Learning in the Presence of Finitely or Infinitely Many Irrelevant Attributes

Avrim Blum*

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
avrim@theory.cs.cmu.edu

Lisa Hellerstein†

Department of EECS
Northwestern University
2145 Sheridan Road
Evanston, IL 60208-3118
hstein@eecs.nwu.edu

Nick Littlestone‡

NEC Research Institute
4 Independence Way
Princeton, NJ 08540
nickl@research.nj.nec.com

*Supported by an NSF Postdoctoral Fellowship. Part of this research was conducted while the author was at MIT, and supported by an NSF Graduate Fellowship.

†Research supported in part by a grant from the Siemens Corporation and NSF Grant 0830-350-F674. Part of this research was conducted while the author was at the Massachusetts Institute of Technology, and the University of California at Santa Cruz.

‡Research was primarily conducted while the author was at Harvard University, and was supported by ONR grant N00014-85-K-0445 and DARPA grant AFOSR-89-0506.

Please send proofs to:

Lisa Hellerstein
Department of EECS
Northwestern University
2145 Sheridan Road
Evanston, IL 60208-3118

Proposed running head: Learning with irrelevant attributes.

List of symbols:

α	alpha
∞	infinity
$\hat{m}$	m-hat
φ	phi
$*$	star
$\tilde{n}$	n-tilde
$\mathcal{A}$	calligraphic A
$\mathcal{B}$	calligraphic B
$\lceil\ \rceil$	ceiling
$\lfloor\ \rfloor$	floor
$\sum$	summation (capital sigma)

Abstract

This paper addresses the problem of learning boolean functions in query and mistake-bound models in the presence of irrelevant attributes. In learning a concept, a learner may observe a great many more attributes than those the concept depends upon, and in some sense the presence of extra, irrelevant attributes does not change the underlying concept being learned. Because of this, we are interested not only in learnability of concept classes, but also in whether the classes can be learned by an algorithm that is *attribute-efficient* in that the dependence of the mistake bound (or number of queries) on the number of *irrelevant* attributes is low.

The results presented here apply to projection and embedding-closed (p.e.c.) concept classes. We show that if a p.e.c. class is learnable attribute-efficiently in the mistake-bound model, then it is learnable in the infinite-attribute mistake bound model as well. We show in addition how to convert any algorithm that learns a p.e.c. class in the mistake-bound model with membership queries into an algorithm that learns the class attribute-efficiently in that model, or even in the infinite attribute version. In the membership query only model, we show that learnability does not always imply attribute-efficient learnability for deterministic algorithms. However, we describe a large class of functions, including the set of monotone functions, for which learnability does imply attribute-efficient learnability in this model.

1 Introduction

This paper addresses the problem of learning boolean functions in the presence of irrelevant attributes. When learning a concept, a learner may observe a great many more attributes than those the concept depends upon. There is some arbitrariness to the selection of the attributes to be observed by the learner, and in some sense the presence of extra, irrelevant attributes does not change the underlying concept being learned. Because of this, we are interested not only in the learnability of concept classes, but also in whether the classes can be learned by an algorithm that is *attribute-efficient.* By this we mean roughly that the dependence of the algorithm's performance (in terms of the number of mistakes or number of queries made) on the number of *irrelevant* attributes is low.

Our work is motivated in part by previous attempts to design polynomial algorithms that learn specific concepts and perform well in the presence of irrelevant attributes [8, 9, 3]. Littlestone [9] describes algorithms for several classes of concepts in the mistake-bound learning model that are particularly efficient in this sense. Blum [3] describes a variant of the standard model, the "Infinite Attribute" model, where examples are presented in such a way that an algorithm may run in polynomial time even in spaces with infinitely many irrelevant attributes, so long as only a small number appear positively in each instance. In this paper, we address the following general questions related to the above results. First, which classes of functions are learnable in the Infinite-Attribute model, where there may be an unbounded number of irrelevant attributes? Second, in both models, which classes of functions are learnable attribute efficiently? Finally, how does the ability of a learner to make queries affect this issue? We present results that partially answer these questions.

1.1 The models

We consider in this paper five learning models — the mistake bound (MB) model, the infinite attribute mistake bound (IMB) model, the standard mistake bound model augmented with membership queries (the MBQ model), the infinite attribute, mistake bound model augmented with membership queries (the IMBQ model), and the model of exact learning with membership queries only (the QU model). We introduce these models here.

Let $V_\infty = \{v_1, v_2, \ldots\}$ be a countably infinite set of variables and $V \subseteq V_\infty$ be a finite or infinite subset of V_∞. An *instance* over V is an assignment of 0 or 1 to each element of V, and we let $\{0,1\}^V$ denote the set of all such assignments. Let V_n be the set $\{v_1, \ldots, v_n\}$. An element A of $\{0,1\}^{V_n}$ can be represented by the Boolean vector $\langle A(v_1), \ldots, A(v_n)\rangle$. A *concept* over V is a boolean function $f : \{0,1\}^V \rightarrow \{0,1\}$, and a *concept class* is a collection of concepts.

In the mistake bound model MB [9], the target function f is over $\{0,1\}^{V_n}$ and learning proceeds in a sequence of trials. In each trial the learner is given an instance $A \in \{0,1\}^{V_n}$, represented as an n-bit tuple. The learner then attempts to predict 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*).

The model IMB [3] is similar but instances are chosen from $\{0,1\}^{V_\infty}$. Because they cannot be represented by finite tuples, they are represented by listing the names of the variables that are set to 1. It is assumed that all instances given to the algorithm are of finite length. Given some finite sequence of trials, we will use m to denote the maximum number of variables of any instance seen. We will use $\hat{m}$ to denote the description length (number of bits) of the longest instance, according to some fixed method of representing lists of variables. Except for the representation of the instances, learning proceeds in model IMB as in model MB.

