the hypothesis T is identical whether the target is T or $f = 0$, because the answers to all the queries are the same in both cases. If the target is $f = 0$, the probability that $\mathcal{A}$ does not output the hypothesis $h = 0$ is at most $\frac{1}{3}$.

Consider a run of $\mathcal{A}$ with the target function $f = 0$. Let $a_1, \ldots, a_{n/2} \in \{0,1\}^{V_n}$ be the sequence of queries asked. Since $\mathcal{A}$ is randomized, $a_1, \ldots, a_{n/2}$ are distributed according to a fixed joint distribution D^*. For $i \in [n/2]$, let D_i be the distribution induced on a_i by D^*.

Therefore

$$
\begin{aligned}
&\sum_{T \in \mathcal{T}} \Pr[\mathcal{A} \text{ outputs } T \text{ on target } T] \\
&\quad \leq \sum_{T \in \mathcal{T}} (\Pr[\mathcal{A} \text{ asks a query } a_i \notin S^T \text{ on target } T] + \\
&\qquad\qquad \Pr[\mathcal{A} \text{ asks only queries in } S^T \text{ and outputs } T \text{ on target } T]) \\
&\quad \leq \sum_{T \in \mathcal{T}} (\Pr[\mathcal{A} \text{ asks a query } a_i \notin S^T \text{ on target } T] + \\
&\qquad\qquad \Pr[\mathcal{A} \text{ asks only queries in } S^T \text{ and outputs } T \text{ on target } f = 0])
\end{aligned}
$$

$$
\begin{aligned}
&\leq \sum_{T \in \mathcal{T}} (\Pr[\mathcal{A} \text{ asks a query } a_i \notin S^T \text{ on target } T] + \\
&\qquad\qquad \sum_{T \in \mathcal{T}} \Pr[\mathcal{A} \text{ outputs } T \text{ on target } f \ = \ 0]) \\
&\leq \sum_{T \in \mathcal{T}} \sum_{i=1}^{n/2} \Pr_{D^*}[T(a_i) = 1] \ + \ \frac{1}{3} \\
&= \sum_{T \in \mathcal{T}} \sum_{i=1}^{n/2} \mathcal{E}_{D_i}[T(a_i)] \ + \ \frac{1}{3} \\
&= \sum_{i=1}^{n/2} \sum_{T \in \mathcal{T}} \mathcal{E}_{D_i}[T(a_i)] \ + \ \frac{1}{3} \\
&= \sum_{i=1}^{n/2} \left[\mathcal{E}_{D_i} \left[\sum_{T \in \mathcal{T}} T(a_i) \right] \right] \ + \ \frac{1}{3} \\
&\leq \sum_{i=1}^{n/2} 1 \ + \ \frac{1}{3} \\
&= \frac{n}{2} + \frac{1}{3}
\end{aligned}
$$

Since $|\mathcal{T}| = n$, there must be some term $T \in \mathcal{T}$ such that $\Pr[\mathcal{A}$ outputs T on target $T] \leq (\frac{n}{2} + \frac{1}{3})/n = \frac{1}{2} + \frac{1}{3n}$, which contradicts that $\mathcal{A}$ is a randomized algorithm for learning C. $\square$

5 Equivalence query learning with simple hypotheses

Let C be a class of functions. The standard halving algorithm learns C with equivalence queries [15]. It is not computationally efficient, and we do not focus on issues of computational efficiency in this section. Throughout this section, the term "equivalence query" is meant to include improper equivalence queries.

For $\delta \in (0, \frac{1}{2}]$, a function $B : \{0,1\}^{V_n} \rightarrow \{0,1\}$ is *δ-good for C* if for all $x \in \{0,1\}^n$, $|\{h \in C | h(x) = B(x)\}| \geq \delta |C|$. The halving algorithm keeps a set H of functions $h \in C$ consistent with answers to previous queries, and asks equivalence queries using hypotheses that are $\frac{1}{2}$-good for H. It makes at most $\log|C_n|$ queries.

