Oracles and Queries that are Sufficient for Exact Learning

Nader H. Bshouty* Richard Cleve* Ricard Gavaldà †
Sampath Kannan‡ Christino Tamon*

Abstract

We show that the class of all circuits is exactly learnable in randomized expected polynomial time using weak subset and weak superset queries. This is a consequence of the following result which we consider to be of independent interest: circuits are exactly learnable in randomized expected polynomial time with equivalence queries and the aid of an *NP*-oracle. We also show that circuits are exactly learnable in deterministic polynomial time with equivalence queries and a Σ_3^{p}-oracle. The hypothesis class for the above learning algorithms is the class of circuits of larger—but polynomially related—size. Also, the algorithms can be adapted to learn the class of DNF formulas with hypothesis class consisting of depth-3 $\wedge$-$\vee$-$\wedge$ formulas (by the work of Angluin [A90], this is optimal in the sense that the hypothesis class cannot be reduced to DNF formulas, i.e. depth-2 $\vee$-$\wedge$ formulas).

We also investigate the power of an *NP*-oracle in the context of learning with *membership* queries. We show that there are deterministic learning algorithms that use membership queries and an *NP*-oracle to learn: monotone boolean functions in time polynomial in the DNF size and CNF size of the target formula; and the class of $O(\log n)$-DNF$\cap O(\log n)$-CNF formulas in time polynomial in n. We also show that, with an *NP*-oracle and membership queries, there is a randomized expected polynomial-time algorithm that learns any class that is learnable from membership queries with unlimited computational power.

Using similar techniques, we show the following both for membership and for equivalence queries (when the hypotheses allowed are precisely the concepts in the class): any class learnable with unbounded computational power is learnable in deterministic polynomial time with a Σ_3^{p}-oracle. Furthermore, we identify the combinatorial properties that completely determine learnability in this information-theoretic sense.

Finally we point out a consequence of our result in structural complexity theory showing that if every *NP* set has polynomial-size circuits then the polynomial hierarchy collapses to ZPP^{NP}.

*Department of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4, E-mail: ⟨last name⟩`@cpsc.ucalgary.ca`. Research supported in part by NSERC of Canada.

†Department of Software (LSI), Universitat Politècnica de Catalunya, Pau Gargallo 5, 08028 Barcelona, Spain, E-mail: `gavalda@lsi.upc.es`. Supported in part by ESPRIT project ALCOM II and by DGICYT, project number PB92–0709.

‡University of Arizona, Tucson, AZ 85721, E-mail: `kannan@cs.arizona.edu`. Supported by NSF grant CCR 91-08969

1 Introduction

One of the outstanding open problems in computational learning theory is whether or not the class of DNF formulas is learnable in polynomial time in any "reasonable" learning model. We focus on this problem as well as the (apparently) more difficult problem of learning the class of all boolean circuits.

In the PAC learning model [V84b], it is an easy result that if $P = NP$ then there is an efficient learning algorithm for the class of DNF formulas as well as the class of circuits. As explained by Pitt and Valiant [PV88], the idea behind the algorithm is that, given a suitable (polynomial) number of examples, it suffices to find *any* member of the class being learned that is consistent with all the examples. With the aid of an *NP*-oracle, this can easily be accomplished in polynomial time.

In the models of exact learning with membership and/or equivalence queries [A88], such a result is not as straightforward. For example, if equivalence queries are permitted, merely *finding* a member from the class being learned which is consistent with the results of all queries made so far is not necessarily a good choice for the next equivalence query; exponentially many such queries may be required to find the target concept in some cases. Nevertheless, Gavaldà [G93] recently showed that if $P = NP$ then the class of circuits is learnable in deterministic polynomial time with equivalence queries.[1]

In this report, we show that the class of circuits is learnable in randomized expected polynomial time with equivalence queries and the aid of an *NP*-oracle. We also show that circuits are learnable in deterministic polynomial time with equivalence queries and a Σ_3^{p}-oracle. Note that the above yields an alternative approach to deriving Gavaldà's [G93] result. (We are currently unaware of any *deterministic* learning algorithm that, with equivalence queries and the aid of an *NP*-oracle, can learn even the class of DNF formulas in polynomial time—even if the equivalence queries are allowed to be arbitrary circuits.)

The hypothesis class for the above learning algorithms is the class of circuits of larger—but polynomially related—size. Also, the algorithms can be adapted to learn the class of DNF formulas with hypothesis class consisting of depth-3 $\wedge$-$\vee$-$\wedge$ formulas (by the work of Angluin [A90], this is optimal in the sense that learning DNF using polynomially many samples is impossible when the hypothesis class is DNF formulas, regardless of the computational power of the learner).

More in general, we also study how much a powerful oracle can help in learning from queries. Take any class that is learnable from a polynomial number of equivalence queries and unlimited computing power. We show that, when the hypotheses are restricted to be from the same class, there is an algorithm that learns in polynomial time using equivalence queries and a Σ_3^{p}-oracle. Furthermore, there is a known combinatorial property that completely characterizes learnability in this sense: the "approximate fingerprints" in [A90]. Angluin showed that the negation of this property is necessary for a class to be learnable with

[1]When we say that a class is learnable without mentioning a hypothesis class, we mean that target class and hypotheses class are the same; otherwise, we explicitly mention which class of hypotheses is used. By "polynomial time" learning we mean that the running time is polynomial in the number of variables and the size of the shortest representation in the class for the target function. Precise definitions are given in Section 2.

equivalence queries and unlimited power; here we show that it is also sufficient.

Learning models where the learner has an *NP*-oracle (or Σ_3^p-oracle) are admittedly quite generous, so some motivation for this work is in order. Many of the queries that have been proposed in learning theory are also quite powerful in the sense that *implementing* them is computationally difficult. For example, when the concept class is DNF formulas, implementing an equivalence query, given the target formula in hand, is an *NP*-hard problem. Angluin [A88] also introduced the subset query (where a negative counterexample is requested) and the superset query (where a positive counterexample is requested). We show that if both subset and superset queries are available then they can be used to simulate an *NP*-oracle as well as equivalence queries. In fact we require only the weak variants of these queries whereby the learner need only be told if there is a counterexample without actually being given one. Therefore, from the above result, it follows that the class of circuits, as well as the subclass of DNF formulas, is learnable in expected polynomial time from weak subset and weak superset queries alone. In this case, the hypothesis class for DNF formulas is that of depth-3 formulas, but both $\wedge$-$\vee$-$\wedge$ and $\vee$-$\wedge$-$\vee$ forms are used. (*Note that these latter algorithms do not use any oracles in addition to their weak subset and weak superset queries.*)

Next we consider learning problems where the learner can make membership queries (but no equivalence queries) and has access to an *NP*-oracle. We show that (in the terminology of [B93]) the number of membership queries required to learn a boolean function is bounded by a polynomial in its monotone dimension, dual monotone dimension, DNF size, and CNF size. Two corollaries of this are that with an *NP*-oracle and membership queries: monotone functions are learnable in time polynomial in their DNF size and CNF size; and n-variable $O(\log n)$-DNF$\cap O(\log n)$-CNF (which generalizes depth $O(\log n)$ decision trees[2]) is learnable in time polynomial in n. We also show some lower bounds on the number of membership queries required (regardless of the computational power).

Then, we show that any class of functions that is learnable from membership queries with unlimited computational power is learnable from membership queries with an *NP*-oracle in randomized expected polynomial time, and with a Σ_3^p-oracle in deterministic polynomial time. As in the case of equivalence queries, we also identify the combinatorial property that determines learnability from membership queries and unlimited computing power.

Our approach in showing that any class learnable from membership queries is learnable from membership queries in randomized expected polynomial time with an *NP*-oracle is based on the following observation. If a class is learnable from membership queries then, at each stage during the learning process, there must be an assignment that is not zero or one on most of the functions consistent with the data known so far. We then can probabilistically estimate two threshold functions, and show that with high probability a point that is distinct on the two thresholds is a "good" assignment that eliminates a large portion of the functions. The probabilistic estimation of two threshold functions employs the approximate uniform generation of Jerrum *et al.* [JVV86].

Finally, we show an application of our results to structural complexity theory. Watanabe [W94] has observed that our results about learning with equivalence queries and an *NP*-oracle imply the following complexity theoretic result: if every *NP* set has polynomial-size circuits

[2]Kushilevitz and Mansour [KM91] have shown that depth $O(\log n)$ decision trees are exactly learnable with high probability using membership queries.

then the polynomial hierarchy collapses to ZPP^{NP}. (This improves the previous result of Karp and Lipton [KL80] where Σ_2^{p} appears in place of ZPP^{NP}.)

