Computational sample complexity and attribute-efficient learning

Rocco A. Servedio*

October 1, 1998

Abstract

Two fundamental measures of the efficiency of a learning algorithm are its running time and the number of examples it requires (its sample complexity). The importance of polynomial time has long been acknowledged in learning theory, while recent work on attribute-efficiency has focused attention on algorithms which can learn from few examples. In this paper we demonstrate that even for simple concept classes, an inherent tradeoff can exist between running time and sample complexity. In our first construction, we present a concept class of 1-decision lists and prove that while a computationally unbounded learner can learn the class from $O(1)$ examples, under a standard cryptographic assumption any polynomial-time learner requires almost $\Theta(n)$ examples. Using a different construction, we present a concept class of k-decision lists which exhibits a similar but stronger gap in sample complexity. These results strengthen the results of Decatur, Goldreich and Ron [9] on distribution-free computational sample complexity and come within a logarithmic factor of the largest possible gap for concept classes of k-decision lists. Finally, we construct a concept class of decision lists which can be learned attribute-efficiently and can be learned in polynomial time but cannot be learned attribute-efficiently in polynomial time. This is the first result which shows that attribute-efficient learning can be computationally hard. The main tools we use are one-way permutations, error-correcting codes and pseudorandom generators.

*ESL 102, Division of Engineering and Applied Sciences, Harvard University, Cambridge MA 02138. Phone: (617) 495-3311. Fax: (617) 495-9837. rocco@deas.harvard.edu

1 Introduction

A broad research goal in computational learning theory is to discover fast (i.e. polynomial-time) learning algorithms for various concept classes. Another broal goal is to discover algorithms which can learn from few examples. This paper studies how these two goals can sometimes be mutually exclusive.

In Valiant's Probably Approximately Correct (PAC) model of concept learning [28], the *sample complexity* of a concept class C is the minimum number of labelled examples which any successful learning algorithm for C must require. Information-theoretic lower bounds on sample complexity were given by Ehrenfeucht et. al. in [10], where it was shown that any algorithm which learns a concept class of Vapnik-Chervonenkis dimension d must use $\Omega(d/\epsilon)$ examples. Similar bounds were subsequently established in [18] for a generalization of the PAC model to learning probabilistic concepts. However, these results do not address the question of how many examples a *polynomial-time* learning algorithm must require. (Of course, since drawing an example takes unit time, a polynomial-time learning algorithm can require at most polynomially many examples.)

The first indication that polynomial-time learning might be computationally hard for the unrestricted class of polynomial-size boolean circuits was given by Valiant in his original paper [28]. Kearns and Valiant [19] subsequently established the existence of concept classes which are hard for any polynomial-time learner but are learnable from polynomially many examples by a computationally unbounded algorithm; their results were refined and extended by Kharitonov [16]. Decatur, Goldreich and Ron [9] were the first to study concept classes in which polynomial-time learning is doable but requires more examples than learning using a computationally unbounded algorithm. In the standard PAC model they showed that if one-way functions exist, then given any polynomial $p(n) \geq n$, a concept class C of polynomial-size circuits exists for which any polynomial-time learning algorithm requires $\Theta(p(n)/\epsilon)$ examples, whereas a computationally unbounded learning algorithm for C requires $O(n/\epsilon)$ examples.

The first contribution of this paper is to strengthen the results of Decatur et. al. by establishing stronger gaps of this sort and showing that they can hold even for concept classes whose concepts are extremely simple; we do this via two constructions. Our first construction yields a concept class whose concepts are 1-decision lists and which has the following property: a computationally unbounded learner can learn the class from $O(1/\epsilon)$ examples, but under a standard cryptographic assumption any polynomial-time learner requires almost $\Theta(n/\epsilon)$ examples. This construction uses error-correcting codes and requires only very basic cryptography (the notion of a one-way function). Our second construction makes more extensive use of cryptographic machinery to prove the following result: for any $k \geq 1$ there is a concept class of k-decision lists which a computationally unbounded algorithm can learn from $O(1/\epsilon)$ examples, but under a widely held cryptographic assumption a polynomial-time learner requires $\Theta(n^k/\epsilon)$ examples. This is within a logarithmic factor of the largest possible gap for concept classes of k-decision lists.

Our last main result concerns attribute-efficiency. Loosely speaking, a concept class C is said to be attribute-efficiently learnable if there is a learning algorithm for C which requires only $\mathrm{poly}(\mathrm{size}(c), \log n)/\epsilon$ examples to learn any concept $c \in C$ over n variables (we give a precise definition in Section 5). Attribute-efficient learning algorithms are particularly useful when the target concept depends on few variables but n, the total number of variables, is large. Results of Haussler [15] and Littlestone [21] yield attribute-efficient learning algorithms for k-CNF and k-DNF formulae; more recent results on attribute-efficiency can be found in [4, 7, 27]. Blum [2] and Valiant [29] have each posed the question of whether there exists a polynomial-time attribute-efficient learning algorithm for the concept class of 1-decision lists of length k. Such an algorithm would require $\mathrm{poly}(k, \log n)/\epsilon$ examples and would be a useful tool in machine learning.

We take a step toward answering Blum and Valiant's question by providing the first proof that attribute-efficient learning can be computationally hard. We do this by exhibiting a concept class of decision lists which can be learned in polynomial time and can be learned by a computationally unbounded attribute-efficient learning algorithm but cannot (under a plausible cryptographic assumption) be learned in polynomial time by any attribute-efficient learning algorithm.

A common paradigm for concepts and examples is used throughout this paper. In each of the concept classes which we consider, each concept is associated with a secret key; it is easy to exactly identify the target concept if this key is known. Also, in each of our constructions examples come in two types, which we call *useful* and *useless*. Useful examples each contain an encrypted version of the secret key as well as a small amount of unencrypted information about the target concept. Useless examples all have label 0 and contain no information about the target concept.

Our constructions are based on the following simple idea: a computationally unbounded learning algorithm can decrypt the secret key and hence can learn the target concept from a single useful example. Consequently, such a learning algorithm requires few examples. On the other hand, a polynomial-time learner cannot decrypt the secret key; instead, it can only use the small amount of unencrypted information in each useful example. Hence a polynomial-time learner will need many useful examples in order to acquire a significant amount of information about the target concept.