A *membership oracle* for f takes as input an instance A and returns $f(A)$. Queries to this oracle are called *membership queries*. The MBQ model is like the MB model, but in each trial the algorithm is allowed to make membership queries before it requests an instance to predict, and we count the total number of mistakes and queries. The IMBQ model is defined analogously, but queries are restricted to instances with a finite number of variables set to 1. The time to make a query is proportional to its length; this is n in the MBQ model and the description length of the instance in the IMBQ model.

By the results of [9], we can (and will) assume without loss of generality that algorithms for the MB, IMB, MBQ, and IMBQ models are "conservative". A conservative algorithm only updates the hypothesis it uses for prediction when it makes a mistake or receives the answer to a query. In particular, each time the algorithm is told it predicted correctly, it "forgets" it ever saw the instance (by not changing its state) and requests another instance instead.

In the QU model (cf. Angluin [1]), the algorithm learns the concept through membership queries alone (no trials). The algorithm is required to terminate and output a representation of the function f. We count the total number of queries. We define the QU model only for finite domains. [1]

We say a function f over $\{0,1\}^V$ *depends* on a variable $v \in V$ if there exist assignments A, B in $\{0,1\}^V$ such that $f(A) \neq f(B)$ and $A(w) = B(w)$ for all $w \neq v$ in V. A variable $v \in V$ is a *relevant* variable of f if f depends on v.

Our bounds will depend on n or m, and two additional parameters: some measure of the "complexity" of the target function, denoted by s, and the number r of relevant variables for the target function. We say that an algorithm to learn C is *attribute-efficient* if for some polynomial Q and some constant $\alpha < 1$, the number of mistakes and/or queries it makes (when the target function is defined on n variables, and r of them are relevant) is at most $Q(r,s)n^\alpha$ (or $Q(r,s)m^\alpha$ in IMB and IMBQ models). We say it is *strongly attribute-efficient* if the number of mistakes and/or queries is at most $Q(r,s)\log n$ (or $Q(r,s)\log m$ in the IMB and IMBQ models). We provide these two definitions because although our transformations to attribute efficiency all give logarithmic dependence on n, our transformations *from* attribute efficient algorithms only require the n^α ($\alpha < 1$) dependency. It is an interesting open question whether learnability in the weaker sense implies learnability in the stronger sense in general for the classes that we study in this paper. The reader may wish to compare this with the status of Occam algorithms [4, 6] whose definition has a similar form but the sublinear dependence is on the number of examples.

A learning algorithms is not given r or s as input. In all but the QU model, however, it is

[1] In the straightforward extension of the QU model to the infinite domain, many concept classes cannot be learned at all in finite time. The problem is that after a finite number of queries, the algorithm may not be able to tell whether it has found all the relevant variables in the target concept.

often possible to guess values for these quantities and double or square the guesses as necessary to reduce the problem to the case where these values are known. The difficulty of applying doubling techniques in the QU model comes from the fact that it may not be feasible with an acceptable number of queries to ascertain whether the target concept has been successfully identified or instead the number of guesses needs to be doubled.

We note that if computational complexity is not a concern, in the MB model we can always achieve a mistake-bound depending logarithmically on n (for a given r) by using the halving algorithm (defined in [9]). A variation of this algorithm (see [3]) gives logarithmic dependence on m for the IMB model. The difficulty in these models is in finding polynomial-time algorithms that exhibit such behavior.

In the models MB and MBQ we say a learning algorithm is *polynomial time* if the worst case time spent in computation during each trial is polynomial in n and s. In the IMB and IMBQ models, we allow time polynomial in $\hat{m}$ and s per trial. In the QU model, a learning algorithm is polynomial time if its *total* running time is polynomial in n and s. The class C is *polynomially learnable* in a model if there is a polynomial time algorithm that learns C in a number of mistakes and queries (where applicable) polynomial in n and s for models MB, MBQ and QU, or in m and s for models IMB and IMBQ. Note that an algorithm for learning a class C in any of the above models will necessarily take time linear in n or m in order to set up the queries or read the counterexamples. In designing attribute efficient algorithms, we therefore do not try to make the time of the algorithm depend sublinearly in n or m as we do with the mistakes and queries. In the rest of this paper, where we say "learnable" we implicitly refer to polynomial learnability.

1.2 Informal statement of results

Our results apply to the set of function classes that are *projection-closed* and *embedding-closed*. Informally, a class of functions is projection-closed if it is closed under the operation of taking a function f in the class and fixing the assignments to some of the variables of f (yielding a new function on a smaller domain). A class is embedding-closed if it is closed under the operation of taking a function f in the class, renaming the variables of its domain, and/or adding additional irrelevant variables to its domain. We will see below that projection and embedding-closed (p.e.c.) function classes are "well-behaved" in a number of ways. Many widely studied concept classes (including k-CNF, k-DNF, k-term CNF, k-term DNF, decision lists, and read-once formulas) are either p.e.c., or are contained in some p.e.c. class that is not significantly harder to learn than the original class. An example of a class that is not projection closed is the set of all functions on n variables variables that depend on at most $\log n$ of them. An example of a class that is not embedding-closed is the set of boolean functions that have exactly one satisfying assignment.

In this paper, we prove the following main results. We show that if a p.e.c. class is learnable attribute-efficiently in the MB model, then it is learnable in the IMB model. Moreover, if the class is learnable strongly attribute-efficiently in the MB model, is also learnable strongly attribute-efficiently in the IMB model.

We show in addition that any p.e.c. class learnable in the MBQ or IMBQ models is also learnable strongly attribute-efficiently in both models. In the QU model, we show that learnability does not imply attribute-efficient learnability if we require the learning algorithm to be deterministic. However, we prove that there is a large class of functions, including the set of monotone functions, for which learnability does imply strongly attribute-efficient learnability.

