

# Efficient Learning Algorithms Yield Circuit Lower Bounds

Lance Fortnow  
University of Chicago  
Computer Science Department  
fortnow@cs.uchicago.edu

Adam R. Klivans  
The University of Texas at Austin  
Computer Science Department  
klivans@cs.utexas.edu

## Abstract

We describe a new approach for understanding the difficulty of designing efficient learning algorithms. We prove that the existence of an efficient learning algorithm for a circuit class  $C$  in Angluin's model of exact learning from membership and equivalence queries or in Valiant's PAC model yields a lower bound against  $C$ . More specifically, we prove that any subexponential time, deterministic exact learning algorithm for  $C$  (from membership and equivalence queries) implies the existence of a function  $f$  in  $\text{EXP}^{\text{NP}}$  such that  $f \notin C$ . If  $C$  is PAC learnable with membership queries under the uniform distribution or Exact learnable in randomized polynomial time, we prove that there exists a function  $f \in \text{BPEXP}$  (the exponential time analog of BPP) such that  $f \notin C$ .

For  $C$  equal to polynomial-size, depth-two threshold circuits (i.e., neural networks with a polynomial number of hidden nodes), our result shows that efficient learning algorithms for this class would solve one of the most challenging open problems in computational complexity theory: proving the existence of a function in  $\text{EXP}^{\text{NP}}$  or  $\text{BPEXP}$  that cannot be computed by circuits from  $C$ . We are not aware of any representation-independent hardness results for learning polynomial-size neural networks. Our approach uses the framework of the breakthrough result due to Kabanets and Impagliazzo showing that derandomizing BPP yields non-trivial circuit lower bounds.

# 1 Introduction

Discovering the limits of efficient learnability remains an important challenge in computational learning theory. Traditionally, computational learning theorists have reduced problems from computational complexity theory or cryptography to learning problems in order to better understand the difficulty of various classification tasks.

There are two lines of research in learning theory in this direction. First, several researchers have shown that *properly* PAC learning well-known concept classes (i.e. learning with the requirement that the output hypothesis be of the same form as the concept being learned) such as DNF formulas, finite automata, or intersections of halfspaces is NP-hard with respect to randomized reductions [PV88, Gol78, ABF<sup>+</sup>04].

Secondly, in seminal work due to Valiant [Val84] and Kearns and Valiant [KV94] initiated a line of research that applied results from cryptography to learning. They proved that learning polynomial-size circuits, regardless of the representation of the hypothesis of the learner, would imply the existence of algorithms for breaking cryptographic primitives. Various researchers have extended this work to smaller subclasses of circuits [Kha93, JKS02]. In particular, Kharitonov [Kha93] has given cryptographic hardness results for learning polynomial-size constant depth circuits with respect to the uniform distribution by showing that certain weak pseudo-random generators can be computed in  $AC^0$  (Kharitonov's results also hold in the case that the learner has query access; it is not known if his results hold for circuits of depth less than five).

It is not known, however, what the minimal circuit complexity is for computing pseudo-random generators. For example, it is a difficult open question as to whether polynomial-size depth-two threshold circuits (polynomial-size neural networks with one hidden layer) can compute strong pseudo-random generators. It is worth noting that Klivans and Sherstov [KS06] have recently shown that polynomial-size depth-two threshold circuits can compute public-key cryptosystems based on the hardness of certain lattice-problems. Still, their cryptographic hardness results do not hold with respect to the uniform distribution or in the case that the learner has query access.

## 1.1 Reducing Circuit Lower Bounds to Learning Concept Classes

We give a new approach for showing the difficulty of proving that certain circuit classes admit efficient learning algorithms. We show that if a class of circuits  $C$  is efficiently learnable in either Angluin's exact model or Valiant's PAC model of learning, then we can prove a circuit lower bound against  $C$ . Hence, the existence of efficient learning algorithms (for many choices of  $C$ ) would settle some important and well-studied open questions in circuit complexity.

Our first theorem states that a deterministic subexponential time exact learning algorithm for a concept class  $C$  implies a lower bound for Boolean circuits for a somewhat large complexity class:

**Theorem 1** *Let  $C$  be any circuit representation class. Assume that  $C$  is exactly learnable from membership and equivalence queries in time  $2^{s^{o(1)}}$  where  $s$  is the size of the target concept. Then  $EXP^{NP} \not\subseteq P/poly[C]$ .*

We borrow the notation of Watanabe and Gavaldà [WG94] and let  $P/poly[C]$  stand for the class of all languages that can be computed by polynomial-size circuits from representation

class  $C$  (e.g.,  $C$  could be DNF formulas or depth-5 threshold circuits). For more discussion of various types of representation classes in a query-learning context we refer the reader to Watanabe and Gavaldà [WG94].

If we take  $C$  to be the class of depth-two neural networks we obtain the following corollary:

**Corollary 1** *If there exists an algorithm for exactly learning depth-two neural networks in time  $2^{s^{o(1)}}$  with membership and equivalence queries then  $\text{EXP}^{\text{NP}} \not\subseteq \text{TC}^0[2]$ .*

The class  $\text{TC}^0[2]$  is the class of all languages that can be computed by polynomial-size, depth-two threshold circuits.

Finding a function in a uniform class such as  $\text{EXP}^{\text{NP}}$  that cannot be computed by polynomial-size, depth-two threshold circuits has been a challenging open problem for over two decades in computational complexity.

