Cryptographic Limitations on Learning Boolean Formulae and Finite Automata *

Michael Kearns
Harvard University

Leslie G. Valiant
Harvard University

1 Introduction

In this paper we consider the problem of learning from examples classes of functions when there are no restrictions on the allowed hypotheses other than that they are polynomial time evaluatable. We prove that for Boolean formulae, finite automata, and constant depth threshold circuits (simplified neural nets), this problem is computationally as difficult as the quadratic residue problem, inverting the RSA function and factoring Blum integers (composite number $p \cdot q$ where p and q are both primes congruent to 3 modulo 4). These results are for the distribution-free model of learning [31]. They hold even when the inference task is that of deriving a probabilistic polynomial-time classification algorithm that predicts the correct value of a random input with probability $\frac{1}{2} + \frac{1}{p(s)}$, where s is the size of the formula, automaton or circuit, and p is any polynomial. (We call this model *weak learning*). Previously the only non-learnability results that were similarly independent of hypothesis representation were those implied by the work of Goldreich, Goldwasser and Micali [17], for such classes as unrestricted Boolean circuits [25,31]. In addition to the particular results stated above we can abstract from our method a general technique for proving nonlearnability based on the existence of trapdoor functions in the sense of Yao [34].

We apply our learning results to prove hardness results for approximating combinatorial optimization problems. In particular, we define a problem that generalizes the graph coloring problem and prove that approximating the optimal solution by a multiplicative factor $(opt)^{\alpha} \mid I \mid^{\beta}$ for any $\alpha \geq 0$ and $0 \leq \beta < \frac{1}{2}$, where *opt* is the value of the optimal solution for instance I, is as hard as the cryptographic problems mentioned above. This illustrates that cryptographic assumptions may suffice to give negative results for combinatorial optimization in some cases where no NP-hardness results are known.

In the final section, we give a positive result in the weak learning model. We give an algorithm for weakly learning the class of all monotone Boolean functions when the distribution on the examples is uniform. We note that this result is optimal in the sense that it would become false under any of the following relaxations: replacing weak learning by learnilng, monotone functions by arbitrary functions, or uniform distributions by arbitrary distributions. We then prove the equivalence of the weak learning model to the model of *group learning*, where a polynomial-sized group of examples that are either all positive or all negative must be classified with high accuracy.

2 Background and Definitions

2.1 Background and Definitions from Computational Learning Theory

Concept Classes and Their Representation. Let X be a set called a *domain*. A *representation class over* X is a pair (σ, C), where $C \subseteq \{0,1\}^*$ and σ is a mapping $\sigma : C \rightarrow 2^X$. For $c \in C$, $\sigma(c)$ is called a *concept* over X; the image space $\sigma(C)$ is the *concept class* that is *represented* by (σ, C). For $c \in C$, we define $pos(c) = \sigma(c)$ (the *positive examples* of c) and $neg(c) = X - \sigma(c)$ (the *negative examples* of c). Although intuitively we can think of algorithms as learning concepts chosen from some concept class, for computational purposes we need a way of naming these concepts; thus we shall discuss the learnability

*Part of this research was done while the authors were visiting Oxford University and while M. Kearns was visiting A.T. & T. Bell Laboratories. M. Kearns was supported by an A.T. & T. Bell Laboratories Scholarship. L.G. Valiant was partially supported by grants NSF-DCR-86-00379, ONR-N00014-85-K-0445, DAAL03-86-K-0171 and by SERC. Authors' address: Aiken Computation Lab, Harvard University, Cambridge MA 02138.

of representations rather than the learnability of the concepts themselves. The domain X and the mapping σ will usually be clear from the context, and we will simply refer to the *representation class* C. We assume that points $x \in X$ and representations $c \in C$ are encoded using any standard scheme, and use the notation $|x|$ and $|c|$ to denote the number of bits in these encodings.

In this paper we will study the learnability of *parameterized* classes of representations. Here we have a stratified domain $X = \bigcup_{n \geq 1} X_n$ and representation class $C = \bigcup_{n \geq 1} C_n$. The parameter n can be regarded as some measure of the complexity of concepts in $\sigma(C)$, and we assume that for a representation $c \in C_n$ we have $pos(c) \subseteq X_n$ and $neg(c) = X_n - pos(c)$. In this paper X_n will always be $\{0,1\}^n$. We may further parameterize C by parameterizing C_n, for example $C_n = \bigcup_{s \geq 1} C_{n,s}$.

Efficient evaluation of representations. If C is a representation class over X, we say that C is *polynomially evaluatable* if there is a (possibly randomized) evaluation algorithm that, on inputs a representation $c \in C$ and $x \in X$, runs in time polynomial in $|c|$ and $|x|$ and decides if $x \in \sigma(c)$.

Boolean Formulae, Finite Automata, Threshold Circuits. Although our results hold for many representation classes, we shall illustrate them primarily via the following examples:

Boolean Formulae: Let BF_n denote the class of Boolean formulae over the variables $x_1, \ldots, x_n$. $BF_{n,s}$ is the subset of BF_n in which each formula has length at most s (in some standard encoding).

Acyclic Finite Automata: Let $ADFA_n$ denote the class of deterministic finite automata that accept only strings of length n. $ADFA_{n,s}$ is the subset of $ADFA_n$ in which each automata has size at most s (in some standard encoding).

Constant-depth Threshold Circuits: A *threshold gate* is an unbounded fan-in gate with Boolean inputs $x_1, \ldots, x_n$ and their negation, and is specified by a value $0 \leq i \leq n$. The output of the gate is 1 if and only if at least i of the inputs are 1. A *threshold circuit* is a circuit of threshold gates and their negation. For constant d, let $CDTC_n^d$ denote the class of threshold circuits of depth at most d over inputs $x_1, \ldots, x_n$. $CDTC_{n,s}^d$ is the subset of $CDTC_n^d$ in which each circuit has size at most s (in some standard encoding).

It is easily seen that these classes are polynomially evaluatable.

Examples and Samples. If C is a representation class and $c \in C$, then a *labeled example* of c is a pair $< x, b >$, where $x \in X$ and $b \in \{0,1\}$ is such that $b = 1$ if and only if $x \in pos(c)$. We say that a representation h and an example $< x, b >$ *agree* if $x \in pos(h)$ if and only if $b = 1$; otherwise they *disagree*. A *sample* $S =< x_1, b_1 >, \ldots, < x_m, b_m >$ of $c \in C$ is a finite sequence of m labeled examples of c. We denote by $|S|$ the number of bits in the sample S. We say that a representation h and a sample S are *consistent* if h agrees with each example in S.

Distributions on Examples and Error Measures. Let $c \in C$ be a distinguished *target representation*. Let D_c^+ be a fixed but arbitrary probability distribution over $pos(c)$, and let D_c^- be a fixed but arbitrary probability distribution over $neg(c)$. We call these distributions the *target distributions*. When learning c, learning algorithms will be given access to two oracles, *POS* and *NEG*, that behave as follows: oracle *POS* (respectively, *NEG*) returns in unit time a positive (negative) example of the target representation, drawn randomly according to the target distribution D_c^+ (D_c^-).

