On Learning, Lower Bounds and (un)Keeping Promises

Ilya Volkovich[⋆]

Computer Science Department and Center for Computational Intractability, Princeton University, Princeton NJ
ilyav@cs.princeton.edu

Abstract. We extend the line of research initiated by Fortnow and Klivans [6] that studies the relationship between efficient learning algorithms and circuit lower bounds. In [6], it was shown that if a Boolean circuit class $\mathcal{C}$ has an efficient *deterministic* exact learning algorithm, (i.e. an algorithm that uses membership and equivalence queries) then $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{P/poly}[\mathcal{C}]$[1]. Recently, in [14] $\mathsf{EXP}^{\mathsf{NP}}$ was replaced by $\mathsf{DTIME}(n^{\omega(1)})$. Yet for the models of *randomized* exact learning or Valiant's PAC learning, the best result so far is a lower bound against BPEXP (the exponential-time analogue of BPP). In this paper, we derive stronger lower bounds as well as some other consequences from *randomized* exact learning and PAC learning algorithms, answering an open question posed in [6] and [14]. In particular, we show that

1. If a Boolean circuit class $\mathcal{C}$ has an efficient *randomized* exact learning algorithm or an efficient PAC learning algorithm[2] then $\mathsf{BPTIME}(n^{\omega(1)})/1 \not\subseteq \mathsf{P/poly}[\mathcal{C}]$.
2. If a Boolean circuit class $\mathcal{C}$ has an efficient *randomized* exact learning algorithm then no strong pseudo-random generators exist in $\mathsf{P/poly}[\mathcal{C}]$.

We note that in both cases the learning algorithms need not be *proper*[3]. The extra bit of advice comes to accommodate the need to keep the promise of bounded away probabilities of acceptance and rejection. The exact same problem arises when trying to prove lower bounds for BPTIME or MA [3,7,16,20]. It has been an open problem to remove this bit. We suggest an approach to settle this problem in our case. Finally, we slightly improve the result of [5] by showing a subclass of MAEXP that requires super-polynomial circuits. Our results combine and extend some of the techniques previously used in [6,14] and [20].

⋆ Research partially supported by NSF Award CCF 0832797. Full version available at http://eccc.hpi-web.de/report/2014/058/.

[1] $\mathsf{P/poly}[\mathcal{C}]$ stands for the class of all languages that can be computed by polynomial-size circuits from the class $\mathcal{C}$.

[2] In fact, our result hold even for a more general model of PAC learning with membership queries.

[3] A learning algorithm is proper if it outputs a hypothesis from the class it learns.

© Springer-Verlag Berlin Heidelberg 2014

1 Introduction

Revealing a hidden function from a set of examples is a natural and basic problem. As in any other problem, identifying the obstacles along your trail is a fundamental task in achieving your goal. In this paper, we continue to identify the obstacles to efficient learnability following the line of research initiated by Fortnow and Klivans [6]. Several results [23,13,15,8] exhibit a two-way connection between learning and cryptography: basing the hardness of learning on cryptography and constructing cryptographic primitives based on hardness of learning. In this paper we identify the obstacles in the form of circuit lower bounds and relationships between complexity classes.

Angluin's model of Exact Learning [2] consists of a (computationally bounded) learner and a (all-powerful) teacher. The learner's goal is to output a target function f from a given function class $\mathcal{C}$. To do so, the learner is allowed to query the value $f(\bar{x})$ on any input $\bar{x}$ (membership query). In addition, the learner is also allowed to propose a hypothesis $\hat{f}$ and ask the teacher whether it is equivalent to f (equivalence query). If this is indeed the case, the learner has achieved his goal. Otherwise, the teacher presents the learner with a counterexample $\bar{a}$ for which $f(\bar{a}) \neq \hat{f}(\bar{a})$. We say that a function class $\mathcal{C}$ is *exactly learnable* if there exists a learner which given any $f \in \mathcal{C}$, in time polynomial in n and $|f|$ (the size of f in the representation scheme) outputs a hypothesis $\hat{f}$ such that $f(\bar{x}) = \hat{f}(\bar{x})$ for all $\bar{x}$, using membership and equivalence queries. In the *randomized* Exact Learning model the learner is allowed to toss coins and, given any $f \in \mathcal{C}$, it must output a correct hypothesis, with high probability. We say that a class $\mathcal{C}$ is *exactly learnable with high probability* (w.h.p) if there exists a learner which given any $f \in \mathcal{C}$, in time polynomial in n and $|f|$ w.h.p outputs a hypothesis $\hat{f}$ such that $f(\bar{x}) = \hat{f}(\bar{x})$ for all $\bar{x}$, using membership and equivalence queries.