If we assume that our exact learning algorithm runs in polynomial-time rather than subexponential time we can show that even randomized exact learning algorithms imply circuit lower bounds against  $\text{BPEXP}$ , the exponential time version of  $\text{BPP}$ :

**Theorem 2** *Let  $C$  be any circuit representation class. Assume there exists a randomized polynomial-time algorithm for exactly learning  $C$  from membership and equivalence queries. Then  $\text{BPEXP} \not\subseteq \text{P/poly}[C]$ .*

We are unaware of any lower bounds for circuit classes such as polynomial-size depth-two threshold circuits against  $\text{BPEXP}$ .

Theorems 1 and 2 also directly applies to arithmetic circuits. If, in addition, we restrict the output hypothesis of the learner to be an arithmetic circuit or formula (of possibly larger size and depth) we can replace the  $\text{NP}$  oracle in Theorem 1 with an  $\text{RP}$  oracle and obtain a finer separation of uniform and non-uniform circuit classes.

**Theorem 3** *Let  $C$  be a class of arithmetic formulas. Assume that  $C$  is exactly learnable from membership and equivalence queries in polynomial-time and the hypothesis of the learner is an arithmetic formula. Then  $\text{EXP}^{\text{RP}} \not\subseteq \text{P/poly}[C]$ .*

*If we allow both  $C$  and the hypothesis to be arithmetic circuits (instead of arithmetic formulas) then we have  $\text{ZPEXP}^{\text{RP}} \not\subseteq \text{P/poly}[C]$ .*

We note here that proving lower bounds against polynomial-size arithmetic formulas and even depth-3 arithmetic circuits remains one of the most difficult challenges in algebraic complexity. Furthermore, as with polynomial-size neural networks, we are not aware of hardness results for learning restricted models of arithmetic circuits even if the learner must output an arithmetic formula.

These results also apply to the PAC model. Due to the inherent role of randomness in the definition of PAC learning, our lower bounds apply to the complexity class  $\text{BPEXP}$ :

**Theorem 4** *Let  $C$  be any circuit representation class. Assume that  $C$  is PAC learnable in polynomial time with respect to the uniform distribution with membership queries. Then  $\text{BPEXP} \not\subseteq \text{P/poly}[C]$ .*

## 1.2 The Difficulty of Proving Uniform Circuit Lower Bounds

Perhaps the ultimate circuit lower bound would be a proof of the existence of a Boolean function  $f$  in NP that cannot be computed by polynomial-size circuits. It is well known that this would separate P from NP. Unfortunately, we are nowhere near proving this type of circuit lower bound.

Kannan [Kan82a] was the first to give a uniform complexity class containing Boolean functions with superpolynomial circuit-size. In particular, he proved that  $\Sigma_2^{\text{EXP}} \cap \Pi_2^{\text{EXP}}$  is not in P/poly.

Currently, the smallest uniform complexity class known to contain languages with superpolynomial Boolean circuit complexity (or even superpolynomial-size depth-two threshold circuit complexity) is  $\text{MA}_{\text{EXP}}$  [BFT98], the exponential-time analog of Merlin-Arthur proofs. It is known that  $\text{MA}_{\text{EXP}}$  is contained in  $\text{ZPEXP}^{\text{NP}}$  [AK01, GZ97]. For a full discussion on the state of the art for separating uniform complexity classes from P/poly we refer the reader to Bro Miltersen et al. [MVW99].

As stated above, our main result is a proof that the existence of a PAC learning algorithm for circuit class  $C$  would imply a separation of  $\text{BPEXP}$  from the corresponding class  $C$ . The class  $\text{BPEXP}$ , the exponential-time analogue of BPP, is easily seen to be contained in  $\text{MA}_{\text{EXP}}$ . We believe that for many interesting classes  $C$ , such a separation will be very difficult to prove given our current set of techniques (for a more extensive discussion of this topic we refer the reader to recent work due to Santhanam [San07] and Aaronson and Wigderson [AW07]).

## 1.3 Our Approach

The proof of Theorem 1 follows the outline of the work of Kabanets and Impagliazzo [KI04] on derandomizing algebraic circuits: we assume that  $\text{EXP}^{\text{NP}}$  is computable by some non-uniform circuit class  $C$  (otherwise there is nothing to prove). This implies, via a sequence of well-known reductions in complexity theory, that the Permanent is complete for  $\text{EXP}^{\text{NP}}$  (the analogous statements with  $\text{EXP}^{\text{NP}}$  replaced by  $\text{BPEXP}$  are not known to be true). At this point we need to use the supposed exact learning algorithm to construct an algorithm for computing the Permanent which runs in subexponential time and has access to an NP oracle. This leads to an immediate contradiction via time hierarchy theorems.

In the work of Kabanets and Impagliazzo, the assumption of a deterministic algorithm for polynomial-identity testing is used to develop a non-deterministic algorithm for computing the Permanent. In this work, we need to use the exact learning algorithm to construct the circuit  $c \in C$  that computes the permanent. The main difficulty is that the exact learning algorithm needs access to a membership and equivalence query oracle, which we do not have. Using an idea from Impagliazzo and Wigderson's result on derandomizing BPP, we can simulate membership queries by inductively constructing circuits for the permanent on shorter input lengths. Simulating equivalence queries is slightly trickier and requires access to an NP oracle to find counterexamples.

To reduce the dependence on the NP oracle we can use randomness, but only in cases regarding arithmetic circuits and formulas, where output hypotheses can be suitably interpreted as low-degree polynomials. Our results on PAC learning and randomized Exact learners require a slightly different approach, as we are not aware of collapse consequences for the class  $\text{BPEXP}$  even if it is contained in P/poly. We appeal to work on derandomization

due to Impagliazzo and Wigderson [IW01] that makes use of the random-self-reducibility of the Permanent.