Given a fixed target representation $c \in C$, and given fixed target distributions D_c^+ and D_c^-, there is a natural measure of the *error* of a representation h from a representation class H. We define $e_c^+(h) = D_c^+(neg(h))$ and $e_c^-(h) = D_c^-(pos(h))$. Note that $e_c^+(h)$ (respectively, $e_c^-(h)$) is simply the probability that a random positive (negative) example of c is identified as negative (positive) by h. If both $e_c^+(h) < \epsilon$ and $e_c^-(h) < \epsilon$, then we say that h is an *ϵ-good* hypothesis (with respect to D_c^+ and D_c^-); otherwise, h is *ϵ-bad*. When the target representation c is clear from the context, we will drop the subscript c and simply write D^+, D^-, e^+ and e^-. We use $\mathbf{Pr}_{\vec{v} \in D^+}(E)$ ($\mathbf{Pr}_{\vec{v} \in D^-}(E)$, respectively) to denote the probability of event E when $\vec{v}$ is drawn randomly from the distribution D^+ (D^-).

Learnability. Let C and H be polynomially evaluatable parameterized representation classes over X. Then C is *learnable by* H if there is an algorithm A with access to *POS* and *NEG*, taking inputs ϵ, δ, with the property that for any $n \geq 1$, for any target representation $c \in C_n$, for any target distributions D^+ over $pos(c)$ and D^- over $neg(c)$, and for any input values $0 < \epsilon, \delta < 1$, algorithm A halts and outputs a representation $h_A \in H$ that with probability greater than $1 - \delta$ satisfies

$$(i)\ e^+(h_A) < \epsilon$$

and

$$(ii)\ e^-(h_A) < \epsilon.$$

We call C the *target class* and H the *hypothesis class*; the output $h_A \in H$ is called the *hypothesis* of

A. A will be called a *learning algorithm* for C. If for target representation $c \in C_n$ algorithm A runs in time polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}$, $|c|$ and n, then we say that C is *polynomially learnable by* H. We say C is *polynomially learnable* to mean that C is polynomially learnable by H for some polynomially evaluatable H. We will sometimes call ϵ the *accuracy parameter* and δ the *confidence parameter*. Learning by such polynomially evaluatable H it is called *prediction* in [18] [19] [25]

Weak Learnability. We will also consider a model in which the hypothesis of the learning algorithm is required only to perform slightly better than random guessing.

Let C and H be polynomially evaluatable parameterized representation classes over X. Then C is *weakly learnable by* H if there is a polynomial p and an algorithm A with access to POS and NEG, taking input δ, with the property that for any $n \geq 1$, for any target representation $c \in C_n$, for any target distributions D^+ over $pos(c)$ and D^- over $neg(c)$, and for any input value $0 < \delta < 1$, algorithm A halts and outputs a representation $h_A \in H$ that with probability greater than $1 - \delta$ satisfies

$$(i)\ e^+(h_A) < \frac{1}{2} - \frac{1}{p(|c|,n)}$$

and

$$(ii)\ e^-(h_A) < \frac{1}{2} - \frac{1}{p(|c|,n)}.$$

A will be called a *weak learning* algorithm for C. We say that C is *polynomially weakly learnable by* H if for target representation $c \in C_n$, A runs in time polynomial in $\frac{1}{\delta}$, $|c|$ and n, and C is *polynomially weakly learnable* if it is polynomially weakly learnable by H for some polynomially evaluatable H.

Distribution-specific Learnability. We will also consider the above models of learnability under restricted classes of target distributions, for instance the uniform distribution. Here the definitions are the same as before, except that we only ask that the performance criteria for learnability be met under these restricted target distributions.

2.2 Background and Definitions from Cryptography

Some Basic Number Theory. For N a natural number, Z_N will denote the ring of integers modulo N, and Z_N^* will denote the multiplicative group modulo N. Thus, we write $Z_N = \{x : 0 \leq x \leq N-1\}$ and $Z_N^* = \{x : 1 \leq x \leq N-1 \text{ and } gcd(x,N) = 1\}$. The *Euler totient function* φ is defined by $\varphi(N) = |Z_N^*|$. For $x \in Z_N^*$, we say that x is a *quadratic residue* modulo N if there is an $a \in Z_N^*$ such that $x = a^2 \ mod\ N$. We denote by QR_N the set of all quadratic residues in Z_N^*. For a prime p and $x \in Z_p^*$, we define the *Legendre symbol* of x with respect to p by $L(x,p) = 1$ if x is a quadratic residue modulo p, and $L(x,p) = -1$ otherwise. For $N = p \cdot q$, where p and q are prime, we define the *Jacobi symbol* of $x \in Z_N^*$ with respect to N by $J(x,N) = L(x,p) \cdot L(x,q)$. Since x is a quadratic residue modulo N if and only if it is a quadratic residue modulo p and modulo q, it follows that $J(x,N) = -1$ implies that x is not a residue modulo N. However, $J(x,N) = 1$ does not necessarily imply that x is a residue mod N. We define the sets $Z_N^*(+1) = \{x \in Z_N^* : J(x,N) = 1\}$ and $QR_N(+1) = \{x \in QR_N : J(x,N) = 1\}$. A *Blum integer* is an integer of the form $p \cdot q$, where p and q are primes both congruent to 3 modulo 4.

We will make use of the following facts from number theory throughout the paper [4], [22]:

Fact 1. On inputs x and N, $gcd(x,N)$ can be computed in polynomial time.

Fact 2. For p a prime and $x \in Z_p^*$, $L(x,p) = x^{\frac{p-1}{2}} \ mod\ p$.

Fact 3. On inputs x and N, $J(x,N)$ can be computed in polynomial time.

Fact 4. For $N = p \cdot q$ where p and q are prime, $|Z_N^*(+1)| = \frac{|Z_N^*|}{2}$ and $|QR_N(+1)| = \frac{|Z_N^*|}{4}$.

Fact 5. For any $x \in Z_N^*$, $x^{\varphi(N)} = 1 \ mod\ N$.

The RSA Encryption Function. Let p and q be primes of length n, and $N = p \cdot q$. Let e be an *encrypting exponent* such that $gcd(e, \varphi(N)) = 1$ and d a *decrypting exponent* such that $d \cdot e = 1 \ mod\ \varphi(N)$. The existence of such a d is guaranteed by multiplicative inverses modulo $\varphi(N)$. We then define the RSA encryption function [30] by

$$RSA(x,N,e) = x^e \ mod\ N.$$

Note that

$$(x^e)^d = x^{e \cdot d} \ mod\ N = x^{1+i \cdot \varphi(N)} \ mod\ n = x \ mod\ n$$

for some i because $e \cdot d = 1 \ mod\ \varphi(n)$ and by Fact 5.

There is currently no known polynomial-time algorithm for *inverting* the RSA encryption function — that is, computing x on inputs $RSA(x,N,e), N$ and e. Furthermore, the following result from [3] indicates that even guessing the least significant bit of x is as hard as inverting RSA.