Let us comment on a particular technique used in this paper. For our results concerning learning with equivalence queries and an *NP*-oracle, our approach builds on the investigation of the query complexity of learning by Kannan in [K93]. Kannan shows that, for circuits, there *exist* "good" equivalence queries, whose responses are guaranteed to eliminate at least a polynomially small (i.e. $(\frac{1}{n})^{O(1)}$) fraction of the remaining concepts[3]. *Finding* these good equivalence queries may not be feasible (in polynomial time), although this can be done in polynomial *space* by an exhaustive search. Using the results of Jerrum *et al.* [JVV86] about "approximate uniform generation" we show how to construct these good equivalence queries with high probability with the aid of an *NP*-oracle.

Goldman *et al.* [GRS93] also employ results about "approximate uniform generation"—and a closely-related problem, "approximate counting"—in a different learning context: learning a total order on a set in time polynomial in the size of the set. This can always be accomplished with a quadratic number of membership queries, but, by an approximate counting technique, Goldman *et al.* reduce the "number of mistakes" in the on-line model [L88] to $O(n \log n)$. It is well known [L88] that learning in the equivalence query model implies learning in the on-line model. Our equivalence query algorithms depart slightly from the standard model in that they use *NP* or Σ_3^p oracles, but they all carry over to the on-line model by giving the on-line algorithms access to the same oracles.

2 The Learning Model

We consider learning scenarios where the concept class C consists of a set of boolean functions of the form $f : \{0,1\}^n \to \{0,1\}$. When n is fixed, we say that C is over $\{0,1\}^n$. Associated to the class there is a representation method for the functions, that we will make clear in each case. Specific representations that we focus on are DNF formulas and boolean circuits. Sometimes we limit the class to those functions having some representation of size at most s — for example, DNF formulas with n inputs and size at most s. In this case, we call the resulting class *s-bounded.*

In the exact learning model the "learner" asks the "teacher" (oracles) certain types of questions (queries) about the target function f. The goal of the exact learning algorithm is to halt after time polynomial in n and the size of the shortest representation for f in the class, and output a representation h that is logically equivalent to f. By "size", in general we mean bit-size, although for formulas and circuits we implicitly use the closely related measure given by the number of gates.

Consequently, the running time allowed to the learning algorithm depends not only on f itself but on the representation method that is chosen. For example, DNF formulas for a function may be exponentially larger than the smallest circuit computing it. Hence, if we insist on expressing f as a DNF formula we may be allowing more time for learning than if we are looking for a small circuit for f.

In this paper we will study the following types of queries. In an *equivalence query*, the

[3]The algorithms in [K93] also use membership queries; however, these can be removed from the algorithms.

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

In a *membership query*, the learning algorithm supplies an assignment b as input to a *membership oracle*, and the reply of the oracle is the value of f at b, i.e. $f(b)$.

In a *superset query* (*subset query*), the learning algorithm supplies any function h as input to a *superset oracle* (*subset oracle*), and the reply of the oracle is either "YES", signifying that $f \Rightarrow h$ ($h \Rightarrow f$), or a *counterexample*, which is an assignment b such that $h(b) \neq f(b) = 1$ ($h(b) \neq f(b) = 0$). The *weak superset query* (*weak subset query*) is a superset query (subset query) that does not return a counterexample.

Note that, initially, the learner has no knowledge about f other than its membership to the target class. Learning must succeed against any valid choice of counterexamples by the teacher.

Unless stated otherwise, functions used as inputs to Equivalence, Subset, and Superset queries are represented in the same way that the class that we are trying to learn. Sometimes, learning may be too hard under this restriction. Then we may allow queries taken from a larger hypothesis class, or expressed according to its representation method. When this occurs the hypothesis class will always be explicitly stated.

To express our results, we will mention the following complexity classes. The classes in the polynomial hierarchy $\{ \Sigma_k^{\mathrm{p}}, \Pi_k^{\mathrm{p}} \}_k \geq 0$ are defined inductively as follows: $\Sigma_0^{\mathrm{p}} = \Pi_0^{\mathrm{p}} = P$, $\Sigma_{k+1}^{\mathrm{p}}$ is the class of all languages accepted by nondeterministic polynomial-time machines that query an oracle in Σ_k^{p}, and Π_k^{p} is the class of languages whose complements are in Σ_k^{p}. Note in particular that $\Sigma_1^{\mathrm{p}} = NP$.

Also, a ZPP machine is a randomized polynomial-time machine with the following property: on every input, either it accepts with probability at least 3/4 (and halts without output with probability $< 1/4$), or it rejects with probability at least 3/4 (and halts without output with probability $< 1/4$). The class ZPP is the subclass of NP languages accepted by ZPP machines. The class ZPP^{NP} is defined analogously by allowing ZPP machines to query NP oracles.

3 Learning with Equivalence Queries and an *NP*-Oracle

In this section, we describe a new randomized technique for learning in expected polynomial time with the aid of an *NP*-oracle and equivalence queries. The main result of this section is Lemma 5, and Theorem 7 highlights two specific consequences of the result: namely, algorithms that, with the aid of an *NP*-oracle, can accomplish the following learning tasks in probabilistic polynomial time:

- DNF formulas of size s using equivalence queries with hypotheses that are depth-3 $\wedge$-$\vee$-$\wedge$ formulas of size $O(s\, n^2/\log^2 n)$.
- Boolean circuits of size s using equivalence queries with hypotheses that are circuits of size $O(sn + n \log n)$.

Our algorithms are weak versions of the so-called "halving algorithm", that we describe briefly.

For a set of concepts C over $\{0,1\}^n$, the *majority concept* it the unique concept that contains exactly those elements of $\{0,1\}^n$ which belong to at least one half of the concepts in C. To learn a concept class C, the halving algorithm starts asking an equivalence query with the majority concept of C. If the answer is "YES", it stops. Otherwise, by definition of majority concept, the counterexample received is also a valid counterexample for at least half the concepts in C; therefore, the number of candidates to be the target function is at least halved. In the second round, the halving algorithm makes an equivalence query with the majority concept of only those concepts that were not discarded in the previous round, again shrinking by one half the candidate space. In the worst case, the algorithm has to ask $\log|C|$ queries before the candidate space is reduced to only one element, which must be the target concept.

The halving algorithm may be difficult to implement for several reasons. At some step, the required majority concept may have no representation within our hypothesis class; or all its representations may be exponentially large with respect to the computation time we are allowed; finally, the majority concept may have some short representations but finding any of them may be computationally too expensive. For this reason, we settle for a weaker requirement: instead of discarding half the concepts at every round, we discard some fraction $\delta > 0$, possibly smaller than $1/2$.

Definition: Let C be a concept class and $\delta \in [0, \frac{1}{2}]$. A hypothesis f is δ-*good* for C if any counterexample to an equivalence query of f eliminates at least a fraction δ of the elements of C.

Definition: For a concept class C over $\{0,1\}^n$, $x \in \{0,1\}^n$, and $b \in \{0,1\}$ define $C_{(x,b)}$ to be the concepts from C for which $f(x) = b$. More formally $C_{(x,b)} = \{f \in C | f(x) = b\}$. This definition can be extended to a collection of labeled examples $I \subseteq \{0,1\}^n \times \{0,1\}$, i.e. we can define $C_I = \{f \in C : (\forall (x,b) \in I) f(x) = b\}$. So C_I is the set of concepts in C that properly classify all examples in the labelled example set I. Also let $\gamma^C_{(x,b)} = |C_{(x,b)}|/|C|$ and $\gamma^C_x = \min(\gamma^C_{(x,0)}, \gamma^C_{(x,1)})$. So $\gamma^C_{(x,b)}$ is the fraction of C that classifies example x with label b.

A weak version of the halving algorithm repeatedly queries a hypothesis that is δ-good for the set of concepts not discarded before. Starting with concept class C, after i queries the number of concepts left is at most $(1-\delta)^i|C|$, so at most $\ln|C|/\ln(\frac{1}{1-\delta}) \leq \frac{1}{\delta}\ln|C|$ equivalence queries are required to isolate the target concept.

