Proceedings of Machine Learning Research vol 125:1–32, 2020 33rd Annual Conference on Learning Theory

Extending Learnability to Auxiliary-Input Cryptographic Primitives and Meta-PAC Learning

Mikito Nanashima NANASHIMA.M.AA@IS.C.TITECH.AC.JP
Department of Mathematical and Computing Science, Tokyo Institute of Technology

Editors: Jacob Abernethy and Shivani Agarwal

Abstract

We investigate the meaning of efficient learnability from several different perspectives. The purpose is to give new insights into central problems in computational learning theory (CoLT). Specifically, we discuss the following two questions related to efficient PAC learnability.

First, we investigate the gap between PAC learnability for polynomial-size circuits and weak cryptographic primitives taking auxiliary-input. Applebaum et al. (2008) observed that such a weak primitive is enough to show the hardness of PAC learning. However, the opposite direction is still unknown. In this paper, we introduce the following two notions: (1) a variant model of PAC learning whose hardness corresponds to auxiliary-input one-way functions; (2) a variant of a hitting set generator corresponding to the hardness of PAC learning. The equivalence gives a clearer insight into the gap between the hardness of learning and weak cryptographic primitives.

Second, we discuss why proving efficient learnability is difficult. This question is natural because few classes are known to be polynomially learnable at present. In this paper, we formulate a task of determining efficient learnability as a meta-PAC learning problem and show that our meta-PAC learning is exactly as hard as PAC learning. Our result insists on one possibility: a hard-to-learn instance itself yields the hardness of proving efficient learnability.

Our technical contribution is to give (1) a general framework for translating the hardness of PAC learning into auxiliary-input primitives, and (2) a new formulation to discuss the hardness of determining efficient learnability. Our work yields new important frontiers related to CoLT, including investigation of the learning hierarchy.

Note. There was an erratum on Definition 8 in the original proceedings paper, which was pointed out by Xu Duo. This version applies a modified definition that is consistent with the follow-up works (e.g. Nanashima, 2021).

Keywords: Computational learning theory, PAC learning, auxiliary-input primitives, meta-learning.

1. Introduction

Intuitively, learning simple concepts is easy, but learning complicated concepts seems to take much more time. Then which concepts can we learn efficiently from experience? This question is one of the central concerns in computational learning theory (CoLT). Since Valiant (1984) introduced the PAC (probabilistically approximately correct) learning model to capture the notion of computationally efficient learning, many researchers have sought answers to this question.

One desirable goal in CoLT is to find general knowledge of efficient learnability and provide reasonable concept classes which are easily learnable but expressive enough to capture many laws hidden in the real-world. To this end, several methods to construct efficient learners have been developed: for example, Occam's razor (Blumer et al., 1987), and boosting (Schapire, 1990; Freund, 1995). Another approach is to acquire knowledge of hard-to-learn concepts to know where

the difficulty of learning arises. Such intractability of efficient learners has been shown by applying assumptions and techniques in other fields: computational complexity (Pitt and Valiant, 1988), cryptography (Kearns and Valiant, 1994; Kharitonov, 1993), and as a relatively new line of work, the hardness of refuting CSPs (Daniely, 2016; Daniely and Shalev-Shwartz, 2016).

However, the above methods are not sufficient to understand the essence of efficient learning. Firstly, almost all of the relationships between CoLT and other fields have a gap. For a well-known example, the existence of one-way functions in cryptography implies the hardness of PAC learning (Valiant, 1984), but showing the opposite direction is still a big open problem in theoretical computer science. In other words, we do not know which part is essential to get the hardness of learning. Secondly, as mentioned by Servedio and Tan (2017), few concept classes are known to be efficiently PAC learnable by the known methods at present: only $O(1)$-degree polynomial threshold functions (Blumer et al., 1989) and $O(1)$-degree $\mathbb{F}_2$-polynomials (Helmbold et al., 1992). For further development in CoLT, we must conquer this problematic situation somehow.

In this paper, we look into the above current situation more closely. First, we investigate the gap between CoLT and cryptography. Applebaum et al. (2008) observed that weak cryptographic primitives taking auxiliary-input are enough to show the hardness of learning. However, this relationship also has an implicit gap. We give a clearer insight into this gap by giving the correspondence between the hardness of learning and auxiliary-input primitives. Second, we review research in CoLT from a meta-perspective and consider why proving efficient learnability is so hard. To this end, we give a new formulation of proving efficient learnability and analyze the computational hardness.

2. Our Results

To overview our results, we specify a learning model and learnability. In this paper, we discuss PAC learning with any efficiently computable hypothesis.

We define a concept class $\mathscr{C}$ as any subset of Boolean functions, and assume each element, called a target function, has a polynomial-size representation for the input size. Let $\mathscr{C}_n$ be the subset of $\mathscr{C}$ restricted to the input size n.

Definition 1 (Valiant 1984, PAC learning) *A concept class $\mathscr{C}$ is (polynomially) PAC learnable*[1] *with $m := m(n)$ samples and an advantage $\gamma := \gamma(n)$ where $m, \gamma^{-1} \leq \mathsf{poly}(n)$ if there exists a probabilistic polynomial-time algorithm L, called a PAC learner, satisfying that for any $n \in \mathbb{N}$, $f \in \mathscr{C}_n$, and target distribution D on $\{0,1\}^n$,*

$$\Pr[L((x_1, f(x_1)), \ldots, (x_m, f(x_m)), x^*) = f(x^*)] \geq 1/2 + \gamma.$$

where examples $x_1, \ldots, x_m \in \{0,1\}^n$ and a challenge $x^* \in \{0,1\}^n$ are selected according to the target distribution D.

We simply say that $\mathscr{C}$ is PAC learnable if there exist $m, \gamma^{-1} \leq \mathsf{poly}(n)$ such that $\mathscr{C}$ is PAC learnable with m samples and an advantage γ.

We also say that a concept class $\mathscr{C}$ is non-uniformly PAC learnable if there exists a family of polynomial-size (possibly randomized) circuits which PAC learns $\mathscr{C}$.

1. Strictly speaking, we adopt the weak prediction model introduced by Pitt and Warmuth (1990). The polynomial learnability in their model is equivalent to the polynomial learnability in the usual PAC learning model by Valiant (1984), which follows from the result by Haussler et al. (1988) and the boosting technique by Schapire (1990).

In this paper, we mainly consider polynomial-size circuits as a concept class. By a simple padding argument (formally, Lemma 20), we can assume w.l.o.g. that the size is restricted to n^2, that is, $\mathscr{C} = \mathsf{SIZE}[n^2] := \{C : \{0,1\}^n \to \{0,1\} |\ n \in \mathbb{N}, C \text{ is a circuit of size at most } n^2\}$.

2.1. Learning Model Corresponding to Auxiliary-Input One-Way Functions

To see the motivation of our work, let us mention further backgrounds. The first relationship between CoLT and cryptography was given by Valiant (1984). They showed that the existence of a one-way function (OWF) yields the hardness of learning polynomial-size circuits. An OWF is the fundamental primitive for modern cryptography (Impagliazzo and Luby, 1989), which is easily computed but hard to invert. Blum et al. (1994) made the first attempt to get the opposite direction, from the hardness of learning to OWFs. They proposed a variant of PAC learning whose hardness corresponds to OWFs. However, the gap between usual PAC learnability and OWFs remains open.

In the sequent work by Applebaum et al. (2008), they pointed out that a weaker cryptographic primitive, auxiliary-input one-way functions (AIOWFs), are indeed enough to show the hardness of PAC learning. An AIOWF, first introduced by Ostrovsky and Wigderson (1993), is weaker than an OWF in the following sense. An AIOWF is defined as a set of efficiently computable functions and regarded as secure if at least one of them is hard-to-invert for each adversary (instead of one specific hard-to-invert function for any adversary). The formal definition is the following.

Definition 2 (Auxiliary-input functions) *A (polynomial-time computable) auxiliary-input function (AIF) is a family $\mathcal{F} = \{f_z : \{0,1\}^{n(|z|)} \to \{0,1\}^{\ell(|z|)}\}_{z\in\{0,1\}^*}$, where $n(|z|)$ and $\ell(|z|)$ are polynomially-related to $|z|$, which satisfies that there exists a polynomial-time evaluation algorithm F such that for any $z \in \{0,1\}^*$ and $x \in \{0,1\}^{n(|z|)}$, $F(z,x)$ outputs $f_z(x)$.*

We may write $n(|z|)$ and $\ell(|z|)$ as n and ℓ, respectively, when the dependence of $|z|$ is obvious.

Definition 3 (Ostrovsky and Wigderson 1993, Auxiliary-input one-way functions) *An AIF $\{f_z : \{0,1\}^n \to \{0,1\}^\ell\}_{z\in\{0,1\}^*}$ is an auxiliary-input one-way function (AIOWF) if for any polynomial-time probabilistic algorithm A, there exists a non-empty set $Z_A \subseteq \{0,1\}^*$ such that for every $z \in Z_A$, f_z is hard to invert by A, that is,*

$$\Pr[A(z, f_z(U_n)) \in f_z^{-1}(f_z(U_n))] < \mathsf{negl}(|z|),$$

where $\mathsf{negl}(|z|)$ is a negligible function satisfying $\mathsf{negl}(|z|) < |z|^{-c}$ for any $c \in \mathbb{N}$ and sufficiently large $|z|$.

Since AIOWFs are weaker than OWFs, the gap to the hardness of learning is expected to become smaller. Therefore, it is a natural attempt to construct AIOWFs from the hardness of PAC learning. Unfortunately, the limitation of the common proof technique (i.e., an oracle separation) for this attempt has been found by Xiao (2009a). In the Ph.D. thesis by Xiao (2009b), another learning problem, CircLearn, was proposed, whose hardness implies the existence of AIOWFs. However, it is also open whether the hardness of CircLearn is equivalent to AIOWFs. Then what is the notion in CoLT corresponding to such weak cryptographic primitives?

Our first result is to find the learning model, PAC learning with a target sampling circuit (TSC), whose learnability is equivalent to the existence of AIOWFs. The requirement of our model is weaker than the usual PAC learning model in the following sense. First, a target function is selected

according to a certain efficiently samplable distribution, and the learner does not need to learn all target functions. Second, the learner is given the distribution on target functions as the polynomial-size circuit generating a target function. The formal definition is the following. We say that a concept class is $s(n)$-represented if each target function has a binary representation of length $s(n)$ for the input size n.

Definition 4 (PAC learning with a target sampling circuit) *An $s(n)$-represented concept class $\mathscr{C}$ is (weakly) PAC learnable with a target sampling circuit (TSC), m samples, and an advantage γ if there exists a probabilistic polynomial-time algorithm L satisfying that for any $n \in \mathbb{N}$, circuit $C : \{0,1\}^r \to \{0,1\}^{s(n)}$ of size $s(n)^2$, and target distribution D on $\{0,1\}^n$,*

$$\Pr[L(C, (x_1, f(x_1)), \ldots, (x_m, f(x_m)), x^*) = f(x^*)] \geq 1/2 + \gamma,$$

where $x_1, \ldots, x_m, x^*$ are selected according to D and the target function is $f := C(s)$, where s is a random seed uniformly selected from $\{0,1\}^r$.

We say that $\mathscr{C}$ is (weakly) PAC learnable with a TSC if there exist $m, \gamma^{-1} \leq \mathsf{poly}(n)$ such that $\mathscr{C}$ is PAC learnable with a TSC, m samples, and an advantage γ.

By a simple padding argument, we can replace the upper bound $s(n)^2$ of the size of a TSC with $s(n)^c$ for any $c > 1$ without changing the learnability (formally, Lemma 19).

Now we state the first main theorem.

Theorem 5 (AIOWF $\Leftrightarrow$ Hardness of PAC learning with a TSC) *There exist AIOWFs iff the class of polynomial-size circuits (specifically, $\mathsf{SIZE}[n^2]$) is not weakly PAC learnable with a TSC.*

Note that we can restrict the target class to $\mathsf{SIZE}[n^2]$ without changing the learnability, which is also shown by a simple padding argument (formally, Lemma 19).

As a corollary, we can observe the following phenomenon: for PAC learning polynomial-size circuits with a TSC, query access to the target function (that is, membership query) does not help the learner at all[2]. For usual PAC learning, such a result was shown by Angluin and Kharitonov (1995) under the assumption that an OWF exists. Interestingly, our result holds unconditionally.

Corollary 6 *The class of polynomial-size circuits is not weakly PAC learnable with a TSC iff the same class is not weakly PAC learnable with a TSC and membership queries.*

2.2. Auxiliary-Input Primitive Corresponding to Hardness of PAC Learning

Now one crucial question arises. What is the equivalent primitive to the hardness of usual PAC learning? Indeed, our proof technique for Theorem 5 gives a reversible method to translate the hardness of learning into auxiliary-input primitives (we will see it in Section 4). By applying this method to the usual PAC learning model, we have some auxiliary-input primitives corresponding to the hardness of PAC learning, which is a variant of a hitting-set generator.

A hitting set generator (HSG), first introduced by Andreev et al. (1998), is a weaker notion of a pseudorandom generator (PRG) in cryptography. Note that the existence of PRGs is equivalent to the existence of OWFs (Håstad et al., 1999).

2. In the membership query model, the learner is given examples and a challenge one by one, and the membership query oracle is also available only before the learner gets the challenge.

Definition 7 (Andreev et al. 1998, Hitting set generators) *An algorithm $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ is a hitting set generator (HSG) (for* BPP *with largeness $1/2$) if the generator G hits all statistical test in* BPP*, that is, for any polynomial-time probabilistic algorithm A, there exist infinitely many n's such that A satisfies either*

$$\exists x \in \{0,1\}^n \text{ s.t. } \Pr_A[A(G(x)) = 0] < 2/3 \text{ or } \Pr_{y\sim\{0,1\}^{\ell(n)}}\left[\Pr_A[A(y) = 1] \geq 2/3\right] \leq 1/2.$$

We call the above function $\ell(n)$ the stretch of G.