Theorem 1 [3] *Let x, N and e be as above. Then the following problems are probabilistic polynomial time reducible to each other:*
(1) On input $RSA(x,N,e), N$ and e, compute x.
(2) On input $RSA(x,N,e), N$ and e, compute $LSB(x)$ with probability exceeding $\frac{1}{2} + \frac{1}{p(n)}$, where p is any fixed

polynomial, n is the length of N, and $LSB(x)$ denotes the least significant bit of x. The probability is taken over x chosen uniformly from Z_N and the coin tosses of A.

The Rabin and Modified Rabin Encryption Functions. The Rabin encryption function [28] is specified by two primes p and q of length n. For $N = p \cdot q$ and $x \in Z_N^*$, we define

$$R(x, N) = x^2 \ mod \ N.$$

Known results regarding the security of the Rabin function include the following:

Theorem 2 [28] *Let x and N be as above. Then the following problems are probabilistic polynomial time reducible to each other:*
(1) On input N, compute a nontrivial factor of N.
(2) On input N and $R(x, N)$, compute x.

Furthermore, this reduction still holds when N is the product of two primes both congruent to 3 modulo 4 in each problem.

The modified Rabin encryption function [3] is specified by two primes p and q of length n, both congruent to 3 modulo 4. Let $N = p \cdot q$ (thus N is a Blum integer). We define a subset M_N of Z_N^* by

$$M_N = \{x : 0 \leq x \leq \frac{N}{2} \text{ and } x \in Z_N^*(+1)\}.$$

For $x \in M_N$, the modified Rabin encryption function is then

$$MR(x, N) = x^2 \ mod \ N \text{ if } x^2 \ mod \ N \leq \frac{n}{2}$$

$$MR(x, N) = N - x^2 \ mod \ N \text{ otherwise.}$$

This defines a 1-1 map from M_N onto M_N.

Theorem 3 [3] *Let x and N be as above. Then the following problems are probabilistic polynomial time reducible to each other:*
(1) On input $MR(x, N)$ and N, compute x.
(2) On input $MR(x, N)$ and N, compute $LSB(x)$ with probability exceeding $\frac{1}{2} + \frac{1}{p(n)}$, where p is any fixed polynomial and n is the length of N. The probability is taken over x chosen uniformly from M_N and the coin tosses of A.

For Blum integers, $R(x, N)$ is a 1-1 mapping of QR_N. Hence if $MR(x, N)$ is invertible then we can invert $R(x, N)$ by attempting to invert MR for both the values $R(x, N)$ and $N - R(x, N)$, and succeeding for just the right one of these. Hence Theorems 2 and 3 together imply that Problem (2) in Theorem 3 is equivalent to factoring Blum integers, a problem for which no polynomial time algorithm is known.

The Quadratic Residue Assumption. Let $N = p \cdot q$, where p and q are primes of length n. For each $x \in Z_N^*(+1)$, define $QR(x, N) = 1$ if x is a quadratic residue mod N and $QR(x, N) = 0$ otherwise. Then the *Quadratic Residue Assumption* states that if A is any probabilistic polynomial-time algorithm that accepts N and x as input, then for infinitely many N we have

$$\mathbf{Pr}(A(N, x) = QR(x, N)) \ < \ \frac{1}{2} + \frac{.1}{p(n)}$$

where p is any fixed polynomial, and the probability is taken over x chosen uniformly from the set $Z_N^*(+1)$ and the coin tosses of A.

3 Learning Problems Based on Cryptographic Functions.

In this section we construct hard learning problems based on the number-theoretic encryption functions described above. For each such function, we first define a representation class based on the function. For each target representation in this class, we then describe the *relevant examples* for this representation. These are the only examples with non-zero probability in the target distributions we define. We then proceed to prove the difficulty of weakly learning the representation class under the chosen distributions, based on some standard cryptographic assumption on the security of the underlying encryption function. Finally, we show the ease of *evaluating* the representation class: more precisely, we show that each representation in the class can be computed by an NC^1 circuit (a polynomial size, log-depth circuit of standard fan-in 2 Boolean gates). In Section 4 we apply these results to prove that weakly learning Boolean formulae, finite automata, constant-depth threshold circuits and a number of other representation classes is hard under cryptographic assumptions.

We adopt the following notation: if $a_1, \ldots, a_m$ are natural numbers, we denote by $bin(a_1, \ldots, a_m)$ the binary representation of the sequence $a_1, \ldots, a_m$. The relevant examples we construct will be of the form

$$< bin(a_1, \ldots, a_m), b >$$

where b is a bit indicating whether the example is positive or negative. We denote by $powers(x, N)$ the sequence $x \ mod \ N$, $x^2 \ mod \ N$, $x^4 \ mod \ N, \ldots$, $x^{2^n} \ mod \ N$ of the first $n+1$ successive square powers of x modulo N, where n is the length of N.

3.1 A Learning Problem Based on the RSA Encryption Function

The Representation Class C_n: Let l be the largest natural number satisfying $4l^2 + 6l \leq n$. Each representation in C_n is a triple (p, q, e). Here p and q are l-bit primes and $e \in Z^*_{\varphi(N)}$, where $N = p \cdot q$ (thus, $gcd(e, \varphi(N)) = 1$).

Relevant Examples for $(p, q, e) \in C_n$: A relevant example of (p, q, e) is of the form

$$< bin(powers(RSA(x, N, e), N), N, e), LSB(x) >$$

where $x \in Z_N$. Note that each such example is of length at most n, so C_n is a concept class over $\{0,1\}^n$. The target distribution D^+ is uniform over the relevant positive examples (i.e., those for which $LSB(x) = 1$) and the target distribution D^- is uniform over the relevant negative examples (i.e., those for which $LSB(x) = 0$).

Difficulty of Weakly Learning C_n: Suppose that for any $n \geq 1$, A is a polynomial-time weak learning algorithm for C_n. We now describe how we can use algorithm A to invert the RSA encryption function. Let N be the product of two unknown l-bit primes p and q, and let $e \in Z^*_{\varphi(N)}$. Then given only N and e, we run algorithm A. Each time A requests a positive example of (p, q, e), we uniformly choose an $x \in Z_N$ such that $LSB(x) = 1$ and give the example

$$< bin(powers(RSA(x, N, e), N), N, e), 1 >$$

to A. This simulation generates the target distribution D^+. Each time that A requests a negative example of (p, q, e), we uniformly choose an $x \in Z_N$ such that $LSB(x) = 0$ and give the example

$$< bin(powers(RSA(x, N, e), N), N, e), 0 >$$

to A. This simulation generates the target distribution D^-. Let h_A be the hypothesis output by algorithm A following this simulation. Then given $r = RSA(x, N, e)$ for some unknown x chosen uniformly from Z_N, $h_A(bin(powers(r, N), N, e)) = LSB(x)$ with probability at least $\frac{1}{2} + \frac{1}{p(n)}$ for some polynomial p by the definition of weak learning. Thus we have a polynomial advantage for inverting the least significant bit of RSA. This allows us to invert RSA by the results of [3].