The remainder of the paper is structured as follows. Section 2 contains preliminary definitions which we use throughout the paper. In Section 3 we exhibit a concept class of 1-decision lists which has a substantial gap between its information-theoretic and computational sample complexities. Section 4 contains analogous results (obtained using a different construction) for a concept class of k-decision lists. In Section 5 we show that attribute-efficient learning of polynomial-time learnable concept classes can be computationally hard. Section 6 concludes with some open problems.

2 Preliminaries

In the boolean PAC learning model, a concept c is a boolean function and a concept class C is a collection of boolean functions. The learner has access to an example oracle $EX(c, \mathcal{D}_n)$ which, on each call, takes one time step and outputs a labelled boolean example $\langle x, c(x)\rangle$ where x is drawn from the distribution $\mathcal{D}_n$ over $\{0,1\}^n$. An algorithm L is said to be a *PAC learning algorithm for concept class* C if the following condition holds: for every distribution $\mathcal{D}_n$, for every $c \in C$ and for every $0 < \epsilon, \delta < 1$, if L is given access to $EX(c, \mathcal{D}_n)$ then with probability at least $1 - \delta$, algorithm L outputs a hypothesis h such that the probability that $h(x) \neq c(x)$ is less than ϵ for x drawn according to $\mathcal{D}_n$. See [20] for a thorough discussion of PAC learning.

The following definitions are from [9]: The *distribution free information theoretic sample complexity* of a concept class C, denoted $\mathcal{ITSC}(C; n, \epsilon)$, is the minimum sample size (as a function of n and ϵ) needed for PAC learning the class C with accuracy ϵ and confidence $\delta = 9/10$, where no computational limitations exist on the learning algorithms which may be used. The *distribution free computational sample complexity* of a concept class C, denoted $\mathcal{CSC}(C; n, \epsilon)$, is the minimum sample size (as a function of n and ϵ) needed for PAC learning the class C with accuracy ϵ and confidence $\delta = 9/10$, where the learning algorithm must operate in polynomial (in n and $1/\epsilon$) time.

A *k-decision list of length ℓ* over the boolean variables $x_1, \ldots, x_n$ is a boolean function L which is represented by a list of ℓ pairs $(m_1, b_1), (m_2, b_2), \ldots, (m_\ell, b_\ell)$, where each m_i is a conjunction of at most k literals over $x_1, \ldots, x_n$ and each b_i is either 0 or 1. Given any $x \in \{0,1\}^n$, the value of $L(x)$ is b_i if i is the smallest index such that m_i is satisfied; if no m_i is satisfied then $L(x) = 0$.

We write $x \circ y$ to denote the concatenation of binary strings x, y and $|x|$ to denote the length of x. A function $f : \{0,1\}^* \to \{0,1\}^*$ is *length-preserving* if $|f(x)| = |x|$ for all x.

A *one-way function* is a function f which can be evaluated in polynomial time but cannot be inverted on a non-negligible (i.e. inverse polynomial) fraction of its inputs by any probabilistic polynomial-time algorithm. The discrete log, the RSA function, and squaring quadratic residues modulo Blum integers are all length-preserving permutations which are widely believed to be one-way.

3 A construction using error-correcting codes

In this section we prove the following:

Theorem 1 *Let $0 < \tau < 1$ be any constant. If length-preserving one-way permutations exist, then there is a concept class C_τ which has $\mathcal{ITSC}(C_\tau; n, \epsilon) = O(1/\epsilon)$ and $\Omega(n^{1-\tau}/\epsilon) = \mathcal{CSC}(C_\tau; n, \epsilon) = O(n/\epsilon)$, where each concept in C_τ over $\{0,1\}^n$ is a 1-decision list.*

3.1 Error-correcting codes

We need some basic terminology from the theory of error-correcting codes. As in [25, 26] we say that a *binary code of block length ℓ and rate r_ℓ* is a code in which codewords are ℓ bits long, where $r_\ell \ell$ positions are "message bits" that can be filled with any combination of 0's and 1's and the remaining $(1 - r_\ell)\ell$ positions have their contents determined by the message bits. Let $A_\ell : \{0,1\}^{r_\ell \ell} \to \{0,1\}^\ell$ be a binary code of block length ℓ and rate r_ℓ; for $x \in \{0,1\}^{r_\ell \ell}$, the j-th bit of the ℓ-bit string $A_\ell(x)$ is denoted by $A_\ell(x)_j$.

We say that the code A_ℓ has *minimum relative distance δ_ℓ* if any pair of distinct codewords $\{A_\ell(x), A_\ell(y)\}$ has Hamming distance at least $\delta_\ell \ell$. For $\alpha_\ell < \delta_\ell/2$, we say that an algorithm D is an *α_ℓ-decoding algorithm for A_ℓ* if, when D is given a string $z \in \{0,1\}^\ell$ which has Hamming distance at most $\alpha_\ell \ell$ from some codeword $A_\ell(x)$, the algorithm D outputs x.

The papers [25, 26] each contain versions of the following important theorem:

Theorem 2 [Sipser, Spielman] *There exists a polynomial-time-constructible family $\{A_i\}_{i=1}^\infty$ of binary error-correcting codes which has the following properties:*

