On the Structure of Learnability Beyond P/Poly

Ninad Rajgopal
University of Warwick, Coventry, UK

Rahul Santhanam
University of Oxford, UK

Abstract

Motivated by the goal of showing stronger structural results about the complexity of learning, we study the learnability of strong concept classes beyond P/poly, such as PSPACE/poly and EXP/poly. We show the following:

1. (Unconditional Lower Bounds for Learning) Building on [31], we prove unconditionally that BPE/poly cannot be weakly learned in polynomial time over the uniform distribution, even with membership and equivalence queries.
2. (Robustness of Learning) For the concept classes EXP/poly and PSPACE/poly, we show unconditionally that worst-case and average-case learning are equivalent, that PAC-learnability and learnability over the uniform distribution are equivalent, and that membership queries do not help in either case.
3. (Reducing Succinct Search to Decision for Learning) For the decision problems R_{Kt} and R_{KS} capturing the complexity of learning EXP/poly and PSPACE/poly respectively, we show a *succinct search to decision* reduction: for each of these problems, the problem is in BPP iff there is a probabilistic polynomial-time algorithm computing circuits encoding proofs for positive instances of the problem. This is shown via a more general result giving succinct search to decision results for PSPACE, EXP and NEXP, which might be of independent interest.
4. (Implausibility of Oblivious Strongly Black-Box Reductions showing NP-hardness of learning NP/poly) We define a natural notion of hardness of learning with respect to oblivious strongly black-box reductions. We show that learning PSPACE/poly is PSPACE-hard with respect to oblivious strongly black-box reductions. On the other hand, if learning NP/poly is NP-hard with respect to oblivious strongly black-box reductions, the Polynomial Hierarchy collapses.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Hardness of Learning, Oracle Circuit Classes, Succinct Search, Black-Box Reductions

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2021.46

Category RANDOM

Funding This work was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2014)/ERC Grant Agreement no. 615075.
Ninad Rajgopal: Supported in part by Tom Gur's UKRI Future Leaders Fellowship MR/S031545/1.[1]

Acknowledgements Ninad is grateful to Igor Carboni Oliveira for many inspiring discussions, one of which led to the results in Section 2.

1 Introduction

What is the complexity of learning polynomial-size circuits? Despite extensive research on this question, our knowledge is still fairly sparse. For weak concept classes such as decision trees [34, 32], DNFs [34, 27] or even constant-depth circuits with parity gates [12],

[1] Most of this work was done when Ninad Rajgopal was affiliated with the University of Oxford.

reasonably efficient learning algorithms under the uniform distribution are known for various models of learning. For stronger concept classes, learning is believed to be hard, but the evidence for this is not as strong as one might hope. Cryptographic assumptions such as the existence of one-way functions are known to imply that learning polynomial-size circuits is hard [29, 18]. However, we still seem far from showing that PAC-learning polynomial-size circuits is NP-hard - indeed [5] give negative results for certain kinds of black-box reductions to learning.

In this paper, we adopt a fresh perspective of approaching the learnability question from above, i.e. via circuit classes which are more powerful than $\mathsf{P/poly}$. We consider commonly held beliefs about the complexity of learning, and establish these beliefs unconditionally for strong concept classes such as $\mathsf{PSPACE/poly}$ and $\mathsf{EXP/poly}$. Of course the very learnability of these concept classes has some unlikely implications, eg., that these classes are approximable by efficient Boolean circuits. The point is that this is still consistent with our complexity-theoretic understanding, and we would like to know what current techniques are capable of proving *unconditionally* about learning. Partly this is to understand the limitations of current techniques, and partly this is to understand what structural properties of the stronger concept classes enable us to show unconditional results about them.

We begin by outlining our main results and comparing them with previous work.

1.1 Unconditional Results for Hardness of Learning

Our first set of results deals with unconditional hardness of learning circuit classes. Most complexity theorists believe that learning polynomial-size circuits is unconditionally hard, but of course proving this is at least as hard as the P vs NP problem. We ask: what is the smallest concept class $\mathcal{C}/\mathsf{poly}$[2] for which we can *prove* learning to be hard? Clearly, if we can prove that $\mathcal{C}/\mathsf{poly}$ cannot be approximated by efficient circuits, i.e., there does not even *exist* a good hypothesis for all concepts in the class, then hardness of learning follows. This observation implies for example that learning MAEXP is hard, by using known circuit lower bounds for this class [11].

But can we show hardness of learning unconditionally for some concept class where it is consistent with our current understanding of complexity theory that a good hypothesis exists for every concept in the class? We give an affirmative answer by ruling out PAC-learning with membership and equivalence queries unconditionally for the class $\mathsf{BPE/poly}$.

The notion of PAC-learning $\mathcal{C}/\mathsf{poly}$, for a uniform class $\mathcal{C}$ above P such as EXP or BPE, can have different interpretations. Standard definitions for PAC-learning (cf. [30]) consider the task of learning to be efficient if it is polynomial in the size of the target concept over n inputs (assume that the accuracy ε and confidence δ are both $1/\mathsf{poly}(n)$) and the hypothesis class is $\mathsf{P/poly}$.[3] The standard definition of PAC-learning in $\mathsf{poly}(n)$ time using $\mathsf{P/poly}$ as its hypothesis class naturally extends to the concept class $\mathcal{C}/\mathsf{poly}$ as the size of the target concept is *still* polynomial in the input size n. For the classes $\mathcal{C}$ we consider, PAC-learnability of $\mathcal{C}/\mathsf{poly}$ in $\mathsf{poly}(n)$ time using polynomial-sized hypothesis circuits is still consistent with our current understanding of complexity theory (as we do not have any unconditional average-case lower bounds for $\mathcal{C}$ against $\mathsf{P/poly}$), and therefore worth studying.

[2] For any uniform complexity class $\mathcal{C}$, define the class $\mathcal{C}/\mathsf{poly}$ as the set of languages L for which there is a language $\mathcal{C}$-machine M and a family of strings $\{a_n\}$, where $a_n \in \{0,1\}^{\mathsf{poly}(n)}$, such that for every $x \in \{0,1\}^n$, $x \in L \iff M$ accepts (x, a_n)

[3] In general, the definition requires the hypothesis class H to be *polynomially evaluatable*, which means that there exists an algorithm that on input any instance $x \in \{0,1\}^n$ and an encoding of the hypothesis $h \in H_n$, outputs the value $h(x)$ in time polynomial in n and the size of the hypothesis encoding. It is well known that $\mathsf{P/poly}$ is polynomially evaluatable.

We say that a class $\mathcal{C}$ is (ε, δ)-learnable using membership queries over distribution $\mathcal{D}$ in polynomial time, if there exists a probabilistic polynomial time learning algorithm which given oracle access to any $f \in \mathcal{C}$, with probability at least $1 - \delta$, outputs a polynomial-sized hypothesis circuit that approximates f up to an error ε over the target distribution $\mathcal{D}$. This definition also extends to the case of (ε, δ)-learning using random examples.

BPE/poly can equivalently be defined as the class of languages computable by polynomial-sized circuit families with oracle gates to some function in BPE, with the oracle query size restricted to $O(n)$. We prove the unconditional hardness of learning BPE/poly in polynomial time using membership queries even over the uniform distribution using P/poly as the hypothesis class. Hardness of exactly learning BPE/poly with membership and equivalence queries, even using randomized algorithms follows directly from this via [4].

▶ **Theorem 1.1.** *For every constant $k \in \mathbb{N}$,* BPE/poly *cannot be $(1/2 - 1/n^k, 1/n)$-learnt over the uniform distribution using membership queries by randomized learning algorithms running in polynomial time.*

To prove this, we adapt techniques used by [31, 36] to show that randomized PAC-learning algorithms imply circuit lower bounds. [38] show the existence of a PSPACE-Complete function f^* which is in DSPACE$[n]$, such that f^* is downward self-reducible and self-correctible (see Appendix A for definitions). Using the techniques of [31], along with the fact that f^* belongs to BPE, we see that PSPACE collapses to BPP. Using a padding argument and diagonalizing DSPACE$[2^{O(n)}]$ against functions which can be approximated by polynomial-sized circuits, we obtain a contradiction to the fact that for every function in BPE/poly, the learner gives a hypothesis circuit which approximates it well.

1.2 Robustness for Hardness of Learning

We believe that polynomial-size circuits are hard to learn in a *robust* sense, i.e., that the precise details of the learning model do not matter. Hardness should hold irrespective of whether we consider PAC-learning or learning over the uniform distribution, worst-case learning or average-case learning over some samplable distribution on concepts, and whether or not the learning model is allowed to use membership queries. We do not know how to show that this robustness holds for P/poly, but we are able to show it unconditionally for EXP/poly and PSPACE/poly.

We now consider the class EXP/poly, which can be equivalently defined as the circuit class $\mathsf{P}^{\mathsf{EXP}}$/poly i.e. the class of languages that can be computed by a polynomial sized circuit family with EXP oracle gates.

Showing non-trivial derandomization of BPP, i.e. EXP $\neq$ BPP, is one of the most fundamental questions in complexity theory.[4] We prove that the problem of non-trivial derandomization of BPP is equivalent to the hardness of learning EXP/poly efficiently in most standard models of PAC-learning. In addition, these results extend to not just showing that EXP/poly is hard to learn in the worst-case, but also on average with respect to polynomially samplable distributions over EXP/poly.[5] This also gives us an intriguing situation, where hardness of learning EXP/poly using random examples also implies the hardness of learning EXP/poly using membership queries.

[4] It is worth mentioning that [26] show that EXP $\neq$ BPP is equivalent to the fact that BPP can be derandomized on average in deterministic sub-exponential time (over infinitely many input lengths).

[5] In particular, the results hold for polynomially samplable distribution families over EXP/poly, where for each n, there exists a distribution in the family over circuit encodings of n-variate functions in EXP/poly, implicitly defining a distribution on n-variate functions in EXP/poly (see Remark A.3 for more details.)

The following results are stated for hardness of strong learning. However, they also hold for the setting of weak learnability, by standard equivalences between weak learning and strong learning for PAC-learners [17].