Ease of Evaluating $(p, q, e) \in C_n$: For each $(p, q, e) \in C_n$, we give an NC^1 circuit for evaluating the concept represented by (p, q, e) on an input of the form

$$bin(powers(r, N), N, e)$$

where $N = p \cdot q$ and $r = RSA(x, N, e)$ for some $x \in Z_N$.

Since $e \in Z^*_{\varphi(N)}$, there is a $d \in Z^*_{\varphi(N)}$ such that $e \cdot d = 1 \ mod \ \varphi(N)$. Thus, $r^d \ mod \ N = x^{e \cdot d} \ mod \ N = x \ mod \ N$. Hence the circuit simply multiplies the appropriate powers of r (which are explicitly provided in the input) to compute $r^d \ mod \ N$, and outputs the least significant bit of the resulting product. This is an NC^1 step by the iterated product circuits of Beame, Cook and Hoover [6].

3.2 A Learning Problem Based on Quadratic Residues

The Representation Class C_n: Let l be the largest natural number satisfying $4l^2 + 4l \leq n$. Each representation in C_n is defined by a pair of l-bit primes (p, q).

Relevant Examples for $(p, q) \in C_n$: For a representation (p, q) in C_n, let $N = p \cdot q$. We consider only points $x \in Z^*_N(+1)$. A relevant example of (p, q) is then of the following form:

$$< bin(powers(x, N), N), QR(x, N) > .$$

The target distribution D^+ is uniform over the relevant positive examples (i.e., those for which $QR(x, N) = 1$) and the target distribution D^- is uniform over the relevant negative examples (i.e., those for which $QR(x, N) = 0$).

Difficulty of Weakly Learning C_n: Suppose that for any $n \geq 1$, A is a polynomial-time weak learning algorithm for C_n. We now describe how we can use algorithm A to recognize quadratic residues. Let N be the product of two unknown l-bit primes p and q. Given only N, we run algorithm A. Every time A requests a positive example of (p, q), we uniformly choose $y \in Z^*_N$ and give the example

$$< bin(powers(y^2 \ mod \ N, N), N), 1 >$$

to A. This simulation generates the target distribution D^+.

In order to generate the negative examples for our simulation of A, we uniformly choose $u \in Z^*_N$ until $J(u, N) = 1$. By Fact 4, this can be done with high probability in polynomial time. The probability that such a u is a non-residue modulo N is $\frac{1}{2}$. Assuming we have obtained a non-residue u, every time A requests a negative example of (p, q), we uniformly choose $y \in Z^*_N$ and give the example

$$< bin(powers(uy^2 \ mod \ N, N), N), 0 >$$

to A. Note that if u actually is a non-residue then this simulation generates the target distribution D^-, and

this run of A will with high probability produce an hypothesis h_A with accuracy at least $\frac{1}{2} + \frac{1}{p(n)}$ with respect to D^+ and D^-, for some polynomial p (call such a run a *good* run); if u is a residue then A has been trained improperly (A has been given positive examples when it requested negative examples), and no performance guarantees can be assumed. The probability of a good run of A is at least $\frac{1}{2}(1-\delta)$.

We thus simulate A as described many times, testing each hypothesis to determine if the run was a good run. To test if a good run has occurred, we first determine if h_A has accuracy at least $\frac{1}{2} + \frac{1}{2p(n)}$ with respect to D^+. This can be determined with high probability by generating D^+ as above and estimating the accuracy of h_A using Chernoff bounds. If h_A passes this test, we then estimate the probability that h_A classifies an example as positive when this example is drawn from the uniform distribution over *all* relevant examples (both positive and negative). This can be done by simply choosing $x \in Z_N^*$ uniformly and computing $h_A(bin(powers(x,N),N))$. The probability that h_A classifies such examples as positive is exactly $\frac{1}{2}$ if and only if h_A has equal accuracy on D^+ and D^-. Thus by estimating the accuracy of h_A on D^+, we can estimate the accuracy of h_A on D^- as well.

We continue to run A and test until a good run of A is obtained with high probability. Then given x chosen randomly from Z_N^*, $h_A(bin(powers(x,N),N)) = QR(x,N)$ with probability at least $\frac{1}{2} + \frac{1}{p(n)}$, contradicting the Quadratic Residue Assumption.

Ease of Evaluating $(p,q) \in C_n$: For each $(p,q) \in C_n$, we give an NC^1 circuit for evaluating the concept represented by (p,q) on an input of the form

$$bin(powers(x,N),N)$$

where $N = p \cdot q$ and $x \in Z_N^*$.

Phase I. Compute the powers

$$x \bmod p, x^2 \bmod p, x^4 \bmod p, \ldots, x^{2^n} \bmod p$$

and the powers

$$x \bmod q, x^2 \bmod q, x^4 \bmod q, \ldots, x^{2^n} \bmod q.$$

Since for any $a \in Z_N^*$ we have that $a \bmod p = (a \bmod N) \bmod p$, these powers can be computed from the inputs by parallel $\bmod p$ and $\bmod q$ circuits. Each such circuit involves only a division step followed by a multiplication and a subtraction. The results of [6] imply that these steps can be carried out by an NC^1 circuit.

Phase II. Compute $x^{\frac{p-1}{2}} \bmod p$ and $x^{\frac{q-1}{2}} \bmod q$. These can be computed by multiplying the appropriate powers mod p and mod q computed in Phase I. Since the iterated product of n n-bit numbers can be computed in NC^1 by the results of [6], this is also an NC^1 step.

Phase III. Determine whether $x^{\frac{p-1}{2}} = 1 \bmod p$ or $x^{\frac{p-1}{2}} = -1 \bmod p$, and whether $x^{\frac{q-1}{2}} = 1 \bmod q$ or $x^{\frac{q-1}{2}} = -1 \bmod q$. That these are the only cases follows from Fact 2; furthermore, this computation determines whether x is a residue mod p and mod q. Given the outputs of Phase II, this is clearly an NC^1 step.

Phase IV. If the results of Phase III were $1 \bmod p$ and $1 \bmod q$, then output 1, otherwise output 0. This is again an NC^1 step.

3.3 A Learning Problem Based on Factoring Blum Integers

The Representation Class C_n: Let l be the largest natural number satisfying $4l^2 + 4l \leq n$. Each representation in C_n is defined by a pair of l-bit primes (p,q), both congruent to 3 modulo 4. Thus the product $N = p \cdot q$ is a Blum integer.

Relevant Examples for $(p,q) \in C_n$: We consider points $x \in M_N$. A relevant example of (p,q) is then of the form

$$< bin(powers(MR(x,N),N),N), LSB(x) > .$$

The target distribution D^+ is uniform over the relevant positive examples (i.e., those for which $LSB(x) = 1$) and the target distribution D^- is uniform over the relevant negative examples (i.e., those for which $LSB(x) = 0$).

Difficulty of Weakly Learning C_n: Proved using [28] and [3].