In Section 6 we show that in the MBQ, IMBQ, and MB models, the number of mistakes (+queries) performed by any learning algorithm is $\Omega(r\lfloor\log_2(n/r)\rfloor)$. In the QU model, the number of queries is bounded below by $\log_2\binom{n}{r}$ by a simple information theory argument.

There remain many open questions about the relation of learnability and attribute-efficient learnability in the various models. One class that may be useful to study in resolving some of these questions is the class of boolean functions that compute the parity of a subset of their inputs. (This class is not itself p.e.c. but can be made so by including the constant functions 0 and 1, and functions that compute the negation of parity). Parity *is* learnable in the MB model. However, it is not known whether it is learnable attribute-efficiently in the MB model and it is not known whether it is learnable in the IMB model. We show that parity is learnable attribute-efficiently in the QU model by a randomized algorithm (if exact identification with high probability is allowed), but not by any deterministic algorithm.

2 Additional Definitions

Given two variable sets V and V', an *embedding* of the domain $\{0,1\}^V$ in the domain $\{0,1\}^{V'}$ is an injective mapping $\varphi : V \to V'$. Given $\varphi : V \to V'$ and a concept f over V (a boolean function over $\{0,1\}^V$) there is a natural way to define a corresponding concept f' over V'. For $B \in \{0,1\}^{V'}$, we define $f'(B)$ to be $f(A)$ where $A(v) = B(\varphi(v))$ for all $v \in V$. We denote the function f' by $\varphi(f)$. We will say $\varphi(f)$ is the *embedding of the concept* f in $\{0,1\}^{V'}$ via φ.

A *partial assignment* P to V is a mapping from V to $\{0,1,*\}$. A variable in v is *assigned by* P if $P(v) \neq *$. Given two total or partial assignments P and A such that P assigns values to some subset of the variables in V, and A assigns values to the remainder, we denote by P/A the 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 given a function f over $\{0,1\}^V$ we let f_P denote the function over $\{0,1\}^{V'}$ defined by $f_P(A) = f(P/A)$. There is also a natural interpretation of f_P as a function over the original domain $\{0,1\}^V$, in which conflicts between the argument to f_P and P are resolved in P's favor. We use f_P to denote the function with either domain.

We consider the problem of learning (under various learning models) concept classes C made up of subclasses $C_1, C_2 \ldots$, where the concepts in C_i are boolean functions defined on V_i (the set $\{v_1,\ldots,v_i\}$). With each such class C we associate a corresponding class C_∞. C_∞ contains all functions $\varphi(f)$ such that $f \in C_i$ and φ embeds V_i into V_∞. Note that each function in C_∞ is the embedding of a function defined on a finite number of variables.[2]

When we say we learn a class C in both the MB and the IMB models, we mean that C is

[2]There are functions defined on V_∞ that do not have this property. An example is the function f such that $f(A) = 1$ for all assignments A such that $\lim_{i\to\infty} A(v_i) = 1$, and $f(A) = 0$ otherwise.

learned in the MB models and C_∞ is learned in the IMB model. We define $C_{r,n}$ to be the set of concepts in C_n that depend on at most r relevant variables.

Let s be a "complexity" function associated with concepts. We assume that our functions s have the property that if f_P is a projection of f, then $s(f) \geq s(f_P)$. We also assume that if $f \in C_i$ for some i and φ is an embedding of V_i into V_k (or V_∞), then $s(f) = s(\varphi(f))$.[3]

A concept class C is *embedding closed* if for all $k \leq n$, and for all embeddings φ of $\{X_1, \ldots, X_k\}$ into $\{X_1, \ldots, X_n\}$, if $f \in C_k$, then $\varphi(f) \in C_n$. C is *projection closed* if for all i, and for all partial assignments P to the variables in V_i, $\varphi(f_P) \in C_{|W|}$, where W is the set of variables not assigned by P, and φ is an embedding of W into $\{v_1, \ldots, v_{|W|}\}$.

If a class C is both projection and embedding closed, then it is easily shown that $C_{r,n} = \{\varphi(f_r) \mid f_r \in C_r,\ \varphi : V_r \to V_n\}$.

In our proofs we implicitly use the fact that if f is defined on a finite set of variables V, and P is a partial assignment that leaves all relevant variables of f unassigned, then $f_P = f$. This also holds for any f in a class of the form of C_∞. However, it does not hold for arbitrary functions defined over V_∞. For example, suppose we let $f(A) = 1$ for all assignments A such that $\lim_{i\to\infty} A(v_i) = 1$ and we let $f(A) = 0$ otherwise. Then no variable is relevant, and there are projections P of f (which assign values to an infinite number of the variables) for which $f_P \neq f$.

Finally, we note that our definition of relevant variables only applies to functions over the entire domain $\{0,1\}^V$ for some V. If, for example, the domain of f consists only of the two points $(0,0)$ and $(1,1)$, and $f(0,0) = 0$ and $f(1,1) = 1$, then it is unclear which variables should be considered relevant. We do not consider such incomplete domains in this paper.

3 Transformation of an attribute-efficient MB algorithm

Clearly, learning in the infinite-attribute mistake-bound model is at least as hard as learning in the standard finite-attribute model. In Theorem 1, we show a partial converse. It is an open question whether the full converse holds; that is, whether MB learnability implies IMB learnability in general.

Theorem 1 *If a p.e.c class C is learnable in the MB model by an attribute-efficient algorithm, then C_∞ is learnable in the IMB model.*

The proof of Theorem 1 follows from the next lemma.