## 1.4 Outline

In Section 2 we define various learning models and state all the theorems from complexity theory necessary for proving our main result. In Section 3 we give a proof of our main result for exact learning in the Boolean case. Our results regarding learnability in the PAC model are in Section 4. In Section 5 we discuss applications to exact learning in the algebraic setting.

## 2 Preliminaries

Valiant's PAC model [Val84] and Angluin's model of Exact Learning from Membership and Equivalence queries [Ang88] are two of the most well-studied learning models in computational learning theory. Recall that in Valiant's PAC model we fix a concept class  $C$  and a distribution  $D$  and a learner receives pairs of the form  $(x, c(x))$  where  $x$  is chosen from  $D$  and  $c$  is some fixed concept in  $C$ . The learner's goal is to output, with probability  $1 - \delta$ , a hypothesis  $h$  such that  $h$  is a  $1 - \epsilon$  accurate hypothesis with respect to  $c$  under  $D$ . We say that the learner is efficient if it requires at most  $t$  examples, runs in time at most  $t$  and outputs a hypothesis that can be evaluated in time  $t$  where  $t = \text{poly}(n, 1/\epsilon, 1/\delta, |c|)$  ( $|c|$  denotes the size of the unknown concept and  $n$  is the length of  $x$ ). If the learner is allowed *membership queries* then it may query the unknown concept  $c$  at any point  $x$  of his choosing.

In Angluin's model of exact learning, the learner is trying to learn an unknown concept  $c : \{0, 1\}^n \rightarrow \{0, 1\}$  and is allowed to make queries of the following form:

1. (*Membership Query*) What is the value of  $c(x)$ ?
2. (*Equivalence Query*) Is  $h$  (the learner's current hypothesis) equal to  $c$ ?

If the equivalence query is answered affirmatively, the learner outputs  $h$  and halts. Otherwise, the learner receives a counterexample, namely a point  $z$  such that  $h(z) \neq c(z)$ . We say that an algorithm  $A$  exactly learns a concept class  $C$  in time  $t$  if for every  $c \in C$ ,  $A$  always halts and the time taken by  $A$  (including membership and equivalence queries) is bounded by  $t$ .

We sometimes allow our learner to run in randomized polynomial time. That is, the learner may flip coins at any time during the learning process but is required to output— with probability at least  $2/3$  over its internal random choices— a deterministic hypothesis that is exactly correct.

### 2.1 Uniform versus Non-uniform models of computation

We frequently mention standard complexity classes EXP, RP, BPP, NP, MA, ZPP, EE, ZPEXP as well as relativized versions of the classes (e.g.,  $\text{EXP}^{\text{NP}}$ ). We refer the reader to the Complexity Zoo (<http://www.complexityzoo.com>) for further details on these classes.

We say a language  $L$  has polynomial-size circuits (P/poly) if there is a polynomial  $p$  and a sequence of logical (AND-OR-NOT) circuits  $C_0, C_1, \dots$  such that for all  $n$ ,

1. The size of  $C_n$  is bounded by  $p(n)$ .
2. For all strings  $x = x_1 \dots x_n$ ,  $x$  is in  $L$  iff  $C(x_1, \dots, x_n) = 1$  where we use 1 for true and 0 for false.

Importantly, circuits describe nonuniform computation: the circuits for one input length may have no relation to the circuits for other lengths.

An algebraic circuit can only have addition, subtraction and multiplication gates, in particular no bit operations. All languages computed by algebraic circuits can be computed by a Boolean circuit (with a polynomial increase in size), but the converse is not known. A formula is a circuit described by a tree. It is well known that an arithmetic circuit is equivalent to a multivariate polynomial of degree at most exponential in the size of the circuit. Arithmetic formulas compute polynomials whose degree is at most a polynomial in the size of the formula.

As mentioned earlier, the smallest complexity class known not to contain polynomial-size circuits is  $\text{MA}_{\text{EXP}}$  ([BFT98], see also [MVW99]), Merlin-Arthur games with an exponential-time verifier. Kabanets and Impagliazzo [KI04] use derandomization techniques to show  $\text{NEXP}^{\text{RP}}$  does not have polynomial-size algebraic circuits.

It is a difficult open problem to improve upon  $\text{MA}_{\text{EXP}}$  as the smallest uniform class containing circuits of superpolynomial size even if we restrict ourselves to polynomial-size formulas, depth-two threshold circuits (neural nets), or constant-depth logical circuits with  $\text{Mod}_m$  gates for any  $m$  not a prime power.

## 2.2 Hierarchy theorems

**Theorem 5**  $\text{EXP}^{\text{NP}}$  is not contained in  $\text{SUBEXP}^{\text{NP}}$ , where  $\text{SUBEXP} = \text{DTIME}(2^{n^{o(1)}})$ .

**Proof:** The seminal paper in computational complexity by Hartmanis and Stearns [HS65] shows that for any time-constructible functions  $t_1(n)$  and  $t_2(n)$  with  $t_1^2(n) = o(t_2(n))$

$$\text{DTIME}(t_1(n)) \subsetneq \text{DTIME}(t_2(n))$$

and their proof relativizes. Theorem 5 follows by taking  $t_1(n) = 2^n$ ,  $t_2(n) = 2^{3n}$  and relativizing to SAT. ■

Let  $\text{EE}$  denote the class of languages computable in doubly-exponential time.