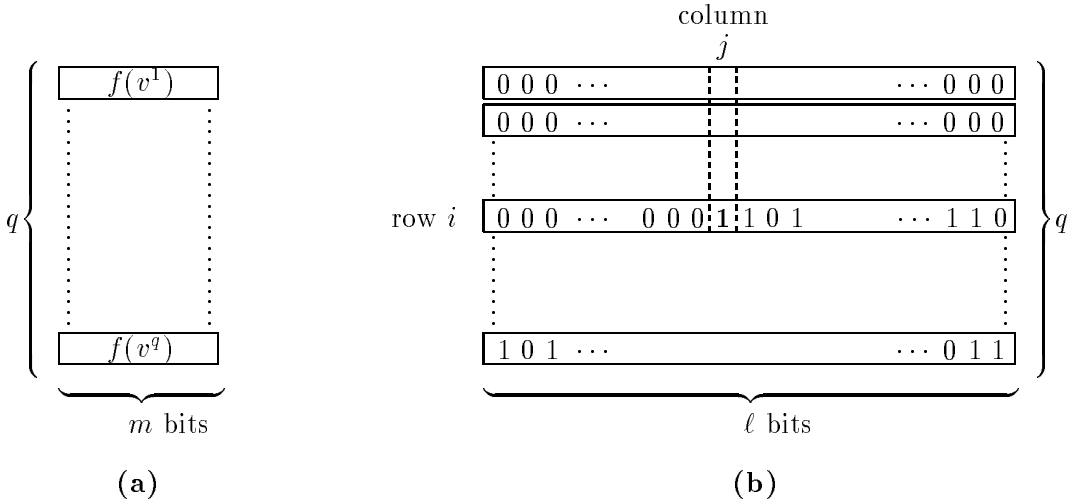
- $\lim_{i\to\infty} r_i > 0$, $\lim_{i\to\infty} \delta_i > 0$ *and* $\lim_{i\to\infty} \alpha_i > 0$,
- *For each ℓ, there is an α_ℓ-decoding algorithm for A_ℓ which runs in time poly(ℓ).*

Recall that in the PAC framework, a learning algorithm succeeds if it can construct a hypothesis which ϵ-approximates the target concept. In the construction which we use to prove Theorem 1, such a hypothesis will yield a string z which is close to a codeword $A_\ell(x)$. By the polynomial-time decoding algorithm of Theorem 2, the ability to find an accurate hypothesis in polynomial time would thus imply the ability to find x in polynomial time. However, we will show that this is impossible (under a cryptographic assumption) if few examples have been seen.

3.2 The concept class C_τ

Before giving a formal description of the concept class C_τ, we mention that in this concept class the secret key for each concept is composed of many small subkeys, each of which is encrypted separately. The reason is that each useful example will contain a small amount of unencrypted information about exactly one of the subkeys. Hence, unless many useful examples have been seen, there will exist subkeys about which no unencrypted information has been revealed.

Figure 1: A useful example $\langle x, A_\ell(v^i)_j \rangle$. Part **(a)** depicts the mq-bit prefix of x; since x is useful this must be $f(v^1) \circ \cdots \circ f(v^q)$. Part **(b)** depicts the $q\ell$-bit suffix $x_{mq+1} \ldots x_n$, where the bit $x_{mq+(r-1)\ell+c}$ is in row r and column c for $1 \leq r \leq q$, $1 \leq c \leq \ell$. As shown in **(b)**, the values of i and j are determined by the location of the first 1 in the $q\ell$-bit suffix.



Now we describe the concept class C_τ. Let f be a fixed length-preserving one-way permutation and let $\{A_i\}_{i=1}^{\infty}$ be a fixed family of error-correcting codes with the properties stated in Theorem 2. For any $m \geq 1$, define the set $D_m \subseteq \{0,1\}^m$ to be $D_m = f^{-1}(\{0,1\}^m)$, and let $q = \lceil m^{\frac{1-\tau}{\tau}} \rceil$. The set $(D_m)^q$ is the set of secret keys; each secret key $v = (v^1, \ldots, v^q) \in (D_m)^q$ is composed of q subkeys each of which is m bits long. The class C_τ has a concept c_v for each secret key v.

Let $n = mq + q\ell$, where ℓ is the smallest integer satisfying $r_\ell \ell \geq m$; we now describe a concept c_v over $\{0,1\}^n$. If c_v is the target concept, then an example $x \in \{0,1\}^n$ is said to be *useful* if $x_1 \cdots x_{mq} = f(v^1) \circ \cdots \circ f(v^q)$ and is *useless* otherwise. Given an example $x \in \{0,1\}^n$, let $i \in \{1, \ldots, q\}, j \in \{1, \ldots, \ell\}$ be such that $x_{mq+(i-1)\ell+j}$ is the first bit of $x_{mq+1} \cdots x_{mq+q\ell}$ whose value is 1. (If $x_{mq+1} = \cdots = x_{mq+q\ell} = 0$ then $i = j = 0$.) Figure 1 illustrates the structure of a useful example. The concept c_v is defined as follows:

- $c_v(x) = 0$ if x is useless,
- $c_v(x) = A_\ell(v^i)_j$, the j-th bit of $A_\ell(v^i)$, if x is useful and $i, j \geq 1$. If x is useful and $i = j = 0$ then $c_v(x) = 0$.

3.3 Proof of Theorem 1

First we establish that c_v is a 1-decision list. For each $1 \leq k \leq mq$, let ℓ_k denote the literal $\overline{x}_k$ if the k-th bit of $f(v^1) \circ \cdots \circ f(v^q)$ is 1, and let ℓ_k denote x_k otherwise. Then the following is seen to be a 1-decision list which computes c_v :

$$(\ell_1, 0), \ldots, (\ell_{mq}, 0), (x_{mq+1}, A_\ell(v^1)_1), \ldots, (x_{mq+(i-1)\ell+j}, A_\ell(v^i)_j), \ldots, (x_{mq+q\ell}, A_\ell(v^q)_\ell).$$

To prove the information-theoretic sample complexity upper bound, we must show that under any distribution at most $O(1/\epsilon)$ examples are required. Since each positive example contains $f(v^1) \circ \ldots \circ f(v^q)$, a computationally unbounded learner can learn the target concept exactly from a single positive example by inverting the one-way permutation f to find each v^i and then

computing each $A_\ell(v^i)$. Such a learner can thus make $20/\epsilon$ calls to the oracle $EX(c, \mathcal{D}_n)$ and output the identically zero hypothesis if all examples are negative, otherwise output the correct hypothesis as described above. A simple calculation shows that this algorithm finds an ϵ-accurate hypothesis with high probability, and hence $\mathcal{ITSC}(C_\tau; n, \epsilon) = O(1/\epsilon)$.