The original aim of HSGs is to develop a general derandomization method (e.g., Goldreich et al., 2011). To characterize the PAC learnability, we need several modifications of HSGs. In the following, we first weaken the condition of HSGs. At this point, the existence of the weakened HSG becomes weaker than the hardness of PAC learning. As far as we know, this is the first generator weaker than the hardness of PAC learning. Secondly, we add two conditions to the weakened HSG. Then the existence of such HSGs will exactly coincide with the hardness of PAC learnability.

WEAKENING STEP

For the derandomization, usually, an exponential-stretch HSG is necessary (i.e., it stretches the input of length n to a pseudorandom string of length $2^{\epsilon n}$ for some $\epsilon > 0$). For polynomial learnability, however, polynomial-stretch is sufficient as cryptographic primitives. In addition, we allow auxiliary-input as in the previous section. These relaxations lead to the following definition.

Definition 8 (Auxiliary-input hitting set generators) *An AIF $\mathcal{G} = \{G_z : \{0,1\}^n \to \{0,1\}^{\ell(n)}\}_{z\in\{0,1\}^*}$ is an auxiliary-input hitting set generator (AIHSG) if $\ell(n) > n$ and for any polynomial-time probabilistic algorithms A, there exists a non-empty set $Z_A \subseteq \{0,1\}^*$ such that for every $z \in Z_A$, G_z hits the language recognized by of A, that is, A satisfies either*

$$\exists x \in \{0,1\}^{|z|} \text{ s.t. } \Pr_A[A(z, G_z(x)) = 0] < 2/3 \text{ or } \Pr_{y\sim\{0,1\}^{\ell(n(|z|))}}\left[\Pr_A[A(z, y) = 1] \geq 2/3\right] \leq 1/2.$$

We call the above function $\ell(\cdot)$ in $n(|z|)$ the stretch of $\mathcal{G}$.

Indeed, it is not so difficult to see that if we regard auxiliary-input of AIHSG as input, then the resulting generator becomes a secure HSG. However, the stretch (i.e., how long the hidden input is stretched) decreases drastically because the input length is stretched from $n(|z|)$ to $|z| + n(|z|)$.

STRENGTHENING STEP

First, we need hitting set generators with arbitrary polynomial stretch. In the case of PRGs, we can easily construct any polynomial stretch PRG from any PRG even if the original stretch is $n + 1$. In the case of AIHSGs, however, such construction is quite non-trivial. Second, we need the following condition on locality: each bit of the output is locally computable from the input and a small part of auxiliary-input, and the way to compute does not depend on the values of input and auxiliary-input. Formally, our auxiliary-input primitive is defined as follows.

Definition 9 (Auxiliary-input local HSGs) *A family of AIF $\mathcal{G} = \{\mathcal{G}^\tau\}_{\tau\in\mathbb{N}}$ is an auxiliary-input local HSG (AILHSG) if $\mathcal{G}$ satisfies the following two conditions:*

- *(arbitrary polynomial stretch) for any τ, $\mathcal{G}^\tau = \{G^\tau_z\}_{z\in\{0,1\}^*}$ is an AIHSG with the stretch $n(|z|)^\tau$;*
- *(locally computable) there exists a non-adaptive polynomial-time oracle machine $localG^?$ such that $localG^z(\tau, |z|, i, x)$ returns the i-th bit of $G^\tau_z(x)$ and the positions of queries determined by $\tau, |z|, i$ (not depending on input x and auxiliary-input z).*

Then the above AILHSG corresponds to the hardness of the standard PAC learning.

Theorem 10 (Hardness of PAC learning $\Leftrightarrow$ AILHSG) *The class of polynomial-size circuits (specifically, $\mathsf{SIZE}[n^2]$) is not PAC learnable iff there exists an AILHSG.*

Although our assumption on locality looks somewhat unusual, the above generator exactly corresponds to usual PAC learnability, and this unusuality seems to essentially bring the gap between learning and cryptographic primitives. Nevertheless, a natural question is what a candidate for the AILHSG is. In this paper, we also give the explicit generator which is complete in the sense that it must be an AILHSG if there exists some (possibly different) AILHSG.

Theorem 11 (Complete AILHSG) *There exists an explicit generator $\{UC^\tau\}_{\tau\in\mathbb{N}}$ satisfying that $\{UC^\tau\}_{\tau\in\mathbb{N}}$ is an AILHSG iff there exists an AILHSG.*

2.3. Determining Polynomial PAC Learnability: Meta-PAC Learning

Now we move on to the second topic of this paper: why is proving efficient learnability so difficult? To give one possible answer to this question, we make a hypothesis and argue the validity.

Our observation is the following: to prove learnability for a certain concept class, we must determine at least whether the concept class is polynomially learnable or not. We hypothesize that such a task of determining learnable classes and not-learnable classes is indeed computationally difficult. In the following, we focus on how to check the validity of this hypothesis.

Our idea is to formulate determining the polynomial learnability of a given concept class as a computational problem and analyze the computational complexity. However, we cannot get a straightforward formulation for such a computational task due to the following reasons. First, a concept class does not have a reasonable binary representation in general. Second, more crucially, polynomial learnability concerns with the asymptotic behavior when sufficiently abundant polynomial-size resources (i.e., samples and running-time) are available. Thus, we need to reflect the notion of "sufficiently abundant but polynomial-size resources" in our formulation.

To resolve the first problem, we focus on an evaluation function for a concept class. Usually, a concept class which we are interested in has an efficiently computable evaluation rule: for example, from input x and a simple disjunctive normal form formula ϕ, we can easily evaluate the value of $\phi(x)$. Therefore, we regard a polynomial-size evaluation circuit for a concept class as the definition of the concept class. Since Boolean circuits have binary encoding, this formulation enables us to encode concept classes and define a language composed of concept classes.

We define a language $\mathsf{Learnable}$ representing efficiently learnable classes as Definition 12. Informally, the third condition in the definition captures the notion of "PAC learnable", where we adopt another formulation of PAC learnability given by Vadhan (2017). Since a usual PAC learner in Definition 1 must work for all (uncountably infinite) target distributions, it is not trivial to verify

the learnability by polynomial-size witness, particularly, whether the given learner satisfies the requirement of learning. On the other hand, our formulation can directly connect the complexity of determining learnability with the polynomial hierarchy (PH).

Definition 12 ($\mathsf{Learnable}_{\tau,m}$) *For any polynomial $\tau := \tau(n)$ and $m := m(n)$, we define a language $\mathsf{Learnable}_{\tau,m}$ as a collection of circuits C satisfying that:*

- *C computes an evaluation function $E : \{0,1\}^{s(n)} \times \{0,1\}^n \to \{0,1\}$ for some $n \in \mathbb{N}$,*
- *the size of C is at most $\tau(n)$,*
- *there exists a circuit[3] $L \in \mathsf{SIZE}[n^2]$ which learns the concept class determined by C with m samples in the following sense: for any $x_1, \ldots, x_m \in \{0,1\}^n$, L satisfies that*
  - *(soundness) for any $u \in \{0,1\}^{s(n)}$, $L(x_1, \ldots, x_m, E(u, x_1), \ldots, E(u, x_m)) = 0$;*
  - *(completeness) $\Pr_{b_1,\ldots,b_m \leftarrow \{0,1\}}[L(x_1, \ldots, x_m, b_1, \ldots, b_m) = 1] \geq 1/2$.*

Then the language $\mathsf{Learnable}_{\tau,m}$ characterize PAC learnability in the following sense.

Theorem 13 *Let $\mathscr{C}$ be a concept class whose evaluation function is computed by a circuit $C_n : \{0,1\}^{s(n)} \times \{0,1\}^n \to \{0,1\}$ of size $\tau(n)$, that is, $\mathscr{C}_n = \{f(x) := C_n(u, x) : u \in \{0,1\}^{s(n)}\}$. The class $\mathscr{C}$ is non-uniformly PAC learnable iff for sufficiently large c, $C_n \in \mathsf{Learnable}_{\tau,n^c}$ for each n.*

In addition, $\mathsf{Learnable}_{\tau,m}$ is contained in the polynomial hierarchy. This is immediately derived from the definition of $\mathsf{Learnable}_{\tau,m}$ and the fact that $\mathsf{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$ by Lautemann (1983). The formal proofs of Theorems 13 and 14 will be given in Appendix C.

Theorem 14 *For any polynomials τ and m, $\mathsf{Learnable}_{\tau,m} \in \Sigma_3^p$. Moreover, if $\mathsf{P} = \mathsf{BPP}$, then indeed $\mathsf{Learnable}_{\tau,m} \in \Sigma_2^p$.*

By Theorem 13, if we can efficiently recognize $\mathsf{Learnable}_{\tau,n^c}$ for sufficiently large c, then we can also efficiently determine the learnability of concept classes evaluated by $\tau(n)$-size circuits. In the opposite direction, if we can efficiently determine the learnability of such concept classes, then we can also efficiently recognize $\mathsf{Learnable}_{\tau,n^c}$ for sufficiently large c because $\mathsf{Learnable}_{\tau,n^c}$ corresponds to the set of learnable classes for sufficiently large c. Therefore, determining the learnability of concept classes evaluated by $\tau(n)$-size circuits corresponds to recognizing $\mathsf{Learnable}_{\tau,n^c}$ for sufficiently large c. This observation is central to resolve the second problem for the formulation. In addition, for sufficiently large c, the polynomial $\tau(n)$ does not affect the asymptotic complexity (formally, Theorem 29). Therefore, w.l.o.g., we can fix τ as $\tau(n) = n^2$. These arguments lead to the following formulation of the hardness of determining polynomial PAC learnability. We name the task meta-PAC learning.

Definition 15 (Meta-PAC learning is hard) *We say that meta-PAC learning is hard if there exists infinitely many $c \in \mathbb{N}$ satisfying that $\mathsf{Learnable}_{n^2,n^c} \notin \mathsf{P}$.*

3. In fact, any exponent factor of the bound on the size of L will not change the asymptotic complexity of $\mathsf{Learnable}$ (formally, the proof of Theorem 13 in Appendix C.1).

The third result is the equivalence between the hardness of meta-PAC learning and the hardness of non-uniform PAC learning for polynomial-sized circuits. This result insists that a hard-to-learn class itself yields the hardness of determining which concept classes are indeed hard-to-learn.

Theorem 16 (Meta-PAC learning is as hard as learning) *Meta-PAC learning is hard iff polynomial-size circuits are not non-uniformly PAC learnable.*

2.4. Related work

We list the related work and briefly mention the differences to our work.

PAC LEARNING WITH A TARGET SAMPLING CIRCUIT

As described in Section 2.1, Xiao (2009b) introduced the learning problem called CircLearn, which is similar to our learning model. In CircLearn, a learner is given a circuit generating a sample (i.e., a pair of input and the label) instead of a target function. Thus, the learner is given more information, including the way to generate examples. This additional information makes the hardness assumption stronger than the hardness of learning with a TSC, equivalently, the existence of AIOWFs.

Our model is also regarded as the restricted case of the learning model introduced by Blum et al. (1994), where the learner can be constructed for fixed distributions on target functions and examples. Since their hardness assumption corresponds to the existence of OWFs, the gap between their model and our model with a TSC represents the gap between OWFs and AIOWFs.

META-PAC LEARNING

To the best of our knowledge, our meta-PAC learning problem is the first formulation of determining polynomial PAC learnability. However, there are several related work which has the same purpose of determining some notions related to learnability: for example, deciding VC dimension (Schaefer, 1999), and the several consistency problems proposed by Ko and Tzeng (1991). However, their formulations do not capture the notion of polynomially PAC learning. Indeed, the complexity of their problems is exactly characterized by the polynomial hierarchy (that is, deciding VC dimension is Σ_3^p-complete and the consistency problems are Σ_2^p-complete). These facts show the differences from our meta-PAC learning because PAC learnability is not characterized by such classes at present.

A similar problem in computational complexity is the minimum circuit-size problem (MCSP, e.g., Kabanets and Cai, 2000). MCSP informally asks whether the given language, equivalently a truth-table of a boolean function, is recognized by small size circuits. There are several differences from our meta-PAC learning, which arise from the difference between settings of computing and learning; for example, the term 'efficient' represents exponentially different running-time because a truth-table of Boolean function has exponential-size representation in general.

3. Discussion and Future Direction

We introduced three notions corresponding to the efficient learnability and extended the meaning of the learnability into auxiliary-input primitives and the complexity of meta-PAC learning. Before giving the proofs, we present further benefits of our results.

3.1. PAC Learning and Auxiliary-Input Primitives

Theorems 5 and 10 give the first equivalence relationship between learning theory and auxiliary-input primitives (that is, weak cryptography), which bring new clearer insights into the implicit gap between the notions in each field. For example, by Theorem 5, we can see that the difference between the hardness of PAC learning and the existence of AIOWFs comes from the average-case hardness of a target function and a TSC in learning theory. Note that, only by previous knowledge including some gap, we could not mention such a difference explicitly.

Another benefit is to make sophisticated techniques in cryptography available even in learning theory, and vice versa. For example, Corollary 6 is essentially proven by the construction of pseudorandom functions from PRGs in cryptography (Goldreich et al., 1986). At present, proving such a result seems to be difficult by only techniques in learning theory. In addition, let us stress that the opposite direction is also quite useful, that is, learning theory to cryptography. There are several intermediate learning models between our PAC learning model with a TSC and the learning model by Blum et al. (1994). By considering the hardness of each model, we can divide the gap between learning and cryptography in the current theory into several stages.

This idea leads to the notion "Learning Hierarchy (LH)," which divides the gap between $\mathsf{NP} \not\subseteq \mathsf{BPP}$ in complexity theory and the existence of OWFs in cryptography in fine-grained manner. This notion provides a method to analyze such a gap (that is, Pessiland, named by Impagliazzo, 1995) more closely from the perspective of learning theory. The details of LH will be given in Appendix D. Indeed, our results play the crucial roles in LH: Theorem 10 connects complexity theory with learning theory smoothly; Theorem 5 shows that LH indeed includes the previous notion (i.e., AIOWFs) of weak cryptography. As the first future direction, we require further investigation of LH.