Lemma 2 *Suppose there exists a polynomial-time attribute-efficient algorithm $\mathcal{A}$ to learn a p.e.c. class C in the MB model which makes at most $Q(r,s)h(n)$ mistakes on concepts $f \in C_{r,n}$ of size*

[3] This last restriction on the complexity function makes theorems significantly simpler to state. However, it does imply that s cannot be a true "size" function in the sense of description length; for example, for $f_1 = x_1 \vee x_{10}$ and $f_2 = x_{10011} \vee x_{101110110}$ we have $s(f_1) = s(f_2)$ if we consider f_1 and f_2 as two different embeddings into C_∞ of the same disjunction of two variables. For the IMB and IMBQ models, the fact that the variables themselves may have a high description length is taken care of in the term $\hat{m}$ denoting the *length* of the longest instance seen (see Section 1.1).

s, where h is non-decreasing and polynomially computable, and $Q(r,s) > 1$. Then there exists a polynomial-time algorithm $\mathcal{B}$ to learn C_∞ in the IMB model that given $Q = Q(r,s)$ and an integral upper bound $m \geq 1$ on the size of the longest instance to be presented, makes at most $Qh(\tilde{n})$ mistakes on $f \in C_{r,\infty}$ of size s, where $\tilde{n}$ is the least integer such that $\tilde{n} \geq m + mQh(\tilde{n})$.

We will now assume the correctness of Lemma 2, postponing its proof to the end of this section.

Proof of Theorem 1 (given Lemma 2): First, suppose Q and m are given to the IMB algorithm. Let $\alpha < 1$ be a constant such that $h(n) = n^\alpha$ in Lemma 2. Then, solving $\tilde{n} \geq m + mQ\tilde{n}^\alpha$ yields: $\tilde{n} \geq [m/\tilde{n}^\alpha + mQ]^{1/(1-\alpha)}$. The value chosen for $\tilde{n}$ will thus be at most $[m + mQ]^{1/(1-\alpha)}$. So, the number of mistakes made by algorithm $\mathcal{B}$ is at most:

$$Q(r,s)\,[m + mQ(r,s)]^{\alpha/(1-\alpha)}, \tag{1}$$

which is polynomial in m, r, and s since $\alpha < 1$. This proves the theorem, when Q and m are given.

If Q and m are not given, let $\mathcal{B}'$ be an algorithm for the IMB model that guesses the unknown values $Q = Q(r,s)$ and m to be used by algorithm $\mathcal{B}$ of Lemma 2 and doubles them when necessary. It doubles the guess for Q when the mistake bound $Q\tilde{n}^\alpha$ is exceeded, and doubles the guess for m when an instance with more than m attributes is seen. When either estimate is revised, the algorithm restarts $\mathcal{B}$ from scratch. Let m_i be the ith guess for m, and let Q_j be the jth guess for Q (starting at $m_0 = Q_0 = 2$). For a fixed guess m_i, the number of mistakes is at most $\sum_j Q_j[m_i + m_i Q_j]^{\frac{\alpha}{1-\alpha}} = m_i^{\frac{\alpha}{1-\alpha}} \sum_j Q_j(1+Q_j)^{\frac{\alpha}{1-\alpha}} \leq m_i^{\frac{\alpha}{1-\alpha}} \sum_j (2Q_j)^{1+\frac{\alpha}{1-\alpha}} = O(m_i^{\frac{\alpha}{1-\alpha}} Q^{1+\frac{\alpha}{1-\alpha}})$. Summing over all guesses m_i, we get a mistake bound of $O(m^{\frac{\alpha}{1-\alpha}} Q^{1+\frac{\alpha}{1-\alpha}})$, which is polynomial in r, s, and m as desired. □

One can notice from the bound given in the proof above that if algorithm $\mathcal{A}$ is sufficiently attribute-efficient that the dependence on n is at most n^α for $\alpha < 1/2$, then the resulting algorithm $\mathcal{B}'$ for the IMB model will be attribute-efficient as well.

We can also use Lemma 2 to prove the following theorem.

Theorem 3 *If a p.e.c class C is learnable in the MB model by a strongly attribute-efficient algorithm, then C_∞ is also learnable by a strongly attribute-efficient algorithm in the IMB model.*

Proof: Using $h(n) = \log n$ in Lemma 2 gives $(3mQ)^2$ as an upper bound on $\tilde{n}$ since: $m + mQ\log((3mQ)^2) \leq 4mQ\log(3mQ)$, and this is at most $(3mQ)^2$ for $mQ \geq 1$. So, the number of mistakes made by algorithm $\mathcal{B}$ is at most:

$$2Q(r,s)\log(3mQ(r,s)), \tag{2}$$

and so $\mathcal{B}$ is strongly attribute-efficient. This proves the theorem if Q and m are given to the IMB algorithm. To remove this assumption, we double our estimate for Q as in the proof of Theorem 1, but we square the guessed value of m when the previous guess is exceeded (doubling m introduces an extra $\log m$ factor). Let m_i be the ith guess for the value of m, and let Q_j be the jth guess for the value of Q. For the duration of a fixed pair of guesses m_i and Q_j, the maximum number of mistakes is $O(2Q_j \log(3m_iQ_j))$, which for simplicity we may upper bound by $O(Q_j[\log Q_j][\log m_i])$. Thus, the total number of mistakes is at most $O(\sum_i \sum_j Q_j[\log Q_j][\log m_i]) = O(\sum_i Q[\log Q][\log m_i]) = O(Q[\log Q][\log m])$, yielding the desired result. □

In the proofs of Theorems 1 and 3, the IMB algorithm to learn C_∞ starts a new iteration of the MB algorithm each time it receives an example whose size exceeds the current estimate. We note that if the size of the largest instance seen so far keeps doubling (or squaring, in Theorem 3) before each iteration of the IMB algorithm is completed, the IMB algorithm will continue making mistakes forever despite always staying within the mistake bound. If convergence is desired in this case, it can be obtained by interleaving the IMB algorithm with another IMB algorithm that is guaranteed to converge but possibly has worse mistake bounds. The interleaved algorithm uses one of the two procedures to predict until it makes a mistake, and then switches to the other one. Since one of the procedures converges, so does the interleaved algorithm, and the mistake bound of the interleaved algorithm is at most twice the mistake bound of the original (possibly non-converging) procedure.