▶ **Theorem 1.2** (Equivalences for hardness of learning EXP/poly). *The following statements are equivalent.*

1. ***Non-trivial derandomization of BPP:*** EXP $\neq$ BPP*.*
2. ***Hardness of PAC-learning* EXP/poly *in the worst-case using random examples:*** *There exists $c \geq 0$, such that* EXP/poly *is not $(1/n^c, 1/20n)$-PAC-learnable in polynomial time using random examples.*
3. ***Hardness of PAC-learning* EXP/poly *in the worst-case using membership queries:*** *There exists $c \geq 0$, such that* EXP/poly *is not $(1/n^c, 1/20n)$-PAC-learnable in polynomial time using membership queries.*
4. ***Hardness of PAC-learning* EXP/poly *on average using random examples:*** *There exists $c \geq 0$, such that* EXP/poly *is not $(1/n^c, 1/20n)$-PAC-learnable in polynomial time on average using random examples, with respect to polynomially samplable distributions over* EXP/poly*.*
5. ***Hardness of PAC-learning* EXP/poly *on average using membership queries:*** *There exists $c \geq 0$, such that* EXP/poly *is not $(1/n^c, 1/20n)$-PAC-learnable in polynomial time on average using membership queries, with respect to polynomially samplable distributions over* EXP/poly*.*

A contrasting result to this is the equivalence between the existence of one-way functions (OWFs) and the hardness of learning P/poly in polynomial time on average with respect to polynomially samplable distributions over P/poly using random examples [25, 8]. Theorem 1.2 not only lends an analogous equivalence between a complexity theoretic assumption that BPP has a non-trivial derandomization and the hardness of learning EXP/poly in polynomial time on average using random examples, but also extends this equivalence to hardness of learning EXP/poly efficiently in the worst-case. Note that showing such an equivalence between the existence of OWFs and hardness of learning P/poly efficiently in the worst-case has been open for decades.[6]

Furthermore, our proof techniques also let us extend all these equivalences to the case where $\mathcal{C}$ = PSPACE.

▶ **Corollary 1.3.** *The following statements are equivalent.*

1. PSPACE $\neq$ BPP*.*
2. *There exists $c \geq 0$, such that* PSPACE/poly *is not $(1/n^c, 1/20n)$-PAC-learnable in polynomial time using random examples (also using membership queries).*
3. *There exists $c \geq 0$, such that* PSPACE/poly *is not $(1/n^c, 1/20n)$-PAC-learnable in polynomial time on average using random examples (also using membership queries) with respect to polynomially samplable distributions over* PSPACE/poly*.*

Essentially, the proof of showing conditional hardness of PAC-learning EXP/poly uses the fact that strongly learning EXP/poly using random examples over the *uniform distribution* implies that EXP = BPP. This also means that the hardest distribution to learn EXP/poly is over the uniform distribution. The same ideas hold for PAC-learning EXP/poly using membership queries too.

[6] In particular, we do not know if hardness of learning P/poly efficiently using random examples in the worst-case implies OWFs.

Our techniques used to show these equivalences are inspired from results on uniform derandomization by [26, 38], which were further used by [15, 31] to show circuit lower bounds based on the existence of learning algorithms. We use special properties of functions in EXP and PSPACE like downward self-reducibility and self-correctibility to show that learning these functions would imply a collapse for EXP and PSPACE to BPP.

1.3 Reducing Succinct Search to Decision for Learning

Recently, [12] established an important connection between *natural proofs* and *learning.* They showed that natural proofs of strong lower bounds against a circuit class $\mathcal{C}/\mathsf{poly}$ imply efficient learning algorithms for $\mathcal{C}/\mathsf{poly}$ over the uniform distribution with membership queries, as long as the class $\mathcal{C}/\mathsf{poly}$ satisfies some mild closure properties. One way to interpret their result is as an *approximate search to decision* reduction for learning. The decision version of learning polynomial-size circuits is the language MCSP consisting of truth tables of functions that have small circuits, i.e., for which a good hypothesis exists. The search version is to find a small circuit for a positive instance of MCSP. [12] show that if MCSP is polynomial-time decidable (which is implied by the existence of natural proofs against P/poly), then the search version of MCSP can be solved approximately, in the sense that we can efficiently compute a polynomially larger sized circuit that approximates the truth table well.

The language R_{Kt} (resp. R_{KS}) of strings with high Kt complexity (resp. high KS complexity) plays an analogous role to MCSP in the theory of learning $\mathsf{EXP}/\mathsf{poly}$ (resp. $\mathsf{PSPACE}/\mathsf{poly}$). We ask if search to decision reductions can be established for these languages as well. However, it is unclear a priori what it would mean to solve search efficiently for a problem that does not have polynomial-size proofs or witnesses. We introduce the notion of *succinct search.* To efficiently solve a search problem succinctly is to efficiently compute for any YES instance of the problem a circuit that encodes a possibly exponential-size proof for the instance. We use the PCP theorem for NEXP [6] and the Easy Witness Lemma [24] to show that for the classes $\mathsf{PSPACE}, \mathsf{EXP}$ and NEXP, efficient decidability of the class is equivalent to efficiently solving succinct search for every language in the class. We then use results from [3] to argue that for R_{Kt} and R_{KS}, efficient solvability is equivalent to solving succinct search efficiently. Note that this connection is for succinctly solving the search problem *exactly* rather than just for approximate search as in [12].

▶ **Theorem 1.4** (Equivalence of Succinct Search and Decision for Learning $\mathsf{EXP}/\mathsf{poly}$ and $\mathsf{PSPACE}/\mathsf{poly}$)**.** *Let L be R_{Kt} or R_{KS}. $L \in \mathsf{BPP}$ iff for each polynomial-time verifier V for L, succinct search is efficiently solvable for L with respect to V.*

1.4 Barriers for Establishing NP-Hardness of Learning

We next look at questions pertaining to hardness of learning classes of the form $\mathcal{C}/\mathsf{poly}$, where $\mathcal{C} \subseteq \mathsf{PH}$. We only focus on the hardness of PAC-learning $\mathcal{C}/\mathsf{poly}$ with random examples. In this section, we consider the limitations of proving the NP-hardness of PAC-learning $\mathsf{NP}/\mathsf{poly}$, i.e. the class of polynomial size non-deterministic circuits, using random examples, via a black-box reduction from deciding SAT.

Informally, a black-box reduction from problem A to B, solves A given access to any oracle solving B. Black-box reductions have been ubiquitously used in complexity theory to prove conditional lower bounds. However, for many fundamental questions in complexity theory, there have been results showing why such reductions are limited in power. Various results have conditionally ruled out special-cases of black-box reductions for showing average-case hardness of NP [14, 10], existence of one-way functions [1, 5, 9] and the existence of hitting set generators [22], from hardness of SAT.

For the case of showing hardness of learnability, a B-adaptive black-box reduction R from some language L to PAC-learning a class $\mathcal{C}$ using random examples is defined by two phases

- The first phase consists of B adaptive rounds of probabilistic polynomial time algorithms, each of which generates queries to the learner oracle. In more detail, each round uses the input z to the reduction, fresh randomness and the hypotheses returned by the $\mathcal{C}$-learner oracle in the previous rounds, and constructs joint distributions (that serve as example oracles for the learner). It then samples a set of independent labeled examples from each of these distributions as queries to the learner oracle.
- In the second phase, a probabilistic polynomial time algorithm takes all the hypotheses from the first phase and decides whether $z \in L$, with high probability.

[5] study the question of the existence of black-box Turing reductions from any language in NP to PAC-learning P/poly using random examples. They consider a strongly black-box reduction, where a reduction is strongly black-box if it runs correctly given any oracle for the learner, as well as the hypotheses output by the learner. For a special case of such a reduction, where the access to the learner and the hypothesis oracles is additionally non-adaptive, they show that such a reduction from SAT to PAC-learning P/poly using random examples collapses NP to CoAM (which implies a collapse of PH to the second level). Additionally, they show that if any language L reduces to PAC-learning P/poly using random examples via an $O(1)$-adaptive black-box reduction, then the hardness of L implies the existence of an auxiliary-input one-way function (which is a major breakthrough in cryptography).[7]

We define a natural special-case of such a reduction, called an *oblivious* strongly black-box reduction, where the obliviousness of a reduction implies that the queries made to the learner do not depend on the input z to the reduction, and try to understand its limitations for showing NP-hardness of PAC-learning NP/poly. At a first glance, ruling out oblivious reductions may seem very restrictive, since ideally, one would like to allow reductions whose queries to the learner can depend on the input to the reduction. However, we observe the proof of Corollary 1.3 which shows hardness of PAC-learning PSPACE/poly assuming PSPACE $\neq$ BPP and reformulate it as an oblivious black-box reduction of the form defined above. In particular, for f^* being the PSPACE-Complete function given by [38] which is downward self-reducible and self-correctible, we observe that

▶ **Lemma 1.5.** *There exists an oblivious, n-adaptive, strongly black-box reduction from deciding f^* to PAC-learning* PSPACE/poly *using random examples over the uniform distribution.*

On the other hand, for the case of learning NP/poly using random examples, we show that oblivious strongly black-box reductions from SAT imply a collapse of the polynomial hierarchy. Our main result for the section is

▶ **Theorem 1.6** (Informal). *If there exists an oblivious,* poly(n)*-adaptive, strongly black-box reduction from deciding* SAT *to learning* NP/poly *using random examples over polynomially samplable distributions, then* PH *collapses to the third level.*[8]

[7] They also show the impossibility of Karp reductions from SAT to PAC-learning P/poly using random examples, unless NP collapses to SZKA.

[8] We actually show a stronger result that the existence of such a reduction implies that NP $\subseteq$ CoAM$^{\text{poly}}$, where CoAM$^{\text{poly}}$ is the class of languages recognized by constant-round CoAM protocols with advice, where we require proper acceptance/rejection probabilities only when the advice is correct.

Theorem 1.6 implies that standard techniques used for worst-case to average-case reductions, pseudo-random generator constructions from uniform hardness assumptions and in particular, hardness of efficiently PAC-learning classes like $\mathsf{PSPACE/poly}$, cannot be used to show the NP-hardness of PAC-learning $\mathsf{NP/poly}$ using random examples.