**Theorem 6**  $\text{EE}$  contains languages with super-polynomial circuit complexity.

**Proof:** We follow a proof idea of Kannan [Kan82b]. We know by counting arguments there is some function that is not computed by circuits of size  $2^{\sqrt{n}}$ . In doubly-exponential time we can, by brute force searching, find and evaluate the lexicographically least such function. ■

## 2.3 Properties of the Permanent

The Permanent of an  $k \times k$  matrix  $A$  is defined by

$$\text{Perm}(A) = \sum_{\sigma \in S_k} a_{1\sigma(1)} a_{2\sigma(2)} \cdots a_{k\sigma(k)}$$

Valiant [Val79] showed that the Permanent is complete for the class  $\#\text{P}$ , i.e., complete for functions counting the number of solutions of NP problems. The Permanent remains  $\#\text{P}$ -complete if we compute the Permanent over a sufficiently large finite field.

Following Kabanets and Impagliazzo [KI04], we consider the following language  $L$ :

$$\{(M, v) \mid M \text{ is a } 0, 1 \text{ matrix; } v = \text{Permanent}(M)\}.$$

Applying Toda's theorem, Kabanets and Impagliazzo [KI04] have shown that if the above language  $L$  is in NP then  $\text{P}^{\#\text{P}} \subseteq \text{NP}$  and if this language is in NSUBEXP then  $\text{P}^{\#\text{P}}$  is in NSUBEXP. We say  $x$  is an instance of Permanent of length  $n$  if we wish to determine whether  $x$  belongs to  $L$  and  $|x| = n$ .

We will use the following two well known facts about the Permanent:

**Fact 2** (*Downward Self Reducibility of the Permanent*) *Computing the permanent of an  $k \times k$  matrix is polynomial-time (Turing) reducible to computing  $k$  instances of the Permanent over  $k - 1 \times k - 1$  matrices.*

Fact 2 follows easily from the cofactor expansion of the Permanent.

The Permanent, when defined over a finite field, is also random-self reducible [Lip91], i.e., there is an efficient randomized procedure that will take an  $k \times k$  matrix  $A$  and produce  $k \times k$  matrices  $A_1, \dots, A_{k+1}$  and a polynomial-time function that takes the Permanents of the  $A_i$ 's and compute the Permanent of  $A$ . Each  $A_i$  is uniformly random over the space of all  $k \times k$  matrices, though  $A_i$  and  $A_j$  are not independent variables.

We state this as follows:

**Theorem 7** (*Random-self-reducibility of the Permanent*) [BF90, Lip91] *Assume that we have a circuit  $c$  that computes the Permanent on all but a  $1/n^2$  fraction of inputs ( $n$  is the length of the instance) with respect to any field  $F, |F| \geq n^2$ . Then there exists a randomized, polynomial-time algorithm  $A$  that uses  $c$  as an oracle such that for every input  $x$ ,  $A$  computes Permanent on  $x$  correctly with probability at least  $1 - 1/n$ .*

Finally, we make use of a lemma due to Kabanets and Impagliazzo:

**Lemma 3** [KI04] *Given an arithmetic circuit  $C$ , the problem of determining if  $C$  computes the Permanent on all inputs is in  $\text{coRP}$ .*

## 2.4 Collapse Theorems

A long line of research in complexity theory is devoted to understanding the consequences of large uniform complexity classes being contained in small, non-uniform ones. Perhaps the most well known collapse is due to Karp and Lipton, stating that if  $\text{NP} \subseteq \text{P/poly}$  then the polynomial-time hierarchy collapses ( $\text{PH} = \Sigma_2$ ). We will need the following two "collapse" theorems:

**Theorem 8** [BH92] If  $\text{EXP}^{\text{NP}} \subseteq \text{P/poly}$  then  $\text{EXP}^{\text{NP}} = \text{EXP}$ .

**Theorem 9** [BFNW93] If  $\text{EXP} \subseteq \text{P/poly}$  then  $\text{EXP} = \text{MA}$ .

Since  $\text{MA} \subseteq \text{PH}$  and  $\text{PH} \subseteq \text{P}^{\#\text{P}}$  [Tod91] we conclude that  $\text{EXP}^{\text{NP}} \subseteq \text{P/poly}$  implies  $\text{EXP}^{\text{NP}} = \text{P}^{\#\text{P}}$ . Applying Valiant's result on the complexity of the Permanent [Val79], we also have that  $\text{EXP}^{\text{NP}} \subseteq \text{P/poly}$  implies Permanent is complete for  $\text{EXP}^{\text{NP}}$ .

### 3 Lower Bounds from Exact Learning Algorithms

In this section we restate and prove our main result: algorithms for exactly learning circuit classes yield lower bounds against those same circuit classes.

**Theorem 1** Let  $C$  be a representation class of circuits. Assume that  $C$  is exactly learnable from membership and equivalence in time  $t = 2^{s^{o(1)}}$  where  $s$  is the size of the target concept. Then  $\text{EXP}^{\text{NP}} \not\subseteq \text{P/poly}[C]$ .

**Proof:** First assume that  $\text{EXP}^{\text{NP}} \subseteq \text{P/poly}[C]$  (since otherwise there is nothing to prove) and notice that by Theorem 8 we have  $\text{EXP}^{\text{NP}} = \text{EXP} = \text{P}^{\#\text{P}}$ . As such, the Permanent function is now complete for  $\text{EXP}^{\text{NP}}$  and is computable by some  $c \in \text{P/poly}[C]$ . We wish to give a  $\text{poly}(n, t)$  time algorithm for computing the permanent that is allowed to make calls to an NP oracle. Because  $\text{EXP}^{\text{NP}}$  can be reduced to Permanent in polynomial-time, such an algorithm would violate a relativized version of the time hierarchy theorem (Theorem 5) and complete the proof. Consider the following algorithm for computing the permanent:

**Algorithm A for Computing Permanent on input  $x$ :**

For  $i = 1$  to  $n = |x|$ :

1. If  $i = 1$ , output the trivial circuit for Permanent on  $1 \times 1$  matrices. Otherwise,
2. Run the exact learning algorithm to find  $c_i$ , the circuit that computes permanent on inputs of length  $i$ .
3. Simulate required membership queries and equivalence queries using  $c_{i-1}$  and the NP oracle.