A sample converging procedure is as follows. It works in phases, starting with phase 0. In the ith phase, the procedure "assumes" that the relevant attributes of the target function are all contained in the set S_i where $S_0 = \{\}$. During phase i, the procedure projects each instance to an assignment over $\{0,1\}^{S_i}$ (i.e., it ignores the assignment to all variables outside of S_i) and runs algorithm $\mathcal{A}$ on the associated Boolean vector. If $\mathcal{A}$ ever makes more than its allotted number of mistakes, the procedure abandons phase i and proceeds to phase $i+1$. The set S_{i+1} is S_i unioned with the set of all variables seen in instances on which a mistakes was made. S_{i+1} must always contain at least one more relevant variable than S_i. Therefore, at some phase S_i will contain all the relevant variables, and the procedure will converge. Note that this procedure does not, however, achieve the mistake bound given in Lemma 2.

Proof of Lemma 2: Let $\mathcal{A}$ be the algorithm for learning C in the MB model making at most $Q(r,s)h(n)$ mistakes on concepts in $C_{r,n}$ of size s. We assume without loss of generality that $\mathcal{A}$ is conservative.

The heart of the conversion consists of a procedure that takes instances presented using the IMB model's representation and converts them into finite tuples of some length n_0 that can be accepted by algorithm $\mathcal{A}$. The fact that our concept class is projection and embedding closed allows us to do this conversion in a straightforward fashion. We describe how n_0 is calculated at the end of the proof. (The $\tilde{n}$ in the statement of the lemma is an upper bound on n_0.)

An IMB-model instance is received in the form of a list of attributes. The conversion procedure maintains a list of all attributes that have been seen in any instance so far, and sequentially gives indices to all of these. If an instance contains new attributes (ones not appearing in any previous instance) the procedure gives these consecutive indices starting with the first unassigned index. Once an attribute has been given an index it keeps it. If there is a need to assign more than a total of n_0 indices, then the conversion procedure halts, reporting failure. Otherwise, the procedure constructs a tuple with n_0 components. The ith component is 1 if an attribute present in the current instance has been given index i and is 0 otherwise (either no attribute has been given the index, or the corresponding attribute is absent from the current instance). The tuple is then given to the MB model algorithm $\mathcal{A}$, its prediction is reported, and the reinforcement ("correct" or "incorrect") is relayed back.

This conversion procedure would be fine except for the following problem: we may unnecessarily bind (give indices to) many variables on instances on which the algorithm predicted correctly. So, if there is a long stretch of instances on which no mistake is made, we could easily exceed our bound on the number of indices. However, $\mathcal{A}$ is conservative. So, if the algorithm predicts correctly in a trial, we may reset the bindings to the state at the beginning of the trial. Thus, we will only permanently give indices to attributes present in instances on which a mistake is made and therefore can use the mistake bound to bound the number of indices we will need. This final procedure is the algorithm $\mathcal{B}$.

Consider a successful run of this conversion on some finite sequence of trials. Let S denote the set of all attributes to which indices were given, and let $\varphi : S \to \{1, \ldots, n_0\}$ give the index associated with each of these attributes by the algorithm. If P is the partial assignment that sets the variables outside of S to zero, then all instances seen by algorithm $\mathcal{A}$ will be consistent with the function $\varphi(f_P)$. By hypothesis, this concept is learnable by $\mathcal{A}$ with a mistake bound of at most $Q(r, s)h(n_0)$.

The required minimum value of n_0 is the total number of distinct attributes present in instances at which mistakes are made plus m additional for temporary bindings of instances on which the algorithm predicts correctly. The number of mistakes is at most $\lfloor Q(r, s)h(n_0)\rfloor$. Thus, so long as $n_0 \geq m + m\lfloor Q(r, s)h(n_0)\rfloor$, the conversion algorithm never runs out of variables to bind and algorithm $\mathcal{B}$ succeeds. The algorithm can find the least such integer n_0 by a linear search if necessary (e.g., if h is not well-understood). Because h is non-decreasing, any value $\tilde{n}$ satisfying $\tilde{n} \geq m + mQh(\tilde{n})$ will be at least as large as n_0, and thus suffices for the mistake bound given in the lemma. □

4 Learning Attribute Efficiently in the MBQ and IMBQ models

It follows by applying a technique from [2] that learnability in the MBQ model implies strongly attribute-efficient learnability in the MBQ and IMBQ models. The technique is based on the following lemma.

Lemma 4 ([2]) *Let $f : \{0, 1\}^V \to \{0, 1\}$. Suppose A and B are assignments such that $f(A) = 1$ and $f(B) = 0$. Let S be the set of variables in V on which the assignments A and B differ, i.e. $S = \{v \in V | A(v) \neq B(v)\}$. Then S contains a relevant variable of f. Moreover, such a variable can be found in at most $\lceil \log |S| \rceil$ queries to a membership oracle for f.*

Proof: Let assignments $A_0, A_1, \ldots, A_{|S|}$ form a "path" from A to B in that $A_0 = A$, $A_{|S|} = B$, and each assignment in the path differs in only one bit from the preceding assignment (treating assignments as bit vectors). Since $f(A) \neq f(B)$, using binary search on this path we can find an index i such that $f(A_i) \neq f(A_{i+1})$, in only $\lceil \log |S| \rceil$ queries. Since A_i and A_{i+1} differ in only one bit, the variable corresponding to that bit must be relevant to f. □

Theorem 5 *Suppose there exists a polynomial algorithm to learn a p.e.c. class C in the MBQ model that makes at most $Q(n,s)$ mistakes+queries on concepts $f \in C_n$ of size s (where Q is non-decreasing). Then there exists a polynomial algorithm to learn C in the MBQ model that makes at most $2(r+1)Q(r,s)+r(\lceil \log n \rceil + 1)$ mistakes + queries on concepts $f \in C_{r,n}$ of size s.*

Because the proof of this theorem mimics a proof in [2], we provide only a sketch here.