In Valiants PAC learning model [23], we (again) have a (computationally bounded) learner that is given a set of samples of the form $(\bar{x}, f(\bar{x}))$ from some fixed function $f \in \mathcal{C}$, where $\bar{x}$ is chosen according to some unknown distribution D. Given $\varepsilon > 0$ and $\delta > 0$, the learner's goal is to output, with probability $1 - \varepsilon$ a hypothesis $\hat{f}$ such that $\hat{f}$ is a $1 - \delta$ close to f under D. We say that a function class $\mathcal{C}$ is PAC *learnable* if there exists a learner which given any $f \in \mathcal{C}$, $\varepsilon > 0$ and $\delta > 0$ in time polynomial in $n, 1/\varepsilon, 1/\delta, |f|$ outputs a hypothesis as required. In a more general model, the learner is allowed membership queries (as in the exact learning model). In this case, we say that $\mathcal{C}$ is PAC *learnable* with *membership queries.*

A learning algorithm is said to be *proper* if it outputs a hypothesis from the class it learns. It is known [1] that both randomized and exact learners can be used to obtain a PAC learner with membership queries. Thus, hardness results for the PAC learning model entail hardness results for the randomized exact learning model. In [6], Fortnow and Klivans have shown that if a Boolean circuit class $\mathcal{C}$ has an efficient *deterministic* exact learning algorithm, then $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{P/poly}[\mathcal{C}]$. Subsequently, Harkins and Hitchcock [10] removed the NP oracle replacing $\mathsf{EXP}^{\mathsf{NP}}$ by EXP, using techniques from resource bounded measure. Further improvement was shown in [14] where $\mathsf{EXP}^{\mathsf{NP}}$ was replaced by $\mathsf{DTIME}(n^{\omega(1)})$ using simpler, diagonalization type of techniques. However, given an efficient *randomized* exact learning algorithm or a PAC learner, the above techniques fail to produce a hard function. Indeed, the best known result so far [6,14] is a lower bound against BPEXP (the exponential-time analogue of BPP), therefore leaving an open question for improvement. In this paper we derive stronger (matching) lower bounds as well as some other consequences from *randomized* exact learning and PAC learning algorithms, answering the open question.

1.1 Our Results

In this section we go briefly over our results comparing them with the previous ones. We now present the first result of the paper, which gives lower bounds against $\mathsf{BPTIME}(n^{\omega(1)})/1$ and $\mathsf{PromiseBPTIME}(n^{\omega(1)})$ assuming an efficient randomized exact learner or an efficient PAC learner with queries. We note that the learning algorithm need not be proper. .

Theorem 1. *Let $\mathcal{C}$ be a circuit class. If $\mathcal{C}$ is exactly learnable w.h.p or is* PAC *learnable with membership queries, then* $\mathsf{BPTIME}(n^{\omega(1)})/1 \not\subseteq \mathsf{P/poly}[\mathcal{C}]$ *and* $\mathsf{PromiseBPTIME}(n^{\omega(1)}) \not\subseteq \mathsf{P/poly}[\mathcal{C}]$.

This matches the recent result of [14], where a lower bound against $\mathsf{DTIME}(n^{\omega(1)})$ was produced assuming a deterministic efficient learner. In fact, the results of [14] also imply fixed polynomial-size circuit lower bounds against P. That is, for each k, $\mathsf{P} \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$, where $\mathsf{SIZE}(n^k)[\mathcal{C}]$ stands for languages accepted by size $\mathcal{O}(n^k)$ circuits from the class $\mathcal{C}$. We match this result as well.

Theorem 2. *Let $\mathcal{C}$ be a circuit class. If $\mathcal{C}$ is exactly learnable w.h.p or is* PAC *learnable with membership queries, then for any $k \geq 1$:* $\mathsf{BPP}/1 \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$ *and* $\mathsf{PromiseBPP} \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$.

Next, we show that efficient randomized exact learner for a circuit class $\mathcal{C}$ gives rise to a P/poly-natural property useful against $\mathsf{P/poly}[\mathcal{C}]$. Following the celebrated result of Razborov & Rudich [19] (and its extensions) this implies that no strong pseudo-random generators exist in $\mathsf{P/poly}[\mathcal{C}]$. For the case of $\mathcal{C} = \mathsf{P/poly}$ or even a smaller class of TC^0 of constant-depth threshold functions such an outcome is very unlikely [18]. Again, the learning algorithm need not be proper.

Theorem 3. *Let $\mathcal{C}$ be a circuit class. If $\mathcal{C}$ is exactly learnable w.h.p then no strong pseudo-random generators exist in* $\mathsf{P/poly}[\mathcal{C}]$.

In a similar fashion to the previous lower bounds and hierarchy theorems for the "bounded" probabilistic classes [3,7,16] and MA [20] we require an extra bit of advice to keep the promise of bounded away probabilities. It has been an open problem to remove this bit. We suggest an approach to settle this problem by trying to "unkeep" the promise using fixed oracles. More specifically, recall that $\mathsf{BPP} = \mathsf{P}^{\mathsf{PromiseBPP}} = \mathsf{P}^{\mathrm{CA}}$ [4]. That is, a language L is in BPP iff it can be decided by a P machine with an oracle to CA. Note, thought, that for some queries the answer of the CA oracle can be arbitrary (e.g. for balanced circuits). We eliminate the need of an advice bit for the cases when oracle is fixed.