It remains to bound the computational sample complexity of C_τ; we begin with the simpler upper bound. We say that a 1-decision list over $\{0,1\}^n$ is *well-structured* if its length is exactly n and it has the following structure: for $1 \leq t \leq mq$ the t-th pair of the decision list has x_t or $\overline{x}_t$ as its conjunction and has 0 as its output bit, and for $1 \leq t \leq q\ell$ the $(mq+t)$-th term of the decision list has x_{mq+t} as its conjunction. Given a sample S of examples which are labelled according to the concept c_v, it is easy for a polynomial-time algorithm to find a well-structured 1-decision list which is consistent with S. Any positive example of S identifies the first mq output bits of the well-structured 1-decision list, and each useful example provides the output bit for one of the last $q\ell$ pairs (note that it is possible to identify useful examples as long as S contains at least one positive example). Since there are 2^n well-structured 1-decision lists, Occam's Razor [6] immediately implies that $O(n/\epsilon)$ examples suffice for this polynomial-time learning algorithm.

Now we show the lower bound on $\mathcal{CSC}(C_\tau; n, \epsilon)$ by exhibiting a particular distribution under which many examples are required. Let $\mathcal{D}_n$ be the distribution on $\{0,1\}^n$ which assigns total weight $1 - 3\epsilon/\alpha_\ell$ uniformly to useless examples and assigns the remaining $3\epsilon/\alpha_\ell$ weight uniformly to the $q\ell$ useful examples $\{f(v^1) \circ \cdots f(v^q) \circ 0^k 10^{q\ell-k-1}\}_{k=0}^{q\ell-1}$. Under this distribution, each bit of each $A_\ell(v^i)$ is equally likely to occur as the label of a useful example. Let S be a sample of $q\alpha_\ell/18\epsilon$ examples which are drawn from $EX(c, \mathcal{D}_n)$. A straightforward argument shows that with extremely high probability S will contain at least two useful examples (whose first mq bits must all agree). Also, by our choice of ℓ and Theorem 2, we have that $\ell = \Theta(m)$ and hence $n = \Theta(mq)$; it follows that S almost certainly will not contain a pair of useless examples whose first mq bits all agree. Consequently, with extremely high probability a polynomial-time learning algorithm which draws $|S|$ examples will be able to identify the bit strings $f(v^1), \ldots, f(v^q)$.

Suppose that a polynomial-time learning algorithm could achieve an ϵ-accurate hypothesis from the sample S. Since the algorithm knows $f(v^1), \ldots, f(v^q)$, using its ϵ-accurate hypothesis on the $q\ell$ useful examples described above it could construct $B^1, \ldots, B^q$ in polynomial time, where each B^i is an ℓ-bit string which is the learning algorithm's guess at $A_\ell(v^i)$. Since the hypothesis is ϵ-accurate under $\mathcal{D}_n$, at most an $\alpha_\ell/3$ fraction of the $q\ell$ bits in the B^i's can be incorrect. By Markov's inequality, at least $2/3$ of the B^i's must each have at most $\alpha_\ell\ell$ incorrect bits; consequently, by using the polynomial-time decoding algorithm for A_ℓ, the learning algorithm can find at least $2/3$ of the subkeys $\{v^1, \ldots, v^q\}$ in polynomial time. However, since $|S| = q\alpha_\ell/18\epsilon$, by a straightforward application of Chernoff bounds (see, e.g., [1, 20]) it is extremely unlikely that S contained more than $q/3$ useful examples; consequently, with very high probability the polynomial-time learner received no information at all (other than $f(v^i)$) for at least $2/3$ of the subkeys. It follows that the poly(n)-time learner was able to invert f on at least $1/3$ of the $f(v^i)$'s "from scratch." Since each subkey v^i is $m = \Theta(n^\tau)$ bits long, though, our poly(n)-time learner is also a poly(m)-time algorithm; but this contradicts the fact that f is one-way. Hence $\mathcal{CSC}(C_\tau; n, \epsilon) > q\alpha_\ell/18\epsilon = \Omega(n^{1-\tau}/\epsilon)$. ■

4 A stronger gap

In this section we prove the following:

Theorem 3 *Let $k \geq 1$ be any integer. If length-preserving one-way permutations exist, then there is a concept class C_k which has $\mathcal{ITSC}(C_k; n, \epsilon) = O(1/\epsilon)$ and $\mathcal{CSC}(C_k; n, \epsilon) = \Theta(n^k/\epsilon)$. where each concept in C_k over $\{0,1\}^n$ is a k-decision list.*

This strengthens the result of Decatur et. al. [9] on distribution-free computational versus information-theoretic sample complexity in two ways: we improve the upper bound on information-theoretic sample complexity from $O(n/\epsilon)$ to $O(1/\epsilon)$, and we prove this stronger gap for the much simpler class of k-decision lists (rather than poly-size circuits).

4.1 Cryptographic preliminaries

The cryptographic definitions we present in this section are slightly more general than the standard definitions (we will need this extra generality in Section 5). Throughout this section the function $q(\cdot)$ denotes an arbitrary nondecreasing integer-valued function which satisfies $q(n) \geq n$. For each of the definitions which we provide below, the standard definition can be obtained by setting $q(n) = n$ (the reader is encouraged to verify this for him/herself). The notation "$x \in \mathcal{D}_n$" means that x is selected from the set $\{0,1\}^n$ according to distribution $\mathcal{D}$; the distribution $\mathcal{U}$ is uniform.

Definition 1 *A length-preserving permutation f is said to be* $q(n)$-one-way *if there is a deterministic polynomial-time algorithm which computes $f(x)$, but for all probabilistic poly$(q(n))$-time algorithms A, for all polynomials Q, for all sufficiently large n, we have* $\Pr_{x \in \mathcal{U}_n}[A(f(x)) = x] < \frac{1}{Q(q(n))}$.

Definition 2 *Let f be a length-preserving permutation. A polynomial-time computable predicate $B : \{0,1\}^* \to \{0,1\}$ is said to be a* $q(n)$-hard-core predicate of f *if the following condition holds: for all probabilistic poly$(q(n))$-time decision algorithms A, for all polynomials Q, for all sufficiently large n, we have* $\Pr_{x \in_{\mathcal{U}_n}}[A(f(x)) = B(x)] < \frac{1}{2} + \frac{1}{Q(q(n))}$.