Theorem 1.6 compares to some previous results in the following way:

- It shows a conditional impossibility result by ruling out a restricted version of adaptive, strongly black-box reductions to learning $\mathsf{P/poly}$ using random examples, in contrast to [5], who only rule out fully non-adaptive, strongly black-box reductions, from a slightly weaker assumption ($\mathsf{NP} \not\subseteq \mathsf{CoAM}$).
- Furthermore, the result by [22] which conditionally rules out a non-adaptive black box reduction from deciding SAT to breaking a Hitting Set Generator (HSG), in turn rules out fully non-adaptive, strongly black-box reductions from SAT to learning $\mathsf{NP/poly}$ using membership queries over the uniform distribution (by suitably changing the definition of the reduction to the learner).
  Indeed, the ideas of [26] can be used to show that hardness of learning $\mathsf{NP/poly}$ using membership queries over the uniform distribution, implies the existence of a hitting set generator which hits sufficiently dense circuits. We strengthen this observation by not only extending the reduction to a restricted version of the adaptive case, but also by ruling out a weaker reduction to learning $\mathsf{NP/poly}$ with random examples.
- In a similar way, [20] conditionally rule out the existence of mildly adaptive (each query length up to n, where n is the length of the input instance, appears in very few levels of adaptivity), strongly black-box reductions from an EXP-Complete problem to learning $\mathsf{NP/poly}$ using membership queries (and in fact, learning $\mathsf{EXP/poly}$).
  Our result rules out the restricted cases of mildly adaptive, strongly black-box reductions which show the NP-hardness of learning $\mathsf{NP/poly}$ using random examples and hence, is a conceptual strengthening of [20], as we rule out a hardness result from a stronger assumption.
- On the other hand, Schapire [37] shows that a *non-uniform* hardness assumption like $\mathsf{NP/poly} \neq \mathsf{P/poly}$ actually implies the hardness of PAC-learning $\mathsf{NP/poly}$ in polynomial time using random examples. They show that if $\mathsf{NP/poly}$ is learnable in polynomial time, then there exists an algorithm which takes any m labeled samples of a target $f_n \in \mathsf{NP/poly}$, runs in time $\mathsf{poly}(m, n)$, and with high probability, outputs a hypothesis of size $\mathsf{poly}(n)$ (independent of m) that is consistent with all the labeled samples.
  In particular, if $m = 2^n$, then for every $f_n \in \mathsf{NP/poly}$, the algorithm outputs a polynomial-sized hypothesis circuit which computes it correctly on all inputs, thus contradicting the assumption $\mathsf{NP/poly} \neq \mathsf{P/poly}$. Note that the result uses that for any f_n, we get a polynomial-sized circuit that computes it, and in fact, the algorithm runs in $\mathsf{poly}(m, n) = 2^{O(n)}$ time and is not useful in terms of contradicting a uniform assumption.

It is worth noting that our result has no implications for showing the impossibility of adaptive, black-box NP-hardness reductions which imply the average-case hardness for NP [14, 10], existence of one-way functions [1, 5] or the existence of HSGs [20, 22].

Overview of the techniques. The proof of Theorem 1.6 builds on the Feigenbaum-Fortnow [14] protocol, which simulates a type of non-adaptive randomized reduction A from SAT to an NP problem $\mathcal{Q}$, by an AM protocol with polynomial-sized advice, and shows that $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.[9]

[9] Their motivation (and [10]) was to rule out certain kinds of non-adaptive, worst-case to average-case black-box reductions for NP.

Suppose that on input x, A makes q non-adaptive queries to $\mathcal{Q}$, sampled independently from certain distribution X. Very briefly, their AM protocol does the following. For K large enough, the verifier first generates K tuples of q non-adaptive queries by running $A(x)$ independently K times. The verifier asks the prover to send a witness to each query which is a YES instance (which it can verify easily). This ensures that the prover cannot cheat if the query is a NO instance and the only way it can cheat is by claiming a YES instance to be a NO instance. Now, if the verifier has the proportion p of YES instances of $\mathcal{Q}$ over the distribution X, then with high probability it knows that the number of YES instances among the Kq queries is concentrated around $q \cdot (pK \pm O(\sqrt{K}))$. The verifier answers with a reject if the number of YES instances is much lesser than pqK.

The honest prover answers each query correctly (with correct witnesses if necessary) and with high probability, the number of YES instances are close to the expectation. Hence, the verifier can pick any of K runs of $A(x)$ using the prover's answers to its queries and the output will be correct with high probability. On the other hand, the cheating prover cannot cheat on more than $O(q\sqrt{K})$ YES instances, with high probability. If we choose $K \gg O(q\sqrt{K})$, then on most of the K independent runs of A, all its queries are answered correctly and the reduction gives the correct answer. Thus, if we pick one of the runs at random and get $A(x)$ by using the prover's answers to its queries, the verifier answers wrongly with low probability.

Consider an oblivious, B-adaptive, strongly black-box reduction R from L to an oracle which learns NP/poly. Suppose we are able to fix $S_1, \ldots, S_t$, which are sets of labeled examples drawn independently from the joint distributions $(X_1, f_1(X_1)), \ldots, (X_t, f_t(X_t))$ where $f_1, \ldots, f_t \in \mathsf{NP/poly}$, as the queries made to the learner. Furthermore, let $h_1, \ldots, h_t$ be a set of fixed hypotheses circuits, some of which are used to generate $S_1, \ldots, S_t$, such that each h_i $(1 - \varepsilon_0)$-approximates f_i over X_i, for some $\varepsilon_0 > 0$. Because R is strongly black-box, each hypothesis is also accessed as an oracle and we see that L is decided by the algorithm M in the second phase, which has access to $h_1, \ldots, h_t$. Now, the t oracles to M can be replaced by a single oracle $\mathcal{O}$ which takes as input $i \in [t]$ and $y \in \{0,1\}^n$, and outputs $h_i(y)$ ($\mathcal{O}$ can be thought of as a table with t rows and 2^n columns). We then adapt the techniques of [14] to design an AM protocol for L with polynomial sized advice, where the verifier expects that the prover answers according to $\mathcal{O}$.

The obliviousness of the reduction helps us in fixing the queries made by R, and implicitly, the corresponding hypotheses output by the oracle. In other words, this helps us fix the proportions of YES instances for each f_i non-uniformly, as the queries generated to the learner do not depend on the input to the reduction. We do this by inductively fixing the queries made by the reduction starting from the first round of adaptivity. Fixing a "good" polynomial-sized random string r^* used by the first phase non-uniformly (using Adleman's trick), we first get the queries to the learner made in the first round.

For any other round $b \geq 2$, assume that the queries to the learner up to round $(b-1)$ and the functionality of the hypothesis oracles used to generate them up to round $(b-2)$ are fixed. Using the fact that r^* is also fixed, we consider the set of all tuples of joint distributions that can be generated in the b^{th} round depending on the answers to the oracle queries of the hypotheses seen so far, and arbitrarily choose one of them. Note that, this implicitly fixes the functionality of the hypothesis oracles for the queries generated in round $b-1$. We continue this process and fix all the queries made to the learner by all the rounds from the first phase.

The details of the results from this section have been delegated to Appendix B because of space constraints.

1.5 Further Discussion

Connections to Karp-Lipton Style Theorems. There is an analogy between our results on implications of learnability and Karp-Lipton style theorems. A Karp-Lipton style theorem for a uniform class $\mathcal{C}$ gives an unlikely uniform implication of the assumption that $\mathcal{C}$ has polynomial-size circuits. The original theorem of Karp and Lipton [28] shows such an implication for $\mathcal{C} = \mathsf{NP}$: if $\mathsf{NP} \subseteq \mathsf{P/poly}$, then $\Sigma_2 = \Pi_2$. Karp-Lipton style theorems are now known for many other classes, including $\mathcal{C} = \mathsf{P}^{\#\mathsf{P}}$ [35], $\mathcal{C} = \mathsf{PSPACE}$ [6] and $\mathcal{C} = \mathsf{EXP}$ [6]. In each of these cases, $\mathcal{C} \subseteq \mathsf{P/poly}$ implies $\mathcal{C} = \mathsf{MA}$, applying techniques from the theory of interactive proofs [35, 6].

Similarly, in some of our results (i.e., Theorem 1.1, Theorem 1.2 and Corollary 1.3), we study implications of learnability for classes $\mathcal{C}/\mathsf{poly}$, where $\mathcal{C} = \mathsf{BPE}, \mathsf{EXP}$ or PSPACE. Since the learner is required to output a polynomial-size Boolean circuit, the learnability assumption already implies that $\mathcal{C}$ is approximated by polynomial-size circuits, where the approximation is over the distribution on the examples. We are interested in establishing strong uniform implications of these assumptions, showing that the assumption is actually false in the case $\mathcal{C} = \mathsf{BPE}$, and that the assumption implies a simulation of $\mathcal{C}$ in BPP in the other cases. What enables us to show stronger implications than in corresponding Karp-Lipton style theorems is that the learner uniformly produces a good hypothesis by our assumption. However, the learner is assumed to have access to random examples or membership queries which cannot be efficiently simulated - this makes our simulation task more challenging, and we therefore exploit various structural properties of complete languages. We also need to deal with the issue of approximation, while standard Karp-Lipton style theorems have as their antecedent an exact simulation by efficient circuits.

Open Questions. One question which stems from our work is to explore the possibility of showing the hardness of PAC-learning $\mathsf{NP/poly}$ efficiently using random examples assuming that $\mathsf{NP} \neq \mathsf{BPP}$. A potential direction is to consider non black-box reductions for the NP-hardness of PAC-learning $\mathsf{NP/poly}$. This viewpoint has lent itself some success in the case of worst-case to average-case reductions [19, 21, 22] and in our case, hardness of efficiently PAC-learning $\mathsf{EXP/poly}$. Indeed, the reduction for $\mathsf{EXP/poly}$ only works if the learning algorithm runs in polynomial time, although the reduction still uses the learning algorithm as an oracle.[10] Moreover, [13] show a non black-box reduction from an approximate version of MCSP to learning $\mathsf{P/poly}$ by sub-exponential-sized circuits (and thus, learning $\mathsf{NP/poly}$). Note that, it is unclear if approximate MCSP is NP-hard and this reduction does not imply the NP-hardness of PAC-learning $\mathsf{NP/poly}$ efficiently.

Another important question is to explore an analogue of the PH collapse for learnability. In other words, does polynomial time learnability of $\mathsf{NP/poly}$ imply polynomial time learnability of $\mathsf{PH/poly}$? Note that, under a strong assumption of the existence of a (possibly adaptive and non-relativizable) worst-case to average-case reduction for NP, we can use the techniques in Lemma 3.2 along with the downward-self-reducibility of SAT to show such a collapse. On the other hand, [23] also shows that there exists an oracle O with respect to which $\mathsf{DistNP}^O \subseteq \mathsf{AvgP}^O$ and $\Sigma_2^O \not\subseteq \mathsf{HeurSIZE}^O[2^{n^\alpha}]$. Essentially, this result negates the existence of any relativizable reductions which show a statement analogous to the PH collapse for average-case algorithms i.e. if NP is easy on average, then Σ_2 is easy on average too. In a similar spirit, can we prove that no relativizable technique can show that if $\mathsf{NP/poly}$ is learnable in polynomial time, then Σ_2/poly is learnable in polynomial time as well?