Ease of Evaluating $(p,q) \in C_n$: Proved using the root finding algoritm in [1].

4 Learning Boolean Formulae, Finite Automata and Threshold Circuits is Hard

The results of Section 3 show that for some polynomial p_1, learning NC^1 circuits of size $p_1(n)$ is computationally as difficult as the problems of inverting

RSA, recognizing quadratic residues, and factoring Blum integers. However, NC^1 circuit families have polynomial formula size. Thus we have proved the following.

Theorem 4 *For n large enough, and for some polynomial p, the problems of recognizing quadratic residues, inverting the RSA encryption function and factoring Blum integers are probabilistic polynomial-time reducible to weakly learning $BF_{n,p(n)}$, the class of Boolean formulae of size at most $p(n)$, where n is the number of variables.*

In fact, we can apply the substitution arguments of [21] to show that Theorem 4 also holds for the class of monotone Boolean formulae in which each variable appears at most once.

Pitt and Warmuth [25] introduced and studied a notion of reducibility among learning problems. They showed that for every polynomial p there is a polynomial p_1 such that if the class $ADFA_{p_1(n)}$ (deterministic finite automata accepting only strings of length $p_1(n)$) is (weakly) learnable, then the class $BF_{n,p(n)}$ is (weakly) learnable. Combining this with Theorem 4, we have:

Theorem 5 *For n large enough, and for some polynomial p, the problems of recognizing quadratic residues, inverting the RSA encryption function and factoring Blum integers are probabilistic polynomial-time reducible to weakly learning $ADFA_{n,p(n)}$, the class of deterministic finite automata of size at most $p(n)$ accepting only strings of length n.*

Using results of [13], [6] and [29], it can be shown that the representations described in Section 3 can each be computed by a polynomial-size, constant-depth threshold circuit. Thus we have:

Theorem 6 *For n large enough, and for some constant d and polynomial p, the problems of recognizing quadratic residues, inverting the RSA encryption function and factoring Blum integers are probabilistic polynomial-time reducible to weakly learning $CDTC^d_{n,p(n)}$, the class of threshold circuits over n inputs with depth at most d and size at most $p(n)$.*

It is important to note that these hardness results hold regardless of the hypothesis representation class of the learning algorithm; that is, Boolean formulae, DFA's and constant-depth threshold circuits are not weakly learnable by *any* polynomially evaluatable representation class (under standard cryptographic assumptions). We note that no NP-hardness results are known for these classes even if we restrict hypothesis representation to the classes themselves and insist on learnability rather than weak learnability. It is also possible to give reductions showing that many other interesting classes (e.g., CFG's and NFA's) are not weakly learnable (again under cryptographic assumptions). In general, any representation class whose computational power subsumes that of NC^1 is not weakly learnable; however, more subtle reductions are also possible. In particular, our results partially resolve a problem posed in [25] by showing that under cryptographic assumptions, the class of all languages accepted by logspace Turing machines is not weakly learnable.

5 A Generalized Construction Based on Any Trapdoor Function

Let us now give a brief overview of the techniques that were used in Sections 3 and 4 to obtain hardness results for learning based on cryptographic assumptions. In each construction (RSA, quadratic residue and factoring Blum integers), we begin with a candidate trapdoor function. We then construct a learning problem in which inverting the function given the trapdoor key corresponds to evaluating the representations being learned, and learning from random examples corresponds to inverting the function without the trapdoor key. To prove hardness results for the simplest possible representation classes, we then ease the computation of the inverse given the trapdoor key by providing the powers of the original input in each example. This additional information provably does not compromise the security of the original function. A key property of trapdoor functions exploited by our constructions is the ability to generate random examples of the target representation without the trapdoor key; this corresponds to the ability to generate encrypted messages given only the public key in a public-key cryptosystem.

By assuming that specific functions are trapdoor functions, we were able to find modified trapdoor functions whose inverse computation could be performed by very simple circuits given the trapdoor. This allowed us to prove hardness results for specific representation classes that are of interest in computational learning theory. However, the above overview suggests a general method for proving hardness results for learning: to show that a representation class C is not learnable, find a trapdoor function whose inverse can be computed by C given the trapdoor key. In this section we prove a theorem demonstrating that this is in fact a viable approach.

We use the following definition for a family of trap-

door functions, which can be derived from [34]: let $P = \{P_n\}$ be a family of probability distributions, where for $n \geq 1$ the distribution P_n is over pairs $(k, k') \in \{0,1\}^n \times \{0,1\}^n$. We think of k as the n-bit *public key* and k' as the associated n-bit *private key*. Then $Q = \{Q_k\}$ is a family of probability distributions parameterized by the public key k, where if $|k| = n$ then Q_k is a distribution over $\{0,1\}^n$. We think of Q as a distribution family over the *message space*. Then the function $f(k, x)$ maps an n-bit public key k and an n-bit *cleartext message* x to the *ciphertext* $f(k, x)$. We call the triple (P, Q, f) an *α-strong trapdoor scheme* if it has the following properties:

(i) There is probabilistic polynomial-time algorithm G (the *key generator*) that on input 1^n outputs a pair (k, k') according to the distribution P_n. Thus, pairs of public and private keys are easily generated.
(ii) There is a probabilistic polynomial-time algorithm M (the *message generator*) that on input k outputs x according to the distribution Q_k. Thus, messages are easily generated given the public key k.
(iii) There is a polynomial-time algorithm E that on input k and x outputs $f(k, x)$. Thus, encryption is easy.
(iv) Let A be any probabilistic polynomial-time algorithm. Perform the following experiment: draw a pair (k, k') according to P_n, and draw x according to Q_k. Give the inputs k and $f(k, x)$ to A. Then the probability that $A(k, f(k, x)) \neq x$ is at least α. Thus, decryption from only the public key and the ciphertext is hard.
(v) There is a polynomial-time algorithm D that on input k, k' and $f(k, x)$ outputs x. Thus, decryption given the private key (or *trapdoor*) is easy.

As an example, consider the RSA cryptosystem [30]. Here the distribution P_n is uniform over (k, k') where $k' = (p, q)$ for p and q n-bit primes and $k = (p \cdot q, e)$ with $e \in Z^*_{\varphi(p \cdot q)}$. The distribution Q_k is uniform over $Z_{p \cdot q}$, and $f(k, x) = f((p \cdot q, e), x) = x^e \bmod p \cdot q$.

We now formalize the notion of the inverse of a trapdoor function being computed in a representation class. Let $C = \cup_{n \geq 1} C_n$ be a paramterized Boolean representation class. We say that a trapdoor scheme (P, Q, f) is *invertible in C given the trapdoor* if for any $n \geq 1$, for any pair of keys $(k, k') \in \{0,1\}^n \times \{0,1\}^n$, and for any $1 \leq i \leq n$, there is a representation $c^i_{(k,k')} \in C_n$ that on input $f(k, x)$ for any $x \in \{0,1\}^n$ outputs the ith bit of x.