We review some simple variants of the halving algorithm. If δ-good hypotheses are used instead of $\frac{1}{2}$-good hypotheses, the resulting algorithm makes at most $\frac{1}{\delta}\log|C_n|$ queries.

If the halving algorithm is given as input s, the size of the target function, and r, the number of relevant variables, then it can initialize H to contain just functions in C_n^r of size s. If C is projection-closed, the number of such functions is at most $\binom{n}{r}|C_r^{r(s)}|$, where $C_r^{r(s)}$ denotes the set of functions in C_r^r of size at most s. The number of equivalence queries in this case is $O(r\log n\ \log|C_r^{r(s)}|)$ which is $O(\text{poly}(r,s)\ \log n)$ (recall that we assume there are at most $2^{\text{poly}(r,s)}$ functions in C_r of size s).

By using doubling of r and s simultaneously, we can remove the assumption that r and s are given as input, while still learning within $O(\text{poly}(r,s)\ \log n)$ queries. This version of the halving algorithm is $\log n$ attribute-efficient, but may use complex hypotheses.

We ask whether there exist $\log(n)$ attribute-efficient equivalence query algorithms that use simple hypotheses. Kannan [13] and Bshouty et al. [7] previously considered the problem of learning within a polynomial number of equivalence queries (without consideration of attribute-efficiency) using simple hypotheses.

The results of Bshouty et al. rely on a lemma relating *amplifiers* and δ-good hypotheses.

Definition: Let $0 \le p' < p < q < q' \le 1$. A Boolean function $B(y_1,\ldots,y_m)$ is a $(p,q)\to(p',q')$ amplifier if the following hold:

- If $y_1,\ldots,y_m$ are independently set to 1 with probability at least q,

  $\Pr[B(y_1,\ldots,y_m)=1] \ge q'$;

- If $y_1, \ldots, y_m$ are independently set to 1 with probability at most p,

  $\Pr[B(y_1, \ldots, y_m) = 1] \le p'$.

Lemma 5 *[7] Let $h(z_1, \ldots, z_m)$ be a $(\delta, 1-\delta) \to (2^{-2n}, 1-2^{-2n})$ amplifier. Let H_n be a subset of Boolean functions defined on V_n, and let $f_1, \ldots, f_m$ be randomly chosen uniformly and independently from H_n. Then with probability at least $1 - 2^{-n}$, $h(f_1, \ldots, f_m)$ is δ-good for H_n.*

We give a polynomial-size amplifier that is a depth-3 formula. Most amplifiers in the literature are designed to have small size, rather than small depth.

Lemma 6 *There exists a depth-3 formula $h(z_1, \ldots, z_q)$ where $q = poly(n)$, h has size polynomial in q, and such that $h(z_1, \ldots, z_q)$ is a $(\frac{1}{4}, \frac{3}{4}) \to (2^{-2n}, 1-2^{-2n})$ amplifier.*

Proof: The formula h is constructed out of two different amplifiers. The first is the $(\frac{1}{n}, 1 - \frac{1}{n}) \to (2^{-2n}, 1-2^{-2n})$ amplifier $A(y_1, \ldots, y_\tau)$ described in [7] (see also [13] and [6]), where $\tau = \frac{4n^2}{\log n \log\log n}$. This amplifier is simply a CNF formula with $2n/\log n$ disjoint clauses, each of which contains $2n/\log\log n$ variables. The second amplifier is the $(\frac{1}{4}, \frac{3}{4}) \to (\frac{1}{n}, 1 - \frac{1}{n})$ amplifier $B(y_1, \ldots, y_{c\log n}) = \text{Majority}(y_1, \ldots, y_{c\log n})$. Since B depends on $c\log n$ variables, it can be written as a DNF formula of size polynomial in n. The formula h is constructed by taking the amplifier A, and replacing the input variables y_i of A by disjoint copies of the amplifier B. $\Box$

The next theorem follows immediately from the above lemmas, and the variants of the halving algorithm described above.