Suppose that g is a length-preserving $q(n)$-one-way permutation. Let $x = p \circ y$ where $|p| = |y| = n$, and let f be the function defined as $f(x) = p \circ g(y)$. It is easy to check that f is also a length-preserving $q(n)$-one-way permutation; Goldreich and Levin [13] have shown that the predicate $B(x) = \sum_{i=1}^{n} p_i y_i \pmod 2$ is a $q(n)$-hard-core predicate for f.

Definition 3 *A family of probability distributions $\{\mathcal{X}_{q(n)}\}$ on $\{0,1\}^{q(n)}$ is* $q(n)$-pseudorandom *if $\{\mathcal{X}_{q(n)}\}$ is poly$(q(n))$-time indistinguishable from $\{\mathcal{U}_{q(n)}\}$. That is, for all probabilistic poly$(q(n))$-time decision algorithms A, for all polynomials Q, for all sufficiently large n, we have*

$$\left| \Pr_{z \in \mathcal{X}_{q(n)}}[A(z) = 1] - \Pr_{z \in \mathcal{U}_{q(n)}}[A(z) = 1] \right| < \frac{1}{Q(q(n))}.$$

Definition 4 *A poly$(q(n))$-time deterministic algorithm $G : \{0,1\}^n \to \{0,1\}^{q(n)}$ is said to be a* $q(n)$-pseudorandom generator *if the family of distributions $\{\mathcal{G}_{q(n)}\}$ is $q(n)$-pseudorandom, where $\mathcal{G}_{q(n)}$ is the distribution on $\{0,1\}^{q(n)}$ obtained as follows: to select $z \in \mathcal{G}_{q(n)}$, pick $x \in \mathcal{U}_n$ and set $z = G(x)$. We write $G(z)_i$ to denote the i-th bit of $G(z)$.*

Now we can state the following useful theorem:

Theorem 4 *Let f be a length-preserving $q(n)$-one-way permutation and let B be a $q(n)$-hard-core predicate of f. Let $G : \{0,1\}^n \to \{0,1\}^{q(n)}$ be defined as follows:*

$$G(x) = B(x) \circ B(f(x)) \circ B(f(f(x))) \circ \cdots \circ B(f^{q(n)-1}(x)).$$

Then G is a $q(n)$-pseudorandom generator. Moreover, the distributions $\{G(z) \circ f^{q(n)}(z)\}_{z \in \mathcal{U}_n}$ and $\{w \circ f^{q(n)}(z)\}_{w \in \mathcal{U}_{q(n)}, z \in \mathcal{U}_n}$ are poly$(q(n))$-time indistinguishable.

In the case where $q(n)$ is a polynomial, this theorem is a standard result (see, e.g., Proposition 3.17 of [12]). This construction of a pseudorandom generator, along with the definition of a pseudorandom generator, is originally from [5]. The proof of the more general theorem which we state above is a straighforward modification of the proof of the standard result; we omit its proof from this extended abstract.

We note that by Theorem 4, even if a poly($q(n)$)-time algorithm is given $f^{q(n)}(z)$ along with some bits of $G(z)$, the algorithm still cannot predict the unseen bits of $G(z)$ with accuracy significantly better than $1/2$. This is because the ability to do such prediction would violate the poly($q(n)$)-time indistinguishability which is asserted in Theorem 4, since clearly no poly($q(n)$)-time algorithm could successfully predict the unseen bits of a uniformly selected random string.

4.2 The concept class C_k

Let f be a length-preserving one-way permutation which has a hard-core predicate, and let G be the corresponding $\binom{n}{k}$-pseudorandom generator (so G maps inputs of length m to outputs of length $\binom{m}{k}$.) Let $n = 2m$ and let $D_m = f^{-1}(\{0,1\}^m)$; D_m is the set of secret keys. For $1 \leq i \leq \binom{m}{k}$, let S_i denote the i-th k-element subset of the set $\{m+1, \ldots, 2m\}$ under some fixed and easily computable ordering (i.e. lexicographic), and let z_i be the conjunction $\prod_{j \in S_i} x_j$. Given any input $x \in \{0,1\}^n$, let i be the the smallest index in $\{1, \ldots, \binom{m}{k}\}$ such that z_i is satisfied by x (if no z_i is satisfied by x for $1 \leq i \leq \binom{m}{k}$ then let $i = 0$).

The class C_k has a concept c_v for each secret key $v \in D_m$. If c_v is the target concept, then an example x is *useful* if $x_1 \cdots x_m = f^{\binom{m}{k}}(v)$ and is *useless* otherwise. (As in Section 4.1, $f^{\binom{m}{k}}(v)$ denotes the result of applying f exactly $\binom{m}{k}$ times to v.) The concept c_v is defined as follows:

- $c_v(x) = 0$ if x is useless,
- $c_v(x) = G(v)_i$, the i-th bit of $G(v)$, if x is useful and $i \geq 1$. If x is useful and $i = 0$ then $c_v(x) = 0$.

4.3 Proof of Theorem 3

First we show that c_v is a k-decision list. For each $1 \leq j \leq m$, let ℓ_j denote the literal $\overline{x}_j$ if the j-th bit of $f^{\binom{m}{k}}(v)$ is 1, and let ℓ_j denote x_j otherwise. The following k-decision list of length $m + \binom{m}{k}$ computes c_v :

$$(\ell_1, 0), \ldots, (\ell_m, 0), (z_1, G(v)_1), \ldots, (z_{\binom{m}{k}}, G(v)_{\binom{m}{k}}).$$

To bound $\mathcal{ITSC}(C_k; n, \epsilon)$, note that upon receiving a single positive example, an unbounded learner can invert $f^{\binom{m}{k}}(v)$ to find v (this is possible since f is a permutation) and thus learn the target concept c_v exactly. As in the proof of Theorem 1, it follows that $\mathcal{ITSC}(C_k; n, \epsilon) = O(1/\epsilon)$.

An analogous argument to the computational sample complexity upper bound proof of Theorem 1 establishes that $\mathcal{CSC}(C_k; n, \epsilon) = O(\binom{m}{k}/\epsilon) = O(n^k/\epsilon)$.