Proof sketch: Let ALG_n be an algorithm for learning C_n in the MBQ model making at most $Q(n,s)$ mistakes+queries on concepts in $C_{r,n}$ of size s. Let V denote a set of n variables on which this algorithm is being run. We assume that ALG_n is conservative. The new algorithm keeps a set S of variables, initially empty, that it "knows" to be relevant. The algorithm chooses an arbitrary partial assignment P setting all the variables in $V-S$, and leaving the variables in S unassigned. It then tries to learn f_P by simulating $ALG_{n'}$, for $n' = |S|$.

If $ALG_{n'}$ makes a membership query on an assignment A to S, the algorithm queries P/A. It then returns the value $f(P/A) = f_P(A)$ to $ALG_{n'}$ and continues simulating.

If $ALG_{n'}$ requests an instance to predict, the algorithm does the same, and receives some instance A. It then gives to $ALG_{n'}$ the restriction of A to the domain S, and has $ALG_{n'}$ make a prediction b on that instance. The algorithm gives b as its prediction for the value of f on A. If the algorithm is told "correct", it tells $ALG_{n'}$ "correct" and continues the simulation. If it is told "incorrect", then the algorithm queries the membership oracle on P/A. If $f(P/A) \neq b$ the algorithm tells $ALG_{n'}$ "incorrect" and proceeds with the simulation. However, if $f(P/A) = b$ this implies that $f(P/A) \neq f(A)$, so the algorithm can use the procedure described in Lemma 4 to find a new relevant variable v in $V-S$. The algorithm now restarts the simulation with $S \leftarrow S \cup \{v\}$.

Since $ALG_{n'}$ is conservative, we need not worry about the instances on which it predicted correctly in the above simulation. It is straightforward now to verify the mistake plus query bound given in the theorem statement. □

A similar proof yields the following theorem, which improves a result of Blum [3].

Theorem 6 *Suppose there exists a polynomial algorithm to learn a p.e.c. class C in the MBQ model that makes at most $Q(n,s)$ mistakes+queries on concepts $f \in C_n$ of size s, (where Q is non-decreasing). Then there exists a polynomial algorithm to learn C_∞ in the IMBQ model that makes at most $2(r+1)Q(r,s)+r(\lceil \log m \rceil + 1)$ mistakes+queries on concepts $f \in C_{r,\infty}$ of size s.*

Algorithms for the IMBQ model can be used to learn concepts in the MBQ model, and the resulting mistake bound is the same except that n replaces m. It therefore follows from Theorem 6 that any p.e.c. class that is learnable in either the MBQ or IMBQ models is also learnable strongly attribute efficiently in both models.

5 Learning Attribute Efficiently with Membership Queries Only

A *constrained instance oracle* [7] for a function f takes as input a partial assignment P to the variables of f, and a constant value b. If f_P is equal to the constant function b, then the constrained instance oracle returns YES, otherwise it returns NO and an assignment D such that $f_P(D) \neq b$.

Theorem 7 *Suppose there exists a polynomial algorithm to learn a p.e.c class C in the QU model that makes at most $Q(n,s)$ queries on concepts $f \in C_n$ of size s. Suppose further that for all $f \in C_{r,n}$ of size s, a constrained instance oracle for f can be simulated in poly-time using a membership oracle for f with at most $T(n,r,s)$ queries. Then there exists a polynomial algorithm to learn C in the QU model that makes at most $(r+1)(T(n,r,s)+1)Q(r,s)+r\lceil\log n\rceil$ queries on concepts $f \in C_{r,n}$ of size s.*

Proof: Let ALG be a polynomial algorithm for learning C that satisfies the conditions of the theorem, and $ALG_{n'}$ be the particular version of the algorithm for learning $C_{n'}$. Let V denote the set of n variables on which this algorithm is being run.

The following algorithm learns C efficiently with at most $(r+1)(T(n,r,s)+1)Q(r,s)+r\lceil\log n\rceil$ queries.

1. *$S := \emptyset$, $n' := 0$. During the execution of the algorithm, S will always be a subset of V and n' will equal the size of S.*
2. *Simulate the execution of $ALG_{n'}$ on set S until $ALG_{n'}$ terminates or asks a membership query. If $ALG_{n'}$ terminates with function $f_{n'}$, return $f_{n'}$. Otherwise, $ALG_{n'}$ asks a membership query on some assignment A to S. Let D be an arbitrary extension of A to all of the variables of V.*
3. *Query the membership oracle to find $b = f(D)$. Simulate the constrained instance oracle on input A (thought of as a partial assignment to V) and b. If the answer is YES then return b to $ALG_{n'}$ and go back to 2. Otherwise, let D' be the assignment returned.*
4. *We have found assignments D and D' such that $f(D) \neq f(D')$, but D and D' agree on the variables in S. Use these assignments and membership queries as described in Lemma 4 to discover a relevant variable v in $V - S$. Add v to S, increase n' by 1, and go to step 2 to begin a new simulation.*

First we show that the algorithm just described will continue until S contains all the relevant variables of f. Suppose there is some relevant variable v in $V - S$ at the start of step 2. Let E and E' be assignments to V that differ only on their assignments to v, such that $f(E) \neq f(E')$. Such assignments must exist since v is relevant. Let P and P' be partial assignments setting all variables in $V - S$ according to E or E' respectively, and leaving the variables in S unassigned. Clearly $f_P \neq f_{P'}$ because $f_P(E_S) = f(E) \neq f(E') = f_{P'}(E_S)$, where E_S is the restriction of E to S. The functions f_P and $f_{P'}$ are both in $C_{n'}$. Furthermore, they have agreed for all membership queries to which the current simulation of $ALG_{n'}$ has received an answer. (This is guaranteed by the test made in step 3 using the simulation of the constrained instance oracle). Since $ALG_{n'}$ cannot distinguish between f_P and $f_{P'}$, it must make a membership query.