[10] For EXP to collapse to PSPACE, we need the $\mathsf{EXP/poly}$ learner to be efficient so that it outputs polynomial-sized hypothesis circuits for any language in EXP, and this further implies $\mathsf{EXP} \subseteq \mathsf{P/poly}$.

2 Unconditional Results for Hardness of Learning

Firstly, we show the hardness of learning BPE/poly over the uniform distribution using membership queries by randomized polynomial time algorithms. The proof of this result uses the following lemma from [31]. Preliminaries and definitions can be found in Appendix A.

▶ **Lemma 2.1.** *Let $\mathcal{C}$ be any circuit class, $s : \mathbb{N} \to \mathbb{N}$ be a size function and f^* be the* PSPACE*-Complete problem from Theorem A.8. There exists constant $c \in \mathbb{N}$ such that if $\mathcal{C}[s(n)]$ is learnable up to error n^{-c} in time $T(n)$, then at least one of the following holds :*

- $f^* \notin \mathcal{C}[s(n)]$.
- $f^* \in \mathsf{BPTIME}[\mathsf{poly}(T(n))]$.

We also need Lemma A.11 (Appendix A) which proves the existence of functions which cannot be approximated by $n^{\log n}$-sized circuits.

Proof of Theorem 1.1. Towards a contradiction, assume that there exists constants $k, d \geq 1$ and a randomized learning algorithm A which learns BPE/poly in $O(n^d)$ time over the uniform distribution using membership queries, up to error $1/2 - 1/n^k$ and confidence $1/n$, for every large enough input length n. By non-uniformly fixing a good random string, we ensure that for every function $g \in \mathsf{BPE/poly}$, there exists c such that A always outputs a hypothesis circuit of size $O(n^c)$ which computes g on at least $(1/2 + 1/n^k)$-fraction of n-length inputs. Thus, for every function in BPE/poly, there exists a family of polynomial-sized circuits $\{h_n\}_{n\in\mathbb{N}}$ which $(1/2 + 1/n^k)$-approximates it, where h_i is the hypothesis output by the learner on input length i.

We next show that the existence of such a learner implies the existence of a function in BPE which cannot be $(1/2 + 1/n^k)$-approximated by polynomial sized circuits. Consider the PSPACE-Complete function f^* from Theorem A.8 which is computable in time $\mathsf{DSPACE}[n]$. f^* is in BPE/poly (since f^* can be computed in E) and we use the learning algorithm for BPE/poly in Lemma 2.1 to see that PSPACE $\subseteq$ BPP. Using a padding argument we observe that $\mathsf{DSPACE}[2^{O(n)}] \subseteq \mathsf{BPE}$. From Lemma A.11, we see that there exists a function which cannot be $(1/2 + 1/n^k)$-computed by circuits of size $n^{\log n}$. We can easily construct a Turing Machine which lexicographically searches for the truth table of a function on n inputs which cannot be $(1/2 + 1/n^k)$-approximated by $n^{\log n}$ sized circuits in $2^{O(n)}$ space and answers according to the first one it finds. From this we have that $\mathsf{DSPACE}[2^{O(n)}]$, and thus BPE/poly cannot be $(1/2 + 1/n^k)$-approximated by $n^{\log n}$ sized circuits, which leads to a contradiction. ◀

▶ Remark 2.2. [36] show that if for each c, a circuit class $\mathcal{C}[n^c]$ is $(1/2 - 1/n^c, 1/n)$-learnable using membership queries over the uniform distribution in $2^n/n^{\omega(1)}$ time, then for each c, there exists $L_c \in \mathsf{BPE}$ such that $L_c \notin \mathcal{C}[n^k]$ (Theorem 12). For any c, the idea of picking $\mathcal{C}[n^c] = \mathsf{SIZE}^{\mathsf{BPE}}[n^c]$, with linear-sized queries to BPE oracles and using the learning algorithm A which learns BPE/poly in their result to achieve a contradiction (as any function in BPE can be computed by constant sized $\mathsf{SIZE}^{\mathsf{BPE}}$-circuits with linear-sized oracle queries) does not work, as [36] crucially uses that $\mathcal{C}[n^c]$ has to be a subset of $\mathsf{SIZE}[n^{c'}]$ for some $c' = O(c)$.

On the other hand, Theorem 4 in [15] shows that if $\mathcal{C}$ is learnable using membership queries over the uniform distribution in polynomial time then $\mathsf{BPE} \not\subseteq \mathcal{C}[\mathsf{poly}(n)]$. Proving Theorem 1.1 by setting $\mathcal{C}$ as BPE/poly again does not really work, as [15]'s result only holds true when $\mathcal{C} = \mathsf{P/poly}$, as it depends on the collapse of EXP to P/poly.

We next consider hardness of learning E/poly deterministically over the uniform distribution using membership queries. E/poly can equivalently be defined as the class of languages which can be computed by polynomial-sized circuit families with oracle access to some function in E, with the constraint that the oracle queries are of size $O(n)$.

We first rule out deterministic *exact* learners for $\mathsf{E/poly}$ running in time $O(2^n/n)$ in Angluin's model of learning [4], i.e. the learners have access to a membership oracle, as well as an equivalence oracle, where the learner presents a hypothesis circuit to the equivalence oracle, receives yes if the hypothesis exactly computes the target concept and receives a counter-example for the hypothesis, otherwise.

We also extend this result to rule out any deterministic learners for $\mathsf{E/poly}$ using membership queries, i.e. even learners which can output an approximate hypothesis. In particular, we can show that, for every constant $\delta \in [0, 1/2 - 1/n)$, $\mathsf{E/poly}$ is hard to learn up to error δ over the uniform distribution using membership queries, even by deterministic learning algorithms which run in time $2^n/n$. These ideas can also be extended to show similar results for unconditional hardness of learning $\mathsf{PSPACE/poly}$ by deterministic polynomial time learners. The proofs follow from simple diagonalization-like arguments such as that used by [31] and can be found in the full version.

3 Robustness of Hardness of Learning

In this section, we establish the equivalences in Theorem 1.2 for hardness of learning $\mathsf{EXP/poly}$. We first state the following results necessary for its proof.

▶ **Lemma 3.1.** *Let* $\mathsf{EXP} = \mathsf{BPP}$*. Then, for every* $c > 0$*,* $\mathsf{EXP/poly}$ *can be* $(1/n^c, 1/20n)$*-PAC-learnt using random examples in time polynomial in* n*.*

Proof Sketch. The proof uses the collapse of EXP into BPP, firstly to observe that $\mathsf{EXP/poly}$ is in $\mathsf{P/poly}$. Next, we construct an exponential time procedure which takes as inputs a size parameter $s(n)$, a set of examples of length n and their labels, and outputs a hypothesis circuit of size at most $s(n)$ which is consistent with the examples, if there exists one, via an exhaustive search. From our assumption, this runs in probabilistic polynomial time. Using an argument based on Occam's razor [30], we obtain a polynomial time learner for $\mathsf{P/poly}$. ◀

▶ **Lemma 3.2.** *Let* $\mathsf{EXP} \neq \mathsf{BPP}$*. Then, there exists* $c \geq 0$ *such that* $\mathsf{EXP/poly}$ *is not* $(1/n^c, 1/20n)$*-learnable in the worst-case using random examples over the uniform distribution in time polynomial in* n*.*

Proof. Towards a contradiction assume that there exists a constant $a > 0$ and an $O(n^a)$-time learner $\mathcal{A}$ that $(1/n^c, 1/20n)$-learns $\mathsf{EXP/poly}$ using random examples over U_n, for every $c \geq 0$. We first show that the existence of the learner $\mathcal{A}$ for $\mathsf{EXP/poly}$ implies that $\mathsf{EXP} \subseteq \mathsf{P/poly}$. Let g^* be an EXP-Complete problem which is self-correctible, whose existence is given by Theorem A.7, with $c_1 \geq 0$ being the corresponding constant. Use $\mathcal{A}$ to $(1/n^{c_1}, 1/20n)$-learn g^* using random examples over the uniform distribution. Let $\mathcal{A}'$ be the algorithm which takes as input $y \in \{0,1\}^n$ in addition to the inputs of $\mathcal{A}$ and runs the learner $\mathcal{A}$, following which it returns the evaluation of the hypothesis circuit output by $\mathcal{A}$ on the input y. In other words, for every $n \in \mathbb{N}$, we have

$$\Pr_{\substack{w \in \{0,1\}^{r(n)} \\ x_1,\ldots,x_m \sim U_n}} \left\{ \Pr_{y \sim U_n} \{\mathcal{A}'(1^n, w, (x_1, g^*(x_1)), \ldots, (x_m, g^*(x_m)), y) = g^*(y)\} \geq 1 - 1/n^{c_1} \right\}$$
$$\geq 1 - 1/20n$$

where both $r(n)$ and $m = m(n) = \mathsf{poly}(n)$.

By amplifying the correctness of $\mathcal{A}'$ using standard techniques, we can then non-uniformly fix the random strings $w, x_1, \ldots, x_m$ and the values of g^* on each x_i to get a polynomial sized circuit C, which takes input $y \in \{0,1\}^n$ and outputs the answer of $\mathcal{A}'$ on the advice string and y. Thus C_n agrees with g^* on at least $(1 - 1/n^{c_1})$-fraction of the inputs. Using C_n with the self-correctibility of g^* (and fixing another "good" random string non-uniformly in the resulting algorithm), we get a polynomial-sized circuit which computes g^* on every input and by the EXP-Completeness of g^*, we see that $\mathsf{EXP} \subseteq \mathsf{P/poly}$.

Since $\mathsf{EXP} \subseteq \mathsf{P/poly}$, we use Lemma A.9 to observe that f^* given by Theorem A.8 is now an EXP-Complete problem that is both downward self-reducible and self-correctible. Let c_2 be the constant associated with the self-corrector for f^*. For any integer k, given a procedure B_k which computes f^* on every instance of size k with high probability, we use $\mathcal{A}$ together with the downward self-reduction for f^*, followed by the self-corrector for f^* to obtain a procedure B_{k+1} that computes f^* on any input of size $k+1$. We use this inductively, to compute f^* on n inputs in probabilistic polynomial time.