For the computational lower bound, consider the distribution $\mathcal{D}_n$ over $\{0,1\}^n$ which assigns total weight $1 - 6\epsilon$ uniformly to useless examples and assigns the remaining 6ϵ weight uniformly to the $\binom{m}{k}$ useful examples $\{f^{\binom{m}{k}}(v) \circ S_i\}_{i=1}^{\binom{m}{k}}$ (here we are viewing each S_i as an m-bit string in the obvious way). Let S be a sample of $\binom{m}{k}/24\epsilon$ examples which are drawn from $EX(c, \mathcal{D}_n)$; a straightforward application of Chernoff bounds shows that with very high probability S will contain fewer than $\binom{m}{k}/2$ useful examples. By Theorem 4, we have that the string-valued random variables $\{G(z) \circ f^{\binom{m}{k}}(z)\}_{z \in \mathcal{U}_m}$ and $\{w \circ f^{\binom{m}{k}}(z)\}_{w \in \mathcal{U}_{\binom{m}{k}}, z \in \mathcal{U}_m}$ are polynomial-time indistinguishable. Consequently,

even though a polynomial-time learner may discover $f^{\binom{m}{k}}(v)$ from any positive example, such a learner cannot predict the bits of $G(v)$ which it has not seen with accuracy significantly better than $1/2$. Since useful examples which correspond to the unseen bits of $G(v)$ have weight at least 3ϵ under the distribution $\mathcal{D}_n$, the polynomial-time learner's overall error rate will exceed ϵ. Hence $\mathcal{CSC}(C_k; n, \epsilon) \geq \binom{m}{k}/24\epsilon = \Theta(n^k/\epsilon)$. ■

It is interesting to contrast the bounds given in Theorem 3 with other known bounds. The upper bound on information-theoretic sample complexity which is given in Theorem 3 is the best possible for nontrivial concept classes. Rivest's polynomial-time algorithm for learning k-decision lists [24] requires $O(\frac{n^k}{\epsilon} \min\{\log n, \log \frac{1}{\epsilon}\})$ examples; thus our lower bound on computational sample complexity could be improved by at most a logarithmic factor for concept classes of k-decision lists. Ehrenfeucht et. al. [10] have shown that $\Omega(n^k/\epsilon)$ examples are required for information-theoretic reasons for learning k-decision lists. Our Theorem 3 shows that $\Omega(n^k/\epsilon)$ examples can be required for learning subclasses of k-decision lists for computational reasons even in the absence of any information-theoretic barriers to learning from fewer examples.

5 Hardness of attribute-efficient learning

We now consider *attribute-efficient* learning algorithms. These algorithms require very few examples relative to the total number of input variables (i.e. attributes), and hence have exceptionally good performance over high-dimensional input spaces which contain many irrelevant attributes. This property has led researchers to apply attribute-efficient algorithms to real-world problems such as calendar scheduling [3], text categorization [8], and context-sensitive spelling correction [11].

Attribute-efficiency has chiefly been studied in the *on-line mistake-bound* model of concept learning which was introduced in [21, 22]. In this model learning proceeds in a series of trials, where in each trial the learner is given an unlabelled boolean example $x \in \{0,1\}^n$ and must predict the value $c(x)$; after each prediction the learner is told the true value of $c(x)$ and can update its hypothesis. The mistake bound of a learning algorithm on a target concept c is measured by the worst-case number of mistakes that the algorithm makes over all (possibly infinite) sequences of examples, and the mistake bound of a learning algorithm on a concept class C is the worst-case mistake bound across all concepts $c \in C$. A learning algorithm L for a concept class C over $\{0,1\}^n$ is said to run in polynomial time if the mistake bound of L on C is poly(n) and the time required by L to make its prediction and update its hypothesis on each example is poly(n).

A boolean function c over $x_1, \ldots, x_n$ is said to *depend* on a variable x_i if there are two vectors $y, z \in \{0,1\}^n$ which have $y_j = z_j$ for all $j \neq i$, $y_i \neq z_i$, and $c(y) \neq c(z)$. Let C be a class of boolean functions on $x_1, \ldots, x_n$ each of which depends on at most r variables and each of which has a description of length at most s under some reasonable encoding scheme. Following [4], we say that a learning algorithm L for C in the mistake-bound model is *attribute-efficient* if the mistake bound of L on any concept $c \in C$ is poly$(r, s, \log n)$.

In this section we provide strong evidence that there are polynomial-time-learnable concept classes for which attribute-efficient learning is information-theoretically possible but computationally hard. We do this by proving the following theorem:

Theorem 5 *For any integer $c \geq 2$, let $\log(c,n)$ denote $\overbrace{\log \cdots \log}^{c} n$. Let $q(c,n) = n^{\log(c,n)}$. If length-preserving $q(c,n)$-one-way permutations exist, then there is a concept class C of $\log(c,n)$-decision lists which has the following properties in the mistake-bound model:*

- *A computationally unbounded learner can learn C with at most 1 mistake,*

- *C can be learned in polynomial time,*
- *C cannot be learned in polynomial time by an attribute-efficient learning algorithm.*

We note that while the existence of length-preserving $q(c,n)$-one-way permutations is a non-standard cryptographic assumption, it is still quite weak. One can easily show, for instance, that the nonexistence of length-preserving $q(c,n)$-one-way permutations would yield a far more powerful algorithm for factoring Blum integers than any which is currently known for this well-studied problem (see the Appendix).

5.1 Proof of Theorem 5

First we define the concept class C. This construction is similar to the construction of Section 4.2 but with some different parameters.

Let f be a length-preserving $q(c,n)$-one-way permutation and let G be the corresponding $q(c,n)$-pseudorandom generator whose existence is guaranteed by Theorem 4. Let $m = \lceil n^{\frac{1}{\log(c,n)}} \rceil$ and let the set of secret keys be $D_m = f^{-1}(\{0,1\}^m)$. Let $k(n)$ be the least integer such that $\binom{m}{k(n)} \geq q(c,m)$. Straightforward manipulations show that $q(c,m) = \Theta(n^{1-\frac{\log(2c-1,n)}{\log(c,n)}})$; using the lower bound $\binom{x}{y} \geq (\frac{x}{y})^y$ it follows that $k(n) \leq \log(c,n)$.