In order to generate δ-good hypotheses, we use *amplifiers*. The concept of amplification was first investigated by Moore and Shannon [MS56], Valiant [V84a], and Boppana [B89].

Definition: Let $0 \leq p' < p < q < q' \leq 1$. A (boolean) function $G(y_1, \ldots, y_m)$ is a $(p,q) \to (p',q')$ *amplifier* if:

(a) When $y_1, \ldots, y_m$ are each independently set to 1 with probability *at least* q, $\Pr[G(y_1, \ldots, y_m) = 1] \geq q'$;

(b) When $y_1, \ldots, y_m$ are each independently set to 1 with probability *at most* p, $\Pr[G(y_1, \ldots, y_m) = 1] \le p'$.

The use of amplification functions in learning theory was considered by Goldman, Kearns, and Schapire [GKS93]. Then Kannan [K93] observed connections between amplification and equivalence queries in learning algorithms. In [K93] there is an exponential-time algorithm that learns boolean circuits and DNF formulas using both membership queries and equivalence queries of polynomial size. In fact, equivalence queries alone suffice by the following lemma.

Lemma 1: *Let $G(y_1, \ldots, y_m)$ be a $(\delta, 1-\delta) \to (2^{-2n}, 1-2^{-2n})$ amplifier. Let C be a concept class over $\{0,1\}^n$ and $f_1, \ldots, f_m$ be functions selected from C independently and uniformly at random. Then, with probability at least $1 - 2^{-n}$, $G(f_1, \ldots, f_m)$ is δ-good for C.*

Proof: Note that, if $\delta \le \gamma^C_{(x,1)} \le 1 - \delta$ then, if x is returned as a counterexample to any equivalence query, a δ-fraction of the elements of C are guaranteed to be eliminated.

Now, let x be any value for which $\gamma^C_{(x,1)} < \delta$. Then, if x is returned as a counterexample to some f_i for which $f_i(x) = 1$, less than a δ-fraction of the elements of C will be eliminated; otherwise, more than a δ-fraction. For a $f_i \in C$ chosen uniformly at random, $\Pr[f_i(x) = 1] < \delta$. Therefore, since $G(y_1, \ldots, y_m)$ is a $(\delta, 1-\delta) \to (2^{-2n}, 1-2^{-2n})$ amplifier,

$$\Pr[G(f_1, \ldots, f_m)(x) = 1] < 2^{-2n}.$$

Thus, the probability that less than a δ-fraction of the elements of C are eliminated when x is returned as a counterexample is $< 2^{-2n}$.

A similar argument applies for any x such that $\gamma^C_{(x,1)} > 1 - \delta$.

Therefore, the probability that there exists an $x \in \{0,1\}^n$ which, when returned as a counterexample to the equivalence query $G(f_1, \ldots, f_m)$ eliminates less than a δ-fraction of the elements of C, is less than $2^n \cdot 2^{-2n} = 2^{-n}$.$\square$

Lemma 2 [B89, K93]:

(a) *The function MAJORITY$(y_1, \ldots, y_{48n})$ is a $(\frac{1}{4}, \frac{3}{4}) \to (2^{-2n}, 1-2^{-2n})$ amplifier.*

(b) *Define $A(y_1, \ldots, y_m)$ as a $(2n/\log n)$-ary $\wedge$ of $(2n/\log n)$-ary $\vee$s of distinct variables. (Thus, the number of inputs to the formula is $m = \frac{4n^2}{\log^2 n}$.) Then $A(y_1, \ldots, y_m)$ is a $(\frac{1}{n^2}, 1-\frac{1}{n^2}) \to (2^{-2n}, 1-2^{-2n})$ amplifier.*

Statement (b) above is a sharpening of a similar result in [K93] but the proof follows along similar lines described below.

Proof: We will use Chernoff bounds on the tails of distributions to prove the above lemma. The following version of the Chernoff bound is taken from Raghavan [R90]. Let $X_1, X_2, \ldots, X_n$ be independent Bernoulli trials with $\Pr[X_i = 1] = p_i, p_i \in (0,1)$. Let $X = \sum_{i=1}^n X_i$ and $\mu = \sum_{i=1}^n p_i$. Then for $\delta > 0$

$$Pr(X > (1+\delta)\mu) < \left[\frac{\exp(\delta)}{(1+\delta)^{(1+\delta)}}\right]^{\mu} = F^+(\mu, \delta). \tag{1}$$

Under the same hypothesis as above, for $\delta \in (0,1]$,

$$Pr(X < (1-\delta)\mu) < \exp(-\mu\delta^2/2) = F^-(\mu,\delta). \tag{2}$$

To prove statement **(a)** note that if each $48n$ random variables are chosen with each $p_i = 1/4$, then the probability that the sum of the random variables exceeds $24n$ is given by $F^+(12n, 1) = (e/4)^{12n} < 2^{-2n}$. Similarly, if $48n$ random variables are chosen each with $p_i = 3/4$, then the probability that the sum of the random variables falls below $24n$ is given by $F^-(36n, 1/3) = e^{-2n} < 2^{-2n}$.

To prove statement **(b)** note that if $p_i \leq \frac{1}{n^2}$ then the probability that any particular $\vee$-gate will compute a 1 is upper-bounded by $(1/n^2)\cdot(2n/\log n) \leq 2/n\log n$, by the union bound. The probability that all of the $\vee$-gates will compute a 1 (and hence the circuit will compute a 1) is upper-bounded by $(2/n\log n)^{2n/\log n}$ which is upper-bounded by 2^{-2n}. If $p_i \geq 1 - \frac{1}{n^2}$ the probability that a particular $\vee$-gate will not compute a 1 is upper-bounded by $(1/n^2)^{2n/\log n} = 2^{-4n}$, and the probability that some $\vee$-gate will not compute a 1 is upper-bounded by $(2n/\log n)2^{-4n}$ which is no more than 2^{-2n}. $\Box$

From the above results, and noting that MAJORITY$(y_1,\ldots,y_{48n})$ is computable by a circuit of size $O(n\log n)$, and $A(f_1,\ldots,f_m)$ is a depth-3 $\wedge$-$\vee$-$\wedge$ formula when $f_1,\ldots,f_m$ are DNF formulas, the following can be concluded.

Corollary 3: *Given exponential computing time, the following learning tasks can be accomplished with polynomially many queries:*

(a) *Learning DNF formulas of size s using equivalence queries that are depth-3 $\wedge$-$\vee$-$\wedge$ formulas of size $O(s\,n^2/\log^2 n)$.*

(b) *Learning boolean circuits of size s using equivalence queries that are circuits of size $O(sn + n\log n)$.*

This establishes information-theoretic solutions to two important learning problems. Kannan [K93] notes that the computational difficulty in implementing the above learning algorithms is in uniformly selecting the formulas from C, which is, in general, exponentially large[4]. Of course, polynomial *space* is achievable by an exhaustive search.

In order to implement the above in polynomial time using an *NP*-oracle, we use a result of Jerrum *et al.* [JVV86] concerning approximate uniform generation, which appears as Theorem 4 below.

Definition: Let ρ be a probability distribution on a discrete probability space Ω and $S \subseteq \Omega$. Then ρ is *uniform on S* if, for all $\omega \in \Omega$: $\rho[\omega] = 1/|S|$ if $\omega \in S$; otherwise, $\rho[\omega] = 0$. Also, for $\epsilon \in (0,1]$, ρ is *approximately uniform on S with tolerance ϵ* if, there exists a $c > 0$ such that, for all $\omega \in \Omega$: $(1+\epsilon)^{-1}c \leq \rho[\omega] \leq (1+\epsilon)c$ if $\omega \in S$; otherwise, $\rho[\omega] = 0$.

[4] In fact, the work of Pitt and Valiant [PV88] implies that, when C consists of DNF formulas of size $p(n)$ (consistent with some counterexamples), the sampling step cannot be accomplished in polynomial time unless $P = NP$.

For our particular applications of the following theorem, we will use sets of the form C_I where I is a set of labelled examples and C_I is the set of all concepts that agree with sample I, possibly imposing some size bound s. Then the class $\{C_I\}_I$ is obtained by ranging over all sets I. However, the theorem applies to more general contexts.