More precisely, consider the following algorithm B_n which computes f^* on a given input x and does the following. First, it starts with a procedure B_{k_0}, for a constant k_0, which can be computed easily using a look-up table. Assuming that we have the procedure B_k for some input length $k \leq n$, we show how to construct the procedure B_{k+1} inductively. We use the learner $\mathcal{A}$ to learn the function f^*_{k+1} up to error $1/(k+1)^{c_2}$. For every input $f^*(y)$ passed to $\mathcal{A}$, where y is a string randomly picked from $\{0,1\}^{k+1}$, we use B_k with the downward self-reduction of f^* to compute $f^*(y)$. $\mathcal{A}$ outputs a hypothesis h_{k+1} which computes f^*_{k+1} on at least a $(1 - 1/(k+1)^c)$-fraction of the inputs with high probability. We now use the self-corrector for f^* to obtain from h_{k+1} a procedure B_{k+1} which is correct on every input of size $k+1$ with probability $1 - \gamma$ (by using standard error reduction arguments), for some $\gamma > 0$ which we pick later. Repeating this process at most n times, we obtain B_n and output $B_n(x)$.

First, we show that B_n outputs $f^*(x)$ with probability at least $2/3$. Let $d(n)$ be the number of queries made by the DSR to the oracle f^*_{n-1} in computing $f^*(x)$ on any input x of length n. The idea is that at each stage k, the procedure B_k fails only if at least one of $m(n) \cdot d(n)$ queries answered by B_{k-1} is incorrect, with probability at most $m(n)d(n)\gamma \leq 1/20n$ for $\gamma = 1/20nm(n)d(n)$, or if $\mathcal{A}$ fails to output the right hypothesis, with probability at most $1/20n$. Thus, the total failure probability at each stage is at most $1/10n$ and over the n stages, using the union bound, the total failure probability is at most $1/10 + \gamma \leq 1/3$.

We inductively observe that every stage B_k runs in time $\mathsf{poly}(k)$. It is easily seen that B_{k_0} runs in constant time. Assume that B_{k-1} runs in $\mathsf{poly}(k-1)$ time. At stage k, the time taken to compute f^* on $m(k)$ many inputs of length k is $O(m(k) \cdot d(k) \cdot \mathsf{poly}(k-1)) \leq \mathsf{poly}(k)$. After this, $\mathcal{A}$ takes $O(k^d)$ time to output h_k of size at most k^d, which is used by the $\mathsf{poly}(k)$-time self-corrector to compute f^* on all inputs of size k with high probability. Thus, B_k runs in time $\mathsf{poly}(k) = \mathsf{poly}(n)$. Since there at most n stages, the total running time of B_n is $\mathsf{poly}(n)$. This shows that $f^* \in \mathsf{BPP}$ and contradicts the original assumption. ◀

Using a very similar proof idea, we obtain an analogous statement to Lemma 3.2, but now for worst-case learning $\mathsf{EXP/poly}$ using membership queries.

▶ **Lemma 3.3.** *Suppose that* $\mathsf{EXP/poly}$ *is* $(1/n^c, 1/20n)$*-learnable in the worst case for every* $c \geq 0$ *over the uniform distribution* U_n *using membership queries in time* $\mathsf{poly}(n)$*. Then,* $\mathsf{EXP} = \mathsf{BPP}$.

Informally (see Appendix A for a formal definition), the task of an (ε, δ)-average-case learner for a class $\mathcal{C}$ over the uniform distribution using random examples, is to (ε, δ)-learn an unknown target function which is be generated according to a fixed distribution over

$\mathcal{C}$ (defined as an ensemble of distributions over appropriate representations for $\mathcal{C}$). The ideas from Lemma 3.2 can also be extended to show similar implications for the setting of average-case learning $\mathsf{EXP/poly}$ using random examples (and membership queries too).

▶ **Lemma 3.4.** *Suppose that* $\mathsf{EXP/poly}$ *is* $(1/n^c, 1/20n)$*-learnable on average for every* $c \geq 0$ *with respect to polynomially samplable distributions over* $\mathsf{EXP/poly}$ *and the uniform distribution* U_n *using random examples in time* $\mathsf{poly}(n)$*. Then,* $\mathsf{EXP} = \mathsf{BPP}$*.*

▶ Remark 3.5. In Lemma 3.4, the samplable distributions we consider are over $\mathsf{SIZE}^{\mathsf{EXP}}[n^k]$-circuit encodings in $R_n \subseteq \{0,1\}^{r(n)}$, where $r(n) = O(n^{2k+1})$ (see Remark A.3 for more details). In particular, for the distribution $\mathcal{P}$ over $\mathsf{SIZE}^{\mathsf{EXP}}[n^k]$ supported only on the function g^* (or f^*) used in the proof of Lemma 3.2, the sampler $S'_{\mathcal{P}}$ takes 1^{n^k} as input and outputs a $\mathsf{SIZE}^{\mathsf{EXP}}[n^k]$-circuit encoding of g^* (or f^*) which is just an EXP-oracle gate on n inputs and this encoding is of size $O(n^2)$. The running time of this sampling algorithm is polynomial in the input size.

We use the same ideas as that of Lemma 3.2 to prove that even learning $\mathsf{EXP/poly}$ in polynomial time using random examples from the uniform distribution with respect to just these two distributions over $\mathsf{EXP/poly}$, is enough to collapse EXP to BPP.

The formal details of the intermediate results can be found in the full version. We now prove the equivalences for efficiently learning $\mathsf{EXP/poly}$.

Proof of Theorem 1.2. The following implications establish the desired equivalences.
$(b) \implies (a), (c) \implies (a)$: The contrapositives of each of these implications follow from Lemma 3.1. In particular, PAC-learning $\mathsf{EXP/poly}$ with error at most $1/n^c$ for any $c > 0$ using random examples, implies PAC-learnability of $\mathsf{EXP/poly}$ using membership queries, where the queries are just made on the random examples given to the learner.
$(d) \implies (b), (e) \implies (c)$: Follows from the definitions, since PAC-learning $\mathsf{EXP/poly}$ in the worst case in $\mathsf{poly}(n)$ time using random examples implies PAC-learnability for $\mathsf{EXP/poly}$ on average in $\mathsf{poly}(n)$ time using random examples, for any distribution over $\mathsf{EXP/poly}$. A similar implication holds for learning with membership queries too.
$(a) \implies (b)$: For any $c > 0$, suppose $\mathsf{EXP/poly}$ is $(1/n^c, 1/20n)$ PAC-learnable in polynomial time using random examples over every arbitrary distribution. In particular, this means that $\mathsf{EXP/poly}$ can be $(1/n^c, 1/20n)$-learnt in polynomial time using random examples over the uniform distribution. The implication follows from the contrapositive of Lemma 3.2.
$(a) \implies (c)$: Similar to the previous implication, we see that $\mathsf{EXP/poly}$ is $(1/n^c, 1/20n)$-learnable in polynomial time using membership queries over the uniform distribution. The implication holds from the contrapositive of Lemma 3.3.
$(a) \implies (d), (a) \implies (e)$: The implications follow from Lemma 3.4 and its corresponding extension to learning on average with membership queries. ◀

The proof of Corollary 1.3 (equivalences for learning $\mathsf{PSPACE/poly}$) follows from the same ideas as Theorem 1.2. In more detail, Lemma 3.1 extends easily as the procedure which searches for a polynomial-sized consistent hypothesis also runs in polynomial space. Lemmas 3.2, 3.3 and 3.4 can also be extended, by learning the downward-self-reducible and self-correctible PSPACE-Complete function f^* (from Theorem A.8) directly, and using it to compute f^* on every input.

4 Reducing Succinct Search to Decision

The key concepts in this section are verifiability and succinct search. We define verifiers first.

▶ **Definition 4.1.** *Given language $L \subseteq \{0,1\}^*$ and polynomial-time computable relation $V(\cdot,\cdot)$, we say that V is a verifier for L if for each $x \in \{0,1\}^*$, $x \in L$ iff $\exists y V(x,y)$.*

Given language L, a verifier V for L, and function $f : \mathbb{N} \to \mathbb{N}$, we say that L has $f(n)$-size proofs with respect to V, such that for each $x \in \{0,1\}^*$, $x \in L$ implies $\exists y, |y| \leq f(|x|) : V(x,y)$. We say that L has $f(n)$-size proofs if there is a verifier V for L such that L has $f(n)$-size proofs with respect to V.

Given language L, a verifier V for L and a machine class $\mathcal{D}$, we say that L has $\mathcal{D}$-computable proofs with respect to V if there is a machine $M \in \mathcal{D}$ such that for each $x \in \{0,1\}^*$, $x \in L$ implies $V(x, M(x))$. We say that L has $\mathcal{D}$-computable proofs if there is a verifier V for L such that L has $\mathcal{D}$-computable proofs with respect to V.

Note that NP is the class of languages with polynomial-sized proofs, NEXP is the class of languages with exponential-sized proofs, and for $\mathcal{D} \in \{\mathsf{EXP}, \mathsf{PSPACE}\}$, $\mathcal{D}$ is the class of languages with $\mathcal{D}$-computable proofs (where we abuse notation and use $\mathcal{D}$ to refer both to a machine class and to the class of languages computable by such machines).

Next we define succinct search. We will assume w.l.o.g. that the proof size for any verifier is a power of 2 - this can be ensured by padding the proof if necessary.

▶ **Definition 4.2.** *Given language L and verifier V for L, we say that* ***succinct search is easy*** *for L with respect to V if there is a probabilistic polynomial-time machine N such that for each $x \in L$, there is a V-proof y such that with probability $1 - o(1)$, $\mathsf{tt}(N(x)) = y$, where for Boolean circuit C, $\mathsf{tt}(C)$ denotes the truth table of the function computed by C.*

Thus succinct search is easy for L with respect to a verifier V if there is a probabilistic polynomial-time machine outputting compressed descriptions of V-proofs with high probability for any positive instance of L.

Using the downward self-reducibility of SAT, it is straightforward to see that $\mathsf{NP} \subseteq \mathsf{BPP}$ iff for each $L \in \mathsf{NP}$ and for every verifier V such that L has poly-size proofs with respect to V, succinct search is easy for L with respect to V. We now show analogous results for PSPACE, EXP and NEXP. First we show for each of these classes that easiness of the class implies easiness of succinct search.
We need the Easy Witness Lemma of Impagliazzo, Kabanets and Wigderson [24].