So, eventually S will contain exactly the relevant variables of f. At this point, let P be an arbitrary partial assignment setting just the variables in $V - S$. Clearly $f_P = f$. During the simulation of $ALG_{n'}$, in step 3 the (simulated) constrained instance oracle will always return YES, because $V - S$ does not contain any relevant variables of f. Responses to the queries of $ALG_{n'}$ in step 3 will all be consistent with f_P. Thus the algorithm will terminate with the correct result. The bounds on the algorithm are easily verified. $\Box$

Corollary 8 *If C is a p.e.c. class of monotone boolean functions that can be polynomially learned in model QU, then C can be learned strongly attribute-efficiently by a polynomial algorithm in QU.*

Proof: A constrained instance oracle for a class of monotone boolean functions can be simulated as follows. If the inputs are P and b, we evaluate f_P on the assignment setting to $1-b$ all the variables not set by P. If the answer is b, then we return YES, else we return the tested assignment. This works since the class is monotone. We then apply Theorem 7. □

In the following corollary, we prove that if there is an efficient algorithm for learning a p.e.c. class C, the question of whether it is possible to learn C (strongly) attribute efficiently in the QU model depends entirely on how many queries it takes for such an algorithm to learn the constant formulas $f \equiv 0$ and $f \equiv 1$. The corollary depends on the fact that a QU algorithm is *not* told the number of relevant variables of the target function.

Corollary 9 *Let C be a p.e.c. class of functions. C can be learned attribute-efficiently in the QU model by a polynomial algorithm iff there is a polynomial algorithm that learns C in the QU model and makes at most $Q'(s)n^\alpha$ $(\alpha < 1)$ queries on any $f \in C_{0,n}$ (i.e. a constant function) of size s, for some polynomial Q'. Similarly, C can be learned strongly attribute-efficiently in QU by a polynomial algorithm iff there is a polynomial algorithm that learns C in the QU model and for some polynomial Q' makes at most $Q'(s)\log n$ queries on any $f \in C_{0,n}$ of size s.*

Proof: We prove the first statement of the corollary. The second is proved similarly.

The forward direction of the corollary follows directly from the definition of attribute-efficient. To prove the converse, let ALG be an algorithm that learns C efficiently in the QU model, and makes $Q'(s)n^\alpha$ queries on any target concept in $C_{0,n}$. We show how to use ALG to efficiently simulate a constrained instance oracle for f using the membership oracle for f. (We assume C does not just contain a single function because otherwise the simulation is trivial.) Suppose P and b are the inputs to the constrained instance oracle. Run ALG on f_P using the membership oracle for f to simulate the membership oracle for f_P, until ALG makes $Q'(s)n^\alpha$ queries, or until it terminates, whichever comes first. If the answers to the membership queries are all the same, then they are consistent with the constant function $f_P(A)$ where A is the first assignment queried. Since $f_P(A)$ lies in C (because C is projection closed), by the assumption of the corollary ALG correctly identifies f_P as this constant function and we can answer the constrained instance oracle. On the other hand, if two assignments A, A' are queried such that $f_P(A) \neq f_P(A')$, again we have an answer to the constrained instance oracle.

The existence of an attribute efficient algorithm for learning C then follows directly from Theorem 7. □

There are p.e.c. classes that *can* be learned by a polynomial-time algorithm in the QU model, but that *cannot* be learned attribute-efficiently in the QU model by deterministic algorithms. For example, consider the class C of parity functions. A parity function is a boolean function that takes a fixed subset of its inputs, and computes the parity (sum mod 2) of the assignments to those inputs. (if the subset is empty, the concept is identically 0). For example, $f(v_1, v_2, v_3, v_4) = v_2 + v_4 \pmod 2$ is a parity function.

C can be learned in polynomial time with n queries by treating both the target function and the assignments as vectors of length n in a binary vector space, where vectors are added componentwise modulo 2 (i.e. an n-dimensional vector space over GF[2]). Assignments correspond to vectors in the natural way, and parity functions correspond to a characteristic vector indicating the subset of variables over which parity is taken (e.g. $[0,1,0,1]$ for the parity function above). The value of a target parity function on an assignment is the inner product of their corresponding vectors.

To learn any target function $f \in C$ (including the constant 0 function, which is in $C_{0,n}$) n linearly independent queries (assignments) are both necessary and sufficient. That is, even if f is identically 0, there will be at least two consistent functions in C if fewer than n queries are made. Therefore C cannot be learned attribute-efficiently by a deterministic algorithm. (Note that this argument again relies on the fact that the algorithm does not know the number of relevant variables.)

However, there is a *randomized* algorithm in the QU model that is in fact strongly attribute-efficient and learns C with high probability. This follows from the fact that a constrained instance oracle for C with probability $1/2$ of success can be simulated as follows. Given partial assignment P and $b \in \{0,1\}$, evaluate f_P on a random assignment A by querying the membership oracle for f on P/A. If $f_P(A) = b$, then return the answer YES, otherwise return NO and A. If f_P is a constant function, then the simulation clearly returns a correct answer. If f_P is not constant, then f_P is equal to the parity (or its negation) of a nonempty subset of its variables, and so the probability that the membership query returns a value other than b is $1/2$. This procedure can be repeated $2\log n$ times to attain failure probability at most $1/n^2$ in each simulation.