Theorem 4 *Let C be a projection-closed class of functions with an associated representation class and size function such that the number of functions in C_n of size s is $2^{poly(n,s)}$. There is* *a* $\log n$ *attribute-efficient equivalence query algorithm for learning* C *with hypotheses of the form* $g(f_1, \ldots, f_z)$*, where* $f_1, \ldots, f_z$ *are functions from* C *of size polynomial in* r *and* s *(where* r *is the number of relevant variables of the target), and* g *is a depth-3 formula of size polynomial in* n*.*

We note that by results of Bshouty et al. [7], a randomized version of the algorithm in the above theorem can be implemented to run in polynomial time using an NP oracle. A deterministic version runs in exponential time and polynomial space.

The above theorem also holds if the statement "g is a depth-3 formula" is replaced by the statement "g is a circuit computing the majority function". This follows easily from results of Bshouty et al. (which relied on the existence of a $(\frac{1}{4}, \frac{3}{4}) \rightarrow (2^{-2n}, 1 - 2^{-2n})$ amplifier computing the majority function). However, the majority function cannot be computed by polynomial size constant depth circuits [11].

The 1/4-good hypotheses implied by Lemmas 5 and 6 can also be used in versions of the halving algorithm in which r and s are not doubled. For example, we have the following corollary.

Corollary 1 *Let* C *be a class of polynomial-sized DNF formulas. There is an equivalence query algorithm that learns* C *in* $O(\log |C_n|)$ *queries using hypotheses that are polynomial-sized depth 4 formulas.*

Proof: Applying the construction of Lemma 6 directly yields hypotheses that are depth 5 formulas. To reduce the depth to 4, it suffices to note that if $g(f_1, \ldots, f_z)$ is the 1/4-good hypothesis given in the statement of theorem, then $\neg g(\neg f_1, \ldots, \neg f_z)$ is also a 1/4-good hypothesis. □

For the class in the above corollary, Bshouty et al. showed the existence of an algorithm whose hypotheses are polynomial-sized depth 3 formulas, which makes $O(n \log |C_n|)$ queries.

Let S be the class of functions represented by the single-variable formulas $f(x_1, \ldots, x_n) = x_i$. Each function in S is of constant size (because each is an embedding of a one-variable function) and contains exactly one relevant variable. Thus any $I(n)$ attribute-efficient algorithm for this class can make only $O(I(n))$ queries. The size and depth of $g(f_1, \ldots, f_z)$ for $f_1, \ldots, f_z \in S$, and g a formula, are equivalent to the size and depth of g, because the f_i are single variables. Theorem 4 implies that S can be learned $\log n$ attribute-efficiently using hypotheses that are polynomial-size formulas of depth 3. In Theorem 5 we show that for any function $h(n)$ that is $o(\sqrt{n/\log n})$, S cannot be learned $h(n)$ attribute-efficiently using hypotheses that are polynomial-size formulas of depth 2. It follows that the depth of g in Theorem 4 cannot be reduced.

Littlestone's WINNOW algorithm learns S in polynomial-time, $\log n$ attribute-efficiently, using hypotheses that are weighted threshold formulas [15]. Theorem 5 implies that the $\log n$ attribute-efficient performance of WINNOW could not be achieved using hypotheses that are polynomial-size DNF formulas.

Theorem 5 *Let S be the class of functions represented by the single variable formulas $f(x_1, \ldots, x_n) = x_i$. There is no equivalence query algorithm for learning S in $o(\sqrt{n/\log n})$ queries such that all hypotheses are Boolean formulas of size polynomial in n, and of depth at most 2.*

Proof: If a is an assignment, let $w(a)$ be the weight of a, that is, the number of 1's in a. We call a function h a (k_1, k_2)-Threshold if $h(x) = 1$, for all a such that $w(a) \geq k_2$, and $h(x) = 0$ for all a such that $w(a) \leq k_1$.