▶ **Lemma 4.3** ([24]). *If $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, then for each $L \in \mathsf{NEXP}$ and for each verifier V for L such that L has exponential-size proofs with respect to V, for each $x \in L$, there is a polynomial-size circuit C_x such that $V(x, \mathsf{tt}(C_x))$ holds.*

▶ **Lemma 4.4.** *The following implications hold:*
1. *Let $\mathcal{D} \in \{\mathsf{PSPACE}, \mathsf{EXP}\}$. If $\mathcal{D} = \mathsf{BPP}$, then for each $L \in \mathcal{D}$ and for each verifier V such that L has $\mathcal{D}$-computable proofs with respect to V, succinct search is easy for L with respect to V.*
2. *If $\mathsf{NEXP} = \mathsf{BPP}$, then for each $L \in \mathsf{NEXP}$ and for each verifier V such that L has exponential-size proofs with respect to V, succinct search is easy for L with respect to V.*

Proof. We establish the first item. Let $\mathcal{D} \in \{\mathsf{PSPACE}, \mathsf{EXP}\}$, and assume $\mathcal{D} = \mathsf{BPP}$. Let $L \in \mathcal{D}$ and V be a verifier for L such that L has $\mathcal{D}$-computable proofs with respect to V. We construct a probabilistic poly-time machine N such that for each input $x \in L$, there is a V-proof y such that with high probability $\mathsf{tt}(N(x)) = y$. Let M be a $\mathcal{D}$-machine outputting V-proofs for positive instances of L.

Consider the language $L' = \{\langle x, i\rangle | i^{\text{th}}$ bit of $M(x)$ is $1\}$. Since M is a $\mathcal{D}$ machine, we have that $L' \in \mathcal{D}$. By assumption, $\mathcal{D} = \mathsf{BPP}$, therefore there is a probabilistic poly-time machine N' deciding L'. Assume w.l.o.g. that N' has error at most $2^{-|y|^2}$ on any input y. Given input x, N operates as follows. It first computes a probabilistic poly-size circuit C' simulating N'. This can be done using the standard efficient conversion of efficient algorithms into small circuits. It then hardwires x into the first part of the input for C', obtaining a circuit C'_x. It then fixes the random input of the circuit C'_x to a uniformly generated random string r to obtain a circuit $D'_{x,r}$, which it outputs.

Since the error of N' is smaller than $2^{-|y|^2}$ on any input y, by a simple union bound, with probability $1 - o(1)$ over the choice of the random string r, $D'_{x,r}$ correctly computes the i'th bit of $M(x)$ for each $i \in [m]$. For $x \in L$, $V(x, M(x))$ holds, and therefore N efficiently solves succinct search for L with respect to V.

We establish the second item. Assume $\mathsf{NEXP} = \mathsf{BPP}$ and let $L \in \mathsf{NEXP}$ and V be a verifier for L such that L has exponential-size proofs with respect to V. Since $\mathsf{NEXP} = \mathsf{BPP}$, we have that $\mathsf{NEXP} \subseteq \mathsf{P/poly}$. By Lemma 4.3, there is a polynomial p such that for each $x \in L$, there is a circuit C_x of size at most $p(|x|)$ such that $V(x, \mathsf{tt}(C_x))$ holds.

Consider the language $L' = \{\langle x, i\rangle |$ There is a circuit C of size $p(|x|)$ such that $V(x, \mathsf{tt}(C))$ is 1, and the i^{th} bit of the lexicographically first such circuit is $1\}$. Clearly $L' \in \mathsf{EXP}$, just by enumerating circuits of size $p(|x|)$ in lexicographic order and finding the first one encoding a V-proof for x, if one exists. Since $\mathsf{EXP} = \mathsf{BPP}$, there is a probabilistic poly-time machine N' deciding L' with error exponentially small. We construct a probabilistic poly-time machine N as follows: on input x, N runs N' on $\{\langle x, i\rangle\}$ for each i at most the description length of a circuit of size $p(|x|)$. It outputs the circuit C whose description has bit i set to 1 iff N' accepts on $\{\langle x, i\rangle\}$. Since N' has error exponentially small, we have that with error exponentially small, N outputs a circuit C encoding a V-proof of x, and therefore N efficiently solves succinct search for L with respect to V. ◀

For the reverse directions, we use the PCP characterization of NEXP [6, 16], where we only require polynomial upper bound on query complexity of the verifier.

▶ **Theorem 4.5** ([6, 16]). *Let $L \in \mathsf{NEXP}$. There is a probabilistic poly-time oracle machine V' such that:*

1. *For each $x \in L$, there is y of length exponential in $|x|$ such that $V'(x)$ accepts with probability at least $2/3$ when given oracle access to y.*
2. *For each $x \notin L$ and for all y, $V'(x)$ accepts with probability at most $1/3$ when given oracle access to y.*

We now show that easiness of succinct search implies easiness of decision for any $L \in \mathsf{NEXP}$.

▶ **Lemma 4.6.** *Let $L \in \mathsf{NEXP}$ and V be a verifier such that L has exponential-size proofs with respect to V. If succinct search is easy for L with respect to V, then $L \in \mathsf{BPP}$.*

Proof. Let $L \in \mathsf{NEXP}$. We show that $L \in \mathsf{BPP}$. By Theorem 4.5, there is a probabilistic poly-time oracle machine V' such that if $x \in L$, there is y of length exponential in $|x|$ for which V' accepts with high probability on x when given oracle access to y, and if $x \notin L$ rejects with high probability irrespective of the oracle.

Now consider a verifier V for L which given input x and proof y, accepts iff $V'(x)$ accepts with oracle y on a majority of its computation paths. Since succinct search is easy for L with respect to V, there is a probabilistic poly-time machine N such that for input $x \in L$, there is a V-proof y for x such that with high probability $\mathsf{tt}(N(x)) = y$. We define a probabilistic poly-time machine W that on input x simulates $V'(x)$ as follows. It first runs $N(x)$ to find a circuit C. It then runs $V(x)$, answering all oracle calls to y by simulating C on input corresponding to the bit of y that is queried. It accepts iff $V(x)$ accepts.

If $x \in L$, by using the assumption that N solves succinct search, $W(x)$ accepts with probability close to 2/3. If $x \notin L$, $W(x)$ rejects with probability close to 2/3 since the circuit C output by $N(x)$ corresponds to some purported V'-proof, and every such V'-proof is rejected with high probability by V when given oracle access to the proof. ◀

▶ **Theorem 4.7.** *Let $\mathcal{D} \in \{\mathsf{PSPACE}, \mathsf{EXP}\}$. $\mathcal{D} = \mathsf{BPP}$ iff for each $L \in \mathcal{D}$ and for each verifier V for L such that L has $\mathcal{D}$-computable proofs with respect to V, succinct search is easy for L with respect to V.*

$\mathsf{NEXP} = \mathsf{BPP}$ *iff for each $L \in \mathsf{NEXP}$ and for each verifier V such that L has exponential-size proofs with respect to V, succinct search is easy for L with respect to V.*

Proof. The forward directions of both items follow from Lemma 4.4. The backward direction of the second item follows Lemma 4.6. The backward direction of the first item follows from Lemma 4.6 and the fact that for $\mathcal{D} \in \{\mathsf{PSPACE}, \mathsf{EXP}\}$, if $L \in \mathcal{D}$ and V is a verifer for L such that L has $\mathcal{D}$-computable proofs with respect to V, then $L \in \mathsf{NEXP}$ and L has exponential-size proofs with respect to V. ◀

We now prove Theorem 1.4. Define R_{Kt} as the language consisting of strings x such that $\mathsf{Kt}(x) \geq |x|/2$. Similarly, R_{KS} is the language consisting of strings x such that $\mathsf{KS}(x) \geq |x|/2$ [3] (see Appendix A.3 for formal definitions of Kt and KS complexity).

▶ **Theorem 4.8** (Theorem 1.4 stated formally). $\mathsf{R}_{\mathsf{Kt}} \in \mathsf{BPP}$ *iff for each verifier V for R_{Kt} such that R_{Kt} has EXP-computable proofs with respect to V, succinct search is easy for R_{Kt} with respect to V.*

$\mathsf{R}_{\mathsf{KS}} \in \mathsf{BPP}$ *iff for each verifier V for R_{KS} such that R_{KS} has PSPACE-computable proofs with respect to V, succinct search is easy for R_{KS} with respect to V.*

Proof. The backward directions of both items follow from Lemma 4.6 and the facts that R_{Kt} and R_{KS} are in NEXP.

For the forward direction of the first item, we use the result shown in [3] that $\mathsf{R}_{\mathsf{Kt}} \in \mathsf{BPP}$ implies $\mathsf{EXP} = \mathsf{BPP}$. Combining this with the first item of Lemma 4.4 completes the proof.

For the forward direction of the second item, we use the theorem shown in [3] that $\mathsf{R}_{\mathsf{KS}} \in \mathsf{BPP}$ implies $\mathsf{PSPACE} = \mathsf{BPP}$. Combining this with the first item of Lemma 4.4 completes the proof. ◀

References

1 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on np-hardness. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 701–710. ACM, 2006. doi:10.1145/1132516.1132614.

2 Eric Allender. When worlds collide: Derandomization, lower bounds, and kolmogorov complexity. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 1–15. Springer, 2001.

3 Eric Allender, Harry Buhrman, Michal Koucký, Dieter Van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35(6):1467–1493, 2006.

4 Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

5 Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 211–220. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.35.

6 László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity*, 1(1):3–40, 1991.

7 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486.

8 Avrim Blum, Merrick Furst, Michael Kearns, and Richard J Lipton. Cryptographic primitives based on hard learning problems. In *Annual International Cryptology Conference*, pages 278–291. Springer, 1993.

9 Andrej Bogdanov and Christina Brzuska. On basing size-verifiable one-way functions on np-hardness. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2015. doi:10.1007/978-3-662-46494-6_1.

10 Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for np problems. *SIAM Journal on Computing*, 36(4):1119–1159, 2006.

11 Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Proceedings. Thirteenth Annual IEEE Conference on Computational Complexity (Formerly: Structure in Complexity Theory Conference)(Cat. No. 98CB36247)*, pages 8–12. IEEE, 1998.

12 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 10:1–10:24. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.CCC.2016.10.

13 Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 70:1–70:48. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.70.

14 Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, 1993.

15 Lance Fortnow and Adam R Klivans. Efficient learning algorithms yield circuit lower bounds. *Journal of Computer and System Sciences*, 75(1):27–36, 2009.

16 Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 134(2):545–557, 1994.

17 Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

18 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct randolli functions. In *25th Annual Symposium on Foundations of Computer Science, 1984.*, pages 464–479. IEEE, 1984.

19 Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP languages are hard on the worst-case, then it is easy to find their hard instances. *Comput. Complex.*, 16(4):412–441, 2007. doi:10.1007/s00037-007-0235-8.