3.2. PAC Learning and Meta-PAC Learning

The main contribution of Theorem 16 is to show the possibility that a hard-to-learn class itself can be a curse of the hardness of proving efficient learnability. Although our proof is quite simple and follows from only fundamental knowledge in complexity theory (as seen in Section 4 and Appendix C), to the best of our knowledge, no one has not insisted on this possibility. Therefore, our result casts new critical doubt on CoLT: the notion of hard-to-learn can be much harder to handle than we expected. Fortunately, our hardness result holds only in the worst-case and the PAC learning model, and our formulation of meta-PAC learning is much severer than the usually desired task of proving efficient learnability. Therefore, as the second future direction, we require further analysis of the complexity of determining polynomial learnability in other settings.

4. Proof Techniques and Sketches

We give proof techniques and sketches of Theorems 5, 10, and 16. For complete arguments and proofs, see Appendices A, B, and C, respectively.

First, we give a general framework to translate the hardness of learning into auxiliary-input primitives, which consists of the following two steps. Theorems 5 and 10 are shown by applying this framework.

STEP 1: FROM THE HARDNESS OF PAC LEARNING TO RRHS-REFUTING THE DUAL-CLASS

The first step was originally introduced by Vadhan (2017). They showed that the hardness of PAC learning $\mathscr{C}$ is equivalent to the hardness of random right-hand-side (RRHS)-refuting the dual-class $\mathscr{C}^*$.

RRHS-refuting a dual-class is a task of distinguishing satisfiable instances from RRHS-instances. An algorithm, called a refuter, is given a sample set $(x_1, b_1), \ldots, (x_m, b_m)$, and (1) if there exists a function $f \in \mathscr{C}$ satisfying the sample, that is, $f(x_i) = b_i$ for any i, then the refuter must output 0 (soundness), and (2) if the labels $b_1, \ldots, b_m$ (i.e., right-hand-side) are selected at uniformly random, then the refuter must output 1 with high probability (completeness). For the formal definition, see Definition 21. Indeed, the equivalence between learning and refutation is shown by standard techniques including Yao's next bit generator (Yao, 1982) and holds even in our setting where a TSC is available. In other words, the hardness of PAC learning with a TSC is equivalent to the hardness of RRHS-refuting the dual-class with a TSC.

The above transformation is not enough to get secure auxiliary-input primitives. For contraposition, it is enough to construct an adversary for auxiliary-input primitives from the refuter. In general, we can regard auxiliary-input as a description of primitives, which corresponds to a target function in learning. When an adversary tries to break auxiliary-input primitives, a description of a target function (i.e., auxiliary-input) is known, but the input is hidden. On the other hand, when an algorithm tries to RRHS-refute the dual-class, a description of a target function is hidden, and the input is known as an example. In other words, the roles of input and target functions are exchanged.

STEP 2: TRANSFORMATION VIA UNIVERSAL CIRCUITS

The key tool for the second step is universal circuits. A universal circuit UC is a polynomial-size circuit evaluating the value of a polynomial-size circuit, that is, $UC(C, x) = C(x)$ for any polynomial-size circuit C and input x (formally, Fact 1 in Appendix A). Since the circuit evaluation problem is contained in P, we can generate universal circuits by a polynomial-time Turing machine.

Now we translate the hardness of RRHS-refutation into auxiliary-input primitives. First, we construct a base circuit which generates the labels for each learning setting. For example, in the case of usual RRHS-refutation, the base circuit is $C(x_1) \circ \cdots \circ C(x_m)$ where C is a polynomial-size circuit computing a target function and x_i's are examples. Second, we replace all calculations in the base circuit with universal circuits such as $UC(C, x_1) \circ \cdots \circ UC(C, x_m)$. At this stage, the description of circuits can be regarded as input. Thus, there is no longer a distinction between circuits and input. Finally, we divide all input into the known part and the hidden part for the learner and construct a generator by regarding the known (resp. hidden) part as auxiliary-input (resp. input). In the case of usual RRHS-refutation, the resulting generator is $G_{x_1 \circ \cdots \circ x_m}(C) = UC(C, x_1) \circ \cdots \circ UC(C, x_m)$. Note that the roles of input and target functions were switched in the above process.

As a simple observation of RRHS-refutation, labels in a satisfiable instance correspond to the output of the above generator. On the other hand, labels in an RRHS-instance correspond to a random string. Therefore, the task of distinguishing these two instances corresponds to the task of distinguishing the output of the above generator from truly random strings. Thus, the hardness of RRHS-refutation yields a kind of pseudorandomness using auxiliary-input.

4.1. AIOWFs and PAC Learning with a TSC: Proof Sketch of Theorem 5

FROM THE HARDNESS OF PAC LEARNING WITH A TSC TO AIOWFS

First, we show the hardness of RRHS-refuting polynomial-size circuits with a TSC by Yao's next bit generator, as in the previous work by Vadhan (2017). In this model, remember that a target function is chosen by the given TSC.

Second, we construct the auxiliary-input primitive. The resulting generator G^{TS} has two stages composed of universal circuits. At the first stage, G^{TS} computes the description of a target function from the description of a given TSC and the random seed, then transmits the description of the target function to the second stage. At the second stage, G^{TS} evaluates the labels from the target function and given examples.

We regard the known information (that is, a description of a TSC and examples) as auxiliary-input and the hidden information (that is, a random seed for sampling a target function) as input to G^{TS}. In a satisfiable instance, notice that the input is selected uniformly at random. Therefore, RRHS-refutation with a TSC corresponds to the task of distinguishing the following two cases: (1) a string generated by G^{TS} where the input is selected at random; (2) a truly random string. Therefore, the hardness of RRHS-refutation implies that G^{TS} is an auxiliary-input version of a pseudorandom generator, whose existence is equivalent to the existence of an AIOWF (Goldreich et al., 1986).

Indeed, some technical flaws remain: for example, we must fix the length of input to G^{TS} in the above beforehand. Since the input to G^{TS} corresponds to a seed for selecting a target function, this fixing yields the restriction on the seed size of a TSC. This flaw will be resolved by a padding argument. For the complete proof, see Claim 2 in Appendix A.3, where the theorem will be proved by contraposition and the argument will proceed along the reverse direction: (1) breaking G^{TS}; (2) RRHS-refuting with a TSC; (3) PAC learning with a TSC.

FROM AIOWFS TO THE HARDNESS OF PAC LEARNING WITH A TSC

The proof mainly follows the proof of the hardness of the usual PAC learning based on AIOWFs by Applebaum et al. (2008) (originally proposed by Valiant, 1984). We use another cryptographic primitive, auxiliary-input pseudorandom functions (AIPRFs), which exist iff AIOWFs exist. As mentioned in the previous work, the hardness of PAC learning is implied immediately by the definition of AIPRFs. The difference in our case is only the part of generating a circuit for sampling pseudorandom functions. However, constructing a circuit sampling an AIPRF is not difficult because the index of an AIPRF is selected uniformly at random.

Note that the above argument holds even if we allow the learner to access to membership query. This fact derives Corollary 6 immediately. For the details, see Claim 1 in Appendix A.3.

4.2. PAC Learning and AILHSGs: Proof Sketch of Theorem 10

FROM THE HARDNESS OF PAC LEARNING TO AILHSGS

First, we translate the hardness of learning into the hardness of RRHS-refuting (Vadhan, 2017). Second, we construct the auxiliary-input primitive composed of universal circuits. As seen in the Step 2 in Section 4, the auxiliary-input is examples and the input is a description of a target function. Note that the stretch of the generator corresponds to the number of samples for RRHS-refutation.

We can show that the above generator is indeed an AILHSG as follows. It is not difficult to see that completeness and soundness of a refuter correspond to conditions for an adversary breaking a

HSG. Therefore, the hardness of RRHS-refutation yields the requirement of HSGs in Definition 8. The condition on arbitrary polynomial stretch in Definition 9 is derived from the assumption that any polynomial-size samples (which correspond to the stretch) are not enough for RRHS-refutation (equivalently, PAC learning) by polynomial-time algorithms. The condition on locality also holds because each bit of the generator can be computed only from input (that is, a target function) and a part of auxiliary-input (that is, one example). For the details, see the proof (1⇒2) of Theorem 24 in Appendix B.3.

FROM AILHSGS TO THE HARDNESS OF PAC LEARNING

Note that the above framework to construct an auxiliary-input primitive is reversible. Thus, if we have the corresponding auxiliary-input primitive, then we can also show the hardness of learning.

Assume that there exists an AILHSG $\mathcal{G}$ with the oracle machine $localG^?$. Remember that for any polynomial stretch n^τ, $localG$ can compute each i-th bit of the output from input x and partial information $z^{(i)}$ of auxiliary-input z. Since $localG$ is a polynomial-time oracle machine, $|z^{(i)}| \leq \mathsf{poly}(|x|)$. Now we consider the generator $UC^\tau_{z'_1 \circ \cdots \circ z'_{n^\tau}}(x) = UC(x, z'_1) \circ \cdots \circ UC(x, z'_{n^\tau})$, which was given from the hardness of PAC learning. Informally, if each auxiliary-input z'_i exactly contains the information of $z^{(i)}$ and the description of $localG$, then UC^τ corresponds to the AILHSG $\mathcal{G}$. Therefore, UC^τ contains any AILHSG as a special case. By applying the above framework in reverse order, we have the hardness of PAC learning.

The formal proof will be given in Theorem 24 (2⇒3 and 3⇒1) in Appendix B.3. As the first step of the proof (that is, 2⇒3), we show that the above generator $\{UC^\tau\}_{\tau \in \mathbb{N}}$ is an AILHSG by assuming that the existence of a (general) AILHSG. This argument also gives candidates for the complete AILHSG in Theorem 11. However, the above argument is not enough for complete proof because the generator depends on the size of another AILHSG in the assumption. In other words, we cannot fix the complete generator beforehand. To resolve this problem, we again apply the padding argument for PAC learning (formally, see Appendix B.3).

4.3. Meta-PAC Learning: Proof Sketch of Theorem 16

In this section, we consider the non-uniform model as a standard computational model. To prove the equivalence between PAC learning and meta-PAC learning, we do not need the above framework anymore. Instead, we have a quite simple and intuitive proof. For simplicity, we omit the argument about the upper bound $\tau(n)$ on the size of evaluation circuits. For the formal proof, see Appendix C.3, where we indeed show slightly stronger results than Theorem 16.

FROM PAC LEARNING TO META-PAC LEARNING

This direction is trivial from Theorem 13. Assume that all polynomial-size circuits are PAC learnable. Notice that if a concept class is evaluated by a $\tau(n)$-size circuit, then the class consists only of $\tau(n)$-size circuits. Therefore, for sufficiently large c, $\mathsf{Learnable}_{\tau(n),n^c}$ exactly corresponds to the set of $\tau(n)$-size circuits. We can easily verify whether the size of a given circuit is at most $\tau(n)$ by a polynomial-time Turing machine.

FROM META-PAC LEARNING TO PAC LEARNING

We show that if PAC learning polynomial-size circuits is hard, then meta-PAC learning is also hard. For contradiction, assume that PAC learning polynomial-size circuits is hard, but meta-PAC learning is not hard.

By the assumption, there exist a polynomial-time Turing machine (a meta-PAC learner) determining PAC learnability and a circuit C evaluating a hard-to-learn concept class. Then we can construct a polynomial-time Turing machine A for solving the circuit-SAT problem. For any given circuit C', the algorithm A constructs a new concept class $\mathscr{C}$ evaluated by $C''(u_1 \circ u_2, x) := C(u_1, x) \wedge C'(u_2)$ and feeds C'' to the meta-PAC learner. Note that the input to C' is regarded as a part of the representation of target functions.

If the given C' is unsatisfiable, then the concept class $\mathscr{C}$ consists only of the constant function 0, which is trivially PAC learnable. On the other hand, if the given C' is satisfiable, then $\mathscr{C}$ contains the original concept class evaluated by C. Therefore, $\mathscr{C}$ must be hard-to-learn, and the meta-PAC learner can distinguish these two cases in polynomial-time. Thus, A can solve the circuit-SAT problem efficiently, which yields $\mathsf{P} = \mathsf{NP}$.

As a well-known fact, if $\mathsf{P} = \mathsf{NP}$, then we can construct an efficient PAC learning algorithm for polynomial-size circuits (Blumer et al., 1987). This contradicts the hardness of PAC learning.

Acknowledgments

This work was supported by JST, ACT-X Grant Number JPMJAX190M, Japan. We thank Toshiya Itoh and the anonymous reviewers for many helpful comments. We thank Xu Duo for pointing out a bug in the previous version of this paper.

References

L. Adleman. Two Theorems on Random Polynomial Time. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, SFCS '78, pages 75–83, Washington, DC, USA, 1978. IEEE Computer Society.

A. E. Andreev, A. E. F. Clementi, and J. D. P. Rolim. A New General Derandomization Method. *J. ACM*, 45(1):179–213, January 1998.

D. Angluin and M. Kharitonov. When Won't Membership Queries Help? *Journal of Computer and System Sciences*, 50(2):336–355, 1995.

B. Applebaum, B. Barak, and D. Xiao. On Basing Lower-Bounds for Learning on Worst-Case Assumptions. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '08, pages 211–220, Washington, DC, USA, 2008. IEEE Computer Society.

S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

A. Blum, M. Furst, M. Kearns, and R. J. Lipton. Cryptographic Primitives Based on Hard Learning Problems. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 278–291, 1994.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's Razor. *Inf. Process. Lett.*, 24(6):377–380, apr 1987.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension. *J. ACM*, 36(4):929–965, October 1989.

A. Daniely. Complexity Theoretic Limitations on Learning Halfspaces. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 105–117, New York, NY, USA, 2016. ACM.