Theorem 4 [JVV86]: *Let $\{C_I\}_I$ be an indexed family of s-bounded sets (i.e., all the elements of C_I have length at most s.) Suppose that there is an algorithm that, on input f and I, determines whether or not $f \in C_I$ in time polynomial in $|I|$ and s. Then there exists a probabilistic algorithm that uses an NP-oracle and, on input I and ϵ, runs in time polynomial in $|I|$, s, and $\log \epsilon^{-1}$ and outputs f according to a distribution that is approximately uniform on C_I with tolerance ϵ.*

We cannot apply Theorem 4 directly to select $f_1, \ldots, f_m$ for Lemma 1, because the sampling provided by Theorem 4 is not exactly uniform. The following lemmas imply that approximately uniform sampling suffices.

Lemma 5: *Let C be a concept class over $\{0,1\}^n$, and let U be an approximately uniform generator for C with tolerance ϵ. Let f be the random function output by U. If $x \in \{0,1\}^n$ with $\gamma^C_{(x,b)} \leq \delta$ then the probability that $f(x) = b$ is at most $\delta(1+\epsilon)^2$ for any $\delta \geq 0$ and $b \in \{0,1\}$.*

Proof: Suppose $x \in \{0,1\}^n$ such that $\gamma^C_{(x,b)} \leq \delta$ for some δ and for some b. Then if a function f is chosen uniformly at random from C, $Pr[f(x) = b] \leq \delta$. Since U can at most oversample the functions f such that $f(x) = b$ by a factor of $(1+\epsilon)^2$, if f is the output of U, $Pr[f(x) = b]$ is bounded by $\delta(1+\epsilon)^2$. □

Lemma 6: *Let $G(y_1, \ldots, y_m)$ be a $(\delta, 1-\delta) \rightarrow (2^{-2n}, 1-2^{-2n})$ amplifier. Let C be a concept class and U an approximately uniform generator for C with tolerance ϵ. If $f_1, \ldots, f_m$ are selected independently using U then, with probability at least $1 - 2^{-n}$, $G(f_1, \ldots, f_m)$ is $\delta/(1+\epsilon)^2$-good for C.*

Proof: The proof is immediate from Lemma 5. For any $x \in \{0,1\}^n$ that has $\gamma^C_{(x,0)} \leq \delta/(1+\epsilon)^2$, $Pr[f_i(x) = 0] \leq \delta$. Thus the probability that $G(f_1, \ldots, f_m)(x) = 0$ is at most 2^{-2n}. A similar analysis holds when the '0' is replaced by a '1'. Thus applying Lemma 1 the result follows. □

Theorem 7: *The following learning tasks can be accomplished with probabilistic polynomial-time algorithms that have access to an NP-oracle and make polynomially many queries:*

(a) *Learning DNF formulas of size s using equivalence queries that are depth-3 $\wedge$-$\vee$-$\wedge$ formulas of size $O(sn^2/\log^2 n)$.*

(b) *Learning boolean circuits of size s using equivalence queries that are circuits of size $O(sn + n \log n)$.*

Proof: Take the amplifiers provided by Lemma 2 and apply them to the output of the generator of Theorem 4, with (say) $\epsilon = 1$. By Lemma 6, an equivalence query that is

$\frac{\log n}{4n^2}$-good (for part (a)) and $\frac{1}{16}$-good (for part (b)) is generated with probability $1 - 2^{-n}$. □

In the next corollary we show that the previous results hold under a perhaps much weaker distribution.

Definition: Let ρ be a probability distribution on a discrete probability space Ω and $S \subseteq \Omega$. Then ρ is *q-bounded uniform on S* if, for all $\omega \in \Omega$: $\rho[\omega] \leq q/|S|$ if $\omega \in S$; otherwise, $\rho[\omega] = 0$.

Corollary 8: *Let q be a polynomially bounded function of n. Then the following learning tasks can be accomplished with probabilistic polynomial-time algorithms that have access to a q-bounded uniform distribution generator and make polynomially many queries:*

(a) *Learning DNF formulas of size s using equivalence queries that are depth-3 $\wedge$-$\vee$-$\wedge$ formulas of size $O(sn^2/\log^2 n)$.*

(b) *Learning boolean circuits of size s using equivalence queries that are circuits of size $O(sn + n \log n)$.*

Proof: Let C be the concept class to be learned. Note that a q-bounded uniform distribution can oversample by a factor of at most q. So, as in Theorem 7, take the amplifiers in Lemma 2 and apply them to the output of a q-bounded uniform distribution. By Lemmas 5 and 6, with probability at least $1 - 2^{-n}$, we get an equivalence query that is δ/q-good. So after $m = \frac{q}{\delta} \log |C|$ steps we finish learning with probability at least $1 - m2^{-n}$. □

4 Learning with Subset and Superset Queries Only

In this section we show that there exists a randomized algorithm that learns boolean circuits and DNF formulas from weak subset and weak superset queries only. We do this by showing that equivalence queries and an *NP*-oracle can both be simulated by weak subset and weak superset queries. Then we can appeal to the results from previous sections to claim the stated result.

Lemma 9: *Weak superset and subset queries can simulate an* NP*-oracle.*

Proof: Given a circuit h, it is easy to see that $h \equiv 1$ if and only if the answers to $WeakSuperset(h)$ and $WeakSubset(\overline{h})$ are "YES". □

Lemma 10: *Weak superset and weak subset queries with boolean circuits can simulate equivalence queries.*

Proof: Clearly, an equivalence query can be replaced by a subset query and a superset query. We show that weak subset queries can simulate a given subset query $Subset(h)$. If the answer to h as a weak subset query is "YES", return "YES". Otherwise, find a counterexample as follows. For $x \in \{0,1\}^n$, let h_x be the function $h_x(y) = h(y) \wedge (y < x)$, where $<$ denotes lexicographical ordering. By doing binary search on x, with at most n weak subset queries of the form h_x we can find a counterexample for $Subset(h)$. Note that circuits for h_x are at most $O(n)$ larger than those for h, so the slowdown in the simulation is a small polynomial. The proof for Superset queries is analogous, using $h_x(y) = h(y) \vee (y < x)$. $\Box$

Theorem 11: *The classes of boolean circuits and DNF formulas are learnable in randomized polynomial time from weak superset and weak subset queries with circuits only.*

Proof: Combine the lemmas above with results from the previous sections. $\Box$

For learning DNF, the hypothesis class in this theorem can be chosen to be depth-3 formulas. Queries with hypotheses of the form h_x as used in Lemma 10 can be made depth-3 $\wedge$-$\vee$-$\wedge$: it is easy to express predicate "$y < x$" as $\wedge$-$\vee$-$\wedge$ formulas of size $O(n^2)$ and, for Superset queries, apply distributivity once. Note however that the query $WeakSubset(\overline{h})$ used in Lemma 9 is a depth-3 $\wedge$-$\vee$-$\wedge$ formula.

5 Learning with Equivalence Queries and a Σ_3^p-Oracle

In this section, we prove the following theorem.

Theorem 12: *The following learning tasks can be accomplished with deterministic polynomial-time algorithms that have access to a Σ_3^p-oracle:*

(a) *Learning DNF formulas of size s using equivalence queries that are depth-3 $\wedge$-$\vee$-$\wedge$ formulas of size $O(sn^2/\log^2 n)$.*

(b) *Learning boolean circuits of size s using equivalence queries that are circuits of size $O(sn + n\log n)$.*

Our tools are: Lemma 1, from Section 3; Theorem 13, due to Sipser (who also credits P. Gács) [S83], stated below; and Theorem 14, due to Stockmeyer [S85], stated below.

Theorem 13 [S83]: *Let $\{C_I\}_I$ be an indexed family of s-bounded sets. Suppose that there is an algorithm that, on input f and I, decides whether or not $f \in C_I$ in time polynomial in $|I|$ and s. Then there exists a polynomial p and a predicate P such that, on input I, $a, b \in \{0,1\}^{p(|I|+s)}$, $r \in \{0, ..., 2^s\}$, and $\delta > 0$, $P(I, a, b, r, \delta)$ is computable in time polynomial in $|I|$, s, and $\frac{1}{\delta}$, and such that, for any I, r, and δ:*

- *If $|C_I| \leq (1+\delta)^{-1} r$ then*

$$\left(\forall a \in \{0,1\}^{p(|I|+s)}\right) \left(\exists b \in \{0,1\}^{p(|I|+s)}\right) P(I, a, b, r, \delta)$$

  evaluates to 1 (true).

- *If $|C_I| \geq (1+\delta) r$ then*

$$\left(\forall a \in \{0,1\}^{p(|I|+s)}\right) \left(\exists b \in \{0,1\}^{p(|I|+s)}\right) P(I, a, b, r, \delta)$$

  evaluates to 0 (false).