Let $\mathcal{A}$ be an equivalence query algorithm for learning S, such that each equivalence query is either a DNF formula having at most n^c terms or a CNF formula having at most n^c clauses, for some constant c. Consider a run of algorithm $\mathcal{A}$ on some target function in S. Suppose that each equivalence query of $\mathcal{A}$ is answered by an adversary that returns either a positive counterexample of maximal weight or a negative counterexample of minimal weight.

Let $S_1 = S$, and for $i > 1$ let S_i denote the set of functions in S that are consistent with the first $i-1$ counterexamples received by the algorithm. Let V_i be the set of variables x_j such that $f = x_j$ is in S_i. Without loss of generality, we may assume that the hypothesis proposed by $\mathcal{A}$ in its ith equivalence query consists only of variables in V_i.

Let m be such that $|S_m| < n/2$ and $|S_{m-1}| \geq n/2$.

Let $n_i = |S_i|$ and $k_i = \sqrt{n_i/(2c\log n_i)}$. We show that the hypothesis presented by $\mathcal{A}$ in its ith equivalence query, for $i \leq m-1$, is not a DNF or CNF formula expressing an $(\frac{n_i}{k_i}, n_i - \frac{n_i}{k_i})$-Threshold function over the variables V_i.

Let $i \leq m-1$. Let $h' = T_1' \vee \cdots \vee T_t'$ be a DNF formula that is an $(\frac{n_i}{k_i}, n_i - \frac{n_i}{k_i})$-Threshold over the variables V_i (the case for CNF formulas is symmetric). Remove all negated variables in terms of h'. Then for all terms T_j' of h' containing more than $\frac{n_i}{k_i}$ variables, throw out all but $\frac{n_i}{k_i}$ of the variables. The resulting formula $h = T_1 \vee \cdots \vee T_t$ is a $(\frac{n_i}{k_i}, n_i - \frac{n_i}{k_i})$-Threshold each of whose terms contains exactly $\frac{n_i}{k_i}$ variables. Each term T_l is satisfied by

$$\binom{n_i - (\frac{n_i}{k_i})}{n_i - 2(\frac{n_i}{k_i})}$$

assignments to V_i of weight $n_i - (\frac{n_i}{k_i})$. Since $h = 1$ on all assignments of weight $n_i - (\frac{n_i}{k_i})$ we have

$$\binom{n_i - (\frac{n_i}{k_i})}{n_i - 2(\frac{n_i}{k_i})} t \geq \binom{n_i}{n_i - (\frac{n_i}{k_i})}.$$

Assume that n is large enough so that $n/2 > 2c\log(n/2)$, and thus $k_i > 1$.

Therefore

$$
\begin{aligned}
t &\geq \frac{\binom{n_i}{n_i-(\frac{n_i}{k_i})}}{\binom{n_i-(\frac{n_i}{k_i})}{n_i-2(\frac{n_i}{k_i})}} \\
&= \frac{n_i(n_i-1)\cdots(n_i-(\frac{n_i}{k_i})+1)}{(n_i-(\frac{n_i}{k_i}))(n_i-(\frac{n_i}{k_i})-1)\cdots(n_i-2(\frac{n_i}{k_i})+1)} \\
&\geq \left(1+\frac{1}{k_i-1}\right)^{(\frac{n_i}{k_i})} \\
&\geq \left(\left(1+\frac{1}{k_i}\right)^{k_i}\right)^{\frac{n_i}{k_i^2}} \\
&\geq 2^{\frac{n_i}{k_i^2}} = n_i^{2c} \\
&\geq (\frac{n}{2})^{2c} > n^c.
\end{aligned}
$$

Since $\mathcal{A}$ does not ask equivalence queries using DNF hypotheses with more than n^c terms (or CNF hypotheses with more than n^c clauses), for $i \leq m-1$, none of the hypothesis used by $\mathcal{A}$ in its equivalence queries expresses $(\frac{n_i}{k_i}, n_i - \frac{n_i}{k_i})$-Threshold functions over the variables in V_i. Therefore, for $i \leq m-1$, the counterexample produced by the adversary on the ith equivalence query eliminates from S_i no more than n_i/k_i functions.