A. Daniely and S. Shalev-Shwartz. Complexity Theoretic limitations on Learning DNF's. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 815–830, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

Y. Freund. Boosting a Weak Learning Algorithm by Majority. *Information and Computation*, 121 (2):256–285, 1995.

O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *J. ACM*, 33(4): 792–807, August 1986.

O. Goldreich, S. Vadhan, and A. Wigderson. *Simplified Derandomization of BPP Using a Hitting Set Generator*, pages 59–67. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A Pseudorandom Generator from Any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, March 1999.

D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of Models for Polynomial Learnability. In *Proceedings of the First Annual Workshop on Computational Learning Theory*, pages 42–55, 1988.

D. Helmbold, R. Sloan, and M. K. Warmuth. Learning Integer Lattices. *SIAM Journal on Computing*, 21(2):240–266, 1992.

R. Impagliazzo. A Personal View of Average-case Complexity. In *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*, pages 134–147, June 1995.

R. Impagliazzo and M. Luby. One-way Functions Are Essential for Complexity Based Cryptography. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 230–235, 1989.

V. Kabanets and J. Cai. Circuit Minimization Problem. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC 00, pages 73–79. Association for Computing Machinery, 2000.

M. Kearns and L. Valiant. Cryptographic Limitations on Learning Boolean Formulae and Finite Automata. *J. ACM*, 41(1):67–95, January 1994.

M. Kharitonov. Cryptographic Hardness of Distribution-specific Learning. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 372–381, New York, NY, USA, 1993. ACM.

K. Ko and W. Tzeng. Three Σ_2^p-complete Problems in Computational Learning Theory. *computational complexity*, 1:269–310, 1991.

C. Lautemann. BPP and the Polynomial Hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.

Mikito Nanashima. On basing auxiliary-input cryptography on np-hardness via nonadaptive black-box reductions. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 29:1–29:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi: 10.4230/LIPICS.ITCS.2021.29. URL https://doi.org/10.4230/LIPIcs.ITCS.2021.29.

R. Ostrovsky and A. Wigderson. One-Way Functions are Essential for Non-Trivial Zero-Knowledge. In *The 2nd Israel Symposium on Theory and Computing Systems*, pages 3–17, June 1993.

L. Pitt and L. G. Valiant. Computational Limitations on Learning from Examples. *J. ACM*, 35(4): 965–984, October 1988.

L. Pitt and M. K. Warmuth. Prediction-Preserving Reducibility. *J. Comput. Syst. Sci.*, 41(3):430–467, 1990.

M. Schaefer. Deciding the Vapnikervonenkis Dimension is Σ_3^p-complete. *Journal of Computer and System Sciences*, 58(1):177–182, 1999.

R. E. Schapire. The Strength of Weak Learnability. *Mach. Learn.*, 5(2):197–227, 1990.

R. A. Servedio and L. Tan. What Circuit Classes Can Be Learned with Non-Trivial Savings? In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 30:1–30:21, 2017.

S. Vadhan. On Learning vs. Refutation. In *Proceedings of the 2017 Conference on Learning Theory (COLT'17)*, volume 65 of *Proceedings of Machine Learning Research*, pages 1835–1848, Amsterdam, Netherlands, 07–10 Jul 2017. PMLR.

L. G. Valiant. A Theory of the Learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

D. Xiao. On basing ZK $\neq$ BPP on the hardness of PAC learning. In *24th Annual IEEE Conference on Computational Complexity*, CCC'09, pages 304–315, 2009a.

D. Xiao. *New Perspectives on the Complexity of Computational Learning, and Other Problems in Theoretical Computer Science*. PhD thesis, Princeton University, 2009b.

A. C. Yao. Theory and Application of Trapdoor Functions. In *23rd Annual Symposium on Foundations of Computer Science (FOCS 1982)*, pages 80–91, Nov 1982.

Appendix A. AIOWFs and PAC Learnability with a TSC

A.1. Preliminaries and Formal Definitions

For a distribution D, we write $x \leftarrow D$ for a sampling x according to D. For a finite set S, we also abuse the notation $x \leftarrow S$ to denote the uniform sampling from S. For any $n \in \mathbb{N}$, let U_n be a uniform distribution over $\{0,1\}^n$. For any $n \in \mathbb{N}$, we define a set $[n] \subseteq \mathbb{N}$ by $[n] = \{1, \dots, n\}$.

In this paper, we define the size of circuits as the total number of all gates, that is, input-gates, output-gates, $\{\vee, \wedge, \neg\}$-gates. Note that this definition does not change the learnability for polynomial-size circuits defined by not taking input-gates and output-gates into account as the size. In addition, we can assure that any circuits of size S has a binary representation of length $e(S)$.

For any function $s : \mathbb{N} \to \mathbb{N}$, let $\mathsf{SIZE}[s(n)]$ be a complexity class of $s(n)$-size circuits, that is,

$$\mathsf{SIZE}[s(n)] := \{C : \{0,1\}^n \to \{0,1\} \mid n \in \mathbb{N},\ C \text{ is a circuit of size is at most } s(n)\}.$$

For any circuit $C : \{0,1\}^n \to \{0,1\}^\ell$ and $i \in [\ell]$, let $C_i : \{0,1\}^n \to \{0,1\}$ be the partial circuit computing i-th bit of C, that is, $C(x) = C_1(x) \circ \cdots \circ C_\ell(x)$ for each $x \in \{0,1\}^n$.

A.1.1. UNIVERSAL CIRCUIT AND CIRCUIT TRANSLATOR

We fix a proper encoding for Boolean circuits. For any circuit C, we use $\langle C \rangle$ to explicitly denote the binary encoding of C. Otherwise, we may abuse the same notation C for the encoding.

For convenience, we assume the following: (1) every binary string $u \in \{0,1\}^*$ represents some circuit C_u, which is done by assigning invalid encodings u to the trivial circuit $C_u(x) = 0$; (2) zero-padding is available, specifically, the minimum valid encoding u for any circuit C ends with 1, and we regard $u \circ 0^*$ is also valid encoding of the circuit C. For any $s \in \mathbb{N}$, let $e(s) = O(s \log s)$ be the size of minimum encoding for s-size circuits satisfying the above assumptions.

The following language $\mathsf{C\text{-}Eval}$ is known as a P-complete problem.

$$\mathsf{C\text{-}Eval} = \{\langle C, x \rangle : C \text{ is an } n\text{-input circuit and } x \in \{0,1\}^n \text{ satisfying } C(x) = 1\}.$$

In addition, the class P is equivalent to the class recognized by uniformly generated polynomial-size circuits (see the textbook by Arora and Barak, 2009). These results lead to the following facts.

Fact 1 (Universal circuits) *There exists a polynomial-size circuit family $UC = \{UC_N\}_{N \in \mathbb{N}}$ satisfying that for any $u, x \in \{0,1\}^*$,*

$$UC_N(\langle u, x \rangle) = \begin{cases} C_u(x) & (\textit{if } u \textit{ represents an } n\textit{-input circuit } C_u \textit{ and } |x| = n) \\ 0 & (\textit{otherwise}), \end{cases}$$

where $N = |\langle u, x \rangle|$. Moreover, UC is uniformly generated, that is, there exists a polynomial-time Turing machine UCM such that $\langle UC_N \rangle \leftarrow UCM(1^N)$ for any $N \in \mathbb{N}$.

Fact 2 (Circuit translator) *There exists a polynomial-time Turing machine CT such that for the input $\langle M, 1^n \rangle$ where M is a description of a $T(n)$-time Turing machine and $n \in \mathbb{N}$, CT outputs $\langle C \rangle \in \{0,1\}^{e(s(n))}$ where C is an n-input circuit of size $s(n) = O(T(n) \log T(n))$ satisfying $C(x) = M(x)$ for any $x \in \{0,1\}^n$. We call the above CT a circuit translator.*

In the second argument of CT, we may use the notation $1^{n_1 \times \cdots \times n_k}$ to denote the length of binary encoding for a k-tuple of binary strings $(x_1, \ldots, x_k)$ where $x_i \in \{0,1\}^{n_i}$ for each $i \in [k]$.

We also define a size-bounded universal circuit

$$UC^{s(n)} = \{UC_n^{s(n)} : \{0,1\}^{e(s(n))} \times \{0,1\}^n \to \{0,1\}\}_{n \in \mathbb{N}}$$

by

$$UC_n^{s(n)}(u,x) = \begin{cases} UC(\langle u, x \rangle) & (\text{if } u = \langle C \rangle \text{ and } C \in \mathsf{SIZE}[s(n)]) \\ 0 & (\text{otherwise}). \end{cases}$$

Note that, by the standard construction of universal circuits (Arora and Barak, 2009),

$$(\text{the size of } UC_n^{s(n)}) = \tilde{O}(s(n) + (s(n) - n)^2).$$

In addition, $\mathsf{SIZE}[s(n)]$ is evaluated by $UC^{s(n)}$ as

$$\mathsf{SIZE}[s(n)]_n = \left\{ C(x) := UC_n^{s(n)}(u,x) : u \in \{0,1\}^{e(s(n))} \right\} \text{ for each } n \in \mathbb{N}.$$

Therefore, we fix $UC^{s(n)}$ for the evaluation function for $\mathsf{SIZE}[s(n)]$.

A.1.2. OTHER AUXILIARY-INPUT PRIMITIVES

We introduce other auxiliary-input primitives whose existence is equivalent to the existence of AIOWFs.

Definition 17 (Auxiliary-input pseudorandom generators: AIPRGs) *An AIF $\{G_z : \{0,1\}^{n(|z|)} \to \{0,1\}^{\ell(n(|z|))}\}_{z \in \{0,1\}^*}$ is an auxiliary-input pseudorandom generator (AIPRG) if $\ell(n) > n$ and for any polynomial-time probabilistic algorithm A, there exists a non-empty set $Z_A \subseteq \{0,1\}^*$ such that for every $z \in Z_A$,*

$$\left| \Pr[A(z, G_z(U_n)) = 1] - \Pr[A(z, U_{\ell(n)}) = 1] \right| < \mathsf{negl}(|z|).$$

Definition 18 (Auxiliary-input pseudorandom functions: AIPRFs) *An AIF $\{F_z : \{0,1\}^{s(|z|)} \times \{0,1\}^{n(|z|)} \to \{0,1\}\}_{z \in \{0,1\}^*}$ is an auxiliary-input pseudorandom function (AIPRF) if for any polynomial-time probabilistic oracle machine $A^?$, there exists a non-empty set $Z_A \subseteq \{0,1\}^*$ such that for every $z \in Z_A$,*

$$\left| \Pr_{A, u \leftarrow U_s}[A^{f_z(u,\cdot)}(z) = 1] - \Pr_{A, \phi_{|z|}}[A^{\phi_{|z|}(\cdot)}(z) = 1] \right| < \mathsf{negl}(|z|),$$

where $\phi_{|z|} : \{0,1\}^{n(|z|)} \to \{0,1\}$ denotes a truly random function.

Fact 3 (Goldreich et al. 1986; Håstad et al. 1999) *The existence of AIOWFs, AIPRGs, and AIPRFs is equivalent.*

A.2. Padding Argument

Lemma 19 *For any $c_1, c_2 > 1$, $\mathsf{SIZE}[n^{c_1}]$ is weakly PAC learnable with a TSC of size $e(n^{c_1})^{c_2}$ iff $\mathsf{SIZE}[n^2]$ is weakly PAC learnable with TSC of size $e(n^2)^2$.*

Proof We consider only the case where $c_1, c_2 > 2$. In the other case, we can show the lemma in the same way.

Assume that $\mathsf{SIZE}[n^2]$ is weakly PAC learnable with $m(n)$ samples and a TSC of size $e(n^2)^2$ by the learner L. It is enough to construct a PAC learner L' for $\mathsf{SIZE}[n^{c_1}]$ with a TSC of size $e(n^{c_1})^{c_2}$ by using L. Let n be the size of example for L', and $r(\leq e(n^{c_1})^{c_2})$ be the seed length of the given TSC C.

Firstly, L' calculates the minimum $n' \in \mathbb{N}$ satisfying that

$$n^{c_1} \leq n'^2 - n' \text{ and } e(n^{c_1})^{c_2} + 2 \leq e(n'^2)^2 - e(n'^2).$$

It is easily checked that $e(n^{c_1}) \leq e(n'^2)$ and n' is polynomially related to n.

Secondly, L' constructs a new TSC $C' : \{0,1\}^r \to \{0,1\}^{e((n')^2)}$ from C by adding $e(n'^2) - e(n^{c_1})$ output-gates and two constant gates (i.e., 0-gate and 1-gate). Then L' plugs the output-gates of C and two constant gates into the output-gates of C' to satisfy that for any seed $s \in \{0,1\}^r$, C' samples the circuit $C(r)$ with additional dummy $n' - n$ input-gate (by using the constant gates). Note that C' samples circuits of size at most $n^{c_1} + (n' - n) \leq n'^2$. Therefore, we have that for any $x \in \{0,1\}^n$ and $s \in \{0,1\}^r$,

$$UC_n^{n^{c_1}}(C(s), x) = UC_{n'}^{n^2}(C'(s), x \circ 0^{n'-n}).$$

Notice that the size of C' becomes at most $e(n'^2)^2$, which is shown as follows:

$$\begin{aligned}(\text{size of } C') &= (\text{size of } C) + (e(n'^2) - e(n^{c_1})) + 2 \\ &< e(n^{c_1})^{c_2} + e(n'^2) + 2 < e(n'^2)^2.\end{aligned}$$

Thirdly, the learner L' executes L w.r.t. n' instead of n. Although L' needs $m(n')$ examples to execute L for n', the sample size is still polynomial in n because n' is polynomially related to n. In the execution, L' selects C' as a TSC for L, and feeds $m(n')$ samples and a challenge of L' to L, where L' stretches the size of examples and the challenge from n to n' by zero-padding.

Finally, if L returns a prediction for the challenge, then L' also outputs the same prediction.