**Output**  $c_n(x)$ .

To finish the proof of Theorem 1, it suffices to verify the following claim:

**Claim 4** Assume Permanent is computable by some  $c \in C$ . Then, given access to an oracle for NP, Algorithm A outputs a polynomial-size circuit that computes Permanent on inputs of length  $n$  in time  $\text{poly}(n, t)$ .

We verify Claim 4 by establishing the following invariant: at the end of each stage  $i \leq n$ , Algorithm A has generated a circuit that computes `Permanent` on inputs of length  $i$ . Given this invariant, Algorithm A can use the final circuit that computes `Permanent` on inputs of length  $n$  to evaluate `Permanent` on input  $x$ .

The invariant is trivially true for  $i = 1$ . Assume that Algorithm A now has access to circuit  $c_{i-1}$ . We must show how to simulate the membership and equivalence query oracles required by the learning algorithm. If we can simulate membership and equivalence query oracles using  $c_{i-1}$  and an NP oracle, then Algorithm A can be carried out in time  $\text{poly}(n, t)$  with an NP oracle, as desired.

### Simulating Membership Queries:

Assume that our learning algorithm makes a membership query and requires the value of the permanent on input  $y$  of length  $i$ . By induction assume we have constructed a circuit  $c_{i-1}$  for computing the permanent on inputs of length  $i - 1$ . Then, by applying Fact 2, the permanent on input  $y$  can be computed via  $i$  calls to the circuit  $c'$ . As we have assumed that  $c'$  is exactly correct for all inputs of length  $i - 1$ , we are certain that we will obtain the correct value for the permanent on  $y$  (this is the same technique used by Impagliazzo and Wigderson [IW01] for making certain derandomization reductions more “uniform”).

### Simulating Equivalence Queries:

Assume that we wish to determine if our current hypothesis  $h$  computes permanent correctly on all inputs of length  $i$ . We make the following query to the NP oracle:

“Does there exist an input  $z$  such that  $h(z)$  does not equal the value obtained by using the downward self-reducible property of the permanent and circuit  $c_{i-1}$ ?”

I.e. does there exist an input  $z$  where the self-reduction fails? By Fact 2, we can compute the true value of the `Permanent` via  $i$  calls to circuit  $c_{i-1}$ , so this predicate can be efficiently computed with access to an NP oracle. If there is no such input  $z$  then we are guaranteed that our hypothesis  $h$  is exactly correct on all inputs of length  $n$ .

If there is such an input  $z$ , we must find this “counterexample.” In order to do this we use binary search: we first make a query of the form

“Does there exist an input  $z$  where the self-reduction fails that begins with a 0?”

Again it is easy to see that this query can be answered efficiently with an NP oracle. If the answer is “no” then we know there must exist a counterexample that begins with a 1. Our next query is “Does there exist an input  $z$  beginning with 10 such that...” and so on for the remaining bits until we have obtained a counterexample.

Since we can construct a circuit for exactly computing the permanent on inputs of length  $i$  using an NP oracle and a circuit for the computing the permanent on inputs of length  $i - 1$ , we have established the desired invariant. Again, since  $c$  is from a polynomial-size circuit class, we can evaluate  $c(x)$  in polynomial time. This shows that Algorithm A is a time  $\text{poly}(t, n)$ , NP-oracle algorithm for computing the permanent and completes the proof. ■

We remark here that the fact that equivalence queries can be simulated by an NP oracle with access to membership queries has been observed before by Watanabe and Gavaldà [WG94]. They also show how to simulate subset and superset queries, rather than just equivalence queries.

Since learnability in the mistake bounded model [Lit88] implies learnability in the exact model of learning, we immediately obtain the following:

**Theorem 10** *Let  $C$  be a representation class of circuits. Assume that  $C$  is efficiently learnable in the mistake bounded model. Then  $\text{EXP}^{\text{NP}} \not\subseteq \text{P}/\text{poly}[C]$ .*

John Hitchcock has informed us that  $\text{EXP}^{\text{NP}}$  can be replaced with  $\text{EXP}$  in the above theorem by an application of results from resource bounded measure [Hit07, LSW00]. Again, we point out that the smallest known complexity class containing functions of superpolynomial-size circuit complexity is  $\text{MA}_{\text{EXP}}$ , the exponential-time analogue of MA [BFT98]. Proving  $\text{MA}_{\text{EXP}} \subseteq \text{EXP}^{\text{NP}}$  would be a non-trivial derandomization.

## 4 Lower Bounds from Randomized Learning Algorithms

In this section we show that efficient PAC algorithms for learning a circuit class  $C$  or polynomial-time randomized Exact learning algorithms imply the existence of a function  $f \in \text{BPEXP}$  such that  $f$  is not in  $C$ . Our result holds even if the PAC algorithm is allowed to make membership queries to the unknown concept. As with the complexity class  $\text{EXP}^{\text{NP}}$ , it is a difficult open problem as to whether  $\text{BPEXP}$  can be computed by circuit classes such as polynomial-size, depth-two threshold circuits, or polynomial-size arithmetic formulas. The smallest uniform complexity class known to not be contained in these circuit classes is  $\text{MA}_{\text{EXP}}$ . It is widely believed that  $\text{BPEXP}$  is strictly less powerful than  $\text{MA}_{\text{EXP}}$ .