Theorem 4. *Let O be an oracle consistent with* CA*. That is, O accepts circuit from* CA_{YES} *and rejects circuit from* CA_{NO}*. Let $\mathcal{C}$ be a circuit class. If $\mathcal{C}$ is exactly learnable w.h.p or is* PAC *learnable with membership queries, then for any $k \geq 1$: $\mathsf{P}^O \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$.*

In [20], Santhanam proved lowers bounds for $\mathsf{MA}/1$ unconditionally. In fact, our conditional results use several techniques from this paper (partially described in Section 1.2). Applying the same idea unconditionally and recalling that $\mathsf{MA} = \mathsf{NP}^{\mathsf{PromiseBPP}} = \mathsf{NP}^{\mathrm{CA}}$ (see e.g. [9]) we obtain the following (unconditional) result:

Theorem 5. *Let O be an oracle consistent with* CA*. That is, O accepts circuit from* CA_{YES} *and rejects circuit from* CA_{NO}*. Then for any $k \geq 1$: $\mathsf{NP}^O \not\subseteq \mathsf{SIZE}(n^k)$.*

The smallest complexity class for which unconditional super-polynomial lower bounds are known is MAEXP [5,17]. We show that this is still true for a subclass of MAEXP, as $\mathsf{MAEXP} = \mathsf{NEXP}^{\mathsf{PromiseBPP}}$. Due to space limitation the proof of this theorem is omitted.

Theorem 6. *There exists a language* $\mathrm{LA} \in \mathsf{PromiseBPP}$ *s.t.* $\mathsf{NEXP}^{\mathrm{LA}} \not\subseteq \mathsf{P/poly}$.

1.2 Our Techniques and Ideas

We now describe our main techniques and ideas.

Learning a "Conveniently Hard" Language. As in various previous papers dealing with conditional and unconditional lower bounds [11,12,6,20,14], we require a "conveniently hard" language L. That is, a language L that possesses some "nice" structural properies (downward self-reducibility, self-testability and self-correctability) and yet can be used to compute a "hard" function (for a formal definition see Definition 6). The "nice" properties of L make it easily learnable given an efficient learner. More specifically, they allow answering the learner's queries efficiently. We now describe the idea in a nutshell. We combine several techniques from [11,6,20,14] and extend them.

Given an efficient learner for a circuit class $\mathcal{C}$, the idea is to learn a non-uniform circuit $C \in \mathcal{C}$ for L and then use C to compute the hard function within L. This puts a hard function in BPP and, obviously, can be carried out only if L is computable by a small-sized family of circuits in $\mathcal{C}$. Similar idea appeared in

[4] CA (Circuit Approximation) is the natural $\mathsf{PromiseBPP}$-complete problem $\mathrm{CA}_{YES} = \left\{C \mid \Pr_{\bar{a}\in\{0,1\}^n}[C(\bar{a}) = 1] \geq 3/4\right\}$ and $\mathrm{CA}_{NO} = \left\{C \mid \Pr_{\bar{a}\in\{0,1\}^n}[C(\bar{a}) = 1] \leq 1/4\right\}$.

[11]. The other option is that L requires large circuits from $\mathcal{C}$. At this point, the approach taken in [6,14] is to combine both hardness results into a single lower bound. This results in a lower bound against BPEXP. We take the approach suggested in [20]: learn a padded version of L. Intuitively, padding allows us to increase the input size and thus allowing more learning time, without actually increasing the "real" size of the input. We try different amounts of padding until the learner has "enough running time" to learn L. Note, that the learning algorithm is guaranteed to succeed w.h.p only given enough running time/samples. Consequently, when executed prior to reaching the "right" amount of padding the acceptance/rejection probabilities of the algorithm can be arbitrarily close to half. Yet in order to remain within the framework of BPP, the algorithm must keep the promise of bounded away acceptance and rejection probabilities. The solution of [7,16,20] was to add one bit of advice indicating whether or not we have reached the necessarily amount of padding. To finish the argument, we show that small circuits for a sufficiently padded version of L imply small circuits for L itself.

(un)Keeping the Promise. For the sake of simplicity, we describe here the intuition behind removing the advice from $\mathsf{MA}/1$. We follow along the lines of the previous section with one change: instead of using the learning algorithm to a learn a non-uniform circuit C for L, we "guess" it. Next, we need to verify that C indeed computes L via a probabilistic procedure referred to as "self-testability" (Property 2 Definition 6). This procedure can detect an error in C only if C if "far enough" from L. Thus, we are dealing with an oracle call to $\mathsf{PromiseBPP}$, which can replaced by an oracle call to CA. However, the answer of the oracle is undefined for circuits that are "close" to L but not equal L. We suggest to circumvent this problem by replacing an oracle call to CA with a fixed oracle O that is consistent with CA. Given a fixed oracle O every call has a defined answer. Repeating the previous reasoning we establish a hard language in NP^O. Yet, for each O it might be a different language. Our glint of hope comes come from the following relation, which follows from the definition of $\mathsf{NP}^{\mathsf{PromiseBPP}}$ (see e.g. [4]): $\mathsf{MA} = \mathsf{NP}^{\mathsf{PromiseBPP}} \triangleq \bigcap_{O \text{ is consistent with CA}} \mathsf{NP}^O$. If one could show that a hard language in each of the NP^O terms on the RHS implies a hard language in their intersection, we would be done. We formalize this approach in Section 5.

1.3 Organization

We start by some basic definitions and notation in Section 2. In Section 3 we give our main result showing that efficient randomized learning algorithms entail circuit lower bounds, proving Theorems 1 and 2. In Section 4 we prove Theorem 3 show that for several circuit classes the very existence of efficient randomized exact learning algorithm "bumps" into the natural barier of Razborov & Rudich. In Section 5 we propose an approach for removing the extra bit of advice. Finally, we discuss some open questions in Section 6.