Then we have that for any distribution D on $\{0,1\}^n$,

$$\begin{aligned}&\Pr[L' \text{ succeeds in learning the target } C(U_r) \text{ under the distribution } D] \\ &\quad = \Pr[L \text{ succeeds in learning the target } C'(U_r) \text{ under the distribution } D \circ 0^{n'-n}] \\ &\quad \geq 1/2 + \gamma(n'),\end{aligned}$$

where γ is a non-negligible advantage of L.

Therefore, L' succeeds in learning $\mathsf{SIZE}[n^{c_1}]$ with any TSC of size $e(n^{c_1})^{c_2}$. ■

A.3. Proof of Theorem 5

First, we show the hardness of learning from the existence of auxiliary-input cryptographic primitives, specifically, AIPRFs. The proof is essentially the same as the observations by Valiant (1984); Applebaum et al. (2008). The difference is only the part of explicitly constructing the sampling circuit for PRFs.

Claim 1 *If there exists an AIPRF* $\{F_z : \{0,1\}^{s(|z|)} \times \{0,1\}^{n(|z|)} \to \{0,1\}\}_{z\in\{0,1\}^*}$*, then there exists* $c > 1$ *such that* $\mathsf{SIZE}[n^c]$ *is not weakly PAC learnable with a TSC and membership query.*

Proof Let M_F be the probabilistic polynomial-time algorithm computing the AIPRF $\{F_z\}_{z\in\{0,1\}^*}$ in polynomial-time in $|z|$. Since $n(|z|)$ is polynomially related to $|z|$, M_F is computable in time $n(|z|)^c$ for some constant c.

First we show the case where a learner is not given access to membership query oracle. For contradiction, we assume that $\mathsf{SIZE}[n^{c+1}]$ is weakly PAC learnable with a TSC by the PAC learner L using $m := m(n)$ examples. Now we construct an adversary A for $\{F_z\}_{z\in\{0,1\}^*}$ from L.

Algorithm A

Input : $z \in \{0,1\}^*$ and oracle access to $f : \{0,1\}^{n(|z|)} \to \{0,1\}$

1 select $x_1, \ldots, x_m, x^* \leftarrow \{0,1\}^n$ and query the values of $f(x_1), \ldots, f(x_m), f(x^*)$
2 $C(\cdot,\cdot,\cdot) \leftarrow CT(M_F, 1^{|z|\times s(|z|)\times n(|z|)})$ and hardwire z into C as $C_z(\cdot,\cdot) := C(z,\cdot,\cdot)$
3 if the size of $C > n^{c+1}$, then **return**$(\perp)$
4 construct a circuit $C'_z : \{0,1\}^s \to \{0,1\}^{e(n^{c+1})}$ such that $C'_z(u)$ embeds the input u to C_z as $C_z(u,\cdot)$, and outputs $\langle C_z(u,\cdot)\rangle$ with zero-padding to the length $e(n^{c+1})$
5 execute $b \leftarrow L(C'_z, (x_1, f(x_1)), \ldots, (x_m, f(x_m)), x^*)$
6 if L succeeds in learning (that is, $b = f(x^*)$), then **return**(1), otherwise **return**(0)

The circuit translator in line 2 outputs a circuit C of size $O(n^c \log n)$. We assume that $|z|$ is large enough to satisfy that the size of C is smaller than n^{c+1}. Under this assumption, A must not output $\perp$ in line 3.

For any $z \in \{0,1\}^*$, $u \in \{0,1\}^{s(|z|)}$, and $x \in \{0,1\}^{n(|z|)}$, we have that

$$UC_n^{n^{c+1}}(C'_z(u), x) = UC_n^{n^{c+1}}(\langle C_z(u,\cdot)\rangle, x) = C_z(u,x) = M_F(z,u,x).$$

Therefore, the distribution on the function $f(x) := UC_n^{n^{c+1}}(C'_z(U_s), x)$ and the distribution on the function $f(x) := F(z, U_s, x)$ are statistically identical. Hence, the TSC C'_z represents a random choice of AIPRF F_z. Note that the size of C'_z is at most $e(n^{c+1})^2$ because the task of C'_z is only to replace input-gates of $\langle C_z\rangle$ with constant-gates and zero-pad the encoded circuit.

Let γ be the advantage of L. If the given f (as oracle access) is a pseudorandom function, L is executed in the valid setting where the target function is f and the example distribution is U_n. Since f is computable by the n^{c+1}-size circuit, f is contained in the concept class $\mathsf{SIZE}[n^{c+1}]$. In this case, L succeeds in learning with probability larger than $1/2 + \gamma(n)$.

On the other hand, if the given function is truly random, then L must not have any information about $f(x^*)$ unless $x^* \in \{x_1, \ldots, x_m\}$. In this case, L cannot predict $f(x^*)$ better than at random. Since m is the polynomial, the event that $x^* \in \{x_1, \ldots, x_m\}$ occurs only with negligible probability. Therefore, L succeeds in learning with probability at most $1/2 + \mathsf{negl}(n)$.

Since A outputs 1 iff L succeeds in learning, A breaks F_z for any sufficiently large $|z|$ with advantage $\gamma(n(|z|)) - \mathsf{negl}(n(|z|))$. Because $n(|z|)$ is polynomially-related to $|z|$, the advantage is non-negligible in $|z|$. This contradicts the assumption that $\{F_z\}_{z\in\{0,1\}^*}$ is an AIPRF.

Even if the learner L is allowed to access to membership query oracle, A can correctly answer the query by using its membership oracle. Therefore, the same argument holds. ■

By the above claim, Fact 3, and Lemma 19, if AIOWFs exist, then $\mathsf{SIZE}[n^2]$ is not PAC learnable with a TSC. For the other direction, we construct an AIPRG from the hardness of learning.

Claim 2 *If $\mathsf{SIZE}[n^2]$ is not weakly PAC learnable with a TSC, then there exists AIPRG.*

Proof Define polynomially-related functions m, n and s as follows:

$$m := m(|z|) = \max m \in \mathbb{N} : e(m^2) \cdot e(e(m^2)^2) + (e(m^2)+1) \cdot m \leq |z|$$

$$n := n(|z|) = e(m(|z|)^2), \quad s(n) = n + 1.$$

Then any binary string $z \in \{0,1\}^*$ is divided as

$$z = z_1^{(1)} \circ \cdots \circ z_{e(m^2)}^{(1)} \circ z_1^{(2)} \circ \cdots \circ z_{e(m^2)+1}^{(2)} \circ z_{last},$$

where $|z_i^{(1)}| = e(n^2) = e(e(m^2)^2)$ for each $i \in [e(m)]$, $|z_j^{(2)}| = m$ for each $j \in [e(m^2)+1]$, and

$$|z_{last}| = |z| - e(m^2)e(e(m^2)^2) - (e(m^2)+1)m.$$

We define a two-stage generator $G_z^{TS} : \{0,1\}^n \to \{0,1\}^{s(n)}$ as follows:

$$G_z^{TS}(x) = UC_m^{n^2}(y, z_1^{(2)}) \circ \cdots \circ UC_m^{n^2}(y, z_{e(m^2)+1}^{(2)}),$$

where $y := UC_n^{n^2}(z_1^{(1)}, x) \circ \cdots \circ UC_n^{n^2}(z_{e(m^2)}^{(1)}, x)$.

We show that $G^{TS} = \{G_z^{TS}\}_{z\in\{0,1\}^*}$ is indeed an AIPRG. By contradiction, assume that G^{TS} is not an AIPRG. Let A be the adversary for G^{TS}. W.l.o.g., we can assume that for any $z \in \{0,1\}^*$

$$\Pr_A[A(z, G_z^{TS}(U_{n(|z|)})) = 1] - \Pr_A[A(z, U_{s(n(|z|))}) = 1] \geq \gamma(m(|z|)),$$

where γ is a non-negligible function.

The argument proceeds as follows: first, we use the adversary A to construct an RRHS-refuter R with a TSC whose seed length is fixed. The restriction arises because the length of input to G^{TS} is fixed beforehand to n. Second, we construct a PAC learner for $\mathsf{SIZE}[n^2]$ with a restricted TSC by using Yao's next-bit generator (implicitly, Yao, 1982). Essentially, the second part corresponds to the previous work by Vadhan (2017). Finally we remove the restriction for a TSC by the simple padding argument.

First, we construct an RRHS-refuter R from the adversary A.

For any $n \in \mathbb{N}$, consider arbitrary circuit $C : \{0,1\}^{e(n^2)} \to \{0,1\}^{e(n^2)}$ of size $e(n^2)^2$, and $x_1, \ldots, x_{e(n^2)+1}$ in the input to R. The algorithm R constructs auxiliary-input z for G^{TS} to satisfy that $|z| = e(e(n^2)^2)e(n^2) + (e(n^2)+1)n$ in line 8. Therefore, in the execution in line 9, we have $m(|z|) = n$, $n(|z|) = e(n^2)$, and for any seed $s \in \{0,1\}^{n(|z|)}$ to C and index $i \in [e(n^2)+1]$,

$$UC_n^{n^2}(C(s), x_i) = UC_{m(|z|)}^{n^2}(UC_{n(|z|)}^{n^2}(z_1, s) \circ \cdots \circ UC_{n(|z|)}^{n^2}(z_{e(n^2)}, s), x_i) = G_{z,i}^{TS}(s).$$

Algorithm R

Input : an $e(n^2)^2$-size circuit $C : \{0,1\}^{e(n^2)} \to \{0,1\}^{e(n^2)}$, $x_1, \ldots, x_{e(n^2)+1} \in \{0,1\}^n$ and $b_1, \ldots, b_{e(n^2)+1} \in \{0,1\}$

encode $z_i = \langle C_i \rangle$ for each $i \in [e(n^2)]$

$z := z_1 \circ \cdots \circ z_{e(n^2)} \circ x_1 \circ \cdots \circ x_{e(n^2)+1}$

execute $A(z, b_1 \circ \cdots \circ b_{e(n^2)+1})$ and output the same value

Thus,

$$\Pr_{R, f \leftarrow C(U_{e(n^2)})}[R(C, x_1, \ldots, x_{e(n^2)+1}, f(x_1), \ldots, f(x_{e(n^2)+1})) = 1] = \Pr_A[A(z, G_z^{TS}(U_{n(|z|)})) = 1],$$

and

$$\Pr_R[R(C, x_1, \ldots, x_{e(n^2)+1}, U_{e(n^2)+1}) = 1] = \Pr_A[A(z, U_{s(n(|z|))}) = 1].$$

Since A breaks G^{TS} for any auxiliary-input z, we have that for any $x_1, \ldots, x_{e(n^2)+1} \in \{0,1\}^n$,

$$\begin{aligned}\Pr_{R, f \leftarrow C(U_{e(n^2)})}&[R(C, x_1, \ldots, x_{e(n^2)+1}, f(x_1), \ldots, f(x_{e(n^2)+1})) = 1] \\ &- \Pr_R[R(C, x_1, \ldots, x_{e(n^2)+1}, U_{e(n^2)+1}) = 1] \\ &= \Pr_A[A(z, G_z^{TS}(U_{n(|z|)})) = 1] - \Pr_A[A(z, U_{s(n(|z|))}) = 1] \geq \gamma(n).\end{aligned}$$

Now we construct Yao's next-bit generator L from R.

Algorithm L

Input : an $e(n^2)^2$-size circuit $C : \{0,1\}^{e(n^2)} \to \{0,1\}^{e(n^2)}$, $(x_1, b_1) \ldots, (x_{e(n^2)}, b_{e(n^2)})$, and x^* where $x_i, x^* \in \{0,1\}^n$, and $b_i \in \{0,1\}$ for each $i \in [e(n^2)]$

select $i \leftarrow [e(n^2)+1]$ and $c^*, c_i, \ldots, c_{e(n^2)} \leftarrow \{0,1\}^n$

execute $b \leftarrow R(C, x_1, \ldots, x_{i-1}, x^*, x_i, \ldots, x_{e(n^2)}, b_1, \ldots, b_{i-1}, c^*, c_i, \ldots, c_{e(n^2)})$

if $b = 1$ **then return**(c^*), **otherwise return**$(1 - c^*)$

By the hybrid argument (for the detail, see Arora and Barak, 2009, Theorem 9.11), we have

$$\Pr_{R, f \leftarrow C(U_{e(n^2)})}[L(C, (x_1, f(x_1)), \ldots, (x_{e(n^2)}, f(x_{e(n^2)}), x^*)) = f(x^*)] \geq 1/2 + \gamma(n)/e(n^2).$$

The advantage $\gamma(n)/e(n^2)$ is still non-negligible for n. Therefore, L is a valid weak learner for $\mathsf{SIZE}[n^2]$ with a TSC whose seed length is restricted to $s(n^2)$. Finally, we remove the restriction on the seed length by the following padding strategy, which is similar to the proof of Lemma 19.

Now we construct a PAC learner L' with a TSC by using L. Let n be the size of example, and $r(\leq e(n^2)^2)$ be the seed length of the given TSC C.

Firstly, L' calculates the minimum $n' \in \mathbb{N}$ satisfying $e(n'^2) \geq \sqrt{\max(r, e(n^2))^2 + 3} + 1$. It is easily checked that $e(n'^2) \geq \max(r, e(n^2))$ and n' is polynomially related to n.

Secondly, L' constructs a new TSC $C' : \{0,1\}^{e(n'^2)} \to \{0,1\}^{e(n'^2)}$ from C by adding $e(n'^2)-r$ dummy input-gates, $e(n'^2) - e(n^2)$ output-gates, and two constant-gates (i.e., 0-gate and 1-gate). Then L' plugs the output-gates of C and two constant-gates into the output-gates of C' to satisfy that on input $s \in \{0,1\}^{e(n'^2)}$, C' discards the suffix of s of length $e(n'^2) - r$ and samples the target $C(s)$ with additional dummy $n' - n$ input-gates (by using constant gates). In other words, we have that for any $x \in \{0,1\}^n$, $s \in \{0,1\}^r$, and $s' \in \{0,1\}^{e(n'^2)-r}$,

$$UC_n^{n^2}(C(s), x) = UC_{n'}^{n^2}(C'(s \circ s'), x \circ 0^{n'-n}).$$

As shown in later, note that the size of C' is at most $e(n'^2)^2$.