We require the following lemma, which states that if the Permanent has polynomial-size circuits from some class  $C$  and  $C$  is PAC learnable, then the Permanent is computable in BPP. This lemma is implicit in the work of Impagliazzo and Wigderson [IW01] (although it was used there to obtain new results in derandomization). We provide a proof in the language of PAC learning:

**Lemma 5** ([IW01], restated) *Assume that the Permanent is computed by  $\text{P}/\text{poly}[C]$  for circuit representation class  $C$ . If  $C$  is PAC learnable with respect to the uniform distribution (with membership queries) then Permanent is in BPP.*

**Proof:** To compute Permanent on input  $x$  of length  $n$ , assume by induction we have a randomized circuit  $c_{n-1}$  such that for every  $x$ ,  $c_{n-1}$  computes the permanent correctly on  $x$  with probability at least  $2/3$ . Since  $C$  is PAC learnable, consider its associated learning algorithm  $A$  that learns any  $c \in C$  in time  $\text{poly}(n, 1/\epsilon, 1/\delta, |c|)$ . Set  $\epsilon = 1/n^2$  and  $\delta = 1/3n$ . Let  $t$  equal the number of labeled examples and membership queries required by  $A$  for this setting of parameters.

It is well known that we can amplify the probability that  $c_{n-1}$  computes Permanent correctly to  $1 - 1/3n^2t$  by taking a majority vote of  $\text{poly}(t, n)$  independent copies of  $c_{n-1}$ .

Now we show how to obtain a pair  $(z_i, \text{Permanent}(z_i))$  using  $c_{n-1}$ . This will allow us to simulate the PAC learning algorithm to find  $c_n$ . To find the labels of the point  $z_i$  (recall

the label of  $z_i$  is  $\text{Permanent}(z_i)$ ), we apply Fact 2 and query  $c_{n-1}$  at the appropriate  $tn$  points. Applying a union bound we see that the probability that any query/random example  $z$  is mislabeled is less than  $1/3n$ . Hence with probability at least  $1 - \delta - 1/3n$  we obtain a hypothesis  $h$  that computes Permanent correctly on all but a  $1/n^2$  fraction of inputs. Applying Theorem 7 (the random self reducibility of the permanent), we obtain a randomized circuit  $c_n$  such that  $c_n$  computes Permanent correctly on each input with probability at least  $1 - 1/n$ .

Applying the union bound over all  $n$  iterations of this process, we see that the probability we fail to construct  $c_n$  properly is at most  $n\delta + 1/3$ . Since  $\delta < 1/3n$ , with probability at least  $2/3$  we have obtained a randomized circuit for computing the permanent that is correct on every input with probability at least  $1 - 1/n$ . The lemma follows. ■

We can now restate and prove our main theorem showing PAC learning algorithms imply lower bounds against BPEXP. Since we do not know if  $\text{BPEXP} \subseteq \text{P/poly}$  implies  $\text{BPEXP} = \text{EXP}$ , we must use the fact that doubly-exponential time contains languages with superpolynomial circuit complexity:

**Theorem 4** *Let  $C$  be any circuit representation class. Assume that  $C$  is PAC learnable (with membership queries) with respect to the uniform distribution in polynomial-time. Then  $\text{BPEXP} \not\subseteq \text{P/poly}[C]$ .*

**Proof:** First assume that  $\text{EXP} \subseteq \text{P/poly}[C]$  as otherwise we have nothing to prove. Then applying Theorem 9 we have  $\text{EXP} = \text{PSPACE} = \text{P}^{\#P}$ . Thus Permanent is complete for EXP, and any EXP complete language  $L$  can be reduced to Permanent in polynomial-time. Applying Lemma 5 we have that Permanent is in BPP and thus  $\text{EXP} = \text{BPP}$ . This implies that  $\text{EE} \subseteq \text{BPEXP}$ . From Theorem 6, we know that EE contains a language not computable by  $C$ . Hence  $\text{BPEXP} \not\subseteq \text{P/poly}[C]$ . ■

Since learnability in the exact model with membership queries implies learnability in the PAC model with membership queries we also have the following lemma:

**Lemma 6** *Let  $C$  be a circuit representation class. Assume that the Permanent is computable in  $\text{P/poly}[C]$ . If  $C$  is Exactly learnable from membership and equivalence queries in randomized polynomial-time then the Permanent is in BPP.*

Applying the same proof for Theorem 4 but using Lemma 6 we obtain

**Theorem 11** *Let  $C$  be any circuit representation class. Assume that  $C$  is Exactly learnable in randomized polynomial-time. Then  $\text{BPEXP} \not\subseteq \text{P/poly}[C]$ .*

N.V. Vinodchandran has proved a statement similar to Lemma 6 for SAT rather than Permanent in the “teaching assistant” model of deterministic exact learning [Vin04].

Buhrman et al. [BFT98] proved that  $\text{MA}_{\text{EXP}}$  contains languages with superpolynomial size circuits; this is still the smallest class known to contain languages of superpolynomial circuit complexity. Theorem 4 shows that PAC learnability of a circuit class such as depth-two threshold circuits, even under the uniform distribution with membership queries, would imply a new lower bound. To contrast this with the work of Kabanets and Impagliazzo [KI04], they

showed that under the assumption that there exists a non-deterministic subexponential time algorithm for polynomial identity-testing, NEXP contains languages with superpolynomial *arithmetic* circuit complexity.