2 Preliminaries

Definition 1 (Functions, Circuits, Languages). *In this paper we deal with Boolean functions, that is* $f : \{0,1\}^n \to \{0,1\}$. *Let* $f, g : \{0,1\}^n \to \{0,1\}$. *We define the* relative distance $\Delta(f,g) \overset{\Delta}{=} \Pr_{\bar{a}\in\{0,1\}^n}[f(\bar{a}) \neq g(\bar{a})]$. *For* $\varepsilon \geq 0$*, we say that* f *is* ε-close *to* g *if* $\Delta(f,g) \leq \varepsilon$*, otherwise we say that* f *is* ε-far *from* g. *Let* $L \subseteq \{0,1\}^*$ *be a language. We denote by* $L|_n$ *the set of the strings of length* n *in* L. *We say that* L *has circuits of size* $a(n)$ *and denote it by* $L \in \mathsf{SIZE}(a(n))$ *if for every* $n \geq 1$ $L|_n$ *can be computed by a Boolean circuit of size* $\mathcal{O}(a(n))$. *A* circuit class $\mathcal{C}$ *is a subset of all Boolean circuits (e.g. circuits with AND,OR and NOT gates,* $\mathsf{AC}^0, \mathsf{ACC}, \mathsf{TC}^0, \mathsf{NC}^2$ *etc.). We assume that the representation in* $\mathcal{C}$ *is chosen in way that a size* s *circuit can be described using* poly(s) *bits. In addition, given a circuit* $C \in \mathcal{C}$ *of size* s *the circuit* $C|_{x_i=b}$ *is also in* $\mathcal{C}$ *and of size at most* s*, when* $C|_{x_i=b}$ *is the circuit resulting from* C *by fixing the variable* x_i *to the bit* $b \in \{0,1\}$. *We denote by* $\mathsf{SIZE}(a(n))[\mathcal{C}]$ *the set of languages having circuits of size* $\mathcal{O}(a(n))$ *from the class* $\mathcal{C}$.

A Promise Problem is a relaxation of a language. Formally:

Definition 2 (Promise Problems). $\Pi = (\Pi_{YES}, \Pi_{NO})$ *is a* promise problem *if* $\Pi_{YES} \cap \Pi_{NO} = \emptyset$. *We say that a language* O *is* consistent *with* Π *iff* $x \in \Pi_{YES} \implies x \in O$ *and* $x \in \Pi_{NO} \implies x \notin O$. *The containment of* O *outside of* $\Pi_{YES} \uplus \Pi_{NO}$ *can be arbitrary.*

Let Π be a promise problem and let M be a deterministic (resp. nondeterministic) polynomial-time Turing machine with an oracle access to Π. Consider a (oracle) language O consistent with Π. By definition (see e.g. [4]), M's language should not depend on the answers of the oracle when a query $q \notin \Pi_{YES} \uplus \Pi_{NO}$ is made. Consequently, $\mathsf{P}^{\Pi} \subseteq \mathsf{P}^{O}$ (resp. $\mathsf{NP}^{\Pi} \subseteq \mathsf{NP}^{O}$). It turns out that the following holds as well, and in fact can be considered as an alternative definition for classes of languages computed by Turing machines with oracles to promise problems.

Definition 3 (Promise Problems as Oracles)

$$\mathsf{P}^{\Pi}\ (\textit{resp.}\ \mathsf{NP}^{\Pi}) \overset{\Delta}{=} \bigcap_{O \textit{ is consistent with } \Pi} \mathsf{P}^{O}\ (\textit{resp.}\ \mathsf{NP}^{O})$$

For more details and discussion see e.g. [4].

Definition 4 (Lower Bounds for Promise Problems). *Let* $\mathcal{C}$ *be a circuit class and* $f(n)$ *be a function. Then* $\Pi \notin \mathsf{SIZE}(f(n)) \iff \forall O$ *consistent with* Π*:* $O \notin \mathsf{SIZE}(f(n))$.

Definition 5 (Complexity Classes). *A language* L *is in* BPP *if there exists a polynomial-time machine* $M(x,r)$ *such that:* $x \in L \implies \Pr_r[M(x,r) = 1] \geq 3/4, x \notin L \implies \Pr_r[M(x,r) = 1] \leq 1/4$. *A language* $L \in \mathsf{BPP}/1$ *if in addition the machine requires an auxiliary* advice *bit* b_n *for each input of length* n. *We note that given the complement advice bit* $\bar{b}_n$ *the machine is not guaranteed to* *preform correctly. In particular, given $\bar{b}_n$ as the advice bit there is no promise for bounded away acceptance and rejection probabilities. Other standard complexity classes include:* $\mathsf{P}, \mathsf{PSPACE}, \mathsf{RP}, \mathsf{NP}, \mathsf{MA}$.