Thirdly, the learner L' executes L w.r.t. n' instead of n. Although L' needs $e(n'^2)$ examples to execute L for n', the sample size is still polynomial in n because n' is polynomially related to n. In the execution, L' selects C' as a TSC for L, and feeds $e(n'^2)$ samples and the challenge of L' to L, where L' stretches the size of examples and the challenge from n to n' by zero-padding.

Finally, if L' returns a prediction for the challenge, then L also outputs the same prediction.

If the size of C' is at most $e(n^2)^2$, then for any distribution D on $\{0,1\}^n$,

$$\begin{aligned}
&\Pr_{L'}[L' \text{ succeeds in learning the target } C(U_r) \text{ under the distribution } D] \\
&\quad = \Pr_{L}[L \text{ succeeds in learning the target } C'(U_{e(n')^2}) \text{ under the distribution } D \circ 0^{n'-n}] \\
&\quad \geq 1/2 + \gamma(n'),
\end{aligned}$$

where γ is a non-negligible function.

Thus, the remaining part of the proof is to bound the size of C'. Remember that

$$e(n'^2) - 1 \geq \sqrt{\max(r, e(n^2))^2 + 3}.$$

By the simple calculation, we have that

$$e(n'^2)^2 \geq 2e(n'^2) + e(n^2)^2 + 2.$$

By the condition in Definition 4, we can assume that the size of C is at most $e(n^2)^2$. Therefore,

$$\begin{aligned}
(\text{size of } C') &= (\text{size of } C) + (e(n'^2) - r) + (e(n'^2) - e(n^2)) + 2 \\
&< e(n^2)^2 + 2e(n'^2) + 2 \leq e(n'^2)^2.
\end{aligned}$$

■

By Claim 2 and Fact 3, if $\mathsf{SIZE}[n^2]$ is not weakly PAC learnable with a TSC, then there exists AIOWF. Therefore, Theorem 5 holds.

Corollary 6 also immediately holds by Fact 3 and Claims 1 and 2 .

Appendix B. PAC Learnability and AILHSGs

We follow the setting in Appendix A.1. Let $E = \{E_n\}_{n\in\mathbb{N}}$ be a family of function $E_n : \{0,1\}^{s(n)} \times \{0,1\}^n \to \{0,1\}$. We say that a class $\mathscr{C} = \{\mathscr{C}_n\}_{n\in\mathbb{N}}$ is evaluated by E if $\mathscr{C}_n = \{f(x) := E_n(u,x) : u \in \{0,1\}^{s(n)}\}$ for any $n \in \mathbb{N}$. We also define the dual-class $\mathscr{C}^* = \{\mathscr{C}_n^*\}_{n\in\mathbb{N}}$ by $\mathscr{C}_n^* = \{g(u) := E_n(u,x) : x \in \{0,1\}^n\}$.

B.1. Padding Argument

Lemma 20 *For any polynomial $s(n) > n$, $\mathsf{SIZE}[s(n)]$ is PAC learnable iff $\mathsf{SIZE}[n^2]$ is PAC learnable.*

Proof First we assume that $s(n) \geq n^2$. Then it is enough to construct a PAC learner L_s for $\mathsf{SIZE}[s(n)]$ from another PAC learner L for $\mathsf{SIZE}[n^2]$. Assume that L needs $m(n)$ samples for learning, where m is a polynomial.

Let n be the size of examples for L_s. Firstly, L_s calculates the minimum value $n' \in \mathbb{N}$ satisfying $s(n) - n \leq n'^2 - n'$. Obviously, n' is polynomially related to n. Secondly, L_s takes $m(n')$ examples and executes L for n', where L_s pads each example and the challenge $x \in \{0,1\}^n$ with 0 as $x \circ 0^{n'-n} \in \{0,1\}^{n'}$. Finally, if L' returns a prediction for the challenge, then L also outputs the same prediction.

Let C be a circuit of size $s(n)$ which computes the target function for L_s and D be a target distribution on $\{0,1\}^n$. Then it is easily checked that L_s executes L with a valid setting where the target function is computed by C (discarding a suffix of length $n' - n$) and the target distribution $D \circ 0^{n'-n}$. Although the target circuit for L needs additional $n' - n$ input-gates (and they increase the size of circuits in our setting), the size is bounded above by $s(n) + n' - n \leq n'^2$. Therefore, L succeeds in learning C with non-negligible advantage, and so does L_s.

Even in the case where $n^2 > s(n) > n$, we also show the lemma in the same way. ■

B.2. PAC Learning and RRHS-Refutation

We formally introduce the notion of RRHS-refutation, first introduced by Vadhan (2017).

Definition 21 (RRHS-refutation) *Let $\mathscr{C}$ be a concept class. We say a polynomial-time probabilistic algorithm R, called a refuter, RRHS-refutes the dual-class $\mathscr{C}^*$ with $m := m(n)$ samples if for any $n \in \mathbb{N}$, $f \in \mathscr{C}_n$, and $x_1, \ldots, x_m \in \{0,1\}^n$, R satisfies that*

1. *(soundness)* $\Pr_R[R(x_1, \ldots, x_m, f(x_1), \ldots, f(x_m)) = 0] \geq 2/3$;
2. *(completeness)* $\Pr_{b \leftarrow \{0,1\}^m}[\Pr_R[R(x_1, \ldots, x_m, b) = 1] \geq 2/3] \geq 1/2$.

We say that $\mathscr{C}^*$ is RRHS-refutable if there exist a polynomial m and a refuter for $\mathscr{C}^*$ with m samples.

Strictly speaking, we adopt another formulation about completeness from the original paper by Vadhan (2017). This is because our formulation gives clearer correspondence to HSGs. The original completeness was given as follows.

2'. (completeness) $\Pr_{b \leftarrow \{0,1\}^m, R}[R(x_1, \ldots, x_m, b) = 1] \geq 2/3$.

The equivalence between two formulations holds by Markov's inequality as follows.

Proof ($2 \Rightarrow 2'$) Assume that there exists a refuter R which satisfies the conditions 1 and 2. As in the usual error reduction, if we repeat the execution of R independently and take the majority

vote, then error probability is reduced to any constant. Therefore, we can assume that for any $x_1, \ldots, x_m \in \{0,1\}^n$,

$$\Pr_{b \leftarrow \{0,1\}^m}\left[\Pr_R[R(x_1, \ldots, x_m, b) = 1] \geq 5/6\right] \geq 1/2.$$

This implies that

$$\Pr_{b \leftarrow \{0,1\}^m, R}[R(x_1, \ldots, x_m, b) = 1] \geq 5/6 \cdot 1/2 = 5/12.$$

On the other hand, by the soundness, we have that for any $f \in \mathscr{C}_n$,

$$\Pr_R[R(x_1, \ldots, x_m, f(x_1), \ldots, f(x_m)) = 1] \leq 1/3.$$

Therefore, if we collect further (but polynomial-size) samples, repeat the execution of R by using distinct samples, and estimate the probability that R outputs 1, then we can distinguish the above two cases with arbitrary small constant error probability, specifically, $1/3$. Thus, there exists a refuter which satisfies the conditions 1 and 2'.

($2 \Leftarrow 2$') Assume that there exists a refuter R which satisfies the conditions 1 and 2'. As seen in the above, we can reduce the error probability to arbitrary constant by taking further (but polynomial-size) samples. Therefore, w.l.o.g., we can assume that for any $X = (x_1, \ldots, x_m)$ where $x_i \in \{0,1\}^n$,

$$\Pr_{b \leftarrow \{0,1\}^m, R}[R(X, b) = 0] \leq 1/6$$

By applying Markov's inequality for the nonnegative random variable $Y := \Pr_R[R(X,b) = 0]$ w.r.t. the random choice of $b \in \{0,1\}^m$, we have the condition 2 as follows:

$$\Pr_b\left[\Pr_R[R(X,b) = 0] \geq 1/3\right] \leq 3 \cdot \Pr_{b \leftarrow \{0,1\}^m, R}[R(X,b) = 0] \leq 3/6 = 1/2.$$

■

Now we introduce two important facts about RRHS-refutation. The first fact is the equivalence between PAC learning and RRHS-refutation.

Fact 4 (PAC learning $\Leftrightarrow$ RRHS-refutation Vadhan 2017) *For any (polynomially represented) concept class $\mathscr{C}$, $\mathscr{C}$ is PAC learnable iff $\mathscr{C}^*$ is RRHS-refutable.*

The second fact was implicitly used in the previous work.

Fact 5 (Pitt and Warmuth 1990; Vadhan 2017) *Let $\mathscr{C}$ and $\mathscr{C}'$ be concept classes evaluated by $\{E_n : \{0,1\}^{s(n)} \times \{0,1\}^n \to \{0,1\}\}_{n \in \mathbb{N}}$ and $\{E'_n : \{0,1\}^{t(n)} \times \{0,1\}^n \to \{0,1\}\}_{n \in \mathbb{N}}$, respectively. Assume that for any $n \in \mathbb{N}$, there exist $n' \in \mathbb{N}$, and polynomial-time computable functions $f_{rep} : \{0,1\}^n \to \{0,1\}^{t(n')}$ and $f_{ex} : \{0,1\}^{s(n)} \to \{0,1\}^{n'}$ satisfying that*

$$E'_{n'}(f_{rep}(x), f_{ex}(u)) = E_n(u, x).$$

Then the following holds:

- *if $\mathscr{C}'$ is RRHS-refutable, then $\mathscr{C}^*$ is also RRHS-refutable;*
- *if $\mathscr{C}'^*$ is RRHS-refutable, then $\mathscr{C}$ is also RRHS-refutable.*

Now we apply Fact 5 to polynomial-size circuits. For any polynomial s, we use the notation $s^\dagger$ to denote the polynomial such that the size of UC^s_n is at most $s^\dagger(e(s(n)))$ for any $n \in \mathbb{N}$. Then we have the following lemma.

Lemma 22 *For any polynomial $s(n)$,*

- $\mathsf{SIZE}[s(n)]^*$ *is RRHS-refutable if* $\mathsf{SIZE}[s^\dagger(n)]$ *is RRHS-refutable;*
- $\mathsf{SIZE}[s(n)]$ *is RRHS-refutable if* $\mathsf{SIZE}[s^\dagger(n)]^*$ *is RRHS-refutable.*

Proof We give functions $f_{ex} : \{0,1\}^{e(s(n))} \to \{0,1\}^{e(s(n))}$ and $f_{rep} : \{0,1\}^n \to \{0,1\}^{e(s^\dagger(e(s(n))))}$ in Fact 5 by

$$f_{ex}(u) = u, \text{ and } f_{rep}(x) = \langle C_x \rangle, \text{ where } C_x(u) = UC^{s(n)}_n(u, x).$$

Note that the input size of C_x is $e(s(n))$ and the size is at most $s^\dagger(e(s(n)))$ by the definition of $s^\dagger$. Since $UC^{s(n)}$ is generated in polynomial-time, f_{ex} and f_{rep} are polynomial-time computable. In addition, these functions satisfy the conditions in Fact 5 as follows:

$$UC^{s^\dagger(n)}_{e(s(n))}(f_{rep}(x), f_{ex}(u)) = UC^{s^\dagger(n)}_{e(s(n))}(\langle C_x \rangle, u) = C_x(u) = UC^{s(n)}_n(u, x).$$

■

B.3. Proof of Theorems 10 and 11

First we introduce candidates for the complete AILHSG in Theorem 11, which is simply composed of concatenated universal circuits.

Definition 23 *For any polynomial $s(n)$ and $\ell(n)$, we define an AIF $UC^{s,\ell} = \{UC^{s,\ell}_z : \{0,1\}^{n(|z|)} \to \{0,1\}^{\ell(n(|z|))}\}_{z \in \{0,1\}^*}$ where $n(|z|) = \max n \in \mathbb{N} : e(s(n)) \cdot \ell(n) \leq |z|$ by*

$$UC^{s,\ell}_z(x) = UC^s(z_1, x) \circ \cdots \circ UC^s(z_{\ell(n)}, x),$$

where $z_i \in \{0,1\}^{e(s(n))}$ for each $i \in [\ell(n)]$, $z = z_1 \circ \cdots \circ z_{\ell(n)} \circ z_{left}$, and $z_{left} \in \{0,1\}^{|z| - e(s(n)) \cdot \ell(n)}$.

Theorem 24 is our main theorem in this section. Theorem 10 is identical to the equivalence between 1 and 2 in the following theorem.

Theorem 24 *The followings are equivalent:*

1. $\mathsf{SIZE}[n^2]$ *is not PAC learnable.*
2. *There exists an AILHSG.*
3. *There exists a polynomial $s(n)$ such that $\{UC^{s(n),n^\tau}\}_{\tau \in \mathbb{N}}$ is an AILHSG.*

Proof (1 ⇒ 2) For any $\tau \in \mathbb{N}$, we construct an AIHSG $G^\tau = \{G^\tau_z\}_{z\in\mathbb{N}}$ for each $\tau \in \mathbb{N}$. Define functions m and n by

$$m := m(|z|) = \max m \in \mathbb{N} : e(m^2)^\tau \cdot m \le |z|, \text{ and } n := n(|z|) = e(m^2).$$

Then m and n are polynomially related to $|z|$ and any binary string $z \in \{0,1\}^*$ is divided as

$$z = z_1 \circ \cdots \circ z_{e(m^2)^\tau} \circ z_{last},$$

where $|z_i| = m$ for each $i \in [e(m^2)^\tau]$ and $|z_{last}| = |z| - e(m^2)^\tau \cdot m$.