Theorem 7 *Let p be any polynomial, and let $\alpha \geq \frac{1}{p(n)}$. Let (P, Q, f) be an α-strong trapdoor scheme, and let C be a parameterized Boolean representation class. Then if (P, Q, f) is invertible in C given the trapdoor, C is not polynomially learnable.*

6 Hardness Results for Approximation Algorithms.

In this section, we use our hardness results for learning to prove that under cryptographic assumptions, certain combinatorial optimization problems, including a natural generalization of graph coloring, cannot be efficiently approximated even in a very weak sense. We begin with a theorem of Blumer et al. [10] that has become known as *Occam's Razor*; informally, it says that finding any hypothesis consistent with a large enough random sample is as good as learning.

Theorem 8 [10] *Let C and H be polynomially evaluatable parameterized Boolean representation classes. Fix $\alpha \geq 1$ and $0 \leq \beta < 1$, and let A be an algorithm that on input a sample S for some $c \in C_n$, consisting of $\frac{m}{2}$ positive examples of c drawn from D^+ and $\frac{m}{2}$ negative examples of c drawn from D^-, outputs an hypothesis $h_A \in H_n$ that is consistent with S and satisfies $|h_A| \leq n^\alpha m^\beta$. Then A is a learning algorithm for C by H that learns with accuracy ϵ and confidence δ if*

$$m = \Omega\left(\max\left(\frac{1}{\epsilon}\log\frac{1}{\delta}, \left(\frac{n^\alpha}{\epsilon}\log\frac{n^\alpha}{\epsilon}\right)^{\frac{1}{1-\beta}}\right)\right).$$

Let $|S| = mn$ denote the number of bits in the sample S. Note that if A instead outputs h_A satisfying $|h_A| \leq n^{\alpha'}|S|^\beta$ for some fixed $\alpha' \geq 1$ and $0 \leq \beta < 1$ then $|h_A| \leq n^{\alpha'}(mn)^\beta = n^{\alpha'+\beta}m^\beta$, so A satisfies the conditon of Theorem 8 for $\alpha = \alpha' + \beta$.

Let C and H be polynomially evaluatable paramterized Boolean representation classes, and define the *Consistency Problem* $Con(C, H)$ as follows:
Input: A sample S of some $c \in C_n$.
Output: $h \in H_n$ such that h is consistent with S and $|h|$ is minimized.

We use $h_{opt}(S) \in H_n$ to denote an hypothesis that achieves the minimum size for an instance S of $Con(C, H)$. Using the results of Section 4 and Theorem 8, we immediately obtain proofs of the following theorems.

Theorem 9 *Let H be any polynomially evaluatable parameterized Boolean representation class. Then the problems of recognizing quadratic residues, inverting the RSA encryption function and factoring Blum integers are probabilistic polynomial-time reducible to approximating the optimal solution of an instance S of*

$Con(BF, H)$ by a hypothesis h satisfying

$$|h| \leq |h_{opt}(S)|^{\alpha}|S|^{\beta}$$

for any $\alpha \geq 1$ and $0 \leq \beta < 1$.

Theorem 10 *As Theorem 9 but with ADFA replacing BF.*

Theorem 11 *As Theorem 9 but with $CDTC^d$ for some constant d replacing BF.*

These theorems demonstrate that the results of Section 4 are not dependent upon the particular models of learnability that we study; we are able to restate the hardness of learning in terms standard combinatorial optimization problems. Using a generalization of Theorem 8, we can in fact prove Theorems 9, 10 and 11 for the *Relaxed Consistency Problem*, where the hypothesis found must agree with only a fraction $\frac{1}{2} + \frac{1}{p(|h_{opt}(S)|,n)}$ of the sample for any fixed polynomial p. Using the results of [17], it is also possible to show similar hardness results for the Boolean circuit consistency problem using the weaker assumption that there exists a one-way function.

Note that Theorem 11 addresses the optimization problem $Con(CDTC^d, CDTC)$ as a special case. This problem is essentially that of finding a set of weights in a neural network that yields the desired input-output behavior. Theorem 11 states that even if we allow a much larger net than is actually required, finding these weights is computationally intractable, even for only a constant number of "hidden layers". This result should be contrasted with those of [20] and [9], which rely on the weaker assumption $P \neq NP$ but do not prove hardness for relaxed consistency and do not allow the hypothesis network to be substantially larger than the smallest consistent network.

Theorems 9, 10 and 11 are interesting for at least two reasons. First, they suggest that it is possible to obtain strong hardness results for combinatorial optimization approximation algorithms by using stronger complexity-theoretic assumptions. Such results seem difficult to obtain using only the assumption $P \neq NP$. In fact, perhaps the most striking such result known is that of Pitt and Warmuth [26] who showed that it is NP-hard to approximate $Con(DFA, DFA)$ to within a polynomial. Second, these results provide us with natural examples of optimization problems for which it is hard to approximate the optimal solution even within a factor that grows as a function of the instance size. Several well-studied problems seem to have this property, but little has been proven in this direction. Perhaps the best example of this phenomenon is graph coloring, where the best polynomial-time algorithms require approximately $n^{1-\frac{1}{k-1}}$ colors on k-colorable n-vertex graphs (see [33], [8]) but coloring has been proven NP-hard only for $(2-\epsilon)k$ colors for any $\epsilon > 0$ (see [16]). This leads us to look for approximation-preserving reductions from our provably hard optimization problems to other natural problems.

We now define a class of optimization problems that we call *formula coloring* problems. Here we have variables $v_1, \ldots, v_m$ assuming values from a set of *colors*. We regard an assignment of colors to the v_i as a partition P of the variable set into equivalence classes; thus two variables have the same color if and only if they are in the same equivalence class. We consider Boolean formulae that are formed by the standard Boolean connectives over atomic elements of the form $(v_i = v_j)$ and $(v_i \neq v_j)$, where the predicate $(v_i = v_j)$ is satisfied if and only if v_i and v_j are assigned the same color.

A *model* for such a formula $F(v_1, \ldots, v_m)$ is a partition P of the variables $v_1, \ldots, v_m$ such that F is satisfied. A *mimimum model* for the F is a model using the fewest colors. For example, the formula

$$(v_1 = v_2) \vee ((v_1 \neq v_2) \wedge (v_3 \neq v_4))$$

has as a model the two-color partition $\{v_1, v_3\}, \{v_2, v_4\}$ and as a mimimum model the one-color partition $\{v_1, v_2, v_3, v_4\}$.

We will be interested in finding mimimum models for restricted classes of formulae. Let $u_1, \ldots, u_l$ be variables. Then if $S(v_1, \ldots, v_m)$ is a formula as above, an *instance of S over $u_1, \ldots, u_l$* is obtained by replacing each v_i with any u_j. Then if $S_1, \ldots, S_l$ are formulae over $v_1, \ldots, v_m$, we define a parameterized *Minimum Model Problem* $MM(S_1; \cdots; S_l)$ as follows:
Input: A formula $F(u_1, \ldots, u_l)$ of the form

$$I_1(u_1, \ldots, u_l) \wedge \cdots \wedge I_r(u_1, \ldots, u_l)$$

where each $I_i(u_1, \ldots, u_l)$ is an instance of some $S_j \in \{S_1, \ldots, S_l\}$ over $u_1, \ldots, u_l$.
Output: A minimum model for F.