For the core of our proofs we require a *conveniently hard* language. Intuitively, it is language that has "nice" structural properies and yet can be used to compute functions that require "large" circuits. It is not hard to see that every conveniently hard language is computable in PSPACE. In [22] a conveniently hard PSPACE-complete language was constructed via arithmezation of the proof that $\mathsf{PSPACE} = \mathsf{IP}$ [21]. In their construction, all the "nice" structural properties follow from the properties of low-degree polynomials and of TQBF. The Embedded Hardness (Part 4) is due to the fact that given a circuit class $\mathcal{C}$, in $\mathsf{DSPACE}(\mathrm{poly}(s))$ one can diagonalize against all the circuits of size s from $\mathcal{C}$, thus obtaining a language that can not be computed by any size s circuit. Formally:

Definition 6. *We say that a language L is* conveniently hard *if it satisfies:*

1. ***Downward Self-Reducibility:*** *we say that a language L is* downward-self-reducible *if there is a deterministic polynomial-time algorithm* $\mathsf{COMPUTE}$ *such that for all $n \geq 1$:* $\mathsf{COMPUTE}^{L|_{n-1}} = L|_n$.
2. ***Self-Testability:*** *we say that a language L is* self-testable *if there is a probabilistic polynomial-time algorithm* TEST *such that for any Boolean function $f : \{0,1\}^n \to \{0,1\}$:*
   - *If $f = L|_n$ then* $\Pr[\mathsf{TEST}^f = 1] = 1$.
   - *If $\Delta(f, L|_n) > 1/n$ then* $\Pr[\mathsf{TEST}^f = 1] \leq 2^{-10n}$.
3. ***Self-Correctability:*** *we say that a language L is* self-correctable *if there is a probabilistic polynomial-time algorithm* $\mathsf{CORRECT}$ *such that, for any Boolean function $f : \{0,1\}^n \to \{0,1\}$ it holds that if $\Delta(f, L|_n) \leq 1/n$ then for all $\bar{x} \in \{0,1\}^n$:* $\Pr[\mathsf{CORRECT}^f(\bar{x}) \neq L|_n(\bar{x})] \leq 2^{-10n}$.
4. ***Embedded Hardness:*** *we say that a language L has an* Embedded Hardness *if for every circuit class $\mathcal{C}$ and $k \geq 1$:* $\mathsf{P}^L \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$.

The following is immediate from the definition:

Observation 1. *Let C be an n-variate circuit of size s such that $\Delta(C, L|_n) \leq 1/n$. Then there exists an n-variate circuit C' of size* $\mathrm{poly}(s, n)$ *such that $\Delta(C, L|_n) = 0$. Moreover, C' can be obtained from C in polynomial time w.h.p.*

3 Lower Bounds from Randomized Learning Algorithms

In this section we prove Theorems 1 and 2. Let L be the conveniently hard PSPACE-complete language of [22]. Following and extending definitions from [20] we define padded versions of L. For simplicity, throughout the section we fix a circuit class $\mathcal{C}$. We will assume w.l.o.g that $\mathsf{P} \subseteq \mathsf{P/poly}[\mathcal{C}]$ (otherwise there is nothing to prove). As $L \in \mathsf{PSPACE} \subseteq \mathsf{EXP}$, by a translation argument there exists $d \geq 1$ such that $L \in \mathsf{SIZE}(2^{n^d})$.

Definition 7 (Padded Languages)
For $r \geq 1$ let $s(r)$ denote the size of the smallest circuit from $\mathcal{C}$ that computes $L|_r$. By the preceding discussion $s(r)$ is well-defined and in particular $s(r) = \mathcal{O}(2^{r^d})$. Let $t(w) : \mathbb{N} \to \mathbb{N}$ be a constructible function. We define the padded version of L:

$$L'_{t(\cdot)} = \left\{ 1^m x \;\middle|\; \begin{array}{l} 1)\ m \text{ is power of } 2. \\ 2)\ r \triangleq |x| \leq m. \\ 3)\ x \in L. \\ 4)\ s(r) \leq t(m). \end{array} \right\}$$

Remark: *Condition 4 can be restated as: there exists a circuit of size $t(m)$ that computes $L|_r$. In addition, observe that for $L'_{t(\cdot)}$ each input has a unique interpretation.*

The main property of the padded languages is that sufficiently small circuits for $L'_{t(\cdot)}$ can be used to construct small circuits for L.

Lemma 1. *Let $k \geq 1$ and $t(w) = \Omega(w^{2k})$. Suppose $L'_{t(\cdot)} \in \mathsf{SIZE}(n^k)$. Then $s(r) = \mathcal{O}(r^{2k})$.*

Proof. For $n \geq 1$ let C'_n be a circuit for $L'_{t(\cdot)}|_n$. Let $r \geq 1$. We will now construct a circuit that computes $L|_r$. Take m to be a minimal power of 2 such that $r \leq m$ and $s(r) \leq t(m)$. As $t(w) = \Omega(w^{2k})$, we have that $t(m) \geq \alpha \cdot m^{2k}$, for some $\alpha > 0$. Hence, it must be the case that $m \leq 2 \cdot \alpha^{-1/2k} \cdot s(r)^{1/2k} + 2r$. Now, set $n = r + m$ and consider the circuit $\tilde{C}$ resulting from C'_n when we hardwire the lower m bits of the input to 1^m. By definition, $\tilde{C}$ computes $L|_r$ and there exists $\beta > 0$ such that $\tilde{C}$ is of size at most $\beta \cdot n^k = \beta \cdot (r+m)^k \leq \beta \cdot (3r + 2 \cdot \alpha^{-1/2k} \cdot s(r)^{1/2k})^k \leq \beta \cdot \sqrt{\alpha} \cdot 6^k \cdot r^k \cdot \sqrt{s(r)}$. By recalling the definition of $s(r)$ we get that $s(r) \leq \gamma \cdot r^k \cdot \sqrt{s(r)}$ which implies $s(r) = \mathcal{O}(r^{2k})$.