By using a Π_2^{p}-oracle to make repeated calls to the predicate in the above theorem with different values of $r \in \{0, ..., 2^s\}$, the following can be concluded.

Theorem 14 [S85]: *Let $\{C_I\}_I$ be an indexed family of s-bounded sets. Suppose that there is an algorithm that, on input f and I, decides whether or not $f \in C_I$ in time polynomial in $|I|$ and s. Then there exists a deterministic algorithm that uses a Π_2^{p}-oracle and, on input I and δ, runs in time polynomial in $|I|$, s, and $\frac{1}{\delta}$, and outputs a number $r \in \{0, ..., 2^s\}$ such that*

$$(1+\delta)^{-1}|C_I| \leq r \leq (1+\delta)|C_I|.$$

Lemma 15: *Let $\{C_I\}_I$ be an indexed family of s-bounded concepts such that, for each I, there is an $n_I \leq |I|$ such that C_I consists of concepts of the form $f : \{0,1\}^{n_I} \rightarrow \{0,1\}$. Suppose that there is an algorithm that, on input I and $f : \{0,1\}^{n_I} \rightarrow \{0,1\}$, decides whether or not $f \in C_I$ in time polynomial in $|I|$ and s. Then there exists a polynomial q and a predicate Q such that, on input $I, c, d \in \{0,1\}^{q(|I|+s)}$, $\delta > 0$, $f : \{0,1\}^{n_I} \rightarrow \{0,1\}$, and $r \in \{0, ..., 2^s\}$, $Q(I, c, d, \delta, f, r)$ is computable in time polynomial in $|I|$, s, and $\frac{1}{\delta}$ and such that, if $(1+\frac{1}{16}\delta)^{-1}|C_I| \leq r \leq (1+\frac{1}{16}\delta)|C_I|$ then:*

- *If f is δ-good for C_I then*

$$\left(\forall c \in \{0,1\}^{q(|I|+s)}\right)\left(\exists d \in \{0,1\}^{q(|I|+s)}\right) Q(I, c, d, \delta, f, r)$$

evaluates to 1 (true).

- *If f is not $\frac{\delta}{2}$-good for C_I then*

$$\left(\forall c \in \{0,1\}^{q(|I|+s)}\right)\left(\exists d \in \{0,1\}^{q(|I|+s)}\right) Q(I, c, d, \delta, f, r)$$

evaluates to 0 (false).

Proof: As a technical convenience, assume δ is sufficiently small so that

$$\begin{aligned} 1-\delta &\leq (1+\tfrac{1}{8}\delta)^{-1}(1-\tfrac{3}{4}\delta)(1+\tfrac{1}{16}\delta)^{-1} \\ 1-\tfrac{\delta}{2} &\geq (1+\tfrac{1}{8}\delta)(1-\tfrac{3}{4}\delta)(1+\tfrac{1}{16}\delta). \end{aligned}$$

For each index I and $(x, y) \in \{0,1\}^{n_I} \times \{0,1\}$, let $C_{I \cup \{(x,y)\}} = \{f \in C_I | f(x) = y\}$.

Let P be the predicate in Theorem 13. If f is δ-good for C_I then

$$(\forall x \in \{0,1\}^{n_I})\left(\left|C_{I \cup \{(x, \overline{f(x)})\}}\right| \leq (1-\delta)|C_I|\right)$$

which implies

$$(\forall x \in \{0,1\}^{n_I})\left(\left|C_{I \cup \{(x, \overline{f(x)})\}}\right| \leq (1-\delta)(1+\tfrac{1}{16}\delta)r\right)$$

which implies

$$(\forall x \in \{0,1\}^{n_I})\left(\left|C_{I \cup \{(x, \overline{f(x)})\}}\right| \leq (1+\tfrac{1}{8}\delta)^{-1}(1-\tfrac{3}{4}\delta)r\right)$$

which implies that the statement

$$(\forall x \in \{0,1\}^{n_I}) \left(\forall a \in \{0,1\}^{p(m+s)}\right) \left(\exists b \in \{0,1\}^{p(m+s)}\right) P\left(I \cup \{(x, \overline{f(x)})\}, a, b, \left(1 - \tfrac{3}{4}\delta\right) r, \tfrac{1}{8}\delta\right)$$

evaluates to 1 (where $m = |I \cup \{(x, \overline{f(x)})\}|$).

If f is not $\frac{\delta}{2}$-good for C_I then

$$\neg (\forall x \in \{0,1\}^{n_I}) \left(\left|C_{I \cup \{(x,\overline{f(x)})\}}\right| \leq \left(1 - \tfrac{\delta}{2}\right) |C_I|\right)$$

which implies

$$\neg (\forall x \in \{0,1\}^{n_I}) \left(\left|C_{I \cup \{(x,\overline{f(x)})\}}\right| \leq \left(1 - \tfrac{\delta}{2}\right) (1 + \tfrac{1}{16}\delta)^{-1} r\right)$$

which implies

$$\neg (\forall x \in \{0,1\}^{n_I}) \left(\left|C_{I \cup \{(x,\overline{f(x)})\}}\right| \leq (1 + \tfrac{1}{8}\delta)(1 - \tfrac{3}{4}\delta) r\right)$$

which implies that the statement

$$(\forall x \in \{0,1\}^{n_I}) \left(\forall a \in \{0,1\}^{p(m+s)}\right) \left(\exists b \in \{0,1\}^{p(m+s)}\right) P\left(I \cup \{(x, \overline{f(x)})\}, a, b, \left(1 - \tfrac{3}{4}\delta\right) r, \tfrac{1}{8}\delta\right)$$

evaluates to 0 (where $m = |I \cup \{(x, \overline{f(x)})\}|$). Therefore, it suffices to set

$$Q(I, (x,a), b, \delta, f, r) = P\left(I \cup \{(x, \overline{f(x)})\}, a, b, \left(1 - \tfrac{3}{4}\delta\right) r, \tfrac{1}{8}\delta\right). \square$$

Proof of Theorem 12: By Theorem 14, we efficiently can find an $r \in \{0, ..., 2^s\}$ such that $(1 + \frac{1}{16}\delta)^{-1}|C_I| \leq r \leq (1 + \frac{1}{16}\delta)|C_I|$. By the appropriate application of Lemma 1 and Lemma 2, we know that there exists an f of the appropriate type and size l that is δ-good. Thus, for the predicate in Lemma 15, we have

$$\left(\exists f \in \{0,1\}^l\right) \left(\forall c \in \{0,1\}^{q(|I|+s)}\right) \left(\exists d \in \{0,1\}^{q(|I|+s)}\right) Q(I, c, d, \delta, f, r).$$

Also by Lemma 15, if a particular candidate for f is not at least $\frac{\delta}{2}$-good then

$$\neg \left(\forall c \in \{0,1\}^{q(|I|+s)}\right) \left(\exists d \in \{0,1\}^{q(|I|+s)}\right) Q(I, c, d, \delta, f, r).$$

This permits an f that is at least $\frac{\delta}{2}$-good to be found by sequentially fixing one bit of f at a time (existentially quantifying on the remaining bits of f) and evaluating the statement using the Σ_3^{p}-oracle.$\square$

The algorithms in this and the previous sections are based on the existence of "good" hypotheses that discard many candidate functions no matter what counterexample they receive. It turns out that this is not only a sufficient condition but a necessary one.

In [A90], Angluin proved that several concept classes (such as finite automata, context-free grammars, and CNF and DNF formulas) are not learnable from a polynomial number of equivalence queries, even with unbounded computational power between queries, provided that the hypotheses used are taken from the same class. These results in [A90] are proved as follows: First, define a general combinatorial property (there called *approximate fingerprints*) saying that "good" hypothesis will fail to exist at some point of the learning process no matter how cleverly queries are asked. Second, show that the above mentioned classes have approximate fingerprints.

Next we show that approximate fingerprints characterize learnability by equivalence queries in an information-theoretic sense, and furthermore, that for any class learnable in this sense the computational power needed is at most that of a Σ_3^{p}-oracle.

These results are proved for the notion of polynomial-time learning used in [A88, A90], that applies also to infinite concepts. In this setting, an algorithm is said to run in polynomial time if the time used at any moment of its execution is at most a fixed polynomial of (1) the length of the shortest representation for the target concept, and (2) the maximal length of any counterexample received so far.