6 Lower Bounds

In this section we prove lower bounds on learning embedding closed classes in the various learning models discussed in the paper. Our lower bounds are based on a lower bound on the *Vapnik-Chervonenkis dimension* (see e.g. [5]) of embedding closed concept classes. The Vapnik-Chervonenkis dimension of a class C, $VCdim(C)$, is the size of the largest set of instances S *shattered* by C. A set S is shattered by a class C if any labeling of the instances in S is consistent with some concept in C (i.e., there are $2^{|S|}$ different ways to label the instances of S using concepts in C).

Lemma 10 *Let C be any embedding-closed concept class containing at least one concept that depends on exactly r relevant variables. Then $VCdim(C_{r,n}) \geq r\lfloor \log_2(n/r) \rfloor$.*

Proof: We construct a set of points shattered by $C_{r,n}$. Let $f \in C_{r,n}$ be such that f has exactly r relevant variables. Let $k = \lfloor \log_2(n/r) \rfloor$ and let $m = 2^k$. Without loss of generality, assume that the relevant variables of f are $\{v_1, \ldots, v_r\}$. For $j = 1, \ldots, r$, let $(x_{j1}, \ldots, x_{jn})$ be some assignment to $v_1, \ldots, v_n$ such that the value of f will change if v_j is complemented (i.e., a witness to the relevance of v_j).

We will construct a matrix composed of subblocks defined as follows: Let X_{ji} denote a block of k rows and m columns, each entry of which is x_{ji} (thus all of the entries of such a block match each other). Let B denote a block of k rows and m columns such that all entries are 0's and 1's and all columns are distinct (since $m = 2^k$, all possible columns of 0's and 1's will be included). Let M be a matrix with kr rows and mr columns formed from an r by r matrix of these blocks. The block in the jth row and ith column of this block array is X_{ji} if $i \neq j$ and B if $i = j$. Note that $mr \leq n$. We form a set of kr elements of $\{0,1\}^{mr}$ by taking the rows of M. We then extend these with arbitrary suffixes if necessary to form points of $\{0,1\}^n$. We claim that this set of points is shattered by the concept class. To create a concept with any desired set of values on these points, we embed the variables on which f depends into the full set of n variables appropriately. We keep the variables in the same order, letting the v_i fall somewhere in the ith group of m variables out of the collection of n variables. Note that in the jth group of k points, the value of the embedded function will only depend on where in the jth group v_j falls (and not on where in their own groups the other variables fall), because of the constancy of each of the off-diagonal blocks. We choose v_j so that the embedded function will have the desired set of values at each point in this jth group of k points. This can be done due to the nature of B and because $(x_{j1}, \ldots, x_{jn})$ is a witness to the relevance of v_j. By doing this for each of the variables we create the desired embedded function, demonstrating that our class shatters a set of kr points, as desired. □

Theorem 11 *Let C be an embedding closed class. Any algorithm for learning C in the MBQ model makes at least $\Omega(r\lfloor\log_2(n/r)\rfloor)$ mistakes+queries on some concept $f \in C_{r,n}$ for all $r > 0$ such that $C_{r,n} - C_{r-1,n}$ is not empty. Similarly, any algorithm for learning C_∞ in the IMBQ model makes at least $\Omega(r\lfloor\log_2(m/r)\rfloor)$ mistakes+queries on concepts $f \in C_{r,\infty}$ for all $r > 0$ such that $C_{r,\infty} - C_{r-1,\infty}$ is not empty.*

Proof: Follows from Lemma 10 and the fact that in the MBQ model, the number of mistakes+queries made in learning a class D is $\Omega(VCdim(D))$ [10]. □

For the MB and IMB models, we can improve the constants and get the following.

Theorem 12 *Let C be an embedding closed class. Any algorithm for learning C in the MB model makes at least $r\lfloor\log_2(n/r)\rfloor$ mistakes on some concept $f \in C_{r,n}$ for all $r > 0$ such that $C_{r,n} - C_{r-1,n}$ is not empty. Similarly, any algorithm for learning C_∞ in the IMB model makes at least $r\lfloor\log_2(m/r)\rfloor$ mistakes on some concept $f \in C_{r,\infty}$ for all $r > 0$ such that $C_{r,\infty} - C_{r-1,\infty}$ is not empty.*

Proof: Consider some non-empty $C_{r,n}$. Since $C_{r,n}$ has at least one member, by Lemma 10 its VC dimension is at least $r\lfloor\log_2(n/r)\rfloor$. By [9], the mistake bound for learning a class is bounded below by its VC dimension. Thus, even if r is given to the learning algorithm, any algorithm for learning C in the MB model must make at least $r\lfloor\log_2(n/r)\rfloor$ mistakes on some concept $f \in C_{r,n}$ The analogous bound for the IMB model follows trivially □

By a simple information theory argument, any QU algorithm for learning a class D makes at least $\log|D|$ queries. Note that if $C_{r,n}$ is not empty and C is embedding closed, then $\log|C| \geq \log\binom{n}{r}$. Therefore, any QU algorithm for learning an embedding closed class C must make at least $\log\binom{n}{r}$ queries on some concept $f \in C_{r,n}$, for all r such that $C_{r,n}$ is not empty.

References

[1] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1987.

[2] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. Technical report, University of California, Report No. UCB/CSD 89/528, 1989. To appear in JACM.

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

[4] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, April 1987.

[5] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[6] Raymond Board and Leonard Pitt. On the necessity of Occam algorithms. *Theoretical Computer Science*, 100:157–184, 1992.

[7] T. Hancock. Identifying μ-decision trees and μ-formulas with constrained instance queries. Manuscript, Harvard University, 1989.

[8] David Haussler. Space efficient learning algorithms. Technical Report UCSC-CRL-88-2, University of California Santa Cruz, September 1985. (revised March 1988).

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

[10] Wolfgang Maass and Gyorgy Turan. On the complexity of learning from counterexamples and membership queries. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 203–217, 1990.