We now give the main result of this section.

Proof (of Theorem 2). Let $\mathcal{A}$ be a PAC learner for $\mathcal{C}$. Fix k. Consider two cases.

Case 1: $L \in \mathsf{P/poly}[\mathcal{C}]$. We will show that $\mathsf{P}^L \subseteq \mathsf{BPP}$ and hence $\mathsf{BPP} \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$. For this purpose will use $\mathcal{A}$ to learn circuits for L and then apply Property 4 of Definition 6 as follows:

– Begin with a lookup table $\tilde{C}_1 = C_1$ for $L|_1$.
– For $i \geq 2$, invoke $\mathcal{A}$ with $\varepsilon = 1/i^3$ and $\delta = 1/i$ to learn a circuit $\tilde{C}_i$ of size $s(i)$ for $L|_i$.
– Given a membership query for $L|_i$ invoke $\mathsf{COMPUTE}$ using $C_{i-1}(x)$.
– Set $C_i \triangleq \mathsf{CORRECT}^{\tilde{C}_i}$

Analysis. We claim that C_i computes $L|_i$ with probability at least $1 - 2^{-10i}$. By induction on i. Basis $i = 1$ is clear. Now assume that hypothesis holds for i. By definition, w.h.p $\mathcal{A}$ will output $\tilde{C}_{i+1}$ to be $1/i$ close to $L|_i$ using Property 1 of Definition 6 to answer membership queries. By Property 3 C_{i+1} will compute $L|_{i+1}$ with probability at least $1-2^{-10(i+1)}$. The total number of steps is $\mathrm{poly}(i)$ while each has an exponential probability error. Hence, for each i: $C_i \equiv L|_i$ w.h.p.

Running Time. Given an input of size n we learn the corresponding circuits of sizes $s(1), \ldots, s(n)$ which is $\mathrm{poly}(n)$ by assumption. In addition, all the algorithms are polynomial-time.

Case 2: $L \notin \mathsf{P/poly}[\mathcal{C}]$. Set $t(w) = w^{2k}$. We show that $L'_{t(\cdot)} \in \mathsf{BPP}/1$ and conclude that in this case $L'_{t(\cdot)} \notin \mathsf{SIZE}(n^k)[\mathcal{C}]$. We follow the learning scheme described in the previous case to learn circuits for L with two changes: first, since each input of $L'_{t(\cdot)}$ has a unique interpretation, we could use the advice bit to determine whether $s(r) \leq t(m)$. If the advice bit is 0, we reject. Otherwise, we carry on with flow of the scheme barring a second change: invoke $\mathcal{A}$ to learn circuits $\tilde{C}$ of size $t(m)$ and use the resulting circuit C_r to decide if $x \in L$. Now, suppose that $L'_{t(\cdot)} \in \mathsf{SIZE}(n^k)$. By Lemma 1 we get that $s(r) = \mathcal{O}(r^{2k})$, which contradicts the fact that $L \notin \mathsf{P/poly}[\mathcal{C}]$.

The proof of Theorem 1 is essentially the same. We leave it as an exercise for the reader. To complete the picture, recall that a randomized exact learner can be used to obtain a PAC learner with membership queries [1] and observe that $\mathsf{BPTIME}\left(t(n)\right)/1 \notin \mathsf{SIZE}\left(f(n)\right)$ implies that $\mathsf{PromiseBPTIME}\left(t(n)\right) \notin \mathsf{SIZE}\left(f(n)\right)$ by adding the advice to the input.

4 Natural Property from Learning

We now describe the approach in more details. First, we recall some related definitions from [19].

Definition 8. *Let $\mathcal{P}$ be a property of Boolean functions. That is, a subset of all Boolean functions. Let Γ, Λ be complexity classes. We say that $\mathcal{P}$ is Γ-*natural *with density* δ_n *if there is a property $\mathcal{P}^* \subseteq \mathcal{P}$ such that the following holds:* ***(i) Constructivity:*** *given a function f by its truth table, $f \in \mathcal{P}^*$ can be decided in Γ.* ***(ii) Largeness:*** *for all n, $\mathcal{P}^*$ contains at least a δ_n fraction of all n-variate Boolean functions. We say that $\mathcal{P}$ is* ***useful*** *against Λ if for every family of Boolean functions $\{f_n\}_{n\geq 1} \subseteq \mathcal{P}$ there are infinitely many n-s such that $f_n \notin \Lambda$.*

The main results of [19] (and its extensions) states as follows:

Lemma 2 ([19] and Extensions). *Let $\mathcal{C}$ be a circuit class. If there exists a $\mathsf{P/poly}$-natural property with density $2^{-\mathcal{O}(n)}$ that is useful against $\mathsf{P/poly}[\mathcal{C}]$ then no strong pseudo-random generators exist in $\mathsf{P/poly}[\mathcal{C}]$.*