Definition: *Let C be an infinite class of concepts containing functions $\{0,1\}^\star \rightarrow \{0,1\}$. We say that C is* p-evaluable *if the following two tests can be made in polynomial time:*

- *Given y, is y a valid representation for any function in $f_y \in C$?*
- *Given y satisfying the first check, and $x \in \{0,1\}^\star$, is $f_y(x) = 1$?*

Definition: *For a language L, $L^{\leq n}$ is the set $L \cap \{0,1\}^{\leq n}$. For a class of concepts C, $C^{\leq n}$ is the class $\{\ L^{\leq n}\ :\ L \in C\ \}$.*

The following is a modified version of Angluin's definition of approximate fingerprints:

Definition [A90]: *A class C has approximate fingerprints if there exist polynomials p_1 and p_2 such that for every polynomial q and infinitely many n there is a concept class $T_n \subseteq C$ with*

- *T_n is $p_1(n)$-bounded (i.e., has some representation of size $\leq p_1(n)$),*
- *$T_n^{\leq p_2(n)}$ contains at least two sets, and*
- *no hypothesis in the class C of length at most $q(n)$ is $\frac{1}{q(n)}$-good for $T_n^{\leq p_2(n)}$.*

This definition is different from that in [A90] in two respects. First the definition in [A90] reads "for all sufficiently large n" instead of "for infinitely many n". Intuitively, to prove non-learnability it is enough to force superpolynomial running time in each algorithm at infinitely many lengths. Second, to have a bounded search space we use $T_n^{\leq p_2(n)}$ in two places where [A90] used T_n. Again, Angluin's proof goes through with these changes, and the approximate fingerprints she finds for dfa, nfa, cfg, and CNF and DNF formulas also witness these properties.

Theorem 16: *Let C be any p-evaluable class, and consider algorithms that make only equivalence queries with hypotheses in C. The following three statements are logically equivalent:*

1. *C is learnable from a polynomial number of equivalence queries of polynomial size (and unlimited computational power).*

2. *C does not have approximate fingerprints.*

3. *There is a deterministic polynomial-time algorithm that learns C using a Σ_3^p-oracle and equivalence queries.*

Proof: "1 implies 2" was by proved by Angluin as the main technical tool in [A90]. "3 implies 1" is immediate since in exponential time we can solve the Σ_3^p queries without increasing the number of queries. So we only have to prove that "2 implies 3", that is, give a learning algorithm for C assuming that it lacks approximate fingerprints.

Let f be a concept in C. The *size* of f is by definition the length of the shortest representation in C for f. For a natural number n, we say that "n is large enough for f" if

- f has some representation of size at most n, and
- any concept $h \neq f$ whose size is at most that of f differs from f in at least one string of length $\leq n$.

We first describe an algorithm A that reads as input a natural number n, and has the following properties:

P1: Whenever A outputs a representation, it is an exact representation for f.

P2: If $A(n)$ receives only counterexamples of length at most n, and n is large enough for the target, then $A(n)$ always outputs some representation.

P3: For every n, $A(n)$ always halts, and it does so in time polynomial in n and the length of the longest counterexample received.

Later we will remove the need for input n.

Fix $p_1(n) = p_2(n) = n$ and let q be the polynomial provided for p_1 and p_2 by the assumption that C does not have approximate fingerprints.

Let f be the target function. Algorithm A assumes that n is large enough for f and builds a set I of pairs $(x, f(x))$, thus making sure that the target function is still in C_I. Set I is initially empty. Suppose that at a certain moment $C_I^{\leq n}$ contains more than one set. This is easily checked with an NP oracle. Then, the assumption that C does not have approximate fingerprints guarantees that there is some representation h of length at most $q(n)$ that is $\frac{1}{q(n)}$-good for $C_I^{\leq n}$. By Lemma 15, algorithm A can find such an h in time polynomial in n using an oracle in Σ_3^p. Then, A queries h as an equivalence query. If the counterexample x received has length greater than n, $A(n)$ stops without output; if $|x| \leq n$, $A(n)$ adds $(x, f(x))$ to I.

If no h is answered "YES" after $q(n) \ln |C|$ queries, either $C_I^{\leq n}$ is empty, or it contains at most one set. In the first case, n is not large enough for the target concept. In the second case, the Σ_3^p oracle can be used to find some representation of size n for that set. If an equivalence query with that representation is not answered "YES", again n is not large enough.

This concludes the description of algorithm A.

To learn C if a large enough n is not known in advance, it is enough to execute $A(i)$ sequentially with inputs $i = 1, 2, 3, 4, \ldots$ If $A(i)$ outputs some representation, output it and halt. If $A(i)$ halts without output, ask for equivalence the i representations of length $\log i$ before moving to $i + 1$.

Correctness is clear because any representation output by any $A(i)$ must be correct (property *P1*). For termination, note that each $A(i)$ is terminating (property *P3*) and that i never exceeds by more than an exponential the length of the shortest correct representation.

To discuss the time complexity, let N be the size of the target concept, n the minimum such that $A(n)$ outputs a representation, and l_i the length of the longest counterexample received by $A(i)$. Note why n and N need not coincide: n can be smaller than N, because $A(n)$ makes hypotheses of size up to $q(n)$, and also greater than N, if n is still not large enough according to the definition above, or if the teacher provides counterexamples whose length exceeds N. For the argument we distinguish two cases:

Case 1: $n \leq N$. Each run of $A(i)$ with $i \leq n$ takes time polynomial in i and l_i, i.e., polynomial in N and $\max_i\{l_i\}$.

Case 2: $n > N$. Let i be such that $N \leq i < n$ and assume that $A(i)$ does not terminate successfully. By property *P2*, either the teacher provides a counterexample of length greater than i, even if shorter counterexamples exist, or i is not large enough for the target. In the latter case, there is another concept of size N or less that differs from the target in no string of length i or less; hence, $A(i)$ must receive a counterexample of length greater than i also in this case. The running time of $A(i)$ for all such i is polynomial in i and $l_i > i$, i.e., polynomial in $\max_i\{l_i\}$.

In summary, each run $A(i)$ takes time polynomial in N and $\max_i\{l_i\}$, and there are at most $\max_i\{N, \max_i\{l_i\}\}$ runs. So the new learner runs in polynomial time. □

6 Learning with Membership Queries

In this section we show how to eliminate equivalence queries using membership queries and an *NP*-oracle and provide some matching lower bounds for the number of membership queries needed. We also provide a characterization similar to Theorem 16 for learning with membership query alone. Finally we show that classes learnable via membership queries and unlimited computational power are also learnable using membership queries and an *NP* oracle in randomized expected polynomial time.

First we describe some upper bound results in learning with membership queries alone.

Lemma 17: *Let C be a concept class over $\{0,1\}^n$. Let L_1 and L_2 be two exact learning algorithm which uses equivalence and membership queries to learn C. Suppose that any hypotheses h_1 and h_2 issued by both are known to satisfy $h_1 \not\equiv h_2$ (except on the last step). Then there is an algorithm that uses membership queries and the NP oracle to learn C.*

Proof: The idea (which appeared implicitly in [BC92]) is to run L_1 and L_2 in parallel until the first equivalence query is issued by each, say h_1 and h_2 (respectively). Since we know $h_1 \not\equiv h_2$, we can use the *NP* oracle to find a c such that $h_1(c) \neq h_2(c)$ (this takes n *NP* queries). One membership query at c will establish which algorithm may continue its execution (we suspend the other). We then repeat the process again until the continued algorithm issues its next equivalence query. By assumption, the suspended equivalence query and the new one are still not equal. Again we use the *NP* oracle to find a counterexample for one of them, and so on. In this way we never ask any equivalence query but at the expense of n *NP* queries and one membership query.□

Definition: Let f be a boolean function over $\{0,1\}^n$. Then we let $d(f) = size_{DNF}(f)$ be the minimum number of terms in a DNF that represents f. Similarly we define $c(f) = size_{CNF}(f)$ to be the minimum number of clauses in a CNF that represents f. These two size measures are polynomially related to the standard size measure of the number of bits required to represent a boolean function in a DNF or a CNF form.