We call the S_j *schemata*. Thus we restrict formulae to be conjunctions of instances of the schemata; the expressive power of the allowed input formulae is then determined by the form of the schemata.

For F an instance of a minimum model problem, and P a model of F, we let $\chi(P)$ denote the number of colors in P and $\chi_{opt}(F)$ the number of colors in a minimum model of F. Note that the single-schema problem $MM((v_1 \neq v_2))$ is exactly the *graph coloring* problem: for each edge (u_i, u_j) in G, we conjunct the instance $(u_i \neq u_j)$ of the schema $(v_1 \neq v_2)$. Then χ_{opt} is exactly the number of colors required to color G. Similarly, the single-schema problem

$$MM(((v_1 \neq v_2) \vee (v_1 \neq v_3) \vee (v_2 \neq v_3)))$$

is the *3-hypergraph coloring* problem (each hyperedge contains 3 vertices).

We now consider the 2-schema problem

$$MM(((v_1 \neq v_2) \vee (v_3 = v_4)); (v_1 \neq v_2)).$$

Theorem 12 *There is a polynomial-time algorithm A that on input an instance S of $Con(ADFA, ADFA)$ outputs an instance $F(S)$ of*

$$MM(((v_1 \neq v_2) \vee (v_3 = v_4)); (v_1 \neq v_2))$$

such that S has a k-state consistent hypothesis $M \in ADFA$ if and only if $F(S)$ has a model of k colors.

Note that if $|S|$ is the number of bits in the sample S and $|F(S)|$ denotes the number of bits in the formula $F(S)$, then the proof of Theorem 12 has $|F(S)| = \Theta(|S|^2 \log|S|) = O(|S|^{2+\gamma})$ for any $\gamma > 0$. Thus by Theorems 10 and 12 we have:

Theorem 13 *The problems of recognizing quadratic residues, inverting the RSA encryption function and factoring Blum integers are probabilistic polynomial-time reducible to approximating the optimal solution to an instance F of*

$$MM(((v_1 \neq v_2) \vee (v_3 = v_4)); (v_1 \neq v_2))$$

by a model P of F satisfying

$$\chi(P) \leq \chi_{opt}(F)^{\alpha} |F|^{\beta}$$

for any $\alpha \geq 1$ and $0 \leq \beta < \frac{1}{2}$.

7 Weakly Learning Any Monotone Function Under Uniform Distributions

For $T \subseteq \{0,1\}^n$ and $\vec{u}, \vec{v} \in \{0,1\}^n$ define

$$\vec{u} \oplus \vec{v} = (u_1 \oplus v_1, \ldots, u_n \oplus v_n)$$

and $T \oplus \vec{v} = \{\vec{u} \oplus \vec{v} : \vec{u} \in T\}$, where $\oplus$ denotes exclusive-or. For $1 \leq i \leq n$ let $\vec{e}_i$ be the vector with the ith bit set to 1 and all other bits set to 0.

The following lemma is from [2].

Lemma 14 *Let $T \subset \{0,1\}^n$ be such that $|T| \leq \frac{2^n}{2}$. Then for some $1 \leq i \leq n$,*

$$|T \oplus \vec{e}_i - T| \geq \frac{|T|}{2n}.$$

Theorem 15 *The class of monotone Boolean functions is polynomially weakly learnable under uniform D^+ and uniform D^-.*

Proof: Let f be any monotone Boolean function on $\{0,1\}^n$. First assume that $|pos(f)| \leq \frac{2^n}{2}$. For $\vec{v} \in \{0,1\}^n$ and $1 \leq i \leq n$, let $\vec{v}_{i=b}$ denote $\vec{v}$ with the ith bit set to $b \in \{0,1\}$, and v_i denote the i^{th} bit of v.

Now suppose that $\vec{v} \in \{0,1\}^n$ is such that $\vec{v} \in neg(f)$ and $v_j = 1$ for some j $1 \leq j \leq n$. Then $\vec{v}_{j=0} \in neg(f)$ by monotonicity of f. Thus for any $1 \leq j \leq n$ we must have

$$(1) \quad \mathbf{Pr}_{\vec{v} \in D^-}(v_j = 1) \leq \frac{1}{2}$$

since D^- is uniform over $neg(f)$.

Let $\vec{e}_i$ be the vector satisfying

$$|pos(f) \oplus \vec{e}_i - pos(f)| \geq \frac{|pos(f)|}{2n}$$

in Lemma 14 above. Let $\vec{v} \in \{0,1\}^n$ be such that $\vec{v} \in pos(f)$ and $v_i = 0$. Then $\vec{v}_{i=1} \in pos(f)$ again by monotonicity of f. However, by Lemma 14, the number of $\vec{v} \in pos(f)$ such that $v_i = 1$ and $\vec{v}_{i=0} \in neg(f)$ is at least $\frac{|pos(f)|}{2n}$. Thus, we have

$$(2) \quad \mathbf{Pr}_{\vec{v} \in D^+}(v_i = 1) \geq \frac{1}{2} + \frac{1}{4n}.$$

Similarly, if $|neg(f)| \leq \frac{2^n}{2}$, then for any $1 \leq j \leq n$ we must have

$$(3) \quad \mathbf{Pr}_{\vec{v} \in D^+}(v_j = 0) \leq \frac{1}{2}$$

and for some $1 \leq i \leq n$,

$$(4) \quad \mathbf{Pr}_{\vec{v} \in D^-}(v_i = 0) \geq \frac{1}{2} + \frac{1}{4n}.$$

Note that either $|pos(f)| \leq \frac{2^n}{2}$ or $|neg(f)| \leq \frac{2^n}{2}$.

We use these differences in probabilities to construct a polynomial-time weak learning algorithm A. A first assumes $|pos(f)| \leq \frac{2^n}{2}$; if this is the case, then (1) and (2) must hold. A then finds an index $1 \leq i \leq n$ satisfying

$$(5) \quad \mathbf{Pr}_{\vec{v} \in D^+}(v_i = 1) \geq \frac{1}{2} + \frac{1}{8n}.$$

This can be done with high probability in polynomial time by applying Chernoff bounds to obtain an estimate p such that

$$\mathbf{Pr}_{\vec{v} \in D^+}(v_i = 1) - \frac{1}{8n} < p < \mathbf{Pr}_{\vec{v} \in D^+}(v_i = 1) + \frac{1}{8n}.$$

If A successfully identifies an index i satisfying (5), then the hypothesis h_A is defined as follows: given an unlabeled input vector $\vec{v}$, h_A flips a biased coin and with probability $\frac{1}{16n}$ classifies $\vec{v}$ as negative. With

probability $1-\frac{1}{16n}$, h_A classifies $\vec{v}$ as positive if $v_i = 1$ and as negative if $v_i = 0$. It is easy to verify by (1) and (5) that this is a randomized hypothesis meeting the conditions of weak learnability.