We show that we can turn an efficient randomized exact learner $\mathcal{A}$ for a circuit class $\mathcal{C}$ into a $\mathsf{P/poly}$-natural property useful against $\mathcal{C}$. First, we amplify the error probability and get a $\mathsf{P/poly}$ algorithm $\mathcal{A}'_s$ which succeeds in learning all the function computable by size s circuits from $\mathcal{C}$. Next, we define the property $\mathcal{P}$ as the set of all functions on which $\mathcal{A}'_s$ fails. For an appropriate choice of s the property is useful against $\mathsf{P/poly}[\mathcal{C}]$. Moreover, observe that if $\mathcal{A}'_s$ succeeds learning a function f then f must have a small circuit. A simple counting argument shows that the majority of Boolean functions require large circuits, thus $\mathcal{A}'_s$ succeeds only a small fraction of functions. Note that since the counting argument is valid for any circuit class, the learning algorithm $\mathcal{A}$ need not be proper. Due to space limitation we omit the formal proof of Theorem 3 from this version.

5 Towards Better Lower Bounds by Unkeeping Promises

The extra bit of advice in Theorems 1 and 2 as well as in the result of [20] (lower bounds for $\mathsf{MA}/1$) comes to accommodate the need to keep the promise of bounded away probabilities of acceptance and rejection required for the "bounded" probabilistic classes. As in other cases, removing this bit is an open problem. In this section we suggest an approach how to settle this problem in the case of conditional and unconditional lower bounds. As previously, we will use the conveniently hard PSPACE-complete language L of [22].

We raise the question whether lower bounds are preserved under consistency. More specifically, given a promise problem Π our hope is that one could show that the existence of hard language in each P^O (or NP^O) implies an existence hard language in their intersection, when O runs over all the consistent with Π languages. We now give a formal treatment to this intuition.

Definition 9. *Let $\mathcal{C}$ be a circuit class. We say that a promise problem Π is* deterministically *(resp.* nondeterministically*)* compact *w.r.t. $\mathcal{C}$ if:* P^Π *(resp.* NP^Π*)* $\subseteq \mathsf{SIZE}(f(n))[\mathcal{C}] \implies \exists O$ *consistent with Π s.t.* P^O *(resp.* NP^O*)* $\subseteq \mathsf{SIZE}(f(n)^{\mathcal{O}(1)})[\mathcal{C}]$.

Note that the promise problems that are languages are trivially compact for all circuit classes in both settings. We can extend the definition to classes of promise problems requiring compactness for each problem in the class. We now give the proofs starting with the conditional case. Recall that CA is the natural $\mathsf{PromiseBPP}$-complete problem of *Circuit Approximation* defined as $\mathrm{CA} \triangleq (\mathrm{CA}_{YES}, \mathrm{CA}_{NO})$ where $(\mathrm{CA}_{YES} = \{C \mid \Delta(C, \bar{0}) \geq 3/4\}, \mathrm{CA}_{NO} = \{C \mid \Delta(C, \bar{0}) \leq 1/4\})$.

Lemma 3. *Let $\mathcal{C}$ be a circuit class. Suppose that* $\mathsf{PromiseBPP}$ *is deterministically compact w.r.t. $\mathcal{C}$. If $\mathcal{C}$ is* PAC *learnable with membership queries, then for any $k \geq 1$:* $\mathsf{BPP} \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$.

Proof. Fix $k \geq 1$ and assume for a contradiction that $\mathsf{BPP} \subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$. As $\mathsf{P}^{\mathrm{CA}} \subseteq \mathsf{BPP}$, from compactness there exists $k' \geq 1$ and a language O consistent with CA such that $\mathsf{P}^O \subseteq \mathsf{SIZE}(n^{k'})[\mathcal{C}]$. By Theorem 2 there exists $\Pi \in \mathsf{PromiseBPP}$ such that $\Pi \notin \mathsf{SIZE}(n^{k'})[\mathcal{C}]$. Since CA is a $\mathsf{PromiseBPP}$-complete language there exists a polynomial-time function $g : \Pi \to \mathrm{CA}$ such that: $x \in \Pi_{YES} \implies g(x) \in \mathrm{CA}_{YES}, x \in \Pi_{NO} \implies g(x) \in \mathrm{CA}_{NO}$. Let us define $O' \triangleq \{x \mid g(x) \in O\}$. First, observe that $O' \in \mathsf{P}^O$. Next, we claim that O' is consistent with Π. That is: $x \in \Pi_{YES} \implies g(x) \in \mathrm{CA}_{YES} \subseteq O \implies x \in O', x \in \Pi_{NO} \implies g(x) \in \mathrm{CA}_{NO} \subseteq \bar{O} \implies x \notin O'$. Recalling Definition 4, $O' \notin \mathsf{SIZE}(n^{k'})[\mathcal{C}]$ thus leading to a contradiction.

The unconditional case is slightly more involved. We refer the reader to the full version of the paper.