For all i,

$$n_i/k_i \leq \sqrt{2cn\log n}.$$

Therefore at most $\sqrt{2cn\log n}$ functions are eliminated from S by each of the first $m-1$ queries. Since a total of at least $n/2$ functions must be eliminated after the first $m-1$ queries, it follows that $m-1 \geq \frac{\sqrt{n}}{2\sqrt{2c\log n}}$. It follows that , $\mathcal{A}$ asks at least $\Omega(\sqrt{n/\log n})$ queries. $\Box$

6 Acknowledgments

Thanks to Ingo Wegener for providing us with a manuscript of his joint paper [18], and for informing us about previous work on splitter construction.

References

[1] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[2] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth, Construction of asymptotically good low-rate error-correcting codes through pseudorandom graphs, *IEEE Transactions on Information Theory*, 38:509-516, 1992.

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

[4] A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. *Journal of Computer and System Sciences*, 50(1):32–40, 1995.

[5] A. Blum. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9:373–386, 1992.

[6] R. Boppana. Amplification of probabilistic boolean formulas. *Advances in Computing Research*, 5(4):27–45, 1989.

[7] N. H. Bshouty, R. Cleve, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 130–139. ACM Press, New York, NY, 1994.

[8] N. H. Bshouty and L. Hellerstein. Attribute efficient learning with queries (Extended Abstract). In *Proc. 9th Annu. ACM Conference on Comput. Learning Theory*, 1996.

[9] J. Friedman. Constructing O(n log n) size monotone formulae for the k-th elementary symmetric polynomial of n boolean variables. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 506–515, 1984.

[10] M. L. Fredman, J. Komlós and E. Szemerédi, Storing a sparse table with $O(1)$ worst case access time, *J. of the ACM*, 31:538-544, 1984.

[11] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math Systems Theory*, 17:13–27, 1984.

[12] J. Hastad, I. Wegener, N. Wurm, and S.-Z. Yi, Optimal depth, very small size circuits for symmetric functions in AC^0. *Information and Computation*, 108:200-211, 1994.

[13] S. Kannan. On the query complexity of learning. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 58–66. ACM Press, New York, NY, 1993.

[14] M. Kleiman and N. Pippenger. An explicit construction of short monotone formulae for the monotone symmetric functions. *Theoretical Computer Science*, 7:325–332, 1978.

[15] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[16] M. Naor, L. Schulman, and A. Srinivasan. Splitters and near-optimal decomposition. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 182–193, ACM Press, New York, NY, 1995.

[17] R. Motwani and P. Raghavan. Randomized Algorithms. Cambridge University Press, 1995.

[18] R. Uehara, K. Tsuchida, and I. Wegener. Optimal attribute-efficient learning of disjunction, parity, and threshold functions. In *Proceedings of 3rd European Conference on Computational Learning Theory, EuroColt '97*, Lecture Notes in Artificial Intelligence, pages 171-184 Springer LNCS 1208, 1997.

List of symbols:

$\mathcal{N}$	natural numbers (calligraphic N)
$\mathcal{A}$	calligraphic A
$\mathcal{T}$	calligraphic T
$\mathcal{E}$	expectation (calligraphic E)
α	alpha
ϕ	phi
μ	mu
τ	tau
γ	gamma
ϵ	epsilon
δ	delta
λ	lambda
Λ	capital lambda
Δ	capital delta
Π	product (capital pi)
$\sum$	summation (capital sigma)

$\wedge$ Boolean AND (wedge)

$\vee$ Boolean OR (vee)

$\neg$ negation

$*$ star

$\cup$ union (cup)

$\hat{f}$ f-hat

$\lceil\,\rceil$ ceiling

$\bar{1}$ 1 bar

$\bar{e}_i$ e sub i bar