Let C be a concept class over $\{0,1\}^n$. In [B93], a measure called the *monotone dimension* $m(C)$, and its *dual* (called the *dual monotone dimension* $m^\partial(C)$), are defined. It was proven that there is an algorithm to learn any $f \in C$ that uses $d(f)m(C)$ equivalence queries and $n^2 d(f)m(C)$ membership queries. More importantly any hypothesis h issued by this algorithm satisfies $h \Rightarrow f$. There is also a *dual* algorithm to learn any $f \in C$ that uses $c(f)m^\partial(C)$ equivalence queries and $n^2 c(f)m^\partial(C)$ membership queries. Also any hypothesis h^∂ issued by this algorithm satisfies $f \Rightarrow h^\partial$.

We observe that if we run both algorithms from [B93] in the manner as in the previous lemma, then the hypotheses issued by both algorithm will never be equal except when they are equal to the target function. Hence we can conclude the following.

Theorem 18: *Let C be a concept class. Then there is an algorithm that learns C using $n(n+1)(d(f)m(C) + c(f)m^\partial(C))$ membership queries and $(n+1)(d(f)m(C) + c(f)m^\partial(C))$ calls to the* NP *oracle.*

Proof: The factor of $(n+1)$ in the number of calls to the *NP* oracle is to account for 1 call to check if the two hypotheses are equal and n calls to find a counterexample if they are not equal.□

Corollary 19: *The following classes are learnable from membership queries and the* NP *oracle in time polynomial in n and $d(f)$ and $c(f)$ where f is the target function.*

1. *Monotone boolean functions.*

2. *The class $O(\log n)$-CNF $\bigcap$ $O(\log n)$-DNF.*

Proof: For (a), we note that as shown in [B93] monotone boolean functions satisfy $m(C) = m^\partial(C) = 1$. For (b), we have the fact that the class $O(\log n)$-CNF is known to have polynomial (in n) monotone dimension while the class $O(\log n)$-DNF is known to have polynomial (in n) dual monotone dimension (see again [B93]). $\square$

Let LMQ^k be the set of s-bounded concept classes C which are learnable using at most n^k membership queries (and unlimited computational power). Each concept class C in LMQ^k has the following property: given a set of labeled examples $I \subseteq \{0,1\}^n \times \{0,1\}$, there is an algorithm that, on input f and I, decides whether or not $f \in C_I$[5]. This decision algorithm must run in time polynomial in $|I|$ and s.

Fact 1: *If $C \in LMQ^k$ then for any subset $C' \subseteq C$ we have $C' \in LMQ^k$.*

We call a point $a \in \{0,1\}^n$ *k-good* for C if $\gamma_a^C \geq n^{-k}(1 - 1/|C|)$.

Fact 2: *Let $C \in LMQ^k$. Then for any $C' \subseteq C$ with $|C'| \geq 2$ there is $a \in \{0,1\}^n$ which is k-good for C'.*

Proof: Assume there is $C' \subseteq C$ so that for all $a \in \{0,1\}^n$ a is not k-good for C', i.e. $\gamma_a^{C'} < n^{-k}(|C'| - 1)/|C'|$. We will show that $C' \notin LMQ^k$ which (by the fact above) will imply $C \notin LMQ^k$. Let A be an arbitrary learning algorithm for C' which uses at most n^k membership queries. Consider the following adversarial strategy for answering queries by A: given the query $MQ(a)$, answer $b \in \{0,1\}$ so that $\gamma_{(a,\bar{b})}^{C'} < n^{-k}(|C'| - 1)/|C'|$. This strategy allows A to eliminate only $< n^{-k}(|C'| - 1)/|C'|$ fraction of C' each time. So after n^k steps A can only eliminate $< |C'| - 1$ elements of C' implying there are at least two concepts remaining uneliminated. Since A is arbitrarily chosen, $C' \notin LMQ^k$ as required.$\square$

As a corollary to the second fact we get that any subset $C' \subset C$, with $|C'| \geq 2$, has a membership point $a \in \{0,1\}^n$ which satisfies

$$\gamma_a^{C'} \geq \frac{1}{2} n^{-k}.$$

Theorem 20: *There is a randomized expected polynomial time algorithm with access to an NP oracle that learns any $C \in LMQ^k$ using at most n^{2k} membership queries.*

Proof: Let $C \in LMQ^k$. Set $N = n^k$, $\alpha = 1/16N$ and $m = N^2$. We say a membership point a is a *ϵ-splitter* for C if $\gamma_a^C \geq \epsilon$. The r-th threshold function on n variables TH_r^n is defined as

$$TH_r^n(x_1, ..., x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^n x_i \geq r \\ 0 & \text{otherwise} \end{cases}$$

Let U be an approximately uniform distribution on C with tolerance $\epsilon = 1$. By a similar argument as in Lemma 5, we claim that U can undersample by a factor of at most $(1+\epsilon)^{-2}$, i.e. if $\gamma_{(x,b)}^C \geq \delta$ then $\Pr_U[f(x) = b] \geq \delta(1+\epsilon)^{-2}$. We sample independently m functions from C according to U, say $F = \{f_i\}_{i=1}^m \in_U C^m$.

Define $T_1 = TH_{\alpha m}^m(F)$ and $T_2 = TH_{(1-\alpha)m}^m(F)$. We prove that with high probability the event $T_1 \not\equiv T_2$ occurs. Since C has a k-good $a \in \{0,1\}^n$, i.e. $\gamma_a^C \geq (2N)^{-1}$, the event $T_1 \equiv T_2$

[5] Recall that $C_I = \{h \in C : (\forall (x,b) \in I)(h(x) = b)\}$; see Section 3.

implies $T_1(a) = T_2(a)$. By Chernoff bounds (equation (2) Section 3) we get

$$\Pr[T_1(a) \equiv T_2(a)] \;=\; \Pr[T_1(a)=0] + \Pr[T_2(a)=1] \;<\; 2F^-(m/8N, 1/2) \;\leq\; 2e^{-\Omega(N)}.$$

Thus with probability $1 - e^{-\Omega(N)}$ we have $T_1 \not\equiv T_2$. Next we show that conditioning on $T_1 \not\equiv T_2$, the event that for all $a \in T_1 \Delta T_2$, $\gamma_a^C \geq (32N)^{-1}$, occurs with high probability. Calling the latter event A, by the union bound and Chernoff bounds (equation (1) Section 3) we have

$$\begin{aligned}\Pr[\overline{A} \mid T_1 \not\equiv T_2] \;&\leq\; \sum_{a \in T_1 \Delta T_2} \Pr\left[T_1(a) \neq T_2(a)\;,\; \gamma_a^C < (32N)^{-1} \mid T_1 \not\equiv T_2\right] \\ &\leq\; 2^n F^+(m/32N, 1) \leq 2^n e^{-\Omega(N)}.\end{aligned}$$

The probability that we failed (at some step) to locate a $(32N)^{-1}$-splitter is at most $\Pr[T_1 \equiv T_2] + \Pr[\overline{A} \mid T_1 \not\equiv T_2] \;\leq\; e^{-\Omega(n)}$.

We use the *NP* oracle (for the second time) to find a $(32N)^{-1}$-splitter $a \in \{0,1\}^n$, which allows progress to be made in learning. We run the above for N^2 times. The probability that at every step we succeed to locate a $(32N)^{-1}$-splitter (for different invocations of C) is at least $1 - N^2 e^{-\Omega(n)} \geq 1 - e^{-\Omega(n)}$. Thus with probability $1 - e^{-\Omega(n)}$ we will finish (i.e. reduce C to one element) within $N^2 = n^{2k}$ steps. □

Putting together some of the previous results, we can give a precise characterization of learnability with membership queries alone, very similar to that of Theorem 16.

Theorem 21: *Let $C = \bigcup_{n>0} C_n$ be any p-evaluable infinite concept class, with each C_n over $\{0,1\}^n$. The following are equivalent:*

1. *There is a k such that, for every n, $C_n \in LMQ^k$.*
2. *There is a k such that, for every n and every $C' \subseteq C_n$ with $|C'| \geq 2$, there is some k-good assignment for C'.*
3. *There is a deterministic polynomial-time algorithm that learns C using a Σ_3^p-oracle and membership queries.*

Proof (Sketch): "1 implies 2" is Fact 2; "2 implies 3" is proved in a way completely analogous to Theorem 20: At any point in the run of the algorithm, let C_I be the set of functions in C_n consistent with the answers seen so far. Then, use the Σ_3^p-oracle to deterministically generate a k-good assignment for C_I. "3 implies 1" is immediate. □

Next we provide some lower bounds on the number of membership queries needed for learning some concept classes. To the best of our knowledge, these lower bound statement are the first of its kind.