6 Discussion and Open Questions

In this paper we show that efficient *randomized* learning algorithms imply circuit lower bounds against $\mathsf{BPTIME}(n^{\omega(1)})/1$, and some other hardness results. This (almost) solves the main open problem posed in [6] and [14], and matches the corresponding result of [14] that *deterministic* learning algorithms imply circuit lower bounds against $\mathsf{DTIME}(n^{\omega(1)})$. We would like to point out that those conditional lower bounds are nearly-optimal if we treat the learning algorithms as black-boxes. More specifically, all the above results only assume an existence of an efficient learning algorithm, which is invoked as a black-box regardless of the class $\mathcal{C}$ it learns. Consequently, the obtained lowers bounds are of form "$\mathcal{C}$ is learnable $\implies \Gamma \nsubseteq \mathsf{P/poly}[\mathcal{C}]$" for every circuit class $\mathcal{C}$ and some complexity class Γ. Given that, we cannot expect to obtain conditional lower bounds of the form: "P or $\mathsf{BPP}/1 \nsubseteq \mathsf{P/poly}[\mathcal{C}]$" since $\mathsf{P} \subseteq \mathsf{BPP}/1 \subseteq \mathsf{P/poly}$ (i.e. when $\mathcal{C}$ is the class of all Boolean circuits with AND,OR and NOT gates). We refer to this as the "black-box barrier". This barrier can be seen as an analog of the relativization barrier for proving lower bounds. So, one open question is to derive lower bounds of the form "P or $\mathsf{BPP} \nsubseteq \mathsf{P/poly}[\mathcal{C}]$" from an efficient learning algorithm for some specific families of circuit classes. In the light of the above, such a conditional lower bound will have to exercise a non black-box technique.

The other open question is, naturally, to remove the extra bit of advice appearing in both conditional and the unconditional bounds. We hope that the approach described in Section 5 will be a step in the right direction.

Acknowledgment. The author would like to extend his gratitude to Dieter van Melkebeek for sharing lots of his knowledge and commenting on the first version of the paper. The author would also like to thank Eric Allender and Nader Bshouty for answering his questions and many useful conversations. Finally, the author would like to thank Akinori Kawachi, Igor Oliveira and anonymous referees for their detailed comments and suggestions.

References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. 75(2), 87–106 (1987)
2. Angluin, D.: Queries and concept learning. Machine Learning 2, 319–342 (1988)

3. Barak, B.: A probabilistic-time hierarchy theorem for slightly non-uniform algorithms. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 194–208. Springer, Heidelberg (2002)
4. Buhrman, H., Fortnow, L.: One-sided versus two-sided error in probabilistic computation. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 100–109. Springer, Heidelberg (1999)
5. Buhrman, H., Fortnow, L., Thierauf, T.: Nonrelativizing separations. In: Proceedings of the 13th Annual IEEE Conference on Computational Complexity (CCC), pp. 8–12 (1998)
6. Fortnow, L., Klivans, A.R.: Efficient learning algorithms yield circuit lower bounds. J. Comput. Syst. Sci. 75(1), 27–36 (2009)
7. Fortnow, L., Santhanam, R.: Hierarchy theorems for probabilistic polynomial time. In: FOCS, pp. 316–324 (2004)
8. Gentry, C., Halevi, S.: Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In: Proceedings of the 52nd Annual FOCS, pp. 107–109 (2011)
9. Goldreich, O., Zuckerman, D.: Another proof that bpp $\subseteq$ ph (and more). In: Studies in Complexity and Cryptography, pp. 40–53 (2011)
10. Harkins, R.C., Hitchcock, J.M.: Exact learning algorithms, betting games, and circuit lower bounds. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 416–423. Springer, Heidelberg (2011)
11. Impagliazzo, R., Wigderson, A.: Randomness vs. time: De-randomization under a uniform assumption. In: FOCS, pp. 734–743 (1998)
12. Kabanets, V., Impagliazzo, R.: Derandomizing polynomial identity tests means proving circuit lower bounds. Computational Complexity 13(1-2), 1–46 (2004)
13. Kearns, M.J., Valiant, L.G.: Cryptographic limitations on learning boolean formulae and finite automata. J. ACM 41(1), 67–95 (1994)
14. Klivans, A., Kothari, P., Oliveira, I.: Constructing hard functions from learning algorithms. In: Proceedings of the 28th Annual IEEE Conference on Computational Complexity (CCC), pp. 86–97 (2013)
15. Klivans, A.R., Sherstov, A.A.: Cryptographic hardness for learning intersections of halfspaces. J. Comput. Syst. Sci. 75(1), 2–12 (2009)
16. van Melkebeek, D., Pervyshev, K.: A generic time hierarchy with one bit of advice. Computational Complexity 16(2), 139–179 (2007)
17. Miltersen, P.B., Vinodchandran, N.V., Watanabe, O.: Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In: Asano, T., Imai, H., Lee, D.T., Nakano, S.-i., Tokuyama, T. (eds.) COCOON 1999. LNCS, vol. 1627, pp. 210–220. Springer, Heidelberg (1999)
18. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. J. ACM 51(2), 231–262 (2004)
19. Razboeov, A.A., Rudich, S.: Natural proofs. J. of Computer and System Sciences 55(1), 24–35 (1997)
20. Santhanam, R.: Circuit lower bounds for merlin–arthur classes. SIAM J. Comput. 39(3), 1038–1061 (2009)
21. Shamir, A.: IP=PSPACE. In: Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science, pp. 11–15 (1990)
22. Trevisan, L., Vadhan, S.P.: Pseudorandomness and average-case complexity via uniform reductions. Computational Complexity 16(4), 331–364 (2007)
23. Valiant, L.G.: A theory of the learnable. Communications of the ACM 27(11), 1134–1142 (1984)