If A is unable to identify an index i satisfying (5), then A assumes that $|neg(f)| \leq \frac{2^n}{2}$, and in a similar fashion proceeds to form a hypothesis h_A based on the differences in probability of (3) and (4). □

It follows from [15] that the class of monotone Boolean functions is not polynomially weakly learnable under arbitrary target distributions. The recent results of [14] show that the class of monotone Boolean functions is not polynomially learnable under uniform target distributions. The results of [15] can also be used to show that the class of all Boolean functions is not polynomially weakly learnable under uniform target distributions. Thus, Theorem 15 is optimal in the sense that generalization in any direction — uniform distributions to arbitrary distributions, weak learning to learning, or monotone functions to arbitrary functions — results in intractability.

8 Equivalence of Weak Learning and Group Learning

We now demonstrate that the notion of weak learnability is equivalent to a notion we call *group learnability*. Group learning is like learning except that the requirement that future single examples have to be correctly classified is relaxed so that only a group of them that are guaranteed all to be positive or all to be negative needs to be classified. It is sufficient that this classification be feasible with accuracy ϵ and confidence δ for groups of size $p(\frac{1}{\epsilon}, \frac{1}{\delta}, |c|, n)$ for some polynomial p.

Theorem 16 *Let C be a class of polynomial evaluatable parametrized Boolean representations. Then C is (polynomially) group learnable if and only if C is (polynomially) weakly learnable.*

The proof of Theorem 16 holds even under specific distributions. Thus C is group learnable for a certain distribution if and only if it is weakly learnable under the same distribution. Hence, for example, Theorem 15 also implies that monotone functions are polynomially group learnable under uniform distributions.

Recently R. Schapire has shown the surprising result that in the distribution-free setting polynomial weak learnability is the same as polynomial learnability. This taken together with our last result illustrates the robustness of the notion of learnability.

References

[1] L. Adleman, K. Manders, G. Miller. On taking roots in finite fields. *Proc. 18th IEEE Symp. on Foundations of Computer Science*, 1977, pp. 175–178.

[2] D. Aldous. On the Markov chain simulation method for uniform combinatorial distributions and simulated annealing. U.C. Berkeley Statistics Department, technical report 60, 1986.

[3] W. Alexi, B. Chor, O. Goldreich, C.P. Schnorr. RSA and Rabin functions: certain parts are as hard as the whole. *SIAM J. on Computing*, 17(2) 1988, pp. 194–209.

[4] D. Angluin. Lecture notes on the complexity of some problems in number theory. Yale University Computer Science Department, technical report TR-243, 1982.

[5] D. Angluin. Learning regular sets from queries and counterexamples. *Inf. and Computation*, 75, 1987, pp. 87-106.

[6] P.W. Beame, S.A. Cook, H.J. Hoover. Log depth circuits for division and related problems. *SIAM J. on Computing*, 15 (4), 1986, pp. 994-1003.

[7] G.M. Benedek, A. Itai. Learnability by fixed distributions. *Proc. of the 1988 Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers, 1988, pp. 80-90.

[8] A. Blum. An $O(n^{0.4})$-approximation algorithm for 3-coloring. Proc. 21st ACM Symp. on Theory of Computing, 1989.

[9] A. Blum, R.L. Rivest. Training a 3-node neural network is NP-complete. *Proc. of the 1988 Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers, 1988, pp. 9-18.

[10] A. Blumer, A. Ehrenfeucht, D. Haussler, M. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. *Proc. of the 18th ACM Symp. on Theory of Computing*, 1986, pp. 273-282.

[11] A. Blumer, A. Ehrenfeucht, D. Haussler, M. Warmuth. Occam's razor. *Inf. Proc. Letters*, 24 1987, pp. 377-380.

[12] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23, 1952, pp. 493–509.

[13] A.K. Chandra, L.J. Stockmeyer and U. Vishkin. Constant depth reducibility. *SIAM J. on Computing* 13 (2), 1984 pp. 423-432.

[14] A. Ehrenfeucht, D. Haussler. Personal communication.

[15] A. Ehrenfeucht, D. Haussler, M. Kearns. L.G. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, to appear. Also in *Proc. of the 1988 Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers, 1988, pp. 139-154.

[16] M. Garey, D. Johnson. Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco, CA, 1979.

[17] O. Goldreich, S. Goldwasser, S. Micali. How to construct random functions. *J. of the ACM*, 33(4) 1986, pp. 792-807.

[18] D. Haussler, M. Kearns, N. Littlestone, M. Warmuth. Equivalence of models for polynomial learnability. *Proc. of the 1988 Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers, 1988, pp 42-55.

[19] D. Haussler, N. Littlestone, M. Warmuth. Predicting 0,1-functions on randomly drawn points. *Proc. of the 29th IEEE Symp. on Foundations of Computer Science*, 1988, pp. 100–109.

[20] S. Judd. Learning in neural networks. *Proc. of the 1988 Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers, 1988, pp 2–8.

[21] M. Kearns, M. Li, L. Pitt, L.G. Valiant. On the learnability of Boolean formulae. *Proc. of the 19th ACM Symp. on Theory of Computing*, 1987, pp. 285–295.

[22] E. Kranakis. Primality and cryptography. John Wiley and Sons, 1986.

[23] L. Levin. One-way functions and pseudorandom generators. *Proc. of the 17th ACM Symp. on Theory of Computing*, 1985, pp. 363–365.

[24] M. Li, U. Vazirani. On the learnability of finite automata. *Proc of the 1988 Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers 1988, pp. 359–370.

[25] L. Pitt, M.K. Warmuth. Reductions among prediction problems: on the difficulty of predicting automata. *Proc. 3d Conference on Structure in Complexity Theory*, 1988, pp. 60–69.

[26] L. Pitt, M.K. Warmuth. The Minimum DFA Consistency Problem Cannot be Approximated Within any Polynomial. Proc. 21st ACM Symp. on Theory of Computing, 1989.

[27] L. Pitt, L.G. Valiant. Computational limitations on learning from examples. *J. of the ACM*, 35(4), 1988, pp. 965-984.

[28] M.O. Rabin. Digital signatures and public key functions as intractable as factorization. M.I.T. Laboratory for Computer Science technical report TM-212, 1979.

[29] J. H. Reif. On threshold circuits and polynomial computation. Proc. 2nd IEEE Conference on Structure in Complexity Theory, 1987.

[30] R. Rivest, A. Shamir, L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Comm. of the ACM*, 21(2) 1978, pp. 120–126.

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

[32] L.G. Valiant. Learning disjunctions of conjunctions. *Proc. 9th International Joint Conference on Artificial Intelligence*, 1985, pp. 560-566.

[33] A. Wigderson. A new approximate graph coloring algorithm. *Proc. 14th ACM Symp. on Theory of Computing*, 1982, pp. 325–329.

[34] A.C. Yao. Theory and application of trapdoor functions. *Proc. 23rd IEEE Symp. on the Foundations of Computer Science*, 1982, 80–91.