Now we define $G^\tau_z : \{0,1\}^n \to \{0,1\}^{n^\tau}$ by

$$G^\tau_z(x) = UC^{n^2}_m(x, z_1) \circ \ldots UC^{n^2}_m(x, z_{e(m^2)^\tau}).$$

The above generator satisfies the conditions in Definition 8 because we can calculate the i-th bit of $G^\tau_z(x)$ in polynomial-time in n by executing $UC^{n^2}_m(x, z_i)$, and the location of z_i in z is determined by only $\tau, |z|$ and i.

For contradiction, assume that there exists τ such that G^τ is not an AIHSG, and let A be the adversary. By Fact 4, it is enough to construct a refuter for $\mathsf{SIZE}[n^2]^*$ from A.

Now we define the refuter R for $\mathsf{SIZE}[n^2]^*$ with $e(n^2)^\tau$ samples as follows:

$$R(x_1, \ldots, x_{e(n^2)^\tau}, b_1, \ldots, b_{e(n^2)^\tau}) := A(x_1 \circ \cdots \circ x_{e(n^2)^\tau}, b_1 \circ \cdots \circ b_{e(n^2)^\tau}).$$

The completeness is easily checked. For the soundness, assume that there exists $y \in \{0,1\}^{e(n^2)}$ such that $UC^{n^2}_n(y, x_i) = b_i$ for any $i \in [e(n^2)^\tau]$. Let $z = x_1 \circ \cdots \circ x_{e(n^2)^\tau}$. Then $m(|z|) = n$ and $n(|z|) = e(n^2)$, and we have

$$G^\tau_z(y) = UC^{n^2}_n(y, x_1) \circ \cdots \circ UC^{n^2}_n(y, x_{e(n^2)^\tau}) = b_1 \circ \cdots \circ b_{e(n^2)^\tau}.$$

Since $b_1 \circ \cdots \circ b_{e(n^2)^\tau}$ is contained in the image of G^τ_z, the adversary A outputs 0 with probability greater than $2/3$. In this case, R also outputs 0 and satisfies the soundness.

(2 ⇒ 3) Let $\{G^\tau\}_{\tau\in\mathbb{N}}$ be an AILHSG and $localG^?$ be the polynomial-time oracle machine in the condition of Definition 9. Since the length of input $(\tau, |z|, i, x)$ is at most $\log\tau + \log|z| + \tau\log(n(|z|)) + n(|z|) \le \mathsf{poly}(n(|z|))$, $localG$ halts in polynomial-time in $n := n(|z|)$.

Assume that $|z|$ is large enough to satisfy that $localG^?$ queries $q(n)$ points where q is a polynomial. Since the queries are non-adaptive and determined by only $\tau, |z|, i$, we can divide $localG^?$ into two Turing machines Pos and M satisfying that for any (sufficiently long) string z,

$$Pos(\tau, |z|, i) = (j_1, \ldots, j_{q(n)});$$
$$M(z_{j_1}, \ldots, z_{j_{q(n)}}, \tau, |z|, i, x) = localG^z(\tau, |z|, i, x) = G^\tau_{z,i}(x),$$

where Pos and M halt in $t(n)$-time where t is a polynomial.

Now we show that $\{UC^{t(n)^2, n^\tau}\}_{\tau\in\mathbb{N}}$ is an AILHSG. The conditions about locality are easily checked as in the above proof of (1 ⇒ 2). For contradiction, we assume that there exists τ such that $UC^{t(n)^2,n^\tau}$ is not an AIHSG by the adversary A, and construct the adversary A' which breaks G^τ by using A.

Algorithm A'

Input : $z \in \{0,1\}^*$ and $y \in \{0,1\}^{n(|z|)^\tau}$

13 execute $C(,,,,) \leftarrow CT(M, 1^{q(n)\times \log \tau \times \log |z| \times \tau \log(n(|z|)) \times n})$

14 **for** $i := 1$ *to* n^τ **do**

15 execute $(j_1^i, \ldots, j_{q(n)}^i) \leftarrow Pos(\tau, |z|, i)$

16 hardwire the indices as $C_i(x) = C(z_{j_1^i} \circ \ldots \circ z_{j_{q(n)}^i}, \tau, |z|, i, x)$, and $u_i = \langle C_i(x) \rangle$

17 truncate/zero-pad u_i to the length $e(t(n)^2)$

18 **end**

19 output the same value to $A(u_1 \circ \cdots \circ u_{n^\tau}, y)$

We analyze the adversary A'. For sufficiently large $|z|$, the size of C in line 13 is at most $t(n(|z|))^2$. In this case, each $C_i(x)$ is properly encoded to the length $e(t(n)^2)$ in line 17.

For any auxiliary-input z to G^τ and $i \in [n^\tau]$, let u_i be the binary encoding given in line 17 and $u_z := u_1 \circ \cdots \circ u_{n^\tau}$ be auxiliary-input in line 19. Then we have that

$$\Pr_y \left[\Pr_{A'}[A'(z,y) = 1] \geq 2/3 \right] = \Pr_y \left[\Pr_A[A(u_z, y) = 1] \geq 2/3 \right] \geq 1/2.$$

If there exists $x \in \{0,1\}^n$ satisfying that $G_z^\tau(x) = y$, then

$$\begin{aligned} y = G_z^\tau(x) &= M(z_{j_1^1}, \ldots, z_{j_{q(n)}^1}, \tau, |z|, 1, x) \circ \cdots \circ M(z_{j_1^{n^\tau}}, \ldots, z_{j_{q(n)}^{n^\tau}}, \tau, |z|, n^\tau, x) \\ &= C_1(x) \circ \cdots \circ C_{n^\tau}(x) \\ &= UC_n^{t(n)^2}(u_1, x) \circ \cdots \circ UC_n^{t(n)^2}(u_{n^\tau}, x) \\ &= UC_{u_z}^{t(n)^2, n^\tau}(x). \end{aligned}$$

Thus, y is also contained in the image of $UC_{u_z}^{t(n)^2,n^\tau}$. Therefore, the adversary A outputs 0 with probability greater than $2/3$, and so does A'. By the above observations, A' breaks G^τ, which contradicts the assumption that $\{G^\tau\}_{\tau \in \mathbb{N}}$ is an AILHSG.

(3 $\Rightarrow$ 1) For a polynomial $s(n)$, assume that $\{UC^{s(n),n^\tau}\}_{\tau \in \mathbb{N}}$ is an AILHSG. We show that $\mathsf{SIZE}[s(n)]$ is not RRHS-refutable.

For contradiction, let R be the refuter for $\mathsf{SIZE}[s(n)]$ with n^c samples. Now we construct an adversary A for $UC^{s(n),n^c}$ by

$$A(z,y) = R(z_1, \ldots, z_{n^c}, y_1, \ldots, y_{n^c})$$

where $z = z_1 \circ \cdots \circ z_{n^c} \circ z_{last}$, $|z_i| = e(n(|z|)^2)$, and $|z_{last}| = |z| - n^c \cdot e(n^2)$.

For any z, if y is randomly selected, then by the soundness of R,

$$\Pr_y \left[\Pr_A[A(z,y) = 1] \geq 2/3 \right] = \Pr_y \left[\Pr_R[R(z_1, \ldots, z_{n^c}, y_1, \ldots, y_{n^c}) = 1] \geq 2/3 \right] \geq 1/2.$$

On the other hand, if there exists $x \in \{0,1\}^n$ such that $UC_z^{s(n),n^c}(x) = y$, then for each $i \in [n^c]$, $y_i = UC(z_i, x)$. By the completeness of R, A outputs 0 with probability greater than $2/3$. Thus, A breaks $UC^{s(n),n^c}$.

Now we have that $\mathsf{SIZE}[s(n)]$ is not RRHS-refutable. By taking a polynomial $t(n)(> n)$ satisfying $s(n) \leq t^\dagger(n)$, we conclude that

$$\begin{aligned}
&\mathsf{SIZE}[s(n)] \text{ is not RRHS-refutable} \\
&\quad \Longrightarrow \mathsf{SIZE}[t^\dagger(n)] \text{ is not RRHS-refutable} && (\because \mathsf{SIZE}[s(n)] \subseteq \mathsf{SIZE}[t^\dagger(n)]) \\
&\quad \Longrightarrow \mathsf{SIZE}[t(n)]^* \text{ is not RRHS-refutable} && (\because \text{Lemma 22}) \\
&\quad \Longrightarrow \mathsf{SIZE}[t(n)] \text{ is not PAC learnable.} && (\because \text{Fact 4})
\end{aligned}$$

By Lemma 20, $\mathsf{SIZE}[n^2]$ is not PAC learnable. ■

To show Theorem 11, the implication 2 ⇒ 3 in Theorem 24 is not sufficient because the function $s(n)$ in the statement 3 depends on the size of AILHSGs in the statement 2. However, this problem is easily resolved by applying Theorem 24 twice. The existence of AILHSG implies hardness of PAC learning $\mathsf{SIZE}[n^2]$ (2 ⇒ 1), and the hardness implies the existence of specific generator $\{UC^{s(n),n^\tau}\}_{\tau\in\mathbb{N}}$ (1⇒ 3). In this case, the function $s(n)$ is determined depending on the size of concept class, that is n^2, thus we can fix $s(n)$ beforehand.

Appendix C. Meta-PAC Learning is as Hard as PAC Learning

In this section, we consider the non-uniform model as a computational model. First, we formally define RRHS-refutation in the non-uniform model as follows.

Definition 25 (RRHS-refutation: non-uniform) *Let $\mathscr{C}$ be a concept class. We say a family of polynomial-size circuits $\{R_n\}_{n\in\mathbb{N}}$, called a refuter, RRHS-refutes the dual-class $\mathscr{C}^*$ with $m := m(n)$ samples if for any $n \in \mathbb{N}$, $f \in \mathscr{C}_n$, and $x_1, \ldots, x_m \in \{0,1\}^n$, R_n satisfies that*

1. *(soundness) $R_n(x_1, \ldots, x_m, f(x_1), \ldots, f(x_m)) = 0$;*
2. *(completeness) $\Pr_{b\leftarrow\{0,1\}^m}[R_n(u_1, \ldots, u_m, b) = 1] \geq 1/2$.*

We say that $\mathscr{C}^*$ is non-uniformly RRHS-refutable if there exist a polynomial m and a polynomial-size refuter for $\mathscr{C}^*$ with m samples.

Then the following equivalence between learning and refutation also holds in the non-uniform model. It is easily shown by combining the original proof for the uniform model with Adleman's trick (Adleman, 1978) to remove randomness from a refuter.

Theorem 26 (PAC learning ⇔ RRHS-refutation: non-uniform) *For any (polynomially represented) concept class $\mathscr{C}$, $\mathscr{C}$ is non-uniformly PAC learnable iff $\mathscr{C}^*$ is non-uniformly RRHS-refutable.*

C.1. Proof of Theorem 13

This theorem follows from Theorem 26 and a simple padding argument. As in the statement, let $\mathscr{C}$ be a concept class evaluated by a circuit family $\{C_n\}_{n\in\mathbb{N}}$, where $C_n : \{0,1\}^{s(n)} \times \{0,1\}^n \to \{0,1\}$ is a circuit of size at most $\tau(n)$.

Assume that there exists c_0 such that $C_n \in \mathsf{Learnable}_{\tau,n^c}$ for any $c \geq c_0$ and n. Then for any $n \in \mathbb{N}$, there exists a circuit $L_n \in \mathsf{SIZE}[n^2]$ satisfying completeness and soundness in Definition 12

with $m = n^{c_0}$ samples. Notice that the family $\{L_n\}_{n\in\mathbb{N}}$ is a RRHS-refuter for $\mathscr{C}^*$ with m examples. Therefore, by Theorem 26, $\mathscr{C}$ is non-uniformly PAC learnable.

For the opposite direction, assume that $\mathscr{C}$ is non-uniformly PAC learnable. By Theorem 26, $\mathscr{C}^*$ is also non-uniformly RRHS-refutable. Let $\{R_n\}_{n\in\mathbb{N}}$ be the polynomial-size refuter for $\mathscr{C}^*$ with m samples. Note that, if each R_n is contained in $\mathsf{SIZE}[n^2]$, then $C_n \in \mathsf{Learnable}_{\tau,m}$. In general, however, we can only assure that each R_n is contained in $\mathsf{SIZE}[n^a]$ for some $a \in \mathbb{N}$. We resolve this problem by the simple padding argument.

Assume that $a > 2$, otherwise, $R_n \in \mathsf{SIZE}[n^2]$. For the size of example n and the sample size m, the length of input to the refuter is $nm + m = (n+1)m$. Therefore, the size of R_n is at most $N := ((n+1)m)^a$. Since m and N are polynomials in n, we can select $c_0 \in \mathbb{N}$ satisfying that for any $c \geq c_0$, $m \leq n^c$ and $N \leq ((n+1)n^c)^2 - (n+1)(n^c - m)$.

For any $c \geq c_0$, let R'_n be a refuter which takes n^c samples, executes R_n by using only m samples, and discards remaining $n^c - m$ samples. Since $\{R_n\}_{n\in\mathbb{N}}$ RRHS-refutes $\mathscr{C}^*$, $\{R'_n\}_{n\in\mathbb{N}}$ also RRHS-refutes $\mathscr{C}^*$. In addition, R'_n can be constructed from R_n and $(n+1)(n^c - m)$ additional input-gates, thus the size of R' is bounded above by $N + (n+1)(n^c - m) \leq ((n+1)n^c)^2$ and $R'_n \in \mathsf{SIZE}[n^2]$. This implies that $C_n \in \mathsf{Learnable}_{\tau,n^c}$ for any $c \geq c_0$.

C.2. Proof of Theorem 14

Fix polynomials τ and m arbitrary. The theorem follows from the definition of $\mathsf{Learnable}_{\tau,m}$ and the result by Lautemann (1983).

Let C be a circuit of size N computing an evaluation $E : \{0,1\}^{s(n)} \times \{0,1\}^n \to \{0,1\}$. If $N > \tau(n)$, then $C \notin \mathsf{Learnable}_{\tau,m}$. This trivial case is easily detected in polynomial-time. Therefore, we can assume that $N \leq \tau(n)$.