For $i = 1, \ldots, q(c,m)$ let S_i denote the i-th $k(n)$-element subset of the set $\{m+1, \ldots, 2m\}$ and let z_i be the conjunction $\prod_{j \in S_i} x_j$. Given any input $x \in \{0,1\}^n$, let i be the the smallest index in $\{1, \ldots, q(c,m)\}$ such that z_i is satisfied by x (if no z_i is satisfied by x then $i = 0$).

For each secret key $v \in D_m$, there exists a corresponding concept $c_v \in C$. If c_v is the target concept, then an example x is *useful* if $x_1 \cdots x_m = f^{q(c,m)}(v)$ and is *useless* otherwise. The concept c_v is defined as follows:

- $c_v(x) = 0$ if x is useless,
- $c_v(x) = G(v)_i$, the i-th bit of $G(v)$, if x is useful and $i \geq 1$. If x is useful and $i = 0$ then $c_v(x) = 0$.

Now we prove that C has the properties listed in Theorem 5. The first property is easy: a computationally unbounded learner can achieve a mistake bound of 1 by predicting 0 until it makes a mistake. From this positive example the unbounded learner can compute v (by inverting $f^{q(c,m)}(v)$) and hence can exactly identify the target concept.

For the second property, note that the concept c_v can be represented as a $\log(c,n)$-decision list of length at most $m + q(c,m)$. As in the computational sample complexity upper bound of Theorem 1, a polynomial-time algorithm can learn the first m pairs of the target decision list from a single positive example, and will make at most one mistake for each of the last $q(c,m)$ pairs of the decision list. Since $q(c,m) = o(n)$, it follows that C can be learned in polynomial time.

Now suppose that there is a polynomial-time attribute-efficient learning algorithm for the concept class C. Since each concept c_v has an m-bit description (the string v), we have that $s = O(m)$. Each function c_v depends only on the variables $x_1, \ldots, x_{2m}$, so r is also $O(m)$. Hence any attribute-efficient learning algorithm for C must have mistake bound $\mathrm{poly}(m, \log n) = \mathrm{poly}(m)$.

Consider the $q(c,m)$-long sequence S of useful examples $\{f^{q(c,m)}(v) \circ S_i \circ 0^{n-2m}\}_{i=1}^{q(c,m)}$. From Theorem 4, we have that no $\mathrm{poly}(q(c,m))$-time learning algorithm can predict an unseen bit of $G(v)$ with accuracy significantly better than $1/2$. Since $q(c,m) = n^{1-o(1)}$, we have that $\mathrm{poly}(q(c,m)) = \mathrm{poly}(n)$. Hence any $\mathrm{poly}(n)$-time learning algorithm will have probability $1/2$ of making a mistake on each example in the sequence S; it follows that with extremely high probability, any $\mathrm{poly}(n)$-time

algorithm will make $\Theta(q(c,m))$ mistakes on S. But this means that no polynomial-time attribute-efficient learning algorithm can exist for C, since $\text{poly}(m) = o(q(c,m))$. ■

6 Conclusion

We have demonstrated the existence of various subclasses of k-decision lists which can be information-theoretically learned from a constant number of examples but which requires any polynomial-time learner to use $\Theta(n^k)$ examples. We have also shown that under a plausible cryptographic assumption, attribute-efficient learning is computationally hard but information-theoretically possible for a polynomial-time learnable class whose concepts are $\log(c,n)$-decision lists of length $o(n)$.

Many directions remain for future research. For one thing, it would be interesting to see if gaps such as the ones we have demonstrated in Sections 3 and 4 can be shown for concept classes whose concepts are even simpler than decision lists, and to determine whether the cryptographic assumptions which are used to establish these gaps can be weakened. In a similar vein, it would be nice to be able to replace the cryptographic assumption which is used to prove Theorem 5 with a more standard cryptographic assumption.

As noted in Section 5, each concept of the class described there has a natural m-bit representation. However, to represent a concept in this class as a decision list requires more than m bits; our attribute-efficient hardness result relies on the m-bit representation. It would be very interesting to see an attribute-efficient hardness result for a concept class of decision lists where the description length of a concept is taken to be the length of the decision list which computes it.

A related goal is to establish hardness results for attribute-efficient learning of simpler concept classes. In particular, let L_k denote the class of 1-decision lists of length k. Using the Halving Algorithm [21], the class L_k can be learned with $O(k \log n)$ mistakes, but no polynomial-time algorithm is known which makes $\text{poly}(k, \log n)$ mistakes; currently known polynomial-time algorithms make $\Theta(\min(kn, k^{2k} \log n))$ mistakes [2]. Perhaps techniques such as those used in this paper can help resolve whether L_k is attribute-efficiently learnable in polynomial time.

7 Acknowledgements

This research was supported by an NSF Graduate Fellowship and by grants NSF-CCR-95-04436 and ONR-N00014-96-1-0550. Thanks go to Les Valiant and Amos Beimel for helpful suggestions which greatly improved the structure of the paper.

References

[1] D. Angluin, L. G. Valiant, *Fast probabilistic algorithms for Hamiltonian circuits and matchings,* J. Comput. System Sci., 18 (1979), pp. 155-193.

[2] A. Blum, *On-Line Algorithms in Machine Learning,* available at `http://www.cs.cmu.edu/~avrim/Papers/pubs.html`, 1996.

[3] A. Blum, *Empirical support for winnow and weighted-majority algorithms: results on a calendar scheduling domain,* Machine Learning 26 (1997), pp. 5-23.

[4] A. Blum, L. Hellerstein, N. Littlestone, *Learning in the presence of finitely or infinitely many irrelevant attributes,* J. Comput. System Sci., 50 (1995), pp. 32-40.

[5] M. Blum, S. Micali, *How to generate cryptographically strong sequences of pseudo-random bits,* SIAM J. Comput., 13 (1984), pp. 850-864.

[6] A. Blumer, A. Ehrenfeucht, D. Haussler, M. K. Warmuth, *Occam's Razor,* Inform. Process. Lett., 24 (1987), pp. 377-380.