## 5 Improved Lower Bounds from Learning Arithmetic Circuits

Several researchers have given deterministic, exact learning algorithms for various classes of algebraic models of computation including read-once arithmetic formulas, algebraic branching programs, and arithmetic circuits [BHH95, BTW98, KS03]. Our main theorem applies to these models of computation as well. In fact, if we restrict the output hypothesis to be a polynomial-size arithmetic circuit or formula (or any hypothesis equal to a multivariate polynomial of degree bounded by  $2^{n^{O(1)}}$ ), then we obtain a finer set of separations. We note that many exact learning algorithms for algebraic concepts do indeed output a hypothesis equal to polynomials of bounded degree (for example Bshouty et al. [BBB<sup>+</sup>96] or Klivans and Shpilka [KS03]).

We require the following lemma:

**Lemma 7** *Assume that polynomial-size arithmetic circuits are exactly learnable in ZPP and the output hypothesis is an arithmetic circuit. Then if Permanent is computed by polynomial-size arithmetic circuits, Permanent is in ZPP<sup>RP</sup>.*

**Proof:** We iteratively construct circuits  $c_1, \dots, c_n$  such that  $c_i$  computes the permanent on inputs of length  $i$ . At stage  $i$ , given  $c_{i-1}$ , membership queries are simulated as in the proof of Theorem 1. In order to find a counterexample, however, we cannot use an NP oracle. Instead, we use the fact that our output hypothesis is a polynomial-size arithmetic circuit. Lemma 3 shows that the problem of determining whether an arithmetic circuit computes permanent exactly is computable in polynomial-time given access to an RP oracle. If we discover that our hypothesis is correct we stop. Otherwise, we know that our hypothesis is not equal to the permanent.

At this point we need to compute a counterexample, namely a point  $z$  such that our current candidate for  $h(z)$  does not equal Permanent of  $z$  where  $h$  is our current candidate for  $c_i$ . Since  $h$  is a polynomial-size arithmetic circuit it is equal to a polynomial of degree at most  $2^{O(n^a)}$  for some fixed constant  $a$  (see Section 2.1). Thus a random  $z$  will be a counterexample if  $z$  is chosen from a field  $F$  of size  $2^{O(n^{2a})}$ . Thus, our algorithm chooses a random  $z \in F$  and checks if  $h(z)$  does not equal Permanent on  $z$  (the label for  $z$  can be computed by applying Fact 2 and using  $c_{n-1}$ ). With high probability we will obtain such a  $z$ . Due to the correctness of the learning algorithm, we will be assured of a correct hypothesis after at most  $n^{O(1)}$  counterexamples. At each stage  $i$  the probability of failure can be amplified to be less than  $1/3n$  so that the overall probability of failure (cumulative over all  $n$  stages) will be less than  $1/3$ . ■

We can now show that learning arithmetic circuits (by arithmetic circuits) yields a lower bound against ZPEXP<sup>RP</sup>. Since we know of no collapse theorems for complexity classes such as ZPEXP<sup>RP</sup> (or even EXP<sup>RP</sup>) we need to use a different argument than in the proof of Theorem 1:

**Theorem 12** *Let  $C$  be any representation class of arithmetic circuits. Assume that  $C$  is exactly learnable from membership and equivalence queries in randomized (zero-sided error) time  $\text{poly}(n)$  and that the output hypothesis is an arithmetic circuit. Then  $\text{ZPEXP}^{\text{RP}} \not\subseteq \text{P/poly}[C]$ .*

**Proof:** We may assume that 1) the Permanent is computable by circuits from  $C$  and 2)  $\text{EXP} \subseteq \text{P/poly}[C]$ , as otherwise there is nothing to prove. Notice that if  $\text{EXP} \subseteq \text{P/poly}[C]$  then  $\text{EXP} = \text{P}^{\#\text{P}}$  by Theorem 9 and Permanent is complete for EXP via a polynomial-time reduction. By Lemma 7, Permanent (and thus EXP) is in  $\text{ZPP}^{\text{RP}}$ . This implies that  $\text{EE} \subseteq \text{ZPEXP}^{\text{RP}}$ , but by Theorem 6, EE contains functions with superpolynomial circuit complexity. Hence  $\text{ZPEXP}^{\text{RP}}$  must also. ■

It is still an open problem as to whether polynomial-size arithmetic formulas are exactly learnable in polynomial-time; much progress has been made on restricted versions of this problem (for example [BHH95, BB98]). For the case of exactly learning arithmetic formulas (recall that no superpolynomial-lower bounds are known for this class) where the learner outputs a polynomial-size formula as its hypothesis, we can improve on Lemma 7:

**Lemma 8** *Assume that polynomial-size arithmetic formulas are exactly learnable in polynomial-time and the output hypothesis is an arithmetic formula. Then if Permanent is computed by polynomial-size arithmetic formulas, Permanent is in  $\text{P}^{\text{RP}}$ .*

**Proof:** The proof is similar to the proof of Lemma 7, except that we can *deterministically* construct counterexamples using an oracle for RP. This is because the hypothesis is a formula rather than a circuit, and, as discussed in Section 2.1, its degree as a polynomial is  $O(n^a)$  for some constant  $a$ . We can then choose a field  $F$  of size  $O(n^{2a})$  and substitute all values of  $F$  for  $x_1$ . For each substitution to  $x_1$ , query the RP oracle to determine if this restricted polynomial is non-zero. For some value  $a = x_1 \in F$ , the restricted polynomial must be non-zero. We can repeat this process to find an appropriate setting for  $x_2, \dots, x_n$ . Since the size of  $F$  is at most  $n^{O(1)}$ , we will have found a polynomial-time algorithm for computing the permanent using an oracle for RP. ■