20 Dan Gutfreund and Salil P. Vadhan. Limitations of hardness vs. randomness under uniform reductions. In Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, volume 5171 of *Lecture Notes in Computer Science*, pages 469–482. Springer, 2008. doi:10.1007/978-3-540-85363-3_37.

21 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 247–258. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00032.

22 Shuichi Hirahara and Osamu Watanabe. On nonadaptive reductions to the set of random strings and its dense subsets. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:25, 2019. URL: `https://eccc.weizmann.ac.il/report/2019/025`.

23 Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 104–114. IEEE Computer Society, 2011. `doi:10.1109/CCC.2011.34`.

24 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

25 Russell Impagliazzo and Levin LA. No better ways to generate hard np instances than picking uniformly at random. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 812–821. IEEE, 1990.

26 Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001. `doi:10.1006/jcss.2001.1780`.

27 Jeffrey C Jackson. An efficient membership-query algorithm for learning dnf with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997.

28 Richard M Karp and Richard J Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 302–309. ACM, 1980.

29 Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

30 Michael J Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory.* MIT press, 1994.

31 Adam R. Klivans, Pravesh Kothari, and Igor Carboni Oliveira. Constructing hard functions using learning algorithms. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 86–97. IEEE Computer Society, 2013. `doi:10.1109/CCC.2013.18`.

32 Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

33 Leonid A Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.

34 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM (JACM)*, 40(3):607–620, 1993.

35 Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 2–10. IEEE, 1990.

36 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 18:1–18:49. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. `doi:10.4230/LIPIcs.CCC.2017.18`.

37 Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

38 Luca Trevisan and Salil Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

39 Chee K Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical computer science*, 26(3):287–300, 1983.

A Preliminaries

Let $\mathcal{F} = \{\mathcal{F}_n\}$, where $\mathcal{F}_n$ is set of all Boolean functions over $\{0,1\}^n$, where each $f_n \in \mathcal{F}_n$ is a function $f_n : \{0,1\}^n \to \{0,1\}$. Define $\mathsf{tt}(f)$ as the truth table of a function f_n of length 2^n. On the other hand, given a string $x \in \{0,1\}^{2^n}$, define $\mathsf{fn}(x)$ as the function on n inputs whose truth table is x. For every $n \in \mathbb{N}$, define U_n as the uniform distribution over $\{0,1\}^n$.

A.1 Samplability and Learnability

Let $\mathcal{C} = \{\mathcal{C}_n\}$, where $\mathcal{C}_n \subseteq \mathcal{F}_n$ be a class of functions over $\{0,1\}^n$ and $\mathcal{D} = \{\mathcal{D}_n\}$ be a distribution family over $\{0,1\}^*$, where $\mathcal{D}_n$ is a distribution over $\{0,1\}^n$.

▶ **Definition A.1** (Worst-case PAC-learning using random examples). *For any $0 \leq \epsilon, \delta < 1/2$, a class $\mathcal{C}$ is (ϵ,δ)-PAC-learnable in the worst-case using random examples in time $T(n)$, if there exists a randomized algorithm $\mathcal{A}$ such that*

- *For every $n \in N$, for every $f \in \mathcal{C}_n$, for every $\mathcal{D}_n$ over $\{0,1\}^n$, $\mathcal{A}$ takes inputs $1^n, \epsilon, \delta$, a set of $m = m(n)$ labeled samples $(x_1, f(x_1)), \ldots, (x_m, f(x_m))$ where each $x_i \sim \mathcal{D}_n$, and $w \in \{0,1\}^{r(n)}$ as internal randomness. $\mathcal{A}$ then outputs the description of a circuit h such that*

$$\Pr_{w\in\{0,1\}^{r(n)}, x_1,\ldots,x_m\sim\mathcal{D}_n}\left\{\Pr_{y\in\mathcal{D}_n}\{h(y)=f(y)\} \geq 1-\varepsilon\right\} \geq 1-\delta$$

- *$\mathcal{A}$ runs in time at most $T(n)$.*[11]

We can also restrict the learnability to a fixed distribution like the uniform distribution U_n, where the learner takes random examples chosen over the uniform distribution and hypothesis error is also measured over the uniform distribution. Unless specified otherwise, we use the class of polynomial-sized Boolean circuits $\mathsf{P/poly}$, as the hypothesis class for our learning algorithms.

Furthermore, we can extend this definition to PAC-learning over membership queries by giving the learner $\mathcal{A}$ oracle access to the function $f \in \mathcal{C}_n$, in addition to the random examples drawn from some fixed distribution $\mathcal{D}_n$ over $\{0,1\}^n$.

To define learnability on average, let $\mathcal{P} = \{\mathcal{P}_n\}$ be a distribution ensemble over $\mathcal{C}$, where $\mathcal{P}_n$ is a fixed distribution over $\mathcal{C}_n$.

▶ **Definition A.2** (Samplable distributions). *Let $\mathcal{P}$ be a distribution ensemble over $\mathcal{C}$, where for every $n \in \mathbb{N}$, $\mathcal{P}_n$ is a distribution over the truth tables of $\mathcal{C}_n$. Let $N = 2^n$. For any non-decreasing function $S(N) \geq N$, we say that $\mathcal{P}$ is samplable in time $S(N)$, if there exists a randomized algorithm A such that for every $N = 2^n$, using $m(N)$ bits of randomness (where $m(N) \leq S(N)$), $A(1^N, y)$ is distributed identically to $\mathcal{P}_n$, where the distribution is over the string y picked uniformly at random from $\{0,1\}^{m(N)}$ and A runs in time $S(N)$.*

In other words, if y is picked uniformly at random from $\{0,1\}^{m(N)}$ then $A(1^N, y)$ outputs a truth table from $\mathcal{C}_n$ which is distributed according to $\mathcal{P}_n$. Furthermore, we say that $\mathcal{P}$ is polynomially samplable if $S(N) = \mathsf{poly}(N)$.

▶ Remark A.3. For the special case where $\mathcal{C}$ is a class of fixed polynomial sized circuits like $\mathsf{SIZE}[n^k]$ (or $\mathsf{SIZE}^{\mathsf{EXP}}[n^k]$) for any arbitrary fixed k, we define a circuit representation scheme for $\mathcal{C}_n$ given by the set $R_n \subset \{0,1\}^{r(n)}$, where $r(n) = O(n^k \log n)$, such that every $\sigma \in R_n$ is

[11] Note that this immediately implies that $m(n) \leq T(n)$

a $\mathcal{C}$-circuit encoding of a function in $\mathcal{C}_n$. Note that this mapping is onto and each function in $\mathcal{C}_n$ has many representations in R_n. We also assume that there exists a uniform circuit sequence in $\mathcal{C}$, which interprets this encoding as a $\mathcal{C}$-circuit and evaluates computations given this encoding.

Now, we can define a distribution ensemble $\mathcal{P}$ over $\mathcal{C}$, where each $\mathcal{P}_n$ is a distribution over the $\mathcal{C}$-circuit encodings, which implicitly defines a distribution over $\mathcal{C}_n$. We also define $S(r(n))$-samplability of $\mathcal{P}$, if there exists a randomized algorithm A running in time $S(r(n))$ such that for every $n \in \mathbb{N}$, $A(1^{r(n)}, y)$ is distributed identically to $\mathcal{P}_n$, where the distribution is over the random strings $y \in \{0,1\}^{m(n)}$.

▶ **Definition A.4** (Average-case learnability [8]). *Let $\mathcal{C}$ be a class of Boolean functions and $\mathcal{P} = \{\mathcal{P}_n\}$ be a distribution ensemble over $\mathcal{C}$. For any $0 < \epsilon, \delta < 1/2$, we say that $\mathcal{C}$ is (ϵ, δ)-PAC-learnable on average using random examples with respect to $\mathcal{P}$ in time $T(n)$, if there exists a randomized algorithm $\mathcal{A}$ running in time at most $T(n)$ such that*

- *For every large enough n, for any fixed f drawn according to $\mathcal{P}_n$, for every $\mathcal{D}_n$ over $\{0,1\}^n$, $\mathcal{A}$ takes inputs $1^n, \varepsilon, \delta$, a set of $m = m(n)$ labeled samples $(x_1, f(x_1)), \ldots, (x_m, f(x_m))$ where each $x_i \sim \mathcal{D}_n$ and $w \in \{0,1\}^*$ (the internal randomness of $\mathcal{A}$) and outputs the description of a circuit h such that*

$$\Pr_{\substack{f\sim\mathcal{P}_n \\ w\in\{0,1\}^* \\ x_1,\ldots,x_m,y\sim\mathcal{D}_n}} \left\{ \Pr_{y\in\mathcal{D}_n}\{h(y) = f(y)\} \geq 1-\varepsilon \right\} \geq 1-\delta$$

- *$\mathcal{A}$ runs in time at most $T(n)$.*

Furthermore, for any $0 < \epsilon, \delta < 1/2$, we say that $\mathcal{C}$ is (ϵ, δ)-PAC-learnable on average with respect to polynomially samplable distributions over $\mathcal{C}$ using random examples in time $T(n)$ if there exists a learning algorithm $\mathcal{A}$ that runs in time $T(n)$ such that for every polynomially samplable distribution ensemble $\mathcal{P}$ over $\mathcal{C}$, we have that for every large enough n, $\mathcal{A}$ (ϵ, δ)-PAC-learns $\mathcal{C}_n$ on average using random examples with respect to $\mathcal{P}_n$.

We can naturally extend this definition to average-case learning $\mathcal{C}$ with respect to $\mathcal{P}$ and a fixed distribution over the examples like U_n, as well as average-case PAC-learning $\mathcal{C}$ with membership queries with respect to $\mathcal{P}$.

A.2 Self-Reducibility

In our reductions, we use the following special properties of a function.

▶ **Definition A.5** (Downward self-reducibility). *A function $f_n : \{0,1\}^n \to \{0,1\}$ is downward-self-reducible if there is a deterministic polynomial time algorithm A such that for all $x \in \{0,1\}^n$, $A^{f_{n-1}}(x) = f_n(x)$.*

▶ **Definition A.6** (Self-Correctibility). *A function $f : \{0,1\}^n \to \{0,1\}$ is said to be self-correctible if there exists a constant $c \geq 0$ and a probabilistic polynomial-time algorithm A such that, for every large enough n, for any function $\mathcal{O} : \{0,1\}^n \to \{0,1\}$ that agrees with f_n with probability $(1 - 1/n^c)$ over the uniform distribution on inputs of length n, we have that $\Pr\{A^{\mathcal{O}}(x) = f_n(x)\} \geq 2/3$ for any $x \in \{0,1\}^n$.*

[7] show that any function f on n Boolean inputs can be transformed into a function f^* on n inputs from a large enough finite field, such that f^* coincides with f on the subset $\{0,1\}^n$.