If $C \in \mathsf{Learnable}_{\tau,m}$, then there must exist a learner $L \in \mathsf{SIZE}[n^2]$ with m samples. We consider the description $\langle L \rangle$ as a witness and verify the following conditions in Π^p_2: for any examples $x_1, \ldots, x_m \in \{0,1\}^n$,

(1) for any $u \in \{0,1\}^{s(n)}$, $L(x_1, \ldots, x_m, E(u, x_1), \ldots, E(u, x_m)) = 0$;

(2) $\Pr_{b_1,\ldots,b_m}[L(x_1, \ldots, x_m, b_1, \ldots, b_m) = 1] \geq 1/2$.

The condition (1) is obviously checked in Π^p_1. For the condition (2), if we fix a learner L and examples $x_1, \ldots, x_m$, then it is easily checked in BPP by estimating the accepting probability of L. Since $\mathsf{BPP} \subseteq \Sigma^p_2 \cap \Pi^p_2$ (Lautemann, 1983), we can replace the probabilistic part with Π^p_2. This concludes that $\mathsf{Learnable}_{\tau,m} \in \Sigma^p_3$.

In the above argument, if we can replace the probabilistic part with a deterministic polynomial-time verifier instead of Π^p_2, then we can also verify the condition (2) in Π^p_1. Therefore, $\mathsf{P} = \mathsf{BPP}$ implies $\mathsf{Learnable}_{\tau,m} \in \Sigma^p_2$.

C.3. Proof of Theorem 16

In fact, we can show stronger statements than Theorem 16 as follows:

Lemma 27 *Let $\tau(n)$ be a polynomial satisfying $\tau(n) > n$. If $\mathsf{SIZE}[n^2]$ is non-uniformly PAC learnable, then $\mathsf{Learnable}_{\tau,n^c} \in \mathsf{P}$ for sufficiently large $c \in \mathbb{N}$.*

Proof Assume that $\mathsf{SIZE}[n^2]$ is non-uniformly PAC learnable. By Theorem 20 (for the non-uniform model) and Theorem 26, $\mathsf{SIZE}[\tau(n)]^*$ is non-uniformly RRHS-refutable. Let R be the refuter with n^{c_0} samples for some $c_0 \in \mathbb{N}$. By the same padding argument as the proof of Theorem 13, we can also assume that $R \in \mathsf{SIZE}[n^2]$.

For a given circuit $C : \{0,1\}^{s(n)} \times \{0,1\}^n \to \{0,1\}$, if the size of C is at most $\tau(n)$, then any target function is computed by a circuit of the size $\tau(n)$. Therefore, R satisfies the third condition in Definition 12. Thus, $C \in \mathsf{Learnable}_{\tau,n^c}$ holds for any $c \geq c_0$.

On the other hand, if the size of C is larger than $\tau(n)$, then C must not satisfy the second condition for $\mathsf{Learnable}_{\tau(n),n^c}$ for any c. In other words, for any $c \geq c_0$, $C \in \mathsf{Learnable}_{\tau(n),n^c}$ iff the size of C is at most $\tau(n)$. The latter condition is easily checked in polynomial-time from the description of C. Therefore, $\mathsf{Learnable}_{\tau,n^c} \in \mathsf{P}$ for any $c \geq c_0$. ■

Lemma 28 *Let $\tau(n)$ be a polynomial satisfying $\tau(n) \geq n^{1+\epsilon}$ for some $\epsilon > 0$. If $\mathsf{SIZE}[n^2]$ is not non-uniformly PAC learnable, then $\mathsf{Learnable}_{\tau,n^c} \notin \mathsf{P}$ for any c.*

Proof Let $s(n) = n + \sqrt{n}$ and $t(n) = (\text{the size of } UC_n^{s(n)}) = \tilde{O}(n + \sqrt{n} + n) = \tilde{O}(n)$. Assume that n is sufficiently large to satisfy that $t(n) + n + 1 \leq n^{1+\epsilon} \leq \tau(n)$.

Assume that $\mathsf{SIZE}[n^2]$ is not non-uniformly PAC learnable. By Theorem 20 (for the non-uniform model), $\mathsf{SIZE}[s(n)]$ is also not non-uniformly PAC learnable. By Definition 12, if $u \in \mathsf{Learnable}_{\tau,n^c}$, then $u \in \mathsf{Learnable}_{\tau,n^{c+1}}$ for any $c \in \mathbb{N}$ and $u \in \{0,1\}^*$. Therefore, by Theorem 13, we have that

$$UC_n^{s(n)} \notin \mathsf{Learnable}_{\tau,n^c} \text{ for any } c \in \mathbb{N}.$$

For contradiction, assume that there exists $c \in \mathbb{N}$ such that $\mathsf{Learnable}_{\tau,n^c} \in \mathsf{P}$ and M be the polynomial-time algorithm for $\mathsf{Learnable}_{\tau,n^c}$.

Now we construct an algorithm A solving the circuit-SAT problem as follows: on input $\langle C \rangle \in \{0,1\}^n$ (assume that the size of C is at most n and the input length is $\ell \leq n$), (1) A constructs another circuit $C' : \{0,1\}^{\ell+e(s(n))} \times \{0,1\}^n \to \{0,1\}$ as $C'(u \circ u', x) = C(u) \wedge UC_n^{s(n)}(u', x)$ where $u \in \{0,1\}^\ell$, $u' \in \{0,1\}^{e(s(n))}$, and $x \in \{0,1\}^n$, and (2) A outputs the value of $\neg M(C')$.

Note that the size of C' is at most $n + t(n) + 1 \leq \tau(n)$. If the given C is satisfiable, then there exists an assignment $u^* \in \{0,1\}^\ell$ such that $C(u^*) = 1$. By applying the partial assignment u^*, C' boils down to $UC_n^{s(n)}$. Remember that $UC_n^{s(n)} \notin \mathsf{Learnable}_{\tau,n^c}$. Therefore, $C' \notin \mathsf{Learnable}_{\tau,n^c}$, and $A(C)$ outputs $\neg M(C') = 1$ for any satisfiable circuit C.

On the other hand, if C is not satisfiable, then $C'(u \circ u', x) \equiv (0 \wedge UC_n^{n^2}(u', x)) \equiv 0$. Since we can easily RRHS-refute the dual-class $\{0\}^*$ (with 2 samples by calculating negation of OR of two labels), $C' \in \mathsf{Learnable}_{\tau,n^c}$. Therefore, $A(C)$ outputs $\neg M(C') = 0$ for any unsatisfiable circuit C.

Therefore, A correctly solves the circuit-SAT problem, and $\mathsf{P} = \mathsf{NP}$. This implies $\mathsf{SIZE}[n^2]$ is (even uniformly) PAC learnable (Blumer et al., 1987). This contradicts the assumption. ■

Lemmas 27 and 28 immediately derive Theorem 16 (by setting $\tau(n) = n^2$) and the following theorem. Theorem 29 shows that the specific setting of $\tau(n)$ in Definition 15 does not affect the asymptotic complexity of our meta-PAC learning problem.

Theorem 29 *Let $\tau(n)$ be a polynomial satisfying $\tau(n) \geq n^{1+\epsilon}$ for some $\epsilon > 0$. Meta-PAC learning is hard iff $\mathsf{Learnable}_{\tau,n^c} \notin \mathsf{P}$ for infinitely many $c \in \mathbb{N}$.*

Appendix D. Learning Hierarchy between BPP vs. NP and One-Way Functions

Based on our results in Section 2, we introduce "Learning Hierarchy (LH)", which divides the task of constructing one-way functions based on the worst-case assumption that $\mathsf{NP} \not\subseteq \mathsf{BPP}$ into several stages from the perspective of weak polynomial learnability for polynomial-size circuits. This notion brings the learning theoretic approach to investigate the gap between complexity theory and cryptography, that is, Pessiland named by Impagliazzo (1995).

LH begins with the hardness of NP-complete problem, that is, the circuit-SAT problem. First we introduce two constraint satisfaction problems (CSPs) based on the circuit-SAT problem.

Definition 30 (CircSAT and RRHS-CircSAT) *Let $s(n), m(n) > n$ be polynomials. We define the language $(s(n), m(n))$-CircSAT as follows:*

$$(s(n), m(n))\text{-CircSAT} = \{\big((C_1, b_1), \ldots, (C_{m(n)}, b_{m(n)})\big) \mid n \in \mathbb{N}, C_i \in \mathsf{SIZE}[s(n)]_n, b_i \in \{0,1\}, \text{ and } \exists x \in \{0,1\}^n \text{ s.t. } C_i(x) = b_i \text{ for any } i \in [m(n)]\}.$$

We also define the problem $(s(n), m(n))$-RRHS-CircSAT as follows. We say that a probabilistic polynomial-time algorithm A solves $(s(n), m(n))$-RRHS-CircSAT if A satisfies the following properties: for any $n \in \mathbb{N}$, $x_1, \ldots, x_{m(n)} \in \{0,1\}^n$, and n-input circuit $C \in \mathsf{SIZE}[s(n)]$,

1. *(soundness)* $\Pr_A[A((x_1, C(x_1)), \ldots, (x_{m(n)}, C(x_{m(n)}))) = 0] \geq 2/3$;
2. *(completeness)* $\Pr_{b \leftarrow \{0,1\}^{m(n)}}[\Pr_A[A((x_1, b_1), \ldots, (x_{m(n)}, b_{m(n)})) = 1] \geq 2/3] \geq 1/2$.

We say that $(s(n), m(n))$-RRHS-CircSAT is hard if any probabilistic polynomial-time algorithm does not solve $(s(n), m(n))$-RRHS-CircSAT.

Since the circuit-SAT problem is NP-complete, it is not hard to see that $(s(n), m(n))$-CircSAT is also NP-complete for any $s(n), m(n)(> n)$. We can also regard $(s(n), m(n))$-RRHS-CircSAT as the CSP given by weakening the requirements of $(s(n), m(n))$-CircSAT. Thus, if $(s(n), m(n))$-RRHS-CircSAT is hard, then $(s(n), m(n))$-CircSAT is also not solvable in BPP.

It is easily checked that the hardness of RRHS-CircSAT corresponds to the proposition that $UC^{s(n),m(n)}$ in Definition 23 is an AIHSG.

Lemma 31 $(s(n), m(n))$-RRHS-CircSAT *is hard iff* $UC^{s(n),m(n)}$ *is an AIHSG.*

Now we state the hierarchies (LH0)–(LH3) by strengthening the worst-case assumption step by step:

(LH0) $\mathsf{NP} \not\subseteq \mathsf{BPP}$, that is, for any polynomials $s(n), m(n)(> n)$, $(s(n), m(n))$-CircSAT$\notin \mathsf{BPP}$;

(LH1) there exist $s(n)$ and $m(n)$ such that $(s(n), m(n))$-RRHS-CircSAT is hard;

(LH2) for any $s(n)$, there exists $m(n)$ such that $(s(n), m(n))$-RRHS-CircSAT is hard;

(LH3) there exists $m(n)$ such that for any $s(n)$, $(s(n), m(n))$-RRHS-CircSAT is hard.

Obviously, (LH3) $\Rightarrow$ (LH2) $\Rightarrow$ (LH1) $\Rightarrow$ (LH0). By Lemma 31 and the proof (2$\Leftrightarrow$ 3) of Theorem 24, (LH1)–(LH3) are also characterized by auxiliary-input primitives in Definitions 8 and 9:

(LH1) there exists an AIHSG;

(LH2) for any polynomial $\ell(n)$, there exists an AIHSG with stretch $\ell(n)$;

(LH3) there exists an AILHSG.

By Theorem 10, (LH3) corresponds to the hardness of PAC learning.

(LH3) PAC learning (polynomial-size circuits) is hard;

Now we strengthen the hardness assumption by weakening requirements of learning. We have two directions for relaxations: (1) target functions and (2) example distributions. First, we consider the relaxation of requirements on target functions:

(LH3-T1) PAC learning with a TSC is hard, where the learner is given black-box query access to the TSC (instead of the description of the TSC);

(LH3-T2) PAC learning with a TSC is hard;

(LH3-T3) there exists polynomially samplable distribution D on target functions such that PAC learning under D is hard.

Obviously, (LH3-T3) $\Rightarrow$ (LH3-T2) $\Rightarrow$ (LH3-T1) $\Rightarrow$ (LH3). Note that Theorem 5 shows that (LH3-T2) corresponds to the existence of AIOWFs.

(LH3-T2) An auxiliary-input one-way function exists.

Second, we consider the other relaxation of requirements on example distributions from (LH3):

(LH3-E1) PAC learning under any polynomially samplable example distribution is hard;

(LH3-E2) PAC learning with an example sampling circuit is hard, where the learner is given the description of the circuit sampling examples (as in Definition 4);

(LH3-E3) there exists a polynomially samplable distribution D such that PAC learning under the example distribution D is hard.

Obviously, (LH3-E3) $\Rightarrow$ (LH3-E2) $\Rightarrow$ (LH3-E1) $\Rightarrow$ (LH3).

We can combine the above conditions on target functions with the conditions on example distributions such as (LH3-TiEj) where $i, j \in \{0, \ldots, 3\}$. Finally, the learning model in (LH3-T3E3) $=$ (LH4) corresponds to the learning model considered by Blum et al. (1994).

(LH4) There exist polynomially samplable distributions D_{tar}, D_{ex} such that PAC learning under the target sampling distribution D_{tar} and the example distribution D_{ex} is hard.

Blum et al. (1994) showed that the above (LH4) corresponds to the existence of OWFs.

(LH4) A one-way function exists.

To the best of our knowledge, all opposite (non-trivial) directions are open at present. However, to get cryptographic primitives (LH4) from the worst-case assumption (LH0), we must collapse all hierarchies in LH regardless of the proof techniques. Therefore, we require further investigation of each stage and each gap between them as a crucial open problem in theoretical computer science.