Following the same outline for the proof of Theorem 12 but using Lemma 8 instead of Lemma 7 we obtain the following theorem:

**Theorem 13** *Let  $C$  be any representation class of arithmetic formulas. Assume that  $C$  is exactly learnable from membership and equivalence queries in polynomial time and that the output hypothesis is an arithmetic formula. Then  $\text{EXP}^{\text{RP}} \not\subseteq \text{P/poly}[C]$ .*

Kabanets and Impagliazzo [KI04] have proved that there exists a function  $f \in \text{NEXP}^{\text{RP}}$  that has superpolynomial arithmetic circuit complexity. Note that  $\text{NEXP}^{\text{RP}}$  is not known to be contained in either  $\text{EXP}^{\text{RP}}$  or  $\text{ZPEXP}^{\text{RP}}$ .

## 6 Conclusions and Open Problems

One interpretation of our results is that we have given added motivation for trying to develop learning algorithms for very restricted concept classes, as they would settle important and

difficult questions in computational complexity theory. Techniques from circuit lower bounds have figured prominently in the development of powerful learning algorithms in the past (e.g., Linial et al. [LMN93]), yet we are unaware of applications from learning theory to circuit lower bounds. An interesting open problem is to show that randomized subexponential time Exact (and PAC) learning algorithms yield new circuit lower bounds.

## References

- [ABF<sup>+</sup>04] M. Alekhnovich, M. Braverman, V. Feldman, A. Klivans, and T. Pitassi. Learnability and automatizability. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.
- [AK01] V. Arvind and J. Kobler. On pseudorandomness and resource-bounded measure. *TCS: Theoretical Computer Science*, 255, 2001.
- [Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.
- [AW07] S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. Manuscript, 2007.
- [BB98] N. Bshouty and D. Bshouty. On interpolating arithmetic read-once formulas with exponentiation. *JCSS: Journal of Computer and System Sciences*, 56, 1998.
- [BBB<sup>+</sup>96] A. Beimel, F. Bergadano, N. Bshouty, E. Kushilevitz, and S. Varricchio. On the applications of multiplicity automata in learning. In *Proceedings of the Thirty-Seventh Annual Symposium on Foundations of Computer Science*, pages 349–358, 1996.
- [BF90] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proceedings of the 7th Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48. Springer, Berlin, 1990.
- [BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.
- [BFT98] H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *Proceedings of the 13th IEEE Conference on Computational Complexity*, pages 8–12. IEEE, New York, 1998.
- [BH92] H. Buhrman and S. Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In *Proceedings of the 12th Conference on the Foundations of Software Technology and Theoretical Computer Science*, volume 652 of *Lecture Notes in Computer Science*, pages 116–127. Springer, Berlin, Germany, 1992.
- [BHH95] N. Bshouty, T. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. *SICOMP: SIAM Journal on Computing*, 24, 1995.

- [BTW98] N. Bshouty, T. Tamon, and D. Wilson. On learning width two branching programs. *IPL: Information Processing Letters*, 65, 1998.
- [Gol78] E. A. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.
- [GZ97] O. Goldreich and D. Zuckerman. Another proof that BPP subseteq PH (and more). In *Electronic Colloquium on Computational Complexity*, number 97-045. 1997.
- [Hit07] J. Hitchcock. Online learning and resource-bounded dimension: Winnow yields new lower bounds for hard sets. *SIAM Journal on Computing*, 36(6):1696–1708, 2007.
- [HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.
- [IW01] R. Impagliazzo and A. Wigderson. Randomness vs. time: Derandomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, December 2001.
- [JKS02] J. Jackson, A. Klivans, and R. Servedio. Learnability beyond  $AC^0$ . In *Proceedings of the 34th ACM Symposium on Theory of Computing*, 2002.
- [Kan82a] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3):40–56, 1982.
- [Kan82b] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55:40–56, 1982.
- [Kha93] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of the Twenty-Fifth Annual Symposium on Theory of Computing*, pages 372–381, 1993.
- [KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.
- [KS03] A. Klivans and A. Shpilka. Learning arithmetic circuits via partial derivatives. In *COLT: Proceedings of the Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers, 2003.
- [KS06] A. Klivans and A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.
- [KV94] M. Kearns and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

- [Lip91] R. Lipton. New directions in testing. In J. Feigenbaum and M. Merritt, editors, *Distributed Computing and Cryptography*, volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 191 – 202. American Mathematical Society, Providence, 1991.
- [Lit88] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
- [LMN93] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM*, 40(3):607–620, 1993.
- [LSW00] W. Lindner, R. Schuler, and O. Watanabe. Resource-bounded measure and learnability. *MST: Mathematical Systems Theory*, 33, 2000.
- [MVW99] P.B. Miltersen, N.V. Vinodchandran, and O. Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *COCOON: Annual International Conference on Computing and Combinatorics*, 1999.
- [PV88] L. Pitt and L. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.
- [San07] R. Santhanam. Circuit lower bounds for merlin-arthur classes. In *STOC*, pages 275–283. ACM, 2007.
- [Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.
- [Val79] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.
- [Val84] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
- [Vin04] N.V. Vinodchandran. Learning DNFs and circuits using teaching assistants. In *COCOON: Annual International Conference on Computing and Combinatorics*, 2004.
- [WG94] O. Watanabe and R. Gavaldà. Structural analysis of polynomial-time query learnability. *MST: Mathematical Systems Theory*, 27, 1994.