▶ **Theorem A.7** ([7]). *There exists an* EXP*-Complete problem* g^* *which is self-correctible.*

Furthermore, Trevisan and Vadhan [38] construct a PSPACE-Complete problem which is based on a careful arithmetization and padding of TQBF (using the interactive proof system for PSPACE), which has both these properties.

▶ **Theorem A.8** ([38]). *There exists a* PSPACE*-Complete language* $f^* \in \mathsf{DSPACE}[n]$ *that is both self-correctible and downward self-reducible (DSR).*

We also use the following results.

▶ **Lemma A.9.** *If* $\mathsf{EXP} \subseteq \mathsf{P/poly}$*, then* $\mathsf{EXP} = \mathsf{PSPACE}$*. In particular, the function* f^* *(from Theorem A.8) is complete for* EXP*.*

▶ **Lemma A.10** (Hoeffding's inequality). *Let* $X_1, \ldots, X_n$ *be independent random variables such that* $0 \le X_i \le 1$ *for every* $i \in [n]$*. Let* $X = \sum_{i=1}^{n} X_i$*. Then, for any* $t > 0$*, we have*

$$\Pr\{|X - \boldsymbol{E}[X]| \ge t\} \le 2\exp(-2t^2/n)$$

The following Lemma proves that a random function cannot even be approximated bysmall-sized circuits and follows from an application of Lemma A.10.

▶ **Lemma A.11** (Lemma 4 [36]). *For any* $s(n) \ge n$ *and* $\delta \in [0, 1/2]$*, we have*

$$\Pr_{f \sim \mathcal{F}_n}\{\exists \text{ circuit of size } \le s(n) \text{ computing } f \text{ on} \ge (1/2 + \delta)\text{-fraction of the inputs}\}$$
$$\le \exp(-\delta^2 2^n + 10 s \log s)$$

A.3 Kolmogorov Complexity

Fix a universal machine U. Levin [33] defined the following notion of time-bounded Kolmogorov complexity: The Kt complexity of a string x is the minimum $\mathsf{Kt}(x)$ over $|p| + \log(t)$ such that $U(p) = x$ in at most t steps. it is known [2] that $\mathsf{Kt}(x)$ is polynomially related to the size of the smallest EXP-oracle circuit computing the function with truth table x (truncating x to its longest initial segment with length a power of two).

Similarly, $\mathsf{KS}(x)$ is the minimum over $|p| + s$ such that $U(p) = x$ in at most space s. It is known [2] that $\mathsf{KS}(x)$ is polynomially related to the size of the smallest PSPACE-oracle circuit computing the function with truth table x (truncating x to its longest initial segment with length a power of two).

Let R_{Kt} be the language consisting of strings x such that $\mathsf{Kt}(x) \ge |x|/2$ [3]. Similarly, let R_{KS} be the language consisting of strings x such that $\mathsf{KS}(x) \ge |x|/2$ [3].

B Barriers for Conditional Hardness of Learning

Firstly, we formally define what it means to have a Black-Box Turing reduction from a language L to a PAC-learning algorithm for a class $\mathcal{C}$. Fix the error of the learner to be $\varepsilon = 1/\mathsf{poly}(n)$ (we ignore the confidence parameter, but this only makes our hardness results stronger).

▶ **Definition B.1** (Turing Reduction to Learning $\mathcal{C}$.). *A* B*-adaptive black-box reduction from deciding* L *to PAC-learning* $\mathcal{C}$ *using random examples up to error* ε*, is a tuple of probabilistic polynomial time algorithms* $R = (T_1, \ldots, T_B, M)$ *where* R *is given an input* $z \in \{0,1\}^n$ *and randomness* $w \in \{0,1\}^*$*. For each query,* R *constructs a joint distribution* $(X, f(X))$

over $\{0,1\}^r \times \{0,1\}$ for some $r \leq n$ and $f \in \mathcal{C}$, samples a set $S = \{(x_i, y_i)\}_{i \leq \mathsf{poly}(n)}$ of independent labeled examples according to (X, Y) and passes it to the learner. Let $t(n)$ be the query complexity of each round of adaptivity. R decides z by doing the following -

- *For each $1 \leq j \leq B$, T_j gets input z, fresh random bits from w and all the $(j-1) \cdot t(n)$ hypothesis circuits answered for the queries from the previous rounds (T_1 only has z and randomness w as input), and outputs $t(n)$ new queries $S_{j1}, \dots, S_{jt}$ for the learner, each of which are sets of labeled examples sampled from joint distributions $(X_{j1}, Y_{j1}), \dots, (X_{jt}, Y_{jt})$.*
- *R only has oracle access to the learner.*
- *M takes as input z, fresh random bits from w and the $B \cdot t(n)$ hypothesis circuits which are the answers made by the learner for all the queries asked by $T_1, \dots, T_B$, and outputs the answer.*
- *The reduction guarantees that if for every oracle A that is a $\mathcal{C}$-circuit learner, if every hypothesis circuit returned by the learner is $(1-\varepsilon)$-close with respect to its corresponding query given to the learner by $T_1, \dots, T_B$, then $M(z) = L(z)$ with high probability over the internal randomness of the reduction R.*

▶ **Definition B.2.** *For any B-adaptive black-box reduction $R = (T_1, \dots, T_B)$ from deciding L to PAC-learning $\mathcal{C}$ using random examples up to error ε, we have*

- *R is called **strongly black-box**, if $T_1, \dots, T_B, M$ only have oracle access to the hypothesis circuits and M decides L given access to any $(1-\varepsilon)$-close hypothesis circuit answered to each query made by $T_1, \dots, T_B$.*
- *If $B = 1$, we call the reduction as **non-adaptive**, and if R is strongly black-box and M also makes only non-adaptive queries to the hypotheses circuits, we call the reduction as **fully non-adaptive**.*
- *R is **oblivious**, if $T_1, \dots, T_B$ output new queries using only fresh randomness from w as input and access to the hypotheses generated during the previous rounds. Furthermore, M accesses each hypothesis using non-adaptively generated, identically distributed queries made from the corresponding distribution over which each hypothesis is guaranteed to be a good approximation. In particular, the obliviousness of the reduction implies the fact that the queries to the learner do not depend on the input z.*

Unless mentioned we think of the query complexity $t(n) = \mathsf{poly}(n)$. It is worth to note that since the algorithms $T_1, \dots, T_B$ are polynomial time algorithms, each joint distribution (X, Y) must be efficiently samplable.

We first prove Lemma 1.5. This is a reformulation of the proof of Lemma 3.2 used to show hardness of learning PSPACE/poly from a PSPACE-Complete language, into the framework of a black-box reduction.

Proof of Lemma 1.5. This a readaptation of the proof of Corollary 1.3 (via Lemma 3.2). Consider $R = (T_1, \dots, T_n, M)$ as an n-adaptive reduction from deciding f^* to learning PSPACE/poly using random examples over the uniform distribution, where $T_1, \dots, T_n, M$ are probabilistic polynomial time algorithms which are defined as follows.

For every $k \leq n$, T_k makes exactly one query to the learner which is the set of examples $S_k = \{(x_i, y_i)\}_{i \leq \mathsf{poly}(n)}$ drawn from the joint distribution $(U_k, f^*(U_k))$, where U_k is the uniform distribution over $\{0,1\}^k$. In the k^{th} round of adaptivity, T_k only makes oracle queries to the hypothesis h_{k-1} output in the last round. Indeed, let h'_{k-1} be the oracle circuit which uses h_{k-1} as an oracle in the self-corrector algorithm for f^*, and computes f^* on all $k-1$ length inputs with high probability. It then outputs a set S_k of independent labeled samples (x_i, y_i), where each x_i is sampled uniformly at random from U_k and $y_i = f^*(x_i)$

computed by using the downward self-reducibility of f^* with h'_{k-1}. M takes the final hypothesis h_n output by the learner over n inputs and outputs the value of the self-corrector of f^* with the oracle h_n. The correctness of R and the run-time analyses of $T_1, \ldots, T_n, M$ follow from the proof techniques of Lemma 3.2.

We next show that R has the required properties. As the self-corrector and the downward self-reduction for f^* work for any oracle which satisfy the appropriate constraints, R is correct for any oracle which outputs *any* correct hypothesis for f^* with respect to the uniform distribution (over different input lengths). Further, it makes only oracle queries to the learner, as well as to all the hypothesis circuits $h_1, \ldots, h_n$. This makes the reduction strongly black-box. By the property of the self-corrector, M only makes queries sampled from U_n to h_n, which is the same as the query made to the learner. The obliviousness now follows, since only f^* is learnt in each query, irrespective of the choice of z. ◀

The main result of the section is the following.

▶ **Theorem B.3.** *There exists a universal constant $c > 0$ such that the following holds. For any language L, $\varepsilon_0 = 1/n^c$ and any $B = \mathsf{poly}(n)$, if there exists an oblivious, B-adaptive, strongly black-box reduction from L to PAC-learning* NP/poly *using random examples over polynomially samplable distributions up to error ε_0, then $\overline{L} \in \mathsf{AM}^{\mathsf{poly}}$.*

Recall that the class $\mathsf{AM}^{\mathsf{poly}}$ refers to the class of languages recognized by constant-round interactive protocols with advice, where we require proper acceptance/rejection probabilities only when the advice is correct. [14] show that $\mathsf{AM}^{\mathsf{poly}} = \mathsf{NP/poly}$. Using Theorem B.3 with $L = \mathsf{SAT}$, we get

▶ **Corollary B.4.** *There exists a universal constant $c > 0$ such that the following holds. For $\varepsilon_0 = 1/n^c$ and any $B = \mathsf{poly}(n)$, if there exists an oblivious, B-adaptive, strongly black-box reduction from deciding* SAT *to learning* NP/poly *using random examples from polynomially samplable distributions up to error ε_0, then* $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.

Corollary B.4 easily implies Theorem 1.6, since $\mathsf{coNP} \subseteq \mathsf{NP/poly}$ implies that $\Sigma_3^P = \Pi_3^P$ [39].

▶ Remark B.5. In addition, we can also extend the proof to the case where M still makes non-adaptive queries but is not constrained distributionally in its access to all the hypotheses, by directly applying the techniques of [10] for the simulation of R in $\mathsf{AM}^{\mathsf{poly}}$.

The details of the proof of Theorem B.3 can be found in the full version (see Section 1.4 for a sketch).