Theorem 22: *We have the following two lower bounds.*

1. *Any algorithm that learns monotone boolean functions with membership query requires at least $\Omega(\max\{c(f), d(f)\})$ queries where f is the target function.*

2. *Any algorithm learning any class C with membership queries requires at least $\Omega(\max\{m(C), m^{\partial}(C)\})$ queries.*

Proof (Sketch): For (1), we use an adversarial argument on the following class of monotone read-twice DNF formulas $C = \{f | f = \bigvee_{i=1}^{k} T_i \vee T\}$. For each i, $1 \leq i \leq k$, let $T_i = \bigwedge_{j=(i-1)k+1}^{ik} x_j$. The last term T consists of all variables except that it is missing exactly one variable from each of T_i. For the lower bound argument we give away to the learner all T_i, $1 \leq i \leq k$, but not T. Suppose the learner asks $MQ(a)$. a cannot be all one in any T_i since f is one and the learner knows this already. If a contains more than one zero in some T_i then the adversary says FALSE iff a falsifies all of T_i, $1 \leq i \leq k$, and TRUE otherwise. This conveys no information about T since a falsifies T. Hence the learner must ask membership queries where there is precisely one zero in each V_i. There are $(n/k)^k$ such questions and the adversary may answer FALSE except for the last one. We omit the proof that the maximum of $c(f)$ and $d(f)$ is at most $\max\{k+1, (n/k)^k\}$.

For (2) we consider the monotone clause $T = \bigvee_{i=1}^{n} x_i$ and the following class $C = \{T(x \oplus a) | a \in A \subseteq \{0,1\}^n\}$. We claim that any learning algorithm using membership query alone requires at least $m(C)$ queries. Here we even assume that the learner has the knowledge of A. First we assert that any membership query c must be such that $c \in A$. Otherwise if $c \notin A$ then $T(c \oplus a) = 1$ (since $c \oplus a \not\equiv 0$) for all $a \in A$. So now the adversary may say YES for all $a \in A$ asked by the learner except for the last one. $\Box$

Remark: Angluin, Hellerstein, and Karpinski [AHK93] have shown that monotone read-once formulas are exactly learnable from membership queries alone. The proof of Theorem 22(1) rules out the possibility for monotone read-twice DNF (since the CNF size might be exponentially large).

7 Applications to Structural Complexity Theory

Watanabe [W94] has observed that a consequence of Theorem 7(b) in Section 3 is an improvement of the following result of Karp and Lipton [KL80].

Theorem 23 [KL80]: *If every NP set has polynomial-size circuits then the polynomial hierarchy collapses to Σ_2^{P}.*

It is clear that ZPP^{NP} is contained in Σ_2^{P} but the other direction of containment is not known (and would be surprising). Watanabe has observed the following, and we reproduce a sketch of his proof below.

Theorem 24 [W94]: *If every NP set has polynomial-size circuits then the polynomial hierarchy collapses to ZPP^{NP}.*

Proof (sketch): Suppose that every *NP* set has polynomial-size circuits. Thus, in particular, *SAT* has polynomial-size circuits.

First, it is shown that that, for each n, one can construct a polynomial-size circuit deciding SAT_n (i.e., the set of strings of length n in SAT) by a ZPP^{NP} computation. The idea is as follows. By Theorem 7(b), it is possible to construct the circuit for SAT_n in random polynomial-time time using an *NP*-oracle *and* asking equivalence queries. Thus, the first result follows if one can simulate a *teacher* answering equivalence queries in P^{NP}.

For a given circuit (description) f, we need to construct a counterexample with respect to SAT_n (i.e., an element in the symmetric difference between $L(f)$ (where $L(f)$ is the strings accepted by f) and SAT_n. A counterexample is either an $x \in SAT_n - L(f)$, or an $x \in L(f) - SAT_n$. The first type of counterexample can be found using an *NP*-oracle to evaluate:

$$(\exists v)(\exists w)(|uv| = n \wedge f(uv) = 0 \wedge uv \in SAT_n \text{ and } w \text{ witnesses this fact})$$

for a series of prefixes u. On the other hand, the latter type of counterexample can be found by using an *NP*-oracle to evaluate:

$$\begin{array}{l}(\exists v)\,(|uv| = n \wedge f(uv) = 1 \wedge \\ \qquad \text{a standard binary search using } f \text{ fails to find a satisfying assignment of } uv)\end{array}$$

for a series of prefixes u. Thus, with an *NP*-oracle, we can simulate a teacher for circuits recognizing SAT_n.

After having a ZPP^{NP}-uniform circuit family for SAT, we can replace any quantified (with a single quantifier) circuit expression with an unquantified circuit expression with only a polynomial blow-up in size. By repeating this process a constant number of times, we can evaluate any quantified (with any *constant* number of quantifiers) circuit expression.□

Acknowledgments

We are grateful to Osamu Watanabe for pointing out an interesting application of our results, and for permitting us to include it in this paper (in Section 7). We also thank Dana Angluin, Sleiman Matar, and Vijay Raghavan for helpful comments. Finally we thank the anonymous referees whose comments greatly improve the presentation of our paper.

References

[A88] Dana Angluin. Queries and Concept Learning. *Machine Learning*, 2:319-342, 1988.

[A90] Dana Angluin. Negative Results for Equivalence Queries. *Machine Learning*, 5:121-150, 1990.

[AHK93] Dana Angluin, Lisa Hellerstein, and Marek Karpinski. Learning Read-Once Formulas with Queries. *Journal of the ACM*, 40(1):185-210, 1993

[B89] Ravi Boppana. Amplification of Probabilistic Boolean Formulas. In *Advances in Computing Research*, **5**:4, pages 27–45, 1989.

[B93] Nader Bshouty. Exact Learning via the Monotone Theory. In *IEEE Foundations of Computer Science*, pages 302–311, 1993.

[BC92] Nader Bshouty and Richard Cleve. On the Exact Learning of Formulas in Parallel. In *Proceedings of the 33rd Symposium on Foundations of Computer Science*, pages 513–522, 1992.

[G93] Ricard Gavaldà. On the Power of Equivalence Queries. In *Proceedings of EUROCOLT '93*, pages 193–203, Oxford University Press 1994.

[GKS93] Sally Goldman, Michael Kearns, and Robert Schapire. Exact Identification of Read-Once Formulas using Fixed Points of Amplification Functions. *SIAM Journal on Computing*, **22**, 1993.

[GRS93] Sally Goldman, Ronald Rivest, and Robert Schapire. Learning Binary Relations and Total Orders. *SIAM Journal on Computing*, **22**:5, pages 1006–1034, 1993.

[JVV86] Mark Jerrum, Leslie Valiant, and Vijay Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theoretical Computer Science*, **43**, pages 169–188, 1986.

[K93] Sampath Kannan. On the Query Complexity of Learning. In *Proceedings of the Sixth Annual Workshop on Computational Learning Theory*, pages 58–66, 1993.

[KL80] Richard M. Karp and Richard J. Lipton. Some Connections Between Nonuniform and Uniform Complexity Classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 302–309, 1980.

[KM91] Eyal Kushilevitz and Yishay Mansour. Learning Decision Trees using the Fourier Spectrum. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 455–464, 1991.

[L88] Nick Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. *Machine Learning*, **2**, pages 285–318, 1988.

[MS56] E.F. Moore and C. Shannon. Reliable Circuits Using Less Reliable Relays. *J. Franklin Inst.*, **262**, pages 191–208, 281–297, 1956.

[PV88] Leonard Pitt and Leslie Valiant. Computational Limitations on Learning from Examples. *Journal of the Association for Computing Machinery*, **35**:4, pages 965–984, October 1988.

[R90] Prabhakar Raghavan. Lecture Notes on Randomized Algorithms. *Research Report, IBM Research Division*, RC15340 (#68237), 1/9/90.

[S83] Michael Sipser. A Complexity Theoretic Approach to Randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 330–334, 1983.

[S85] Larry Stockmeyer. On Approximation Algorithms for $\#P$. *SIAM Journal on Computing*, **14**:4, pages 849–861, 1985.

[V84a] Leslie Valiant. Short Monotone Formulae for the Majority Function. *J. Algorithms*, **5**, pages 363–366, 1984.

[V84b] Leslie Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[W94] Osamu Watanabe. Personal communication, 1994.