[7] N. Bshouty, L. Hellerstein, *Attribute efficient learning with queries,* Proc. Ninth Ann. Conf. on Comp. Learning Theory, ACM Press, New York, NY, 1996.

[8] I. Dagan, Y. Karov, D. Roth, *Mistake-Driven Learning in Text Categorization,* in 2nd Conf. on Empirical Methods in Natural Language Processing (EMNLP-97), 1997.

[9] S. E. Decatur, O. Goldreich, D. Ron, *Computational Sample Complexity,* in Proc. Tenth Ann. Conf. on Comp. Learning Theory, ACM Press, New York, NY, 1997, pp. 130-142.

[10] A. Ehrenfeucht, D. Haussler, M. Kearns, L. Valiant, *A general lower bound on the number of examples needed for learning,* Inform. and Comput., 82 (3) (1989), pp. 247-261.

[11] A.R. Golding, D. Roth, *A Winnow-based approach to spelling correction,* Machine Learning, Special Issue on Machine Learning and Natural Language Processing (1998), to appear.

[12] O. Goldreich, *Foundations of Cryptography (Fragments of a Book)*, at `http://www.wisdom.weizmann.ac.il/~oded/frag.html`, 1995.

[13] O. Goldreich, L. Levin, *A Hard-Core Predicate for all One-Way Functions,* in Proc. 21st Ann. Symp. on Theory of Comp., ACM Press, New York, NY, 1995, pp. 25-32.

[14] S. Goldwasser, M. Bellare, *Lecture notes on cryptography,* at `http://www-cse.ucsd.edu/users/mihir/papers/gb.html`, 1996.

[15] D. Haussler, *Quantifying inductive bias: AI learning algorithms and Valiant's learning framework,* Artificial Intelligence, 36 (1988), pp. 177-221.

[16] M. Kharitonov, *Cryptographic lower bounds for learnability of boolean functions on the uniform distribution,* J. Comput. System Sci., 50 (1995), pp. 600-610.

[17] M. Kearns, M. Li, L. Pitt, L. Valiant, *Recent results on boolean concept learning,* in P. Langley, editor, Proc. 4th Int. Workshop on Machine Learning, Morgan Kaufmann, Los Altos, CA, 1987, pp. 337-352.

[18] M. J. Kearns, R. E. Schapire, *Efficient distribution-free learning of probabilistic concepts,* in Proc. 31st Symp. on Found. of Comp. Sci. , IEEE Comp. Society Press, Los Alamitos, CA, 1990, pp. 382-391.

[19] M. Kearns, L. G. Valiant, *Cryptographic limitations on learning boolean formulae and finite automata,* J. ACM 41(1), (1994), pp. 67-95.

[20] M. Kearns, U. Vazirani, *An introduction to computational learning theory,* MIT Press, Cambridge, MA, 1994.

[21] N. Littlestone, *Learning quickly when irrelevant attributes abound: a new linear-threshold learning algorithm,* Machine Learning 2 (1988), pp. 285-318.

[22] N. Littlestone, *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms,* Ph.D. thesis, Technical Report UCSC-CRL-89-11, Univ. of Calif., Santa Cruz, 1989.

[23] M. Rabin, *Digitalized signatures as intractable as factorization,* Technical Report MIT/LCS/TR-212, MIT Lab. for Comp. Sci., 1979.

[24] R. L. Rivest, *Learning decision lists,* Machine Learning 2(3), (1987), pp. 229-246.

[25] M. Sipser, D. A. Spielman, *Expander Codes,* in Proc. 35th Symp. on Found. of Comp. Sci., IEEE Comp. Society Press, Los Alamitos, CA, 1994, pp. 566-576.

[26] D. A. Spielman, *Linear-time encodable and decodable error-correcting codes,* in Proc. 27th Ann. Symp. on Theory of Comp., ACM Press, New York, NY, 1995, pp. 388-397.

[27] R. Uehara, K. Tsuchida, I. Wegener, *Optimal attribute-efficient learning of disjunction, parity, and threshold functions,* Electronic Colloquium on Computational Complexity (061): (1996).

[28] L. G. VALIANT, *A theory of the learnable,* Comm. ACM, 27(11) (1984), pp. 1134-1142.

[29] L. G. VALIANT, *Projection learning,* in Proc. Eleventh Ann. Conf. on Comp. Learning Theory, ACM Press, New York, NY, 1998, pp. 287-293.

8 Appendix

We give a simple proof that if Blum integers are sufficiently hard to factor, then length-preserving $q(c,n)$-one-way permutations exist. An integer is a Blum integer if it is the product of two distinct primes each of which is congruent to 3 mod 4.

Definition 5 *The* $q(n)$-Blum factoring assumption *is the following: for all probabilistic poly*$(q(n))$*-time factoring algorithms* A*, for all polynomials* Q*, for all sufficiently large* n*, the probability (taken over the internal coin tosses of algorithm* A *and the choice of inputs for algorithm* A*) that* A *can produce a nontrivial divisor of its input (where the input is the product of two randomly chosen distinct* $n/2$*-bit primes each congruent to 3 mod 4) is less than* $1/Q(q(n))$*.*

Claim 1 *If the* $q(n)$*-Blum factoring assumption is true, then length-preserving* $q(n)$*-one-way permutations exist.*

Proof: Let N be an n-bit Blum integer selected as in Definition 5, and let Q_N be the set of quadratic residues mod N. Let the function $R_N : Q_N \rightarrow Q_N$ be defined as $R_N(x) = x^2 \bmod N$. As Blum and Williams have noted, R_N is a length-preserving permutation on Q_N (see Lemma 2.3.29 of [14] for a proof). Rabin has shown [23] that a poly$(q(n))$-time algorithm which succeeds in inverting R_N with probability $1/\text{poly}(q(n))$ would yield a probabilistic poly$(q(n))$-time algorithm for factoring N with success probability $1/\text{poly}(q(n))$, where the success probability is taken over the choice of N and over the algorithm's internal coin tosses. But this would contradict the $q(n)$-Blum factoring assumption. Hence if the $q(n)$-Blum factoring assumption is true, there exists a length-preserving $q(n)$-one-way permutation. ■